Arm® Cortex®-A73 MPCore Processor

Revision: r1p0

Technical Reference Manual

Arm® Cortex®-A73 MPCore Processor

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-01	23 June 2015	Confidential	First release for r0p0
0001-02	27 October 2015	Confidential	First release for r0p1
0002-03	10 March 2016	Confidential	First release for r0p2
0002-04	03 June 2016	Non-Confidential	Second release for r0p2
0002-05	10 August 2016	Non-Confidential	Third release for r0p2
0100-06	21 June 2018	Non-Confidential	First release for r1p0

Non-Confidential Proprietary Notice

Contents

Arm® Cortex®-A73 MPCore Processor Technical Reference Manual

Preface

This preface introduces the *Arm® Cortex®-A73 MPCore Processor Technical Reference Manual*.

It contains the following:
- *About this book* on page 9.
- *Feedback* on page 13.

About this book

This document gives reference documentation for the Cortex-A73 processor. It contains programming details for registers and describes the memory system, caches, debug trace, and interrupts.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r1p2, where:

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

Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the Cortex®-A73 processor.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction
This chapter introduces the Cortex-A73 processor and its features.

Chapter 2 Functional Description
This chapter describes the functionality of the Cortex-A73 processor.

Chapter 3 Programmers Model
This chapter describes the registers and provides information for programming the Cortex-A73 processor.

Chapter 4 System Control
Describes the system registers, their structure, operation, and how to use them.

Chapter 5 Memory Management Unit
This chapter describes the *Memory Management Unit* (MMU).

Chapter 6 Level 1 Memory System
This chapter describes the *Level 1* (L1) memory system.

Chapter 7 Level 2 Memory System
This chapter describes the *Level 2* (L2) memory system.

Chapter 8 Generic Interrupt Controller CPU Interface
This chapter describes the Cortex-A73 processor implementation of the Arm *Generic Interrupt Controller* (GIC) CPU interface.

Chapter 9 Generic Timer
This chapter describes the Generic Timer for the Cortex-A73 processor.

Chapter 10 Debug
This chapter describes the Cortex-A73 processor debug registers and shows examples of how to use them.

Chapter 11 Performance Monitor Unit
This chapter describes the *Performance Monitor Unit* (PMU) and the registers that it uses.

Chapter 12 Embedded Trace Macrocell
This chapter describes the *Embedded Trace Macrocell* (ETM) for the Cortex-A73 processor.

Chapter 13 Cross Trigger
This chapter describes the cross trigger interface for the Cortex-A73 processor.

Chapter 14 Advanced SIMD and Floating-point Support

This chapter describes the Advanced SIMD and floating-point features and registers in the Cortex-A73 processor.

Appendix A Signal Descriptions

This appendix describes the Cortex-A73 processor signals.

Appendix B Revisions

This appendix describes the technical changes between released issues of this document.

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.

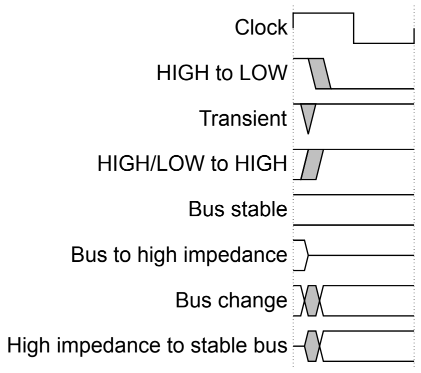

Figure 1 Key to timing diagram conventions

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 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® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite* (IHI 0022).
- *Arm® AMBA® APB Protocol Specification* (IHI 0024).
- *Arm® AMBA® 4 ATB Protocol Specification ATBv1.0 and ATBv1.1* (IHI 0032).
- *Arm® AMBA® 4 AXI4-Stream Protocol Specification* (IHI 0051).
- *Arm® Low Power Interface Specification* (IHI 0068).
- *Arm® CoreSight™ DAP-Lite Technical Reference Manual* (DDI 0316).
- *Arm® CoreSight™ SoC-400 Technical Reference Manual* (DDI 0480).
- *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* (IHI 0064).
- *Arm® Generic Interrupt Controller Architecture Specification* (IHI 0069).

The following confidential books are only available to licensees:

- *Arm® Cortex®-A73 MPCore Processor Cryptographic Extension Technical Reference Manual* (100049).
- *Arm® Cortex®-A73 MPCore Processor Configuration and Sign-off Guide* (100050).
- *Arm® Cortex®-A73 MPCore Processor Integration Manual* (100051).
- *Arm® Cortex®-A73 MPCore Processor Release Note*.

Other publications

This section lists relevant documents published by third parties:

- *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-A73 MPCore Processor Technical Reference Manual*.
- The number 100048_0100_06_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.

Chapter 1
Introduction

This chapter introduces the Cortex-A73 processor and its features.

It contains the following sections:

1.1 About the Cortex-A73 processor

The Cortex-A73 processor is a high-performance, low-power, Arm macrocell that implements the Armv8-A architecture.

It contains up to four cores, each with a *Level 1* (L1) memory system and one shared *Level 2* (L2) cache. The Cortex-A73 processor includes a superscalar, variable-length, out-of-order pipeline. Each core can handle a sustained maximum throughput of two instructions per cycle.

The following figure shows an example of a configuration with four cores and an ACE interface.

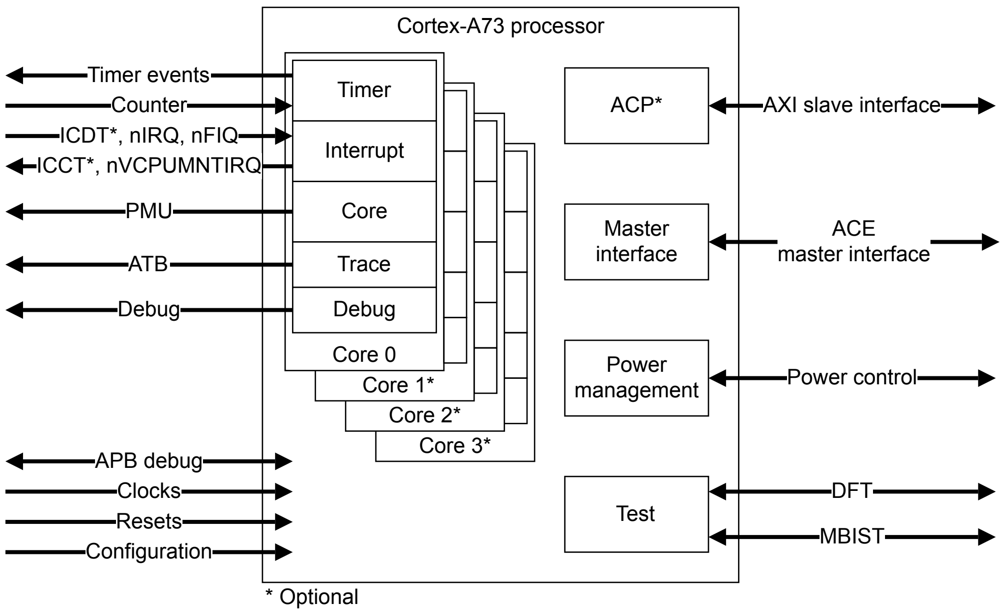

Figure 1-1 Example Cortex-A73 processor configuration

See *2.1 About the Cortex-A73 processor functions* on page 2-26 for more information about the functional components.

1.2 Compliance

The Cortex-A73 processor complies with, or implements, the specifications described in this section.

This TRM complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

This section contains the following subsections:

1.2.1 Arm® architecture

The Cortex-A73 processor implements the Armv8-A architecture with Advanced SIMD support and the optional Armv8 Cryptographic Extension.

The Armv8-A architecture that is implemented in the Cortex-A73 processor includes:

- Support for both AArch32 and AArch64 Execution states.
- Support for all Exception levels, EL0, EL1, EL2, and EL3, in each execution state.
- Armv8 Debug architecture.
- GICv4 interface.
- AMBA 4 ACE bus architecture.
- ETMv4 architecture (instruction trace only).
- The A32 instruction set, previously called the Arm instruction set.
- The T32 instruction set, previously called the Thumb instruction set.
- The A64 instruction set.

The Cortex-A73 processor supports the following features:

- Floating-point and Advanced SIMD support for integer and floating-point vector operations.

  Note

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

- Optional Armv8 Cryptographic Extension.

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

1.2.2 Interconnect architecture

The Cortex-A73 bus interface natively supports the AMBA 4 ACE bus architecture.

See the *Arm®AMBA® AXI* and *ACE Protocol Specification AXI3*, *AXI4*, and *AXI4-Lite*, *ACE and ACE-Lite*.

1.2.3 Generic Interrupt Controller architecture

The Cortex-A73 processor implements the *Generic Interrupt Controller* (GIC) v4 architecture.

The Cortex-A73 processor includes only the GIC CPU Interface. See the *Arm® Generic Interrupt Controller Architecture Specification, GICv4*.

1.2.4 Generic Timer architecture

The Cortex-A73 processor implements the Arm Generic Timer architecture with a 64-bit counter for each core.

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

1.2.5 Debug architecture

The Cortex-A73 processor implements the Armv8 Debug architecture.

The CoreSight™ *Cross Trigger Interface* (CTI) enables the debug logic, the *Embedded Trace Macrocell* (ETMv4), and the *Performance Monitor Unit* (PMU), to interact with each other and with other CoreSight components.

For more information, see the following documents:
- *Arm® CoreSight™ Architecture Specification*.
- *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

1.2.6 Embedded Trace Macrocell architecture

The Cortex-A73 processor implements the ETMv4 architecture for Program Trace only.

See the *Arm® ETM Architecture Specification, ETMv4*.

1.3 Features

The Cortex-A73 processor includes the following features:

- Full implementation of the Armv8-A architecture instruction set with the architecture options listed in *1.2.1 Arm® architecture* on page 1-16.
- AArch64 and AArch32 support at all Exception levels (EL0 to EL3).
- Superscalar, variable-length, out-of-order pipeline with symmetrical dual-issue of most instructions.
- *Level 2* (L2) memory system providing cluster memory coherency and an L2 cache.
- An integrated Data Engine unit that executes floating-point and Advanced SIMD instruction sets. The Data Engine provides full IEEE 754-2008 support for SIMD pipelines.
- An optional Armv8 Cryptographic accelerator engine for AES, SHA1, and SHA2-256.

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

1.4 Interfaces

The Cortex-A73 processor has the following external interfaces:

- Memory interface that implements an ACE interface.
- Optional *Accelerator Coherency Port* (ACP) that implements an AXI slave interface.
- Debug interface that implements an AMBA APB slave interface.
- Trace interface that implements an AMBA ATB interface.
- *Cross Trigger Interface* (CTI).
- *Design for Test* (DFT).
- *Memory Built-In Self Test* (MBIST).
- Q-channel, for power management.

See *2.2 Interfaces* on page 2-31 for more information on each of these interfaces.

1.5 Main Implementation options

The following table lists the main implementation options at build time for the Cortex-A73 processor.

Table 1-1 Cortex-A73 processor implementation options

Feature	Range of options
Number of cores	1-4
L1 data cache size	• 32KB • 64KB
L2 cache size	• 256KB • 512KB • 1024KB • 2048KB • 4096KB • 8192KB
L2 ECC support	Can be included or not included.
L2 Tag RAM setup latency	0-7
L2 Tag RAM read latency	0-7
L2 Tag RAM write latency	0-7
L2 Data RAM setup latency	0-7
L2 Data RAM read latency	0-7
L2 Data RAM write latency	0-7
Cryptographic Extension	Can be included or not included.
Accelerator Coherency Port (ACP)	Can be included or not included.

Note

- All the cores share a common L2 cache, and each core has the same configuration for all parameters.
- If the design is configured for one core, the ACE system level coherency support is retained.

This section contains the following subsection:

- *1.5.1 Processor configuration* on page 1-20.

1.5.1 Processor configuration

All cores in a cluster have identical configurations, that were determined during the build configuration. These configurations cannot be changed by software.

1.6 Test features

The Cortex-A73 processor provides test signals that enable the use of both DFT and MBIST interface signals to test the processor and its memory arrays.

See *A.1 About the signal descriptions* on page Appx-A-586 for more information.

1.7 Product documentation and design flow

This section outlines the documentation available for the Cortex-A73 processor and when to use it in the design flow.

See *1.7.1 Documentation* on page 1-22 for a list of publications. For information on the relevant architectural standards and protocols, see *1.2 Compliance* on page 1-16.

1.7.1 Documentation

There are three specific documents describing the Cortex-A73 processor: Technical Reference Manual, Configuration and Sign-off Guide, and Integration Manual.

The Cortex-A73 processor documentation is as follows:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the Cortex-A73 processor. This information is required at all stages of the design flow. However, the choices that are made during the design flow can mean that some behavior described in the TRM is not relevant. If you are programming the Cortex-A73 processor, then contact:

- The implementer to determine:
  - The build configuration of the implementation.
  - What integration, if any, was performed before implementing the Cortex-A73 processor.
- The integrator to determine the pin configuration of the device that you are using.

There is a separate TRM for the optional Cryptographic Extension.

Configuration and Sign-off Guide

The *Configuration and Sign-off Guide* (CSG) describes:

- The available build configuration options and related issues in selecting them.
- How to configure the *Register Transfer Level* (RTL) source files with the build configuration options.
- How to integrate RAM arrays.
- How to run test vectors.
- The processes to sign off the configured design.

The Arm product deliverables include reference scripts and information about using them to implement your design. Reference methodology flows supplied by Arm are example reference implementations. Contact your EDA vendor for EDA tool support.

The CSG is a confidential document that is only available to licensees.

Integration Manual

The *Integration Manual* (IM) describes how to integrate the Cortex-A73 processor into a SoC. It includes a description of the pins that the integrator must tie off to configure the processor. Some of the integration is affected by the configuration options that are used when implementing the Cortex-A73 processor.

The IM is a confidential document that is only available to licensees.

1.7.2 Design flow

The Cortex-A73 processor is delivered as synthesizable RTL. Before it can be used in a product, it must go through the process described in this section.

The process is divided into three stages:

Implementation

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

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

Programming
This is the last process. The system programmer develops the software required to configure and initialize the Cortex-A73 processor, and tests the required application software.

Each process:

- Can be performed by a different person or team.
- Can include implementation and integration choices that affect the behavior and features of the Cortex-A73 processor.

The operation of the final device depends on:

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 Cortex-A73 processor by tying inputs to specific values. These configurations 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 Cortex-A73 processor by programming particular values into registers. This affects the behavior of the processor.

——— **Note** ———

This manual refers to implementation-defined features that apply to build configuration options. Reference to a feature that is included means that the appropriate build and pin configuration options have been selected. Reference to an enabled feature means that the feature has also been configured by software.

1.8 Product revisions

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

r0p0

First release.

r0p1

Q-channel interface now supports *Wait for Event* (WFE).

r0p2

Fixes most of the errata from r0p1.

r1p0

Includes Inter-Exception level isolation of branch predictor structures so that an Exception level cannot train branch prediction for a different Exception level to reliably hit in these trained prediction entries.

Adds a new field CSV3 saying that data loaded under control flow speculation with a permission or domain fault, if used as an address in a speculative load, cannot cause cache allocation. Previous revisions of Cortex-A73 were also immune to this issue but did not have the field CSV3 to indicate it.

Chapter 2
Functional Description

This chapter describes the functionality of the Cortex-A73 processor.

It contains the following sections:

2.1 About the Cortex-A73 processor functions

The following figure shows a top-level functional diagram of the Cortex-A73 processor.

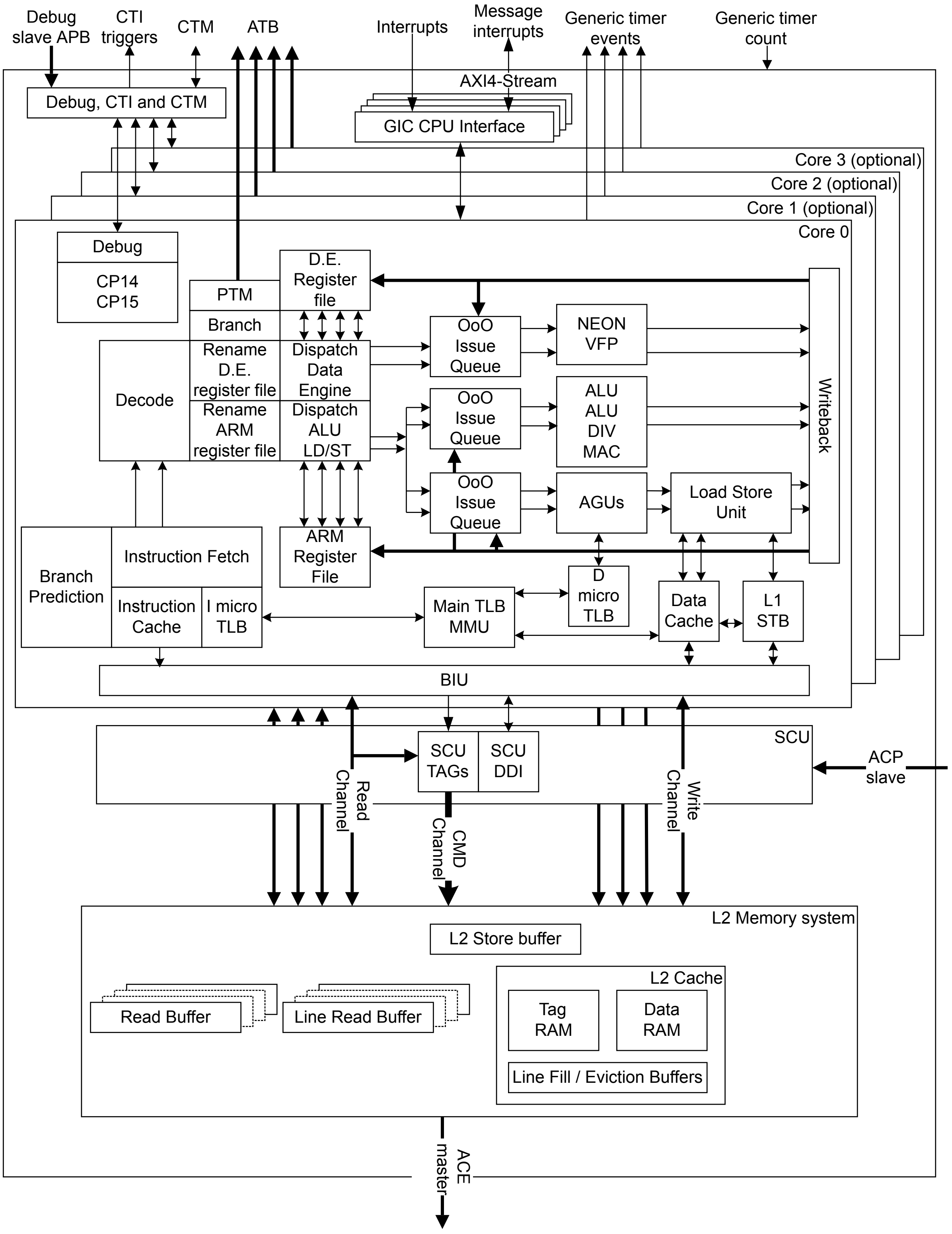

Figure 2-1 Cortex-A73 processor block diagram

This section contains the following subsections:
- *2.1.1 Instruction side memory system* on page 2-28.
- *2.1.2 Integer core* on page 2-28.

2.1.1 Instruction side memory system

The *Instruction side* (I-side) fetches instructions from the *Level 2* (L2) cache, stores them in a 4-way *Level 1* (L1) cache, and feeds up to two instructions at the same time to a core.

The Cortex-A73 I-side performs static and dynamic branch prediction. Dynamic prediction uses a two-level global history buffer and a branch target address cache. A nested return stack speeds up function returns.

The instruction cache is indexed by virtual address, and the tag is matched with the physical address. The physical address is generated by the Instruction micro TLB.

The I-side can issue up to four 64-byte wide requests to L2 memory. The preload hint instruction, PLI, is supported and only performs L2 cache preloading.

The instruction cache is part of the L1 Memory system.

See *Chapter 6 Level 1 Memory System* on page 6-341 for more information.

The I-side performs the following dynamic branch predictions, using *Branch Target Address Cache* (BTAC). It predicts:
- Whether the branch is likely to be taken, or not.
- The target of the branch, if it is taken.
- The target state (A32/T32).
- The type of the branch that is taken.

2.1.2 Integer core

The integer core holds most of the program-visible state of the processor, such as general-purpose registers and system registers.

The integer core provides configuration and control of the memory system and its associated functionality. It decodes and executes instructions, operating on data held in the registers in accordance with the Armv8-A architecture. Instructions are fed to the integer core from the *Instruction side memory system* (I-side). The integer core executes instructions that require data to be transferred to or from the memory system by interfacing to the *Data side memory system* (D-side), that manages all load and store operations.

See *Chapter 3 Programmers Model* on page 3-54 and *Chapter 4 System Control* on page 4-65 for more information.

2.1.3 Data side memory system

The Data side memory system (D-side) handles load and store requests from the Load Store Unit (LSU). It uses memory attributes from the MMU to determine the type of access required, and whether permission to access a memory region is granted.

Data side memory system (D-side) includes:
- A prefetcher, which detects regular patterns of loads or stores, and prefetches cache lines.
- A *Store Buffer* (STB), which holds committed store operations that have not yet been allocated in the L1 data cache, or that have been directly sent to the L2 when a stream of full cache line writes that misses in the L1 cache is detected. The STB can merge several store operations into a single transaction if they are sent to the same 64-bit aligned address.

- A *Bus Interface Unit* (BIU), which can send up to 8 concurrent cacheable linefill requests to the L2 memory system.
- A *Virtually Indexed Physically Tagged* (VIPT) Level-1 (L1) data cache, which behaves as either an eight-way set associative cache (for 32KB configurations) or a 16-way set associative PIPT cache (for 64KB configurations).
- A 48-entry fully-associative L1 Data TLBs.
- An L1 data cache auto-prefetching buffer supporting up to eight streams.

2.1.4 Memory Management Unit

The processor includes a *Memory Management Unit* (MMU) to enable virtual addresses to be translated to physical addresses.

The MMU uses a set of translation tables that map a virtual address to a physical address. The translation table entries also hold memory attributes and access permissions. The MMU uses the attributes and permissions to allow or prevent the processor from accessing memory regions. The MMU autonomously accesses memory to read the translation tables. This is referred to as a translation table walk.

The results of a translation table walk are cached in Translation Lookaside Buffers (TLBs). The Cortex-A73 processor has three TLBs:

- A main, unified, TLB that holds translations for both instruction and data accesses (L2 TLB).
- An instruction micro TLB that holds translations for instruction accesses (L1 instruction TLB).
- A data micro TLB that holds translations for data accesses (L1 data TLB).

Both micro TLBs are implemented to give single-cycle access to translation results.

For more information, see *Chapter 5 Memory Management Unit* on page 5-334.

2.1.5 Advanced SIMD and Floating-point support

Advanced SIMD and floating-point technology are media and signal processing architectures that add instructions targeted at audio, video, 3-D graphics, image, and speech processing. Advanced SIMD instructions are available in AArch64 and AArch32 states.

The floating-point architecture includes the floating-point register file and status registers. It performs floating-point operations on the data held in the floating-point register file.

See *Chapter 14 Advanced SIMD and Floating-point Support* on page 14-562 for more information.

2.1.6 Cryptographic Extension

The optional Cortex-A73 MPCore Cryptographic Extension supports the Armv8 Cryptographic Extensions.

The Cryptographic Extension adds new A64, A32, and T32 instructions to Advanced SIMD that accelerate *Advanced Encryption Standard* (AES) and other encryption and decryption algorithms.

See the *Arm® Cortex®-A73 MPCore Processor Cryptographic Extension Technical Reference Manual* for more information.

2.1.7 Level 2 memory system

The Cortex-A73 *Level 2* (L2) memory system contains the L2 cache pipeline and all logic that is required to maintain memory coherence between the cores of the cluster.

The Cortex-A73 L2 memory system has the following features:

- An L2 cache that:
  - Has a cache RAM size of 256KB, 512KB, 1MB, 2MB, 4MB or 8MB.
  - Is 16-way set associative with optional tag and data *Error Correction Code* (ECC) protection per 64 bits.
  - Supports 64-byte cache lines.
- A 512-bit wide fetch path from the L2 cache.

- Two 128-bit slave interfaces per core for instruction fetches and data fetches.
- A single 128-bit wide master interface to external memory that:
  - Implements the AMBA 4 ACE architecture.
  - Supports a 40-bit physical address range.
- An optional 128-bit wide I/O coherent ACP interface that can allocate to the L2 cache.

An integrated *Snoop Control Unit* (SCU) connects the cores to the L2 memory system. The SCU maintains data cache coherency between the cores and between the ACP port and cores, as well as arbitrates L2 requests from the cores.

When the Cortex-A73 processor is implemented with a single core, it still includes the SCU.

Note

The SCU supports hardware management of coherency of the instruction cache requests. Instruction cache linefills perform coherent reads. There is no coherency management of data held in the instruction cache.

See *Chapter 7 Level 2 Memory System* on page 7-363 for more information.

2.1.8 Performance monitoring

The Cortex-A73 processor provides performance counters and event monitors that can be configured to gather statistics on the operation of the processor and the memory system.

See *Chapter 11 Performance Monitor Unit* on page 11-442 for more information.

2.1.9 Debug and trace

The Cortex-A73 processor provides extensive support for real-time debug and performance profiling.

The Cortex-A73 processor supports a range of debug and trace features including:

- Arm-v8 debug features in each core.
- ETMv4 instruction trace unit for Program Trace only, for each core.
- CoreSight *Cross Trigger Interface* (CTI).
- CoreSight *Cross Trigger Matrix* (CTM).
- Debug ROM.

The Cortex-A73 processor has an *Advanced Peripheral Bus version 3* (APBv3) debug interface that is CoreSight compliant. This permits system access to debug resources, for example, the setting of watchpoints and breakpoints.

The Cortex-A73 processor provides performance monitors that can be configured to gather statistics on the operation of each core and the memory system. The performance monitors implement the Arm PMUv3 architecture.

See *Chapter 10 Debug* on page 10-391, *Chapter 11 Performance Monitor Unit* on page 11-442, and *Chapter 12 Embedded Trace Macrocell* on page 12-481 for more information.

2.2 Interfaces

This section describes the external interfaces of the Cortex-A73 processor.

This section contains the following subsections:

2.2.1 Master memory interface

The Cortex-A73 processor implements the AMBA 4 ACE interface.

ACE is an extension to the AXI protocol and provides the following enhancements:

- Support for hardware cache coherency.
- Barrier transactions that guarantee transaction ordering.
- Distributed virtual memory messaging, enabling management of a virtual memory system.

  See the *Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite* for more information.

2.2.2 Accelerator Coherency Port

The processor supports an *Accelerator Coherency Port* (ACP). The ACP reduces software cache maintenance operations when sharing memory regions with other masters, and allows other masters to allocate data into the L2 cache.

The ACP is an AMBA 4 slave interface.

The ACP slave interface allows an external master to make coherent requests to shared memory, but it does not support cache maintenance, coherency, barrier, or DVM transactions.

See *7.6 ACP* on page 7-372 and the *Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite* for more information.

2.2.3 External debug interface

The Cortex-A73 processor supports an AMBA 3 APB v1.0 interface that enables access to the debug registers.

The Armv8-A debug includes watchpoint and breakpoint support as well as the APB slave interface to a CoreSight debug system.

This also includes *CoreSight Cross Trigger Interface* (CTI) and *Cross Trigger Matrix* (CTM) modules for multi-core debugging.

See the *Arm® CoreSight™ Architecture Specification* for more information.

2.2.4 Trace interface

The processor supports dedicated AMBA 4 ATB interfaces for each core that outputs trace information for debugging.

The ATB interface is compatible with the CoreSight architecture. See the *Arm® AMBA® 4 ATB Protocol Specification* for more information.

2.2.5 Cross Trigger Interface

The Cortex-A73 processor implements a single cross trigger channel interface.

This external interface is connected to the CoreSight *Cross Trigger Interface* (CTI) corresponding to each core through a simplified *Cross Trigger Matrix* (CTM). See *Chapter 13 Cross Trigger* on page 13-545 for more information.

2.2.6 Design For Test interface

The processor implements a *Design For Test* (DFT) interface that enables an industry standard *Automatic Test Pattern Generation* (ATPG) tool to test logic outside of the embedded memories.

See *A.16.1 DFT interface* on page Appx-A-613 for information on these test signals.

2.2.7 Memory Built-In Self Test interface

The *Memory Built-In Self Test* (MBIST) controller interface provides support for manufacturing tests of the memories embedded in the Cortex-A73 processor.

See *Appendix A Signal Descriptions* on page Appx-A-585 for information on this interface.

2.2.8 Q-channel interface

The Q-channel interfaces enable communication to an external power controller to support dynamic retention of the cores and the L2 cache RAM.

See *2.4 Power management* on page 2-42.

2.3 Clocking and resets

The following sections describe clocking and resets:

- *2.3.1 Clocks* on page 2-33.
- *2.3.2 Input synchronization* on page 2-36.
- *2.3.3 Resets* on page 2-37.

2.3.1 Clocks

The Cortex-A73 processor has a single clock input, **CLK**.

All cores in the Cortex-A73 processor, SCU, top units and L2 control are clocked with a distributed version of **CLK**.

The Cortex-A73 processor has the following clock enable signals:

- **PCLKENDBG** see *PCLKENDBG clock enable signal* on page 2-33.
- **ACLKENM** see *ACLKENM clock enable signal* on page 2-33.
- **ACLKENS** see *ACLKENS clock enable signal* on page 2-34.
- **ATCLKEN** see *ATCLKEN clock enable signal* on page 2-35.
- **CNTCLKEN** see *CNTCLKEN clock enable signal* on page 2-35.
- **CORECLKEN[CN:0]** see *CORECLKEN[CN:0] clock enable signal* on page 2-36.

PCLKENDBG clock enable signal

The processor includes an APB interface to access the debug and performance monitoring registers.

Internally this interface is driven from **CLK**. A separate enable signal, **PCLKENDBG**, is provided to enable the external APB bus to be driven at a lower frequency, that must be an integer ratio of **CLK**. If the debug infrastructure in the system is required to be fully asynchronous to the processor clock, you can use a synchronizing component to connect the external AMBA APB to the processor.

The following figure shows a timing example of **PCLKENDBG** and **PCLK**, representing an internal clock signal, that changes the **CLK** to **PCLK** frequency ratio from 3:1 to 1:1.

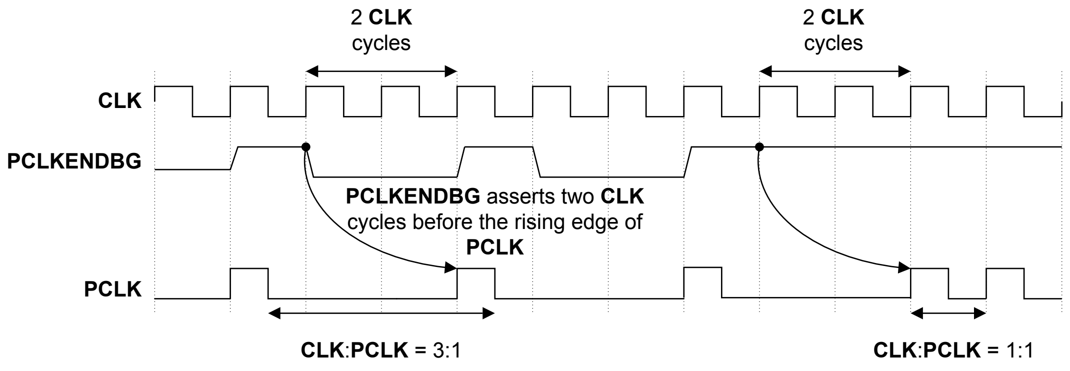

Figure 2-2 PCLKENDBG with CLK:PCLK ratio changing from 3:1 to 1:1

Note

The previous figure shows the timing relationship between the APB master clock, **PCLK** and **PCLKENDBG**, where **PCLKENDBG** asserts two clock cycles before the rising edge of **PCLK**. It is important that this relationship between **PCLK** and **PCLKENDBG** is maintained.

ACLKENM clock enable signal

The master interface supports integer ratios of the **CLK** frequency, for example 1:1, 2:1, 3:1.

These ratios are configured through the **ACLKENM** signal. In all cases AXI transfers remain synchronous.

ACLKENM asserts two **CLK** cycles before the rising edge of the external ACE clock signal, **ACLKM**. If you change the **CLK** to **ACLKM** frequency ratio, you must change **ACLKENM** correspondingly.

The following figure shows a timing example of **ACLKENM** that changes the **CLK** to **ACLKM** frequency ratio from 3:1 to 1:1.

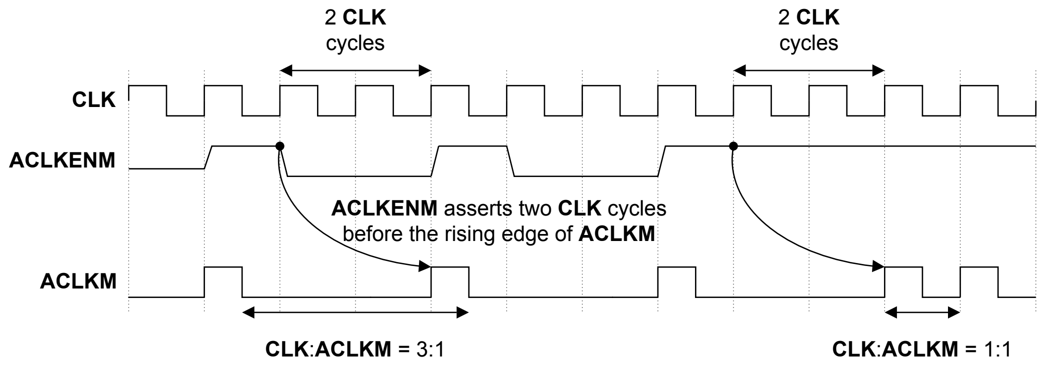

Figure 2-3 ACLKENM with CLK:ACLKM ratio changing from 3:1 to 1:1

Note

If there are any physical effects that could occur while changing the clock frequency, Arm recommends that the clock ratio is changed only while the **STANDBYWFIL2** output of the processor is asserted.

ACLKENS clock enable signal

This signal is present only if the slave interface is configured to use the ACP protocol.

The slave interface supports integer ratios of the **CLK** frequency, for example 1:1, 2:1, 3:1. These ratios are configured through the **ACLKENS** signal. In all cases AXI transfers remain synchronous.

ACLKENS asserts two **CLK** cycles before the rising edge of the external ACP clock signal, **ACLKS**. If you change the **CLK** to **ACLKS** frequency ratio, you must change **ACLKENS** correspondingly.

The following figure shows a timing example of **ACLKENS** that changes the **CLK** to **ACLKS** frequency ratio from 3:1 to 1:1.

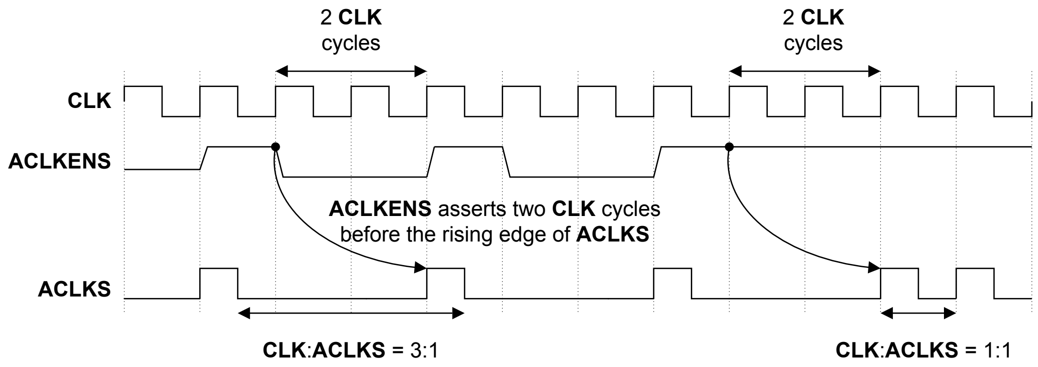

Figure 2-4 ACLKENS with CLK:ACLKS ratio changing from 3:1 to 1:1

———— **Note** ————

If there are any physical effects that could occur while changing the clock frequency, Arm recommends that the clock ratio is changed only while the **STANDBYWFIL2** output of the processor is asserted.

————————————

ATCLKEN clock enable signal

The **ATCLKEN** signal enables the ATB synchronous interface to operate at any integer multiple that is equal to or slower than the main processor clock, **CLK**.

For example, the **CLK** to **ATCLK** frequency ratio can be 1:1, 2:1, or 3:1, where **ATCLK** is the ATB bus clock. **ATCLKEN** asserts two **CLK** cycles before the rising edge of **ATCLK**. If you change the **CLK** to **ATCLK** frequency ratio, you must change **ATCLKEN** correspondingly.

The following figure shows a timing example of **ATCLKEN** that changes the **CLK** to **ATCLK** frequency ratio from 3:1 to 1:1.

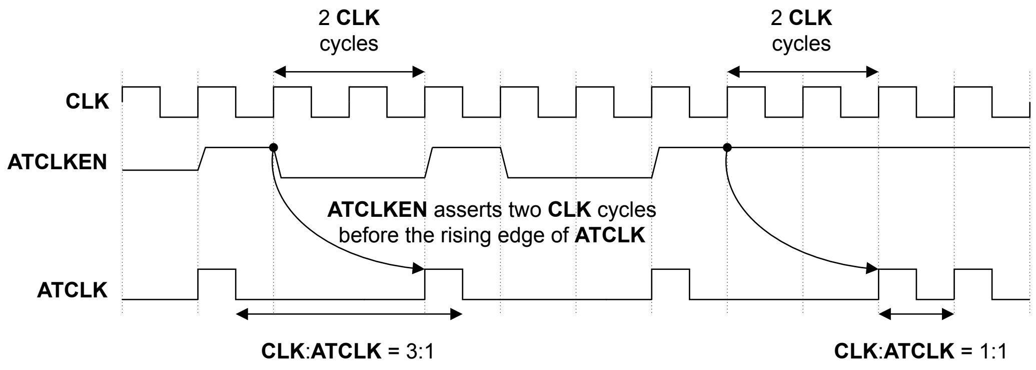

Figure 2-5 ATCLKEN with CLK:ATCLK ratio changing from 3:1 to 1:1

CNTCLKEN clock enable signal

The **CNTVALUEB** is a synchronous 64-bit binary encoded counter value that can operate at any integer multiple that is equal to or slower than the main processor clock, **CLK**, using the **CNTCLKEN** signal.

———— **Note** ————

CNTCLKEN is synchronous with **CLK**, and can be set at any cycle to follow the system clock, which is typically in the range 10 to 50 MHz.

————————————

For example, you can set the **CLK** to **CNTCLK** frequency ratio to 1:1, 2:1, or 3:1, where **CNTCLK** is the system counter clock. **CNTCLKEN** asserts two **CLK** cycles prior to the rising edge of **CNTCLK**.

The following figure shows a timing example of **CNTCLKEN** that changes the **CLK** to **CNTCLK** frequency ratio from 3:1 to 1:1.

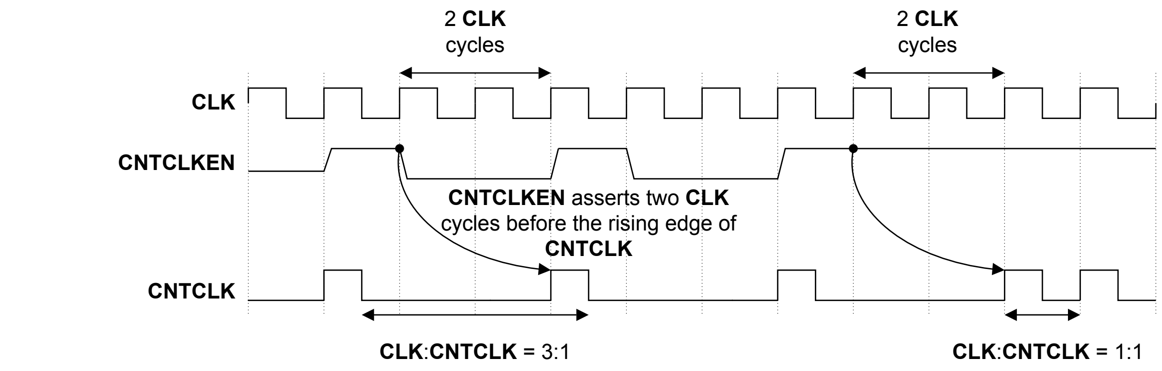

Figure 2-6 CNTCLKEN with CLK:CNTCLK ratio changing from 3:1 to 1:1

CORECLKEN[CN:0] clock enable signal

The **CORECLKEN[CN:0]** is used with the global core clock, **CLK** to generate the internal CPU CORE level clock for each CPU domain.

The following figure shows the timing of **CORECLKEN[CN:0]** that changes the **CLK** frequency to CPU CORE level clock frequency at a ratio of 4:1.

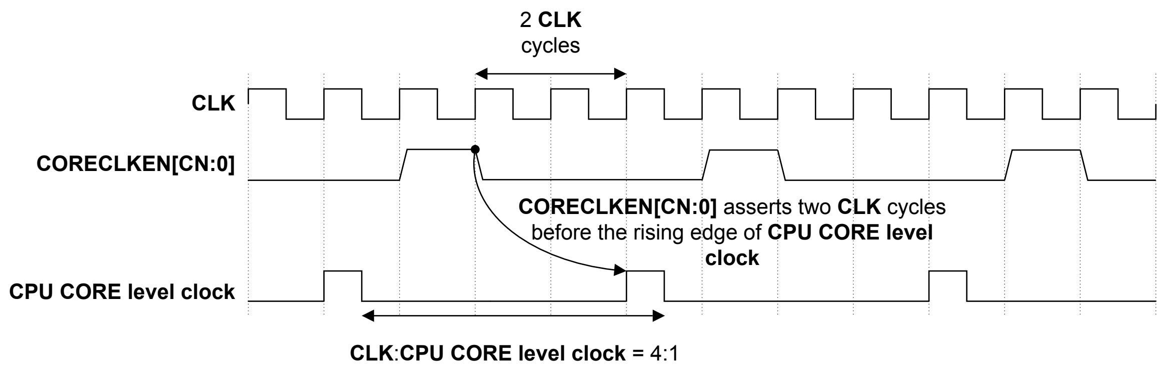
2 **CLK** cycles
CLK
CORECLKEN[CN:0]
CORECLKEN[CN:0] asserts two **CLK** cycles before the rising edge of **CPU CORE level clock**
CPU CORE level clock
CLK:**CPU CORE level clock** = 4:1

Figure 2-7 CORECLKEN[CN:0] with CLK:CPU CORE level clock

In this example, the **CLK** to **CORECLKEN[CN:0]** frequency ratio is 4:1 that enables four **CLK** cycles for each **CORECLKEN[CN:0]** cycle.

Note

The **CLK** to **CORECLKEN[CN:0]** frequency ratio must not exceed 5:1.

2.3.2 Input synchronization

The Cortex-A73 processor synchronizes the following input signals:

- **CPUQREQn[3:0]**.
- **L2QREQn**.
- **nIRQ[3:0]**.
- **nFIQ[3:0]**.
- **nVIRQ[3:0]**.
- **nVFIQ[3:0]**.
- **CTICHIN[3:0]**.
- **CTICHOUTACK[3:0]**.

The SoC can present these inputs asynchronously. All other external signals must be synchronous with reference to **CLK**.

Note

The synchronized versions of the **CTICHIN** and **CTICHOUTACK** input signals are only used if the **CISBYPASS** input signal is deasserted LOW. If the **CISBYPASS** signal is asserted HIGH the **CTICHIN** and **CTICHOUTACK** synchronizers are not used, and the SoC must present the **CTICHIN** and **CTICHOUTACK** balanced with respect to **CLK**.

2.3.3 Resets

The following table shows the reset signals and their connection information:

Table 2-1 Reset signals

Signal	Direction	Description	Connection information
nCORERESET[CN:0]	Input	Individual core resets not including Debug or ETM trace unit.	Connect to reset control logic.
nCPUPORESET[CN:0]	Input	Individual core powerup resets including both Debug and ETM trace unit.	
nPRESETDBG	Input	APB reset.	
nL2RESET	Input	L2 memory system reset.	
nMBISTRESET	Input	MBIST reset.	
L2RSTDISABLE	Input	Disable automatic L2 cache invalidate at reset. This pin is sampled during reset of the processor.	Drive this signal HIGH to disable L2 cache invalidate at reset. Drive this signal LOW to enable L2 cache invalidate at reset. This signal must only be changed when the processor is in the reset state.
WARMRSTREQ[CN:0]	Output	Processor Warm reset request.	Connect to reset control logic.
DBGRSTREQ[CN:0]	Output	Debug Warm reset request.	Connect to reset control logic.
DBGL1RSTDISABLE	Input	Disable automatic L1 cache invalidate at reset. This pin is sampled during reset of the processor.	Drive this signal LOW for normal L1 data cache behavior on reset. Drive this signal HIGH to disable automatic invalidation of L1 instruction and data caches on reset for debugging purposes only. This signal must be driven LOW during normal processor powerup sequences.

The Cortex-A73 processor has the following active-LOW reset input signals:

nCORERESET[CN:0]
Where **CN** is the number of cores minus one.
These signals initialize all resettable registers in the processor, excluding debug registers, breakpoint logic, watchpoint logic, and ETM trace unit registers.

nCPUPORESET[CN:0]
Where **CN** is the number of cores minus one.
These signals initialize all resettable registers in the processor, including debug registers, breakpoint logic, watchpoint logic, and ETM trace unit registers.

nPRESETDBG
This single, cluster-wide signal initializes the shared APB, PMU, ETM trace unit, debug registers, CTI, and CTM.

nL2RESET
This single, cluster-wide signal initializes all resettable registers in the L2 memory system, top-level logic, and the SCU logic.

nMBISTRESET
An external MBIST controller can use this signal to reset the entire design. The **nMBISTRESET** signal resets all resettable registers in the processor, for entry and exit from MBIST mode. It has priority over all other resets.

All these resets can be asynchronously:

- Asserted, HIGH to LOW.
- Deasserted, LOW to HIGH.

Reset synchronization logic inside the Cortex-A73 processor ensures that reset deassertion is synchronous for all resettable registers. The processor clock is not required for reset assertion, but the processor clock must be present for reset deassertion to ensure reset synchronization.

Note

You have to only hold reset signals active for eight processor clock cycles for the reset to take effect.

The following table shows the valid reset combinations:

Table 2-2 Valid reset combinations

Reset combination	Signals	Value	Description
Processor Cold reset	**nCPUPORESET[CN:0]**	all = 0[a]	All logic is held in reset.
	nCORERESET[CN:0]	all = X[a]	
	nPRESETDBG	0	
	nL2RESET	0	
	nMBISTRESET	1	
Processor Cold reset with debug active	**nCPUPORESET[CN:0]**	all = 0[a]	All cores are held in reset so they can be powered up. The L2 is held in reset, but must remain powered up. This enables external debug over power down for the processor.
	nCORERESET[CN:0]	all = X[a]	
	nPRESETDBG	1	
	nL2RESET	0	
	nMBISTRESET	1	

Table 2-2 Valid reset combinations (continued)

Reset combination	Signals	Value	Description
Individual core Cold reset with debug active	**nCPUPORESET[CN:0]**	[n] = 0[a]	Individual core is held in reset, so that the core can be powered up. This enables external debug over power down for the core that is held in reset.
	nCORERESET[CN:0]	[n] = X[a]	
	nPRESETDBG	1	
	nL2RESET	1	
	nMBISTRESET	1	
Individual core Warm reset with trace and debug active	**nCPUPORESET[CN:0]**	[n] = 1	Individual core is held in reset.
	nCORERESET[CN:0]	[n] = 0	
	nPRESETDBG	1	
	nL2RESET	1	
	nMBISTRESET	1	
Debug logic reset	**nCPUPORESET[CN:0]**	all = 1	Processor debug logic is held in reset.
	nCORERESET[CN:0]	all = 1	
	nPRESETDBG	0	
	nL2RESET	1	
	nMBISTRESET	1	
MBIST reset	**nCPUPORESET[CN:0]**	all = 1	All logic is held in reset.
	nCORERESET[CN:0]	all = 1	
	nPRESETDBG	1	
	nL2RESET	1	
	nMBISTRESET	0	
Normal state	**nCPUPORESET[CN:0]**	all = 1	No logic is held in reset.
	nCORERESET[CN:0]	all = 1	
	nPRESETDBG	1	
	nL2RESET	1	
	nMBISTRESET	1	

Reset sequence with reset repeaters

The reset repeater provides an asynchronously asserted, synchronously deasserted reset signal to internal registers.

When reset repeaters are present, **CLK** must be asserted while **n*RESET** (where n* generically represents any of the reset signals) is asserted. However, there is no need to stop the **CLK** when **nRESET** is deasserted.

[a] For Cold reset **nCPUPORESET** must be asserted. **nCORERESET** can be asserted but is not required.

Reset sequence without reset repeaters

When there is no reset repeater, the following steps must be taken:

1. **CLK** must be asserted while **n*RESET** (where n* generically represents any of the reset signals) is asserted LOW.
2. **CLK** must be turned off.
3. **nRESET** is deasserted.
4. **CLK** can be reasserted as explained in the following diagram.

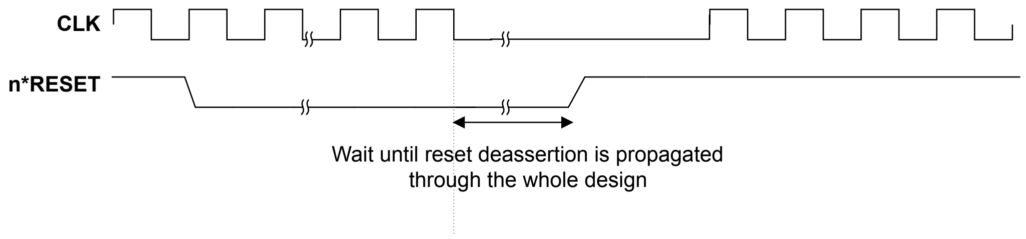
CLK
n*RESET
Wait until reset deassertion is propagated through the whole design

Figure 2-8 Reset sequence without reset repeaters

Warm reset

The Warm reset initializes all logic in the individual core apart from the Debug and ETM logic in the **CLK** domain. All breakpoints and watchpoints are retained during a Warm reset sequence.

Individual core Warm reset initializes all logic in a single core apart from its Debug, ETM, breakpoint, and watchpoint logic. Breakpoints and watchpoints for that core are retained. You must apply the correct sequence before applying Warm reset to that core.

For individual processor Warm reset:

- You must apply steps *1* on page 2-50 to *6* on page 2-50 in the core powerdown sequence, see *Individual core shutdown mode* on page 2-50, and wait until **STANDBYWFI** asserts indicating the processor is idle, before asserting **nCORERESET** for that core.
- **nCORERESET** for that core must assert as described in the following sections (*Reset sequence with reset repeaters* on page 2-39/*Reset sequence without reset repeaters* on page 2-40).
- **nL2RESET** must not assert while any individual core is active.
- **nPRESETDBG** must not assert while any individual core is actively being debugged in normal operating mode.

Note

If core dynamic retention using the CPU Q-channel interface is used, the core must be in quiescent state with **STANDBYWFI** asserted and **CPUQREQn**, **CPUQACCEPTn**, and **CPUQACCEPT** must be LOW before **nCORERESET** is applied.

WARMRSTREQ and DBGRSTREQ

The Armv8-A architecture provides a mechanism to configure whether a processor uses AArch32 or AArch64 at EL3 as a result of a Warm reset.

When the Reset Request bit in the RMR or RMR_EL3 register is set to 1, the processor asserts the **WARMRSTREQ** signal and the SoC reset controller can use this request to trigger a Warm reset of the core and change the register width state. The AA64 bit in the RMR or RMR_EL3 register selects the register width at the next Warm reset, at the highest Exception level, EL3.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information about the recommended code sequence to use, to request a Warm reset.

You must apply steps *1* on page 2-50 to *6* on page 2-50 in the core powerdown sequence, and wait until **STANDBYWFI** asserts indicating the processor is idle, before asserting **nCORERESET** for that core. **nCORERESET** must satisfy the timing requirements described in the Warm reset section.

The *Core Warm Reset Request* (CWRR) bit in the External Debug Power/Reset Control Register, EDPRCR, controls the **DBGRSTREQ** signal. An external debugger can use this bit to request a Warm reset of the processor, if it does not have access to the core Warm reset signal. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about the EDPRCR.

2.4 Power management

The Cortex-A73 processor provides mechanisms and support to control both dynamic and static power dissipation.

This section contains the following subsections:
- *2.4.1 Power domains* on page 2-42.
- *2.4.2 Dynamic Power Management* on page 2-42.
- *2.4.3 Power modes* on page 2-49.

2.4.1 Power domains

The Cortex-A73 processor can support multiple power domains. Each power domain has four possible states.

The Cortex-A73 processor supports the following power domains:

- The top-level domain (PDARTEMIS) including:
  - — SCU logic and its tag RAM.
  - — BIST multiplexors and clock modules.
  - — Various interface modules.
  - — L2 logic.
  - — Debug.
- Core 0-3 domain (PDCPU<n>), including RAMs.
- The L2 data, tag, and cache replacement RAM domain (PDL2RAM).

The figure below shows an example of the organization of the power domains.

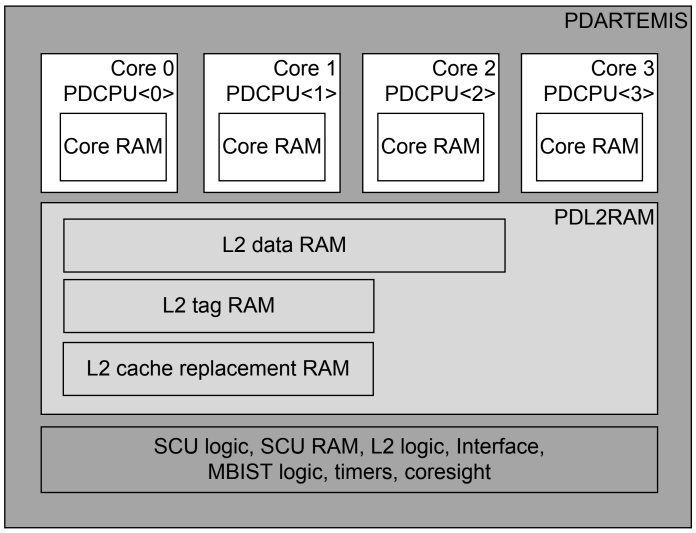

Figure 2-9 Cortex-A73 processor power domain diagram

The Cortex-A73 power domains have the following features:
- Each power domain has a separate controllable clock.
- Except for RAM-related domains, each power domain has a separate controllable reset.
- Each power domain has its own level of hierarchy.

2.4.2 Dynamic Power Management

This section describes the following dynamic power management features in the processor:

- *Normal state* on page 2-43.
- *Standby state* on page 2-43.

- *Core Wait for Interrupt* on page 2-43.
- *Core Wait for Event* on page 2-44.
- *Event communication using WFE or SEV* on page 2-44.
- *CLREXMON request and acknowledge signaling* on page 2-45.
- *L2 Wait for Interrupt* on page 2-45.
- *L2 hardware cache flush* on page 2-45.
- *Core dynamic retention* on page 2-46.
- *L2 RAMs dynamic retention* on page 2-48.

The processor exits standby mode when interrupts or external events occur.

See *Event communication using WFE or SEV* on page 2-44 for more information.

You must only use dynamic power management sequences described in the sections listed above. Any deviation from these sequences can lead to UNPREDICTABLE results.

Normal state

This is the normal mode of operation where all of the processor functionality is available. The Cortex-A73 processor uses gated clocks and gates to disable inputs to unused functional blocks. Only the logic in use to perform an operation consumes dynamic power.

Standby state

When a power domain receives a signal to enter standby state, the domain waits for transactions to complete before entering into standby state.

The following sections describe the methods of entering standby state:

- *Core Wait for Interrupt* on page 2-43.
- *Core Wait for Event* on page 2-44.
- *L2 Wait for Interrupt* on page 2-45.

Core Wait for Interrupt

Wait for Interrupt (WFI) is a feature of the Armv8-A architecture that puts the core in a standby state by disabling most of the clocks in the core while keeping the core powered up.

This reduces the power that is drawn to static leakage current only, except for a small dynamic power overhead on the logic to enable the core to wake up from WFI standby state.

Software indicates that the core can enter the WFI standby state by executing the `WFI` instruction.

When the core is executing the `WFI` instruction, the core waits for all instructions in the core to retire before entering the standby state. The `WFI` instruction ensures that all explicit memory accesses, that occurred before the `WFI` instruction in program order, have retired. For example, the `WFI` instruction ensures that the following instructions received the required data or responses from the L2 memory system:

- Load instructions.
- Cache and TLB maintenance operations.
- Store exclusive instructions.

In addition, the `WFI` instruction ensures that store instructions have updated the cache or have been issued to the SCU.

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

- A snoop request that must be serviced by the core L1 Data cache.
- A cache or TLB maintenance operation that must be serviced by the core L1 Instruction cache, data cache, or TLB.
- An APB access to the debug or trace registers residing in the core power domain.

Exit from WFI standby state occurs when the core detects a reset or one of the WFI wake-up events as described in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

On entry into WFI standby state, **STANDBYWFI[CN:0]** for that core is asserted. Assertion of **STANDBYWFI[CN:0]** guarantees that the core is in a standby state. **STANDBYWFI[CN:0]** continues to assert even if the clocks in the core are temporarily enabled because of an L2 snoop request, cache, or TLB maintenance operation or an APB access.

Note

STANDBYWFI[CN:0] does not indicate completion of L2 memory system transactions that are initiated by the processor. All Cortex-A73 processor implementations contain an L2 memory system.

Core Wait for Event

Wait for Event (WFE) is a feature of the Armv8-A architecture that can be used by a locking mechanism that is based on events to put the core in a standby state by disabling most of the clocks in the core while keeping the core powered up.

This reduces the power that is drawn to static leakage current only, except for a small dynamic power overhead on the logic to enable the core to wake up from WFE standby state.

A core enters into WFE standby state by executing the `WFE` instruction. When executing the `WFE` instruction, the core waits for all instructions in the core to complete before entering the standby state.

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 standby state, the clocks in the core are temporarily enabled without causing the core to exit WFE standby state when any of the following events are detected:

- A snoop request that must be serviced by the core L1 Data cache.
- A cache or TLB maintenance operation that must be serviced by the core L1 Instruction cache, data cache, or TLB.
- An APB access to the debug or trace registers residing in the core power domain.

A core exits WFE standby state when the core detects a reset, the assertion of the **EVENTI** input signal, or one of the WFE wake-up events as described in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

When a core enters WFE standby state, the **STANDBYWFE[CN:0]** signal for that core is asserted. Assertion of **STANDBYWFE[CN:0]** guarantees that the core is in a standby state. **STANDBYWFE[CN:0]** continues to assert even if the clocks in the core are temporarily enabled because of an L2 snoop request, cache, and TLB maintenance operation or an APB access.

Event communication using WFE or SEV

An external agent can use the **EVENTI** pin to participate in a WFE or SEV event communication with the Cortex-A73 processor.

When this pin is asserted, it sends an event message to all the cores in the device. This is similar to executing a `SEV` instruction on one core in the cluster. This enables the external agent to signal to the cores that it has released a semaphore and that the cores can leave the WFE standby state. The **EVENTI** input pin must remain HIGH for at least one **CLK** clock cycle to be visible by the cores.

The external agent can determine that at least one of the cores in the cluster has executed an `SEV` instruction by checking the **EVENTO** pin. When `SEV` is executed by any of the cores in the cluster, an event is signaled to all the cores in the device, and the **EVENTO** pin is asserted. This pin is asserted HIGH for one **CLK** clock cycle when any core in the cluster executes an `SEV` instruction.

CLREXMON request and acknowledge signaling

When the **CLREXMONREQ** input is asserted, it signals the clearing of an external global exclusive monitor and acts as a WFE wake-up event to all the cores in the cluster.

The **CLREXMONREQ** signal has a corresponding **CLREXMONACK** response signal. This forms a standard 2-wire, 4-phase handshake that can be used to signal across the voltage and frequency boundary between the core and system.

The following figure shows the **CLREXMON** request and acknowledge handshake. When the request signal is asserted, it continues to assert until an acknowledge is received. When the request is deasserted, the acknowledge can then deassert.

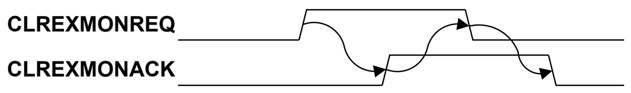

Figure 2-10 CLREXMON request and acknowledge handshake

L2 Wait for Interrupt

When all the cores are in WFI standby state, the shared L2 memory system logic that is common to all the cores can also enter a WFI standby state.

Entry into L2 WFI standby state can occur only if specific requirements are met and the following sequence applied:

- All cores are in WFI standby state and therefore, the **STANDBYWFI** output for each core is asserted. Assertion of all the core **STANDBYWFI** outputs guarantees that all the cores are in standby state.
- The SoC asserts the input pin **ACINACTM** to idle the AXI master interface. This indicates that no snoop requests will be made from the external memory system.
- If configured with an ACP interface, the SoC asserts the **AINACTS** input pin to idle the ACP interface. This indicates that the SoC will no longer send transactions on the ACP interface.

When the L2 memory system completes the outstanding transactions for the AXI interface, it can then enter the L2 WFI standby state. On entry into L2 WFI standby state, **STANDBYWFIL2** is asserted. Assertion of **STANDBYWFIL2** guarantees that the L2 memory system is idle and will not accept new transactions.

Exit from L2 WFI standby state occurs on one of the following events:
- **AINACTS** or **ACINACTM** is deasserted.
- A processor exits a standby state (**STANDBYWFI** LOW).

When a core exits from WFI standby state, **STANDBYWFI** for that core is deasserted. When the L2 memory system logic exits from WFI standby state, **STANDBYWFIL2** is deasserted.

L2 hardware cache flush

The processor provides an efficient way to fully clean and invalidate the L2 cache in preparation for powering it down without requiring the waking of a core to perform the clean and invalidate through software.

Use of L2 hardware cache flush can only occur if specific requirements are met and the following sequence applied:
1. All cores are in the WFI low-power state, so all the core **STANDBYWFI** outputs are asserted.
2. When ACP is present and all outstanding ACP transactions are complete, the SoC asserts the **AINACTS** signal to idle the ACP. This is necessary to prevent ACP transactions from allocating new entries in the L2 cache while the hardware cache flush is occurring. When **AINACTS** has been asserted, the SoC must not assert **ARVALIDS**, **AWVALIDS**, or **WVALIDS**.
3. The SoC can now assert the **L2FLUSHREQ** input.
4. The L2 performs a series of internal clean and invalidate operations to each set and way of the L2 cache. Any dirty cache lines are written back to the system using WriteBack operations. Clean cache lines can cause Evict transactions if the L2 is configured to perform these actions.

5. When the L2 completes the clean and invalidate sequence, it asserts the **L2FLUSHDONE** signal. The SoC can now deassert **L2FLUSHREQ** signal and then the L2 deasserts **L2FLUSHDONE**. Note that **L2FLUSHDONE** is asserted as soon as the last line to be flushed has been processed. It is still possible to have pending evictions on AXI interface when **L2FLUSHDONE** goes high.
6. When all outstanding snoop transactions are completed, the SoC can assert the **ACINACTM** signal. In response, the L2 asserts the **STANDBYWFIL2** signal.

The following figure shows the L2 hardware cache flush timing.

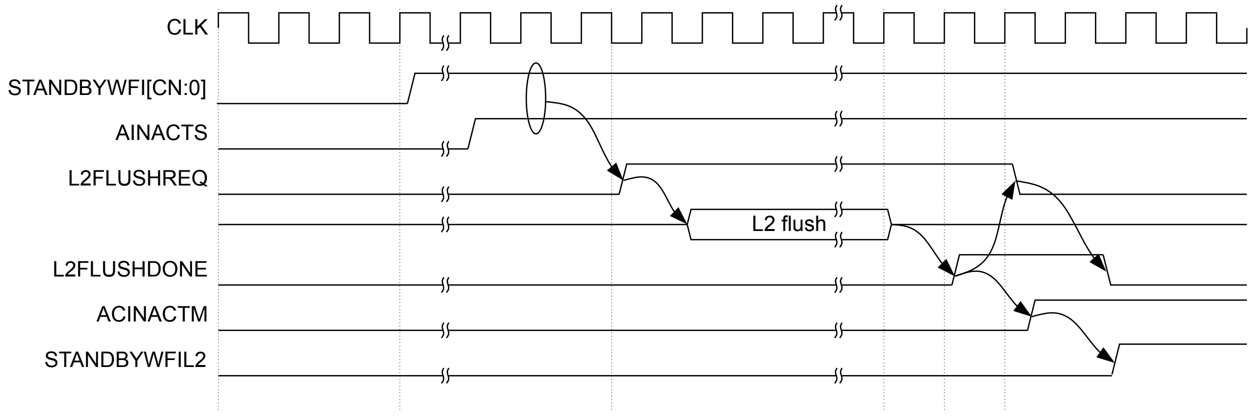

Figure 2-11 L2 hardware cache flush timing

Core dynamic retention

When a core is in WFI low-power state or WFE low-power state, the clocks to the core are stopped. During these low-power states, the core might start the clocks for short periods of time to allow it to handle snoops or other short events but it remains in the low-power state.

Whenever the clocks to a core are stopped, it is possible for an external power controller to place the core in a retention state to reduce leakage power consumption without state loss.

Each core in the processor has a CPU Q-channel interface that allows an external power controller to place the core into a retention state. This interface consists of four pins:

- **CPUQACTIVE**.
- **CPUQREQn**.
- **CPUQACCEPTn**.
- **CPUQDENY**.

The operational relationship of these signals are:

- **CPUQREQn** can only go LOW, if **CPUQACCEPTn** is HIGH and **CPUQDENY** is LOW.
- After **CPUQREQn** goes LOW, it must remain LOW until either **CPUQACCEPTn** goes LOW or **CPUQDENY** goes HIGH.
- **CPUQREQn** can then go HIGH, and must remain HIGH until both **CPUQACCEPTn** is HIGH and **CPUQDENY** is LOW.
- Each **CPUQREQn** request is followed by the assertion of either **CPUQACCEPTn** or **CPUQDENY**, but not both. **CPUQACCEPTn** cannot be asserted LOW at the same time as **CPUQDENY** is asserted HIGH.

For more information, see the *Arm® Low Power Interface Specification*.

A typical sequence of the external power controller successfully placing the core in retention state is:

1. The core executes a `WFI` instruction. The clocks in the core are stopped and **STANDBYWFI** is asserted. After the programmed number of Generic Timer **CNTVALUEB** ticks specified by

ECTLR[2:0] field has elapsed, the **CPUQACTIVE** for that core is deasserted. This hints that retention is possible for that core.
2. The external power controller asserts **CPUQREQn** to indicate that it wants to put that core into retention state.
3. If the core is still in WFI low-power state and the clocks are stopped, the core accepts the retention request by asserting **CPUQACCEPTn**.
4. While **CPUQREQn** and **CPUQACCEPTn** are both asserted, and after one core clock cycle the core is in quiescent state and the external power controller can safely put the core into retention state.
5. During retention, the **CPUQACTIVE** signal may be asserted to request exit from retention, for example if a snoop occurs to access the cache of the quiescent core.
6. The external power controller brings the core out of retention and deasserts **CPUQREQn**.
7. The core deasserts **CPUQACCEPTn** to complete the handshake.
8. The clocks in the core are restarted temporarily to allow the snoop request to the core to proceed.
9. After the snoop access is complete, and after the programmed number of Generic Timer **CNTVALUEB** ticks specified by **ECTLR[2:0]** field has elapsed, the **CPUQACTIVE** for that core is deasserted.
10. **CPUQREQn** and **CPUQACCEPTn** are then asserted in sequence. After on core clock cycle, the core has reentered quiescent state and the external power controller can put the core into retention state again.
11. When the core exits WFI low-power state, **CPUQACTIVE** is asserted.
12. The power controller brings the core out of retention and **CPUQREQn** is then deasserted. The core exits WFI low-power state, and **CPUQACCEPTn** is deasserted.

The following figure shows a typical sequence where the external power controller successfully places the core in retention state.

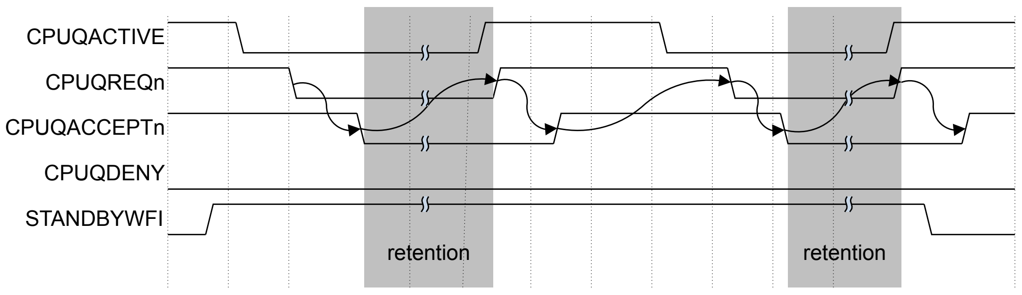

Figure 2-12 Successful retention timing

The core enters WFI low-power state and deasserts **CPUQACTIVE**. The external power controller asserts **CPUQREQn**. If the core cannot safely enter quiescent state, it asserts **CPUQDENY** instead of **CPUQACCEPTn**. When this occurs, the external power controller cannot put that core into retention state. The external power controller must then deassert **CPUQREQn**, then the core deasserts **CPUQDENY**.

The following figure shows a sequence where the external power controller attempts to put a core in retention state but the core denies the request.

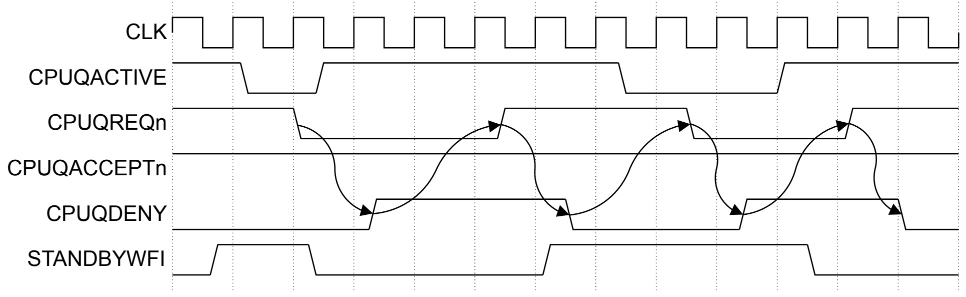

Figure 2-13 Denied retention timing

Guidelines on the use of core dynamic retention

As cores generally only stay in WFE low-power state for a short period of time, Arm recommends that you only take a core into retention when it is in WFI low-power state.

If the L1 data cache of a core that is in WFI low-power state contains data that is likely to be the target of frequent snoops from other cores, entering quiescent state and retention is likely to be inefficient.

When using the core retention feature, you must consider the following points:

- The Processor dynamic retention control field in the Extended Control Register, ECTLR, must be set to a nonzero value to enable this feature. If this field is `0b000`, all assertions of **CPUQREQn** LOW receive **CPUQDENY** responses.
- If the core dynamic retention feature is not used, **CPUQREQn** must be tied HIGH and the ECTLR retention control field set to disabled.

L2 RAMs dynamic retention

L2 RAM dynamic retention mode provides a way of saving power in an idle processor while allowing quick wake-up to service a snoop from ACE. The core supports dynamic retention of the L2 Data, Tag, and Replacement RAMs.

The processor has an L2 Q-channel interface that allows an external power controller to place the L2 RAMs into a retention state.

L2 RAM dynamic retention mode is entered and exited using the following sequence of events:

1. All cores are in WFI or WFE low-power state and therefore, all the cores **STANDBYWFI** or **STANDBYWFE** outputs are asserted.
2. When all pending L2 activity is complete, and the L2 remains idle for the programmed number of Generic Timer **CNTVALUEB** ticks, as specified by L2ECTLR[2:0] field, the L2 deasserts **L2QACTIVE**.
3. The external power controller asserts **L2QREQn** to indicate that it wants to put the L2 RAMs into retention state.
4. If the L2 is still idle, it accepts the retention request by asserting **L2QACCEPTn**.
5. While **L2QREQn** and **L2QACCEPTn** are both asserted, the power controller can safely put the L2 RAMs into retention state.
6. If the L2 detects that one or more cores have exited WFI low-power state, the ACP becomes active, a snoop request must be serviced, or there is a debug APB access, the L2 asserts **L2QACTIVE** to request exit from retention.
7. The power controller brings the L2 RAMs out of retention and deasserts **L2QREQn**.
8. The L2 deasserts **L2QACCEPTn** to complete the handshake.

The following figure shows the L2 dynamic retention timing.

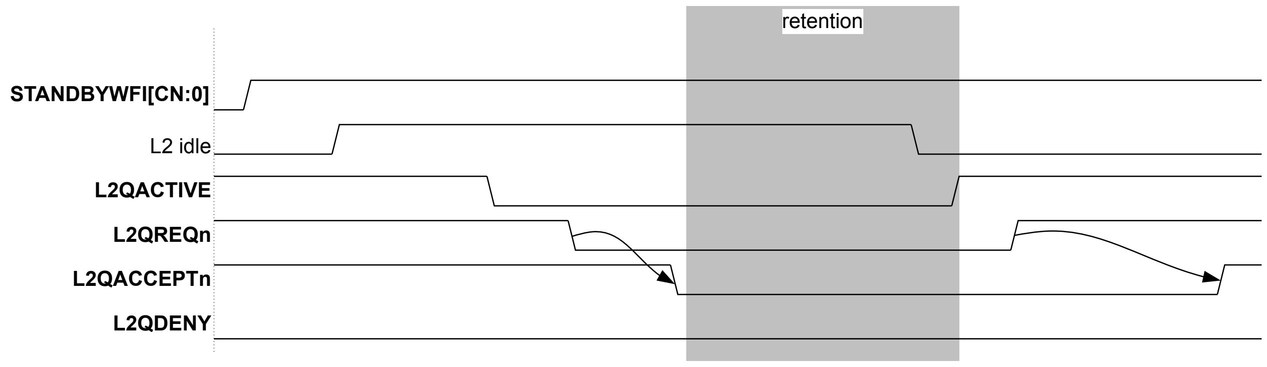

Figure 2-14 L2 dynamic retention timing

If the L2 exits idle before step *4* on page 2-48, it asserts **L2QDENY** instead of **L2QACCEPTn**. In response, the power controller must deassert **L2QREQn**, causing the L2 to deassert **L2QDENY**.

The L2 dynamic retention control field in the L2 Extended Control Register, L2ECTLR, must be set to a nonzero value to enable this feature. If this field is `0b000`, all assertions of **L2QREQn** LOW receive **L2QDENY** HIGH responses.

If the L2 dynamic retention feature is not used, **L2QREQn** must be tied HIGH and the L2ECTLR retention control field set to disabled.

2.4.3 Power modes

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

However, only some power domain state combinations are valid and supported.

Table 2-4 Supported processor power states on page 2-50 shows the supported power domain states for the Cortex-A73 processor. The following table defines the terms used.

Table 2-3 Power state description

Power state	Description
Off	Block is power gated
Ret	Logic or RAM retention power only
Standby	Core is architecturally clock gated at the top of the clock tree
On	Block is active

Caution

States not shown in *Table 2-4 Supported processor power states* on page 2-50 are unsupported and must not occur.

Table 2-4 Supported processor power states

Power domains			Description
PDARTEMIS	**PDL2RAM**	**PDCPU<n>**	
Off	Off	Off	Cluster off.
Off	On/Ret	Off	L2 Cache Dormant Mode.
On	Ret	Off/Ret/Standby	All cores are off, in Standby, or in Retention state. L2 RAMs retained.
On	On	Off/Ret/Standby	Processor and L2 RAMs on. All cores are off, in standby, or in retention state.
On	On	On	Processor on, SCU/L2 RAMs Active.

You must follow the dynamic power management and powerup and powerdown sequences described in the following sections. Any deviation from these sequences can lead to UNPREDICTABLE results.

Individual core shutdown mode

In this mode, the PDCPU power domain for an individual core is shut down and all state is lost.

For full shutdown of the Cortex-A73 processor, including implementations with a single core, see *Cluster shutdown mode without system driven L2 flush* on page 2-51 and *Cluster shutdown mode with system driven L2 flush* on page 2-51.

To enable a core to be powered down, the implementation must place the core on a separately controlled power supply. In addition, you must clamp the outputs of the core to benign values while the entire cluster is powered down, to indicate that the core is idle.

To power down the core, apply the following sequence:

1. Disable the data cache, by clearing the SCTLR.C bit, or the HSCTLR.C bit if in Hyp mode. This prevents more data cache allocations and causes cacheable memory attributes to change to Normal Non-cacheable. Subsequent loads and stores do not access the L1 or L2 caches.
2. Clean and invalidate all data from the L1 data cache to make sure all dirty lines are evicted.
3. Disable data coherency with other cores in the cluster, by clearing the ECTLR.SMPEN bit. Clearing the SMPEN bit enables the core to be taken out of coherency by preventing the core from receiving cache or TLB maintenance operations that are broadcast by other cores in the cluster.
4. Execute an `ISB` instruction to ensure that all of the register changes from the previous steps have been committed.
5. Execute a `DSB SY` instruction to ensure that all cache, TLB and branch predictor maintenance operations that are issued by any core in the cluster device before the SMPEN bit was cleared have completed.
6. Execute a `WFI` instruction and wait until the **STANDBYWFI** output is asserted to indicate that the core is the WFI standby state.
7. Deassert **DBGPWRDUP** LOW. This prevents any external debug access to the core.
8. Assert **nCPUPORESET** LOW for at least eight cycles to propagate reset.
9. Activate the core output clamps.
10. Remove power from the PDCPU power domain.

Note

CORECLKEN[CN:0] must be maintained while the core is shut down.

To power up the core, apply the following sequence:

1. Ensure **DBGPWRDUP** is held LOW to prevent any external debug access to the core.
2. Apply power to the PDCPU power domain. Keep the state of the signals **nCPUPORESET** and **DBGPWRDUP** LOW.
3. Release the core output clamps.
4. Deassert resets.
5. Assert **DBGPWRDUP** HIGH to allow external debug access to the core.
6. Set the SMPEN bit to 1 to enable snooping into the core.
7. If required, use software to restore the state of the core as it was prior to powerdown.

Cluster shutdown mode without system driven L2 flush

This is the mode where the PDARTEMIS, PDL2RAM, and PDCPU power domains are shut down and all state is lost.

In this section, a lead core is defined as the last core to switch off, or the first core to switch on. To power down the cluster, apply the following sequence:

1. Ensure all non-lead cores are in shutdown mode, see *Individual core shutdown mode* on page 2-50.
2. Follow steps 1 and 2 in *Individual core shutdown mode* on page 2-50.
3. If the ACP interface is configured, ensure that any master connected to the interface does not send new transactions, then assert **AINACTS**.
4. Clean and invalidate all data from the L2 Data cache.
5. Follow steps 3 to 10 in *Individual core shutdown mode* on page 2-50.
6. To ensure that the core is removed from system coherency, or that it will no longer receive snoop/DVM messages, assert **ACINACTM**. Then, wait until the **STANDBYWFIL2** output is asserted to indicate that the L2 memory system is idle. All Cortex-A73 processor implementations contain an L2 memory system.
7. Assert **nL2RESET** LOW. The **nCPUPORESET** should already be LOW for all cores at this step.
8. Activate the cluster output clamps.
9. Remove power from the PDARTEMIS and PDL2RAM power domains.

Note

For device powerdown, all operations on the lead core must occur after the equivalent step on all non-lead cores.

To power up the cluster, apply the following sequence:

1. Hold **L2RSTDISABLE** LOW.
2. Apply power to the PDARTEMIS and PDL2RAM domains while keeping the **nCPUPORESET** and **nL2RESET** signals LOW.
3. Release the cluster output clamps.
4. The cores can then be powered as required for the Cortex-A73 processor.

Cluster shutdown mode with system driven L2 flush

This is the mode where the PDARTEMIS, PDL2RAM, and PDCPU power domains are shut down and all state is lost.

To power down the cluster, apply the following sequence:

1. Ensure that all cores are in shutdown mode, see *Individual core shutdown mode* on page 2-50.
2. Ensure that there will not be any ACP transactions.
3. The SoC asserts the **AINACTS** signal to idle the ACP. This is necessary to prevent ACP transactions from allocating new entries in the L2 cache while the hardware cache flush is occurring.
4. Assert **L2FLUSHREQ** HIGH.
5. Hold **L2FLUSHREQ** HIGH until **L2FLUSHDONE** is asserted.
6. Deassert **L2FLUSHREQ**.
7. Stop incoming snoops/DVM by disconnecting from coherency.

8. Assert **ACINACTM**, then wait until the **STANDBYWFIL2** output is asserted to indicate that the L2 memory system is idle. All Cortex-A73 processor implementations contain an L2 memory system.
9. For each core in the cluster, assert **nCPUPORESET** LOW and assert **nL2RESET** LOW.
10. Activate the cluster output clamps.
11. Remove power from the PDARTEMIS and PDL2RAM power domains.

To power up the cluster, apply the following sequence:

1. Hold **L2RSTDISABLE** LOW.
2. Apply power to the PDARTEMIS and PDL2RAM domains while keeping **nCPUPORESET**, **nL2RESET**, and **L2RSTDISABLE** LOW.
3. Release the cluster output clamps.
4. The cores can then be powered as required for the Cortex-A73 processor.

Dormant mode

Optionally, the Dormant mode is supported in the cluster. In this mode, all the cores and L2 control logic are powered down while the L2 cache RAMs are powered up and retain state.

The RAM blocks that remain powered up during Dormant mode are:

- L2 tag RAMs.
- L2 data RAMs.
- L2 replacement RAMs.

To support Dormant mode, you must ensure:

- The L2 cache RAMs are in a separate power domain.
- That all inputs to the L2 cache RAMs are clamped to benign values. This avoids corrupting data when the cores and L2 control power domains enter and exit power-down state.

Before entering Dormant mode the architectural state of the cluster, excluding the contents of the L2 cache RAMs that remain powered up, must be saved to external memory.

As part of the exit from Dormant mode to Normal state, the SoC must perform a Cold reset sequence. The SoC must assert the reset signals until power is restored. After power is restored, the cluster exits the Cold reset sequence, and the architectural state must be restored.

To enter Dormant mode, apply the following sequence:

1. Disable the data cache, by clearing the SCTLR.C bit, or the HSCTLR.C bit if in Hyp mode. This prevents more data cache allocations and causes cacheable memory attributes to change to Normal Non-cacheable. Subsequent loads and stores do not access the L1 or L2 caches.
2. Clean and invalidate all data from the L1 Data cache. The L2 duplicate snoop tag RAM for this core is now empty. This prevents any new data cache snoops or data cache maintenance operations from other cores in the cluster being issued to this core.
3. Disable data coherency with other cores in the cluster, by clearing the ECTLR.SMPEN bit. Clearing the SMPEN bit enables the core to be taken out of coherency by preventing the core from receiving cache or TLB maintenance operations broadcast by other cores in the cluster.
4. Execute an `ISB` instruction to ensure that all of the register changes from the previous steps have been committed.
5. Execute a `DSB SY` instruction to ensure that all cache, TLB and branch predictor maintenance operations that are issued by any core in the cluster before the SMPEN bit was cleared have completed. In addition, this ensures that all state saving has completed.
6. Execute a `WFI` instruction and wait until the **STANDBYWFI[CN:0]** output is asserted, to indicate that the core is in idle and low-power state.
7. Repeat the previous steps for all cores, and wait for all **STANDBYWFI[CN:0]** outputs to be asserted.
8. If the ACP interface is configured, ensure that any master connected to the interface does not send new transactions, then assert **AINACTS**.
9. If ACE is implemented, the SoC asserts the input pin **ACINACTM** to idle the AXI master interface after all snoop transactions have been sent on the interface.

When the L2 has completed the outstanding transactions for the AXI master and slave interfaces, **STANDBYWFIL2** is asserted to indicate that the L2 memory system is idle. All Cortex-A73 processor implementations contain an L2 memory system.

10. When all cores **STANDBYWFI[n]** and **STANDBYWFIL2** are asserted, the cluster is ready to enter Dormant mode.
11. Activate the L2 cache RAM output clamps.
12. For each core in the cluster, assert **nCPUPORESET** LOW and assert **nL2RESET** LOW.
13. Remove power from the PDCPU and PDARTEMIS power domains.

To exit Dormant mode, apply the following sequence:

1. Apply a Cold reset sequence. You must apply resets to the cores and the L2 memory system logic until power is restored. During this reset sequence, **L2RSTDISABLE** must be held HIGH to disable the L2 cache hardware reset mechanism.
2. When power has been restored, release the L2 cache RAM input clamps.
3. Continue the Cold reset sequence with **L2RSTDISABLE** held HIGH.
4. The architectural state must be restored, if required.

Chapter 3
Programmers Model

This chapter describes the registers and provides information for programming the Cortex-A73 processor.

It contains the following sections:

3.1 About the programmers model

The Cortex-A73 processor implements the Armv8-A architecture. This includes:

- Support for all the Exception levels, EL0-EL3.
- Support for both Execution states, AArch64 and AArch32, at each Exception level.
- The following instruction sets:

  AArch64 Execution state
  The A64 instruction set.

  AArch32 Execution state
  The T32 and A32 instruction sets.
- Optionally, an implementation can include The Cryptographic Extension, which provides additional instructions in all instruction sets.

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

This section contains the following subsections:

- *3.1.1 Advanced SIMD and Floating-point support* on page 3-55.
- *3.1.2 Memory model* on page 3-55.
- *3.1.3 Jazelle implementation* on page 3-56.
- *3.1.4 Modes of operation* on page 3-56.

3.1.1 Advanced SIMD and Floating-point support

Advanced SIMD is a media and signal processing architecture that adds instructions targeted primarily at audio, video, 3-D graphics, image, and speech processing.

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

Note

Advanced SIMD, its associated implementations, and supporting software, are commonly referred to as NEON.

All scalar floating-point instructions are available in the A64 instruction set. All VFP instructions are available in the A32 and T32 instruction sets.

The same Advanced SIMD instructions are available in both the A32 and T32 instruction sets. The A64 instruction set offers additional Advanced SIMD instructions, including double-precision floating-point vector operations.

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

See *Chapter 14 Advanced SIMD and Floating-point Support* on page 14-562 for implementation-specific information.

3.1.2 Memory model

The Cortex-A73 processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word.

The processor can store words in memory as either:

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

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about big-endian and little-endian memory systems.

Note

Instructions are always little-endian.

3.1.3 Jazelle implementation

The processor supports a trivial Jazelle implementation. This means:

- Jazelle state is not supported.
- The `BXJ` instruction behaves as a `BX` instruction.

In the trivial Jazelle implementation, the core does not accelerate the execution of any bytecodes, and the JVM uses software routines to execute all bytecodes. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

3.1.4 Modes of operation

In AArch32, the core has the following instruction set operating states controlled by the T bit and J bit in the CPSR.

A32 The processor executes 32-bit, word-aligned A32 instructions.

T32 The processor executes 16-bit and 32-bit, halfword-aligned T32 instructions.

The J bit and the T bit determine the instruction set used by the processor. The following table shows the encoding of these bits.

Table 3-1 CPSR J and T bit encoding

J	T	Instruction set state
0	0	A32
0	1	T32

Note

- The processor does not support Jazelle state. This means there is no processor state where the J bit is 1 and T bit is 0.
- The processor does not support T32EE state. This means there is no processor state where the J bit is 1 and T bit is 1.
- Transition between A32 and T32 instruction set states does not affect the processor mode or the register contents.

3.2 Armv8-A architecture concepts

The following sections describe the Armv8-A architectural concepts. Each section introduces the corresponding terms that are used to describe the architecture.

———— **Note** ————

A thorough understanding of the terminology defined in this section is a prerequisite for reading the remainder of this manual.

————————————

This section contains the following subsections:

3.2.1 Execution state

The execution state defines the processor execution environment, including:

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

The execution states are:

AArch64

The 64-bit execution state. This execution state:

- Features 31 64-bit general purpose registers, with a 64-bit *Program Counter* (PC), *Stack Pointer* (SP), and *Exception Link Registers* (ELRs).
- Provides a single instruction set, A64. For more information, see *3.2.7 Instruction set state* on page 3-63.
- Defines the Armv8 exception model, with four exception levels, EL0-EL3, that provide an execution privilege hierarchy.
- Features *Virtual Addresses* (VAs) held in 64-bit registers. The Cortex-A73 VMSA implementation maps these to 40-bit *Physical Address* (PA) maps.
- Defines a number of PSTATE elements that hold processor state. The A64 instruction set includes instructions that operate directly on various PSTATE elements.
- Names each system register using a suffix that indicates the lowest exception level at which the register can be accessed.

AArch32

The 32-bit execution state. This execution state is backwards-compatible with implementations of the Armv7-A architecture profile that include the Security Extensions and the Virtualization Extensions:

- Features 13 32-bit general purpose registers, and a 32-bit PC, SP, and *link register* (LR). Some of these registers have multiple banked instances for use in different processor modes.
- Provides two instruction sets, A32 and T32. For more information, see *3.2.7 Instruction set state* on page 3-63.
- Provides an exception model that maps the Armv7 exception model onto the Armv8 exception model and Exception levels. For exceptions taken to an Exception level that is using AArch32, this supports the Armv7 exception model use of processor *modes*.
- Features 32-bit VAs. The VMSA maps these to PA maps that can support PAs of up to 40 bits.
- Collects processor state into the *Current Program State Register* (CPSR).

The processor can move between Execution states only on a change of Exception level, and subject to the rules given in *3.2.4 Rules for changing execution state* on page 3-60. This means different software layers, such as an application, an operating system kernel, and a hypervisor, executing at different Exception levels, can execute in different Execution states.

3.2.2 Exception levels

The Armv8 exception model defines exception levels EL0-EL3, where:

- EL0 has the lowest software execution privilege, and execution at EL0 is called unprivileged execution.
- Increased exception levels, from 1 to 3, indicate increased software execution privilege.
- EL2 provides support for processor virtualization.
- EL3 provides support for a Secure state, see *3.2.3 Security state* on page 3-59.

The Cortex-A73 processor implements all the Exception levels, EL0-EL3, and supports both Execution states, AArch64 and AArch32, at each Exception level.

Execution can move between Exception levels only on taking an exception, or on returning from an exception:

- On taking an exception, the Exception level either increases or remains the same. The Exception level cannot decrease on taking an exception.
- On returning from an exception, the Exception level either decreases or remains the same. The Exception level cannot increase on returning from an exception.

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

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

Exception levels, and privilege levels, are defined within a particular Security state, and *3.2.6 Armv8 security model* on page 3-61 describes the permitted combinations of Security state and Exception level.

Exception terminology

This section defines terms used to describe the navigation between exception levels.

Terminology for taking an exception

An exception is generated when the processor first responds to an exceptional condition.

The processor state at this time is the state the exception is *taken from*. The processor state immediately after taking the exception is the state the exception is *taken to*.

Terminology for returning from an exception

To return from an exception, the processor must execute an exception return instruction. The processor state when an exception return instruction is committed for execution is the state the exception *returns from*. The processor state immediately after the execution of that instruction is the state the exception *returns to*.

Exception level terminology

An Exception level, ELn, with a larger value of n than another Exception level, is described as being a higher Exception level than the other Exception level. For example, EL3 is a higher Exception level than EL1.

An Exception level with a smaller value of n than another Exception level is described as being a lower Exception level than the other Exception level. For example, EL0 is a lower Exception level than EL1.

An Exception level is described as:
- *Using AArch64* when execution in that Exception level is in the AArch64 Execution state.
- *Using AArch32* when execution in that Exception level is in the AArch32 Execution state.

Typical exception level usage model

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

EL0	Applications.
EL1	OS kernel and associated functions that are typically described as *privileged*.
EL2	Hypervisor.
EL3	Secure monitor.

3.2.3 Security state

An Armv8 implementation that includes the EL3 Exception level provides the following Security states, each with an associated memory address space:

Secure state

In Secure state, the processor:
- Can access both the Secure memory address space and the Non-secure memory address space.
- When executing at EL3, can access all the system control resources.

Non-secure state

In Non-secure state, the processor:
- Can access only the Non-secure memory address space.
- Cannot access the Secure system control resources.

The AArch32 Security state model is unchanged from the model for an Armv7 implementation that includes the Security Extensions and the Virtualization Extensions. When the implementation uses the AArch32 Execution state for all Exception levels, many system registers are banked to provide Secure and Non-secure instances, and:
- The Secure instance is accessible only at EL3.
- The Non-secure instance is accessible at EL1 or higher.
- The two instances of a banked register have the same name.

The *3.2.6 Armv8 security model* on page 3-61 describes how the Security state interacts with other aspects of the Armv8 architectural state.

3.2.4 Rules for changing execution state

This introduction to moving between execution states does not consider exceptions caused by debug events.

The execution state, AArch64 or AArch32, can change only on a change of exception level, meaning it can change only on either:

- Taking an exception to a higher exception level.
- Returning from an exception to a lower exception level.

———— **Note** ————

The execution state cannot change if, on taking an exception or on returning from an exception, the exception level remains the same.

————————————

On taking an exception to a higher exception level, the execution state:

- Can either:
  - Remain the same.
  - Increase from AArch32 state to AArch64 state.
- Cannot decrease from AArch64 state to AArch32 state.

On returning from an exception to a lower exception level, the execution state:

- Can either:
  - Remain the same.
  - Decrease from AArch64 state to AArch32 state.
- Cannot increase from AArch32 state to AArch64 state.

On powerup and on reset, the processor enters EL3, the highest exception level. The execution state for this exception level is a property of the implementation, and is determined by a configuration input signal. For the other exception levels the execution state is determined as follows:

- For an exception return to EL0, the EL0 execution state is specified as part of the exception return, subject to the rules given in this section.
- Otherwise, the execution state is determined by one or more system register configuration bits, that can be set only in a higher exception level.

3.2.5 Stack pointer selection

Stack pointer behavior depends on the execution state, as follows:

AArch64

In EL0, the stack pointer, SP, maps to the SP_EL0 stack pointer register.

Taking an exception selects the default stack pointer for the target exception level, meaning SP maps to the SP_ELx stack pointer register, where x is the exception level.

Software executing in the target exception level can execute an `MSR SPSel, #Imm1` instruction to select whether to use the default SP_ELx stack pointer, or the SP_EL0 stack pointer.

The selected stack pointer can be indicated by a suffix to the exception level:

t	Indicates use of the SP_EL0 stack pointer.
h	Indicates use of the SP_ELx stack pointer.

The following table shows the set of AArch64 stack pointer options.

Table 3-2 AArch64 stack pointer options

Exception level	AArch64 stack pointer options
EL0	EL0t
EL1	EL1t, EL1h
EL2	EL2t, EL2h
EL3	EL3t, EL3h

AArch32

In AArch32 state, each mode that can be the target of an exception has its own banked copy of the stack pointer. For example, the banked stack pointer for Hyp mode is called SP_hyp. Software executing in one of these modes uses the banked stack pointer for that mode.

The modes that have banked copies of the stack pointer are FIQ mode, IRQ mode, Supervisor mode, Abort mode, Undefined mode, Hyp mode, and Monitor mode. Software executing in User mode or System mode uses the User mode stack pointer, SP_usr.

For more information, see *3.2.8 AArch32 execution modes* on page 3-63.

3.2.6 Armv8 security model

The Cortex-A73 processor implements all of the exception levels. This means:

- EL3 exists only in Secure state and a change from Secure state to Non-secure state is made only by an exception return from EL3.
- EL2 exists only in Non-secure state.

To provide compatibility with Armv7, the exception levels available in Secure state are modified when EL3 is using AArch32. The following sections describe the security model:

- *Security model when EL3 is using AArch64* on page 3-61.
- *Security model when EL3 is using AArch32* on page 3-62.

Security model when EL3 is using AArch64

The following figure shows the security model and the expected use of the different exception levels when EL3 is using AArch64, as well as how instances of EL0 and EL1 are present in both security states.

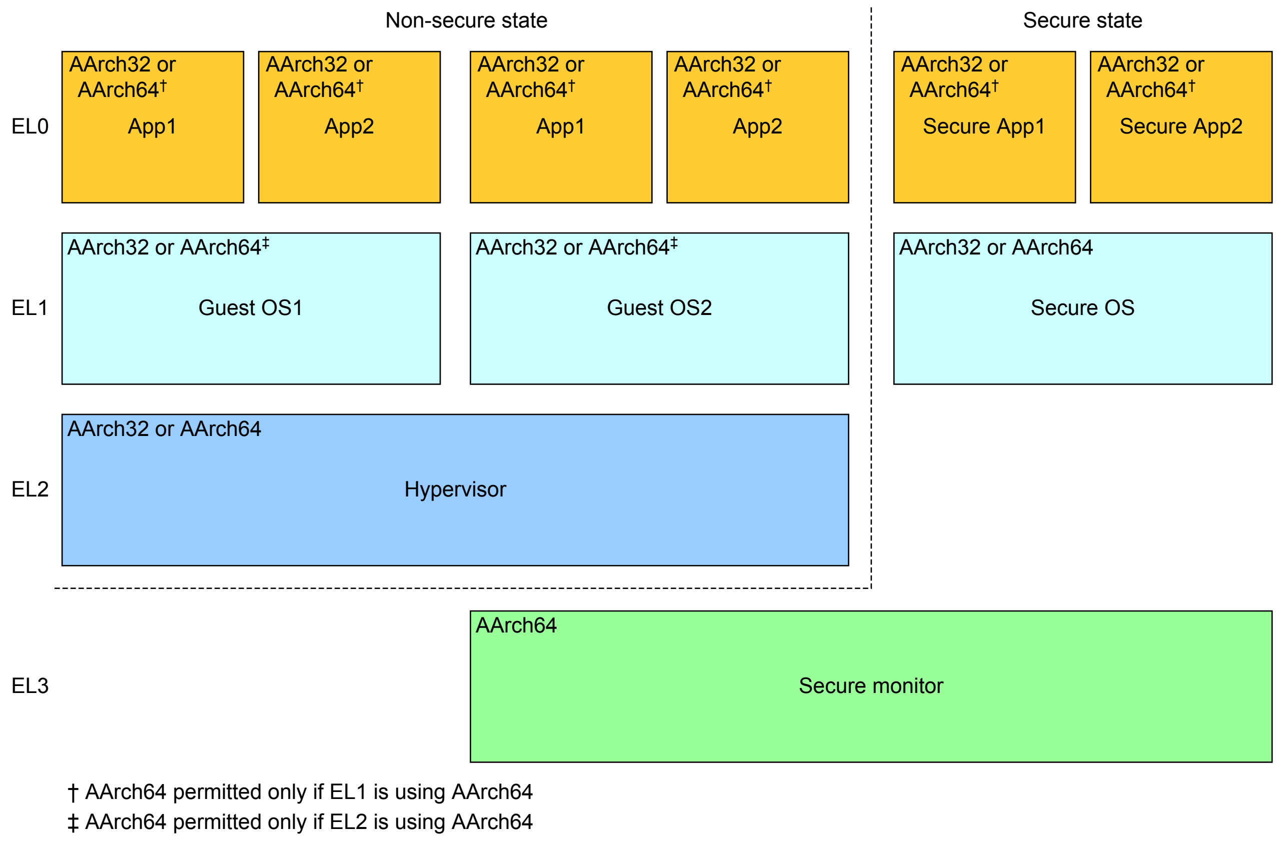

Figure 3-1 Armv8 security model when EL3 is using AArch64

Security model when EL3 is using AArch32

To provide software compatibility with VMSAv7 implementations that include the security extensions, in Secure AArch32 state, all modes other than User mode have the same execution privilege.

This means that, in an implementation where EL3 is using AArch32, the security model is as shown in the following figure. This figure also shows the expected use of the different exception levels and processor modes.

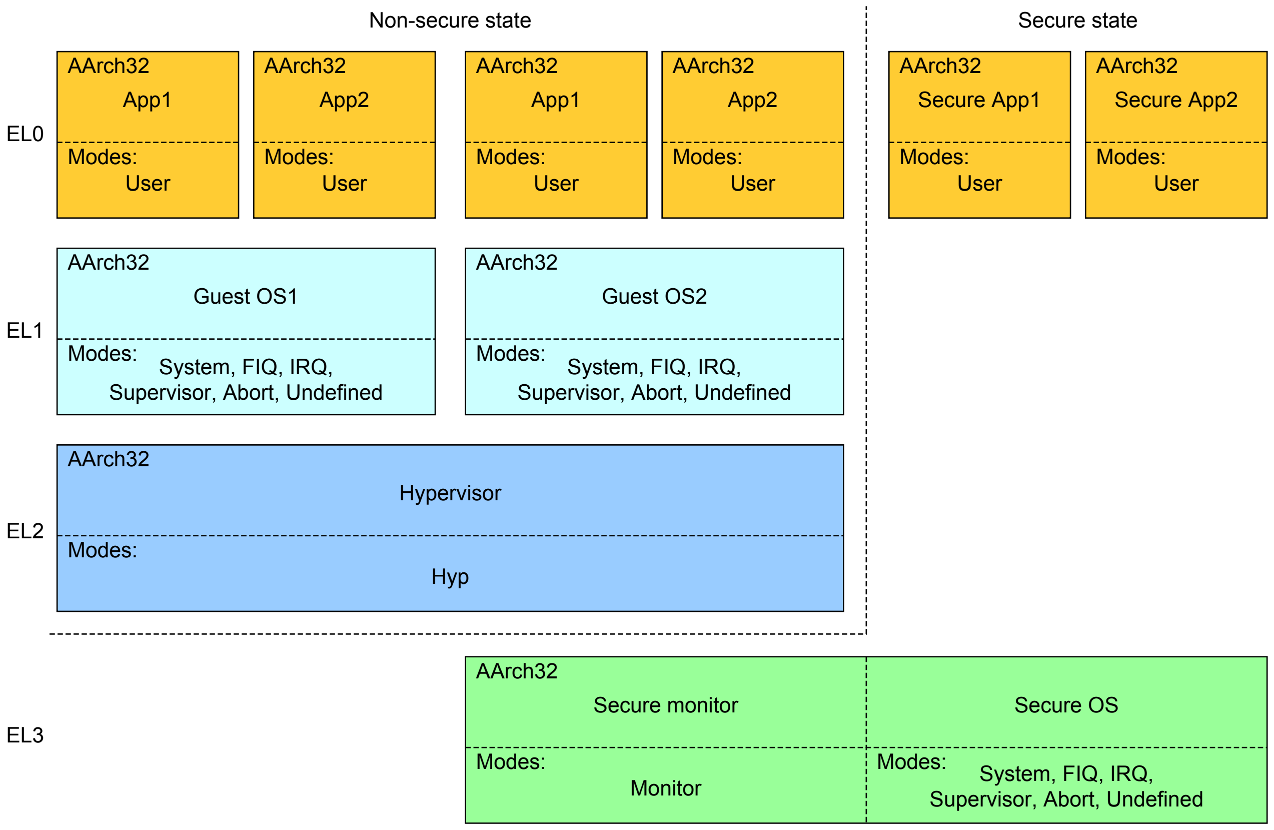

Figure 3-2 Armv8 security model when EL3 is using AArch32

For more information about the AArch32 processor modes see *3.2.8 AArch32 execution modes* on page 3-63.

3.2.7 Instruction set state

The processor instruction set state determines the instruction set that the processor executes.

The instruction sets depend on the execution state:

AArch64

AArch64 state supports only a single instruction set, called A64. This is a fixed-width instruction set that uses 32-bit instruction encodings.

AArch32

AArch32 state supports the following instruction sets:

A32 This is a fixed-length instruction set that uses 32-bit instruction encodings. Before the introduction of Armv8, it was called the Arm instruction set.

T32 This is a variable-length instruction set that uses both 16-bit and 32-bit instruction encodings. Before the introduction of Armv8, it was called the Thumb instruction set state.

3.2.8 AArch32 execution modes

Armv7 and earlier versions of the Arm architecture define a set of named processor modes, including modes that correspond to different exception types.

For compatibility, AArch32 state retains these processor modes.

The following table shows the AArch32 processor modes, and the exception level of each mode.

Table 3-3 AArch32 processor modes and associated exception levels

AArch32 processor mode	EL3 using	Security state	Exception level
User	AArch32 or AArch64	Non-secure or Secure	EL0
System, FIQ, IRQ, Supervisor, Abort, Undefined	AArch64	Non-secure or Secure	EL1
	AArch32	Non-secure	EL1
		Secure	EL3
Hyp	AArch32 or AArch64	Non-secure only	EL2
Monitor	AArch32	Secure only	EL3

When the *EL3 using* column of the above table shows:

AArch64

The row refers to information shown in *Figure 3-1 Armv8 security model when EL3 is using AArch64* on page 3-62.

AArch32

The row refers to information shown in *Figure 3-2 Armv8 security model when EL3 is using AArch32* on page 3-63.

A processor mode name does not indicate the current security state. To distinguish between a mode in Secure state and the equivalent mode in Non-secure state, the mode name is qualified as Secure or Non-secure. For example, a description of AArch32 operation in EL1 might reference the Secure FIQ mode, or to the Non-secure FIQ mode.

Chapter 4
System Control

Describes the system registers, their structure, operation, and how to use them.

It contains the following sections:

4.1 About system control

The system registers control and provide status information for the functions implemented in the processor.

The main functions of the system registers are:

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

The system registers are accessible in AArch64 and AArch32 Execution states. The execution states are described in *3.2 Armv8-A architecture concepts* on page 3-57.

The system registers accessed in the AArch64 Execution state are described in *4.3 AArch64 register descriptions* on page 4-82.

The system registers accessed in the AArch32 Execution state are described in *4.5 AArch32 register descriptions* on page 4-221.

Some of the system registers can be accessed through the memory-mapped or external debug interfaces.

Bits in the system registers that are described in the Armv7 architecture are redefined in the Armv8-A architecture:

- UNK/SBZP, RAZ/SBZP, and RAZ/WI are redefined as RES0.
- UNK/SBOP and RAO/SBOP are redefined as RES1.

RES0 and RES1 are described in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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

4.2 AArch64 register summary

This section gives a summary of the system registers in the AArch64 Execution state.

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

This section contains the following subsections:

4.2.1 AArch64 identification registers

The following table shows the identification registers in AArch64 state. Bits[63:32] are reset to `0x00000000` for all 64-bit registers except for ID_AA64PFR0_EL1.

Table 4-1 AArch64 identification registers

Name	Type	Reset	Width	Description
MIDR_EL1	RO	`0x411FD090`	32	*4.3.1 Main ID Register, EL1* on page 4-83
MPIDR_EL1	RO	-[b]	64	*4.3.2 Multiprocessor Affinity Register* on page 4-84
REVIDR_EL1	RO	`0x00000000`	32	*4.3.3 Revision ID Register* on page 4-86
ID_PFR0_EL1	RO	`0x00010131`	32	*4.3.4 AArch32 Processor Feature Register 0* on page 4-87
ID_PFR1_EL1	RO	`0x10011011`[c]	32	*4.3.5 AArch32 Processor Feature Register 1* on page 4-88

[b] The reset value depends on the primary inputs, **CLUSTERIDAFF1** and **CLUSTERIDAFF2**, and the number of cores that the device implements.
[c] Bits [31:28] are `0x1` if the GIC CPU interface is enabled, and `0x0` otherwise.

Table 4-1 AArch64 identification registers (continued)

Name	Type	Reset	Width	Description
ID_PFR2_EL1	RO	`0x00000001`	32	*4.3.6 AArch32 Processor Feature Register 2* on page 4-89
ID_DFR0_EL1	RO	`0x03010066`	32	*4.3.7 AArch32 Debug Feature Register 0* on page 4-90
ID_AFR0_EL1	RO	`0x00000000`	32	*4.3.8 AArch32 Auxiliary Feature Register 0* on page 4-91
ID_MMFR0_EL1	RO	`0x10201105`	32	*4.3.9 AArch32 Memory Model Feature Register 0* on page 4-92
ID_MMFR1_EL1	RO	`0x40000000`	32	*4.3.10 AArch32 Memory Model Feature Register 1* on page 4-93
ID_MMFR2_EL1	RO	`0x01260000`	32	*4.3.11 AArch32 Memory Model Feature Register 2* on page 4-94
ID_MMFR3_EL1	RO	`0x02102211`	32	*4.3.12 AArch32 Memory Model Feature Register 3* on page 4-96
ID_ISAR0_EL1	RO	`0x02101110`	32	*4.3.13 AArch32 Instruction Set Attribute Register 0* on page 4-98
ID_ISAR1_EL1	RO	`0x13112111`	32	*4.3.14 AArch32 Instruction Set Attribute Register 1* on page 4-99
ID_ISAR2_EL1	RO	`0x21232042`	32	*4.3.15 AArch32 Instruction Set Attribute Register 2* on page 4-100
ID_ISAR3_EL1	RO	`0x01112131`	32	*4.3.16 AArch32 Instruction Set Attribute Register 3* on page 4-102
ID_ISAR4_EL1	RO	`0x00011142`	32	*4.3.17 AArch32 Instruction Set Attribute Register 4* on page 4-104
ID_ISAR5_EL1	RO	`0x00011121` or `0x00010001`[d]	32	*4.3.18 AArch32 Instruction Set Attribute Register 5* on page 4-105

[d] The value is `0x00011121` if the Cryptographic Extension is implemented and enabled. The value is `0x00010001` if the Cryptographic Extension is not implemented or enabled.

Table 4-1 AArch64 identification registers (continued)

Name	Type	Reset	Width	Description
ID_MMFR4_EL1	RO	`0x00000000`	32	*4.3.19 AArch32 Memory Model Feature Register 4* on page 4-106
ID_AA64PFR0_EL1	RO	`0x11000000_01002222` [e]	64	*4.3.20 AArch64 Processor Feature Register 0* on page 4-107
ID_AA64PFR1_EL1	RO	`0x00000000`	64	*4.3.21 AArch64 Processor Feature Register 1* on page 4-109
ID_AA64DFR0_EL1	RO	`0x10305106`	64	*4.3.22 AArch64 Debug Feature Register 0, EL1* on page 4-109
ID_AA64DFR1_EL1	RO	`0x00000000`	64	*4.3.23 AArch64 Debug Feature Register 1* on page 4-111
ID_AA64AFR0_EL1	RO	`0x00000000`	64	*4.3.24 AArch64 Auxiliary Feature Register 0* on page 4-111
ID_AA64AFR1_EL1	RO	`0x00000000`	64	*4.3.25 AArch64 Auxiliary Feature Register 1* on page 4-111
ID_AA64ISAR0_EL1	RO	`0x00011120` or `0x00010000`[f]	64	*4.3.26 AArch64 Instruction Set Attribute Register 0, EL1* on page 4-111
ID_AA64ISAR1_EL1	RO	`0x00000000`	64	*4.3.27 AArch64 Instruction Set Attribute Register 1, EL1* on page 4-112
ID_AA64MMFR0_EL1	RO	`0x00101122` The supported Physical Address Range is 40-bit.	64	*4.3.28 AArch64 Memory Model Feature Register 0, EL1* on page 4-112
ID_AA64MMFR1_EL1	RO	`0x00000000`	64	*4.3.29 AArch64 Memory Model Feature Register 1* on page 4-114
CCSIDR_EL1	RO	UNK[g]	32	*4.3.30 Cache Size ID Register* on page 4-114
CLIDR_EL1	RO	`0x0A200023`	64	*4.3.31 Cache Level ID Register* on page 4-115
AIDR_EL1	RO	`0x00000000`	32	*4.3.32 Auxiliary ID Register* on page 4-117

e Bits [63:32] are `11000000`. Bits [31:0] are `01002222`
f The value is `0x00011120` if the Cryptographic Extension is implemented and enabled. The value is `0x00010000` if the Cryptographic Extension is not implemented or enabled.
g The reset value depends on the implementation. See the register description for details.

Table 4-1 AArch64 identification registers (continued)

Name	Type	Reset	Width	Description
CSSELR_EL1	RW	UNK	32	*4.3.33 Cache Size Selection Register* on page 4-117
CTR_EL0	RO	`0x84448004`	32	*4.3.34 Cache Type Register* on page 4-118
DCZID_EL0	RO	`0x00000004`	32	*4.3.35 Data Cache Zero ID Register* on page 4-119
VPIDR_EL2	RW	`0x411FD090`	32	*4.3.36 Virtualization Processor ID Register* on page 4-120
VMPIDR_EL2	RO	-[h]	64	*4.3.37 Virtualization Multiprocessor ID Register* on page 4-121

4.2.2 AArch64 exception handling registers

The following table shows the fault handling registers in AArch64 state.

Table 4-2 AArch64 exception handling registers

Name	Type	Reset	Width	Description
AFSR0_EL1	RW	`0x00000000`	32	*4.3.63 Auxiliary Fault Status Register 0, EL1, EL2 and EL3* on page 4-164
AFSR1_EL1	RW	`0x00000000`	32	*4.3.64 Auxiliary Fault Status Register 1, EL1, EL2 and EL3* on page 4-164
ESR_EL1	RW	UNK	32	*4.3.65 Exception Syndrome Register, EL1* on page 4-164
IFSR32_EL2	RW	UNK	32	*4.3.66 Instruction Fault Status Register, EL2* on page 4-166
AFSR0_EL2	RW	`0x00000000`	32	*4.3.63 Auxiliary Fault Status Register 0, EL1, EL2 and EL3* on page 4-164
AFSR1_EL2	RW	`0x00000000`	32	*4.3.64 Auxiliary Fault Status Register 1, EL1, EL2 and EL3* on page 4-164
ESR_EL2	RW	UNK	32	*4.3.67 Exception Syndrome Register, EL2* on page 4-169
AFSR0_EL3	RW	`0x00000000`	32	*4.3.63 Auxiliary Fault Status Register 0, EL1, EL2 and EL3* on page 4-164
AFSR1_EL3	RW	`0x00000000`	32	*4.3.64 Auxiliary Fault Status Register 1, EL1, EL2 and EL3* on page 4-164
ESR_EL3	RW	UNK	32	*4.3.68 Exception Syndrome Register, EL3* on page 4-170
FAR_EL1	RW	UNK	64	*4.3.69 Fault Address Register, EL1* on page 4-171
FAR_EL2	RW	UNK	64	*4.3.70 Fault Address Register, EL2* on page 4-172
HPFAR_EL2	RW	UNK	64	*4.3.71 Hypervisor IPA Fault Address Register, EL2* on page 4-173
FAR_EL3	RW	UNK	64	*4.3.74 Fault Address Register, EL3* on page 4-178
VBAR_EL1	RW	UNK	64	*4.3.79 Vector Base Address Register, EL1* on page 4-184
ISR_EL1	RO	UNK	32	*4.3.84 Interrupt Status Register* on page 4-188
VBAR_EL2	RW	UNK	64	*4.3.80 Vector Base Address Register, EL2* on page 4-185
VBAR_EL3	RW	UNK	64	*4.3.81 Vector Base Address Register, EL3* on page 4-186

[h] The reset value is the value of the Multiprocessor Affinity Register.

4.2.3 AArch64 virtual memory control registers

The following table shows the virtual memory control registers in AArch64 state.

Table 4-3 AArch64 virtual memory control registers

Name	Type	Reset	Width	Description
SCTLR_EL1	RW	`0x00C50838`[i]	32	*4.3.38 System Control Register, EL1* on page 4-122
SCTLR_EL2	RW	UNK	32	*4.3.43 System Control Register, EL2* on page 4-129
SCTLR_EL3	RW	`0x00C50838`[i]	32	*4.3.49 System Control Register, EL3* on page 4-143
TTBR0_EL1	RW	UNK	64	*4.3.52 Translation Table Base Register 0, EL1* on page 4-148
TTBR1_EL1	RW	UNK	64	*4.3.53 Translation Table Base Register 1* on page 4-149
TCR_EL1	RW	UNK	64	*4.3.56 Translation Control Register, EL1* on page 4-153
TTBR0_EL2	RW	UNK	64	Translation Table Base Address Register 0, EL2[j]
TCR_EL2	RW	UNK	32	*4.3.57 Translation Control Register, EL2* on page 4-156
VTTBR_EL2	RW	UNK	64	Virtualization Translation Table Base Address Register, EL2[j]
VTCR_EL2	RW	UNK	32	*4.3.58 Virtualization Translation Control Register, EL2* on page 4-158
TTBR0_EL3	RW	UNK	64	*4.3.60 Translation Table Base Register 0, EL3* on page 4-161
TCR_EL3	RW	`0x00000000`	32	*4.3.61 Translation Control Register, EL3* on page 4-162
MAIR_EL1	RW	UNK	64	*4.3.76 Memory Attribute Indirection Register, EL1* on page 4-181
AMAIR_EL1	RW	`0x00000000`	64	*4.3.62 Auxiliary Memory Attribute Indirection Register, EL1, EL2 and EL3* on page 4-164
MAIR_EL2	RW	UNK	64	*4.3.77 Memory Attribute Indirection Register, EL2* on page 4-183
AMAIR_EL2	RW	`0x00000000`	64	*4.3.62 Auxiliary Memory Attribute Indirection Register, EL1, EL2 and EL3* on page 4-164
MAIR_EL3	RW	UNK	64	*4.3.78 Memory Attribute Indirection Register, EL3* on page 4-184
AMAIR_EL3	RW	`0x00000000`	64	*4.3.62 Auxiliary Memory Attribute Indirection Register, EL1, EL2 and EL3* on page 4-164
CONTEXTIDR_EL1	RW	UNK	32	Context ID Register, EL1[j]

4.2.4 AArch64 other system control registers

The following table shows the other system control registers in AArch64 state.

[i] The reset value depends on primary inputs **VINITHI, CFGTE** and **CFGEND**. The information in this table assumes these signals are LOW.
[j] See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table 4-4 AArch64 other system control registers

Name	Type	Reset	Width	Description
ACTLR_EL1	RW	`0x00000000`	32	*4.3.39 Auxiliary Control Register, EL1* on page 4-125
CPACR_EL1	RW	`0x00000000`	32	*4.3.42 Architectural Feature Access Control Register* on page 4-128
ACTLR_EL2	RW	`0x00000000`	32	*4.3.40 Auxiliary Control Register, EL2* on page 4-126
ACTLR_EL3	RW	`0x00000000`	32	*4.3.41 Auxiliary Control Register, EL3* on page 4-127

4.2.5 AArch64 cache maintenance operations

The following table shows the System instructions for cache and maintenance operations in AArch64 state.

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

Table 4-5 AArch64 cache maintenance operations

Name	Description
`IC IALLUIS`	Instruction cache invalidate all to PoU[k] Inner Shareable
`IC IALLU`	Instruction cache invalidate all to PoU
`IC IVAU`	Instruction cache invalidate by *virtual address* (VA) to PoU
`DC IVAC`	Data cache invalidate by VA to PoC[l]
`DC ISW`	Data cache invalidate by set/way
`DC CSW`	Data cache clean by set/way
`DC CISW`	Data cache clean and invalidate by set/way
`DC ZVA`	Data cache zero by VA
`DC CVAC`	Data cache clean by VA to PoC
`DC CVAU`	Data cache clean by VA to PoU
`DC CIVAC`	Data cache clean and invalidate by VA to PoC

4.2.6 AArch64 TLB maintenance operations

The following table shows the System instructions for TLB maintenance operations in AArch64 state.

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

[k] PoU = Point of Unification. PoU is set by the **BROADCASTINNER** signal and can be in the L1 data cache or outside of the processor, in which case PoU is dependent on the external memory system.
[l] PoC = Point of Coherence. The PoC is always outside of the processor and is dependent on the external memory system.

Table 4-6 AArch64 TLB maintenance operations

Name	Description
`TLBI VMALLE1IS`	Invalidate all stage 1 translations used at EL1 with the current *virtual machine identifier* (VMID) in the Inner Shareable
`TLBI VAE1IS`	Invalidate translation used at EL1 for the specified VA and *Address Space Identifier* (ASID) and the current VMID, Inner Shareable
`TLBI ASIDE1IS`	Invalidate all translations used at EL1 with the current VMID and the supplied ASID, Inner Shareable
`TLBI VAAE1IS`	Invalidate all translations used at EL1 for the specified address and current VMID and for all ASID values, Inner Shareable
`TLBI VALE1IS`	Invalidate all entries from the last level of stage 1 translation table walk used at EL1 with the supplied ASID and current VMID, Inner Shareable
`TLBI VAALE1IS`	Invalidate all entries from the last level of stage 1 translation table walk used at EL1 for the specified address and current VMID and for all ASID values, Inner Shareable
`TLBI VMALLE1`	Invalidate all stage 1 translations used at EL1 with the current VMID
`TLBI VAE1`	Invalidate translation used at EL1 for the specified VA and ASID and the current VMID
`TLBI ASIDE1`	Invalidate all translations used at EL1 with the current VMID and the supplied ASID
`TLBI VAAE1`	Invalidate all translations used at EL1 for the specified address and current VMID and for all ASID values
`TLBI VALE1`	Invalidate all entries from the last level of stage 1 translation table walk used at EL1 with the supplied ASID and current VMID
`TLBI VAALE1`	Invalidate all entries from the last level of stage 1 translation table walk used at EL1 for the specified address and current VMID and for all ASID values

The Virtualization registers include additional TLB operations for use in Hyp mode. For more information, see *4.2.13 AArch64 EL2 TLB maintenance operations* on page 4-77.

4.2.7 AArch64 address translation operations

The following table shows the address translation register in AArch64 state.

Table 4-7 AArch64 address translation register

Name	Type	Reset	Width	Description
PAR_EL1	RW	UNK	64	*4.3.75 Physical Address Register, EL1* on page 4-179

The following table shows the System instructions for address translation operations in AArch64 state. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

Table 4-8 AArch64 address translation operations

Name	Description
`AT S1E1R`	Stage 1 current state EL1 read
`AT S1E1W`	Stage 1 current state EL1 write
`AT S1E0R`	Stage 1 current state unprivileged read
`AT S1E0W`	Stage 1 current state unprivileged write

Table 4-8 AArch64 address translation operations (continued)

Name	Description
`AT S1E2R`	Stage 1 Hyp mode read
`AT S1E2W`	Stage 1 Hyp mode write
`AT S12E1R`	Stages 1 and 2 Non-secure EL1 read
`AT S12E1W`	Stages 1 and 2 Non-secure EL1 write
`AT S12E0R`	Stages 1 and 2 Non-secure unprivileged read
`AT S12E0W`	Stages 1 and 2 Non-secure unprivileged write
`AT S1E3R`	Stage 1 current state EL3 read
`AT S1E3W`	Stage 1 current state EL3 write

4.2.8 AArch64 miscellaneous operations

The following table shows the miscellaneous operations in AArch64 state.

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

Table 4-9 AArch64 miscellaneous System operations

Name	Type	Reset	Width	Description
TPIDR_EL0	RW	UNK	64	Thread Pointer / ID Register, EL0
TPIDR_EL1	RW	UNK	64	Thread Pointer / ID Register, EL1
TPIDRRO_EL0	RW [m]	UNK	64	Thread Pointer / ID Register, Read-Only, EL0
TPIDR_EL2	RW	UNK	64	Thread Pointer / ID Register, EL2
TPIDR_EL3	RW	UNK	64	Thread Pointer / ID Register, EL3

4.2.9 AArch64 performance monitor registers

The following table shows the performance monitor registers in AArch64 state.

Table 4-10 AArch64 performance monitor registers

Name	Type	Reset	Width	Description
PMCR_EL0	RW	`0x41043000`	32	Performance Monitors Control Register
PMCNTENSET_EL0	RW	UNK	32	Performance Monitors Count Enable Set Register [n]
PMCNTENCLR_EL0	RW	UNK	32	Performance Monitors Count Enable Clear Register [n]
PMOVSCLR_EL0	RW	UNK	32	Performance Monitors Overflow Flag Status Clear Register [n]
PMSWINC_EL0	WO	-	32	Performance Monitors Software Increment Register [n]

[m] RO at EL0.

Table 4-10 AArch64 performance monitor registers (continued)

Name	Type	Reset	Width	Description
PMSELR_EL0	RW	UNK	32	Performance Monitors Event Counter Selection Register [n]
PMCEID0_EL0	RO	`0x7BFF7F3F`	32	*11.4.2 Performance Monitors Common Event Identification Register 0* on page 11-450
PMCEID1_EL0	RO	`0x00000000`	32	*11.4.3 Performance Monitors Common Event Identification Register 1* on page 11-453[n]
PMCCNTR_EL0	RW	UNK	64	Performance Monitors Cycle Counter[n]
PMXEVTYPER_EL0	RW	UNK	32	Performance Monitors Selected Event Type and Filter Register [n]
PMXEVCNTR_EL0	RW	UNK	32	Performance Monitors Selected Event Counter Register [n]
PMUSERENR_EL0	RW	`0x00000000`	32	Performance Monitors User Enable Register [n]
PMINTENSET_EL1	RW	UNK	32	Performance Monitors Interrupt Enable Set Register [n]
PMINTENCLR_EL1	RW	UNK	32	Performance Monitors Interrupt Enable Clear Register [n]
PMOVSSET_EL0	RW	UNK	32	Performance Monitors Overflow Flag Status Set Register [n]
PMEVCNTR0_EL0	RW	UNK	32	Performance Monitor Event Count Registers
PMEVCNTR1_EL0	RW	UNK	32	
PMEVCNTR2_EL0	RW	UNK	32	
PMEVCNTR3_EL0	RW	UNK	32	
PMEVCNTR4_EL0	RW	UNK	32	
PMEVCNTR5_EL0	RW	UNK	32	
PMEVTYPER0_EL0	RW	UNK	32	Performance Monitor Event Type Registers
PMEVTYPER1_EL0	RW	UNK	32	
PMEVTYPER2_EL0	RW	UNK	32	
PMEVTYPER3_EL0	RW	UNK	32	
PMEVTYPER4_EL0	RW	UNK	32	
PMEVTYPER5_EL0	RW	UNK	32	
PMCCFILTR_EL0	RW	`0x00000000`	32	Performance Monitors Cycle Count Filter Register [n]

4.2.10 AArch64 reset registers

The following table shows the reset registers in AArch64 state.

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

Table 4-11 AArch64 reset management registers

Name	Type	Reset	Width	Description
RVBAR_EL3	RO	-[o]	64	*4.3.82 Reset Vector Base Address Register, EL3* on page 4-186
RMR_EL3	RW	`0x00000001`[p]	32	*4.3.83 Reset Management Register* on page 4-187

4.2.11 AArch64 Secure registers

The following table shows the Secure registers in AArch64 state.

Table 4-12 AArch64 Secure registers

Name	Type	Reset	Width	Description
SCR_EL3	RW	`0x00000000`	32	*4.3.50 Secure Configuration Register* on page 4-144
SDER32_EL3	RW	`0x00000000`	32	*4.3.51 Secure Debug Enable Register* on page 4-147
CPTR_EL3	RW	`0x00000000`	32	*4.3.54 Architectural Feature Trap Register, EL3* on page 4-150
MDCR_EL3	RW	UNK	32	*4.3.55 Monitor Debug Configuration Register, EL3* on page 4-151
AFSR0_EL3	RW	`0x00000000`	32	*4.3.63 Auxiliary Fault Status Register 0, EL1, EL2 and EL3* on page 4-164
AFSR1_EL3	RW	`0x00000000`	32	*4.3.64 Auxiliary Fault Status Register 1, EL1, EL2 and EL3* on page 4-164
VBAR_EL3	RW	UNK	64	*4.3.81 Vector Base Address Register, EL3* on page 4-186

4.2.12 AArch64 virtualization registers

The following table shows the virtualization registers in AArch64 state.

Table 4-13 AArch64 virtualization registers

Name	Type	Reset	Width	Description
VPIDR_EL2	RW	`0x411FD090`	32	*4.3.36 Virtualization Processor ID Register* on page 4-120
VMPIDR_EL2	RW	-[q]	64	*4.3.37 Virtualization Multiprocessor ID Register* on page 4-121
SCTLR_EL2	RW	UNK	32	*4.3.43 System Control Register, EL2* on page 4-129
ACTLR_EL2	RW	`0x00000000`	32	*4.3.40 Auxiliary Control Register, EL2* on page 4-126
HCR_EL2	RW	`0x00000002`	64	*4.3.44 Hypervisor Configuration Register* on page 4-131
MDCR_EL2	RW	`0x00000006`	32	*4.3.45 Hyp Debug Control Register* on page 4-136
CPTR_EL2	RW	`0x000033FF`	32	*4.3.46 Architectural Feature Trap Register, EL2* on page 4-139
HSTR_EL2	RW	`0x00000000`	32	*4.3.47 Hyp System Trap Register* on page 4-140
HACR_EL2	RW	`0x00000000`	32	*4.3.48 Hyp Auxiliary Configuration Register* on page 4-142
TTBR0_EL2	RW	UNK	64	Translation Table Base Address Register 0, EL2[r]
TCR_EL2	RW	UNK	32	*4.3.57 Translation Control Register, EL2* on page 4-156

o The reset value depends on the **RVBARADDR** signal.
p This value depends on the AA64NAA32 pin value, which is Asserted in AArch64 state.
q The reset value is the value of the Multiprocessor Affinity Register.

Table 4-13 AArch64 virtualization registers (continued)

Name	Type	Reset	Width	Description
VTTBR_EL2	RW	UNK	64	Virtualization Translation Table Base Address Register, EL2 [r]
VTCR_EL2	RW	UNK	32	*4.3.58 Virtualization Translation Control Register, EL2* on page 4-158
DACR32_EL2	RW	UNK	32	*4.3.59 Domain Access Control Register* on page 4-160
AFSR0_EL2	RW	`0x00000000`	32	*4.3.63 Auxiliary Fault Status Register 0, EL1, EL2 and EL3* on page 4-164
AFSR1_EL2	RW	`0x00000000`	32	*4.3.64 Auxiliary Fault Status Register 1, EL1, EL2 and EL3* on page 4-164
ESR_EL2	RW	UNK	32	*4.3.67 Exception Syndrome Register, EL2* on page 4-169
FAR_EL2	RW	UNK	64	*4.3.70 Fault Address Register, EL2* on page 4-172
HPFAR_EL2	RW	UNK	64	*4.3.71 Hypervisor IPA Fault Address Register, EL2* on page 4-173
MAIR_EL2	RW	UNK	64	*4.3.77 Memory Attribute Indirection Register, EL2* on page 4-183
AMAIR_EL2	RW	`0x00000000`	64	*4.3.62 Auxiliary Memory Attribute Indirection Register, EL1, EL2 and EL3* on page 4-164
VBAR_EL2	RW	UNK	64	*4.3.80 Vector Base Address Register, EL2* on page 4-185

4.2.13 AArch64 EL2 TLB maintenance operations

The following table shows the System instructions for TLB maintenance operations added in AArch64 state.

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

Table 4-14 AArch64 TLB maintenance operations

Name	Description
`TLBI IPAS2E1IS`	Invalidate stage 2 only translations used at EL1 for the specified IPA for the current VMID, Inner Shareable
`TLBI IPAS2LE1IS`	Invalidate entries from the last level of stage 2 only translation used at EL1 for the specified IPA for the current VMID, Inner Shareable
`TLBI ALLE2IS`	Invalidate all stage 1 translations used at EL2, Inner Shareable
`TLBI VAE2IS`	Invalidate translation used at EL2 for the specified VA and ASID and the current VMID, Inner Shareable
`TLBI ALLE1IS`	Invalidate all stage 1 translations used at EL1, Inner Shareable
`TLBI VALE2IS`	Invalidate all entries from the last level of stage 1 translation table walk used at EL2 with the supplied ASID and current VMID, Inner Shareable
`TLBI VMALLS12E1IS`	Invalidate all stage 1 and 2 translations used at EL1 with the current VMID, Inner Shareable
`TLBI IPAS2E1`	Invalidate stage 2 only translations used at EL1 for the specified IPA for the current VMID
`TLBI IPAS2LE1`	Invalidate entries from the last level of stage 2 only translation used at EL1 for the specified IPA for the current VMID
`TLBI ALLE2`	Invalidate all stage 1 translations used at EL2

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

Table 4-14 AArch64 TLB maintenance operations (continued)

Name	Description
`TLBI VAE2`	Invalidate translation used at EL2 for the specified VA and ASID and the current VMID
`TLBI ALLE1`	Invalidate all stage 1 translations used at EL1
`TLBI VALE2`	Invalidate all entries from the last level of stage 1 translation table walk used at EL2 with the supplied ASID and current VMID
`TLBI VMALLS12E1`	Invalidate all stage 1 and 2 translations used at EL1 with the current VMID
`TLBI ALLE3IS`	Invalidate all stage 1 translations used at EL3, Inner Shareable
`TLBI VAE3IS`	Invalidate translation used at EL3 for the specified VA and ASID and the current VMID, Inner Shareable
`TLBI VALE3IS`	Invalidate all entries from the last level of stage 1 translation table walk used at EL3 with the supplied ASID and current VMID, Inner Shareable
`TLBI ALLE3`	Invalidate all stage 1 translations used at EL3
`TLBI VAE3`	Invalidate translation used at EL3 for the specified VA and ASID and the current VMID
`TLBI VALE3`	Invalidate all entries from the last level of stage 1 translation table walk used at EL3 with the supplied ASID and current VMID

4.2.14 AArch64 GIC system registers

The following table shows the GIC system registers in AArch64 state.

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

Table 4-15 GIC system registers

Name	Type	Reset	Width	Description
ICC_AP0R0_EL1	RW	`0x00000000`	32	Active Priorities 0 Register 0
ICC_AP1R0_EL1	RW	`0x00000000`	32	Active Priorities 1 Register 0
ICC_ASGI1R_EL1	WO	-	64	Alternate SGI Generation Register 1
ICC_BPR0_EL1	RW	`0x00000002`	32	Binary Point Register 0
ICC_BPR1_EL1	RW	`0x00000003`[s]	32	Binary Point Register 1
ICC_CTLR_EL1	RW	`0x00000400`	32	Interrupt Control Register for EL1
ICC_CTLR_EL3	RW	`0x00000400`	32	Interrupt Control Register for EL3
ICC_DIR_EL1	WO	-	32	Deactivate Interrupt Register
ICC_EOIR0_EL1	WO	-	32	End Of Interrupt Register 0
ICC_EOIR1_EL1	WO	-	32	End Of Interrupt Register 1
ICC_HPPIR0_EL1	RO	-	32	Highest Priority Pending Interrupt Register 0
ICC_HPPIR1_EL1	RO	-	32	Highest Priority Pending Interrupt Register 1
ICC_IAR0_EL1	RO	-	32	Interrupt Acknowledge Register 0
ICC_IAR1_EL1	RO	-	32	Interrupt Acknowledge Register 1
ICC_IGRPEN0_EL1	RW	`0x00000000`	32	Interrupt Group Enable Register 0

[s] This is the reset value in Non-secure states. In Secure states, the reset value is `0x00000002`.

Table 4-15 GIC system registers (continued)

Name	Type	Reset	Width	Description
ICC_IGRPEN1_EL1	RW	`0x00000000`	32	Interrupt Group Enable Register 1
ICC_IGRPEN1_EL3	RW	`0x00000000`	32	Interrupt Group Enable Register 1 for EL3
ICC_PMR_EL1	RW	`0x00000000`	32	Priority Mask Register
ICC_RPR_EL1	RO	-	32	Running Priority Register
ICC_SGI0R_EL1	WO	-	64	SGI Generation Register 0
ICC_SGI1R_EL1	WO	-	64	SGI Generation Register 1
ICC_SRE_EL1	RW	`0x00000000`	32	System Register Enable Register for EL1
ICC_SRE_EL2	RW	`0x00000000`	32	System Register Enable Register for EL2
ICC_SRE_EL3	RW	`0x00000000`	32	System Register Enable Register for EL3
ICH_AP0R0_EL2	RW	`0x00000000`	32	Interrupt Controller Hyp Active Priorities Register (0,0)
ICH_AP1R0_EL2	RW	`0x00000000`	32	Interrupt Controller Hyp Active Priorities Register (1,0)
ICH_EISR_EL2	RO	`0x00000000`	32	Interrupt Controller End of Interrupt Status Register
ICH_ELRSR_EL2	RO	`0x0000000F`	32	Interrupt Controller Empty List Register Status Register
ICH_HCR_EL2	RW	`0x00000000`	32	Interrupt Controller Hyp Control Register
ICH_LR0_EL2	RW	`0x00000000 00000000`	64	Interrupt Controller List Register 0
ICH_LR1_EL2	RW	`0x00000000 00000000`	64	Interrupt Controller List Register 1
ICH_LR2_EL2	RW	`0x00000000 00000000`	64	Interrupt Controller List Register 2
ICH_LR3_EL2	RW	`0x00000000 00000000`	64	Interrupt Controller List Register 3
ICH_MISR_EL2	RO	`0x00000000`	32	Interrupt Controller Maintenance Interrupt State Register
ICH_VMCR_EL2	RW	`0x004C0000`	32	Interrupt Controller Virtual Machine Control Register
ICH_VTR_EL2	RO	`0x90000003`	32	Interrupt Controller VGIC Type Register

4.2.15 AArch64 Generic Timer registers

See *9.2 Generic Timer functional description* on page 9-388 for information on the Generic Timer registers.

4.2.16 AArch64 thread registers

The following table shows the thread registers in AArch64 state.

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

Table 4-16 AArch64 miscellaneous system control operations

Name	Type	Reset	Width	Description
TPIDR_EL0	RW	UNK	64	Thread Pointer/ID Register, EL0
TPIDR_EL1	RW	UNK	64	Thread Pointer/ID Register, EL1
TPIDRRO_EL0	RW	UNK	64	Thread Pointer/ID Register, Read-Only, EL0

Table 4-16 AArch64 miscellaneous system control operations (continued)

Name	Type	Reset	Width	Description
TPIDR_EL2	RW	UNK	64	Thread Pointer/ID Register, EL2
TPIDR_EL3	RW	UNK	64	Thread Pointer/ID Register, EL3

4.2.17 AArch64 implementation defined registers

The following table shows the IMPLEMENTATION DEFINED registers in AArch64 state.

These registers provide test features and any required configuration options specific to the Cortex-A73 processor. If a register is not indicated as mapped to an AArch32 64-bit register, bits[63:32] are `0x00000000`.

Table 4-17 AArch64 IMPLEMENTATION DEFINED registers

Name	Type	Reset	Width	Description
L2CTLR_EL1	RW	-[t]	32	*4.3.72 L2 Control Register* on page 4-174
L2ECTLR_EL1	RW	`0x00000000`	32	*4.3.73 L2 Extended Control Register* on page 4-177
ECTLR_EL1[u]	RW	`0x00000000 00000580`	64	*4.3.85 Extended Control Register, EL1* on page 4-189
L2MERRSR_EL1[u]	RW	`0x00000000`	64	*4.3.86 L2 Memory Error Syndrome Register* on page 4-191
CBAR_EL1	RO	-[v]	64	*4.3.87 Configuration Base Address Register, EL1* on page 4-193
CDBGDR0_EL3	RO	UNK	32	Direct access to internal memory, Data Register 0, see *6.7 Direct access to internal memory* on page 6-351.
CDBGDR1_EL3	RO	UNK	32	Direct access to internal memory, Data Register 1, see *6.7 Direct access to internal memory* on page 6-351.
CDBGDR2_EL3	RO	UNK	32	Direct access to internal memory, Data Register 2, see *6.7 Direct access to internal memory* on page 6-351.
CDBGDR3_EL3	RO	UNK	32	Direct access to internal memory, Data Register 3, see *6.7 Direct access to internal memory* on page 6-351.

4.2.18 AArch64 implementation defined operations

The following table shows the implementation defined operations in AArch64 state.

Table 4-18 AArch64 implementation defined maintenance operations

Name	Type	Reset	Width	Description
DC CIALL	WO	UNK	32	Clean/invalidate all data or unified caches, see *4.3.88 DC CIALL Clean Invalidate All* on page 4-194.
CDBGDCT_EL3	WO	UNK	32	Data Cache Tag Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351.

[t] The reset value depends on the processor implementation and the state of the **L2RSTDISABLE** signal.
[u] Mapped to a 64-bit AArch32 register.
[v] The reset value depends on the **PERIPHBASE** signal.

Table 4-18 AArch64 implementation defined maintenance operations (continued)

Name	Type	Reset	Width	Description
CDBGICT_EL3	WO	UNK	32	Instruction Cache Tag Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351.
CDBGDCD_EL3	WO	UNK	32	Data Cache Data Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351.
CDBGICD_EL3	WO	UNK	32	Instruction Cache Data Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351.
CDBGTD_EL3	WO	UNK	32	TLB Data Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351.

4.3 AArch64 register descriptions

This section describes all the system registers, in register number order, when the system is in the AArch64 Execution state.

See *4.2.1 AArch64 identification registers* on page 4-67, which provides cross-references to individual registers.

This section contains the following subsections:

4.3.1 Main ID Register, EL1

The MIDR_EL1 characteristics are:

Purpose Provides identification information for the processor, including an implementer code for the device and a device ID number.

Usage constraints 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

Configurations The MIDR_EL1 is:

- Architecturally mapped to the AArch32 MIDR register. See *4.5.1 Main ID Register* on page 4-222.
- Architecturally mapped to external MIDR_EL1 register.

Attributes MIDR_EL1 is a 32-bit register.

The following figure shows the MIDR_EL1 bit assignments.

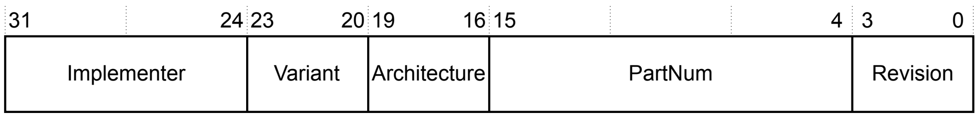

Figure 4-1 MIDR_EL1 bit assignments

The following table shows the MIDR_EL1 bit assignments.

Table 4-19 MIDR_EL1 bit assignments

Bits	Name	Function
[31:24]	Implementer	Indicates the implementer code. This value is: `0x41` ASCII character 'A' - implementer is Arm.
[23:20]	Variant	Indicates the variant number of the processor. 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: `0x1` r1p0.
[19:16]	Architecture	Indicates the architecture code. This value is: `0xF` Defined by CPUID scheme.
[15:4]	PartNum	Indicates the primary part number. This value is: `0xD09` Cortex-A73 processor.
[3:0]	Revision	Indicates the minor revision number of the processor. 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: `0x0` r1p0.

To access the MIDR_EL1:

```
MRS <Xt>, MIDR_EL1 ; Read MIDR_EL1 into Xt
```

The following table shows the register access encoding:

Table 4-20 MIDR_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0000	000

The MIDR_EL1 can be accessed through the memory-mapped interface and the external debug interface, offset `0xD00`.

4.3.2 Multiprocessor Affinity Register

The MPIDR_EL1 characteristics are:

Purpose Provides an additional core identification mechanism for scheduling purposes in a cluster system.

Usage constraints 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

Configurations The MPIDR_EL1[31:0] is:

- Architecturally mapped to the AArch32 MPIDR register. See *4.5.2 Multiprocessor Affinity Register* on page 4-223.
- Mapped to external EDDEVAFF0 register.

MPIDR_EL1[63:32] is mapped to external EDDEVAFF1 register.

Attributes MPIDR_EL1 is a 64-bit register.

The following figure shows the MPIDR_EL1 bit assignments.

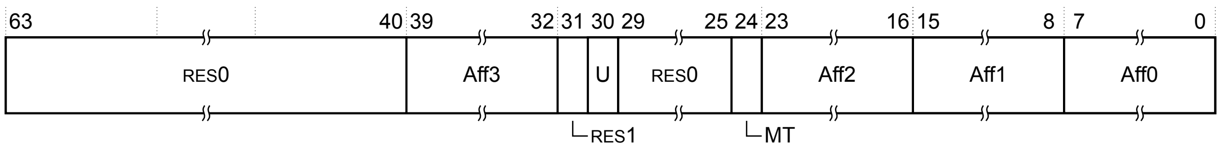

Figure 4-2 MPIDR_EL1 bit assignments

The following table shows the MPIDR_EL1 bit assignments.

Table 4-21 MPIDR_EL1 bit assignments

Bits	Name	Function
[63:40]	-	Reserved, RES0.
[39:32]	Aff3	Affinity level 3. Highest level affinity field. Reserved, RES0.
[31]	-	Reserved, RES1.
[30]	U	Processor is part of a multiprocessor system.
[29:25]	-	Reserved, RES0.
[24]	MT	Indicates whether the lowest level of affinity consists of logical cores that are implemented using a multi-threading type approach. This value is: `0` Performance of cores at the lowest affinity level is largely independent.
[23:16]	Aff2	Affinity level 2. Second highest level affinity field. Indicates the value read in the **CLUSTERIDAFF2** configuration signal.

Table 4-21 MPIDR_EL1 bit assignments (continued)

Bits	Name	Function
[15:8]	Aff1	Affinity level 1. Third highest level affinity field. Indicates the value read in the **CLUSTERIDAFF1** configuration signal.
[7:0]	Aff0	Affinity level 0. Lowest level affinity field. Indicates the core number in the Cortex-A73 processor. The possible values are: `0x0` A cluster with one core only. `0x0`, `0x1` A cluster with two cores. `0x0`, `0x1`, `0x2` A cluster with three cores. `0x0`, `0x1`, `0x2`, `0x3` A cluster with four cores.

To access the MPIDR_EL1:

```
MRS <Xt>, MPIDR_EL1 ; Read MPIDR_EL1 into Xt
```

Register access is encoded as follows:

Table 4-22 MPIDR_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0000	101

The EDDEVAFF0 and EDDEVAFF1 can be accessed through the external debug interface, offsets `0xFA8` and `0xFAC` respectively.

4.3.3 Revision ID Register

The REVIDR_EL1 characteristics are:

Purpose Provides implementation-specific minor revision information that can be interpreted only in conjunction with the Main ID Register.

Usage constraints 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

Configurations REVIDR_EL1 is architecturally mapped to AArch32 register REVIDR. See *4.5.3 Revision ID Register* on page 4-225.

Attributes REVIDR_EL1 is a 32-bit register.

The following figure shows the REVIDR_EL1 bit assignments.

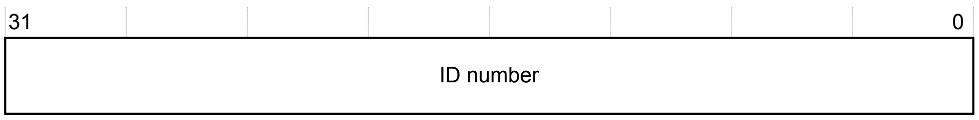

Figure 4-3 REVIDR_EL1 bit assignments

The following table shows the REVIDR_EL1 bit assignments.

Table 4-23 REVIDR_EL1 bit assignments

Bits	Name	Function
[31:0]	ID number	Implementation-specific revision information. The reset value is determined by the specific Cortex-A73 processor implementation. `0x00000000` Revision code is zero.

To access the REVIDR_EL1:

```
MRS <Xt>, REVIDR_EL1 ; Read REVIDR_EL1 into Xt
```

Register access is encoded as follows:

Table 4-24 REVIDR_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0000	110

4.3.4 AArch32 Processor Feature Register 0

The ID_PFR0_EL1 characteristics are:

Purpose Gives top-level information about the instruction sets supported by the processor in AArch32.

Usage constraints 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

Configurations ID_PFR0_EL1 is architecturally mapped to AArch32 register ID_PFR0. See *4.5.6 Processor Feature Register 0* on page 4-226.

Attributes ID_PFR0_EL1 is a 32-bit register.

The following figure shows the ID_PFR0_EL1 bit assignments.

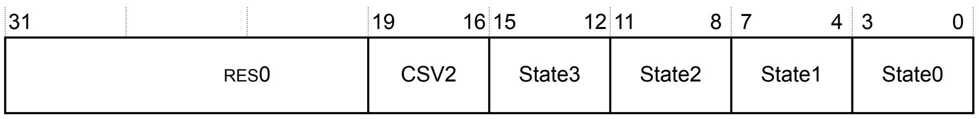

Figure 4-4 ID_PFR0_EL1 bit assignments

The following table shows the ID_PFR0_EL1 bit assignments.

Table 4-25 ID_PFR0_EL1 bit assignments

Bits	Name	Function
[31:20]	-	Reserved, RES0.
[19:16]	CSV2	`0x1` Branch targets trained in one hardware described context cannot affect speculative execution in a different hardware described context.
[15:12]	State3	Indicates support for *Thumb Execution Environment* (T32EE) instruction set. This value is: `0x0` Processor does not support the T32EE instruction set.
[11:8]	State2	Indicates support for Jazelle. This value is: `0x1` Processor supports trivial implementation of Jazelle.
[7:4]	State1	Indicates support for T32 instruction set. This value is: `0x3` Processor supports T32 encoding after the introduction of Thumb-2 technology, and for all 16-bit and 32-bit T32 basic instructions.
[3:0]	State0	Indicates support for A32 instruction set. This value is: `0x1` A32 instruction set implemented.

To access the ID_PFR0_EL1:

```
MRS <Xt>, ID_PFR0_EL1 ; Read ID_PFR0_EL1 into Xt
```

Register access is encoded as follows:

Table 4-26 ID_PFR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0001	000

4.3.5 AArch32 Processor Feature Register 1

The ID_PFR1_EL1 characteristics are:

Purpose Provides information about the programmers model and architecture extensions supported by the processor.

Usage constraints 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

Configurations ID_PFR1_EL1 is architecturally mapped to AArch32 register ID_PFR1. See *4.5.7 Processor Feature Register 1* on page 4-227.

Attributes ID_PFR1_EL1 is a 32-bit register.

The following figure shows the ID_PFR1_EL1 bit assignments.

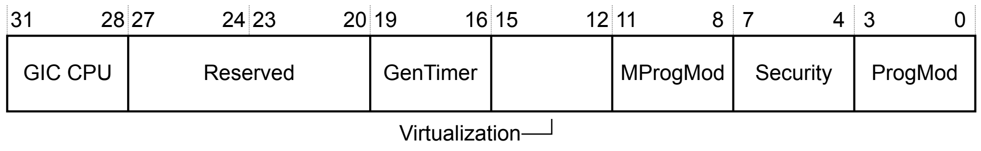

Figure 4-5 ID_PFR1_EL1 bit assignments

The following table shows the ID_PFR1_EL1 bit assignments.

Table 4-27 ID_PFR1_EL1 bit assignments

Bits	Name	Function
[31:28]	GIC CPU	GIC CPU support: `0x0` GIC CPU interface is disabled, **GICCDISABLE** is HIGH. `0x1` GIC CPU interface is enabled.
[27:20]	-	Reserved, RES0.
[19:16]	GenTimer	Generic Timer support: `0x1` Generic Timer supported.
[15:12]	Virtualization	Virtualization support: `0x1` Virtualization is implemented.
[11:8]	MProgMod	M profile programmers model support: `0x0` Not supported.
[7:4]	Security	Security support: `0x1` Security implemented. This includes support for Monitor mode and the SMC instruction.
[3:0]	ProgMod	Indicates support for the standard programmers model for Armv4 and later. Model must support User, FIQ, IRQ, Supervisor, Abort, Undefined, and System modes: `0x1` Supported.

To access the ID_PFR1_EL1:

```
MRS <Xt>, ID_PFR1_EL1 ; Read ID_PFR1_EL1 into Xt
```

Register access is encoded as follows:

Table 4-28 ID_PFR1_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0001	001

4.3.6 AArch32 Processor Feature Register 2

The ID_PFR2_EL1 characteristics are:

Purpose Provides information about the programmers model and architecture extensions supported by the processor.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Configurations ID_PFR2_EL1 is architecturally mapped to AArch32 register ID_PFR2. See *4.5.8 Processor Feature Register 2* on page 4-229.

Attributes ID_PFR2_EL1 is a 32-bit register.

The following figure shows the ID_PFR2_EL1 bit assignments.

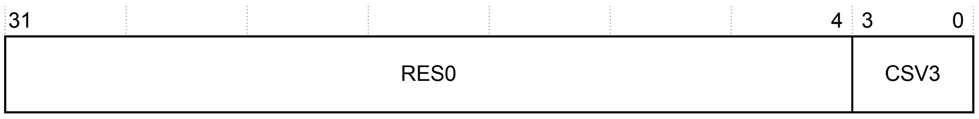

Figure 4-6 ID_PFR2_EL1 bit assignments

The following table shows the ID_PFR2_EL1 bit assignments.

Table 4-29 ID_PFR2_EL1 bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	CSV3	`0x1` Data loaded under control flow speculation with a permission or domain fault, if used as an address in a speculative load, cannot cause cache allocation.

To access the ID_PFR2_EL1:

```
MRS <Xt>, ID_PFR2_EL1 ; Read ID_PFR2_EL2 into Xt
```

Register access is encoded as follows:

Table 4-30 ID_PFR2_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	100

4.3.7 AArch32 Debug Feature Register 0

The ID_DFR0_EL1 characteristics are:

Purpose Provides top level information about the debug system in AArch32.

Usage constraints 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

Configurations ID_DFR0_EL1 is architecturally mapped to AArch32 register ID_DFR0. See *4.5.9 Debug Feature Register 0* on page 4-230.

Attributes ID_DFR0_EL1 is a 32-bit register.

The following figure shows the ID_DFR0_EL1 bit assignments.

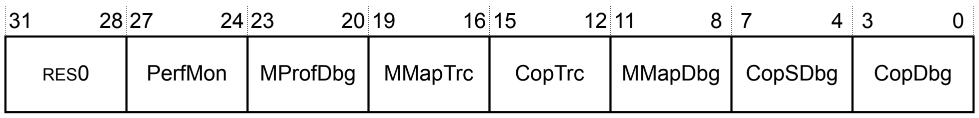

Figure 4-7 ID_DFR0_EL1 bit assignments

The following table shows the ID_DFR0_EL1 bit assignments.

Table 4-31 ID_DFR0_EL1 bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RES0.
[27:24]	PerfMon	Indicates support for performance monitor model: `0x3` Support for *Performance Monitor Unit version 3* (PMUv3) system registers.
[23:20]	MProfDbg	Indicates support for memory-mapped debug model for M profile processors: `0x0` Processor does not support M profile Debug architecture.
[19:16]	MMapTrc	Indicates support for memory-mapped trace model: `0x1` Support for Arm trace architecture, with memory-mapped access. In the Trace registers, the ETMIDR register gives more information about the implementation.
[15:12]	CopTrc	Indicates support for coprocessor-based trace model: `0x0` Processor does not support Arm trace architecture with CP14 access.
[11:8]	MMapDbg	Reserved, RES0.
[7:4]	CopSDbg	Indicates support for coprocessor-based Secure debug model: `0x6` Processor supports v8 Debug architecture, with CP14 access.
[3:0]	CopDbg	Indicates support for coprocessor-based debug model: `0x6` Processor supports v8 Debug architecture, with CP14 access.

To access the ID_DFR0_EL1:

```
MRS <Xt>, ID_DFR0_EL1 ; Read ID_DFR0_EL1 into Xt
```

Register access is encoded as follows:

Table 4-32 ID_DFR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0001	010

4.3.8 AArch32 Auxiliary Feature Register 0

This register is always RES0.

4.3.9 AArch32 Memory Model Feature Register 0

The ID_MMFR0_EL1 characteristics are:

Purpose Provides information about the memory model and memory management support in AArch32.

Usage constraints 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

Configurations ID_MMFR0_EL1 is architecturally mapped to AArch32 register ID_MMFR0. See *4.5.11 Memory Model Feature Register 0* on page 4-231.

Attributes ID_MMFR0_EL1 is a 32-bit register.

The following figure shows the ID_MMFR0_EL1 bit assignments.

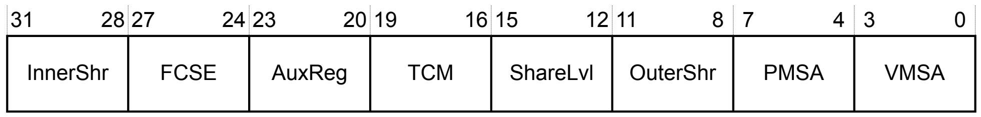

Figure 4-8 ID_MMFR0_EL1 bit assignments

The following table shows the ID_MMFR0_EL1 bit assignments.

Table 4-33 ID_MMFR0_EL1 bit assignments

Bits	Name	Function
[31:28]	InnerShr	Indicates the innermost shareability domain implemented: `0x1` Implemented with hardware coherency support.
[27:24]	FCSE	Indicates support for *Fast Context Switch Extension* (FCSE): `0x0` Not supported.
[23:20]	AuxReg	Indicates support for Auxiliary registers: `0x2` Support for *Auxiliary Fault Status Registers* (AIFSR and ADFSR) and Auxiliary Control Register.
[19:16]	TCM	Indicates support for TCMs and associated DMAs: `0x0` Not supported.
[15:12]	ShareLvl	Indicates the number of shareability levels implemented: `0x1` Two levels of shareability implemented.
[11:8]	OuterShr	Indicates the outermost shareability domain implemented: `0x1` Implemented with hardware coherency support.

Table 4-33 ID_MMFR0_EL1 bit assignments (continued)

Bits	Name	Function
[7:4]	PMSA	Indicates support for a *Protected Memory System Architecture* (PMSA): `0x0` Not supported.
[3:0]	VMSA	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.

To access the ID_MMFR0_EL1:

```
MRS <Xt>, ID_MMFR0_EL1 ; Read ID_MMFR0_EL1 into Xt
```

Register access is encoded as follows:

Table 4-34 ID_MMFR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0001	100

4.3.10 AArch32 Memory Model Feature Register 1

The ID_MMFR1_EL1 characteristics are:

Purpose Provides information about the memory model and memory management support in AArch32.

Usage constraints 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

Configurations ID_MMFR1_EL1 is architecturally mapped to AArch32 register ID_MMFR1. See *4.5.12 Memory Model Feature Register 1* on page 4-232.

Attributes ID_MMFR1_EL1 is a 32-bit register.

The following figure shows the ID_MMFR1_EL1 bit assignments.

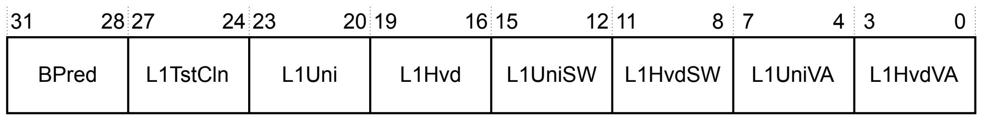

Figure 4-9 ID_MMFR1_EL1 bit assignments

The following table shows the ID_MMFR1_EL1 bit assignments.

Table 4-35 ID_MMFR1_EL1 bit assignments

Bits	Name	Function
[31:28]	BPred	Indicates branch predictor management requirements: `0x4` For execution correctness, branch predictor requires no flushing at any time.
[27:24]	L1TstCln	Indicates the supported L1 Data cache test and clean operations, for Harvard or unified cache implementation: `0x0` None supported.
[23:20]	L1Uni	Indicates the supported entire L1 cache maintenance operations, for a unified cache implementation: `0x0` None supported.
[19:16]	L1Hvd	Indicates the supported entire L1 cache maintenance operations, for a Harvard cache implementation: `0x0` None supported.
[15:12]	L1UniSW	Indicates the supported L1 cache line maintenance operations by set/way, for a unified cache implementation: `0x0` None supported.
[11:8]	L1HvdSW	Indicates the supported L1 cache line maintenance operations by set/way, for a Harvard cache implementation: `0x0` None supported.
[7:4]	L1UniVA	Indicates the supported L1 cache line maintenance operations by MVA, for a unified cache implementation: `0x0` None supported.
[3:0]	L1HvdVA	Indicates the supported L1 cache line maintenance operations by MVA, for a Harvard cache implementation: `0x0` None supported.

To access the ID_MMFR1_EL1:

```
MRS <Xt>, ID_MMFR1_EL1 ; Read ID_MMFR1_EL1 into Xt
```

Register access is encoded as follows:

Table 4-36 ID_MMFR1_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0001	101

4.3.11 AArch32 Memory Model Feature Register 2

The ID_MMFR2_EL1 characteristics are:

Purpose Provides information about the implemented memory model and memory management support in AArch32.

Usage constraints 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

Configurations ID_MMFR2_EL1 is architecturally mapped to AArch32 register ID_MMFR2. See *4.5.13 Memory Model Feature Register 2* on page 4-234.

Attributes ID_MMFR2_EL1 is a 32-bit register.

The following figure shows the ID_MMFR2_EL1 bit assignments.

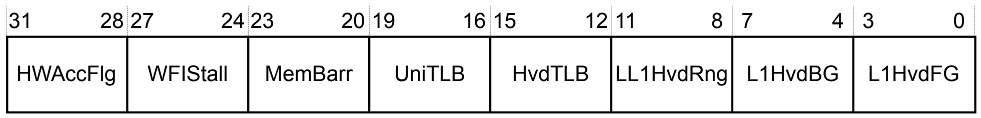

Figure 4-10 ID_MMFR2_EL1 bit assignments

The following table shows the ID_MMFR2_EL1 bit assignments.

Table 4-37 ID_MMFR2_EL1 bit assignments

Bits	Name	Function
[31:28]	HWAccFlg	Hardware access flag. Indicates support for a hardware access flag, as part of the VMSAv7 implementation: `0x0` Not supported.
[27:24]	WFIStall	Wait For Interrupt Stall. Indicates the support for *Wait For Interrupt* (WFI) stalling: `0x1` Support for WFI stalling.
[23:20]	MemBarr	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).
[19:16]	UniTLB	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`.
[15:12]	HvdTLB	Harvard TLB. Indicates the supported TLB maintenance operations, for a Harvard TLB implementation: `0x0` Not supported.
[11:8]	LL1HvdRng	L1 Harvard cache Range. Indicates the supported L1 cache maintenance range operations, for a Harvard cache implementation: `0x0` Not supported.

Table 4-37 ID_MMFR2_EL1 bit assignments (continued)

Bits	Name	Function
[7:4]	L1HvdBG	L1 Harvard cache Background fetch. Indicates the supported L1 cache background prefetch operations, for a Harvard cache implementation: `0x0` Not supported.
[3:0]	L1HvdFG	L1 Harvard cache Foreground fetch. Indicates the supported L1 cache foreground prefetch operations, for a Harvard cache implementation: `0x0` Not supported.

To access the ID_MMFR2_EL1:

```
MRS <Xt>, ID_MMFR2_EL1 ; Read ID_MMFR2_EL1 into Xt
```

Register access is encoded as follows:

Table 4-38 ID_MMFR2_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0001	110

4.3.12 AArch32 Memory Model Feature Register 3

The ID_MMFR3_EL1 characteristics are:

Purpose Provides information about the memory model and memory management support in AArch32.

Usage constraints 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

Configurations ID_MMFR3_EL1 is architecturally mapped to AArch32 register ID_MMFR3. See *4.5.14 Memory Model Feature Register 3* on page 4-236.

Attributes ID_MMFR3_EL1 is a 32-bit register.

The following figure shows the ID_MMFR3_EL1 bit assignments.

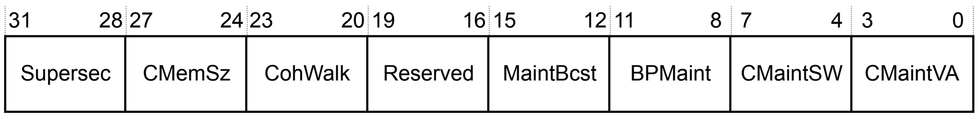

Figure 4-11 ID_MMFR3_EL1 bit assignments

The following table shows the ID_MMFR3_EL1 bit assignments.

Table 4-39 ID_MMFR3_EL1 bit assignments

Bits	Name	Function
[31:28]	Supersec	Supersections. Indicates support for supersections: `0x0` Supersections supported.
[27:24]	CMemSz	Cached memory size. Indicates the size of physical memory supported by the processor caches: `0x2` 1TB, corresponding to a 40-bit physical address range.
[23:20]	CohWalk	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.
[19:16]	-	Reserved, RES0.
[15:12]	MaintBcst	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.
[11:8]	BPMaint	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.
[7:4]	CMaintSW	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.
[3:0]	CMaintVA	Cache maintenance by MVA. Indicates the supported cache maintenance operations by MVA. `0x1` Supported hierarchical cache maintenance operations by MVA are: • Invalidate data cache by MVA. Invalidate data cache by MVA operations are treated as clean and invalidate data cache by MVA operations on the executing core. If the operation is broadcast to another core then it is broadcast as an invalidate data cache by MVA operation. • Clean data cache by MVA. • Clean and invalidate data cache by MVA. • Invalidate instruction cache by MVA. • Invalidate all instruction cache entries.

To access the ID_MMFR3_EL1:

```
MRS <Xt>, ID_MMFR3_EL1 ; Read ID_MMFR3_EL1 into Xt
```

Register access is encoded as follows:

Table 4-40 ID_MMFR3_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0001	111

4.3.13 AArch32 Instruction Set Attribute Register 0

The ID_ISAR0_EL1 characteristics are:

Purpose Provides information about the instruction sets implemented by the processor in AArch32.

Usage constraints 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

Configurations ID_ISAR0_EL1 is architecturally mapped to AArch32 register ID_ISAR0. See *4.5.15 Instruction Set Attribute Register 0* on page 4-237.

Attributes ID_ISAR0_EL1 is a 32-bit register.

The following figure shows the ID_ISAR0_EL1 bit assignments.

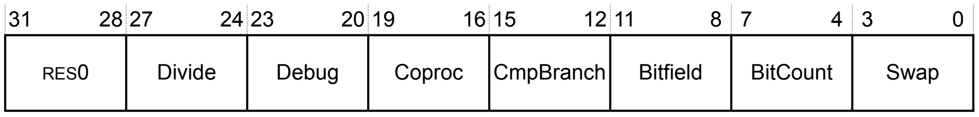

Figure 4-12 ID_ISAR0_EL1 bit assignments

The following table shows the ID_ISAR0_EL1 bit assignments.

Table 4-41 ID_ISAR0_EL1 bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RES0.
[27:24]	Divide	Indicates the implemented Divide instructions: `0x2` • `SDIV` and `UDIV` in the T32 instruction set. • `SDIV` and `UDIV` in the A32 instruction set.
[23:20]	Debug	Indicates the implemented Debug instructions: `0x1` `BKPT`.
[19:16]	Coproc	Indicates the implemented coprocessor instructions: `0x0` None implemented, except for separately attributed by the architecture including CP15, CP14, Advanced SIMD and floating-point.
[15:12]	CmpBranch	Indicates the implemented combined Compare and Branch instructions in the T32 instruction set: `0x1` `CBNZ` and `CBZ`.

Table 4-41 ID_ISAR0_EL1 bit assignments (continued)

Bits	Name	Function
[11:8]	Bitfield	Indicates the implemented bit field instructions: `0x1` `BFC`, `BFI`, `SBFX`, and `UBFX`.
[7:4]	BitCount	Indicates the implemented Bit Counting instructions: `0x1` `CLZ`.
[3:0]	Swap	Indicates the implemented Swap instructions in the A32 instruction set: `0x0` None implemented.

To access the ID_ISAR0_EL1:

```
MRS <Xt>, ID_ISAR0_EL1 ; Read ID_ISAR0_EL1 into Xt
```

Register access is encoded as follows:

Table 4-42 ID_ISAR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0010	000

4.3.14 AArch32 Instruction Set Attribute Register 1

The ID_ISAR1_EL1 characteristics are:

Purpose Provides information about the instruction sets implemented by the processor in AArch32.

Usage constraints 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

Configurations ID_ISAR1_EL1 is architecturally mapped to AArch32 register ID_ISAR1. See *4.5.16 Instruction Set Attribute Register 1* on page 4-239.

Attributes ID_ISAR1_EL1 is a 32-bit register.

The following figure shows the ID_ISAR1_EL1 bit assignments.

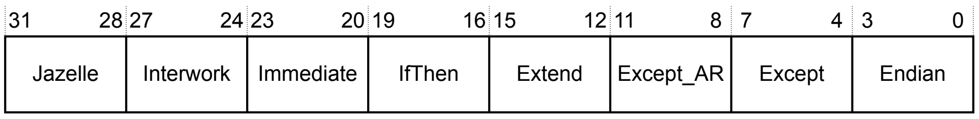

Figure 4-13 ID_ISAR1_EL1 bit assignments

The following table shows the ID_ISAR1_EL1 bit assignments.

Table 4-43 ID_ISAR1_EL1 bit assignments

Bits	Name	Function
[31:28]	Jazelle	Indicates the implemented Jazelle state instructions: `0x1` Adds the `BXJ` instruction, and the J bit in the PSR. This setting might indicate a trivial implementation of the Jazelle Extension.
[27:24]	Interwork	Indicates the implemented interworking instructions: `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.
[23:20]	Immediate	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.
[19:16]	IfThen	Indicates the implemented `If-Then` instructions in the T32 instruction set: `0x1` The `IT` instructions, and the IT bits in the PSRs.
[15:12]	Extend	Indicates the implemented Extend instructions: `0x2` • The `SXTB`, `SXTH`, `UXTB`, and `UXTH` instructions. • The `SXTB16`, `SXTAB`, `SXTAB16`, `SXTAH`, `UXTB16`, `UXTAB`, `UXTAB16`, and `UXTAH` instructions.
[11:8]	Except_AR	Indicates the implemented A profile exception-handling instructions: `0x1` The `SRS` and `RFE` instructions, and the A profile forms of the `CPS` instruction.
[7:4]	Except	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.
[3:0]	Endian	Indicates the implemented Endian instructions: `0x1` The `SETEND` instruction, and the E bit in the PSRs.

To access the ID_ISAR1_EL1:

```
MRS <Xt>, ID_ISAR1_EL1 ; Read ID_ISAR1_EL1 into Xt
```

Register access is encoded as follows:

Table 4-44 ID_ISAR1_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0010	001

4.3.15 AArch32 Instruction Set Attribute Register 2

The ID_ISAR2_EL1 characteristics are:

Purpose Provides information about the instruction sets implemented by the processor in AArch32.

Usage constraints 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

Configurations ID_ISAR2_EL1 is architecturally mapped to AArch32 register ID_ISAR2. See *4.5.17 Instruction Set Attribute Register 2* on page 4-240.

Attributes ID_ISAR2_EL1 is a 32-bit register.

The following figure shows the ID_ISAR2_EL1 bit assignments.

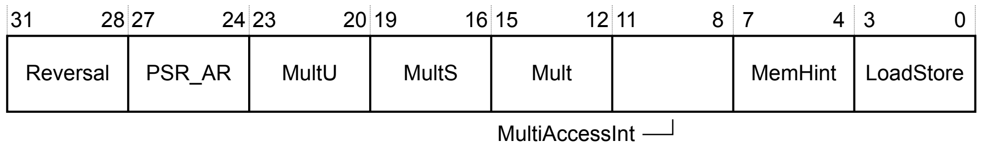

Figure 4-14 ID_ISAR2_EL1 bit assignments

The following table shows the ID_ISAR2_EL1 bit assignments.

Table 4-45 ID_ISAR2_EL1 bit assignments

Bits	Name	Function
[31:28]	Reversal	Indicates the implemented Reversal instructions: `0x2` The `REV`, `REV16`, `REVSH`, and `RBIT` instructions.
[27:24]	PSR_AR	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. **Note** 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 `SUBS PC, LR, #N` instruction.
[23:20]	MultU	Indicates the implemented advanced unsigned Multiply instructions: `0x2` The `UMULL`, `UMLAL` and `UMAAL` instructions.
[19:16]	MultS	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.

Table 4-45 ID_ISAR2_EL1 bit assignments (continued)

Bits	Name	Function
[15:12]	Mult	Indicates the implemented additional Multiply instructions: `0x2` The `MUL`, `MLA` and `MLS` instructions.
[11:8]	MultiAccessInt	Indicates the support for interruptible multi-access instructions: `0x0` No support. This means the LDM and STM instructions are not interruptible.
[7:4]	MemHint	Indicates the implemented memory hint instructions: `0x4` The `PLD` instruction. The `PLI` instruction. The `PLDW` instruction.
[3:0]	LoadStore	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.

To access the ID_ISAR2_EL1:

```
MRS <Xt>, ID_ISAR2_EL1 ; Read ID_ISAR2_EL1 into Xt
```

Register access is encoded as follows:

Table 4-46 ID_ISAR2_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0010	010

4.3.16 AArch32 Instruction Set Attribute Register 3

The ID_ISAR3_EL1 characteristics are:

Purpose Provides information about the instruction sets implemented by the processor in AArch32.

Usage constraints 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

Configurations ID_ISAR3_EL1 is architecturally mapped to AArch32 register ID_ISAR3. See *4.5.18 Instruction Set Attribute Register 3* on page 4-242.

Attributes ID_ISAR3_EL1 is a 32-bit register.

The following figure shows the ID_ISAR3_EL1 bit assignments.

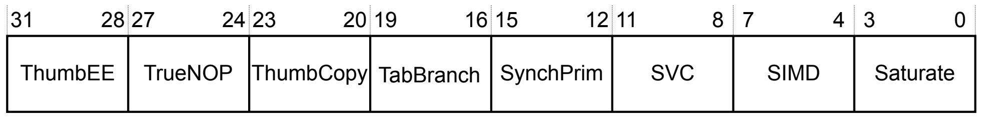

Figure 4-15 ID_ISAR3_EL1 bit assignments

The following table shows the ID_ISAR3_EL1 bit assignments.

Table 4-47 ID_ISAR3_EL1 bit assignments

Bits	Name	Function
[31:28]	ThumbEE	Indicates the implemented Thumb Execution Environment (T32EE) instructions: `0x0` None implemented.
[27:24]	TrueNOP	Indicates support for True NOP instructions: `0x1` True `NOP` instructions in both the A32 and T32 instruction sets, and additional NOP-compatible hints.
[23:20]	ThumbCopy	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.
[19:16]	TabBranch	Indicates the implemented Table Branch instructions in the T32 instruction set. `0x1` The `TBB` and `TBH` instructions.
[15:12]	SynchPrim	Indicates the implemented synchronization primitive instructions: `0x2` • The `LDREX` and `STREX` instructions. • The `CLREX`, `LDREXB`, `STREXB`, and `STREXH` instructions. • The `LDREXD` and `STREXD` instructions.
[11:8]	SVC	Indicates the implemented SVC instructions: `0x1` The `SVC` instruction.
[7:4]	SIMD	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.
[3:0]	Saturate	Indicates the implemented Saturate instructions: `0x1` The `QADD`, `QDADD`, `QDSUB`, `QSUB`, and the Q bit in the PSRs.

To access the ID_ISAR3_EL1:

```
MRS <Xt>, ID_ISAR3_EL1 ; Read ID_ISAR3_EL1 into Xt
```

Register access is encoded as follows:

Table 4-48 ID_ISAR3_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0010	011

4.3.17 AArch32 Instruction Set Attribute Register 4

The ID_ISAR4_EL1 characteristics are:

Purpose Provides information about the instruction sets implemented by the processor in AArch32.

Usage constraints 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

Configurations ID_ISAR4_EL1 is architecturally mapped to AArch32 register ID_ISAR4. See *4.5.19 Instruction Set Attribute Register 4* on page 4-244.

Attributes ID_ISAR4_EL1 is a 32-bit register.

The following figure shows the ID_ISAR4_EL1 bit assignments.

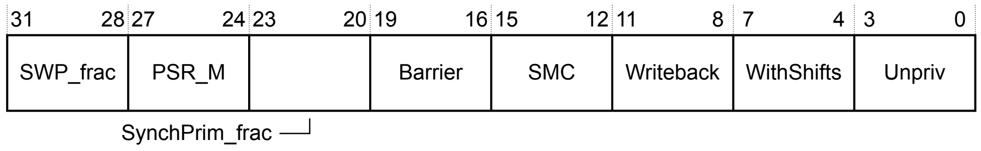

Figure 4-16 ID_ISAR4_EL1 bit assignments

The following table shows the ID_ISAR4_EL1 bit assignments.

Table 4-49 ID_ISAR4_EL1 bit assignments

Bits	Name	Function
[31:28]	SWP_frac	Indicates support for the memory system locking the bus for `SWP` or `SWPB` instructions: `0x0` `SWP` and `SWPB` instructions not implemented.
[27:24]	PSR_M	Indicates the implemented M profile instructions to modify the PSRs: `0x0` None implemented.
[23:20]	SynchPrim_frac	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.
[19:16]	Barrier	Indicates the supported Barrier instructions in the A32 and T32 instruction sets: `0x1` The `DMB`, `DSB`, and `ISB` barrier instructions.

Table 4-49 ID_ISAR4_EL1 bit assignments (continued)

Bits	Name	Function
[15:12]	SMC	Indicates the implemented `SMC` instructions: `0x1` The `SMC` instruction.
[11:8]	WriteBack	Indicates the support for write-back addressing modes: `0x1` Processor supports all of the write-back addressing modes defined in Armv8.
[7:4]	WithShifts	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.
[3:0]	Unpriv	Indicates the implemented unprivileged instructions. `0x2` • The `LDRBT`, `LDRT`, `STRBT`, and `STRT` instructions. • The `LDRHT`, `LDRSBT`, `LDRSHT`, and `STRHT` instructions.

To access the ID_ISAR4_EL1:

```
MRS <Xt>, ID_ISAR4_EL1 ; Read ID_ISAR4_EL1 into Xt
```

Register access is encoded as follows:

Table 4-50 ID_ISAR4_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0010	100

4.3.18 AArch32 Instruction Set Attribute Register 5

The ID_ISAR5_EL1 characteristics are:

Purpose Provides information about the instruction sets that the processor implements in AArch32.

Note

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension only under an additional license to the Cortex-A73 MPCore processor and Advanced SIMD and floating-point support licenses.

Usage constraints 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

Configurations ID_ISAR5_EL1 is architecturally mapped to AArch32 register ID_ISAR5. See *4.5.20 Instruction Set Attribute Register 5* on page 4-246.

Attributes ID_ISAR5_EL1 is a 32-bit register.

The following figure shows the ID_ISAR5_EL1 bit assignments.

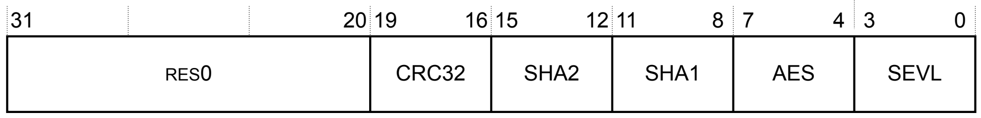

Figure 4-17 ID_ISAR5_EL1 bit assignments

The following table shows the ID_ISAR5_EL1 bit assignments.

Table 4-51 ID_ISAR5_EL1 bit assignments

Bits	Name	Function
[31:20]	-	Reserved, RES0.
[19:16]	CRC32	Indicates whether CRC32 instructions are implemented in AArch32 state. The value is: `0x1` CRC32 instructions are implemented.
[15:12]	SHA2	Indicates whether SHA2 instructions are implemented in AArch32 state. The possible values are: `0x0` No SHA2 instructions are implemented. This is the value if the implementation does not include the Cryptographic Extension. `0x1` `SHA256H`, `SHA256H2`, `SHA256SU0`, and `SHA256SU1` are implemented. This is the value if the implementation includes the Cryptographic Extension.
[11:8]	SHA1	Indicates whether SHA1 instructions are implemented in AArch32 state. The possible values are: `0x0` No SHA1 instructions are implemented. This is the value if the implementation does not include the Cryptographic Extension. `0x1` `SHA1C`, `SHA1P`, `SHA1M`, `SHA1H`, `SHA1SU0`, and `SHA1SU1` are implemented. This is the value if the implementation includes the Cryptographic Extension.
[7:4]	AES	Indicates whether AES instructions are implemented in AArch32 state. The possible values are: `0x0` No AES instructions are implemented. This is the value if the implementation does not include the Cryptographic Extension. `0x2` `AESE`, `AESD`, `AESMC`, and `AESIMC` are implemented, plus `PMULL` and `PMULL2` instructions operating on 64-bit elements. This is the value if the implementation includes the Cryptographic Extension.
[3:0]	SEVL	Indicates whether the `SEVL` instruction is implemented. The value is: `0x1` `SEVL` is implemented to send event local.

To access the ID_ISAR5_EL1:

```
MRS <Xt>, ID_ISAR5_EL1 ; Read ID_ISAR5_EL1 into Xt
```

Register access is encoded as follows:

Table 4-52 ID_ISAR5_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0010	101

4.3.19 AArch32 Memory Model Feature Register 4

The ID_MMFR4_EL1 characteristics are:

Purpose Provides information about the memory model and memory management support in AArch32.

Usage constraints 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

Configurations ID_MMFR4_EL1 is architecturally mapped to AArch32 register ID_MMFR4. See *4.5.21 Memory Model Feature Register 4* on page 4-248.

Attributes ID_MMFR4_EL1 is a 32-bit register.

The following figure shows the ID_MMFR4_EL1 bit assignments.

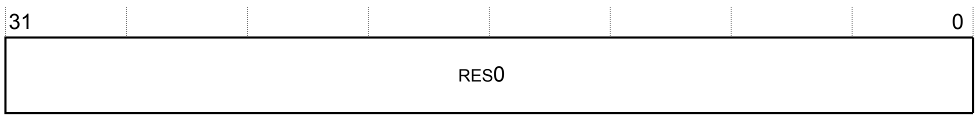

Figure 4-18 ID_MMFR4_EL1 bit assignments

The following table shows the ID_MMFR4_EL1 bit assignments.

Table 4-53 ID_MMFR4_EL1 bit assignments

Bits	Name	Function
[31:0]	-	Reserved, RES0.

To access the ID_MMFR4_EL1:

```
MRS <Xt>, ID_MMFR4_EL1 ; Read ID_MMFR4_EL1 into Xt
```

Register access is encoded as follows:

Table 4-54 ID_MMFR4_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0010	110

4.3.20 AArch64 Processor Feature Register 0

The ID_AA64PFR0_EL1 characteristics are:

Purpose Provides additional information about implemented processor features in AArch64.

Usage constraints 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

Configurations ID_AA64PFR0_EL1 is architecturally mapped to external register ID_AA64PFR0_EL1.

Attributes ID_AA64PFR0_EL1 is a 64-bit register.

The following figure shows the ID_AA64PFR0_EL1 bit assignments.

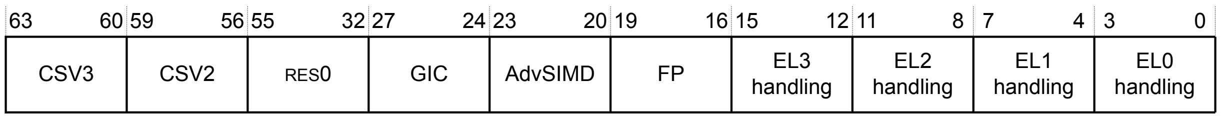

Figure 4-19 ID_AA64PFR0_EL1 bit assignments

The following table shows the ID_AA64PFR0_EL1 bit assignments.

Table 4-55 ID_AA64PFR0_EL1 bit assignments

Bits	Name	Function
[63:60]	CSV3	`0x1` Data loaded under control flow speculation with a permission or domain fault, if used as an address in a speculative load, cannot cause cache allocation.
[59:56]	CSV2	`0x1` Branch targets trained in one hardware described context cannot affect speculative execution in a different hardware described context.
[55:32]		Reserved, RES0.
[27:24]	GIC	GIC CPU interface: `0x0` GIC CPU interface is disabled. `0x1` GIC CPU interface is enabled. The input **GICDISABLE** defines the value of the GIC bit.
[23:20]	AdvSIMD[w]	Advanced SIMD. The possible values are: `0x0` Advanced SIMD is implemented.
[19:16]	FP[w]	Floating-point. The possible values are: `0x0` Floating-point is implemented.
[15:12]	EL3 handling	EL3 exception handling: `0x2` Instructions can be executed at EL3 in AArch64 or AArch32 state.
[11:8]	EL2 handling	EL2 exception handling: `0x2` Instructions can be executed at EL2 in AArch64 or AArch32 state.
[7:4]	EL1 handling	EL1 exception handling. The possible values are: `0x2` Instructions can be executed at EL1 in AArch64 or AArch32 state.
[3:0]	EL0 handling	EL0 exception handling. The possible values are: `0x2` Instructions can be executed at EL0 in AArch64 or AArch32 state.

To access the ID_AA64PFR0_EL1:

```
MRS <Xt>, ID_AA64PFR0_EL1 ; Read ID_AA64PFR0_EL1 into Xt
```

Register access is encoded as follows:

Table 4-56 ID_AA64PFR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0100	000

4.3.21 AArch64 Processor Feature Register 1

The ID_AA64PFR1_EL1 characteristics are:

Purpose Provides additional information about implemented processor features in AArch64.

Usage constraints 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

Configurations ID_AA64PFR1_EL1 is architecturally mapped to external register ID_AA64PFR1_EL1.

Attributes ID_AA64PFR1_EL1 is a 64-bit register.

The following figure shows the ID_AA64PFR1_EL1 bit assignments.

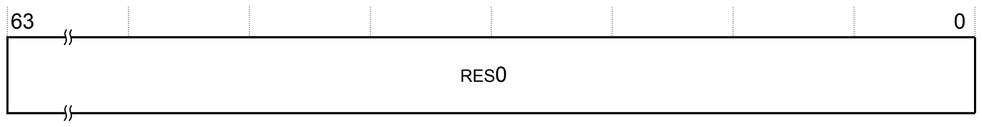

Figure 4-20 ID_AA64PFR1_EL1 bit assignments

The following table shows the ID_AA64PFR1_EL1 bit assignments.

Table 4-57 ID_AA64PFR1_EL1 bit assignments

Bits	Name	Function
[63:0]	-	Reserved, RES0.

To access the ID_AA64PFR1_EL1:

```
MRS <Xt>, ID_AA64PFR1_EL1 ; Read ID_AA64PFR1_EL1 into Xt
```

Register access is encoded as follows:

Table 4-58 ID_AA64PFR1_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0100	000

4.3.22 AArch64 Debug Feature Register 0, EL1

The ID_AA64DFR0_EL1 characteristics are:

[w] The FP and AdvSIMD both take the same value.

Purpose Provides top level information of the debug system in the AArch64 Execution state.

Usage constraints 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

Configurations ID_AA64DFR0_EL1 is architecturally mapped to external register ID_AA64DFR0.

Attributes ID_AA64DFR0_EL1 is a 64-bit register.

The following figure shows the ID_AA64DFR0_EL1 bit assignments.

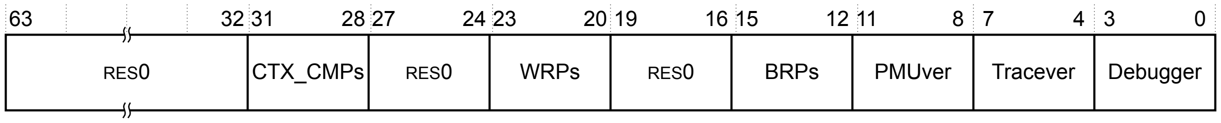

Figure 4-21 ID_AA64DFR0_EL1 bit assignments

The following table shows the ID_AA64DFR0_EL1 bit assignments.

Table 4-59 ID_AA64DFR0_EL1 bit assignments

Bits	Name	Function
[63:32]	-	Reserved, RES0.
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints: `0x1` Two breakpoints are context-aware.
[27:24]	-	Reserved, RES0.
[23:20]	WRPs	The number of watchpoints minus 1: `0x3` Four watchpoints.
[19:16]	-	Reserved, RES0.
[15:12]	BRPs	The number of breakpoints minus 1: `0x5` Six breakpoints.
[11:8]	PMUver	Performance Monitors extension version. `0x1` Performance monitor system registers implemented, PMUv3.
[7:4]	Tracever	Trace extension: `0x0` Trace system registers not implemented.
[3:0]	Debugger	Debug architecture version: `0x6` Armv8-A debug architecture implemented.

To access the ID_AA64DFR0_EL1:

```
MRS <Xt>, ID_AA64DFR0_EL1 ; Read ID_AA64DFR0_EL1 into Xt
```

Register access is encoded as follows:

Table 4-60 ID_AA64DFR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0101	000

4.3.23 AArch64 Debug Feature Register 1

The processor does not implement ID_AA64DFR1_EL1, therefore this register is always RES0.

4.3.24 AArch64 Auxiliary Feature Register 0

The processor does not implement ID_AA64AFR0_EL1, therefore this register is always RES0.

4.3.25 AArch64 Auxiliary Feature Register 1

The processor does not implement ID_AA64AFR1_EL1, therefore this register is always RES0.

4.3.26 AArch64 Instruction Set Attribute Register 0, EL1

The ID_AA64ISAR0_EL1 characteristics are:

Purpose Provides information about the optional cryptographic instructions that the processor can support.

Note

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension only under an additional license to the Cortex-A73 MPCore processor and Advanced SIMD and floating-point support licenses.

Usage constraints 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

Configurations ID_AA64ISAR0_EL1 is architecturally mapped to external register ID_AA64ISAR0_EL1.

Attributes ID_AA64ISAR0_EL1 is a 64-bit register.

The following figure shows the ID_AA64ISAR0_EL1 bit assignments.

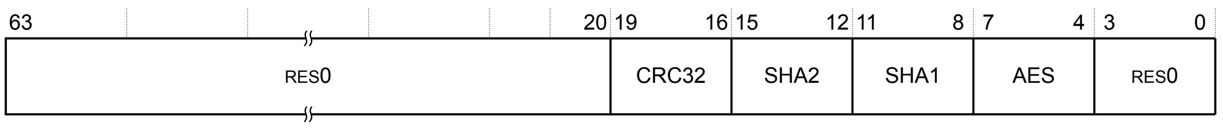

Figure 4-22 ID_AA64ISAR0_EL1 bit assignments

The following table shows the ID_AA64ISAR0_EL1 bit assignments.

Table 4-61 ID_AA64ISAR0_EL1 bit assignments

Bits	Name	Function
[63:20]	-	Reserved, RES0.
[19:16]	CRC32	Indicates whether CRC32 instructions are implemented. The value is: `0x1` CRC32 instructions are implemented.
[15:12]	SHA2	Indicates whether SHA2 instructions are implemented. The possible values are: `0x0` No SHA2 instructions are implemented. This is the value if the implementation does not include the Cryptographic Extension. `0x1` `SHA256H`, `SHA256H2`, `SHA256SU0`, and `SHA256SU1` are implemented. This is the value if the implementation includes the Cryptographic Extension.
[11:8]	SHA1	Indicates whether SHA1 instructions are implemented. The possible values are: `0x0` No SHA1 instructions are implemented. This is the value if the implementation does not include the Cryptographic Extension. `0x1` `SHA1C`, `SHA1P`, `SHA1M`, `SHA1H`, `SHA1SU0`, and `SHA1SU1` are implemented. This is the value if the implementation includes the Cryptographic Extension.
[7:4]	AES	Indicates whether AES instructions are implemented. The possible values are: `0x0` No AES instructions are implemented. This is the value if the implementation does not include the Cryptographic Extension. `0x2` `AESE`, `AESD`, `AESMC`, and `AESIMC` are implemented, plus `PMULL` and `PMULL2` instructions operating on 64-bit elements. This is the value if the implementation includes the Cryptographic Extension.
[3:0]	-	Reserved, RES0.

To access the ID_AA64ISAR0_EL1:

```
MRS <Xt>, ID_AA64ISAR0_EL1 ; Read ID_AA64ISAR0_EL1 into Xt
```

Register access is encoded as follows:

Table 4-62 ID_AA64ISAR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0110	000

4.3.27 AArch64 Instruction Set Attribute Register 1, EL1

The processor does not implement ID_AA64ISAR1_EL1, therefore this register is `RES0`.

4.3.28 AArch64 Memory Model Feature Register 0, EL1

The ID_AA64MMFR0_EL1 characteristics are:

Purpose Provides information about the implemented memory model and memory management support in the AArch64 Execution state.

Usage constraints 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

Configurations ID_AA64MMFR0_EL1 is architecturally mapped to external register ID_AA64MMFR0_EL1.

Attributes ID_AA64MMFR0_EL1 is a 64-bit register.

The following figure shows the ID_AA64MMFR0_EL1 bit assignments.

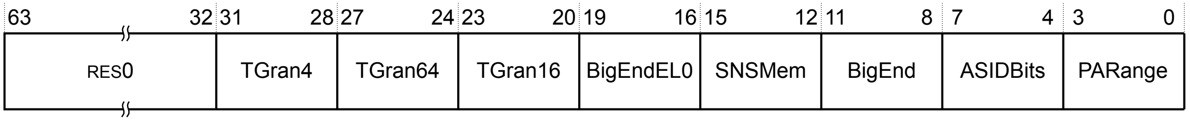

Figure 4-23 ID_AA64MMFR0_EL1 bit assignments

The following table shows the ID_AA64MMFR0_EL1 bit assignments.

Table 4-63 ID_AA64MMFR0_EL1 bit assignments

Bits	Name	Function
[63:32]	-	Reserved, RES0.
[31:28]	TGran4	Support for 4KB memory translation granule size: `0x0` Indicates that the 4KB granule is supported.
[27:24]	TGran64	Support for 64KB memory translation granule size: `0x0` Indicates that the 64KB granule is supported.
[23:20]	TGran16	Support for 16KB memory translation granule size: `0x1` Indicates that the 16KB granule is supported.
[19:16]	BigEndEL0	Mixed-endian support only at EL0. RES0
[15:12]	SNSMem	Secure versus Non-secure Memory distinction: `0x1` Supports a distinction between Secure and Non-secure Memory.
[11:8]	BigEnd	Mixed-endian configuration support: `0x1` Mixed-endian support. The SCTLR_ELx.EE and SCTLR_EL1.E0E bits are RW.
[7:4]	ASIDBits	Number of ASID bits: `0x2` 16 bits.
[3:0]	PARange	Physical address range supported: `0x2` 40-bit physical address range, that is, 1TByte.

To access the ID_AA64MMFR0_EL1:

```
MRS <Xt>, ID_AA64MMFR0_EL1 ; Read ID_AA64MMFR0_EL1 into Xt
```

Register access is encoded as follows:

Table 4-64 ID_AA64MMFR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0111	000

4.3.29 AArch64 Memory Model Feature Register 1

The processor does not implement ID_AA64MMFR0_EL1, therefore this register is RES0.

4.3.30 Cache Size ID Register

The CCSIDR_EL1 characteristics are:

Purpose Provides information about the architecture of the caches.

Usage constraints 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

Configurations CCSIDR_EL1 is architecturally mapped to AArch32 register CCSIDR. See *4.5.22 Cache Size ID Register* on page 4-248.

Attributes CCSIDR_EL1 is a 32-bit register.

The following figure shows the CCSIDR_EL1 bit assignments.

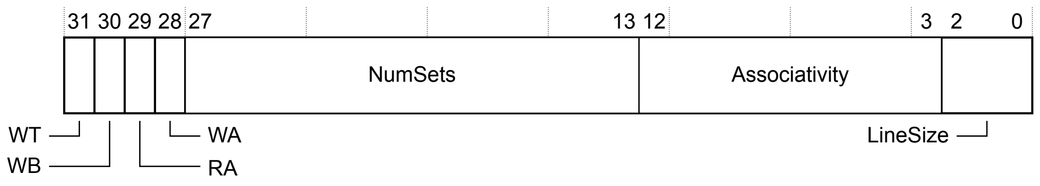

Figure 4-24 CCSIDR_EL1 bit assignments

The following table shows the CCSIDR_EL1 bit assignments.

Table 4-65 CCSIDR_EL1 bit assignments

Bits	Name	Function
[31]	WT	Indicates support for write-through: `0` Cache level does not support write-through.
[30]	WB	Indicates support for write-back: `0` Cache level does not support write-back. `1` Cache level supports write-back.

Table 4-65 CCSIDR_EL1 bit assignments (continued)

Bits	Name	Function
[29]	RA	Indicates support for Read-Allocation: `0` Cache level does not support Read-Allocation. `1` Cache level supports Read-Allocation.
[28]	WA	Indicates support for Write-Allocation: `0` Cache level does not support Write-Allocation. `1` Cache level supports Write-Allocation.
[27:13]	NumSets[x]	Indicates the number of sets in cache minus 1. Therefore, a value of 0 indicates 1 set in the cache. The number of sets does not have to be a power of 2.
[12:3]	Associativity[y]	Indicates the associativity of cache minus 1. Therefore, a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2.
[2:0]	LineSize[y]	Indicates the ($\log_2$ (number of words in cache line)) minus 2: `0x2` 16 words per line.

The following table shows the individual bit field and complete register encodings for the CCSIDR_EL1. The CSSELR determines which CCSIDR_EL1 to select.

To access the CCSIDR_EL1:

```
MRS <Xt>, CCSIDR_EL1 ; Read CCSIDR_EL1 into Xt
```

Register access is encoded as follows:

Table 4-66 CCSIDR_EL1 access encoding

op0	op1	CRn	CRm	op2
11	001	0000	0000	000

4.3.31 Cache Level ID Register

The CLIDR_EL1 characteristics are:

Purpose Identifies:
- The type of cache, or caches, implemented at each level.
- The Level of Coherency and Level of Unification for the cache hierarchy.

Usage constraints 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

Configurations CLIDR_EL1 is architecturally mapped to AArch32 register CLIDR. See *4.5.23 Cache Level ID Register* on page 4-250.

[x] For more information about encoding, see *Table 4-202 CCSIDR encodings* on page 4-250.
[y] *Table 4-202 CCSIDR encodings* on page 4-250 shows the individual bit field and complete register encodings for the CCSIDR. The CSSELR determines which CCSIDR to select.

Attributes CLIDR_EL1 is a 64-bit register.

The following figure shows the CLIDR_EL1 bit assignments.

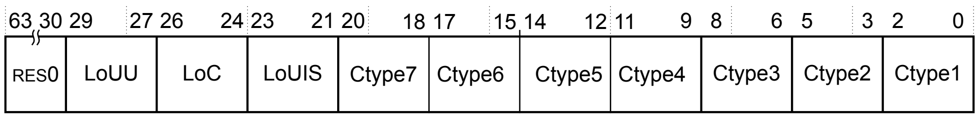

Figure 4-25 CLIDR_EL1 bit assignments

The following table shows the CLIDR_EL1 bit assignments.

Table 4-67 CLIDR_EL1 bit assignments

Bits	Name	Function
[63:30]	-	Reserved, RES0.
[29:27]	LoUU	Indicates the Level of Unification Uniprocessor for the cache hierarchy: `0x1` L1 cache is the last level of cache that must be cleaned or invalidated when cleaning or invalidating to the point of unification for the processor.
[26:24]	LoC	Indicates the Level of Coherency for the cache hierarchy: `0x2` L2 cache implemented. A clean to the point of coherency operation requires the L1 and L2 caches to be cleaned.
[23:21]	LoUIS	Indicates the Level of Unification Inner Shareable for the cache hierarchy: `0x1` L1 cache is the last level of cache that must be cleaned or invalidated when cleaning or invalidating to the point of unification for the Inner Shareable shareability domain.
[20:18]	Ctype7	Indicates the type of cache if the processor implements L7 cache: `0x0` L7 cache not implemented.
[17:15]	Ctype6	Indicates the type of cache if the processor implements L6 cache: `0x0` L6 cache not implemented.
[14:12]	Ctype5	Indicates the type of cache if the processor implements L5 cache: `0x0` L5 cache not implemented.
[11:9]	Ctype4	Indicates the type of cache if the processor implements L4 cache: `0x0` L4 cache not implemented.
[8:6]	Ctype3[z]	Indicates the type of cache if the processor implements L3 cache: `0x0` L3 cache not implemented.

[z] If software reads the Cache Type fields from Ctype1 upwards, after it has seen a value of `0b000`, no caches exist at further-out levels of the hierarchy. So, for example, if Ctype2 is the first Cache Type field with a value of `0b000`, the value of Ctype3 must be ignored.

Table 4-67 CLIDR_EL1 bit assignments (continued)

Bits	Name	Function
[5:3]	Ctype2	Indicates the type of cache if the processor implements L2 cache: `0x4` L2 cache is implemented as a unified cache.
[2:0]	Ctype1	Indicates the type of cache implemented at L1: `0x3` Separate instruction and data caches at L1.

To access the CLIDR_EL1:

```
MRS <Xt>, CLIDR_EL1 ; Read CLIDR_EL1 into Xt
```

Register access is encoded as follows:

Table 4-68 CLIDR_EL1 access encoding

op0	op1	CRn	CRm	op2
11	001	0000	0000	001

4.3.32 Auxiliary ID Register

The processor does not implement AIDR_EL1, so this register is always RES0.

4.3.33 Cache Size Selection Register

The CSSELR_EL1 characteristics are:

Purpose Selects the current *4.5.22 Cache Size ID Register* on page 4-248, by specifying:
- The required cache level.
- The cache type, either instruction or data cache.

Usage constraints 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

Configurations CSSELR_EL1 is architecturally mapped to AArch32 register CSSELR(NS). See *4.5.25 Cache Size Selection Register* on page 4-252.

Attributes CSSELR_EL1 is a 32-bit register.

The following figure shows the CSSELR_EL1 bit assignments.

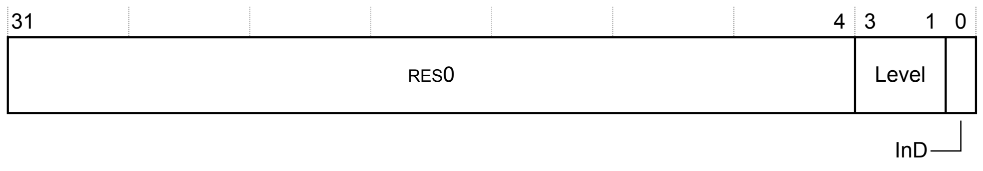

Figure 4-26 CSSELR_EL1 bit assignments

The following table shows the CSSELR_EL1 bit assignments.

Table 4-69 CSSELR_EL1 bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0
[3:1]	Level[aa]	Cache level of required cache: `0b000` L1. `0b001` L2. `0b010-0b111` Reserved.
[0]	InD[aa]	Instruction not Data bit: `0` Data or unified cache. `1` Instruction cache.

To access the CSSELR_EL1:

```
MRS <Xt>, CSSELR_EL1 ; Read CSSELR_EL1 into Xt
MSR CSSELR_EL1, <Xt> ; Write Xt to CSSELR_EL1
```

Register access is encoded as follows:

Table 4-70 CSSELR_EL1 access encoding

op0	op1	CRn	CRm	op2
11	010	0000	0000	000

4.3.34 Cache Type Register

The CTR_EL0 characteristics are:

Purpose Provides information about the architecture of the caches.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
Config	RO	RO	RO	RO	RO

This register is accessible at EL0 when SCTLR_EL1.UCT is set to 1.

Configurations CTR_EL0 is architecturally mapped to AArch32 register CTR. See *4.5.26 Cache Type Register* on page 4-253.

Attributes CTR_EL0 is a 32-bit register.

The following figure shows the CTR_EL0 bit assignments.

[aa] The combination of Level=`0b001` and InD=`1` is reserved.

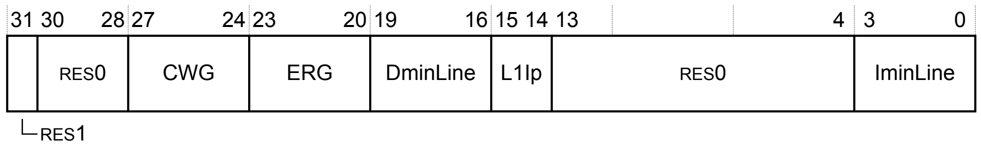

Figure 4-27 CTR_EL0 bit assignments

The following table shows the CTR_EL0 bit assignments.

Table 4-71 CTR_EL0 bit assignments

Bits	Name	Function
[31]	-	Reserved, RES1.
[30:28]	-	Reserved, RES0.
[27:24]	CWG	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: `0x4` Cache Write-Back granule size is 16 words.
[23:20]	ERG	Exclusives Reservation Granule. Log_2 of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions: `0x4` Exclusive reservation granule size is 16 words.
[19:16]	DminLine	Log_2 of the number of words in the smallest cache line of all the data and unified caches that the processor controls: `0x4` Smallest data cache line size is 16 words.
[15:14]	L1Ip	L1 Instruction cache policy. Indicates the indexing and tagging policy for the L1 Instruction cache: `0b10` *Virtually Indexed Physically Tagged* (VIPT).
[13:4]	-	Reserved, RES0.
[3:0]	IminLine	Log_2 of the number of words in the smallest cache line of all the instruction caches that the processor controls: `0x4` Smallest instruction cache line size is 16 words.

To access the CTR_EL0:

```
MRS <Xt>, CTR_EL0 ; Read CTR_EL0 into Xt
```

Register access is encoded as follows:

Table 4-72 CTR_EL0 access encoding

op0	op1	CRn	CRm	op2
11	011	0000	0000	001

4.3.35 Data Cache Zero ID Register

The DCZID_EL0 characteristics are:

Purpose Indicates the block size written with byte values of zero by the `DC ZVA` (Cache Zero by Address), system instruction.

Usage constraints 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	RO

Configurations There are no configuration notes.

Attributes DCZID_EL0 is a 32-bit register.

The following figure shows the DCZID_EL0 bit assignments.

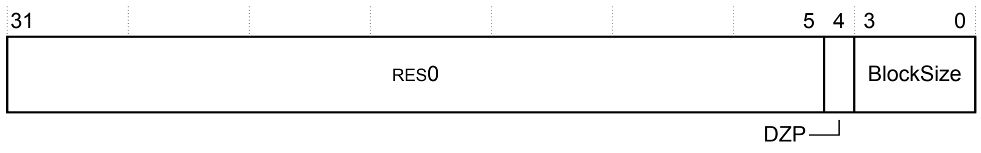

Figure 4-28 DCZID_EL0 bit assignments

The following table shows the DCZID_EL0 bit assignments.

Table 4-73 DCZID_EL0 bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0.
[4]	DZP	`0` DC ZVA instruction permitted.
[3:0]	BlockSize	Log2 of the block size in words: `0x4` The block size is 16 words.

To access the DCZID_EL0:

```
MRS <Xt>, DCZID_EL0 ; Read DCZID_EL0 into Xt
```

Register access is encoded as follows:

Table 4-74 DCZID_EL0 access encoding

op0	op1	CRn	CRm	op2
11	011	0000	0000	111

4.3.36 Virtualization Processor ID Register

The VPIDR_EL2 characteristics are:

Purpose

Holds the value of the Virtualization Processor ID. This is the value returned by Non-secure EL1 reads of MIDR. See *4.3.1 Main ID Register, EL1* on page 4-83.

Usage constraints

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	-

Configurations

VPIDR_EL2 is architecturally mapped to AArch32 register VPIDR. See *4.5.27 Virtualization Processor ID Register* on page 4-255.

Attributes

VPIDR_EL2 is a 32-bit register.

VPIDR_EL2 resets to the value of MIDR_EL1.

The following figure shows the VPIDR_EL2 bit assignments.

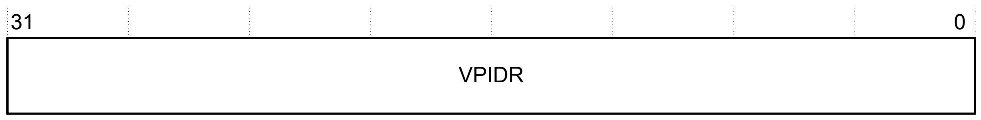

Figure 4-29 VPIDR_EL2 bit assignments

The following table shows the VPIDR_EL2 bit assignments.

Table 4-75 VPIDR_EL2 bit assignments

Bits	Name	Function
[31:0]	VPIDR	MIDR value returned by Non-secure PL1 reads of the MIDR. The MIDR description defines the subdivision of this value. See *4.3.1 Main ID Register, EL1* on page 4-83.

To access the VPIDR_EL2:

```
MRS <Xt>, VPIDR_EL2 ; Read VPIDR_EL2 into Xt
MSR VPIDR_EL2, <Xt> ; Write Xt to VPIDR_EL2
```

Register access is encoded as follows:

Table 4-76 VPIDR_EL2 access encoding

op0	op1	CRn	CRm	op2
11	100	0000	0000	000

4.3.37 Virtualization Multiprocessor ID Register

The VMPIDR_EL2 characteristics are:

Purpose

Provides the value of the Virtualization Multiprocessor ID. This is the value returned by Non-secure EL1 reads of MPIDR.

Usage constraints

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	-

Configurations

VMPIDR_EL2[31:0] is architecturally mapped to AArch32 register VMPIDR. See *4.5.28 Virtualization Multiprocessor ID Register* on page 4-255.

Attributes

VMPIDR_EL2 is a 64-bit register.

VMPIDR_EL2 resets to the value of MPIDR_EL2.

The following figure shows the VMPIDR_EL2 bit assignments.

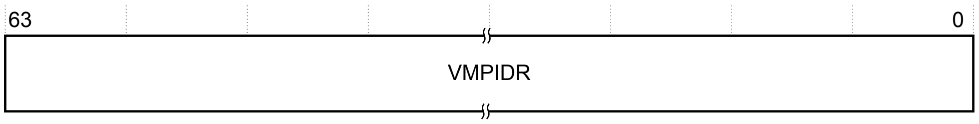

Figure 4-30 VMPIDR_EL2 bit assignments

The following table shows the VMPIDR_EL2 bit assignments.

Table 4-77 VMPIDR_EL2 bit assignments

Bits	Name	Function
[63:0]	VMPIDR	MPIDR value returned by Non-secure EL1 reads of the MPIDR_EL1. The MPIDR description defines the subdivision of this value. See *4.3.2 Multiprocessor Affinity Register* on page 4-84.

To access the VMPIDR_EL2:

```
MRS <Xt>, VMPIDR_EL2 ; Read VMPIDR_EL2 into Xt
MSR VMPIDR_EL2, <Xt> ; Write Xt to VMPIDR_EL2
```

Register access is encoded as follows:

Table 4-78 VMPIDR_EL2 access encoding

op0	op1	CRn	CRm	op2
11	100	0000	0000	101

4.3.38 System Control Register, EL1

The SCTLR_EL1 characteristics are:

Purpose Provides top-level control of the system, including its memory system at EL1.

SCTLR_EL1 is part of the Virtual memory control registers functional group.

Usage constraints 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

Configurations SCTLR_EL1 is architecturally mapped to AArch32 register SCTLR(NS) See *4.5.29 System Control Register* on page 4-256.

Attributes SCTLR_EL1 is a 32-bit register.

The following figure shows the SCTLR_EL1 bit assignments.

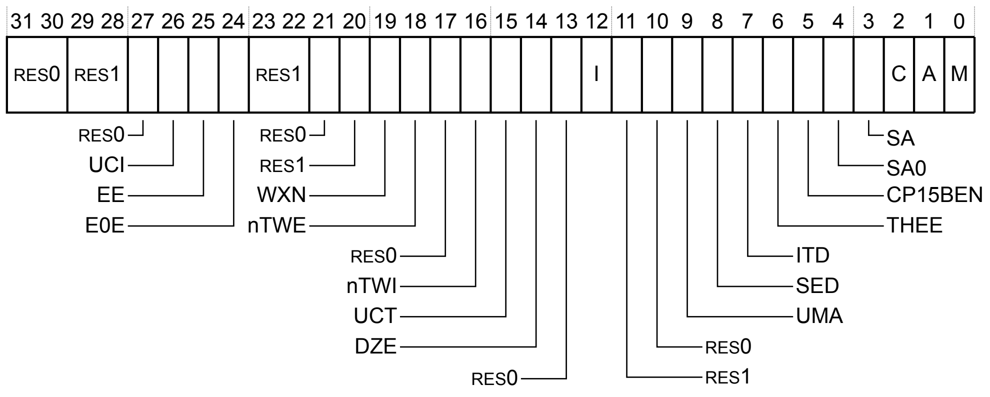

Figure 4-31 SCTLR_EL1 bit assignments

The following table shows the SCTLR_EL1 bit assignments.

Table 4-79 SCTLR_EL1 bit assignments

Bits	Name	Function
[31:30]	-	Reserved, RES0.
[29:28]	-	Reserved, RES1.
[27]	-	Reserved, RES0.
[26]	UCI	Enables EL0 access to the `DC CVAU`, `DC CIVAC`, `DC CVAC` and `IC IVAU` instructions in the AArch64 Execution state. The possible values are: `0` EL0 access disabled. This is the reset value. `1` EL0 access enabled.
[25]	EE	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. The reset value of this bit is determined by the CFGEND configuration pin.

Table 4-79 SCTLR_EL1 bit assignments (continued)

Bits	Name	Function
[24]	E0E	Endianness of explicit data access at EL0. The possible values are: `0` Explicit data accesses at EL0 are little-endian. This is reset value. `1` Explicit data accesses at EL0 are big-endian.
[23:22]	-	Reserved, RES1.
[21]	-	Reserved, RES0.
[20]	-	Reserved, RES1.
[19]	WXN	Write permission implies *Execute Never* (XN). This bit can be used to require all memory regions with write permissions to be treated as XN. The possible values are: `0` Regions with write permission are not forced XN. This is the reset value. `1` Regions with write permissions are forced XN.
[18]	nTWE	`WFE` non-trapping. The possible values are: `0` A `WFE` instruction executed at EL0, that, if this bit was set to 1, would permit entry to a low-power state, is trapped to EL1. `1` `WFE` instructions executed as normal. This is the reset value.
[17]	-	Reserved, RES0.
[16]	nTWI	`WFI` non-trapping. The possible values are: `0` A `WFI` instruction executed at EL0, that, if this bit was set to 1, would permit entry to a low-power state, is trapped to EL1. `1` `WFI` instructions executed as normal. This is the reset value.
[15]	UCT	Enables EL0 access to the CTR_EL0 register in AArch64 Execution state. The possible values are: `0` Disables EL0 access to the CTR_EL0 register. This is the reset value. `1` Enables EL0 access to the CTR_EL0 register.
[14]	DZE	Enables access to the `DC ZVA` instruction at EL0. The possible values are: `0` Disables execution access to the `DC ZVA` instruction at EL0. The instruction is trapped to EL1. This is the reset value. `1` Enables execution access to the `DC ZVA` instruction at EL0.
[13]	-	Reserved, RES0.
[12]	I	Instruction cache enable. The possible values are: `0` Instruction (L1) and unified (L2) caches disabled for instruction fetch. This is the reset value. `1` Instruction (L1) and unified (L2) caches enabled for instruction fetch.
[11]	-	Reserved, RES1.
[10]	-	Reserved, RES0.

Table 4-79 SCTLR_EL1 bit assignments (continued)

Bits	Name	Function
[9]	UMA	User Mask Access. Controls access to interrupt masks from EL0, when EL0 is using AArch64. The possible values of this bit are: `0` Disable access to the interrupt masks from EL0. This is the reset value. `1` Enable access to the interrupt masks from EL0.
[8]	SED	SETEND instruction disable. The possible values are: `0` The SETEND instruction is enabled. This is the reset value. `1` The SETEND instruction is UNDEFINED.
[7]	ITD	Reserved, RES0. All IT instruction functionality is enabled at EL0 using AArch32.
[6]	THEE	RES0 T32EE is not implemented.
[5]	CP15BEN	CP15 barrier enable. The possible values are: `0` CP15 barrier operations disabled. Their encodings are UNDEFINED. `1` CP15 barrier operations enabled. This is the reset value.
[4]	SA0	Enable EL0 stack alignment check. The possible values are: `0` Disable EL0 stack alignment check. `1` Enable EL0 stack alignment check. This is the reset value.
[3]	SA	Enable SP alignment check. The possible values are: `0` Disable SP alignment check. `1` Enable SP alignment check. This is the reset value.
[2]	C	Cache enable. The possible values are: `0` Data (L1) and unified (L2) caches disabled for data access. This is the reset value. `1` Data (L1) and unified (L2) caches enabled for data access.
[1]	A	Alignment check enable. The possible values are: `0` Alignment fault checking disabled. This is the reset value. `1` Alignment fault checking enabled.
[0]	M	MMU enable. The possible values are: `0` EL1 and EL0 stage 1 MMU disabled. This is the reset value. `1` EL1 and EL0 stage 1 MMU enabled.

To access the SCTLR_EL1:

```
MRS <Xt>, SCTLR_EL1 ; Read SCTLR_EL1 into Xt
MSR SCTLR_EL1, <Xt> ; Write Xt to SCTLR_EL1
```

4.3.39 Auxiliary Control Register, EL1

The processor does not implement the ACTLR_EL1 register. This register is always RES0.

4.3.40 Auxiliary Control Register, EL2

The ACTLR_EL2 characteristics are:

Purpose Controls write access to IMPLEMENTATION DEFINED registers in Non-secure EL1 modes, such as ECTLR, L2CTLR, and L2ECTLR.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations The ACTLR_EL2 is architecturally mapped to the AArch32 HACTLR register. See *4.5.36 Hyp Auxiliary Control Register* on page 4-269.

Attributes ACTLR_EL2 is a 32-bit register.

The following figure shows the ACTLR_EL2 bit assignments.

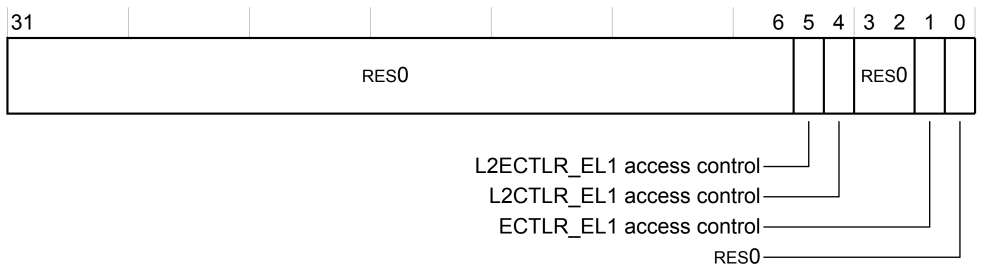

Figure 4-32 ACTLR_EL2 bit assignments

The following table shows the ACTLR_EL2 bit assignments.

Table 4-80 ACTLR_EL2 bit assignments

Bits	Name	Function
[31:6]	-	Reserved, RES0.
[5]	L2ECTLR_EL1 access control	L2ECTLR_EL1 write access control. The possible values are: `0` The register is not write accessible from Non-secure EL1. This is the reset value. `1` The register is write accessible from Non-secure EL1. Write access from Non-secure EL1 also requires ACTLR_EL3[5] to be set.
[4]	L2CTLR_EL1 access control	L2CTLR_EL1 write access control. The possible values are: `0` The register is not write accessible from Non-secure EL1. This is the reset value. `1` The register is write accessible from Non-secure EL1. Write access from Non-secure EL1 also requires ACTLR_EL3[4] to be set.
[3:2]	-	Reserved, RES0.

Table 4-80 ACTLR_EL2 bit assignments (continued)

Bits	Name	Function
[1]	ECTLR_EL1 access control	ECTLR_EL1 write access control. The possible values are: `0` The register is not write accessible from Non-secure EL1. This is the reset value. `1` The register is write accessible from Non-secure EL1. Write access from Non-secure EL1 also requires ACTLR_EL3[1] to be set.
[0]	-	Reserved, RES0.

To access the ACTLR_EL2:

```
MRS <Xt>, ACTLR_EL2 ; Read ACTLR_EL2 into Xt
MSR ACTLR_EL2, <Xt> ; Write Xt to ACTLR_EL2
```

4.3.41 Auxiliary Control Register, EL3

The ACTLR_EL3 characteristics are:

Purpose Controls write access to IMPLEMENTATION DEFINED registers in EL2, such as ECTLR, L2CTLR, and L2ECTLR.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations ACTLR_EL3 is mapped to AArch32 register ACTLR (S). See *4.5.30 Auxiliary Control Register* on page 4-260.

Attributes ACTLR_EL3 is a 32-bit register.

The following figure shows the ACTLR_EL3 bit assignments.

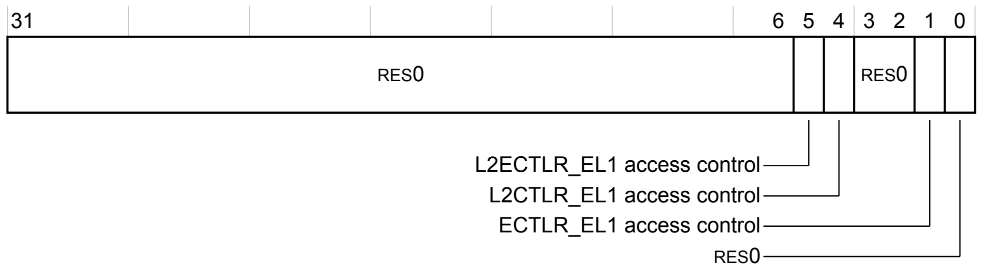

Figure 4-33 ACTLR_EL3 bit assignments

The following table shows the ACTLR_EL3 bit assignments.

Table 4-81 ACTLR_EL3 bit assignments

Bits	Name	Function
[31:6]	-	Reserved, RES0.
[5]	L2ECTLR_EL1 access control	L2ECTLR_EL1 write access control. The possible values are: `0` The register is not write accessible from a lower exception level. This is the reset value. `1` The register is write accessible from EL2.
[4]	L2CTLR_EL1 access control	L2CTLR_EL1 write access control. The possible values are: `0` The register is not write accessible from a lower exception level. This is the reset value. `1` The register is write accessible from EL2.
[3:2]	-	Reserved, RES0.
[1]	ECTLR_EL1 access control	ECTLR_EL1 write access control. The possible values are: `0` The register is not write accessible from a lower exception level. This is the reset value. `1` The register is write accessible from EL2.
[0]	-	Reserved, RES0.

To access the ACTLR_EL3:

```
MRS <Xt>, ACTLR_EL3 ; Read ACTLR_EL3 into Xt
MSR ACTLR_EL3, <Xt> ; Write Xt to ACTLR_EL3
```

4.3.42 Architectural Feature Access Control Register

The CPACR_EL1 characteristics are:

Purpose Controls access to trace functionality and access to registers associated with Advanced SIMD and Floating-point execution.

CPACR_EL1 is part of the Other system registers functional group.

Usage constraints 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

Configurations CPACR_EL1 is architecturally mapped to AArch32 register CPACR. See *4.5.31 Architectural Feature Access Control Register* on page 4-261.

Attributes CPACR_EL1 is a 32-bit register.

The following figure shows the CPACR_EL1 bit assignments.

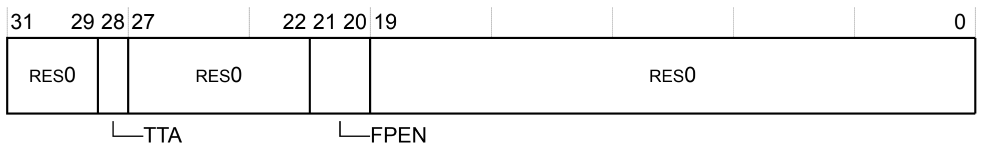

Figure 4-34 CPACR_EL1 bit assignments

The following table shows the CPACR_EL1 bit assignments.

Table 4-82 CPACR_EL1 bit assignments

Bits	Name	Function
[31:29]	-	Reserved, RES0.
[28]	TTA	Causes access to the Trace functionality to trap to EL1 when executed from EL0 or EL1. This bit is RES0.
[27:22]	-	Reserved, RES0.
[21:20]	FPEN	Traps instructions that access registers associated with Advanced SIMD and Floating-point execution to trap to EL1 when executed from EL0 or EL1. The possible values are: `0bX0` Trap any instruction in EL0 or EL1 that uses registers associated with Advanced SIMD and Floating-point execution. The reset value is `0b00`. `0b01` Trap any instruction in EL0 that uses registers associated with Advanced SIMD and Floating-point execution. Instructions in EL1 are not trapped. `0b11` No instructions are trapped.
[19:0]	-	Reserved, RES0.

To access the CPACR_EL1:

```
MRS <Xt>, CPACR_EL1 ; Read CPACR_EL1 into Xt
MSR CPACR_EL1, <Xt> ; Write Xt to CPACR_EL1
```

4.3.43 System Control Register, EL2

The SCTLR_EL2 characteristics are:

Purpose Provides top level control of the system, including its memory system at EL2.

SCTLR_EL2 is part of:
- The Virtual memory control registers functional group.
- The Hypervisor and virtualization registers functional group.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations SCTLR_EL2 is architecturally mapped to AArch32 register HSCTLR. See *4.5.37 Hyp System Control Register* on page 4-270.

Attributes SCTLR_EL2 is a 32-bit register.

The following figure shows the SCTLR_EL2 bit assignments.

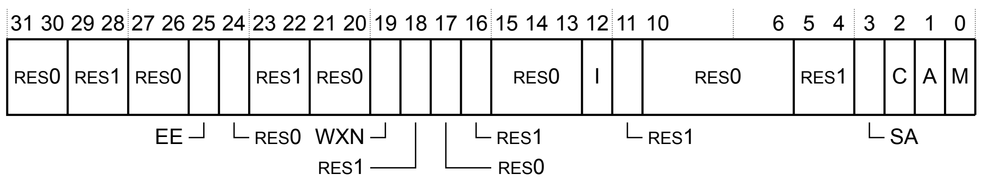

Figure 4-35 SCTLR_EL2 bit assignments

The following table shows the SCTLR_EL2 bit assignments.

Table 4-83 SCTLR_EL2 bit assignments

Bits	Name	Function
[31:30]	-	Reserved, RES0.
[29:28]	-	Reserved, RES1.
[27:26]	-	Reserved, RES0.
[25]	EE	Exception endianness. The possible values are: `0` Little endian. `1` Big endian.
[24]	-	Reserved, RES0.
[23:22]	-	Reserved, RES1.
[21:20]	-	Reserved, RES0.
[19]	WXN	Force treatment of all memory regions with write permissions as XN. The possible values are: `0` Regions with write permissions are not forced XN. This is the reset value. `1` Regions with write permissions are forced XN.
[18]	-	Reserved, RES1.
[17]	-	Reserved, RES0.
[16]	-	Reserved, RES1.
[15:13]	-	Reserved, RES0.
[12]	I	Instruction cache enable. The possible values are: `0` Instruction (L1) and unified (L2) caches disabled. `1` Instruction (L1) and unified (L2) caches enabled.
[11]	-	Reserved, RES1.
[10:6]	-	Reserved, RES0.
[5:4]	-	Reserved, RES1.
[3]	SA	Enables stack alignment check. The possible values are: `0` Disables stack alignment check. `1` Enables stack alignment check.
[2]	C	Global enable for data and unifies caches. The possible values are: `0` Data (L1) and unified (L2) caches disabled. `1` Data (L1) and unified (L2) caches enabled.

Table 4-83 SCTLR_EL2 bit assignments (continued)

Bits	Name	Function
[1]	A	Enable alignment fault check. The possible values are: `0` Disables alignment fault checking. `1` Enables alignment fault checking.
[0]	M	Global enable for the EL2 MMU. The possible values are: `0` Disables EL2 MMU. `1` Enables EL2 MMU.

To access the SCTLR_EL2:

```
MRS <Xt>, SCTLR_EL2 ; Read SCTLR_EL2 into Xt
MSR SCTLR_EL2, <Xt> ; Write Xt to SCTLR_EL2
```

4.3.44 Hypervisor Configuration Register

The HCR_EL2 characteristics are:

Purpose Provides configuration control for virtualization, including whether various Non-secure operations are trapped to EL2.

HCR_EL2 is part of the Hypervisor and virtualization registers functional group.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations HCR_EL2[31:0] is architecturally mapped to AArch32 register HCR. See *4.5.38 Hyp Configuration Register* on page 4-273.

HCR_EL2[63:32] is architecturally mapped to AArch32 register HCR2. See *4.5.39 Hyp Configuration Register 2* on page 4-278.

Attributes HCR_EL2 is a 64-bit register.

The following figure shows the HCR_EL2 bit assignments.

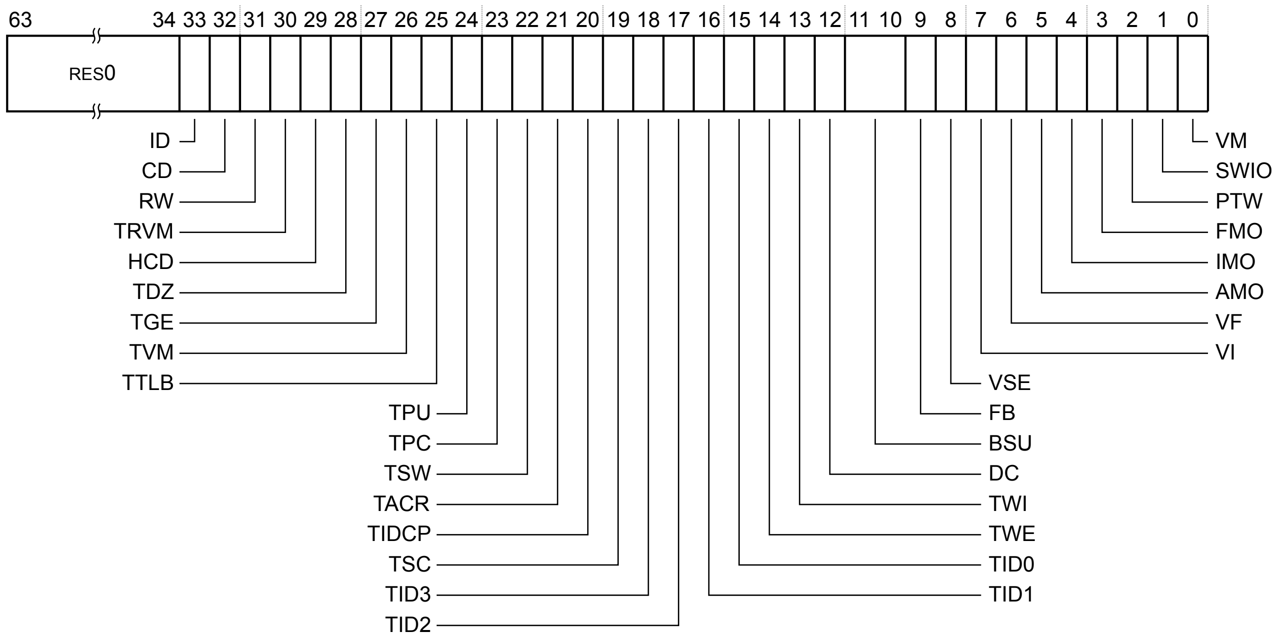

Figure 4-36 HCR_EL2 bit assignments

The following table shows the HCR_EL2 bit assignments.

Table 4-84 HCR_EL2 bit assignments

Bits	Name	Function
[63:34]	-	Reserved, RES0.
[33]	ID	Disables stage 2 instruction cache. When HCR_EL2.VM is 1, this forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable for the EL1/EL0 translation regimes. The possible values are: `0` Has no effect on stage 2 EL1/EL0 translation regime for instruction accesses. This is the reset value. `1` Forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable for the EL1/EL0 translation regime.
[32]	CD	Disables stage 2 data cache. When HCR_EL2.VM is 1, this forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable for the EL1/EL0 translation regimes. The possible values are: `0` Has no effect on stage 2 EL1/EL0 translation regime for data access or translation table walks. This is the reset value. `1` Forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable for the EL1/EL0 translation regime.
[31]	RW	Register width control for lower Exception levels. The possible values are: `0` Lower levels are all AArch32. This is the reset value. `1` EL1 is AArch64. EL0 is determined by the register width that is described in the current processing state when executing at EL0.
[30]	TRVM	Trap reads of Virtual Memory controls.[ab] The possible values are: `0` Non-secure EL1 reads are not trapped. This is the reset value. `1` Non-secure EL1 reads are trapped to EL2.

Table 4-84 HCR_EL2 bit assignments (continued)

Bits	Name	Function
[29]	HCD	Reserved, RES0.
[28]	TDZ	Traps DC ZVA instruction. The possible values are: `0` DC ZVA instruction is not trapped. This is the reset value. `1` DC ZVA instruction is trapped to EL2 when executed in Non-secure EL1 or EL0.
[27]	TGE	Traps general exceptions. If this bit is set, and SCR_EL3.NS is set, then: • All Non-secure EL1 exceptions are routed to EL2. • For Non-secure EL1, the SCTLR_EL1.M bit is reset to 0 regardless of its actual state other than the purpose of reading the bit. • The HCR_EL2.FMO, HCR_EL2.IMO, and HCR_EL2.AMO bits are treated as 1 regardless of their actual state other than for the purpose of reading the bits. • All virtual interrupts are disabled. • Any implementation defined mechanisms for signaling virtual interrupts are disabled. • An exception return to Non-secure EL1 is treated as an illegal exception return. The reset value is 0.
[26]	TVM	Trap virtual memory controls.[a][b] The possible values are: `0` Non-secure EL1 writes are not trapped. This is the reset value. `1` Non-secure EL1 writes are trapped to EL2.
[25]	TTLB	Traps TLB maintenance instructions.[a][b] The possible values are: `0` Non-secure EL1 TLB maintenance instructions are not trapped. This is the reset value. `1` TLB maintenance instructions that are executed from Non-secure EL1 that are not UNDEFINED are trapped to EL2.
[24]	TPU	Traps cache maintenance instructions to *Point of Unification* (POU).[a][b] The possible values are: `0` Cache maintenance instructions are not trapped. This is the reset value. `1` Cache maintenance instructions to the POU executed from Non-secure EL1 or EL0 that are not UNDEFINED are trapped to EL2.
[23]	TPC	Traps data or unified cache maintenance instructions to *Point of Coherency* (POC).[a][b] The possible values are: `0` Data or unified cache maintenance instructions are not trapped. This is the reset value. `1` Data or unified cache maintenance instructions by address to the POC executed from Non-secure EL1 or EL0 that are not UNDEFINED are trapped to EL2.
[22]	TSW	Traps data or unified cache maintenance instructions by Set or Way.[a][b] The possible values are: `0` Data or unified cache maintenance instructions are not trapped. This is the reset value. `1` Data or unified cache maintenance instructions by Set or Way executed from Non-secure EL1 that are not UNDEFINED are trapped to EL2 are not trapped.
[21]	TACR	Traps Auxiliary Control registers. The possible values are: `0` Accesses to Auxiliary Control registers are not trapped. This is the reset value. `1` Accesses to ACTLR in AArch32 state or the ACTLR_EL1 in the AArch64 state from Non-secure EL1 are trapped to EL2.

Table 4-84 HCR_EL2 bit assignments (continued)

Bits	Name	Function
[20]	TIDCP	Trap Implementation Dependent functionality. When 1, this causes accesses to the following instruction set space executed from Non-secure EL1 to be trapped to EL2: **AArch32** All CP15 `MCR` and `MRC` instructions as follows: • CRn is 9, Opcode1 is 0 to 7, CRm is c0, c1, c2, c5, c6, c7, or c8, and Opcode2 is 0 to 7. • CRn is 10, Opcode1 is 0 to 7, CRm is c0, c1, c4, or c8, and Opcode2 is 0 to 7. • CRn is 11, Opcode1 is 0 to 7, CRm is c0 to c8, or c15, and Opcode2 is 0 to 7. **AArch64** Reserved control space for IMPLEMENTATION DEFINED functionality. Accesses from EL0 are UNDEFINED. The reset value is 0.
[19]	TSC	Traps `SMC` instruction. The possible values are: `0` `SMC` instruction is not trapped. This is the reset value. `1` `SMC` instruction executed in Non-secure EL1 is trapped to EL2 for AArch32 and AArch64 Execution states.
[18]	TID3	Traps ID group 3 registers.[ab] The possible values are: `0` ID group 3 register accesses are not trapped. This is the reset value. `1` Reads to ID group 3 registers executed from Non-secure EL1 are trapped to EL2.
[17]	TID2	Traps ID group 2 registers.[ab] The possible values are: `0` ID group 2 register accesses are not trapped. This is the reset value. `1` Reads to ID group 2 registers and writes to CSSELR and CSSELR_EL1executed from Non-secure EL1 or EL0, if not UNDEFINED, are trapped to EL2.
[16]	TID1	Traps ID group 1 registers.[ab] The possible values are: `0` ID group 1 register accesses are not trapped. This is the reset value. `1` Reads to ID group 1 registers executed from Non-secure EL1 are trapped to EL2.
[15]	TID0	Traps ID group 0 registers.[ab] The possible values are: `0` ID group 0 register accesses are not trapped. This is the reset value. `1` Reads to ID group 0 registers executed from Non-secure EL1 are trapped to EL2.
[14]	TWE	Traps `WFE` instruction if it would cause suspension of execution. For example, if there is no pending `WFE` event. The possible values are: `0` `WFE` instruction is not trapped. This is the reset value. `1` `WFE` instruction executed in Non-secure EL1 or EL0 is trapped to EL2 for AArch32 and AArch64 Execution states.
[13]	TWI	Traps `WFI` instruction if it causes suspension of execution. For example, if there is no pending `WFI` event. The possible values are: `0` `WFI` instruction is not trapped. This is the reset value. `1` `WFI` instruction executed in Non-secure EL1 or EL0 is trapped to EL2 for AArch32 and AArch64 Execution states.

Table 4-84 HCR_EL2 bit assignments (continued)

Bits	Name	Function
[12]	DC	Default cacheable. When this bit is set it causes: • SCTLR_EL1.M to behave as 0 for all purposes other than reading the bit. • HCR_EL2.VM to behave as 1 for all purposes other than reading the bit. The memory type produced by the first stage of translation in Non-secure EL1 and EL0 is Non-shareable, Inner Write-Back Write-Allocate, Outer Write-Back Write-Allocate. The reset value is `0`.
[11:10]	BSU	Barrier shareability upgrade. Determines the minimum shareability domain that is supplied to any barrier executed from Non-secure EL1 or EL0. The possible values are: `0b00` No effect. This is the reset value. `0b01` Inner Shareable. `0b10` Outer Shareable. `0b11` Full system. This value is combined with the specified level of the barrier held in its instruction, according to the algorithm for combining shareability attributes.
[9]	FB	Forces broadcast.[ac] The possible values are: `0` Instructions are not broadcast. This is the reset value. `1` Forces instruction broadcast within Inner Shareable domain when executing from Non-secure EL1.
[8]	VSE	Virtual System Error/Asynchronous Abort. The possible values are: `0` Virtual System Error/Asynchronous Abort is not pending by this mechanism. This is the reset value. `1` Virtual System Error/Asynchronous Abort is pending by this mechanism. The virtual System Error/Asynchronous Abort is enabled only when the HCR_EL2.AMO bit is set.
[7]	VI	Virtual IRQ interrupt. The possible values are: `0` Virtual IRQ is not pending by this mechanism. This is the reset value. `1` Virtual IRQ is pending by this mechanism. The virtual IRQ is enabled only when the HCR_EL2.IMO bit is set.
[6]	VF	Virtual FIQ interrupt. The possible values are: `0` Virtual FIQ is not pending by this mechanism. This is the reset value. `1` Virtual FIQ is pending by this mechanism. The virtual FIQ is enabled only when the HCR_EL2.FMO bit is set.
[5]	AMO	Asynchronous abort and error interrupt routing. The possible values are: `0` Asynchronous external aborts and SError Interrupts while executing at Exception levels lower than EL2 are not taken at EL2. Virtual System Error/Asynchronous Abort is disabled. This is the reset value. `1` Asynchronous external aborts and SError Interrupts while executing at EL2 or lower are taken in EL2 unless routed by SCTLR_EL3.EA bit to EL3. Virtual System Error/Asynchronous Abort is enabled.

Table 4-84 HCR_EL2 bit assignments (continued)

Bits	Name	Function
[4]	IMO	Physical IRQ routing. The possible values are: `0` Physical IRQ while executing at Exception levels lower than EL2 are not taken at EL2. Virtual IRQ interrupt is disabled. This is the reset value. `1` Physical IRQ while executing at EL2 or lower are taken in EL2 unless routed by SCTLR_EL3.IRQ bit to EL3. Virtual IRQ interrupt is enabled.
[3]	FMO	Physical FIQ routing. The possible values are: `0` Physical FIQ while executing at Exception levels lower than EL2 are not taken at EL2. Virtual FIQ interrupt is disabled. This is the reset value. `1` Physical FIQ while executing at EL2 or lower are taken in EL2 unless routed by SCTLR_EL3.FIQ bit to EL3. Virtual FIQ interrupt is enabled.
[2]	PTW	Protected Table Walk. When this bit is set, if the stage 2 translation of a translation table access, made as part of a stage 1 translation table walk at EL0 or EL1, maps to Device memory, the access is faulted as a stage 2 Permission fault. The reset value is `0`.
[1]	SWIO	Set/Way Invalidation Override. Non-secure EL1 execution of the data cache invalidate by set/way instruction is treated as data cache clean and invalidate by set/way. When this bit is set: • DCISW is treated as DCCISW when in the AArch32 Execution state. • DC ISW is treated as DC CISW when in the AArch64 Execution state. This bit is RES1.
[0]	VM	Enables second stage of translation. The possible values are: `0` Disables second stage translation. This is the reset value. `1` Enables second stage translation for execution in Non-secure EL1 and EL0.

To access the HCR_EL2:

```
MRS <Xt>, HCR_EL2 ; Read HCR_EL2 into Xt
MSR HCR_EL2, <Xt> ; Write Xt to HCR_EL2
```

4.3.45 Hyp Debug Control Register

The MDCR_EL2 characteristics are:

Purpose Controls the trapping to Hyp mode of Non-secure accesses, at EL1 or lower, to functions provided by the debug and trace architectures and the Performance Monitor.

Usage constraints This register is accessible as follows:

EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

[ab] See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for the registers covered by this setting.
[ac] See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for the instructions covered by this setting.

Configurations
- MDCR_EL2 is architecturally mapped to AArch32 register HDCR. See *4.5.40 Hyp Debug Control Register* on page 4-279.
- This register is accessible only at EL2 or EL3.

Attributes MDCR_EL2 is a 32-bit register.

The following figure shows the MDCR_EL2 bit assignments.

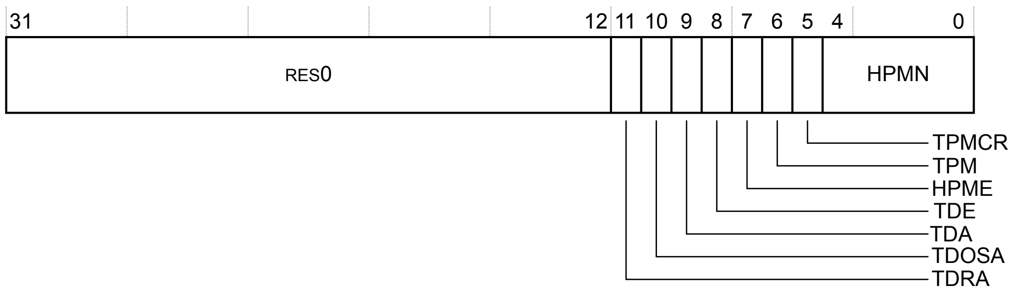

Figure 4-37 MDCR_EL2 bit assignments

The following table shows the MDCR_EL2 bit assignments.

Table 4-85 MDCR_EL2 bit assignments

Bits	Name	Function
[31:12]	-	Reserved, RES0.
[11]	TDRA	Trap debug ROM address register access. `0` Has no effect on accesses to debug ROM address registers from EL1 and EL0. `1` Trap valid Non-secure EL1 and EL0 access to debug ROM address registers to Hyp mode. When this bit is set to 1, any access to the following registers from EL1 or EL0 is trapped to EL2: • AArch32: DBGDRAR, DBGDSAR. • AArch64: MDRAR_EL1. If HCR_EL2.TGE is 1 or MDCR_EL2.TDE is 1, then this bit is ignored and treated as though it is 1 other than for the value read back from MDCR_EL2. On Warm reset, the field resets to 0.
[10]	TDOSA	Trap Debug OS-related register access: `0` Has no effect on accesses to OS-related debug registers. `1` Trap valid Non-secure accesses to OS-related debug registers to EL2. When this bit is set to 1, any access to the following registers from EL1 or EL0 is trapped to EL2: • AArch32: DBGOSLAR, DBGOSLSR, DBGOSDLR, DBGPRCR. • AArch64: OSLAR_EL1, OSLSR_EL1, OSDLR_EL1, DBGPRCR_EL1. If HCR_EL2.TGE is 1 or MDCR_EL2.TDE is 1, then this bit is ignored and treated as though it is 1 other than for the value read back from MDCR_EL2. On Warm reset, the field resets to 0.

Table 4-85 MDCR_EL2 bit assignments (continued)

Bits	Name	Function
[9]	TDA	Trap Debug Access: `0` Has no effect on accesses to Debug registers. `1` Trap valid Non-secure accesses to Debug registers to EL2. When this bit is set to 1, any valid Non-secure access to the debug registers from EL1 or EL0, other than the registers trapped by the TDRA and TDOSA bits, is trapped to EL2. If HCR_EL2.TGE is 1 or MDCR_EL2.TDE is 1, then this bit is ignored and treated as though it is 1 other than for the value read back from MDCR_EL2. On Warm reset, the field resets to 0.
[8]	TDE	Trap software debug exceptions: `0` Has no effect on software debug exceptions. `1` Route Software debug exceptions from Non-secure EL1 and EL0 to EL2. Also enables traps on all debug register accesses to EL2. If HCR_EL2.TGE is 1, then this bit is ignored and treated as though it is 1 other than for the value read back from MDCR_EL2. This bit resets to 0.
[7]	HPME	Hypervisor Performance Monitor Enable: `0` EL2 performance monitor counters disabled. `1` EL2 performance monitor counters enabled. When this bit is set to 1, the Performance Monitors counters that are reserved for use from EL2 or Secure state are enabled. For more information see the description of the HPMN field. This bit resets to 0.
[6]	TPM	Trap Performance Monitor accesses: `0` Has no effect on performance monitor accesses. `1` Trap Non-secure EL0 and EL1 accesses to Performance Monitors registers that are not unallocated to EL2. This bit resets to 0.

Table 4-85 MDCR_EL2 bit assignments (continued)

Bits	Name	Function
[5]	TPMCR	Trap PMCR_EL0 accesses: `0` Has no effect on PMCR_EL0 accesses. `1` Trap Non-secure EL0 and EL1 accesses to PMCR_EL0 to EL2. This bit resets to 0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[4:0]	HPMN	Hyp Performance Monitor count. Defines the number of Performance Monitors counters that are accessible from Non-secure EL1 and EL0 modes. In Non-secure state, HPMN divides the Performance Monitors counters as follows. For counter `n` in Non-secure state: For example, If PMnEVCNTR is performance monitor counter *n* then, in Non-secure state: • If *n* is in the range 0 ≤ *n* < HPMN, the counter is accessible from EL1 and EL2, and from EL0 if permitted by PMUSERENR_EL0. PMCR_EL0.E enables the operation of counters in this range. • If *n* is in the range HPMN ≤ *n* < 6, the counter is accessible only from EL2. MDCR_EL2.HPME enables the operation of counters in this range. There are six performance counters, specified by PMCR.N. If the field is set to 0, then Non-secure EL0 or EL1 has no access to any counters. If the field is set to a value greater than six, the behavior is the same as if the value is six. For reads of MDCR_EL2.HPMN by EL2 or higher, if this field is set to 0 or to a value larger than PMCR_EL0.N, the processor returns the value that was written to MDCR_EL2.HPMN. This field resets to `0x6`.

To access the MDCR_EL2:

```
MRS <Xt>, MDCR_EL2 ; Read MDCR_EL2 into Xt
MSR MDCR_EL2, <Xt> ; Write Xt to MDCR_EL2
```

4.3.46 Architectural Feature Trap Register, EL2

The CPTR_EL2 characteristics are:

Purpose Controls trapping to EL2 for accesses to CPACR, Trace functionality and registers associated with Advanced SIMD and Floating-point execution. Controls EL2 access to this functionality.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations CPTR_EL2 is architecturally mapped to AArch32 register HCPTR. See *4.5.41 Hyp Architectural Feature Trap Register* on page 4-282.

Attributes CPTR_EL2 is a 32-bit register.

The following figure shows the CPTR_EL2 bit assignments.

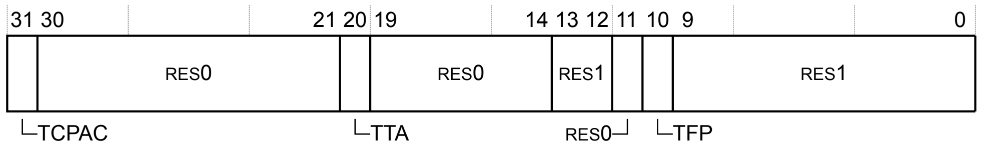

Figure 4-38 CPTR_EL2 bit assignments

The following table shows the CPTR_EL2 bit assignments.

Table 4-86 CPTR_EL2 bit assignments

Bits	Name	Function
[31]	TCPAC	Traps direct access to CPACR from Non-secure EL1 to EL2. The possible values are: `0` Access to CPACR is not trapped. This is the reset value. `1` Access to CPACR is trapped.
[30:21]	-	Reserved, RES0.
[20]	TTA	Trap Trace Access. Not implemented. RES0.
[19:14]	-	Reserved, RES0.
[13:12]	-	Reserved, RES1.
[11]	-	Reserved, RES0.
[10]	TFP	Traps instructions that access registers associated with Advanced SIMD and Floating-point execution from a lower exception level to EL2, unless trapped to EL1. The possible values are: `0` Instructions are not trapped. This is the reset value. `1` Instructions are trapped.
[9:0]	-	Reserved, RES1.

To access the CPTR_EL2:

```
MRS <Xt>, CPTR_EL2 ; Read CPTR_EL2 into Xt
MSR CPTR_EL2, <Xt> ; Write Xt to CPTR_EL2
```

4.3.47 Hyp System Trap Register

The HSTR_EL2 characteristics are:

Purpose Controls access to ThumbEE and coprocessor registers at lower exception levels in AArch32.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations HSTR_EL2 is architecturally mapped to AArch32 register HSTR. See *4.5.48 Hyp System Trap Register* on page 4-297.

Attributes HSTR_EL2 is a 32-bit register.

The following figure shows the HSTR_EL2 bit assignments.

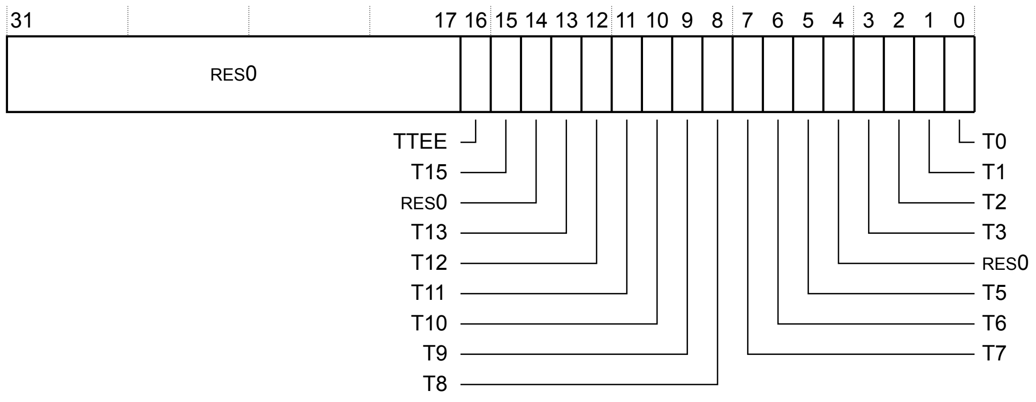

Figure 4-39 HSTR_EL2 bit assignments

The following table shows the HSTR_EL2 bit assignments.

Table 4-87 HSTR_EL2 bit assignments

Bits	Name	Function
[31:17]	-	Reserved, RES0.
[16]	TTEE	Trap T32EE. This value is: `0` T32EE is not supported.
[15]	T15	Trap coprocessor primary register CRn = 15. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 15 to Hyp mode.
[14]	-	Reserved, RES0.
[13]	T13	Trap coprocessor primary register CRn = 13. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 13 to Hyp mode.
[12]	T12	Trap coprocessor primary register CRn = 12. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 12 to Hyp mode.
[11]	T11	Trap coprocessor primary register CRn = 11. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 11 to Hyp mode.
[10]	T10	Trap coprocessor primary register CRn = 10. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 10 to Hyp mode.

Table 4-87 HSTR_EL2 bit assignments (continued)

Bits	Name	Function
[9]	T9	Trap coprocessor primary register CRn = 9. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 9 to Hyp mode.
[8]	T8	Trap coprocessor primary register CRn = 8. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 8 to Hyp mode.
[7]	T7	Trap coprocessor primary register CRn = 7. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 7 to Hyp mode.
[6]	T6	Trap coprocessor primary register CRn = 6. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 6 to Hyp mode.
[5]	T5	Trap coprocessor primary register CRn = 5. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 5 to Hyp mode.
[4]	-	Reserved, RES0.
[3]	T3	Trap coprocessor primary register CRn = 3. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 3 to Hyp mode.
[2]	T2	Trap coprocessor primary register CRn = 2. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 2 to Hyp mode.
[1]	T1	Trap coprocessor primary register CRn = 1. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 1 to Hyp mode.
[0]	T0	Trap coprocessor primary register CRn = 0. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. This is the reset value. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 0 to Hyp mode.

To access the HSTR_EL2:

```
MRS <Xt>, HSTR_EL2 ; Read HSTR_EL2 into Xt
MSR HSTR_EL2, <Xt> ; Write Xt to HSTR_EL2
```

4.3.48 Hyp Auxiliary Configuration Register

The processor does not implement HACR_EL2, so this register is always RES0.

4.3.49 System Control Register, EL3

The SCTLR_EL3 characteristics are:

Purpose Provides top level control of the system, including its memory system at EL3.

SCTLR_EL3 is part of the Virtual memory control registers functional group.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations SCTLR_EL3 is mapped to AArch32 register SCTLR(S). See *4.5.29 System Control Register* on page 4-256.

Attributes SCTLR_EL3 is a 32-bit register.

The following figure shows the SCTLR_EL3 bit assignments.

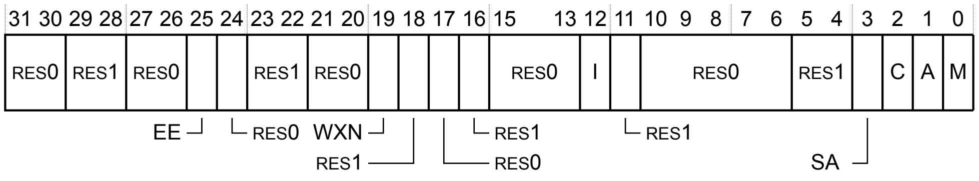

Figure 4-40 SCTLR_EL3 bit assignments

The following table shows the SCTRLR_EL3 bit assignments.

Table 4-88 SCTLR_EL3 bit assignments

Bits	Name	Function
[31:30]	-	Reserved, RES0.
[29:28]	-	Reserved, RES1.
[27:26]	-	Reserved, RES0.
[25]	EE	Exception endianness. This bit controls the endianness for: • Explicit data accesses at EL3. • Stage 1 translation table walks at EL3. The possible values are: `0` Little endian. This is the reset value. `1` Big endian. The input **CFGEND** defines the reset value of the EE bit.
[24]	-	Reserved, RES0.
[23:22]	-	Reserved, RES1.
[21:20]	-	Reserved, RES0.
[19]	WXN	Force treatment of all memory regions with write permissions as XN. The possible values are: `0` Regions with write permissions are not forced XN. This is the reset value. `1` Regions with write permissions are forced XN.

Table 4-88 SCTLR_EL3 bit assignments (continued)

Bits	Name	Function
[18]	-	Reserved, RES1.
[17]	-	Reserved, RES0.
[16]	-	Reserved, RES1.
[15:13]	-	Reserved, RES0.
[12]	I	Global instruction cache enable. The possible values are: `0` Instruction (L1) and unified (L2) caches disabled. This is the reset value. `1` Instruction (L1) and unified (L2) caches enabled.
[11]	-	Reserved, RES1.
[10:6]	-	Reserved, RES0.
[5:4]	-	Reserved, RES1.
[3]	SA	Enables stack alignment check. The possible values are: `0` Disables stack alignment check. `1` Enables stack alignment check. This is the reset value.
[2]	C	Global enable for data and unified caches. The possible values are: `0` Data (L1) and unified (L2) caches disabled. This is the reset value. `1` Data (L1) and unified (L2) caches enabled.
[1]	A	Enable alignment fault check. The possible values are: `0` Disables alignment fault checking. This is the reset value. `1` Enables alignment fault checking.
[0]	M	Global enable for the EL3 MMU. The possible values are: `0` Disables EL3 MMU. This is the reset value. `1` Enables EL3 MMU.

To access the SCTLR_EL3:

```
MRS <Xt>, SCTLR_EL3 ; Read SCTLR_EL3 into Xt
MSR SCTLR_EL3, <Xt> ; Write Xt to SCTLR_EL3
```

4.3.50 Secure Configuration Register

The SCR_EL3 characteristics are:

Purpose Defines the configuration of the security state. SCR_EL3 specifies:
- Security state of EL0 and EL1, either Secure or Non-secure.
- Register width at lower Exception levels.
- The Exception level that the processor takes exceptions at, if an IRQ, FIQ, or external abort occurs.

SCR_EL3 is part of the Security registers functional group.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations SCR_EL3 is mapped to AArch32 register SCR. See *4.5.32 Secure Configuration Register* on page 4-263.

Attributes SCR_EL3 is a 32-bit register.

The following figure shows the SCR_EL3 bit assignments.

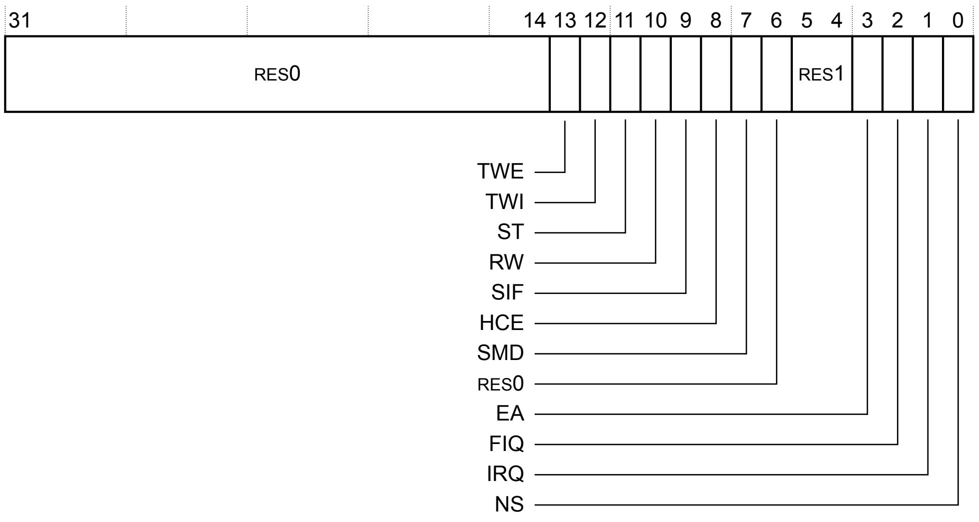

Figure 4-41 SCR_EL3 bit assignments

The following table shows the SCR_EL3 bit assignments.

Table 4-89 SCR_EL3 bit assignments

Bits	Name	Function
[31:14]	-	Reserved, RES0.
[13]	TWE	Traps `WFE` instructions. The possible values are: `0` `WFE` instructions are not trapped. This is the reset value. `1` `WFE` instructions executed in AArch32 or AArch64 from EL2, EL1 or EL0 are trapped to EL3 if the instruction would otherwise cause suspension of execution, that is if: • The event register is not set. • There is not a pending WFE wakeup event. • The instruction is not trapped at EL2 or EL1. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

Table 4-89 SCR_EL3 bit assignments (continued)

Bits	Name	Function
[12]	TWI	Traps `WFI` instructions. The possible values are: `0` `WFI` instructions are not trapped. This is the reset value. `1` `WFI` instructions executed in AArch32 or AArch64 from EL2, EL1 or EL0 are trapped to EL3 if the instruction would otherwise cause suspension of execution, that is if there is not a pending WFI wakeup event and the instruction is not trapped at EL2 or EL1. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[11]	ST	Enable Secure EL1 access to CNTPS_TVAL_EL1, CNTS_CTL_EL1, and CNTPS_CVAL_EL1 registers. The possible values are: `0` Registers accessible only in EL3. This is the reset value. `1` Registers accessible in EL3 and EL1 when SCR_EL3.NS is 0.
[10]	RW	Register width control for lower Exception levels. The possible values are: `0` Lower levels are all AArch32. This is the reset value. `1` The next lower level is AArch64.
[9]	SIF	Secure Instruction Fetch. When the processor is in Secure state, this bit disables instruction fetches from Non-secure memory. The possible values are: `0` Secure state instruction fetches from Non-secure memory are permitted. This is the reset value. `1` Secure state instruction fetches from Non-secure memory are not permitted.
[8]	HCE	Hyp Call enable. This bit enables the use of `HVC` instructions. The possible values are: `0` The `HVC` instruction is UNDEFINED at all Exception levels. This is the reset value. `1` The `HVC` instruction is enabled at EL1, EL2 or EL3.
[7]	SMD	`SMC` instruction disable. The possible values are: `0` The `SMC` instruction is enabled at EL1, EL2, and EL3. This is the reset value. `1` The `SMC` instruction is UNDEFINED at all Exception levels. At EL1, in the Non-secure state, the HCR_EL2.TSC bit has priority over this control.
[6]	-	Reserved, RES0.
[5:4]	-	Reserved, RES1.
[3]	EA	External abort and SError interrupt Routing. This bit controls which mode takes external aborts. The possible values are: `0` External aborts and SError Interrupts while executing at Exception levels other than EL3 are not taken in EL3. This is the reset value. `1` External aborts and SError Interrupts while executing at all Exception levels are taken in EL3.
[2]	FIQ	Physical FIQ Routing. The possible values are: `0` Physical FIQ while executing at Exception levels other than EL3 are not taken in EL3. This is the reset value. `1` Physical FIQ while executing at all Exception levels are taken in EL3.

Table 4-89 SCR_EL3 bit assignments (continued)

Bits	Name	Function
[1]	IRQ	Physical IRQ Routing. The possible values are: `0` Physical IRQ while executing at Exception levels other than EL3 are not taken in EL3. This is the reset value. `1` Physical IRQ while executing at all Exception levels are taken in EL3.
[0]	NS	Non-secure bit. The possible values are: `0` EL0 and EL1 are in Secure state, memory accesses from those Exception levels can access Secure memory. This is the reset value. `1` EL0 and EL1 are in Non-secure state, memory accesses from those Exception levels cannot access Secure memory.

To access the SCR_EL3:

```
MRS <Xt>, SCR_EL3 ; Read SCR_EL3 into Xt
MSR SCR_EL3, <Xt> ; Write Xt to SCR_EL3
```

4.3.51 Secure Debug Enable Register

The SDER32_EL3 characteristics are:

Purpose Allows access to the AArch32 register SDER only from AArch64 state. Its value has no effect on execution in AArch64 state.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations SDER32_EL3 is architecturally mapped to AArch32 register SDER. See *4.5.33 Secure Debug Enable Register* on page 4-265.

Attributes SDER32_EL3 is a 32-bit register.

The following figure shows the SDER32_EL3 bit assignments.

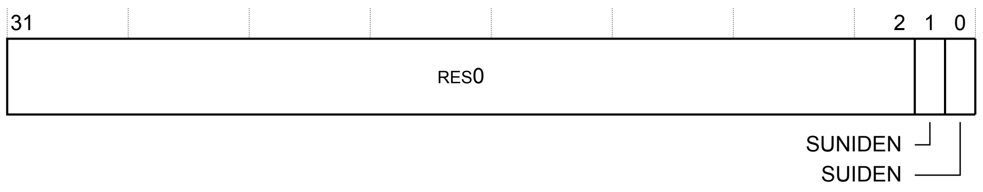

Figure 4-42 SDER32_EL3 bit assignments

The following table shows the SDER32_EL3 bit assignments.

Table 4-90 SDER32_EL3 bit assignments

Bits	Name	Function
[31:2]	-	Reserved, RES0.
[1]	SUNIDEN	Secure User Non-invasive Debug Enable. The possible values are: `0` Non-invasive debug not permitted in Secure EL0 mode. This is the Warm reset value. `1` Non-invasive debug permitted in Secure EL0 mode.
[0]	SUIDEN	Secure User Invasive Debug Enable. The possible values are: `0` Invasive debug not permitted in Secure EL0 mode. This is the Warm reset value. `1` Invasive debug permitted in Secure EL0 mode.

To access the SDER32_EL3:

```
MRS <Xt>, SDER32_EL3 ; Read SDER32_EL3 into Xt
MSR SDER32_EL3, <Xt> ; Write Xt to SDER32_EL3
```

4.3.52 Translation Table Base Register 0, EL1

The TTBR0_EL1 characteristics are:

Purpose 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.

Usage constraints 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

Any of the fields in this register are permitted to be cached in a TLB.

Configurations TTBR0_EL1 is architecturally mapped to AArch32 register TTBR0. See *4.5.42 Translation Table Base Register 0* on page 4-284.

Attributes TTBR0_EL1 is a 64-bit register.

The following figure shows the TTBR0_EL1 bit assignments.

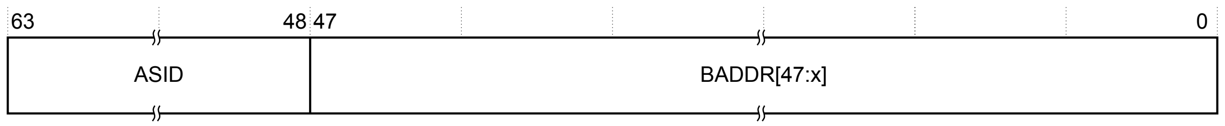

Figure 4-43 TTBR0_EL1 bit assignments

The following table shows the TTBR0_EL1 bit assignments.

Table 4-91 TTBR0_EL1 bit assignments

Bits	Name	Function
[63:48]	ASID	An ASID for the translation table base address. The TCR_EL1.A1 field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.
[47:0]	BADDR[47:x]	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-1:0] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:0] are treated as if all the bits are zero. The value read back from those bits is the value written.

To access the TTBR0_EL1:

```
MRS <Xt>, TTBR0_EL1 ; Read TTBR0_EL1 into Xt
MSR TTBR0_EL1, <Xt> ; Write Xt to TTBR0_EL1
```

4.3.53 Translation Table Base Register 1

The TTBR1_EL1 characteristics are:

Purpose 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.

Usage constraints 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

Any of the fields in this register are permitted to be cached in a TLB.

Configurations TTBR1_EL1 is architecturally mapped to AArch32 register TTBR1 (NS). See *4.5.43 Translation Table Base Register 1* on page 4-287.

Attributes TTBR1_EL1 is a 64-bit register.

The following figure shows the TTBR1_EL1 bit assignments.

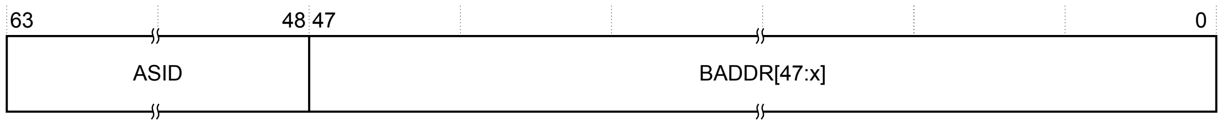

Figure 4-44 TTBR1_EL1 bit assignments

The following table shows the TTBR1_EL1 bit assignments.

Table 4-92 TTBR1_EL1 bit assignments

Bits	Name	Function
[63:48]	ASID	An ASID for the translation table base address. The TCR_EL1.A1 field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.
[47:0]	BADDR[47:x]	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-1:0] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:0] are treated as if all the bits are zero. The value read back from those bits is the value written.

To access the TTBR1_EL1:

```
MRS <Xt>, TTBR1_EL1 ; Read TTBR1_EL1 into Xt
MSR TTBR1_EL1, <Xt> ; Write Xt to TTBR1_EL1
```

4.3.54 Architectural Feature Trap Register, EL3

The CPTR_EL3 characteristics are:

Purpose Controls trapping to EL3 for accesses to CPACR, Trace functionality and registers associated with Advanced SIMD and Floating-point execution. Controls EL3 access to this functionality.

CPTR_EL3 is part of the Security registers functional group.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations There are no configuration notes.

Attributes CPTR_EL3 is a 32-bit register.

The following figure shows the CPTR_EL3 bit assignments.

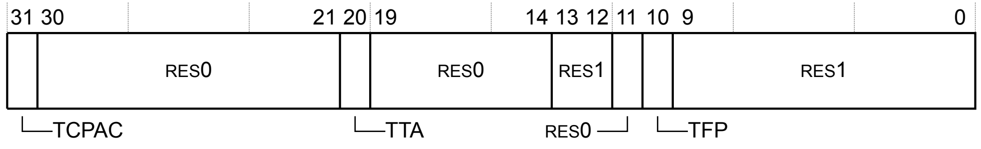

Figure 4-45 CPTR_EL3 bit assignments

The following table shows the CPTR_EL3 bit assignments.

Table 4-93 CPTR_EL3 bit assignments

Bits	Name	Function
[31]	TCPAC	This causes a direct access to the CPACR_EL1 from EL1 or the CPTR_EL2 from EL2 to trap to EL3 unless it is trapped at EL2. The possible values are: `0` Does not cause access to the CPACR_EL1 or CPTR_EL2 to be trapped. `1` Causes access to the CPACR_EL1 or CPTR_EL2 to be trapped.
[30:21]	-	Reserved, RES0.
[20]	TTA	Trap Trace Access. Not implemented. RES0.
[19:14]	-	Reserved, RES0.
[13:12]	-	Reserved, RES1.
[11]	-	Reserved, RES0.
[10]	TFP	This causes instructions that access the registers associated with Advanced SIMD or floating-point execution to trap to EL3 when executed from any exception level, unless trapped to EL1 or EL2. The possible values are: `0` Does not cause any instruction to be trapped. This is the reset value. `1` Causes any instructions that use the registers associated with Advanced SIMD or floating-point execution to be trapped.
[9:0]	-	Reserved, RES1.

To access the CPTR_EL3:

```
MRS <Xt>, CPTR_EL3 ; Read CPTR_EL3 into Xt
MSR CPTR_EL3, <Xt> ; Write Xt to CPTR_EL3
```

4.3.55 Monitor Debug Configuration Register, EL3

The MDCR_EL3 characteristics are:

Purpose Provides configuration options for Security to self-hosted debug.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations MDCR_EL3 is mapped to AArch32 register SDCR. See *4.5.35 Secure Debug Configuration Register* on page 4-267.

Attributes MDCR_EL3 is a 32-bit register.

The following figure shows the MDCR_EL3 bit assignments.

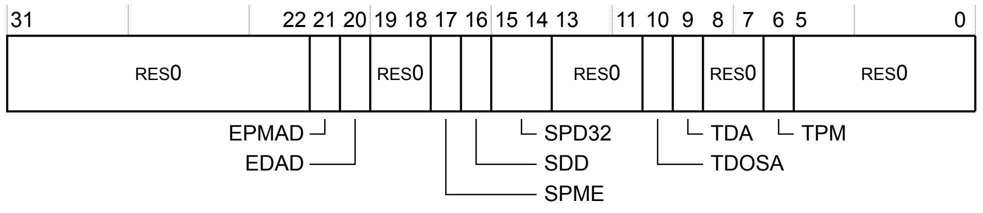

Figure 4-46 MDCR_EL3 bit assignments

The following table shows the MDCR_EL3 bit assignments.

Table 4-94 MDCR_EL3 bit assignments

Bits	Name	Function
[31:22]	-	Reserved, RES0.
[21]	EPMAD	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. The reset value is `0b0`.
[20]	EDAD	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. The reset value is `0b0`.
[19:18]	-	Reserved, RES0.
[17]	SPME	Secure performance monitors enable. This enables event counting exceptions from Secure state. The possible values are: `0` Event counting prohibited in Secure state. This is the reset value. `1` Event counting allowed in Secure state.
[16]	SDD	AArch64 secure debug disable. Disables Software debug exceptions from Secure state if Secure EL1 is using AArch64, other than from Software breakpoint instructions. The possible values are: `0` Debug exceptions from Secure EL0 are enabled, and debug exceptions from Secure EL1 are enabled if MDSCR_EL1.KDE is 1 and PSTATE.D is 0. `1` Debug exceptions from all exception levels in Secure state are disabled. The reset value is UNKNOWN.

Table 4-94 MDCR_EL3 bit assignments (continued)

Bits	Name	Function
[15:14]	SPD32	AArch32 secure privileged debug. Enables or disables debug exceptions from Secure state if Secure EL1 is using AArch32, other than Software breakpoint instructions. The possible values are: `0b00` Legacy mode. Debug exceptions from Secure EL1 are enabled only if `AArch32SelfHostedSecurePrivilegedInvasiveDebugEnabled()`. `0b01` Reserved. `0b10` Secure privileged debug disabled. Debug exceptions from Secure EL1 are disabled. `0b11` Secure privileged debug enabled. Debug exceptions from Secure EL1 are enabled. The reset value is `0b00`.
[13:11]	-	Reserved, RES0.
[10]	TDOSA	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.
[9]	TDA	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.
[8:7]	-	Reserved, RES0.
[6]	TPM	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.
[5:0]	-	Reserved, RES0.

To access the MDCR_EL3:

```
MRS <Xt>, MDCR_EL3 ; Read EL3 Monitor Debug Configuration Register
MSR MDCR_EL3, <Xt> ; Write EL3 Monitor Debug Configuration Register
```

4.3.56 Translation Control Register, EL1

The TCR_EL1 characteristics are:

Purpose 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.

TCR_EL1 is part of the Virtual memory control registers functional group.

Usage constraints 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

Configurations TCR_EL1 is architecturally mapped to AArch32 register TTBCR(NS). See *4.5.44 Translation Table Base Control Register* on page 4-289.

Attributes TCR_EL1 is a 64-bit register.

The following figure shows the TCR_EL1 bit assignments.

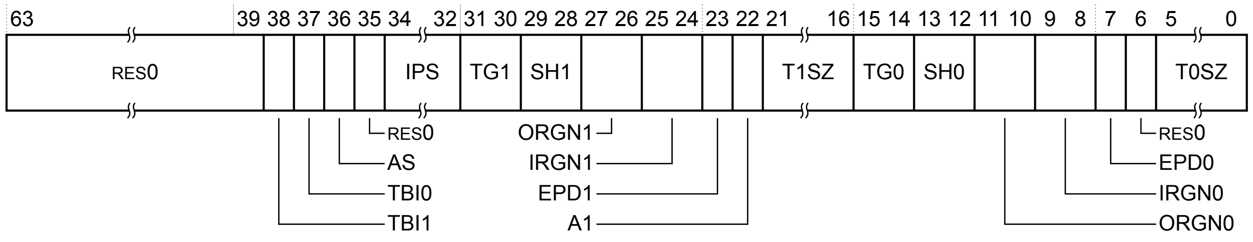

Figure 4-47 TCR_EL1 bit assignments

The following table shows the TCR_EL1 bit assignments.

Table 4-95 TCR_EL1 bit assignments

Bits	Name	Function
[63:39]	-	Reserved, RES0.
[38]	TBI1	Top Byte Ignored. Indicates whether the top byte of the input address is used for address match for the TTBR1_EL1 region. The possible values are: `0` Top byte used in the address calculation. `1` Top byte ignored in the address calculation.
[37]	TBI0	Top Byte Ignored. Indicates whether the top byte of the input address is used for address match for the TTBR0_EL1 region. The possible values are: `0` Top byte used in the address calculation. `1` Top byte ignored in the address calculation.
[36]	AS	ASID size. The possible values are: `0` 8-bit. `1` 16-bit.
[35]	-	Reserved, RES0.
[34:32]	IPS	Intermediate Physical Address Size. The possible values are: `0b000` 32 bits, 4GB. `0b001` 36 bits, 64GB. `0b010` 40 bits, 1TB. All other values are reserved.

Table 4-95 TCR_EL1 bit assignments (continued)

Bits	Name	Function
[31:30]	TG1	TTBR1_EL1 granule size. The possible values are: `0b01` 16KB. `0b10` 4KB. `0b11` 64KB. All other values are not supported.
[29:28]	SH1	Shareability attribute for memory associated with translation table walks using TTBR1_EL1. The possible values are: `0b00` Non-shareable. `0b01` Reserved. `0b10` Outer shareable. `0b11` Inner shareable.
[27:26]	ORGN1	Outer cacheability attribute for memory associated with translation table walks using TTBR1_EL1. The possible values are: `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.
[25:24]	IRGN1	Inner cacheability attribute for memory associated with translation table walks using TTBR1_EL1. The possible values are: `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.
[23]	EPD1	Translation table walk disable for translations using TTBR1_EL1. Controls whether a translation table walk is performed on a TLB miss for an address that is translated using TTBR1_EL1. The possible values are: `0` Perform translation table walk using TTBR1_EL1. `1` A TLB miss on an address translated from TTBR1_EL1 generates a Translation fault. No translation table walk is performed.
[22]	A1	Selects whether TTBR0_EL1 or TTBR1_EL1 defines the ASID. The possible values are: `0` TTBR0_EL1.ASID defines the ASID. `1` TTBR1_EL1.ASID defines the ASID.
[21:16]	T1SZ	Size offset of the memory region addressed by TTBR1_EL1. The region size is $2^{(64\text{-T1SZ})}$ bytes.

Table 4-95 TCR_EL1 bit assignments (continued)

Bits	Name	Function
[15:14]	TG0	TTBR0_EL1 granule size. The possible values are: `0b00` 4KB. `0b01` 64KB. `0x10` 16KB. All other values are not supported.
[13:12]	SH0	Shareability attribute for memory associated with translation table walks using TTBR0_EL1. The possible values are: `0b00` Non-shareable. `0b01` Reserved. `0b10` Outer shareable. `0b11` Inner shareable.
[11:10]	ORGN0	Outer cacheability attribute for memory associated with translation table walks using TTBR0_EL1. The possible values are: `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.
[9:8]	IRGN0	Inner cacheability attribute for memory associated with translation table walks using TTBR0_EL1. The possible values are: `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.
[7]	EPD0	Translation table walk disable for translations using TTBR0_EL1. Controls whether a translation table walk is performed on a TLB miss for an address that is translated using TTBR0_EL1. The possible values are: `0` Perform translation table walk using TTBR0_EL1. `1` A TLB miss on an address translated from TTBR0_EL1 generates a Translation fault. No translation table walk is performed.
[6]	-	Reserved, RES0.
[5:0]	T0SZ	Size offset of the memory region addressed by TTBR0_EL1. The region size is $2^{(64\text{-T0SZ})}$ bytes.

To access the TCR_EL1:

```
MRS <Xt>, TCR_EL1 ; Read TCR_EL1 into Xt
MSR TCR_EL1, <Xt> ; Write Xt to TCR_EL1
```

4.3.57 Translation Control Register, EL2

The TCR_EL2 characteristics are:

Purpose Controls translation table walks required for stage 1 translation of a memory access from EL2 and holds cacheability and shareability information.

TCR_EL2 is part of:
- The Virtual memory control registers functional group.
- The Hypervisor and virtualization registers functional group.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations TCR_EL2 is architecturally mapped to AArch32 register HCTR. See *4.5.45 Hyp Translation Control Register* on page 4-293.

Attributes TCR_EL2 is a 32-bit register.

The following figure shows the TCR_EL2 bit assignments.

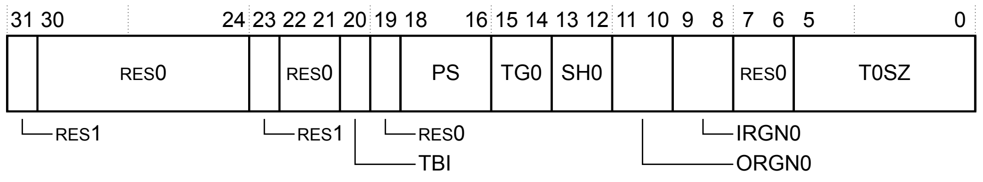

Figure 4-48 TCR_EL2 bit assignments

The following table shows the TCR_EL2 bit assignments.

Table 4-96 TCR_EL2 bit assignments

Bits	Name	Function
[31]	-	Reserved, RES1.
[30:24]	-	Reserved, RES0.
[23]	-	Reserved, RES1.
[22:21]	-	Reserved, RES0.
[20]	TBI	Top Byte Ignored. Indicates whether the top byte of the input address is used for address match. The possible values are: `0` Top byte used in the address calculation. `1` Top byte ignored in the address calculation.
[19]	-	Reserved, RES0.
[18:16]	PS	Physical address size. The possible values are: `0b000` 32 bits, 4GB. `0b001` 36 bits, 64GB. `0b010` 40 bits, 1TB. Other values are reserved.

Table 4-96 TCR_EL2 bit assignments (continued)

Bits	Name	Function
[15:14]	TG0	TTBR0_EL2 granule size. The possible values are: `0b00` 4KB. `0b01` 64KB. `0x10` 16KB. All other values are not supported.
[13:12]	SH0	Shareability attribute for memory associated with translation table walks using TTBR0_EL2. The possible values are: `0b00` Non-shareable. `0b01` Reserved. `0b10` Outer shareable. `0b11` Inner shareable.
[11:10]	ORGN0	Outer cacheability attribute for memory associated with translation table walks using TTBR0_EL2. The possible values are: `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.
[9:8]	IRGN0	Inner cacheability attribute for memory associated with translation table walks using TTBR0_EL2. The possible values are: `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.
[7:6]	-	Reserved, RES0.
[5:0]	T0SZ	Size offset of the memory region addressed by TTBR0_EL2. The region size is $2^{(64\text{-T0SZ})}$ bytes.

To access the TCR_EL2:

```
MRS <Xt>, TCR_EL2 ; Read EL2 Translation Control Register
MSR TCR_EL2, <Xt> ; Write EL2 Translation Control Register
```

4.3.58 Virtualization Translation Control Register, EL2

The VTCR_EL2 characteristics are:

Purpose Controls the translation table walks required for the stage 2 translation of memory accesses from Non-secure EL0 and EL1, and holds cacheability and shareability information for the accesses.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Any of the bits in VTCR_EL2 are permitted to be cached in a TLB.

Configurations VTCR_EL2 is architecturally mapped to AArch32 register VTCR. See *4.5.46 Virtualization Translation Control Register* on page 4-294.

Attributes VTCR_EL2 is a 32-bit register.

The following figure shows the VTCR_EL2 bit assignments.

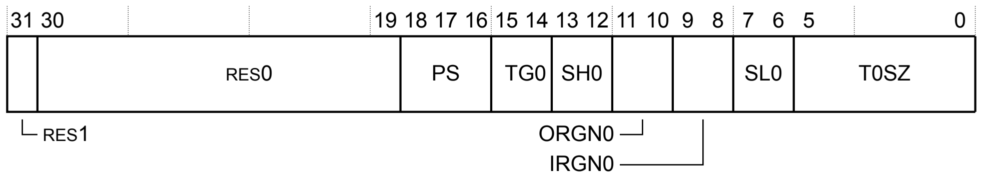

Figure 4-49 VTCR_EL2 bit assignments

The following table shows the VTCR_EL2 bit assignments.

Table 4-97 VTCR_EL2 bit assignments

Bits	Name	Function
[31]	-	Reserved, RES1.
[30:19]	-	Reserved, RES0.
[18:16]	PS	Physical Address Size. The possible values are: `0b000` 32 bits, 4GB. `0b001` 36 bits, 64GB. `0b010` 40 bits, 1TB. All other values are reserved.
[15:14]	TG0	Granule size for the corresponding VTTBR_EL2. `0b00` 4KB. `0b01` 64KB. `0b11` Reserved. All other values are not supported.
[13:12]	SH0	Shareability attribute for memory associated with translation table walks using VTTBR_EL2. `0b00` Non-shareable. `0b01` Reserved. `0b10` Outer Shareable. `0b11` Inner Shareable.

Table 4-97 VTCR_EL2 bit assignments (continued)

Bits	Name	Function
[11:10]	ORGN0	Outer cacheability attribute for memory associated with translation table walks using VTTBR_EL2. `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.
[9:8]	IRGN0	Inner cacheability attribute for memory associated with translation table walks using VTTBR_EL2. `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.
[7:6]	SL0	Starting level of the VTCR_EL2 addressed region.
[5:0]	T0SZ	The size offset of the memory region addressed by VTTBR_EL2. The region size is $2^{(64\text{-T0SZ})}$ bytes.

To access the VTCR_EL2:

```
MRS <Xt>, VTCR_EL2 ; Read VTCR_EL2 into Xt
MSR VTCR_EL2, <Xt> ; Write Xt to VTCR_EL2
```

4.3.59 Domain Access Control Register

The DACR32_EL2 characteristics are:

Purpose Allows access to the AArch32 DACR register from AArch64 state only. Its value has no effect on execution in AArch64 state.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations DACR32_EL2 is architecturally mapped to AArch32 register DACR (NS). See *4.5.47 Domain Access Control Register* on page 4-296.

Attributes DACR32_EL2 is a 32-bit register.

The following figure shows the DACR32_EL2 bit assignments.

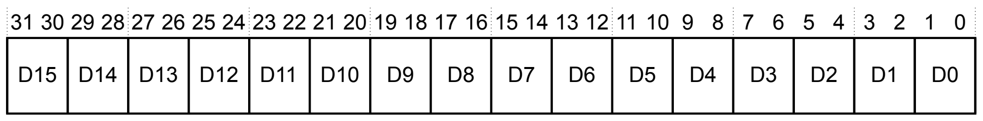

Figure 4-50 DACR32_EL2 bit assignments

The following table shows the DACR32_EL2 bit assignments.

Table 4-98 DACR32_EL2 bit assignments

Bits	Name	Function
[31:0]	D<*n*>, bits [2*n*+1:2*n*], for *n* = 0 to 15	Domain *n* access permission, where *n* = 0 to 15. Permitted values are: `0b00` No access. Any access to the domain generates a Domain fault. `0b01` Client. Accesses are checked against the permission bits in the translation tables. `0b11` Manager. Accesses are not checked against the permission bits in the translation tables. The value `0b10` is reserved.

To access the DACR32_EL2:

```
MRS <Xt>, DACR32_EL2 ; Read DACR32_EL2 into Xt
MSR DACR32_EL2, <Xt> ; Write Xt to DACR32_EL2
```

4.3.60 Translation Table Base Register 0, EL3

The TTBR0_EL3 characteristics are:

Purpose Holds the base address of the translation table for the stage 1 translation of memory accesses from EL3.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations TTBR0_EL3 is mapped to AArch32 register TTBR0 (S). See *4.5.42 Translation Table Base Register 0* on page 4-284.

Attributes TTBR0_EL3 is a 64-bit register.

The following figure shows the TTBR0_EL3 bit assignments.

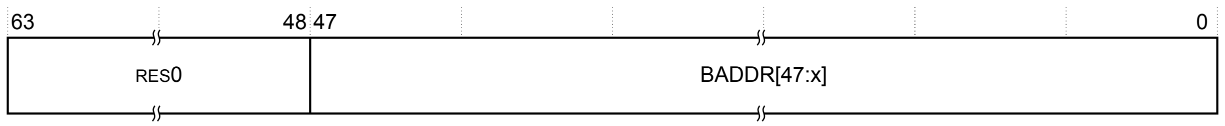

Figure 4-51 TTBR0_EL3 bit assignments

The following table shows the TTBR0_EL3 bit assignments.

Table 4-99 TTBR0_EL3 bit assignments

Bits	Name	Function
[63:48]	-	Reserved, RES0.
[47:0]	BADDR[47:x]	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-1:0] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:0] are treated as if all the bits are zero. The value read back from those bits is the value written.

To access the TTBR0_EL3:

```
MRS <Xt>, TTBR0_EL3 ; Read TTBR0_EL3 into Xt
MSR TTBR0_EL3, <Xt> ; Write Xt to TTBR0_EL3
```

4.3.61 Translation Control Register, EL3

The TCR_EL3 characteristics are:

Purpose Controls translation table walks required for stage 1 translation of memory accesses from EL3 and holds cacheability and shareability information for the accesses.

TCR_EL3 is part of the Virtual memory control registers functional group.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations TCR_EL3 is mapped to AArch32 register TTBR(S).

Attributes TCR_EL3 is a 32-bit register.

The following figure shows the TCR_EL3 bit assignments.

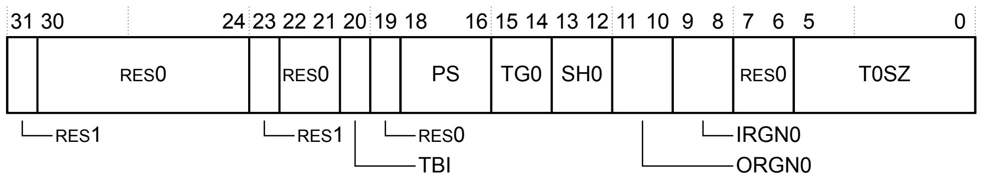

Figure 4-52 TCR_EL3 bit assignments

The following table shows the TCR_EL3 bit assignments.

Table 4-100 TCR_EL3 bit assignments

Bits	Name	Function
[31]	-	Reserved, RES1.
[30:24]	-	Reserved, RES0.

Table 4-100 TCR_EL3 bit assignments (continued)

Bits	Name	Function
[23]	-	Reserved, RES1.
[22:21]	-	Reserved, RES0.
[20]	TBI	Top Byte Ignored. Indicates whether the top byte of the input address is used for address match. The possible values are: `0` Top byte used in the address calculation. `1` Top byte ignored in the address calculation. The reset value is `0`.
[19]	-	Reserved, RES0.
[18:16]	PS	Physical address size. The possible values are: `0b000` 32 bits, 4GB. `0b001` 36 bits, 64GB. `0b010` 40 bits, 1TB. Other values are reserved. The reset value is `0b000`.
[15:14]	TG0	TTBR0_EL3 granule size. The possible values are: `0b00` 4KB. `0b01` 64KB. `0x10` 16KB. All other values are not supported. The reset value is `0b00`.
[13:12]	SH0	Shareability attribute for memory associated with translation table walks using TTBR0_EL3. The possible values are: `0b00` Non-shareable. `0b01` Reserved. `0b10` Outer shareable. `0b11` Inner shareable. The reset value is `0b00`.
[11:10]	ORGN0	Outer cacheability attribute for memory associated with translation table walks using TTBR0_EL3. The possible values are: `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable. The reset value is `0b00`.

Table 4-100 TCR_EL3 bit assignments (continued)

Bits	Name	Function
[9:8]	IRGN0	Inner cacheability attribute for memory associated with translation table walks using TTBR0_EL3. The possible values are: `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable. The reset value is `0b00`.
[7:6]	-	Reserved, RES0.
[5:0]	T0SZ	Size offset of the memory region addressed by TTBR0_EL3. The region size is $2^{(64\text{-T0SZ})}$ bytes. The reset value is `0b000000`.

To access the TCR_EL3:

```
MRS <Xt>, TCR_EL3 ; Read EL3 Translation Control Register
MRS TCR_EL3, <Xt> ; Read EL3 Translation Control Register
```

4.3.62 Auxiliary Memory Attribute Indirection Register, EL1, EL2 and EL3

The processor does not implement AMAIR_EL1, AMAIR_EL2 and AMAIR_EL3, therefore these registers are always RES0.

4.3.63 Auxiliary Fault Status Register 0, EL1, EL2 and EL3

The processor does not implement AFSR0_EL1, AFSR0_EL2 and AFSR0_EL3, therefore these registers are always RES0.

4.3.64 Auxiliary Fault Status Register 1, EL1, EL2 and EL3

The processor does not implement AFSR1_EL1, AFSR1_EL2 and AFSR1_EL3, therefore these registers are always RES0.

4.3.65 Exception Syndrome Register, EL1

The ESR_EL1 characteristics are:

Purpose Holds syndrome information for an exception taken to EL1.

Usage constraints 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

Configurations ESR_EL1 is architecturally mapped to AArch32 register DFSR (NS). See *4.5.50 Data Fault Status Register* on page 4-300.

Attributes ESR_EL1 is a 32-bit register.

The following figure shows the ESR_EL1 bit assignments.

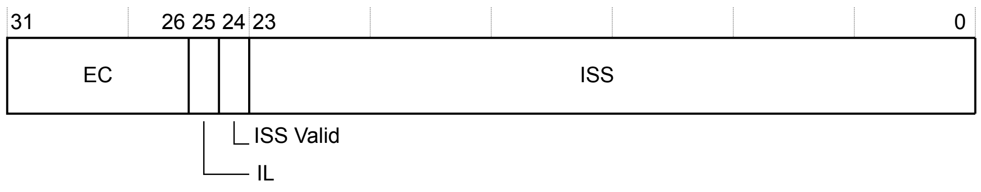

Figure 4-53 ESR_EL1 bit assignments

The following table shows the ESR_EL1 bit assignments.

Table 4-101 ESR_EL1 bit assignments

Bits	Name	Function
[31:26]	EC	Exception Class. Indicates the reason for the exception that this register holds information about.
[25]	IL	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.
[24]	ISS Valid	Syndrome valid. The possible values are: `0` ISS not valid, ISS is RES0. `1` ISS valid.
[23:0]	ISS	Syndrome information.

When the EC field is `0x2F`, indicating an SError interrupt has occurred, the ISS field contents are IMPLEMENTATION DEFINED. The following table shows the definition of the ISS field contents for the Cortex-A73 processor.

Table 4-102 ISS field contents for the Cortex-A73 processor

ISS[23:22]	ISS[1:0]	Description
`0b00`	`0b00`	DECERR on external access
`0b00`	`0b01`	Double-bit error detected on dirty line in L2 cache
`0b00`	`0b10`	SLVERR on external access
`0b01`	`0b00`	**nSEI**, or **nVSEI** in a guest OS, asserted
`0b01`	`0b01`	**nREI** asserted

Note

ISS is not valid when a virtual SError is generated on a Guest OS using the HCR_EL2.VSE bit.

To access the ESR_EL1:

```
MRS <Xt>, ESR_EL1 ; Read EL1 Exception Syndrome Register
MSR ESR_EL1, <Xt> ; Write EL1 Exception Syndrome Register
```

4.3.66 Instruction Fault Status Register, EL2

The IFSR32_EL2 characteristics are:

Purpose Allows access to the AArch32 IFSR register from AArch64 state only. Its value has no effect on execution in AArch64 state.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations IFSR32_EL2 is architecturally mapped to AArch32 register IFSR(NS). See *4.5.51 Instruction Fault Status Register* on page 4-304.

Attributes IFSR32_EL2 is a 32-bit register.

There are two formats for this register. The current translation table format determines which format of the register is used. This section describes:
- *IFSR when using the Short-descriptor translation table format* on page 4-304.
- *IFSR when using the Long-descriptor translation table format* on page 4-305.

IFSR32_EL2 when using the Short-descriptor translation table format

The following figure shows the IFSR32_EL2 bit assignments when using the Short-descriptor translation table format.

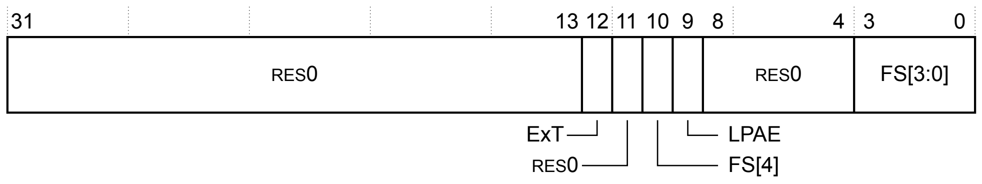

Figure 4-54 IFSR32_EL2 bit assignments for Short-descriptor translation table format

The following table shows the IFSR32_EL2 bit assignments when using the Short-descriptor translation table format.

Table 4-103 IFSR32_EL2 bit assignments for Short-descriptor translation table format

Bits	Name	Function
[31:13]	-	Reserved, RES0.
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: `0` External abort marked as DECERR. `1` External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11]	-	Reserved, RES0.
[10]	FS[4]	Part of the Fault Status field. See bits [3:0] in this table.

Table 4-103 IFSR32_EL2 bit assignments for Short-descriptor translation table format (continued)

Bits	Name	Function
[9]	LPAE	On taking a Data Abort exception, this bit is set as follows: `0` Using the Short-descriptor translation table formats. `1` Using the Long-descriptor translation table formats. Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.
[8:4]	-	Reserved, RES0.
[3:0]	FS[3:0]	Fault Status bits. This field indicates the type of exception generated. Interpreted with bit [10]. Any encoding not listed is reserved. `0b00010` Debug event. `0b00011` Access flag fault, section. `0b00101` Translation fault, section. `0b00110` Access flag fault, page. `0b00111` Translation fault, page. `0b01000` Synchronous external abort, non-translation. `0b01001` Domain fault, section. `0b01011` Domain fault, page. `0b01100` Synchronous external abort on translation table walk, first level. `0b01101` Permission Fault, Section. `0b01110` Synchronous external abort on translation table walk, second Level. `0b01111` Permission fault, page. `0b10000` TLB conflict abort. `0b11001` Synchronous parity error on memory access. `0b11100` Synchronous parity error on translation table walk, first level. `0b11110` Synchronous parity error on translation table walk, second level.

IFSR32_EL2 when using the Long-descriptor translation table format

The following figure shows the IFSR32_EL2 bit assignments when using the Long-descriptor translation table format.

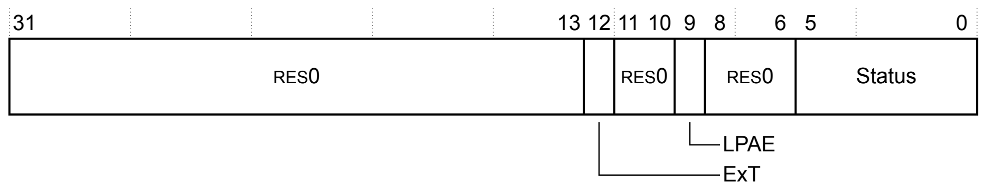

Figure 4-55 IFSR32_EL2 bit assignments for Long-descriptor translation table format

The following table shows the IFSR32_EL2 bit assignments when using the Long-descriptor translation table format.

Table 4-104 IFSR32_EL2 bit assignments for Long-descriptor translation table format

Bits	Name	Function
[31:13]	-	Reserved, RES0.
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: `0` External abort marked as DECERR. `1` External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11:10]	-	Reserved, RES0.
[9]	LPAE	On taking a Data Abort exception, this bit is set as follows: `0` Using the Short-descriptor translation table formats. `1` Using the Long-descriptor translation table formats. Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.
[8:6]	-	Reserved, RES0.
[5:0]	Status	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved. `0b000000` Address size fault in TTBR0 or TTBR1. `0b0001LL` Translation fault, LL bits indicate level. `0b0010LL` Access fault flag, LL bits indicate level. `0b0011LL` Permission fault, LL bits indicate level. `0b010000` Synchronous external abort. `0b0101LL` Synchronous external abort on translation table walk, LL bits indicate level. `0b011000` Synchronous parity error on memory access. `0b0111LL` Synchronous parity error on memory access on translation table walk, LL bits indicate level. `0b100001` Alignment fault. `0b100010` Debug event. `0b110000` TLB conflict abort.

The following table shows how the LL bits in the Status field encode the lookup level associated with the MMU fault.

Table 4-105 Encodings of LL bits associated with the MMU fault

Bits	Meaning
`0b00`	Reserved
`0b01`	Level 1
`0b10`	Level 2
`0b11`	Level 3

Note

If a Data Abort exception is generated by an instruction cache maintenance operation when the Long-descriptor translation table format is selected, the fault is reported as a Cache Maintenance fault in the

DFSR or HSR with the appropriate Fault Status code. For such exceptions reported in the DFSR, the corresponding IFSR32_EL2 is UNKNOWN.

To access the IFSR32_EL2:

```
MRS <Xt>, IFSR32_EL2 ; Read IFSR32_EL2 into Xt
MSR IFSR32_EL2, <Xt> ; Write Xt to IFSR32_EL2
```

Register access is encoded as follows:

Table 4-106 IFSR32_EL2 access encoding

op0	op1	CRn	CRm	op2
11	000	0101	0000	001

4.3.67 Exception Syndrome Register, EL2

The ESR_EL2 characteristics are:

Purpose Holds syndrome information for an exception taken to EL2.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations ESR_EL2 is architecturally mapped to AArch32 register HSR. See *4.5.56 Hyp Syndrome Register* on page 4-307.

Attributes ESR_EL2 is a 32-bit register.

The following figure shows the ESR_EL2 bit assignments.

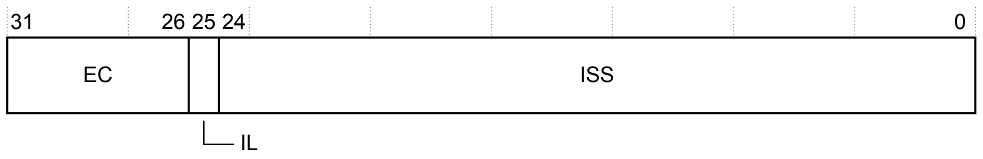

Figure 4-56 ESR_EL2 bit assignments

The following table shows the ESR_EL2 bit assignments.

Table 4-107 ESR_EL2 bit assignments

Bits	Name	Function
[31:26]	EC	Exception Class. Indicates the reason for the exception that this register holds information about.
[25]	IL	Instruction Length for synchronous exceptions. The possible values are: `0` 16-bit. `1` 32-bit.
[24:0]	ISS	Syndrome information.

When the EC field is `0x2F`, indicating an SError interrupt has occurred, the ISS field contents are IMPLEMENTATION DEFINED. The following table shows the definition of the ISS field contents for the Cortex-A73 processor.

Table 4-108 ISS field contents for the Cortex-A73 processor

ISS[23:22]	ISS[1:0]	Description
`0b00`	`0b00`	DECERR on external access
`0b00`	`0b01`	Double-bit error detected on dirty line in L2 cache
`0b00`	`0b10`	SLVERR on external access
`0b01`	`0b00`	**nSEI** or **nVSEI** in a guest OS, asserted
`0b01`	`0b01`	**nREI** asserted

Note

ISS is not valid when a virtual SError is generated on a Guest OS using the HCR_EL2.VSE bit.

To access the ESR_EL2:

```
MRS <Xt>, ESR_EL2 ; Read EL1 Exception Syndrome Register
MSR ESR_EL2, <Xt> ; Write EL1 Exception Syndrome Register
```

4.3.68 Exception Syndrome Register, EL3

The ESR_EL3 characteristics are:

Purpose Holds syndrome information for an exception taken to EL3.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations ESR_EL3 is mapped to AArch32 register DFSR(S). See *4.5.50 Data Fault Status Register* on page 4-300.

Attributes ESR_EL3 is a 32-bit register.

The following figure shows the ESR_EL3 bit assignments.

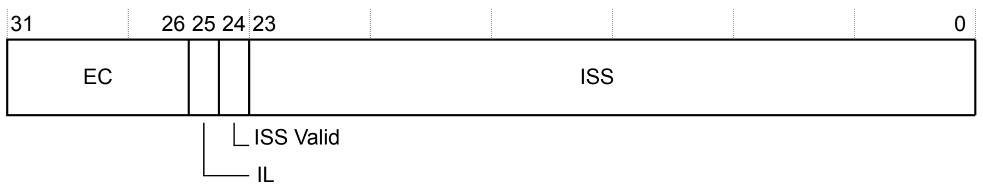

Figure 4-57 ESR_EL3 bit assignments

The following table shows the ESR_EL3 bit assignments.

Table 4-109 ESR_EL3 bit assignments

Bits	Name	Function
[31:26]	EC	Exception Class. Indicates the reason for the exception that this register holds information about.
[25]	IL	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.
[24]	ISS Valid	Syndrome valid. The possible values are: `0` ISS not valid, ISS is RES0. `1` ISS valid.
[23:0]	ISS	Syndrome information.

When the EC field is `0x2F`, indicating an SError interrupt has occurred, the ISS field contents are IMPLEMENTATION DEFINED. The following table shows the definition of the ISS field contents for the Cortex-A73 processor.

Table 4-110 ISS field contents for the Cortex-A73 processor

ISS[23:22]	ISS[1:0]	Description
`0b00`	`0b00`	DECERR on external access
`0b00`	`0b01`	Double-bit error detected on dirty line in L2 cache
`0b00`	`0b10`	SLVERR on external access
`0b01`	`0b00`	**nSEI**, or **nVSEI** in a guest OS, asserted
`0b01`	`0b01`	**nREI** asserted

Note

ISS is not valid when a virtual SError is generated on a Guest OS using the HCR_EL2.VSE bit.

To access the ESR_EL3:

```
MRS <Xt>, ESR_EL3 ; Read EL3 Exception Syndrome Register
MSR ESR_EL3, <Xt> ; Write EL3 Exception Syndrome Register
```

4.3.69 Fault Address Register, EL1

The FAR_EL1 characteristics are:

Purpose Holds the faulting Virtual Address for all synchronous instruction or data aborts, or exceptions from a misaligned PC or a Watchpoint debug event, taken to EL1.

Usage constraints 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

Configurations FAR_EL1[31:0] is architecturally mapped to AArch32 register DFAR (NS). See *4.5.57 Data Fault Address Register* on page 4-309.

FAR_EL1[63:32] is architecturally mapped to AArch32 register IFAR (NS). See *4.5.58 Instruction Fault Address Register* on page 4-309.

Attributes FAR_EL1 is a 64-bit register.

The following figure shows the FAR_EL1 bit assignments.

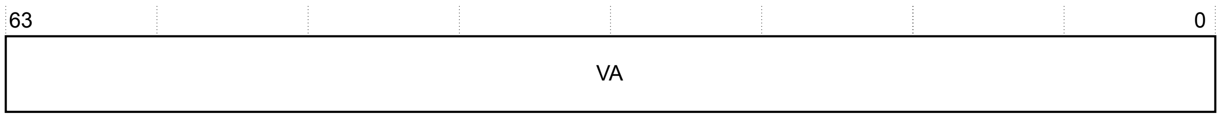

Figure 4-58 FAR_EL1 bit assignments

The following table shows the FAR_EL1 bit assignments.

Table 4-111 FAR_EL1 bit assignments

Bits	Name	Function
[63:0]	VA	The faulting Virtual Address for all synchronous instruction or data aborts, or an exception from a misaligned PC, taken in EL1. If a memory fault that sets the FAR is generated from one of the data cache instructions, this field holds the address specified in the register argument of the instruction.

To access the FAR_EL1:

```
MRS <Xt>, FAR_EL1 ; Read EL1 Fault Address Register
MSR FAR_EL1, <Xt> ; Write EL1 Fault Address Register
```

4.3.70 Fault Address Register, EL2

The FAR_EL2 characteristics are:

Purpose Holds the faulting Virtual Address for all synchronous instruction or data aborts, or exceptions from a misaligned PC or a Watchpoint debug event, taken to EL2.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations FAR_EL2[31:0] is architecturally mapped to AArch32 registers:

- HDFAR. See *4.5.59 Hyp Data Fault Address Register* on page 4-310.
- DFAR (S). See *4.5.57 Data Fault Address Register* on page 4-309.

FAR_EL2[63:32] is architecturally mapped to AArch32 registers:

- HIFAR. See *4.5.60 Hyp Instruction Fault Address Register* on page 4-311.
- IFAR (S). See *4.5.58 Instruction Fault Address Register* on page 4-309.

Attributes FAR_EL2 is a 64-bit register.

The following figure shows the FAR_EL2 bit assignments.

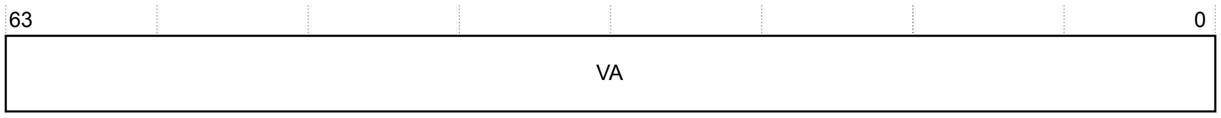

Figure 4-59 FAR_EL2 bit assignments

The following table shows the FAR_EL2 bit assignments.

Table 4-112 FAR_EL2 bit assignments

Bits	Name	Function
[63:0]	VA	The faulting Virtual Address for all synchronous instruction or data aborts, or an exception from a misaligned PC, taken in EL2. If a memory fault that sets the FAR is generated from one of the data cache instructions, this field holds the address specified in the register argument of the instruction.

To access the FAR_EL2:

```
MRS <Xt>, FAR_EL2 ; Read EL2 Fault Address Register
MSR FAR_EL2, <Xt> ; Write EL2 Fault Address Register
```

4.3.71 Hypervisor IPA Fault Address Register, EL2

The HPFAR_EL2 characteristics are:

Purpose Holds the faulting IPA for some aborts on a stage 2 translation taken to EL2.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations HPFAR_EL2[31:0] is mapped to AArch32 register HPFAR. See *4.5.61 Hyp IPA Fault Address Register* on page 4-312.

Attributes HPFAR_EL2 is a 64-bit register.

The following figure shows the HPFAR_EL2 bit assignments.

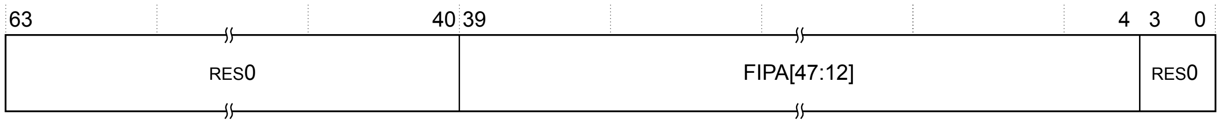

Figure 4-60 HPFAR_EL2 bit assignments

The following table shows the HPFAR_EL2 bit assignments.

Table 4-113 HPFAR_EL2 bit assignments

Bits	Name	Function
[63:40]	-	Reserved, RES0.
[39:4]	FIPA[47:12]	Bits [47:12] of the faulting intermediate physical address. The equivalent upper bits in this field are RES0.
[3:0]	-	Reserved, RES0.

To access the HPFAR_EL2:

```
MRS <Xt>, HPFAR_EL2 ; Read EL2 Fault Address Register
MSR HPFAR_EL2, <Xt> ; Write EL2 Fault Address Register
```

4.3.72 L2 Control Register

The L2CTLR_EL1 characteristics are:

Purpose Provides IMPLEMENTATION DEFINED control options for the L2 memory system.

Usage constraints 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

Note

This register is write accessible in EL1 if:

- ACTLR_EL3.EN_L2CTLR is 1 and
- ACTLR_EL2.EN_L2CTLR is 1 or ACTLR_EL3.EN_EN_L2CTLR is 1 and
- SCR.NS is 0.

If write access is not possible, then L2CTLR_EL1 is trapped to the lowest exception level that denied access (EL2 or EL3).

Configurations L2CTLR_EL1 is architecturally mapped to the AArch32 L2CTLR register. See *4.5.63 L2 Control Register* on page 4-312.:

There is one L2CTLR_EL1 for all cores in the Cortex-A73 processor.

Attributes L2CTLR_EL1 is a 32-bit register.

The following figure shows the L2CTLR_EL1 bit assignments.

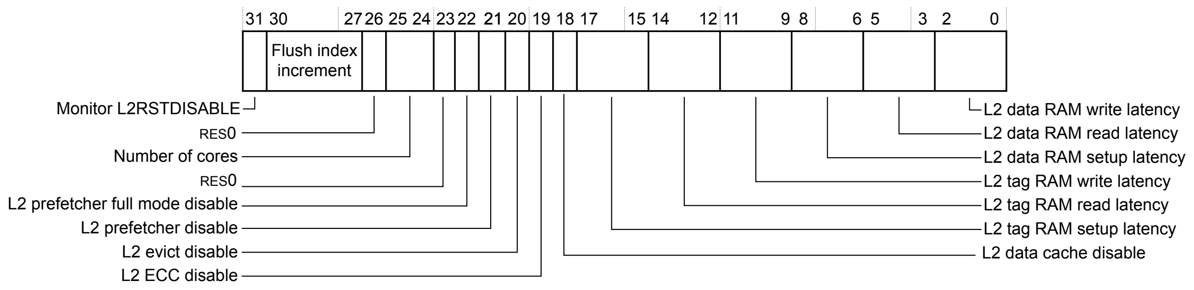

Figure 4-61 L2CTLR_EL1 bit assignments

The following table shows the L2CTLR_EL1 bit assignments.

Table 4-114 L2CTLR_EL1 bit assignments

Bits	Name	Function
[31]	L2RSTDISABLE	Monitor L2RSTDISABLE.
[30:27]	Flush index increment	L2 cache clean and clean and invalidate all operations loop on all ways and on all sets. Each set is referenced by an index. These operations evict cache lines marked as dirty. This corresponds to the cache cleaning algorithm specified in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. The Flush index increment field controls how the index increments. The reset value `0b0000` gives an increment of 1, this evicts dirty lines on index 0,1,2,3,4,5, ... (modulo the total number of sets). Setting the field to `0b0100` gives an increment value 15, which evicts dirty lines on index 0,15,30,45, ... (modulo the total number of sets). `0b0000` 1. This is the reset value. `0b0001` 1. `0b0010` 3. `0b0011` 7. `0b0100` 15. `0b0101` 31. `0b0110` 63. ... n (2^n)-1. `0b1100` 4095. `0b1101` 8191. `0b1110` 8191. `0b1111` 8191. You can program the flush index increment field to optimize cache flush speed for system configuration of address decode supporting a complex multi-channel multi-rank memory system.
[26]	-	Reserved, RES0.
[25:24]	Number of cores	Number of cores present: `0b00` One core, core 0. `0b01` Two cores, core 0 and core 1. `0b10` Three cores, cores 0 to 2. `0b11` Four cores, cores 0 to 3. These bits are read-only and the value of this field is set to the number of cores present in the configuration.
[23]	-	Reserved, RES0.
[22]	L2 prefetcher full mode disable	Disables the full mode of the L2 prefetcher: `0` Enables full mode. `1` Disables full mode. The reset value is 0. **Note** This bit has no effect when the L2 prefetcher disable bit is set to 1.

Table 4-114 L2CTLR_EL1 bit assignments (continued)

Bits	Name	Function
[21]	L2 prefetcher disable	Disables the L2 prefetcher: `0` Enables prefetcher. `1` Disables prefetcher. The reset value is 0.
[20]	L2 evict disable	Disables evict information sent to interconnect: `0` Enables evict information. `1` Disables evict information. The reset value is 0.
[19]	L2 ECC disable	Disables L2 ECC protection: `0` Enables ECC protection. `1` Disables ECC protection. The reset value is 0.
[18]	L2 data cache disable[ad]	Disables L2 data cache: `0` Enables L2 data cache. `1` Disables L2 data cache. The reset value is 0.
[17:15]	L2 tag RAM setup latency	L2 tag RAM setup latency: `<n>` <n+1>-cycle setup delay from L2 tag RAMs.
[14:12]	L2 tag RAM read latency	L2 tag RAM read latency: `<n>` <n+1>-cycle read delay from L2 tag RAMs.
[11:9]	L2 tag RAM write latency	L2 tag RAM write latency: `<n>` <n+1>-cycle write delay from L2 tag RAMs.
[8:6]	L2 data RAM setup latency	L2 data RAM setup latency: `<n>` <n+1>-cycle setup delay from L2 data RAMs.
[5:3]	L2 data RAM read latency	L2 data RAM read latency: `<n>` <n+1>-cycle read delay from L2 data RAMs.
[2:0]	L2 data RAM write latency	L2 data RAM write latency: `<n>` <n+1>-cycle write delay from L2 data RAMs.

To access the L2CTLR_EL1:

```
MRS <Xt>, S3_1_C11_C0_2 ; Read L2CTLR_EL1 into Xt
MSR S3_1_C11_C0_2, <Xt>; Write Xt to L2CTLR_EL1
```

[ad] The L2 cache must never be disabled in a multi-cluster configuration.

4.3.73 L2 Extended Control Register

The L2ECTLR_EL1 characteristics are:

Purpose Provides additional IMPLEMENTATION DEFINED control options for the L2 memory system. This register is used for dynamically changing, but implementation specific, control bits.

Usage constraints 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

The L2ECTLR_EL1 can be written dynamically.

Note

This register is write accessible in EL1 if:
- ACTLR_EL3.EN_L2CTLR is 1 and
- ACTLR_EL2.EN_EN_L2CTLR is 1 or ACTLR_EL3.EN_EN_L2CTLR is 1 and
- SCR.NS is 0.

If write access is not possible, then L2ECTLR_EL1 is trapped to the lowest exception level that denied access (EL2 or EL3).

Configurations L2ECTLR_EL1 is architecturally mapped to the AArch32 L2ECTLR register. See *4.5.64 L2 Extended Control Register* on page 4-315.

There is one copy of this register that is used in both Secure and Non-secure states.

There is only one L2ECTLR_EL1 for all cores in the Cortex-A73 processor.

Attributes L2ECTLR_EL1 is a 32-bit register.

The following figure shows the L2ECTLR_EL1 bit assignments.

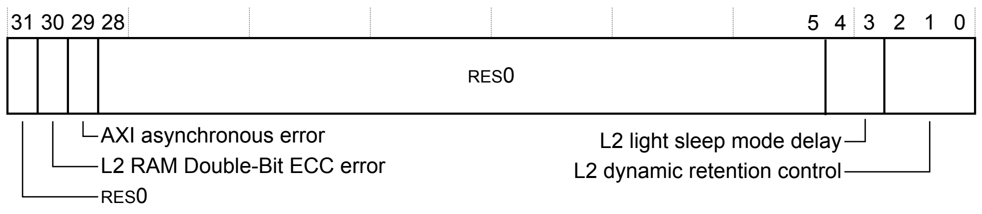
31 30 29 28
5 4 3 2 1 0
RES0
AXI asynchronous error
L2 RAM Double-Bit ECC error
RES0
L2 light sleep mode delay
L2 dynamic retention control

Figure 4-62 L2ECTLR_EL1 bit assignments

The following table shows the L2ECTLR_EL1 bit assignments.

Table 4-115 L2ECTLR_EL1 bit assignments

Bits	Name	Function
[31]	-	Reserved, RES0.
[30]	L2 RAM Double-Bit ECC error	L2 RAM Double-Bit ECC error indication. The possible values are: `0` No pending Double-Bit ECC error. `1` A fatal Double-Bit ECC error has occurred. A write of 0 clears this bit. A write of 1 is ignored.
[29]	AXI asynchronous error	AXI asynchronous error indication. The possible values are: `0` No pending asynchronous error. `1` An asynchronous error has occurred. A write of 0 clears this bit. A write of 1 is ignored.
[28:5]	-	Reserved, RES0.
[4:3]	L2 light sleep mode delay	L2 light sleep mode delay. The possible values are: `0b00` 0 clock cycles before L2 RAM enters sleep mode. `0b01` 16 clock cycles before L2 RAM enters sleep mode. `0b10` 32 clock cycles before L2 RAM enters sleep mode. `0b11` 64 clock cycles before L2 RAM enters sleep mode.
[2:0]	L2 dynamic retention control	L2 Data dynamic retention control. The possible values are: `0b000` L2 dynamic retention disabled. This is the reset value. `0b001` 2 Generic Timer ticks required before retention entry. `0b010` 8 Generic Timer ticks required before retention entry. `0b011` 32 Generic Timer ticks required before retention entry. `0b100` 64 Generic Timer ticks required before retention entry. `0b101` 128 Generic Timer ticks required before retention entry. `0b110` 256 Generic Timer ticks required before retention entry. `0b111` 512 Generic Timer ticks required before retention entry. —— **Note** —— If the **CNTVALUEB[63:0]** bus increments by more than 1024, this bit is not taken into account and the retention entry may never occur. ——————

To access the L2ECTLR_EL1:

```
MRS Rt, S3_1_C11_C0_3; Read L2ECTLR_EL1 into Rt
MSR S3_1_C11_C0_3, Rt; Write Rt to L2ECTLR_EL1
```

4.3.74 Fault Address Register, EL3

The FAR_EL3 characteristics are:

Purpose Holds the faulting Virtual Address for all synchronous instruction or data aborts, or exceptions from a misaligned PC, taken to EL3.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations There is no additional configuration data for FAR_EL3.

Attributes FAR_EL3 is a 64-bit register.

The following figure shows the FAR_EL3 bit assignments.

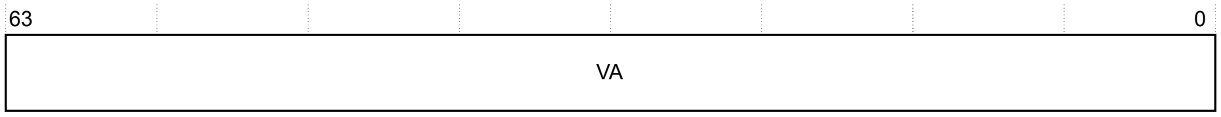

Figure 4-63 FAR_EL3 bit assignments

The following table shows the FAR_EL3 bit assignments.

Table 4-116 FAR_EL3 bit assignments

Bits	Name	Function
[63:0]	VA	The faulting Virtual Address for all synchronous instruction or data aborts, or an exception from a misaligned PC, taken in EL3. If a memory fault that sets the FAR is generated from one of the data cache instructions, this field holds the address specified in the register argument of the instruction.

To access the FAR_EL3:

```
MRS <Xt>, FAR_EL3 ; Read EL3 Fault Address Register
MSR FAR_EL3, <Xt> ; Write EL3 Fault Address Register
```

4.3.75 Physical Address Register, EL1

The PAR_EL1 characteristics are:

Purpose The Physical Address returned from an address translation.

Usage constraints 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

Configurations PAR_EL1 is architecturally mapped to AArch32 register PAR(NS). See *4.5.62 Physical Address Register* on page 4-312.

Attributes PAR_EL1 is a 64-bit register.

The following figure shows the PAR_EL1 bit assignments when the Virtual Address to Physical Address conversion completes successfully.

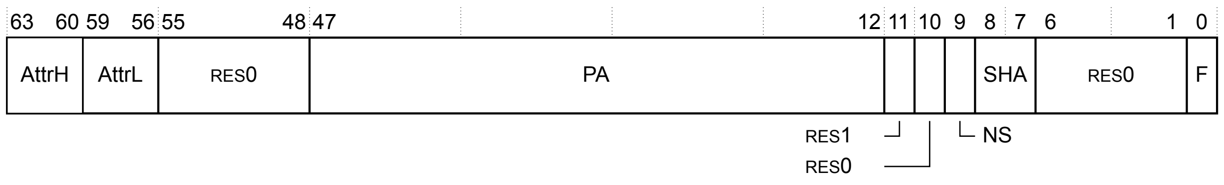

Figure 4-64 PAR_EL1 pass bit assignments

The following table shows the PAR_EL1 bit assignments when the Virtual Address to Physical Address conversion completes successfully.

Table 4-117 PAR_EL1 pass bit assignments

Bits	Name	Function
[63:60]	AttrH	Defines Normal or Device memory and outer cacheability. Must be used in conjunction with AttrL. See *Table 4-119 Attr<n>[7:4] bit assignments* on page 4-182.
[59:56]	AttrL	Defines Device memory, and Inner cacheability. Must be interpreted in conjunction with AttrH. See *Table 4-120 Attr<n>[3:0] bit assignments* on page 4-183.
[55:48]	-	Reserved, RES0.
[47:12]	PA	Physical address. The Physical Address corresponding to the supplied Virtual Address. Returns address bits[47:12].
[11]	-	Reserved, RES1.
[10]	-	Reserved, RES0.
[9]	NS	Non-secure. The NS attribute for a translation table entry read from Secure state. This bit is UNKNOWN for a translation table entry from Non-secure state.
[8:7]	SHA	Shareability attribute for the Physical Address returned from a translation table entry. The possible values are: `0b00` Non-shareable. `0b01` Reserved. `0b10` Outer Shareable `0b11` Inner Shareable. ——— **Note** ——— Takes the value of `0b10` for: • Any type of device memory. • Normal memory with both Inner Non-cacheable and Outer-cacheable attributes. ———————
[6:1]	-	Reserved, RES0.
[0]	F	Pass/Fail bit. Indicates whether the conversion completed successfully. This value is: `0` Virtual Address to Physical Address conversion completed successfully.

The following figure shows the PAR_EL1 bit assignments when the Virtual Address to Physical Address conversion is aborted.

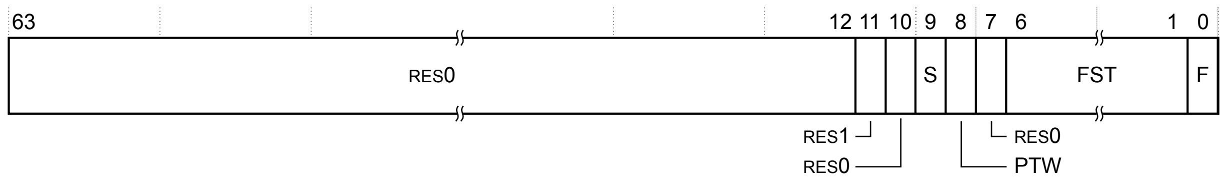

Figure 4-65 PAR_EL1 fail bit assignments

The following table shows the PAR_EL1 bit assignments when the Virtual Address to Physical Address conversion is aborted.

Table 4-118 PAR_EL1 fail bit assignments

Bits	Name	Function
[63:12]	-	Reserved, RES0.
[11]	-	Reserved, RES1.
[10]	-	Reserved, RES0.
[9]	S	Stage of fault. Indicates the state where the translation aborted. The possible values are: `0` Translation aborted because of a fault in stage 1 translation. `1` Translation aborted because of a fault in stage 2 translation.
[8]	PTW	Indicates a stage 2 fault during a stage 1 table walk. The possible values are: `0` No stage 2 fault during a stage 1 table walk. `1` Translation aborted because of a stage 2 fault during a stage 1 table walk.
[7]	-	Reserved, RES0.
[6:1]	FST	Fault status code, as shown in the Data Abort ESR encoding. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[0]	F	Pass/Fail bit. Indicates whether the conversion completed successfully. This value is: `1` Virtual Address to Physical Address conversion aborted.

[ae]

To access the PAR_EL1:

```
MRS <Xt>, PAR_EL1 ; Read EL1 Physical Address Register
MSR PAR_EL1, <Xt> ; Write EL1 Physical Address Register
```

4.3.76 Memory Attribute Indirection Register, EL1

The MAIR_EL1 characteristics are:

Purpose Provides the memory attribute encodings corresponding to the possible AttrIndx values in a Long-descriptor format translation table entry for stage 1 translations at EL1.

[ae] The transient hint is ignored.

Usage constraints 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

MAIR_EL1 is permitted to be cached in a TLB.

Configurations MAIR_EL1[31:0] is architecturally mapped to AArch32 register:

- PRRR (NS) when TTBCR.EAE is 0. See *4.5.65 Primary Region Remap Register* on page 4-318.
- MAIR0 (NS) when TTBCR.EAE is 1. See *4.5.66 Memory Attribute Indirection Registers 0 and 1* on page 4-321.

MAIR_EL1[63:32] is architecturally mapped to AArch32 register:

- NMRR (NS) when TTBCR.EAE is 0. See *4.5.67 Normal Memory Remap Register* on page 4-323.
- MAIR1(NS) when TTBCR.EAE is 1. See *4.5.66 Memory Attribute Indirection Registers 0 and 1* on page 4-321.

Attributes MAIR_EL1 is a 64-bit register.

The following figure shows the MAIR_EL1 bit assignments.

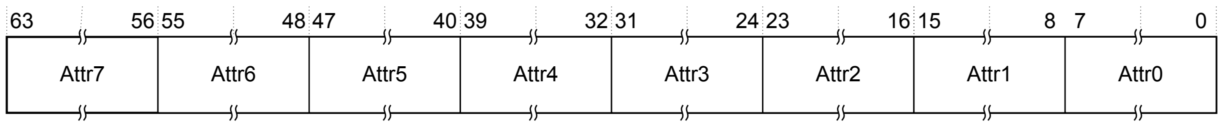

Figure 4-66 MAIR_EL1 bit assignments

Attr<n> is the memory attribute encoding for an AttrIndx[2:0] entry in a Long descriptor format translation table entry, where AttrIndx[2:0] gives the value of <n> in Attr<n>.

The following table shows the encoding of bits [7:4] of the Attr<n> field.

Table 4-119 Attr<n>[7:4] bit assignments

Bits	Meaning
`0b0000`	Device memory. See The following table for the type of Device memory.
`0b00RW`, RW not `00`	Normal Memory, Outer Write-through transient.[af]
`0b0100`	Normal Memory, Outer Non-Cacheable.
`0b01RW`, RW not `00`	Normal Memory, Outer Write-back transient.[af]
`0b10RW`	Normal Memory, Outer Write-through non-transient.
`0b11RW`	Normal Memory, Outer Write-back non-transient.

The following table shows the encoding of bits [0:3] of the Attr<n> field.

[af] The transient hint is ignored.

Table 4-120 Attr<n>[3:0] bit assignments

Bits	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
`0b0000`	Device-nGnRnE memory	UNPREDICTABLE
`0b00RW`, RW not `00`	UNPREDICTABLE	Normal Memory, Inner Write-through transient
`0b0100`	Device-nGnRE memory	Normal memory, Inner Non-Cacheable
`0b01RW`, RW not `00`	UNPREDICTABLE	Normal Memory, Inner Write-back transient
`0b1000`	Device-nGRE memory	Normal Memory, Inner Write-through non-transient (RW=00)
`0b10RW`, RW not `00`	UNPREDICTABLE	Normal Memory, Inner Write-through non-transient
`0b1100`	Device-GRE memory	Normal Memory, Inner Write-back non-transient (RW=00)
`0b11RW`, RW not `00`	UNPREDICTABLE	Normal Memory, Inner Write-back non-transient

The following table shows the encoding of the R and W bits that are used, in some Attr<n> encodings in the previous tables, to define the read-allocate and write-allocate policies:

Table 4-121 Encoding of R and W bits in some Attr<n> fields

R or W	Meaning
0	Do not allocate
1	Allocate

To access the MAIR_EL1:

```
MRS <Xt>, MAIR_EL1 ; Read EL1 Memory Attribute Indirection Register
MSR MAIR_EL1, <Xt> ; Write EL1 Memory Attribute Indirection Register
```

4.3.77 Memory Attribute Indirection Register, EL2

The MAIR_EL2 characteristics are:

Purpose Provides the memory attribute encodings corresponding to the possible AttrIndx values in a Long-descriptor format translation table entry for stage 1 translations at EL2.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

MAIR_EL2 is permitted to be cached in a TLB.

Configurations MAIR_EL2[31:0] is architecturally mapped to AArch32 register HMAIR0.

MAIR_EL2[63:32] is architecturally mapped to AArch32 register HMAIR1.

Attributes MAIR_EL2 is a 64-bit register.

The MAIR_EL2 bit assignments follow the same pattern as described in *4.3.76 Memory Attribute Indirection Register, EL1* on page 4-181.

The description of the MAIR_EL2 bit assignments are the same as described in *4.3.76 Memory Attribute Indirection Register, EL1* on page 4-181.

To access the MAIR_EL2:

```
MRS <Xt>, MAIR_EL2 ; Read EL2 Memory Attribute Indirection Register
MSR MAIR_EL2, <Xt> ; Write EL2 Memory Attribute Indirection Register
```

4.3.78 Memory Attribute Indirection Register, EL3

The MAIR_EL3 characteristics are:

Purpose Provides the memory attribute encodings corresponding to the possible AttrIndx values in a Long-descriptor format translation table entry for stage 1 translations at EL3.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

MAIR_EL3 is permitted to be cached in a TLB.

Configurations MAIR_EL3[31:0] is mapped to AArch32 register PRRR (S) when TTBCR.EAE is 0. See *4.5.65 Primary Region Remap Register* on page 4-318.

MAIR_EL3[63:32] is mapped to AArch32 register NMRR (S) when TTBCR.EAE is 0. See *4.5.67 Normal Memory Remap Register* on page 4-323.

Attributes MAIR_EL3 is a 64-bit register.

The MAIR_EL3 bit assignments follow the same pattern as described in *4.3.76 Memory Attribute Indirection Register, EL1* on page 4-181.

The description of the MAIR_EL3 bit assignments are the same as described in *4.3.76 Memory Attribute Indirection Register, EL1* on page 4-181.

To access the MAIR_EL3:

```
MRS <Xt>, MAIR_EL3 ; Read EL3 Memory Attribute Indirection Register
MSR MAIR_EL3, <Xt> ; Write EL3 Memory Attribute Indirection Register
```

4.3.79 Vector Base Address Register, EL1

The VBAR_EL1 characteristics are:

Purpose Holds the exception base address for any exception that is taken to EL1.

Usage constraints 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

Configurations The VBAR_EL1[31:0] is architecturally mapped to the Non-secure AArch32 VBAR register. See *4.5.72 Vector Base Address Register* on page 4-324.

Attributes VBAR_EL1 is a 64-bit register.

The following figure shows the VBAR_EL1 bit assignments.

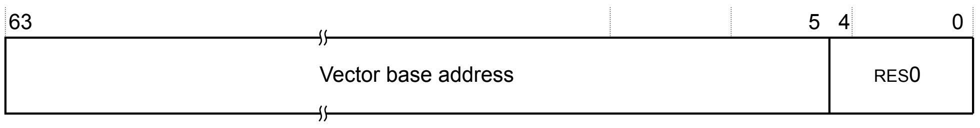

Figure 4-67 VBAR_EL1 bit assignments

The following table shows the VBAR_EL1 bit assignments.

Table 4-122 VBAR_EL1 bit assignments

Bits	Name	Function
[63:5]	Vector base address	Base address of the exception vectors for exceptions taken in this exception level.
[4:0]	-	Reserved, RES0.

To access the VBAR_EL1:

```
MRS <Xt>, VBAR_EL1 ; Read VBAR_EL1 into Xt
MSR VBAR_EL1, <Xt> ; Write Xt to VBAR_EL1
```

4.3.80 Vector Base Address Register, EL2

The VBAR_EL2 characteristics are:

Purpose Holds the exception base address for any exception that is taken to EL2.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations The VBAR_EL2[31:0] is architecturally mapped to the AArch32 HVBAR register. See *4.5.75 Hyp Vector Base Address Register* on page 4-327.

Attributes VBAR_EL2 is a 64-bit register.

The following figure shows the VBAR_EL2 bit assignments.

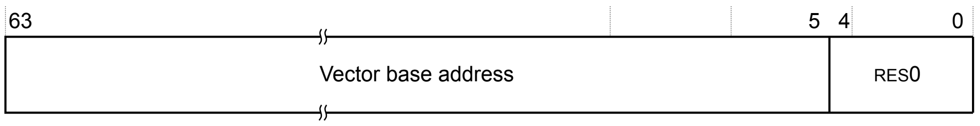

Figure 4-68 VBAR_EL2 bit assignments

The following table shows the VBAR_EL2 bit assignments.

Table 4-123 VBAR_EL2 bit assignments

Bits	Name	Function
[63:5]	Vector base address	Base address of the exception vectors for exceptions taken in this exception level.
[4:0]	-	Reserved, RES0.

To access the VBAR_EL2:

```
MRS <Xt>, VBAR_EL2 ; Read VBAR_EL2 into Xt
MSR VBAR_EL2, <Xt> ; Write Xt to VBAR_EL2
```

Register access is encoded as follows:

Table 4-124 VBAR_EL2 access encoding

op0	op1	CRn	CRm	op2
11	100	1100	0000	000

4.3.81 Vector Base Address Register, EL3

The VBAR_EL3 characteristics are:

Purpose Holds the exception base address for any exception that is taken to EL3.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Attributes VBAR_EL3 is a 64-bit register.

The following figure shows the VBAR_EL3 bit assignments.

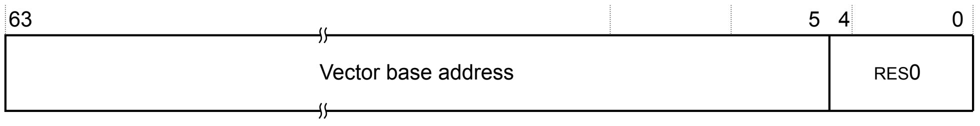

Figure 4-69 VBAR_EL3 bit assignments

The following table shows the VBAR_EL3 bit assignments.

Table 4-125 VBAR_EL3 bit assignments

Bits	Name	Function
[63:5]	Vector base address	Base address of the exception vectors for exceptions taken in this exception level.
[4:0]	-	Reserved, RES0.

To access the VBAR_EL3:

```
MRS <Xt>, VBAR_EL3 ; Read EL3 Vector Base Address Register
MSR VBAR_EL3, <Xt> ; Write EL3 Vector Base Address Register
```

4.3.82 Reset Vector Base Address Register, EL3

The RVBAR_EL3 characteristics are:

Purpose Contains the address that execution starts from after reset when executing in the AArch64 state.

RVBAR_EL3 is part of the Reset management registers functional group.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RO	RO

Configurations There is no configuration information.

Attributes RVBAR_EL3 is a 64-bit register.

The following figure shows the RVBAR_EL3 bit assignments.

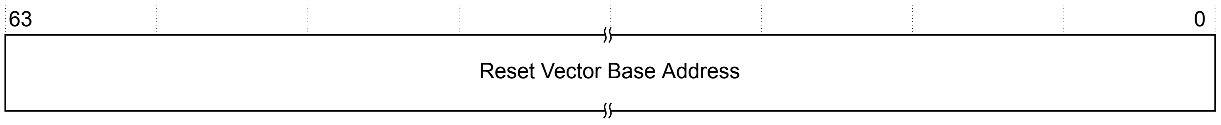

Figure 4-70 RVBAR_EL3 bit assignments

The following table shows the RVBAR_EL3 bit assignments.

Table 4-126 RVBAR_EL3 bit assignments

Bits	Name	Function
[63:0]	RVBA	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 processor.

To access the RVBAR_EL3:

```
MRS <Xt>, RVBAR_EL3 ; Read RVBAR_EL3 into Xt
```

4.3.83 Reset Management Register

The RMR_EL3 characteristics are:

Purpose Controls the execution state that the processor boots into and allows request of a Warm reset.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW

Configurations The RMR_EL3 is architecturally mapped to the AArch32 RMR register.

Attributes RMR_EL3 is a 32-bit register.

The following figure shows the RMR_EL3 bit assignments.

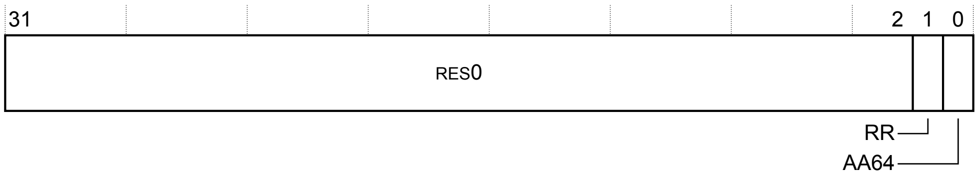

Figure 4-71 RMR_EL3 bit assignments

The following table shows the RMR_EL3 bit assignments.

Table 4-127 RMR_EL3 bit assignments

Bits	Name	Function
[31:2]	-	Reserved, RES0.
[1]	RR	Reset Request. The possible values are: `0` This is the reset value. `1` Requests a Warm reset. This bit is set to 0 by either a cold or Warm reset. The bit is strictly a request.
[0]	AA64	Determines which execution state the processor boots into after a Warm reset. The possible values are: `0` AArch32 Execution state. `1` AArch64 Execution state. The reset vector address on reset takes a choice between two values, depending on the value in the AA64 bit. This ensures that even with reprogramming of the AA64 bit, it is not possible to change the reset vector to go to a different location. The Cold reset value depends on the **AA64nAA32** signal.

To access the RMR_EL3:

```
MRS <Xt>, RMR_EL3 ; Read RMR_EL3 into Xt
MSR RMR_EL3, <Xt> ; Write Xt to RMR_EL3
```

4.3.84 Interrupt Status Register

The ISR_EL1 characteristics are:

Purpose Shows whether an IRQ, FIQ, or external abort is pending. An indicated pending abort might be a physical abort or a virtual abort.

Usage constraints 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

Configurations ISR_EL1 is architecturally mapped to AArch32 register ISR. See *4.5.74 Interrupt Status Register* on page 4-326.

Attributes ISR_EL1 is a 32-bit register.

The following figure shows the ISR_EL1 bit assignments.

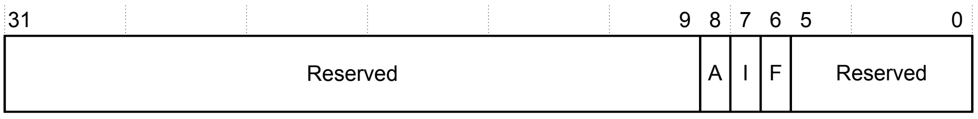

Figure 4-72 ISR_EL1 bit assignments

The following table shows the ISR_EL1 bit assignments.

Table 4-128 ISR_EL1 bit assignments

Bits	Name	Function
[31:9]	-	Reserved, RES0.
[8]	A	External abort pending bit: `0` No pending external abort. `1` An external abort is pending.
[7]	I	IRQ pending bit. Indicates whether an IRQ interrupt is pending: `0` No pending IRQ. `1` An IRQ interrupt is pending.
[6]	F	FIQ pending bit. Indicates whether an FIQ interrupt is pending: `0` No pending FIQ. `1` An FIQ interrupt is pending.
[5:0]	-	Reserved, RES0.

To access the ISR_EL1:

```
MRS <Xt>, ISR_EL1 ; Read ISR_EL1 into Xt
```

Register access is encoded as follows:

Table 4-129 ISR_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	1100	0001	000

4.3.85 Extended Control Register, EL1

ECTLR_EL1 characteristics are:

Purpose Provides configuration and control options for the L1 and L2 memory systems.

Usage constraints 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

The ECTLR_EL1 can be written dynamically.

The ECTLR_EL1 is write accessible in EL1 if ACTLR_EL3.ECTLR_EN is 1 and ACTLR_EL2.ECTLR_EN is 1, or ACTLR_EL3.ECTLR_EN is 1 and SCR.NS is 0.

The ECTLR_EL1 is write accessible in EL2 if ACTLR_EL3.ECTLR is 1.

Configurations The ECTLR_EL1 is architecturally mapped to the AArch32 ECTLR register. See *4.5.77 Extended Control Register* on page 4-328.

Attributes ECTLR_EL1 is a 64-bit register.

The following figure shows the ECTLR_EL1 bit assignments.

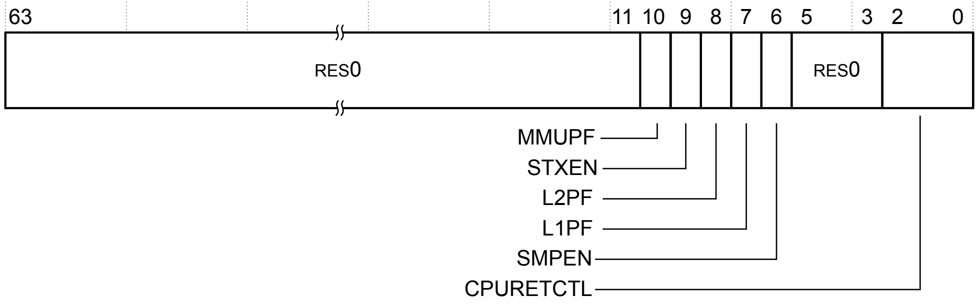

Figure 4-73 ECTLR_EL1 bit assignments

The following table shows the ECTLR_EL1 bit assignments.

Table 4-130 ECTLR_EL1 bit assignments

Bits	Name	Function
[63:11]	-	Reserved, RES0.
[10]	MMUPF	Enables MMU prefetch. The reset value is `0b1`.
[9]	STXEN	Enables exclusive signaling on the ACE master interface for CleanUnique requests due to cacheable Exclusive stores. The reset value is `0b0`.
[8]	L2PF	Enable L2 prefetch requests sent by the stride prefetcher. The reset value is `0b1`.
[7]	L1PF	Enable L1 prefetch requests sent by the stride prefetcher. The reset value is `0b1`.
[6]	SMPEN	Enable hardware management of data coherency with other cores in the cluster. The possible values are: `0` Disables data coherency with other cores in the cluster. This is the reset value. `1` Enables data coherency with other cores in the cluster. **Note** Set the SMPEN bit before enabling the caches, even if there is only one core in the system.

Table 4-130 ECTLR_EL1 bit assignments (continued)

Bits	Name	Function
[5:3]	-	Reserved, RES0.
[2:0]	CPURETCTL	CPU retention control. The possible values are: `0b000` Disable the retention circuit. This is the reset value. `0b001` 2 Architectural Timer ticks are required before retention entry. `0b010` 8 Architectural Timer ticks are required before retention entry. `0b011` 32 Architectural Timer ticks are required before retention entry. `0b100` 64 Architectural Timer ticks are required before retention entry. `0b101` 128 Architectural Timer ticks are required before retention entry. `0b110` 256 Architectural Timer ticks are required before retention entry. `0b111` 512 Architectural Timer ticks are required before retention entry. **Note** If the **CNTVALUEB[63:0]** bus increments by more than 1024, this bit is not taken into account and the retention entry may never occur.

To access the ECTLR_EL1:

```
MRS <Xt>, S3_1_C15_C2_1; Read EL1 Extended Control Register
MSR S3_1_C15_C2_1, <Xt>; Write EL1 Extended Control Register
```

4.3.86 L2 Memory Error Syndrome Register

The L2MERRSR_EL1 characteristics are:

Purpose Holds information about ECC errors on the:
- L2 data RAMs.
- L2 tag RAMs.

Usage constraints 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

Configurations The L2MERRSR_EL1 is:
- Mapped to the AArch32 L2MERRSR register. See *4.5.78 L2 Memory Error Syndrome Register* on page 4-330.
- There is one copy of this register that is used in both Secure and Non-secure states.
- A write of any value to the register updates the register to `0x0`.

Attributes L2MERRSR_EL1 is a 64-bit register.

The following figure shows the L2MERRSR_EL1 bit assignments.

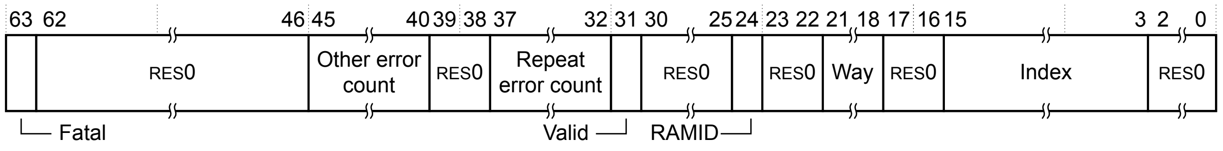

Figure 4-74 L2MERRSR_EL1 bit assignments

The following table shows the L2MERRSR_EL1 bit assignments.

Table 4-131 L2MERRSR_EL1 bit assignments

Bits	Name	Function					
[63]	Fatal	Fatal bit. This bit is set to 1 on the first memory error that caused a data abort. It is a sticky bit so that after it is set, it remains set until the register is written. The reset value is 0.					
[62:46]	-	Reserved, RES0.					
[45:40]	Other error count	This field is set to 0 on the first memory error and is incremented on any memory error that does not match the RAMID and Bank/Way information in this register while the sticky Valid bit is set. The reset value is 0.					
[39:38]	-	Reserved, RES0.					
[37:32]	Repeat error count	This field is set to 0 on the first memory error and is incremented on any memory error that exactly matches the RAMID and Bank/Way information in this register while the sticky Valid bit is set. The reset value is 0.					
[31]	Valid	Valid bit. This bit is set to 1 on the first memory error. It is a sticky bit so that after it is set, it remains set until the register is written. The reset value is 0.					
[30:25]	-	Reserved, RES0.					
[24]	RAMID	RAM Identifier. Indicates the RAM in which the first memory error occurred. The possible values are:					
		`0x0`	L2 tag RAM.				
		`0x1`	L2 data RAM.				
[23:22]	-	Reserved, RES0.					
[21:18]	Way	Indicates the RAM where the first memory error occurred.					
		L2 tag RAM	`0x0`	Way 0	**L2 data RAM**	`0x0`	Bank 0
			`0x1`	Way 1		`0x1`	Bank 1
			...			...	
			`0xE`	Way 14		`0x7`	Bank 7
			`0xF`	Way 15		`0x8-0xF`	Unused
[17:16]	-	Reserved, RES0.					

Table 4-131 L2MERRSR_EL1 bit assignments (continued)

Bits	Name	Function
[15:3]	Index	Indicates the index address of the first memory error.
[2:0]	-	Reserved, RES0.

Note

- If two or more memory errors in the same RAM occur in the same cycle, only one error is reported.
- If two or more first memory error events from different RAMs occur in the same cycle, one of the errors is selected arbitrarily, while the Other error count field is incremented only by one.
- If two or more memory error events from different RAMs, that do not match the RAMID, bank, way, or index information in this register while the sticky Valid bit is set, occur in the same cycle, the Other error count field is incremented only by one.

To access the L2MERRSR_EL1:

```
MRS <Xt>, S3_1_C15_C2_3 ; Read L2MERRSR_EL1 into Xt
MSR S3_1_C15_C2_3, <Xt> ; Write Xt into L2MERRSR_EL1
```

4.3.87 Configuration Base Address Register, EL1

The CBAR_EL1 characteristics are:

Purpose Holds the physical base address of the memory-mapped GIC CPU interface registers.

Usage constraints 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

Configurations There is one copy of this register that is used in both Secure and Non-secure states.

Attributes CBAR_EL1 is a 64-bit register.

The following figure shows the CBAR_EL1 bit assignments.

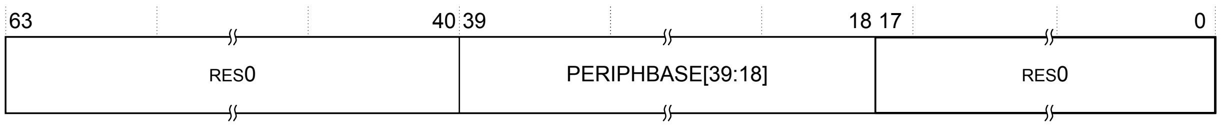

Figure 4-75 CBAR_EL1 bit assignments

The following table shows the CBAR_EL1 bit assignments.

Table 4-132 CBAR_EL1 bit assignments

Bits	Name	Function
[63:40]	-	Reserved, RES0.
[39:18]	PERIPHBASE[39:18]	The input **PERIPHBASE[39:18]** determines the reset value.
[17:0]	-	Reserved, RES0.

To access the CBAR_EL1:

```
MRS <Xt>, S3_1_C15_C3_0 ; Read CBAR_EL1 into Xt
```

4.3.88 DC CIALL Clean Invalidate All

The DC CIALL characteristics are:

Purpose Cleans and invalidates all data caches or unified caches.

Usage constraints This operation can be performed at the following exception levels:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	WO	WO	WO	WO	WO

Configurations DC CIALL performs the same function as AArch32 operation DCCIALL.

Attributes DC CIALL is a 32-bit system operation.

The following figure shows the DC CIALL input value bit assignments.

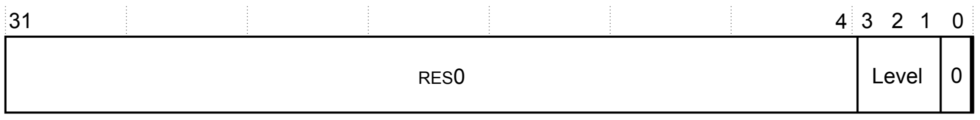

Figure 4-76 DC CIALL input value bit assignments

The DC CIALL input value bit assignments are:.

Table 4-133 DC CIALL bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:1]	Level	Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.
[0]	-	Reserved, RES0.

To perform the DC CIALL operation:

```
MSR S1_1_C15_C14_0, <Xt>
```

The operation is encoded as follows:

Table 4-134 DC CIALL access encoding

op0	op1	CRn	CRm	op2
1	001	1111	1110	000

4.4 AArch32 register summary

In AArch32 state you access the system registers through a conceptual coprocessor, identified as CP15, the System Control Coprocessor.

Within CP15, there is a top-level grouping of system registers by a primary coprocessor register number, c0-c15. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about using the conceptual System Control Coprocessor in a VMSA context.

The system register space includes system registers and system operations. The description of the system register space describes the permitted access, RO, WO, or RW, to each register or operation.

The following sections describe the CP15 system control registers grouped by CRn order, and are accessed by the MCR and MRC instructions.

- *4.4.1 c0 registers* on page 4-196.
- *4.4.2 c1 registers* on page 4-198.
- *4.4.3 c2 registers* on page 4-198.
- *4.4.4 c3 registers* on page 4-199.
- *4.4.5 c4 registers* on page 4-199.
- *4.4.6 c5 registers* on page 4-199.
- *4.4.7 c6 registers* on page 4-200.
- *4.4.8 c7 registers* on page 4-200.
- *4.4.9 c7 System operations* on page 4-200.
- *4.4.10 c8 System operations* on page 4-202.
- *4.4.11 c9 registers* on page 4-203.
- *4.4.12 c10 registers* on page 4-204.
- *4.4.13 c11 registers* on page 4-205.
- *4.4.14 c12 registers* on page 4-205.
- *4.4.15 c13 registers* on page 4-207.
- *4.4.16 c14 registers* on page 4-207.
- *4.4.17 c15 registers* on page 4-209.

The following subsection describes the 64-bit registers and provides cross-references to individual register descriptions:

- *4.4.18 64-bit registers* on page 4-209.

In addition to listing the CP15 system registers by CRn ordering, the following subsections describe the CP15 system registers by functional group:

- *4.4.19 AArch32 Identification registers* on page 4-210.
- *4.4.20 AArch32 Virtual memory control registers* on page 4-212.
- *4.4.21 AArch32 Fault handling registers* on page 4-212.
- *4.4.22 AArch32 Other System registers* on page 4-213.
- *4.4.23 AArch32 Address translation registers* on page 4-213.
- *4.4.24 AArch32 Thread ID registers* on page 4-213.
- *4.4.25 AArch32 Performance monitor registers* on page 4-214.
- *4.4.26 AArch32 Secure registers* on page 4-215.
- *4.4.27 AArch32 Virtualization registers* on page 4-216.
- *4.4.28 AArch32 GIC system registers* on page 4-218.
- *4.4.29 AArch64 Generic Timer registers* on page 4-219.
- *4.4.30 AArch32 Implementation defined registers* on page 4-220.

The following table describes the column headings in the CP15 register summary tables used throughout this section.

Table 4-135 System register field values

Heading	Description
CRn	System control primary register number.
Op1	Arguments to the register access instruction.
CRm	
Op2	
Name	The name of the register or operation. Some assemblers support aliases that you can use to access the registers and operations by name.
Reset	Reset value of register.
Description	Cross-reference to the register description.

4.4.1 c0 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c0.

Table 4-136 c0 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c0	0	c0	0	MIDR	`0x411FD090`	*4.5.1 Main ID Register* on page 4-222
			1	CTR	`0x84448004`	*4.5.26 Cache Type Register* on page 4-253
			2	TCMTR	`0x00000000`	*4.5.4 TCM Type Register* on page 4-226
			3	TLBTR	`0x00000000`	*4.5.5 TLB Type Register* on page 4-226
			4, 7	MIDR	`0x411FD090`	Aliases of Main ID Register, *4.5.1 Main ID Register* on page 4-222
			5	MPIDR	-[ag]	*4.5.2 Multiprocessor Affinity Register* on page 4-223
			6	REVIDR	`0x00000000`	*4.5.3 Revision ID Register* on page 4-225
		c1	0	ID_PFR0	`0x00010131`	*4.5.6 Processor Feature Register 0* on page 4-226
			1	ID_PFR1	`0x10011011`[ah]	*4.5.7 Processor Feature Register 1* on page 4-227
			2	ID_DFR0	`0x03010066`	*4.5.9 Debug Feature Register 0* on page 4-230
			3	ID_AFR0	`0x00000000`	*4.5.10 Auxiliary Feature Register 0* on page 4-231
			4	ID_MMFR0	`0x10201105`	*4.5.11 Memory Model Feature Register 0* on page 4-231
			5	ID_MMFR1	`0x40000000`	*4.5.12 Memory Model Feature Register 1* on page 4-232
			6	ID_MMFR2	`0x01260000`	*4.5.13 Memory Model Feature Register 2* on page 4-234
			7	ID_MMFR3	`0x02102211`	*4.5.14 Memory Model Feature Register 3* on page 4-236
		c2	0	ID_ISAR0	`0x02101110`	*4.5.15 Instruction Set Attribute Register 0* on page 4-237
			1	ID_ISAR1	`0x13112111`	*4.5.16 Instruction Set Attribute Register 1* on page 4-239
			2	ID_ISAR2	`0x21232042`	*4.5.17 Instruction Set Attribute Register 2* on page 4-240
			3	ID_ISAR3	`0x01112131`	*4.5.18 Instruction Set Attribute Register 3* on page 4-242
			4	ID_ISAR4	`0x00011142`	*4.5.19 Instruction Set Attribute Register 4* on page 4-244
			5	ID_ISAR5	`0x00011121` or `0x00010001`[ai]	*4.5.20 Instruction Set Attribute Register 5* on page 4-246
			6	ID_MMFR4	`0x00000000`	*4.5.21 Memory Model Feature Register 4* on page 4-248
		c3	4	ID_PFR2	`0x00000001`	*4.5.8 Processor Feature Register 2* on page 4-229
	1	c0	0	CCSIDR	UNK	*4.5.22 Cache Size ID Register* on page 4-248
			1	CLIDR	`0x0A200023`	*4.5.23 Cache Level ID Register* on page 4-250
			7	AIDR	`0x00000000`	*4.5.24 Auxiliary ID Register* on page 4-252
	2	c0	0	CSSELR	UNK	*4.5.25 Cache Size Selection Register* on page 4-252
	4	c0	0	VPIDR	`0x411FD090`	*4.5.27 Virtualization Processor ID Register* on page 4-255
	5	c0	0	VMPIDR	[aj]	*4.5.28 Virtualization Multiprocessor ID Register* on page 4-255

[ag] The reset value depends on the primary inputs, **CLUSTERIDAFF1** and **CLUSTERIDAFF2**, and the number of cores that the device implements.

4.4.2 c1 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c1.

Table 4-137 c1 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c1	0	c0	0	SCTLR	`0x00C50838`[ak]	*4.5.29 System Control Register* on page 4-256
			1	ACTLR	`0x00000000`	*4.5.30 Auxiliary Control Register* on page 4-260
			2	CPACR	`0x00000000`	*4.5.31 Architectural Feature Access Control Register* on page 4-261
		c1	0	SCR	`0x00000000`	*4.5.32 Secure Configuration Register* on page 4-263
			1	SDER	`0x00000000`	*4.5.33 Secure Debug Enable Register* on page 4-265
			2	NSACR	`0x00000000`[al] If EL3 is AArch64 then the NSACR reads as `0x00000C00`.	*4.5.34 Non-Secure Access Control Register* on page 4-266
		c3	1	SDCR	`0x00000000`	*4.5.35 Secure Debug Configuration Register* on page 4-267
	4	c0	0	HSCTLR	UNK	*4.5.37 Hyp System Control Register* on page 4-270
			1	HACTLR	`0x00000000`	*4.5.36 Hyp Auxiliary Control Register* on page 4-269
		c1	0	HCR	`0x00000002`	*4.5.38 Hyp Configuration Register* on page 4-273
			1	HDCR	`0x00000006`	*4.5.40 Hyp Debug Control Register* on page 4-279
			2	HCPTR	`0x000033FF`	*4.5.41 Hyp Architectural Feature Trap Register* on page 4-282
			3	HSTR	`0x00000000`	*4.5.48 Hyp System Trap Register* on page 4-297
			4	HCR2	`0x00000000`	*4.5.39 Hyp Configuration Register 2* on page 4-278
			7	HACR	UNK	*4.5.41 Hyp Architectural Feature Trap Register* on page 4-282

4.4.3 c2 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c2.

ah Bits [31:28] are `0x1` if the GIC CPU interface is enabled, and `0x0` otherwise.
ai The value is `0x00011121` if the Cryptographic Extension is implemented and enabled. The value is `0x00010001` if the Cryptographic Extension is not implemented and enabled.
aj The reset value is the value of the Multiprocessor Affinity Register.
ak The reset value depends on inputs, **CFGTE**, **CFGEND**, and **VINITHI**. This reset value assumes these signals are set to LOW.
al If EL3 is AArch64 then the NSACR reads as `0x00000C00`.

Table 4-138 c2 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c2	0	c0	0	TTBR0	UNK	*4.5.42 Translation Table Base Register 0* on page 4-284
			1	TTBR1	UNK	*4.5.43 Translation Table Base Register 1* on page 4-287
			2	TTBCR	`0x00000000`[am]	*4.5.44 Translation Table Base Control Register* on page 4-289
	4	c0	2	HTCR	UNK	*4.5.45 Hyp Translation Control Register* on page 4-293
		c1	2	VTCR	UNK	*4.5.46 Virtualization Translation Control Register* on page 4-294

4.4.4 c3 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c3.

Table 4-139 c3 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c3	0	c0	0	DACR	UNK	*4.5.47 Domain Access Control Register* on page 4-296

4.4.5 c4 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c4.

Table 4-140 c3 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c4	0	c6	0	ICC_PMR	`0x00000000`	Priority Mask Register

4.4.6 c5 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c5.

Table 4-141 c5 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c5	0	c0	0	DFSR	UNK	*4.5.50 Data Fault Status Register* on page 4-300
			1	IFSR	UNK	*4.5.51 Instruction Fault Status Register* on page 4-304
		c1	0	ADFSR	`0x00000000`	*4.5.52 Auxiliary Data Fault Status Register* on page 4-307
			1	AIFSR	`0x00000000`	*4.5.53 Auxiliary Instruction Fault Status Register* on page 4-307
c5	4	c1	0	HADFSR	`0x00000000`	*4.5.54 Hyp Auxiliary Data Fault Status Syndrome Register* on page 4-307
			1	HAIFSR	`0x00000000`	*4.5.55 Hyp Auxiliary Instruction Fault Status Syndrome Register* on page 4-307
		c2	0	HSR	UNK	*4.5.56 Hyp Syndrome Register* on page 4-307

[am] The reset value is `0x00000000` for the Secure copy of the register. The reset value for the EAE bit of the Non-secure copy of the register is `0x0`.

4.4.7 c6 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c6.

Table 4-142 c6 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c6	0	c0	0	DFAR	UNK	*4.5.57 Data Fault Address Register* on page 4-309
			2	IFAR	UNK	*4.5.58 Instruction Fault Address Register* on page 4-309
	4	c0	0	HDFAR	UNK	*4.5.59 Hyp Data Fault Address Register* on page 4-310
			2	HIFAR	UNK	*4.5.60 Hyp Instruction Fault Address Register* on page 4-311
			4	HPFAR	UNK	*4.5.61 Hyp IPA Fault Address Register* on page 4-312

4.4.8 c7 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c7.

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

Table 4-143 c7 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c7	0	c4	0	PAR	UNK	*4.5.62 Physical Address Register* on page 4-312

4.4.9 c7 System operations

The following table shows the System operations when CRn is c7 and the processor is in AArch32 state.

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

Table 4-144 c7 System operation summary

op1	CRm	op2	Name	Description
0	c1	0	`ICIALLUIS`	Invalidate all instruction caches Inner Shareable to PoU[an]
		6	`BPIALLIS`	Invalidate all entries from branch predictors Inner Shareable
	c5	0	`ICIALLU`	Invalidate all Instruction Caches to PoU
		1	`ICIMVAU`	Invalidate Instruction Caches by VA to PoU
		4	`CP15ISB`	Instruction Synchronization Barrier operation, this operation is deprecated in Armv8-A
		6	`BPIALL`	Invalidate all entries from branch predictors
		7	`BPIMVA`	Invalidate VA from branch predictors
	c6	1	`DCIMVAC`	Invalidate data cache line by VA to PoC[ao]
		2	`DCISW`	Invalidate data cache line by set/way
	c8	0	`ATS1CPR`	Stage 1 current state PL1 read
		1	`ATS1CPW`	Stage 1 current state PL1 write
		2	`ATS1CUR`	Stage 1 current state unprivileged read
		3	`ATS1CUW`	Stage 1 current state unprivileged write
		4	`ATS12NSOPR`	Stages 1 and 2 Non-secure only PL1 read
		5	`ATS12NSOPW`	Stages 1 and 2 Non-secure only PL1 write
		6	`ATS12NSOUR`	Stages 1 and 2 Non-secure only unprivileged read
		7	`ATS12NSOUW`	Stages 1 and 2 Non-secure only unprivileged write
	c10	1	`DCCMVAC`	Clean data cache line by VA to PoC
		2	`DCCSW`	Clean data cache line by set/way
		4	`CP15DSB`	Data Synchronization Barrier operation, this operation is deprecated in Armv8-A
		5	`CP15DMB`	Data Memory Barrier operation, this operation is deprecated in Armv8-A
	c11	1	`DCCMVAU`	Clean data cache line by VA to PoU
	c14	1	`DCCIMVAC`	Clean and invalidate data cache line by VA to PoC
		2	`DCCISW`	Clean and invalidate data cache line by set/way
4	c8	0	`ATS1HR`	Stage 1 Hyp mode read
		1	`ATS1HW`	Stage 1 Hyp mode write

[an] PoU = Point of Unification. PoU is set by the **BROADCASTINNER** signal and can be in the L1 data cache or outside of the processor, in which case PoU is dependent on the external memory system.
[ao] PoC = Point of Coherence. The PoC is always outside of the processor and is dependent on the external memory system.

4.4.10 c8 System operations

The following table shows the System operations when CRn is c8 and the processor is in AArch32 state.

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

Table 4-145 c8 System operations summary

op1	CRm	op2	Name	Description
0	c3	0	`TLBIALLIS`	Invalidate entire TLB Inner Shareable
		1	`TLBIMVAIS`	Invalidate unified TLB entry by VA and ASID Inner Shareable
		2	`TLBIASIDIS`	Invalidate unified TLB by ASID match Inner Shareable
		3	`TLBIMVAAIS`	Invalidate unified TLB entry by VA all ASID Inner Shareable
		5	`TLBIMVALIS`	Invalidate unified TLB entry by VA Inner Shareable, Last level
		7	`TLBIMVAALIS`	Invalidate unified TLB by VA all ASID Inner Shareable, Last level
	c5	0	`ITLBIALL`	Invalidate instruction TLB
		1	`ITLBIMVA`	Invalidate instruction TLB entry by VA and ASID
		2	`ITLBIASID`	Invalidate instruction TLB by ASID match
	c6	0	`DTLBIALL`	Invalidate data TLB
		1	`DTLBIMVA`	Invalidate data TLB entry by VA and ASID
		2	`DTLBIASID`	Invalidate data TLB by ASID match
	c7	0	`TLBIALL`	Invalidate unified TLB
		1	`TLBIMVA`	Invalidate unified TLB by VA and ASID
		2	`TLBIASID`	Invalidate unified TLB by ASID match
		3	`TLBIMVAA`	Invalidate unified TLB entries by VA all ASID
		5	`TLBIMVAL`	Invalidate last level of stage 1 TLB entry by VA
		7	`TLBIMVAAL`	Invalidate last level of stage 1 TLB entry by VA all ASID

Table 4-145 c8 System operations summary (continued)

op1	CRm	op2	Name	Description
4	c0	1	`TLBIIPAS2IS`	TLB Invalidate entry by Intermediate Physical Address, Stage 2, Inner Shareable
		5	`TLBIIPAS2LIS`	TLB Invalidate entry by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
	c3	0	`TLBIALLHIS`	Invalidate entire Hyp unified TLB Inner Shareable
		1	`TLBIMVAHIS`	Invalidate Hyp unified TLB entry by VA Inner Shareable
		4	`TLBIALLNSNHIS`	Invalidate entire Non-secure non-Hyp unified TLB Inner Shareable
		5	`TLBIMVALHIS`	Invalidate Unified Hyp TLB entry by VA Inner Shareable, Last level
	c4	1	`TLBIIPAS2`	TLB Invalidate entry by Intermediate Physical Address, Stage 2
		5	`TLBIIPAS2L`	TLB Invalidate entry by Intermediate Physical Address, Stage 2, Last level
	c7	0	`TLBIALLH`	Invalidate entire Hyp unified TLB
		1	`TLBIMVAH`	Invalidate Hyp unified TLB entry by VA
		4	`TLBIALLNSNH`	Invalidate entire Non-secure non-Hyp unified TLB
		5	`TLBIMVALH`	Invalidate Unified Hyp TLB entry by VA, Last level

4.4.11 c9 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c9.

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

Table 4-146 c9 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c9	0	c12	0	PMCR	`0x41043000`	Performance Monitor Control Register
			1	PMNCNTENSET	UNK	Performance Monitors Count Enable Set Register
			2	PMNCNTENCLR	UNK	Performance Monitors Count Enable Clear Register
			3	PMOVSR	UNK	Performance Monitors Overflow Flag Status Clear Register
			4	PMSWINC	UNK	Performance Monitors Software Increment Register
			5	PMSELR	UNK	Performance Monitors Event Counter Selection Register
			6	PMCEID0	`0x7BFF7F3F`	Common Event Identification Register 0
			7	PMCEID1	`0x00000000`	Common Event Identification Register 1
		c13	0	PMCCNTR	UNK	Performance Monitors Cycle Counter
			1	PMXEVTYPER	UNK	Performance Monitors Selected Event Type and Filter Register
			2	PMXEVCNTR	UNK	Performance Monitors Selected Event Counter Register
		c14	0	PMUSERENR	`0x00000000`	Performance Monitors User Enable Register
			1	PMINTENSET	UNK	Performance Monitors Interrupt Enable Set Register
			2	PMINTENCLR	UNK	Performance Monitors Interrupt Enable Clear Register
			3	PMOVSSET	UNK	Performance Monitor Overflow Flag Status Set Register
	1	c0	2	L2CTLR	-[ap]	*4.5.63 L2 Control Register* on page 4-312
			3	L2ECTLR	`0x00000000`	*4.5.64 L2 Extended Control Register* on page 4-315

4.4.12 c10 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c10.

[ap] The reset value depends on the processor configuration.

Table 4-147 c10 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c10	0	c2	0	PRRR	UNK	*4.5.65 Primary Region Remap Register* on page 4-318
			0	MAIR0	UNK	*4.5.66 Memory Attribute Indirection Registers 0 and 1* on page 4-321
			1	NMRR	UNK	*4.5.67 Normal Memory Remap Register* on page 4-323
			1	MAIR1	UNK	*4.5.66 Memory Attribute Indirection Registers 0 and 1* on page 4-321
		c3	0	AMAIR0	`0x00000000`	*4.5.68 Auxiliary Memory Attribute Indirection Register 0* on page 4-324
			1	AMAIR1	`0x00000000`	*4.5.69 Auxiliary Memory Attribute Indirection Register 1* on page 4-324
	4	c2	0	HMAIR0	UNK	Hyp Memory Attribute Indirection Register 0[aq]
			1	HMAIR1	UNK	Hyp Memory Attribute Indirection Register 1[aq]
		c3	0	HAMAIR0	`0x00000000`	*4.5.70 Hyp Auxiliary Memory Attribute Indirection Register 0* on page 4-324
			1	HAMAIR1	`0x00000000`	*4.5.71 Hyp Auxiliary Memory Attribute Indirection Register 1* on page 4-324

4.4.13 c11 registers

There are no system registers to access when the processor is in AArch32 state and the value of CRn is c11.

4.4.14 c12 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c12.

Table 4-148 c12 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c12	0	c0	0	VBAR	UNK[ar]	*4.5.72 Vector Base Address Register* on page 4-324.
			1	MVBAR	UNK	Monitor Vector Base Address Register. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
			2	RMR	[as]	*4.5.73 Reset Management Register* on page 4-325.
		c1	0	ISR	UNK	*4.5.74 Interrupt Status Register* on page 4-326.
		c8	0	ICC_IAR0	-	Interrupt Acknowledge Register 0
			1	ICC_EOIR0	-	End Of Interrupt Register 0
			2	ICC_HPPIR0	-	Highest Priority Pending Interrupt Register 0
			3	ICC_BPR0	`0x00000002`	Binary Point Register 0
			4	ICC_AP0R0	`0x00000000`	Active Priorities 0 Register 0

[aq] See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[ar] This is the reset value in Non-secure state. In Secure state, the reset value is 0x00000000.
[as] The Cold reset value depends on the AA64nAA32 signal.

Table 4-148 c12 register summary (continued)

CRn	Op1	CRm	Op2	Name	Reset	Description
c12	0	c9	0	ICC_AP1R0	`0x00000000`	Active Priorities 1 Register 0
		c11	1	ICC_DIR	-	Deactivate Interrupt Register
			3	ICC_RPR	-	Running Priority Register
		c12	0	ICC_IAR1	-	Interrupt Acknowledge Register 1
			1	ICC_EOIR1	-	End Of Interrupt Register 1
			2	ICC_HPPIR1	-	Highest Priority Pending Interrupt Register 1
			3	ICC_BPR1	`0x00000003`[at]	Binary Point Register 1
			4	ICC_CTLR	`0x00000400`	Interrupt Control Register
			5	ICC_SRE	`0x00000000`	System Register Enable Register
			6	ICC_IGRPEN0	`0x00000000`	Interrupt Group Enable Register 0
			7	ICC_IGRPEN1	`0x00000000`	Interrupt Group Enable Register 1
	4	c0	0	HVBAR	UNK	*4.5.75 Hyp Vector Base Address Register* on page 4-327.
		c8	0	ICH_AP0R0	`0x00000000`	Interrupt Controller Hyp Active Priorities Register (0,0)
		c9	0	ICH_AP1R0	`0x00000000`	Interrupt Controller Hyp Active Priorities Register (1,0)
			4	ICH_VSEIR	`0x00000000`	Interrupt Controller Virtual System Error Interrupt Register
			5	ICC_HSRE	`0x00000000`	System Register Enable Register for EL2
		c11	0	ICH_HCR	`0x00000000`	Interrupt Controller Hyp Control Register
			1	ICH_VTR	`0x90000003`	Interrupt Controller VGIC Type Register
			2	ICH_MISR	`0x00000000`	Interrupt Controller Maintenance Interrupt State Register
			3	ICH_EISR	`0x00000000`	Interrupt Controller End of Interrupt Status Register
			7	ICH_VMCR	`0x004C0000`	Interrupt Controller Virtual Machine Control Register
			5	ICH_ELRSR	`0x0000000F`	Interrupt Controller Empty List Register Status Register
		c12	0	ICH_LR0	`0x00000000`	Interrupt Controller List Register 0
			1	ICH_LR1	`0x00000000`	Interrupt Controller List Register 1
			2	ICH_LR2	`0x00000000`	Interrupt Controller List Register 2
			3	ICH_LR3	`0x00000000`	Interrupt Controller List Register 3
		c14	0	ICH_LRC0	`0x00000000`	Interrupt Controller List Register 0
			1	ICH_LRC1	`0x00000000`	Interrupt Controller List Register 1
			2	ICH_LRC2	`0x00000000`	Interrupt Controller List Register 2
			3	ICH_LRC3	`0x00000000`	Interrupt Controller List Register 3
c12	6	c12	4	ICC_MCTLR	`0x00000400`	Interrupt Control Register for EL3
			5	ICC_MSRE	`0x00000000`	System Register Enable Register for EL3
			7	ICC_MGRPEN1	`0x00000000`	Interrupt Controller Monitor Interrupt Group 1 Enable register

[at] This is the reset value in Non-secure state. In Secure state, the reset value is 0x00000002.

4.4.15 c13 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c13.

Table 4-149 c13 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c13	0	c0	0	FCSEIDR	`0x00000000`	*4.5.76 FCSE Process ID Register* on page 4-328
			1	CONTEXTIDR	UNK	Context ID Register[au]
			2	TPIDRURW	UNK	User Read/Write Thread ID Register[au]
			3	TPIDRURO	UNK	User Read-Only Thread ID Register[au]
			4	TPIDRPRW	UNK	EL1 only Thread ID Register[au]
	4	c0	2	HTPIDR	UNK	Hyp Software Thread ID Register[au]

4.4.16 c14 registers

The following table shows the CP15 system registers when the processor is in AArch32 state and the value of CRn is c14.

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

[au] See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table 4-150 c14 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	CNTFRQ	UNK	Timer Counter Frequency Register
	c1	0	CNTKCTL	-[av]	Timer Control Register
	c2	0	CNTP_TVAL	UNK	Physical Timer TimerValue Register
		1	CNTP_CTL	-[aw]	Physical Timer Control Register
	c3	0	CNTV_TVAL	UNK	Virtual Timer TimerValue Register
		1	CNTV_CTL	-[aw]	Counter-timer Virtual Timer Control Register
	c8	0	PMEVCNTR0	UNK	Performance Monitors Event Count Registers
		1	PMEVCNTR1	UNK	
		2	PMEVCNTR2	UNK	
		3	PMEVCNTR3	UNK	
		4	PMEVCNTR4	UNK	
		5	PMEVCNTR5	UNK	
	c12	0	PMEVTYPER0	UNK	Performance Monitors Event Type Registers
		1	PMEVTYPER1	UNK	
		2	PMEVTYPER2	UNK	
		3	PMEVTYPER3	UNK	
		4	PMEVTYPER4	UNK	
		5	PMEVTYPER5	UNK	
	c15	7	PMCCFILTR	`0x00000000`	Performance Monitors Cycle Count Filter Register.
4	c1	0	CNTHCTL	-[ax]	Timer Control Register (EL2)
	c2	0	CNTHP_TVAL	UNK	Physical Timer TimerValue (EL2)
		1	CNTHP_CTL	-[aw]	Physical Timer Control Register (EL2)

[av] The reset value for bits[9:8, 2:0] is `0b00000`.
[aw] The reset value for bit[0] is 0.
[ax] The reset value for bit[2] is 0 and for bits[1:0] is `0b11`.

4.4.17 c15 registers

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c15.

Table 4-151 c15 register summary

Op1	CRm	Op2	Name	Reset	Description
1	c2	1	ECTLR	`0x00000580`	*4.5.77 Extended Control Register* on page 4-328
		3	L2MERRSR	`0x00000000`	*4.5.78 L2 Memory Error Syndrome Register* on page 4-330
	c3	0	CBAR	-[ay]	*4.5.79 Configuration Base Address Register* on page 4-332
	c14	0	DCCIALL	UNK	*4.5.80 DCCIALL Clean Invalidate All* on page 4-332
3	c0	0	CDBGDR0	UNK	Cache Debug Data Register 0, see *6.7 Direct access to internal memory* on page 6-351
		1	CDBGDR1	UNK	Cache Debug Data Register 1, see *6.7 Direct access to internal memory* on page 6-351
		2	CDBGDR2	UNK	Cache Debug Data Register 2, see *6.7 Direct access to internal memory* on page 6-351
		3	CDBGDR3	UNK	Cache Debug Data Register 3, see *6.7 Direct access to internal memory* on page 6-351
	c2	0	CDBGDCT	UNK	Cache Debug Data Cache Tag Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351
		1	CDBGICT	UNK	Cache Debug Instruction Cache Tag Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351
	c4	0	CDBGDCD	UNK	Cache Debug Cache Debug Data Cache Data Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351
		1	CDBGICD	UNK	Cache Debug Instruction Cache Data Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351
		2	CDBGTD	UNK	Cache Debug TLB Data Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351

4.4.18 64-bit registers

The following table shows the 64-bit wide CP15 system registers, accessed by the `MCRR` and `MRRC` instructions.

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

Table 4-152 64-bit register summary

Op1	CRm	Name	Reset	Description
0	c2	TTBR0	UNK	Translation Table Base Register 0
1	c2	TTBR1	UNK	Translation Table Base Register 1
4	c2	HTTBR	UNK	Hyp Translation Table Base Register
6	c2	VTTBR	UNK	Virtualization Translation Table Base Register

[ay] The reset value depends on the processor configuration.

Table 4-152 64-bit register summary (continued)

Op1	CRm	Name	Reset	Description
0	c7	PAR	UNK	*4.5.62 Physical Address Register* on page 4-312
3	c13	PMCCNTR	UNK	Cycle Count Register
0	c14	CNTPCT	UNK	Physical Timer Count Register
1	c14	CNTVCT	UNK	Virtual Timer Count Register
2	c14	CNTP_CVAL	UNK	Physical Timer CompareValue Register
3	c14	CNTV_CVAL	UNK	Virtual Timer CompareValue Register
4	c14	CNTVOFF	UNK	Virtual Timer Offset Register
6	c14	CNTHP_CVAL	UNK	Physical Timer CompareValue Register
1	c15	ECTLR	`0x0000000000000580`	*4.5.77 Extended Control Register* on page 4-328
3	c15	L2MERRSR	`0x0000000000000000`	*4.5.78 L2 Memory Error Syndrome Register* on page 4-330

4.4.19 AArch32 Identification registers

The following table shows the identification registers.

Table 4-153 Identification registers

Name	CRn	Op1	CRm	Op2	Reset	Description
MIDR	c0	0	c0	0	`0x411FD090`	*4.5.1 Main ID Register* on page 4-222
CTR				1	`0x84448004`	*4.5.26 Cache Type Register* on page 4-253
TCMTR				2	`0x00000000`	*4.5.4 TCM Type Register* on page 4-226
TLBTR				3	`0x00000000`	*4.5.5 TLB Type Register* on page 4-226
MPIDR				5	-[az]	*4.5.2 Multiprocessor Affinity Register* on page 4-223
REVIDR				6	`0x00000000`	*4.5.3 Revision ID Register* on page 4-225
ID_PFR0			c1	0	`0x00010131`	*4.5.6 Processor Feature Register 0* on page 4-226
ID_PFR1				1	`0x10011011`[ba]	*4.5.7 Processor Feature Register 1* on page 4-227
ID_DFR0				2	`0x03010066`	*4.5.9 Debug Feature Register 0* on page 4-230
ID_AFR0				3	`0x00000000`	*4.5.10 Auxiliary Feature Register 0* on page 4-231
ID_MMFR0				4	`0x10201105`	*4.5.11 Memory Model Feature Register 0* on page 4-231
ID_MMFR1				5	`0x40000000`	*4.5.12 Memory Model Feature Register 1* on page 4-232
ID_MMFR2				6	`0x01260000`	*4.5.13 Memory Model Feature Register 2* on page 4-234
ID_MMFR3				7	`0x02102211`	*4.5.14 Memory Model Feature Register 3* on page 4-236
ID_ISAR0			c2	0	`0x02101110`	*4.5.15 Instruction Set Attribute Register 0* on page 4-237
ID_ISAR1				1	`0x13112111`	*4.5.16 Instruction Set Attribute Register 1* on page 4-239
ID_ISAR2				2	`0x21232042`	*4.5.17 Instruction Set Attribute Register 2* on page 4-240
ID_ISAR3				3	`0x01112131`	*4.5.18 Instruction Set Attribute Register 3* on page 4-242
ID_ISAR4				4	`0x00011142`	*4.5.19 Instruction Set Attribute Register 4* on page 4-244
ID_ISAR5				5	`0x00011121` or `0x00010001`[bb]	*4.5.20 Instruction Set Attribute Register 5* on page 4-246
ID_PFR2			c3	4	`0x00000001`	*4.5.8 Processor Feature Register 2* on page 4-229
CCSIDR		1	c0	0	-	*4.5.22 Cache Size ID Register* on page 4-248
CLIDR				1	`0x0A200023`	*4.5.23 Cache Level ID Register* on page 4-250
AIDR				7	`0x00000000`	*4.5.24 Auxiliary ID Register* on page 4-252
CSSELR		2	c0	0	UNK	*4.5.25 Cache Size Selection Register* on page 4-252

az The reset value depends on the primary inputs, **CLUSTERIDAFF1** and **CLUSTERIDAFF2**, and the number of cores that the device implements.
ba Bits [31:28] are `0x1` if the GIC CPU interface is enabled, and `0x0` otherwise.
bb The value is `0x00011121` if the Cryptographic Extension is implemented and enabled. The value is `0x00010001` if the Cryptographic Extension is not implemented and enabled.

4.4.20 AArch32 Virtual memory control registers

The following table shows the virtual memory control registers.

Table 4-154 Virtual memory control registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
SCTLR	c1	0	c0	0	`0x00C50838`[bc]	32-bit	*4.5.29 System Control Register* on page 4-256
TTBR0	c2	0	c0	0	UNK	32-bit	Translation Table Base Register 0, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*
	-	0	c2	-		64-bit	
TTBR1	-	0	c0	1	UNK	32-bit	Translation Table Base Register 1, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*
	-	1	c2	-		64-bit	
TTBCR	-	0	c0	2	`0x00000000`[bd]	32-bit	*4.5.44 Translation Table Base Control Register* on page 4-289
DACR	c3	0	c0	0	UNK	32-bit	*4.5.47 Domain Access Control Register* on page 4-296
PRRR	c10	0	c2	0	UNK	32-bit	*4.5.65 Primary Region Remap Register* on page 4-318
MAIR0				0	UNK	32-bit	*4.5.66 Memory Attribute Indirection Registers 0 and 1* on page 4-321
NMRR				1	UNK	32-bit	*4.5.67 Normal Memory Remap Register* on page 4-323
MAIR1				1	UNK	32-bit	*4.5.66 Memory Attribute Indirection Registers 0 and 1* on page 4-321
AMAIR0			c3	0	`0x00000000`	32-bit	*4.5.68 Auxiliary Memory Attribute Indirection Register 0* on page 4-324
AMAIR1				1	`0x00000000`	32-bit	*4.5.69 Auxiliary Memory Attribute Indirection Register 1* on page 4-324
CONTEXTIDR	c13	0	c0	1	UNK	32-bit	Context ID Register, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*

4.4.21 AArch32 Fault handling registers

The following table shows the Fault handling registers in the AArch32 Execution state.

Table 4-155 Fault handling registers

Name	CRn	Op1	CRm	Op2	Reset	Description
DFSR	c5	0	c0	0	UNK	*4.5.50 Data Fault Status Register* on page 4-300
IFSR				1	UNK	*4.5.51 Instruction Fault Status Register* on page 4-304
ADFSR			c1	0	`0x00000000`	*4.5.52 Auxiliary Data Fault Status Register* on page 4-307
AIFSR				1	`0x00000000`	*4.5.53 Auxiliary Instruction Fault Status Register* on page 4-307

[bc] The reset value depends on inputs, **CFGTE**, **CFGEND**, and **VINITHI**. This reset value assumes these signals are set to LOW.
[bd] The reset value is `0x00000000` for the Secure copy of the register. The reset value for the EAE bit of the Non-secure copy of the register is `0x0`. You must program the Non-secure copy of the register with the required initial value, as part of the processor boot sequence.

Table 4-155 Fault handling registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
DFAR	c6	0	c0	0	UNK	Data Fault Address Register, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*
IFAR				2	UNK	Instruction Fault Address Register, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*

The Virtualization registers include additional fault handling registers. For more information see *4.4.27 AArch32 Virtualization registers* on page 4-216.

4.4.22 AArch32 Other System registers

The following table shows the other system registers.

Table 4-156 Other system registers

Name	CRn	Op1	CRm	Op2	Reset	Description
ACTLR	c1	0	c0	1	`0x00000000`	*4.5.30 Auxiliary Control Register* on page 4-260
CPACR				2	`0x00000000`	*4.5.31 Architectural Feature Access Control Register* on page 4-261
FCSEIDR	c13	0	c0	0	`0x00000000`	*4.5.76 FCSE Process ID Register* on page 4-328

4.4.23 AArch32 Address translation registers

The following table shows the address translation registers.

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

Table 4-157 Address translation registers.

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
PAR	c7	0	c4	0	UNK	32-bit	*4.5.62 Physical Address Register* on page 4-312
	-		c7	-		64-bit	

4.4.24 AArch32 Thread ID registers

The following table shows the thread ID registers.

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

Table 4-158 Thread ID registers

Name	CRn	Op1	CRm	Op2	Reset	Description
TPIDRURW	c13	0	c0	2	UNK	User Read/Write Thread ID Register
TPIDRURO				3	UNK	User Read-Only Thread ID Register
TPIDRPRW				4	UNK	EL1 only Thread ID Register
HTPIDR		4	c0	2	UNK	Hyp Software Thread ID Register

4.4.25 AArch32 Performance monitor registers

The following table shows the performance monitor registers.

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

Table 4-159 Performance monitor registers

Name	CRn	Op1	CRm	Op2	Reset	Description
PMCR	c9	0	c12	0	`0x41043000`	*11.4.1 Performance Monitors Control Register* on page 11-448
PMCNTENSET				1	UNK	Performance Monitors Count Enable Set Register
PMCNTENCLR				2	UNK	Performance Monitors Count Enable Clear Register
PMOVSR				3	UNK	Performance Monitors Overflow Flag Status Register
PMSWINC				4	UNK	Performance Monitors Software Increment Register
PMSELR				5	UNK	Performance Monitors Event Counter Selection Register
PMCEID0				6	`0x7BFF7F3F`	*11.4.2 Performance Monitors Common Event Identification Register 0* on page 11-450
PMCEID1				7	`0x00000000`	*11.4.3 Performance Monitors Common Event Identification Register 1* on page 11-453
PMCCNTR			c13	0	UNK	Performance Monitors Cycle Count Register
PMXEVTYPER				1	UNK	Performance Monitors Selected Event Type Register
PMXEVCNTR				2	UNK	Performance Monitors Event Count Registers
PMUSERENR			c14	0	`0x00000000`	Performance Monitors User Enable Register
PMINTENSET				1	UNK	Performance Monitors Interrupt Enable Set Register
PMINTENCLR				2	UNK	Performance Monitors Interrupt Enable Clear Register
PMOVSSET				3	UNK	Performance Monitor Overflow Flag Status Set Register

Table 4-159 Performance monitor registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
PMEVCNTR0	c14	0	c8	0	UNK	Performance Monitors Event Count Register 0
PMEVCNTR1				1	UNK	Performance Monitors Event Count Register 1
PMEVCNTR2				2	UNK	Performance Monitors Event Count Register 2
PMEVCNTR3				3	UNK	Performance Monitors Event Count Register 3
PMEVCNTR4				4	UNK	Performance Monitors Event Count Register 4
PMEVCNTR5				5	UNK	Performance Monitors Event Count Register 5
PMEVTYPER0			c12	0	UNK	Performance Monitors Event Type Register 0
PMEVTYPER1				1	UNK	Performance Monitors Event Type Register 1
PMEVTYPER2				2	UNK	Performance Monitors Event Type Register 2
PMEVTYPER3				3	UNK	Performance Monitors Event Type Register 3
PMEVTYPER4				4	UNK	Performance Monitors Event Type Register 4
PMEVTYPER5				5	UNK	Performance Monitors Event Type Register 5
PMCCFILTR			c15	7	`0x00000000`	Performance Monitors Cycle Count Filter Register

4.4.26 AArch32 Secure registers

The following table shows the Secure registers.

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

Table 4-160 Secure registers

Name	CRn	Op1	CRm	Op2	Reset	Description
SCR	c1	0	c1	0	`0x00000000`	*4.5.32 Secure Configuration Register* on page 4-263
SDER				1	UNK	Secure Debug Enable Register
NSACR				2	`0x00000000`[be]	*4.5.34 Non-Secure Access Control Register* on page 4-266
VBAR	c12	0	c0	0	UNK[bf]	*4.5.72 Vector Base Address Register* on page 4-324
MVBAR				1	UNK	Monitor Vector Base Address Register
ISR			c1	0	UNK	Interrupt Status Register

be If EL3 is AArch64 then the NSACR reads as `0x00000C00`.
bf This is the reset value in Non-secure state. In Secure state, the reset value is 0x00000000.

4.4.27 AArch32 Virtualization registers

The following table shows the Virtualization registers.

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

Table 4-161 Virtualization registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
VPIDR	c0	4	c0	0	`0x411FD090`	32-bit	*4.5.27 Virtualization Processor ID Register* on page 4-255
VMPIDR				5	-[bg]	32-bit	*4.5.28 Virtualization Multiprocessor ID Register* on page 4-255
HSCTLR	c1	4	c0	0	UNK	32-bit	*4.5.37 Hyp System Control Register* on page 4-270
HACTLR				1	UNK		*4.5.36 Hyp Auxiliary Control Register* on page 4-269
HCR			c1	0	`0x00000002`	32-bit	Hyp Configuration Register
HDCR				1	`0x00000006`	32-bit	*4.5.40 Hyp Debug Control Register* on page 4-279
HCPTR				2	`0x000033FF`	32-bit	*4.5.41 Hyp Architectural Feature Trap Register* on page 4-282
HSTR				3	`0x00000000`	32-bit	Hypervisor System Trap Register
HTCR	c2	4	c0	2	UNK	32-bit	*4.5.45 Hyp Translation Control Register* on page 4-293
VTCR			c1	2	UNK	32-bit	Virtualization Translation Control Register
HTTBR	-	4	c2	-	UNK	64-bit	Hyp Translation Table Base Register
VTTBR	-	6	c2	-	UNK	64-bit	Virtualization Translation Table Base Register
HADFSR	c5	4	c1	0	`0x00000000`	32-bit	*4.5.54 Hyp Auxiliary Data Fault Status Syndrome Register* on page 4-307
HAIFSR				1	`0x00000000`	32-bit	*4.5.55 Hyp Auxiliary Instruction Fault Status Syndrome Register* on page 4-307
HSR			c2	0	UNK	32-bit	*4.5.56 Hyp Syndrome Register* on page 4-307
HDFAR	c6	4	c0	0	UNK	32-bit	Hyp Data Fault Address Register
HIFAR				2	UNK	32-bit	Hyp Instruction Fault Address Register
HPFAR				4	UNK	32-bit	Hyp IPA Fault Address Register

[bg] The reset value is the value of the Multiprocessor Affinity Register.

Table 4-161 Virtualization registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
HMAIR0	c10	4	c2	0	UNK	32-bit	Hyp Memory Attribute Indirection Register 0
HMAIR1				1	UNK	32-bit	Hyp Memory Attribute Indirection Register 1
HAMAIR0			c3	0	`0x00000000`	32-bit	*4.5.70 Hyp Auxiliary Memory Attribute Indirection Register 0* on page 4-324
HAMAIR1				1	`0x00000000`	32-bit	*4.5.71 Hyp Auxiliary Memory Attribute Indirection Register 1* on page 4-324
HVBAR	c12	4	c0	0	UNK	32-bit	Hyp Vector Base Address Register

4.4.28 AArch32 GIC system registers

The following table shows the GIC system registers in AArch32 state.

Table 4-162 AArch32 GIC system registers

Name	CRn	Op1	CRm	Op2	Type	Reset	Width	Description
ICC_SGI1R	-	0	c12	-	WO	-	64-bit	SGI Generation Register 1
ICC_ASGI1R		1	c12	-	WO	-	64-bit	Alternate SGI Generation Register 1
ICC_SGI0R		2	c12	-	WO	-	64-bit	SGI Generation Register 0
ICC_PMR	c4	0	c6	0	RW	`0x00000000`	32-bit	Priority Mask Register
ICC_IAR0	c12	0	c8	0	RO	-	32-bit	Interrupt Acknowledge Register 0
ICC_EOIR0				1	WO	-	32-bit	End Of Interrupt Register 0
ICC_HPPIR0				2	RO	-	32-bit	Highest Priority Pending Interrupt Register 0
ICC_BPR0				3	RW	`0x00000002`	32-bit	Binary Point Register 0
ICC_AP0R0				4	RW	`0x00000000`	32-bit	Active Priorities 0 Register 0
ICC_AP1R0			c9	0	RW	`0x00000000`	32-bit	Active Priorities 1 Register 0
ICC_DIR			c11	1	WO	-	32-bit	Deactivate Interrupt Register
ICC_RPR				3	RO	-	32-bit	Running Priority Register
ICC_IAR1			c12	0	RO	-	32-bit	Interrupt Acknowledge Register 1
ICC_EOIR1				1	WO	-	32-bit	End Of Interrupt Register 1
ICC_HPPIR1				2	RO	-	32-bit	Highest Priority Pending Interrupt Register 1
ICC_BPR1				3	RW	`0x00000003`[bh]	32-bit	Binary Point Register 1
ICC_CTLR				4	RW	`0x00000400`	32-bit	Interrupt Control Register
ICC_SRE				5	RW	`0x00000000`	32-bit	System Register Enable Register
ICC_IGRPEN0				6	RW	`0x00000000`	32-bit	Interrupt Group Enable Register 0
ICC_IGRPEN1				7	RW	`0x00000000`	32-bit	Interrupt Group Enable Register 1
ICH_AP0R0		4	c8	0	RW	`0x00000000`	32-bit	Interrupt Controller Hyp Active Priorities Register (0,0)
ICH_AP1R0			c9	0	RW	`0x00000000`	32-bit	Interrupt Controller Hyp Active Priorities Register (1,0)
ICC_HSRE				5	RW	`0x00000000`	32-bit	System Register Enable Register for EL2
ICH_HCR			c11	0	RW	`0x00000000`	32-bit	Interrupt Controller Hyp Control Register
ICH_VTR				1	RO	`0x90000003`	32-bit	Interrupt Controller VGIC Type Register
ICH_MISR				2	RO	`0x00000000`	32-bit	Interrupt Controller Maintenance Interrupt State Register

[bh] This is the reset value in Non-secure state. In Secure state, the reset value is 0x00000002.

Table 4-162 AArch32 GIC system registers (continued)

Name	CRn	Op1	CRm	Op2	Type	Reset	Width	Description
ICH_EISR	c12	4	c8	3	RO	`0x00000000`	32-bit	Interrupt Controller End of Interrupt Status Register
ICH_VMCR				7	RW	`0x004C0000`	32-bit	Interrupt Controller Virtual Machine Control Register
ICH_ELRSR				5	RO	`0x0000000F`	32-bit	Interrupt Controller Empty List Register Status Register
ICH_LR0			c12	0	RW	`0x00000000`	32-bit	Interrupt Controller List Register 0
ICH_LR1				1	RW	`0x00000000`	32-bit	Interrupt Controller List Register 1
ICH_LR2				2	RW	`0x00000000`	32-bit	Interrupt Controller List Register 2
ICH_LR3				3	RW	`0x00000000`	32-bit	Interrupt Controller List Register 3
ICH_LRC0			c14	0	RW	`0x00000000`	32-bit	Interrupt Controller List Register 0
ICH_LRC1				1	RW	`0x00000000`	32-bit	Interrupt Controller List Register 1
ICH_LRC2				2	RW	`0x00000000`	32-bit	Interrupt Controller List Register 2
ICH_LRC3				3	RW	`0x00000000`	32-bit	Interrupt Controller List Register 3
ICC_MCTLR		6	c12	4	RW	`0x00000400`	32-bit	Interrupt Control Register for EL3
ICC_MSRE				5	RW	`0x00000000`	32-bit	System Register Enable Register for EL3
ICC_MGRPEN1				7	RW	`0x00000000`	32-bit	Interrupt Controller Monitor Interrupt Group 1 Enable register

4.4.29 AArch64 Generic Timer registers

See *Chapter 9 Generic Timer* on page 9-386 for information on the Generic Timer registers.

4.4.30 AArch32 Implementation defined registers

The following table shows the 32-bit wide implementation defined registers.

These registers provide test features and any required configuration options specific to the Cortex-A73 processor.

Table 4-163 Memory access registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
L2CTLR	c9	1	c0	2	-[bi]	32-bit	*4.5.63 L2 Control Register* on page 4-312
L2ECTLR				3	`0x00000000`	32-bit	*4.5.64 L2 Extended Control Register* on page 4-315
CBAR	c15	1	c3	0	-[bi]	32-bit	*4.5.79 Configuration Base Address Register* on page 4-332
CDBGDR0		3[bj]	c0	0	UNK	32-bit	Data Register 0, see *6.7 Direct access to internal memory* on page 6-351
CDBGDR1				1	UNK	32-bit	Data Register 1, see *6.7 Direct access to internal memory* on page 6-351
CDBGDR2				2	UNK	32-bit	Data Register 2, see *6.7 Direct access to internal memory* on page 6-351
ECTLR	-	1	c15	-	`0x00000580`	32-bit	*4.5.77 Extended Control Register* on page 4-328
L2MERRSR	-	3	c15	-	`0x0000000000000000`	64-bit	*4.5.78 L2 Memory Error Syndrome Register* on page 4-330

4.4.31 AArch32 Implementation defined operations

The following table shows the implementation defined operations in AArch32 state.

Table 4-164 Memory access registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
DCCIALL	c15	1	c14	0	UNK	32-bit	*4.5.80 DCCIALL Clean Invalidate All* on page 4-332
CDBGDCT		3	c2	0	UNK	32-bit	Data Cache Tag Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351
CDBGICT				1	UNK	32-bit	Instruction Cache Tag Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351
CDBGDCD			c4	0	UNK	32-bit	Data Cache Data Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351
CDBGICD				1	UNK	32-bit	Instruction Cache Data Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351
CDBGTD				2	UNK	32-bit	TLB Data Read Operation Register, see *6.7 Direct access to internal memory* on page 6-351

[bi] The reset value depends on the processor configuration.
[bj] See *6.7 Direct access to internal memory* on page 6-351 for information on how these registers are used.

4.5 AArch32 register descriptions

This section describes all the CP15 system registers in register number order.

See *4.4.1 c0 registers* on page 4-196 and *4.4.28 AArch32 GIC system registers* on page 4-218, which provide cross-references to individual registers.

This section contains the following subsections:

4.5.1 Main ID Register

The MIDR characteristics are:

Purpose Provides identification information for the processor, including an implementer code for the device and a device ID number.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Configurations The MIDR is:

- Architecturally mapped to the AArch64 MIDR_EL1 register. See *4.3.2 Multiprocessor Affinity Register* on page 4-84.
- Architecturally mapped to external MIDR_EL1 register.

Attributes MIDR is a 32-bit register.

The following figure shows the MIDR bit assignments.

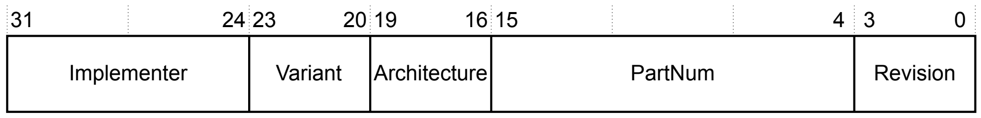

Figure 4-77 MIDR bit assignments

The following table shows the MIDR bit assignments.

Table 4-165 MIDR bit assignments

Bits	Name	Function
[31:24]	Implementer	Indicates the implementer code. This value is: `0x41` ASCII character 'A' - implementer is Arm Limited.
[23:20]	Variant	Indicates the variant number of the processor. This is the major revision number *n* in the *rn* part of the *rnpn* description of the product revision status. This value is: `0x1` r1p0.
[19:16]	Architecture	Indicates the architecture code. This value is: `0xF` Defined by CPUID scheme.
[15:4]	PartNum	Indicates the primary part number. This value is: `0xD09` Cortex-A73 processor.
[3:0]	Revision	Indicates the minor revision number of the processor. This is the minor revision number *n* in the *pn* part of the *rnpn* description of the product revision status. This value is: `0x0` r1p0.

To access the MIDR:

```
MRC p15, 0, <Rt>, c0, c0, 0; Read MIDR into Rt
```

Register access is encoded as follows:

Table 4-166 MIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0000	000

The MIDR can be accessed through the external debug interface, offset `0xD00`.

4.5.2 Multiprocessor Affinity Register

The MPIDR characteristics are:

Purpose Provides an additional core identification mechanism for scheduling purposes in a cluster.

EDDEVAFF0 is a read-only copy of MPIDR accessible from the external debug interface.

Usage constraints This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and Enables There are no traps or enables affecting this register.

Configurations The internal MPIDR is architecturally mapped to external EDDEVAFF0 register.

There is one copy of this register that is used.

Attributes MPIDR is a 32-bit register.

The following figure shows the MPIDR bit assignments.

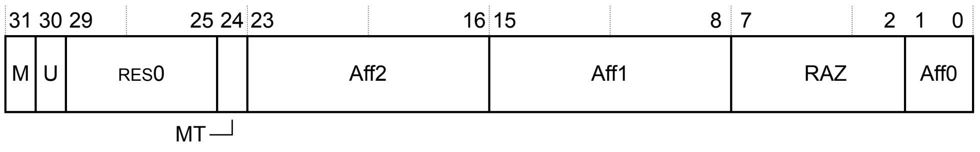

Figure 4-78 MPIDR bit assignments

The following table shows the MPIDR bit assignments.

Table 4-167 MPIDR bit assignments

Bits	Name	Function
[31]	M	RES1.
[30]	U	Indicates a single core system, as distinct from core 0 in a cluster. This value is: `0` Core is part of a cluster.
[29:25]	-	Reserved, RES0.
[24]	MT	Indicates whether the lowest level of affinity consists of logical cores that are implemented using a multi-threading type approach. This value is: `0` CPU does not support a multi-threaded micro-architecture.
[23:16]	Aff2	Affinity level 2. Second highest level affinity field. Indicates the value read in the **CLUSTERIDAFF2** configuration signal.
[15:8]	Aff1	Affinity level 1. Third highest level affinity field. Indicates the value read in the **CLUSTERIDAFF1** configuration signal.

Table 4-167 MPIDR bit assignments (continued)

Bits	Name	Function
[7:2]	RAZ	Read as zero.

[1:0]	Aff0	Affinity level 0. Lowest level affinity field.
		Indicates the core number in the Cortex-A73 processor. The possible values are:
		`0x0` A processor with one core only.
		`0x0` A cluster with two cores.
		`0x1`
		`0x0` A cluster with three cores.
		`0x1`
		`0x2`
		`0x0` A cluster with four cores.
		`0x1`
		`0x2`
		`0x3`

To access the MPIDR:

```
MRC p15,0,<Rt>,c0,c0,5 ; Read MPIDR into Rt
```

Register access is encoded as follows:

Table 4-168 MPIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0000	101

The EDDEVAFF0 can be accessed through the external debug interface, offset `0xFA8`.

4.5.3 Revision ID Register

The REVIDR characteristics are:

Purpose Provides implementation-specific minor revision information that can be interpreted only in conjunction with the Main ID Register.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Configurations REVIDR is architecturally mapped to AArch64 register REVIDR_EL1. See *4.3.3 Revision ID Register* on page 4-86.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes REVIDR is a 32-bit register.

The following figure shows the REVIDR bit assignments.

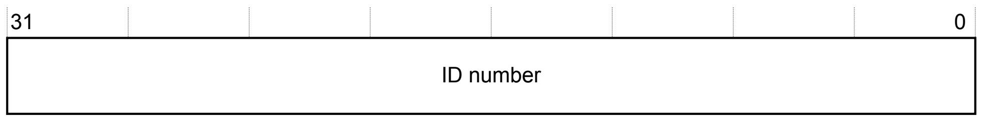

Figure 4-79 REVIDR bit assignments

The following table shows the REVIDR bit assignments.

Table 4-169 REVIDR bit assignments

Bits	Name	Function
[31:0]	ID number	Implementation-specific revision information. The reset value is determined by the specific Cortex-A73 MPCore implementation. `0x00000000` Revision code is zero.

To access the REVIDR:

```
MRC p15, 0, <Rt>, c0, c0, 6; Read REVIDR into Rt
```

Register access is encoded as follows:

Table 4-170 REVIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0000	110

4.5.4 TCM Type Register

The processor does not implement the features described by the TCMTR, so this register is always RAZ.

4.5.5 TLB Type Register

The processor does not implement the features described by the TLBTR, so this register is always RAZ.

4.5.6 Processor Feature Register 0

The ID_PFR0 characteristics are:

Purpose Gives top-level information about the instruction sets supported by the processor in AArch32.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

ID_PFR0 must be interpreted with ID_PFR1.

Configurations ID_PFR0 is architecturally mapped to AArch64 register ID_PFR0_EL1. See *4.3.4 AArch32 Processor Feature Register 0* on page 4-87.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_PFR0 is a 32-bit register.

The following figure shows the ID_PFR0 bit assignments.

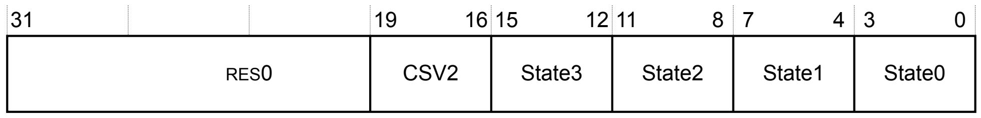

Figure 4-80 ID_PFR0 bit assignments

The following table shows the ID_PFR0 bit assignments.

Table 4-171 ID_PFR0 bit assignments

Bits	Name	Function
[31:20]	-	Reserved, RES0.
[19:16]	CSV2	`0x1` Branch targets trained in one hardware described context cannot affect speculative execution in a different hardware described context.
[15:12]	State3	Indicates support for *Thumb Execution Environment* (T32EE) instruction set. This value is: `0x0` Processor does not support the T32EE instruction set.
[11:8]	State2	Indicates support for Jazelle. This value is: `0x1` Processor supports trivial implementation of Jazelle.
[7:4]	State1	Indicates support for T32 instruction set. This value is: `0x3` Processor supports T32 encoding after the introduction of Thumb-2 technology, and for all 16-bit and 32-bit T32 basic instructions.
[3:0]	State0	Indicates support for A32 instruction set. This value is: `0x1` A32 instruction set implemented.

To access the ID_PFR0:

```
MRC p15,0,<Rt>,c0,c1,0 ; Read ID_PFR0 into Rt
```

Register access is encoded as follows:

Table 4-172 ID_PFR0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	000

4.5.7 Processor Feature Register 1

The ID_PFR1 characteristics are:

Purpose Provides information about the programmers model and architecture extensions supported by the processor.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_PFR0.

Configurations

ID_PFR1 is architecturally mapped to AArch64 register ID_PFR1_EL1. See *4.3.5 AArch32 Processor Feature Register 1* on page 4-88.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_PFR1 is a 32-bit register.

The following figure shows the ID_PFR1 bit assignments.

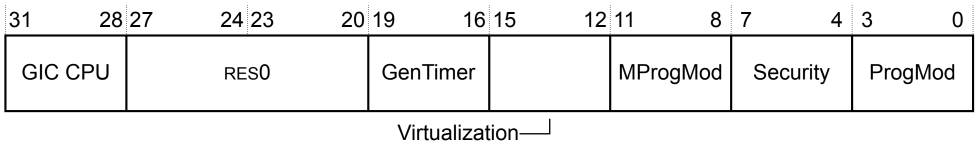

Figure 4-81 ID_PFR1 bit assignments

The following table shows the ID_PFR1 bit assignments.

Table 4-173 ID_PFR1 bit assignments

Bits	Name	Function
[31:28]	GIC CPU	GIC CPU support: `0x0` GIC CPU interface is disabled, **GICCDISABLE** is HIGH. `0x1` GIC CPU interface is enabled, **GICCDISABLE** is LOW.
[27:20]	-	Reserved, RES0.
[19:16]	GenTimer	Generic Timer support: `0x1` Generic Timer implemented.
[15:12]	Virtualization	Indicates support for Virtualization: `0x1` Virtualization implemented.
[11:8]	MProgMod	M profile programmers' model support: `0x0` Not supported.
[7:4]	Security	Security support: `0x1` Security implemented. This includes support for Monitor mode and the SMC instruction.
[3:0]	ProgMod	Indicates support for the standard programmers model for Armv4 and later. Model must support User, FIQ, IRQ, Supervisor, Abort, Undefined and System modes: `0x1` Supported.

To access the ID_PFR1:

```
MRC p15,0,<Rt>,c0,c1,1 ; Read ID_PFR1 into Rt
```

Register access is encoded as follows:

Table 4-174 ID_PFR1 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	001

4.5.8 Processor Feature Register 2

The ID_PFR2 characteristics are:

Purpose Provides information about the programmers model and architecture extensions supported by the core.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_PFR0.

Configurations ID_PFR2 is architecturally mapped to AArch64 register ID_PFR2_EL1. See *4.3.6 AArch32 Processor Feature Register 2* on page 4-89.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_PFR2 is a 32-bit register.

The following figure shows the ID_PFR2 bit assignments.

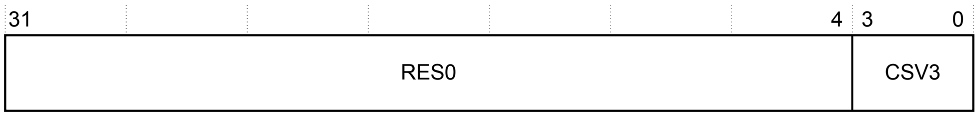

Figure 4-82 ID_PFR2 bit assignments

The following table shows the ID_PFR2 bit assignments.

Table 4-175 ID_PFR2 bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	CSV3	`0x1` Data loaded under control flow speculation with a permission or domain fault, if used as an address in a speculative load, cannot cause cache allocation.

To access the ID_PFR2:

```
MRC p15,0,<Rt>,c0,c3,4 ; Read ID_PFR2 into Rt
```

Register access is encoded as follows:

Table 4-176 ID_PFR2 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	C0	C3	100

4.5.9 Debug Feature Register 0

The ID_DFR0 characteristics are:

Purpose Provides top level information about the debug system in AArch32.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with the Main ID Register, MIDR.

Configurations ID_DFR0 is architecturally mapped to AArch64 register ID_DFR0_EL1. See *4.3.7 AArch32 Debug Feature Register 0* on page 4-90.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_DFR0 is a 32-bit register.

The following figure shows the ID_DFR0 bit assignments.

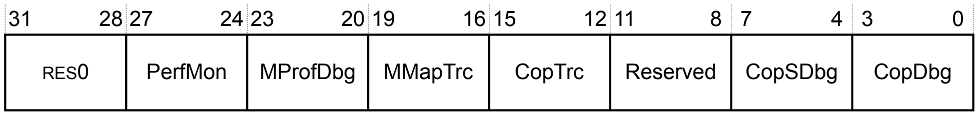

Figure 4-83 ID_DFR0 bit assignments

The following table shows the ID_DFR0 bit assignments.

Table 4-177 ID_DFR0 bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RES0.
[27:24]	PerfMon	Indicates support for performance monitor model: `0x3` Support for *Performance Monitor Unit version 3* (PMUv3) system registers.
[23:20]	MProfDbg	Indicates support for memory-mapped debug model for M profile processors: `0x0` Processor does not support M profile Debug architecture.
[19:16]	MMapTrc	Indicates support for memory-mapped trace model: `0x1` Support for Arm trace architecture, with memory-mapped access. In the Trace registers, the ETMIDR gives more information about the implementation.
[15:12]	CopTrc	Indicates support for coprocessor-based trace model: `0x0` Processor does not support Arm trace architecture, with CP14 access.
[11:8]	-	Reserved, RAZ.

Table 4-177 ID_DFR0 bit assignments (continued)

Bits	Name	Function
[7:4]	CopSDbg	Indicates support for coprocessor-based Secure debug model: `0x6` Processor supports v8 Debug architecture, with CP14 access.
[3:0]	CopDbg	Indicates support for coprocessor-based debug model: `0x6` Processor supports v8 Debug architecture, with CP14 access.

To access the ID_DFR0:

```
MRC p15,0,<Rt>,c0,c1,2 ; Read ID_DFR0 into Rt
```

Register access is encoded as follows:

Table 4-178 ID_DFR0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	010

4.5.10 Auxiliary Feature Register 0

This register is always RES0.

4.5.11 Memory Model Feature Register 0

The ID_MMFR0 characteristics are:

Purpose Provides information about the memory model and memory management support in AArch32.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_MMFR1, ID_MMFR2, and ID_MMFR3. See:
- *4.5.12 Memory Model Feature Register 1* on page 4-232.
- *4.5.13 Memory Model Feature Register 2* on page 4-234.
- *4.5.14 Memory Model Feature Register 3* on page 4-236.

Configurations ID_MMFR0 is architecturally mapped to AArch64 register ID_MMFR0_EL1.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_MMFR0 is a 32-bit register.

The following figure shows the ID_MMFR0 bit assignments.

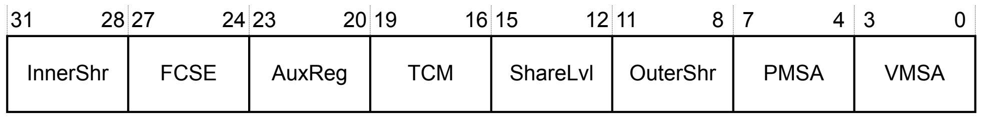

Figure 4-84 ID_MMFR0 bit assignments

The following table shows the ID_MMFR0 bit assignments.

Table 4-179 ID_MMFR0 bit assignments

Bits	Name	Function
[31:28]	InnerShr	Indicates the innermost shareability domain implemented: `0x1` Implemented with hardware coherency support.
[27:24]	FCSE	Indicates support for *Fast Context Switch Extension* (FCSE): `0x0` Not supported.
[23:20]	AuxReg	Indicates support for Auxiliary registers: `0x2` Support for Auxiliary Fault Status Registers (AIFSR and ADFSR) and Auxiliary Control Register.
[19:16]	TCM	Indicates support for TCMs and associated DMAs: `0x0` Not supported.
[15:12]	ShareLvl	Indicates the number of shareability levels implemented: `0x1` Two levels of shareability implemented.
[11:8]	OuterShr	Indicates the outermost shareability domain implemented: `0x1` Implemented with hardware coherency support.
[7:4]	PMSA	Indicates support for a *Protected Memory System Architecture* (PMSA): `0x0` Not supported.
[3:0]	VMSA	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.

To access the ID_MMFR0:

```
MRC p15,0,<Rt>,c0,c1,4 ; Read ID_MMFR0 into Rt
```

Register access is encoded as follows:

Table 4-180 ID_MMFR0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	100

4.5.12 Memory Model Feature Register 1

The ID_MMFR1 characteristics are:

Purpose Provides information about the memory model and memory management support in AArch32.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_MMFR0, ID_MMFR2, and ID_MMFR3. See:
- *4.5.11 Memory Model Feature Register 0* on page 4-231.
- *4.5.13 Memory Model Feature Register 2* on page 4-234.
- *4.5.14 Memory Model Feature Register 3* on page 4-236.

Configurations

ID_MMFR1 is architecturally mapped to AArch64 register ID_MMFR1_EL1.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_MMFR1 is a 32-bit register.

The following figure shows the ID_MMFR1 bit assignments.

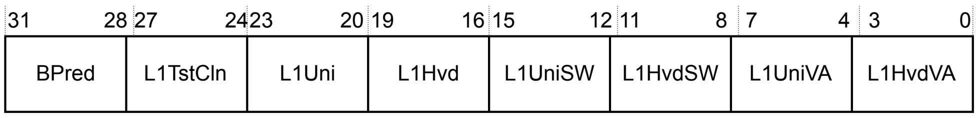

Figure 4-85 ID_MMFR1 bit assignments

The following table shows the ID_MMFR1 bit assignments.

Table 4-181 ID_MMFR1 bit assignments

Bits	Name	Function
[31:28]	BPred	Indicates branch predictor management requirements: `0x4` For execution correctness, branch predictor requires no flushing at any time.
[27:24]	L1TstCln	Indicates the supported L1 Data cache test and clean operations, for Harvard or unified cache implementation: `0x0` None supported.
[23:20]	L1Uni	Indicates the supported entire L1 cache maintenance operations, for a unified cache implementation: `0x0` None supported.
[19:16]	L1Hvd	Indicates the supported entire L1 cache maintenance operations, for a Harvard cache implementation: `0x0` None supported.
[15:12]	L1UniSW	Indicates the supported L1 cache line maintenance operations by set/way, for a unified cache implementation: `0x0` None supported.
[11:8]	L1HvdSW	Indicates the supported L1 cache line maintenance operations by set/way, for a Harvard cache implementation: `0x0` None supported.

Table 4-181 ID_MMFR1 bit assignments (continued)

Bits	Name	Function
[7:4]	L1UniVA	Indicates the supported L1 cache line maintenance operations by MVA, for a unified cache implementation: `0x0` None supported.
[3:0]	L1HvdVA	Indicates the supported L1 cache line maintenance operations by MVA, for a Harvard cache implementation: `0x0` None supported.

To access the ID_MMFR1:

```
MRC p15, 0, <Rt>, c0, c1, 5; Read ID_MMFR1 into Rt
```

Register access is encoded as follows:

Table 4-182 ID_MMFR1 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	101

4.5.13 Memory Model Feature Register 2

The ID_MMFR2 characteristics are:

Purpose Provides information about the implemented memory model and memory management support in AArch32.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_MMFR0, ID_MMFR1, and ID_MMFR3. See:
- *4.5.11 Memory Model Feature Register 0* on page 4-231.
- *4.5.12 Memory Model Feature Register 1* on page 4-232.
- *4.5.14 Memory Model Feature Register 3* on page 4-236

Configurations ID_MMFR2 is architecturally mapped to AArch64 register ID_MMFR2_EL1.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_MMFR2 is a 32-bit register.

The following figure shows the ID_MMFR2 bit assignments.

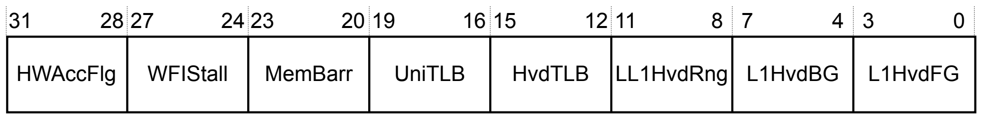

Figure 4-86 ID_MMFR2 bit assignments

The following table shows the ID_MMFR2 bit assignments.

Table 4-183 ID_MMFR2 bit assignments

Bits	Name	Function
[31:28]	HWAccFlg	Hardware Access Flag. Indicates support for a Hardware Access flag, as part of the VMSAv7 implementation: `0x0` Not supported.
[27:24]	WFIStall	Wait For Interrupt Stall. Indicates the support for *Wait For Interrupt* (WFI) stalling: `0x1` Support for WFI stalling.
[23:20]	MemBarr	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).
[19:16]	UniTLB	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`.
[15:12]	HvdTLB	Harvard TLB. Indicates the supported TLB maintenance operations, for a Harvard TLB implementation: `0x0` Not supported.
[11:8]	LL1HvdRng	L1 Harvard cache Range. Indicates the supported L1 cache maintenance range operations, for a Harvard cache implementation: `0x0` Not supported.
[7:4]	L1HvdBG	L1 Harvard cache Background fetch. Indicates the supported L1 cache background prefetch operations, for a Harvard cache implementation: `0x0` Not supported.
[3:0]	L1HvdFG	L1 Harvard cache Foreground fetch. Indicates the supported L1 cache foreground prefetch operations, for a Harvard cache implementation: `0x0` Not supported.

To access the ID_MMFR2:

```
MRC p15,0,<Rt>,c0,c1,6 ; Read ID_MMFR2 into Rt
```

Register access is encoded as follows:

Table 4-184 ID_MMFR2 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	110

4.5.14 Memory Model Feature Register 3

The ID_MMFR3 characteristics are:

Purpose Provides information about the memory model and memory management support in AArch32.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_MMFR0, ID_MMFR1, and ID_MMFR2. See:
- *4.5.11 Memory Model Feature Register 0* on page 4-231
- *4.5.12 Memory Model Feature Register 1* on page 4-232
- *4.5.13 Memory Model Feature Register 2* on page 4-234

Configurations ID_MMFR3 is architecturally mapped to AArch64 register ID_MMFR3_EL1. See *4.3.12 AArch32 Memory Model Feature Register 3* on page 4-96.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_MMFR3 is a 32-bit register.

The following figure shows the ID_MMFR3 bit assignments.

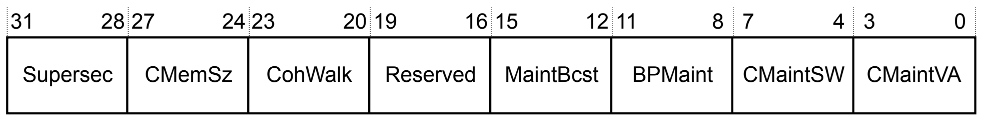

Figure 4-87 ID_MMFR3 bit assignments

The following table shows the ID_MMFR3 bit assignments.

Table 4-185 ID_MMFR3 bit assignments

Bits	Name	Function
[31:28]	Supersec	Supersections. Indicates support for supersections: `0x0` Supersections supported.
[27:24]	CMemSz	Cached Memory Size. Indicates the size of physical memory supported by the processor caches: `0x2` 1TByte, corresponding to a 40-bit physical address range.
[23:20]	CohWalk	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.
[19:16]	-	Reserved, RES0.

Table 4-185 ID_MMFR3 bit assignments (continued)

Bits	Name	Function
[15:12]	MaintBcst	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.
[11:8]	BPMaint	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.
[7:4]	CMaintSW	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.
[3:0]	CMaintVA	Cache maintenance by MVA. Indicates the supported cache maintenance operations by MVA. `0x1` Supported hierarchical cache maintenance operations by MVA are: • Invalidate data cache by MVA. Invalidate data cache by MVA operations are treated as clean and invalidate data cache by MVA operations on the executing core. If the operation is broadcast to another core then it is broadcast as an invalidate data cache by MVA operation. • Clean data cache by MVA. • Clean and invalidate data cache by MVA. • Invalidate instruction cache by MVA. • Invalidate all instruction cache entries.

To access the ID_MMFR3:

```
MRC p15, 0, <Rt>, c0, c1, 7; Read ID_MMFR3 into Rt
```

Register access is encoded as follows:

Table 4-186 ID_MMFR3 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	111

4.5.15 Instruction Set Attribute Register 0

The ID_ISAR0 characteristics are:

Purpose Provides information about the instruction sets implemented by the processor in AArch32.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_ISAR1, ID_ISAR2, ID_ISAR3, ID_ISAR4, and ID_ISAR5. See:
- *4.5.16 Instruction Set Attribute Register 1* on page 4-239.
- *4.5.17 Instruction Set Attribute Register 2* on page 4-240.
- *4.5.18 Instruction Set Attribute Register 3* on page 4-242.
- *4.5.19 Instruction Set Attribute Register 4* on page 4-244.
- *4.5.20 Instruction Set Attribute Register 5* on page 4-246.

Configurations ID_ISAR0 is architecturally mapped to AArch64 register ID_ISAR0_EL1. See *4.3.13 AArch32 Instruction Set Attribute Register 0* on page 4-98.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_ISAR0 is a 32-bit register.

The following figure shows the ID_ISAR0 bit assignments.

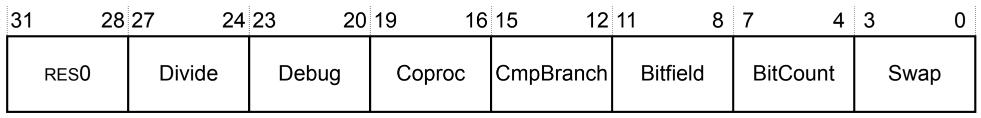

Figure 4-88 ID_ISAR0 bit assignments

The following table shows the ID_ISAR0 bit assignments.

Table 4-187 ID_ISAR0 bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RES0.
[27:24]	Divide	Indicates the implemented Divide instructions: `0x2` • `SDIV` and `UDIV` in the T32 instruction set. • `SDIV` and `UDIV` in the A32 instruction set.
[23:20]	Debug	Indicates the implemented Debug instructions: `0x1` `BKPT`.
[19:16]	Coproc	Indicates the implemented Coprocessor instructions: `0x0` None implemented, except for separately attributed by the architecture including CP15, CP14, Advanced SIMD and Floating-point.
[15:12]	CmpBranch	Indicates the implemented combined Compare and Branch instructions in the T32 instruction set: `0x1` `CBNZ` and `CBZ`.
[11:8]	Bitfield	Indicates the implemented bit field instructions: `0x1` `BFC`, `BFI`, `SBFX`, and `UBFX`.

Table 4-187 ID_ISAR0 bit assignments (continued)

Bits	Name	Function
[7:4]	BitCount	Indicates the implemented Bit Counting instructions: `0x1` CLZ.
[3:0]	Swap	Indicates the implemented Swap instructions in the A32 instruction set: `0x0` None implemented.

To access the ID_ISAR0:

```
MRC p15, 0, <Rt>, c0, c2, 0 ; Read ID_ISAR0 into Rt
```

Register access is encoded as follows:

Table 4-188 ID_ISAR0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	000

4.5.16 Instruction Set Attribute Register 1

The ID_ISAR1 characteristics are:

Purpose Provides information about the instruction sets implemented by the processor in AArch32.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_ISAR0, ID_ISAR2, ID_ISAR3, ID_ISAR4 and ID_ISAR5. See:
- *4.5.15 Instruction Set Attribute Register 0* on page 4-237.
- *4.5.17 Instruction Set Attribute Register 2* on page 4-240.
- *4.5.18 Instruction Set Attribute Register 3* on page 4-242.
- *4.5.19 Instruction Set Attribute Register 4* on page 4-244.
- *4.5.20 Instruction Set Attribute Register 5* on page 4-246.

Configurations ID_ISAR1 is architecturally mapped to AArch64 register ID_ISAR1_EL1. See *4.3.14 AArch32 Instruction Set Attribute Register 1* on page 4-99.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_ISAR1 is a 32-bit register.

The following figure shows the ID_ISAR1 bit assignments.

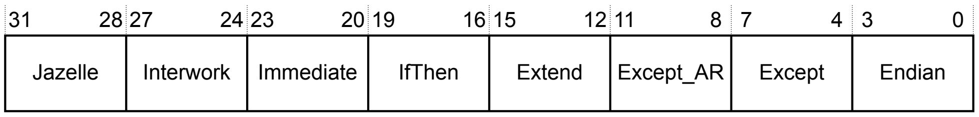

Figure 4-89 ID_ISAR1 bit assignments

The following table shows the ID_ISAR1 bit assignments.

Table 4-189 ID_ISAR1 bit assignments

Bits	Name	Function
[31:28]	Jazelle	Indicates the implemented Jazelle state instructions: `0x1` The `BXJ` instruction, and the J bit in the PSR.
[27:24]	Interwork	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.
[23:20]	Immediate	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.
[19:16]	IfThen	Indicates the implemented `If-Then` instructions in the T32 instruction set: `0x1` The `IT` instructions, and the IT bits in the PSRs.
[15:12]	Extend	Indicates the implemented Extend instructions: `0x2` • The `SXTB`, `SXTH`, `UXTB`, and `UXTH` instructions. • The `SXTB16`, `SXTAB`, `SXTAB16`, `SXTAH`, `UXTB16`, `UXTAB`, `UXTAB16`, and `UXTAH` instructions.
[11:8]	Except_AR	Indicates the implemented A profile exception-handling instructions: `0x1` The `SRS` and `RFE` instructions, and the A profile forms of the `CPS` instruction.
[7:4]	Except	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.
[3:0]	Endian	Indicates the implemented Endian instructions: `0x1` The `SETEND` instruction, and the E bit in the PSRs.

To access the ID_ISAR1:

```
MRC p15, 0, <Rt>, c0, c2, 1 ; Read ID_ISAR1 into Rt
```

Register access is encoded as follows:

Table 4-190 ID_ISAR1 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	001

4.5.17 Instruction Set Attribute Register 2

The ID_ISAR2 characteristics are:

Purpose Provides information about the instruction sets implemented by the processor in AArch32.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR3, ID_ISAR4 and ID_ISAR5. See.
- *4.5.15 Instruction Set Attribute Register 0* on page 4-237.
- *4.5.16 Instruction Set Attribute Register 1* on page 4-239.
- *4.5.18 Instruction Set Attribute Register 3* on page 4-242.
- *4.5.19 Instruction Set Attribute Register 4* on page 4-244.
- *4.5.20 Instruction Set Attribute Register 5* on page 4-246.

Configurations ID_ISAR2 is architecturally mapped to AArch64 register ID_ISAR2_EL1. See *4.3.15 AArch32 Instruction Set Attribute Register 2* on page 4-100.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_ISAR2 is a 32-bit register.

The following figure shows the ID_ISAR2 bit assignments.

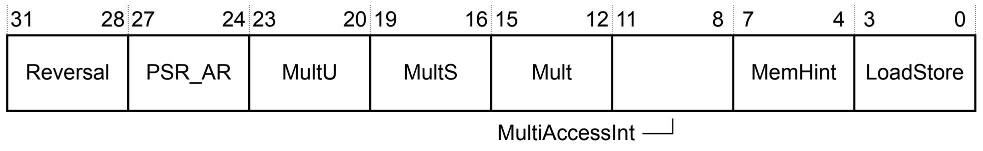

Figure 4-90 ID_ISAR2 bit assignments

The following table shows the ID_ISAR2 bit assignments.

Table 4-191 ID_ISAR2 bit assignments

Bits	Name	Function
[31:28]	Reversal	Indicates the implemented Reversal instructions: `0x2` The `REV`, `REV16`, and `REVSH` instructions. The `RBIT` instruction.
[27:24]	PSR_AR	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. **Note** 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 `SUBS PC, LR, #N` instruction.
[23:20]	MultU	Indicates the implemented advanced unsigned Multiply instructions: `0x2` The `UMULL` and `UMLAL` instructions. The `UMAAL` instruction.

Table 4-191 ID_ISAR2 bit assignments (continued)

Bits	Name	Function
[19:16]	MultS	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.
[15:12]	Mult	Indicates the implemented additional Multiply instructions: `0x2` The `MUL` instruction. The `MLA` instruction. The `MLS` instruction.
[11:8]	MultiAccessInt	Indicates the support for interruptible multi-access instructions: `0x0` No support. This means the LDM and STM instructions are not interruptible.
[7:4]	MemHint	Indicates the implemented memory hint instructions: `0x4` The `PLD` instruction. The `PLI` instruction. The `PLDW` instruction.
[3:0]	LoadStore	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.

To access the ID_ISAR2:

```
MRC p15, 0, <Rt>, c0, c2, 2 ; Read ID_ISAR2 into Rt
```

Register access is encoded as follows:

Table 4-192 ID_ISAR2 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	010

4.5.18 Instruction Set Attribute Register 3

The ID_ISAR3 characteristics are:

Purpose Provides information about the instruction sets implemented by the processor in AArch32.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR2, ID_ISAR4, and ID_ISAR5. See:

- *4.5.15 Instruction Set Attribute Register 0* on page 4-237.
- *4.5.16 Instruction Set Attribute Register 1* on page 4-239.
- *4.5.17 Instruction Set Attribute Register 2* on page 4-240.
- *4.5.19 Instruction Set Attribute Register 4* on page 4-244.
- *4.5.20 Instruction Set Attribute Register 5* on page 4-246.

Configurations ID_ISAR3 is architecturally mapped to AArch64 register ID_ISAR3_EL1. See *4.3.16 AArch32 Instruction Set Attribute Register 3* on page 4-102.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_ISAR3 is a 32-bit register.

The following figure shows the ID_ISAR3 bit assignments.

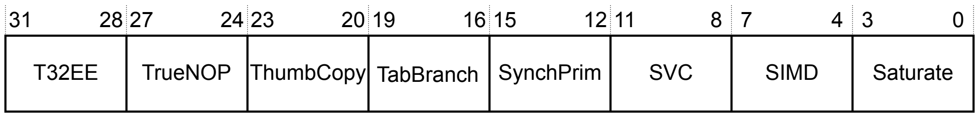

Figure 4-91 ID_ISAR3 bit assignments

The following table shows the ID_ISAR3 bit assignments.

Table 4-193 ID_ISAR3 bit assignments

Bits	Name	Function
[31:28]	T32EE	Indicates the implemented Thumb Execution Environment (T32EE) instructions: `0x0` None implemented.
[27:24]	TrueNOP	Indicates support for True NOP instructions: `0x1` True `NOP` instructions in both the A32 and T32 instruction sets, and additional NOP-compatible hints.
[23:20]	ThumbCopy	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.
[19:16]	TabBranch	Indicates the implemented Table Branch instructions in the T32 instruction set: `0x1` The `TBB` and `TBH` instructions.

Table 4-193 ID_ISAR3 bit assignments (continued)

Bits	Name	Function
[15:12]	SynchPrim	Indicates the implemented Synchronization Primitive instructions: `0x2` • The `LDREX` and `STREX` instructions. • The `CLREX`, `LDREXB`, `STREXB`, and `STREXH` instructions. • The `LDREXD` and `STREXD` instructions.
[11:8]	SVC	Indicates the implemented SVC instructions: `0x1` The `SVC` instruction.
[7:4]	SIMD	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.
[3:0]	Saturate	Indicates the implemented Saturate instructions: `0x1` The `QADD`, `QDADD`, `QDSUB`, `QSUB` and the Q bit in the PSRs.

To access the ID_ISAR3:

```
MRC p15, 0, <Rt>, c0, c2, 3 ; Read ID_ISAR3 into Rt
```

Register access is encoded as follows:

Table 4-194 ID_ISAR3 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	011

4.5.19 Instruction Set Attribute Register 4

The ID_ISAR4 characteristics are:

Purpose Provides information about the instruction sets implemented by the processor in AArch32.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR2, ID_ISAR3, and ID_ISAR5. See:

- *4.5.15 Instruction Set Attribute Register 0* on page 4-237.
- *4.5.16 Instruction Set Attribute Register 1* on page 4-239.
- *4.5.17 Instruction Set Attribute Register 2* on page 4-240.
- *4.5.18 Instruction Set Attribute Register 3* on page 4-242.
- *4.5.20 Instruction Set Attribute Register 5* on page 4-246.

Configurations

ID_ISAR4 is architecturally mapped to AArch64 register ID_ISAR4_EL1. See *4.3.17 AArch32 Instruction Set Attribute Register 4* on page 4-104.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_ISAR4 is a 32-bit register.

The following figure shows the ID_ISAR4 bit assignments.

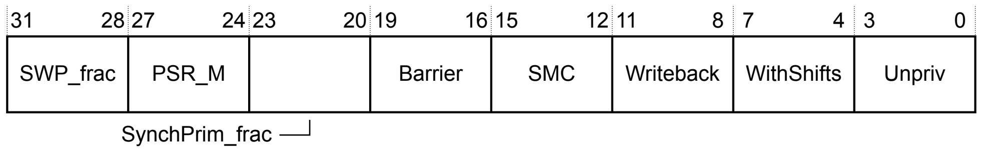

Figure 4-92 ID_ISAR4 bit assignments

The following table shows the ID_ISAR4 bit assignments.

Table 4-195 ID_ISAR4 bit assignments

Bits	Name	Function
[31:28]	SWP_frac	Indicates support for the memory system locking the bus for `SWP` or `SWPB` instructions: `0x0` `SWP` and `SWPB` instructions not implemented.
[27:24]	PSR_M	Indicates the implemented M profile instructions to modify the PSRs: `0x0` None implemented.
[23:20]	SynchPrim_frac	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.
[19:16]	Barrier	Indicates the supported Barrier instructions in the A32 and T32 instruction sets: `0x1` The `DMB`, `DSB`, and `ISB` barrier instructions.
[15:12]	SMC	Indicates the implemented `SMC` instructions: `0x1` The `SMC` instruction.

Table 4-195 ID_ISAR4 bit assignments (continued)

Bits	Name	Function
[11:8]	Writeback	Indicates the support for writeback addressing modes: `0x1` Processor supports all of the writeback addressing modes defined in Armv8.
[7:4]	WithShifts	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.
[3:0]	Unpriv	Indicates the implemented unprivileged instructions: `0x2` • The `LDRBT`, `LDRT`, `STRBT`, and `STRT` instructions. • The `LDRHT`, `LDRSBT`, `LDRSHT`, and `STRHT` instructions.

To access the ID_ISAR4:

```
MRC p15, 0, <Rt>, c0, c2, 4 ; Read ID_ISAR4 into Rt
```

Register access is encoded as follows:

Table 4-196 ID_ISAR4 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	100

4.5.20 Instruction Set Attribute Register 5

The ID_ISAR5 characteristics are:

Purpose Provides information about the instruction sets that the processor implements.

Note

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension only under an additional license to the Cortex-A73 MPCore processor and Advanced SIMD and floating-point support licenses.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

ID_ISAR5 must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR2, ID_ISAR3, and ID_ISAR4. See:

- *4.5.15 Instruction Set Attribute Register 0* on page 4-237.
- *4.5.16 Instruction Set Attribute Register 1* on page 4-239.
- *4.5.17 Instruction Set Attribute Register 2* on page 4-240.
- *4.5.18 Instruction Set Attribute Register 3* on page 4-242.
- *4.5.19 Instruction Set Attribute Register 4* on page 4-244.

Configurations ID_ISAR5 is architecturally mapped to AArch64 register ID_ISAR5_EL1. See *4.3.18 AArch32 Instruction Set Attribute Register 5* on page 4-105.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_ISAR5 is a 32-bit register.

The following figure shows the ID_ISAR5 bit assignments.

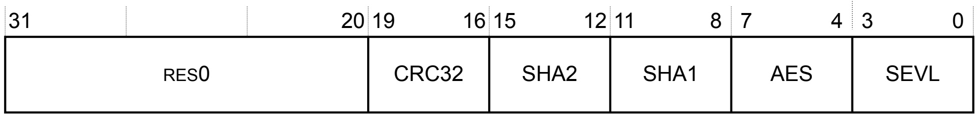

Figure 4-93 ID_ISAR5 bit assignments

The following table shows the ID_ISAR5 bit assignments.

Table 4-197 ID_ISAR5 bit assignments

Bits	Name	Function
[31:20]	-	Reserved, RES0.
[19:16]	CRC32	Indicates whether CRC32 instructions are implemented in AArch32 state. The value is: `0x1` CRC32 instructions are implemented.
[15:12]	SHA2	Indicates whether SHA2 instructions are implemented in AArch32 state. The possible values are: `0x0` No SHA2 instructions are implemented. This is the value if the implementation does not include the Cryptographic Extension. `0x1` `SHA256H`, `SHA256H2`, `SHA256SU0`, and `SHA256SU1` are implemented. This is the value if the implementation includes the Cryptographic Extension.
[11:8]	SHA1	Indicates whether SHA1 instructions are implemented in AArch32 state. The possible values are: `0x0` No SHA1 instructions are implemented. This is the value if the implementation does not include the Cryptographic Extension. `0x1` `SHA1C`, `SHA1P`, `SHA1M`, `SHA1H`, `SHA1SU0`, and `SHA1SU1` are implemented. This is the value if the implementation includes the Cryptographic Extension.
[7:4]	AES	Indicates whether AES instructions are implemented in AArch32 state. The possible values are: `0x0` No AES instructions are implemented. This is the value if the implementation does not include the Cryptographic Extension. `0x2` `AESE`, `AESD`, `AESMC`, and `AESIMC` are implemented, plus `PMULL` and `PMULL2` instructions operating on 64-bit elements. This is the value if the implementation includes the Cryptographic Extension.
[3:0]	SEVL	Indicates whether the `SEVL` instruction is implemented. The value is: `0x1` `SEVL` is implemented to send event local.

To access the ID_ISAR5:

```
MRC p15,0,<Rt>,c0,c2,5 ; Read ID_ISAR5 into Rt
```

Register access is encoded as follows:

Table 4-198 ID_ISAR5 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	101

4.5.21 Memory Model Feature Register 4

The ID_MMFR4 characteristics are:

Purpose Provides information about the memory model and memory management support in AArch32.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Configurations ID_MMFR4 is architecturally mapped to AArch64 register ID_MMFR4_EL1.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ID_MMFR4 is a 32-bit register.

The following figure shows the ID_MMFR4 bit assignments.

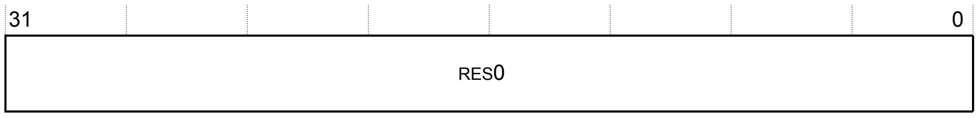

Figure 4-94 ID_MMFR4 bit assignments

The following table shows the ID_MMFR4 bit assignments.

Table 4-199 ID_MMFR4 bit assignments

Bits	Name	Function
[31:0]	-	Reserved, RES0.

To access the ID_MMFR4:

```
MRC p15,0,<Rt>,c0,c2,6 ; Read ID_MMFR4 into Rt
```

Register access is encoded as follows:

Table 4-200 ID_MMFR4 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	110

4.5.22 Cache Size ID Register

The CCSIDR characteristics are:

Purpose Provides information about the architecture of the caches.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

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).

Configurations CCSIDR is architecturally mapped to AArch64 register CCSIDR_EL1. See *4.3.30 Cache Size ID Register* on page 4-114.

There is one copy of this register that is used in both Secure and Non-secure states.

The implementation includes one CCSIDR for each cache that it can access. CSSELR selects which Cache Size ID Register is accessible.

Attributes CCSIDR is a 32-bit register.

The following figure shows the CCSIDR bit assignments.

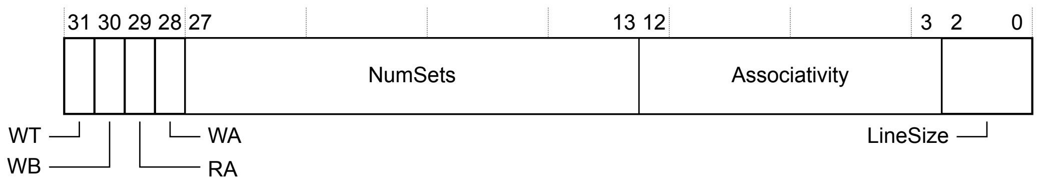

Figure 4-95 CCSIDR bit assignments

The following table shows the CCSIDR bit assignments.

Table 4-201 CCSIDR bit assignments

Bits	Name	Function
[31]	WT	Indicates support for Write-Through: `0` Cache level does not support Write-Through.
[30]	WB	Indicates support for Write-Back: `0` Cache level does not support Write-Back. `1` Cache level supports Write-Back.
[29]	RA	Indicates support for Read-Allocation: `0` Cache level does not support Read-Allocation. `1` Cache level supports Read-Allocation.
[28]	WA	Indicates support for Write-Allocation: `0` Cache level does not support Write-Allocation. `1` Cache level supports Write-Allocation.
[27:13]	NumSets[bk]	Indicates the number of sets in cache minus 1. Therefore, a value of 0 indicates 1 set in the cache. The number of sets does not have to be a power of 2.

Table 4-201 CCSIDR bit assignments (continued)

Bits	Name	Function
[12:3]	Associativity[bk]	Indicates the associativity of cache minus 1. Therefore, a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2.
[2:0]	LineSize[bk]	Indicates the ($\log_2$ (number of words in cache line)) minus 2: `0b010` 16 words per line.

The following table shows the individual bit field and complete register encodings for the CCSIDR. The CSSELR determines which CCSIDR to select.

Table 4-202 CCSIDR encodings

CSSELR	Cache	Size	Complete register encoding	Register bit field encoding						
				WT	WB	RA	WA	NumSets	Associativity	LineSize
`0x0`	L1 data cache	32KB	`0x7007E03A`	0	1	1	1	`0x3F`	`0x7`	`0x2`
		64KB	`0x7007E07A`					`0x3F`	`0xF`	`0x2`
`0x1`	L1 instruction cache	64KB	`0x201FE01A`	0	0	1	0	`0xFF`	`0x3`	`0x2`
`0x2`	L2 cache	256KB	`0x701FE07A`	0	1	1	1	`0xFF`	`0xF`	`0x2`
		512KB	`0x703FE07A`					`0x1FF`	`0xF`	`0x2`
		1MB	`0x707FE07A`					`0x3FF`	`0xF`	`0x2`
		2MB	`0x70FFE07A`					`0x7FF`	`0xF`	`0x2`
		4MB	`0x71FFE07A`					`0xFFF`	`0xF`	`0x2`
		8MB	`0x73FFE07A`					`0x1FFF`	`0xF`	`0x2`
`0x3-0xF`	Reserved	-	-	-	-	-	-	-	-	-

To access the CCSIDR:

```
MRC p15, 1, <Rt>, c0, c0, 0 ; Read CCSIDR into Rt
```

Register access is encoded as follows:

Table 4-203 CCSIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	001	0000	0000	000

4.5.23 Cache Level ID Register

The CLIDR characteristics are:

Purpose Identifies:
- The type of cache, or caches, implemented at each level.
- The Level of Coherency and Level of Unification for the cache hierarchy.

[bk] For more information about encoding, see *Table 4-202 CCSIDR encodings* on page 4-250.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Configurations CLIDR is architecturally mapped to AArch64 register CLIDR_EL1. See *4.3.31 Cache Level ID Register* on page 4-115.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes CLIDR is a 32-bit register.

The following figure shows the CLIDR bit assignments.

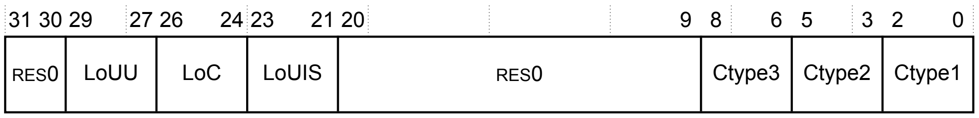

Figure 4-96 CLIDR bit assignments

The following table shows the CLIDR bit assignments.

Table 4-204 CLIDR bit assignments

Bits	Name	Function
[31:30]	-	Reserved, RES0.
[29:27]	LoUU	Indicates the Level of Unification Uniprocessor for the cache hierarchy: `0b001` L1 cache is the last level of cache that must be cleaned or invalidated when cleaning or invalidating to the point of unification for the processor.
[26:24]	LoC	Indicates the Level of Coherency for the cache hierarchy: `0b001` L2 cache not implemented. `0b010` A clean to the point of coherency operation requires the L1 and L2 caches to be cleaned.
[23:21]	LoUIS	Indicates the Level of Unification Inner Shareable for the cache hierarchy: `0b001` L1 cache is the last level of cache that must be cleaned or invalidated when cleaning or invalidating to the point of unification for the Inner Shareable shareability domain.
[20:9]	-	Reserved, RES0.
[8:6]	Ctype3[bl]	Indicates the type of cache if the processor implements L3 cache: `0b000` L3 cache not implemented.

[bl] If software reads the Cache Type fields from Ctype1 upwards, after it has seen a value of `0b000`, no caches exist at further-out levels of the hierarchy. So, for example, if Ctype2 is the first Cache Type field with a value of `0b000`, the value of Ctype3 must be ignored.

Table 4-204 CLIDR bit assignments (continued)

Bits	Name	Function
[5:3]	Ctype2	Indicates the type of cache if the processor implements L2 cache: `0b000` L2 cache is not implemented. `0b100` L2 cache is implemented as a unified cache.
[2:0]	Ctype1	Indicates the type of cache implemented at L1: `0b011` Separate instruction and data caches at L1.

To access the CLIDR:

```
MRC p15,1,<Rt>,c0,c0,1 ; Read CLIDR into Rt
```

Register access is encoded as follows:

Table 4-205 CLIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	001	0000	0000	001

4.5.24 Auxiliary ID Register

The processor does not implement AIDR, so this register is always RES0.

4.5.25 Cache Size Selection Register

The CSSELR characteristics are:

Purpose Selects the current CCSIDR, see *4.5.22 Cache Size ID Register* on page 4-248, by specifying:
- The required cache level.
- The cache type, either instruction or data cache.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

If the CSSELR level field is programmed to a cache level that is not implemented, then a read of CSSELR returns an UNKNOWN value in CSSELR.Level.

Configurations CSSELR (NS) is architecturally mapped to AArch64 register CSSELR_EL1. See *4.3.33 Cache Size Selection Register* on page 4-117.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

Attributes CSSELR is a 32-bit register.

The following figure shows the CSSELR bit assignments.

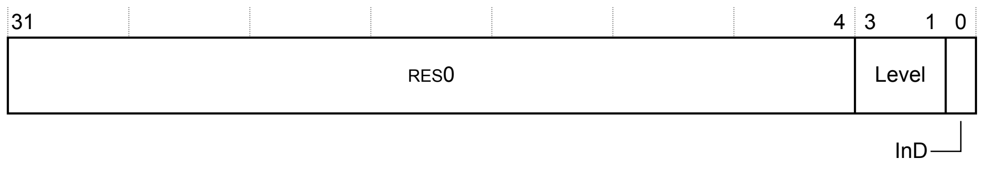

Figure 4-97 CSSELR bit assignments

The following table shows the CSSELR bit assignments.

Table 4-206 CSSELR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:1]	Level[bm]	Cache level of required cache: `0b000` L1. `0b001` L2. `0b010-0b111` Reserved.
[0]	InD[bm]	Instruction not Data bit: `0` Data or unified cache. `1` Instruction cache.

To access the CSSELR:

```
MRC p15, 2, <Rt>, c0, c0, 0; Read CSSELR into Rt
MCR p15, 2, <Rt>, c0, c0, 0; Write Rt to CSSELR
```

Register access is encoded as follows:

Table 4-207 CSSELR access encoding

coproc	opc1	CRn	CRm	opc2
1111	010	0000	0001	000

4.5.26 Cache Type Register

The CTR_EL0 characteristics are:

Purpose Provides information about the architecture of the caches.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

bm The combination of Level=`0b001` and InD=`1` is reserved.

Configurations CTR is architecturally mapped to AArch64 register CTR_EL0. See *4.3.34 Cache Type Register* on page 4-118.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes CTR is a 32-bit register.

The following figure shows the CTR bit assignments.

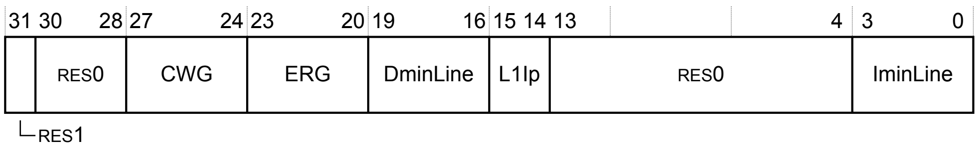

Figure 4-98 CTR bit assignments

The following table shows the CTR bit assignments.

Table 4-208 CTR bit assignments

Bits	Name	Function
[31]	-	Reserved, RES1.
[30:28]	-	Reserved, RES0.
[27:24]	CWG	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: `0x4` Cache Write-Back granule size is 16 words.
[23:20]	ERG	Exclusives Reservation Granule. Log_2 of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions: `0x4` Exclusive reservation granule size is 16 words.
[19:16]	DminLine	Log_2 of the number of words in the smallest cache line of all the data and unified caches that the processor controls: `0x4` Smallest data cache line size is 16 words.
[15:14]	L1Ip	L1 Instruction cache policy. Indicates the indexing and tagging policy for the L1 Instruction cache: `0b10` *Virtually Indexed Physically Tagged* (VIPT).
[13:4]	-	Reserved, RES0.
[3:0]	IminLine	Log_2 of the number of words in the smallest cache line of all the instruction (L1) and unified (L2) caches that the processor controls. `0x4` Smallest instruction cache line size is 16 words.

To access the CTR:

```
MRC p15,0,<Rt>,c0,c0,1 ; Read CTR into Rt
```

Register access is encoded as follows:

Table 4-209 CTR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0000	001

4.5.27 Virtualization Processor ID Register

The VPIDR characteristics are:

Purpose Holds the value of the Virtualization Processor ID. This is the value returned by Non-secure EL1 reads of MIDR. See *4.5.1 Main ID Register* on page 4-222.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations VPIDR is architecturally mapped to AArch64 register VPIDR_EL2. See *4.3.36 Virtualization Processor ID Register* on page 4-120.

Attributes VPIDR is a 32-bit register.

VPIDR resets to the value of MIDR.

The following figure shows the VPIDR bit assignments.

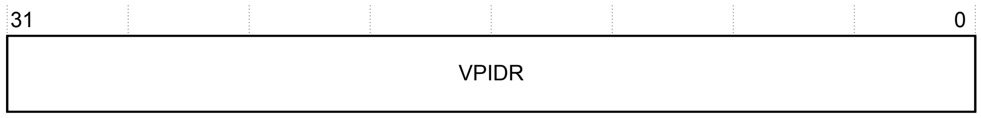

Figure 4-99 VPIDR bit assignments

The following table shows the VPIDR bit assignments.

Table 4-210 VPIDR bit assignments

Bits	Name	Function
[31:0]	VPIDR	MIDR value returned by Non-secure PL1 reads of the MIDR. The MIDR description defines the subdivision of this value. See *4.5.1 Main ID Register* on page 4-222.

To access the VPIDR:

```
MRC p15,4,<Rt>,c0,c0,0 ; Read VPIDR into Rt
MCR p15,4,<Rt>,c0,c0,0 ; Write Rt to VPIDR
```

Register access is encoded as follows:

Table 4-211 VPIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0000	0000	000

4.5.28 Virtualization Multiprocessor ID Register

The VMPIDR characteristics are:

Purpose Provides the value of the Virtualization Multiprocessor ID. This is the value returned by Non-secure EL1 reads of MPIDR.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations VMPIDR is architecturally mapped to AArch64 register VMPIDR_EL2[31:0]. See *4.3.37 Virtualization Multiprocessor ID Register* on page 4-121.

This register is accessible only at EL2 or EL3.

Attributes VMPIDR is a 32-bit register.

VMPIDR resets to the value of MPIDR.

The following figure shows the VMPIDR bit assignments.

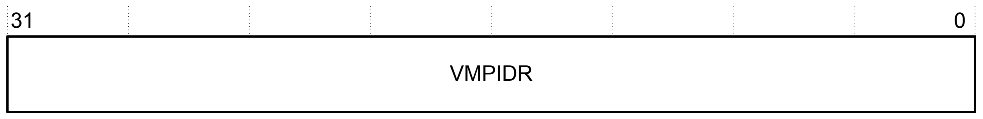

Figure 4-100 VMPIDR bit assignments

The following table shows the VMPIDR bit assignments.

Table 4-212 VMPIDR bit assignments

Bits	Name	Function
[31:0]	VMPIDR	MPIDR value returned by Non-secure EL1 reads of the MPIDR. The MPIDR description defines the subdivision of this value. See *4.5.2 Multiprocessor Affinity Register* on page 4-223.

To access the VMPIDR:

```
MRC p15,4,<Rt>,c0,c0,5 ; Read VMPIDR into Rt
MCR p15,4,<Rt>,c0,c0,5 ; Write Rt to VMPIDR
```

Register access is encoded as follows:

Table 4-213 VMPIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0000	0000	101

4.5.29 System Control Register

The SCTLR characteristics are:

Purpose Provides the top level control of the system, including its memory system.

Usage constraints The SCTLR is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Control bits in the SCTLR that are not applicable to a VMSA implementation read as the value that most closely reflects that implementation, and ignore writes.

Some bits in the register are read-only. These bits relate to non-configurable features of an implementation, and are provided for compatibility with previous versions of the architecture.

Configurations SCTLR (NS) is architecturally mapped to AArch64 register SCTLR_EL1. See *4.3.38 System Control Register, EL1* on page 4-122.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

Attributes SCTLR is a 32-bit register.

The following figure shows the SCTLR bit assignments.

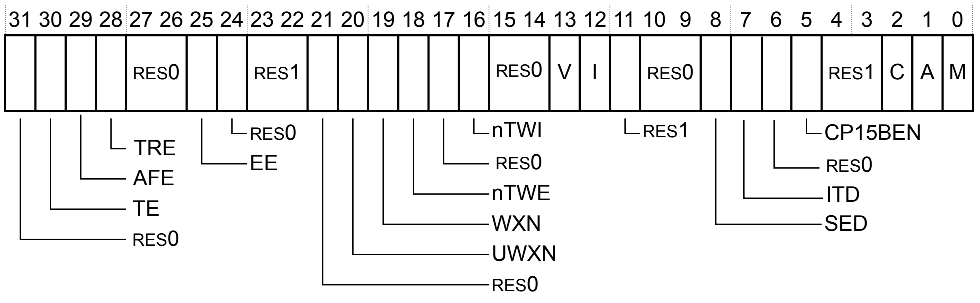

Figure 4-101 SCTLR bit assignments

The following table shows the SCTLR bit assignments.

Table 4-214 SCTLR bit assignments

Bits	Name	Function
[31]	-	Reserved, RES0.
[30]	TE	T32 Exception enable. This bit controls whether exceptions are taken in A32 or T32 state: `0` Exceptions, including reset, taken in A32 state. `1` Exceptions, including reset, taken in T32 state. The input **CFGTE** defines the reset value of the TE bit.
[29]	AFE	Access Flag Enable. This bit enables use of the AP[0] bit in the translation descriptors as the Access flag. It also restricts access permissions in the translation descriptors to the simplified model: `0` In the translation table descriptors, AP[0] is an access permissions bit. The full range of access permissions is supported. No Access flag is implemented. This is the reset value. `1` In the translation table descriptors, AP[0] is the Access flag. Only the simplified model for access permissions is supported.

Table 4-214 SCTLR bit assignments (continued)

Bits	Name	Function
[28]	TRE	TEX remap enable. This bit enables remapping of the TEX[2:1] bits for use as two translation table bits that can be managed by the operating system. Enabling this remapping also changes the scheme used to describe the memory region attributes in the VMSA: `0` TEX remap disabled. TEX[2:0] are used, with the C and B bits, to describe the memory region attributes. This is the reset value. `1` TEX remap enabled. TEX[2:1] are reassigned for use as bits managed by the operating system. The TEX[0], C and B bits are used to describe the memory region attributes, with the MMU remap registers.
[27:26]	-	Reserved, RES0.
[25]	EE	Exception Endianness bit. The value of this bit defines the value of the CPSR.E bit on entry to an exception vector, including reset. This value also indicates the endianness of the translation table data for translation table lookups: `0` Little endian. `1` Big endian. The input **CFGEND** defines the reset value of the EE bit.
[24]	-	Reserved, RES0.
[23:22]	-	Reserved, RES1.
[21]	-	Reserved, RES0.
[20]	UWXN	Unprivileged write permission implies EL1 *Execute Never* (XN). This bit can be used to require all memory regions with unprivileged write permissions to be treated as XN for accesses from software executing at EL1. `0` Regions with unprivileged write permission are not forced to be XN, this is the reset value. `1` Regions with unprivileged write permission are forced to be XN for accesses from software executing at EL1.
[19]	WXN	Write permission implies *Execute Never* (XN). This bit can be used to require all memory regions with write permissions to be treated as XN. `0` Regions with write permission are not forced to be XN. This is the reset value. `1` Regions with write permissions are forced to be XN.
[18]	nTWE	Not trap WFE. `0` If a WFE instruction executed at EL0 would cause execution to be suspended, such as if the event register is not set and there is not a pending WFE wakeup event, it is taken as an exception to EL1 using the 0x1 ESR code. `1` WFE instructions are executed as normal. The reset value is `0b1`.
[17]	-	Reserved, RES0.

Table 4-214 SCTLR bit assignments (continued)

Bits	Name	Function
[16]	nTWI	Not trap WFI. `0` If a WFI instruction executed at EL0 would cause execution to be suspended, such as if there is not a pending WFI wakeup event, it is taken as an exception to EL1 using the 0x1 ESR code. `1` WFI instructions are executed as normal. The reset value is `0b1`.
[15:14]	-	Reserved, RES0.
[13]	V	Vectors bit. This bit selects the base address of the exception vectors: `0` Normal exception vectors, base address `0x00000000`. Software can remap this base address using the VBAR. `1` High exception vectors, base address `0xFFFF0000`. This base address is never remapped. The input **VINITHI** defines the reset value of the V bit.
[12]	I	Instruction cache enable. The possible values are: `0` Instruction (L1) and unified (L2) caches disabled for instruction fetch. This is the reset value. `1` Instruction (L1) and unified (L2) caches enabled for instruction fetch.
[11]	-	Reserved, RES1
[10:9]	-	Reserved, RES0
[8]	SED	SETEND Disable: `0` The SETEND instruction is available. This is the reset value. `1` The SETEND instruction is unallocated.
[7]	ITD	Reserved, RES0 IT Disable. Disables some uses of IT instructions at PL1 and PL0.
[6]	-	Reserved, RES0
[5]	CP15BEN	CP15 barrier enable. `0` CP15 barrier operations disabled. Their encodings are UNDEFINED. `1` CP15 barrier operations enabled. The reset value is `0b1`.
[4:3]	-	Reserved, RES1.
[2]	C	Cache enable. This is a global enable bit for data and unified caches: `0` Data (L1) and unified (L2) caches disabled for data access. This is the reset value. `1` Data (L1) and unified (L2) caches enabled for data access.

Table 4-214 SCTLR bit assignments (continued)

Bits	Name	Function
[1]	A	Alignment check enable. This is the enable bit for Alignment fault checking: `0` Alignment fault checking disabled. This is the reset value. `1` Alignment fault checking enabled.
[0]	M	MMU enable. This is a global enable bit for the MMU stage 1 address translation: `0` EL1 and EL0 stage 1 MMU disabled. This is the reset value. `1` EL1 and EL0 stage 1 MMU enabled.

To access the SCTLR:

```
MRC p15, 0, <Rt>, c1, c0, 0 ; Read SCTLR into Rt
MCR p15, 0, <Rt>, c1, c0, 0 ; Write Rt to SCTLR
```

4.5.30 Auxiliary Control Register

The ACTLR characteristics are:

Purpose Controls write access to IMPLEMENTATION DEFINED registers in EL2, such as ECTLR, L2CTLR, and L2ECTLR.

Usage constraints This register is accessible as follows:

EL0 NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Configurations The processor does not implement the ACTLR (NS) register. This register is always RES0. It is mapped to AArch64 register ACTLR_EL1. See *4.3.39 Auxiliary Control Register, EL1* on page 4-125.

ACTLR (S) is mapped to AArch64 register ACTLR_EL3. See *4.3.41 Auxiliary Control Register, EL3* on page 4-127.

Attributes ACTLR is a 32-bit register.

The following figure shows the ACTLR bit assignments.

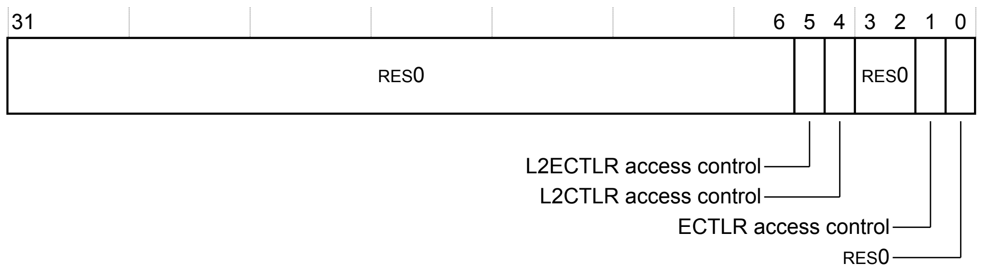

Figure 4-102 ACTLR bit assignments

The following table shows the ACTLR bit assignments.

Table 4-215 ACTLR bit assignments

Bits	Name	Function
[31:6]	-	Reserved, RES0.
[5]	L2ECTLR access control	L2ECTLR write access control. The possible values are: `0` The register is not write accessible from a lower exception level. This is the reset value. `1` The register is write accessible from EL2.
[4]	L2CTLR access control	L2CTLR write access control. The possible values are: `0` The register is not write accessible from a lower exception level. This is the reset value. `1` The register is write accessible from EL2.
[3:2]	-	Reserved, RES0.
[1]	ECTLR access control	ECTLR write access control. The possible values are: `0` The register is not write accessible from a lower exception level. This is the reset value. `1` The register is write accessible from EL2.
[0]	-	Reserved, RES0.

To access the ACTLR:

```
MRC p15, 0, <Rt>, c1, c0, 1 ; Read ACTLR into Rt
MCR p15, 0, <Rt>, c1, c0, 1 ; Write Rt to ACTLR
```

4.5.31 Architectural Feature Access Control Register

The CPACR characteristics are:

Purpose Controls access to CP0 to CP13, and indicates which of CP0 to CP13 are implemented.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

The CPACR has no effect on instructions executed at EL2.

Configurations CPACR is architecturally mapped to AArch64 register CPACR_EL1. See *4.3.42 Architectural Feature Access Control Register* on page 4-128.

There is one copy of this register that is used in both Secure and Non-secure states.

Bits in the NSACR control Non-secure access to the CPACR fields. See the field descriptions cp10 and cp11.

Attributes CPACR is a 32-bit register.

The following figure shows the CPACR bit assignments.

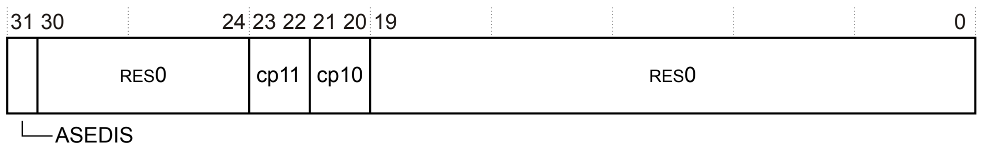

Figure 4-103 CPACR bit assignments

The following table shows the CPACR bit assignments.

Table 4-216 CPACR bit assignments

Bits	Name	Function
[31]	ASEDIS	Disable Advanced SIMD Functionality: `0` Does not cause any instructions to be UNDEFINED. This is the reset value. `1` All instruction encodings that are part of Advanced SIMD, but that are not floating-point instructions, are UNDEFINED.
[30:24]	-	Reserved, RES0.
[23:22]	cp11[bn][bo]	Defines the access rights for CP11, that control the Advanced SIMD and floating-point features. Possible values of the fields are: `0b00` Access denied. Any attempt to access Advanced SIMD and floating-point registers or instructions generates an Undefined Instruction exception. This is the reset value. `0b01` Access at EL1 only. Any attempt to access Advanced SIMD and floating-point registers or instructions from software executing at EL0 generates an Undefined Instruction exception. `0b10` Reserved. `0b11` Full access.
[21:20]	cp10[bo]	Defines the access rights for CP10, that control the Advanced SIMD and floating-point features. Possible values of the fields are: `0b00` Access denied. Any attempt to access Advanced SIMD and floating-point registers or instructions generates an Undefined Instruction exception. This is the reset value. `0b01` Access at EL1 only. Any attempt to access Advanced SIMD and floating-point registers or instructions from software executing at EL0 generates an Undefined Instruction exception. `0b10` Reserved. `0b11` Full access.
[19:0]	-	Reserved, RES0.

To access the CPACR:

```
MRC p15,0,<Rt>,c1,c0,2 ; Read CPACR into Rt
MCR p15,0,<Rt>,c1,c0,2 ; Write Rt to CPACR
```

[bn] The floating-point and Advanced SIMD features controlled by these fields are:
- Floating-point instructions.
- Advanced SIMD instructions, both integer and floating-point.
- Advanced SIMD and floating-point registers D0-D31 and their views as S0-S31 and Q0-Q15.
- FPSCR, FPSID, MVFR0, MVFR1, MVFR2, FPEXC system registers.

[bo] If the cp11 and cp10 fields are set to different values, the behavior is the same as if both fields were set to the value of cp10, in all respects other than the value read back by explicitly reading cp11.

4.5.32 Secure Configuration Register

The SCR characteristics are:

Purpose Defines the configuration of the current security state. It specifies:

- The security state of the processor, Secure or Non-secure.
- What state the processor branches to, if an IRQ, FIQ or external abort occurs.
- Whether the CPSR.F and CPSR.A bits can be modified when SCR.NS = 1.

Usage constraints This register is accessible as follows:

EL0 NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	-	RW	RW

Any read or write to SCR in Secure EL1 state in AArch32 is trapped as an exception to EL3.

Configurations The SCR is a Restricted access register that exists only in the Secure state.

The SCR is mapped to the AArch64 SCR_EL3 register.

Attributes SCR is a 32-bit register.

The following figure shows the SCR bit assignments.

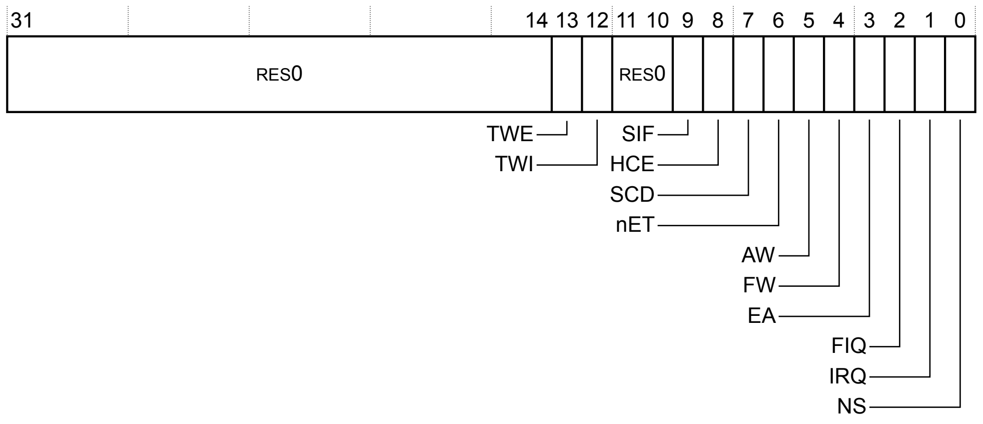

Figure 4-104 SCR bit assignments

The following table shows the SCR bit assignments.

Table 4-217 SCR bit assignments

Bits	Name	Function
[31:14]	-	Reserved, RES0.
[13]	TWE	Trap `WFE` instructions. The possible values are: `0` `WFE` instructions are not trapped. This is the reset value. `1` `WFE` instructions executed in any mode other than Monitor mode are trapped to Monitor mode as UNDEFINED if the instruction would otherwise cause suspension of execution, that is if: • The event register is not set. • There is not a pending WFE wakeup event. • The instruction does not cause another exception.

Table 4-217 SCR bit assignments (continued)

Bits	Name	Function
[12]	TWI	Trap `WFI` instructions. The possible values are: `0` `WFI` instructions are not trapped. This is the reset value. `1` `WFI` instructions executed in any mode other than Monitor mode are trapped to Monitor mode as UNDEFINED if the instruction would otherwise cause suspension of execution.
[11:10]	-	Reserved, RES0.
[9]	SIF	Secure Instruction Fetch. When the processor is in Secure state, this bit disables instruction fetches from Non-secure memory. The possible values are: `0` Secure state instruction fetches from Non-secure memory permitted. This is the reset value. `1` Secure state instruction fetches from Non-secure memory not permitted.
[8]	HCE	Hyp Call enable. This bit enables use of the `HVC` instruction from Non-secure EL1 modes. The possible values are: `0` The `HVC` instruction is UNDEFINED in any mode. This is the reset value. `1` The `HVC` instruction enabled in Non-secure EL1, and performs a Hyp Call.
[7]	SCD	Secure Monitor Call disable. Makes the `SMC` instruction UNDEFINED in Non-secure state. The possible values are: `0` `SMC` executes normally in Non-secure state, performing a Secure Monitor Call. This is the reset value. `1` The `SMC` instruction is UNDEFINED in Non-secure state. A trap of the `SMC` instruction to Hyp mode takes priority over the value of this bit.
[6]	nET	Not Early Termination. This bit disables early termination. This bit is not implemented, RES0.
[5]	AW	A bit writable. This bit controls whether CPSR.A can be modified in Non-secure state. • CPSR.A can be modified only in Secure state. This is the reset value. • CPSR.A can be modified in any security state.
[4]	FW	F bit writable. This bit controls whether CPSR.F can be modified in Non-secure state: • CPSR.F can be modified only in Secure state. This is the reset value. • CPSR.F can be modified in any security state.
[3]	EA	External Abort handler. This bit controls which mode takes external aborts. The possible values are: `0` External aborts taken in abort mode. This is the reset value. `1` External aborts taken in Monitor mode.
[2]	FIQ	FIQ handler. This bit controls which mode takes FIQ exceptions. The possible values are: `0` FIQs taken in FIQ mode. This is the reset value. `1` FIQs taken in Monitor mode.

Table 4-217 SCR bit assignments (continued)

Bits	Name	Function
[1]	IRQ	IRQ handler. This bit controls which mode takes IRQ exceptions. The possible values are: `0` IRQs taken in IRQ mode. This is the reset value. `1` IRQs taken in Monitor mode.
[0]	NS	Non-secure bit. Except when the processor is in Monitor mode, this bit determines the security state of the processor. The possible values are: `0` Processor is in Secure state. This is the reset value. `1` Processor is in Non-secure state.

To access the SCR:

```
MRC p15,0,<Rt>,c1,c1,0 ; Read SCR into Rt
MCR p15,0,<Rt>,c1,c1,0 ; Write Rt to SCR
```

4.5.33 Secure Debug Enable Register

The SDER characteristics are:

Purpose Controls invasive and non-invasive debug in the Secure EL0 state.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	-	RW	RW

Configurations SDER is architecturally mapped to AArch64 register SDER32_EL3. See *4.3.51 Secure Debug Enable Register* on page 4-147.

This register is accessible only in Secure state.

Attributes SDER is a 32-bit register.

The following figure shows the SDER bit assignments.

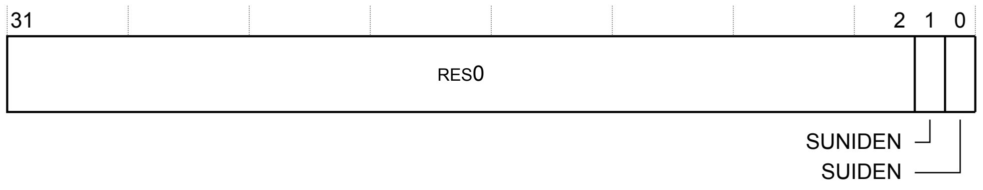

Figure 4-105 SDER bit assignments

The following table shows the SDER bit assignments.

Table 4-218 SDER bit assignments

Bits	Name	Function
[31:2]	-	Reserved, RES0.
[1]	SUNIDEN	Secure User Non-invasive Debug Enable. The possible values are: `0` Non-invasive debug not permitted in Secure EL0 state. This is the Warm reset value. `1` Non-invasive debug permitted in Secure EL0 state.
[0]	SUIDEN	Secure User Invasive Debug Enable. The possible values are: `0` Invasive debug not permitted in Secure EL0 state. This is the Warm reset value. `1` Invasive debug permitted in Secure EL0 state.

To access the SDER:

```
MRC p15,0,<Rt>,c1,c1,1 ; Read SDER into Rt
MCR p15,0,<Rt>,c1,c1,1 ; Write Rt to SDER
```

4.5.34 Non-Secure Access Control Register

The NSACR characteristics are:

Purpose Defines the Non-secure access permission to CP0 to CP13.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RW	RW

Any read or write to NSACR in Secure EL1 state in AArch32 is trapped as an exception to EL3.

Configurations There is one copy of this register that is used in both Secure and Non-secure states.

If EL3 is using AArch64, then any reads of the NSACR from Non-secure EL2 or Non-secure EL1 using AArch32 return a fixed value of `0x00000C00`.

In AArch64, the NSACR functionality is replaced by the behavior in CPTR_EL3.

Attributes NSACR is a 32-bit register.

The following figure shows the NSACR bit assignments.

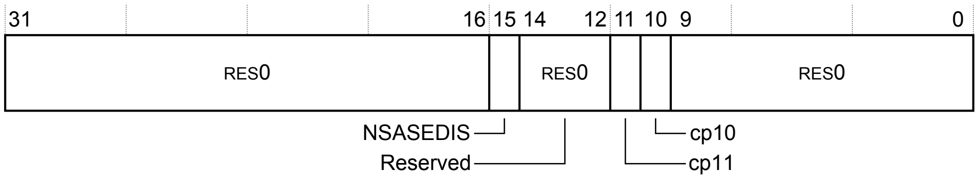

Figure 4-106 NSACR bit assignments

The following table shows the NSACR bit assignments.

Table 4-219 NSACR bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15]	NSASEDIS	Disable Non-secure Advanced SIMD functionality: `0` This bit has no effect on the ability to write CPACR.ASEDIS, this is the reset value. `1` When executing in Non-secure state, the CPACR.ASEDIS bit has a fixed value of 1 and writes to it are ignored.
[14:12]	-	Reserved, RES0.
[11]	cp11	Non-secure access to CP11 enable: `0` Secure access only. Any attempt to access CP11 in Non-secure state results in an Undefined Instruction exception. If the processor is in Non-secure state, the corresponding bits in the CPACR ignore writes and read as `0b00`, access denied. This is the reset value. `1` Secure or Non-secure access.
[10]	cp10	Non-secure access to CP10 enable: `0` Secure access only. Any attempt to access CP10 in Non-secure state results in an Undefined Instruction exception. If the processor is in Non-secure state, the corresponding bits in the CPACR ignore writes and read as `0b00`, access denied. This is the reset value. `1` Secure or Non-secure access.
[9:0]	-	Reserved, RES0.

——— **Note** ———

If the CP11 and CP10 fields are set to different values, the behavior is CONSTRAINED UNPREDICTABLE. It is the same as if both fields were set to the value of CP10, in all respects other than the value read back by explicitly reading CP11.

———————

To access the NSACR:

```
MRC p15, 0, <Rt>, c1, c1, 2 ; Read NSACR into Rt
MCR p15, 0, <Rt>, c1, c1, 2 ; Write Rt to NSACR
```

4.5.35 Secure Debug Configuration Register

The SDCR characteristics are:

Purpose Controls debug and performance monitors functionality in Secure state.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	-	RW	RW

Configurations SDCR is mapped to AArch64 register MDCR_EL3.

Attributes SDCR is a 32-bit register.

The following figure shows the SDCR bit assignments.

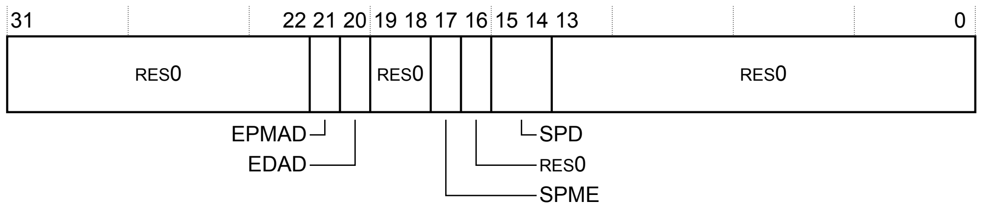

Figure 4-107 SDCR bit assignments

The following table shows the SDCR bit assignments

Table 4-220 SDCR bit assignments

Bits	Name	Function
[31:22]	-	Reserved, RES0.
[21]	EPMAD	Disables access to the performance monitor configuration registers by an external debugger: `0` Access to Performance Monitors registers from external debugger is permitted. This is the reset value. `1` Access to Performance Monitors registers from external debugger is disabled, unless overridden by authentication interface.
[20]	EDAD	Disables access to the breakpoint and watchpoint registers by an external debugger: `0` Access to breakpoint and watchpoint registers from external debugger is permitted. This is the reset value. `1` Access to breakpoint and watchpoint registers from external debugger is disabled, unless overridden by authentication interface.
[19:18]	-	Reserved, RES0.
[17]	SPME	Secure performance monitors enable. This allows event counting in Secure state: `0` Event counting prohibited in Secure state, unless overridden by the authentication interface. This is the reset value. `1` Event counting allowed in Secure state.
[16]	-	Reserved, RES0.
[15:14]	SPD	AArch32 secure privileged debug. Enables or disables debug exceptions in Secure state, other than Software breakpoint instructions. The possible values are: `0b00` Legacy mode. Debug exceptions from Secure EL1 are enabled by the authentication interface. `0b10` Secure privileged debug disabled. Debug exceptions from Secure EL1 are disabled. `0b11` Secure privileged debug enabled. Debug exceptions from Secure EL1 are enabled. The value `0b01` is reserved. If debug exceptions from Secure EL1 are enabled, then debug exceptions from Secure EL0 are also enabled. Otherwise, debug exceptions from Secure EL0 are enabled only if SDER32_EL3.SUIDEN is 1. SPD is ignored in Non-secure state. Debug exceptions from Software breakpoint instruction debug events are always enabled. The reset value is `0b00`.
[13:0]	-	Reserved, RES0.

To access the SDCR:

```
MRC p15,0,<Rt>,c1,c3,1 ; Read SDCR into Rt
MCR p15,0,<Rt>,c1,c3,1 ; Write Rt to SDCR
```

4.5.36 Hyp Auxiliary Control Register

The HACTLR characteristics are:

Purpose Controls write access to IMPLEMENTATION DEFINED registers in Non-secure EL1 modes, such as ECTLR, L2CTLR, and L2ECTLR.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations The HACTLR is architecturally mapped to the AArch4 ACTLR_EL2 register. See *4.3.40 Auxiliary Control Register, EL2* on page 4-126.

Attributes HACTLR is a 32-bit register.

The following figure shows the HACTLR bit assignments.

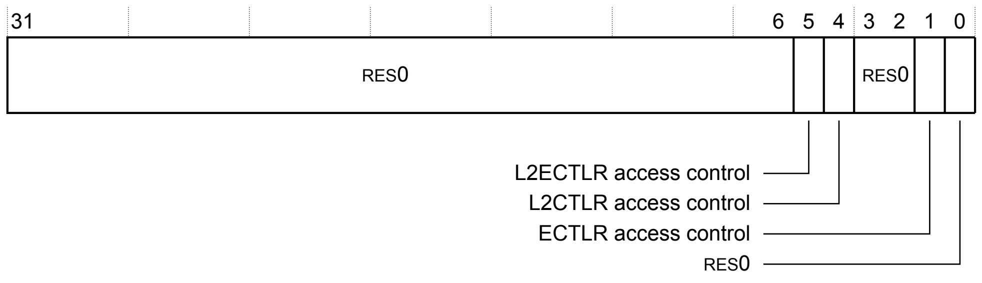

Figure 4-108 HACTLR bit assignments

The following table shows the HACTLR bit assignments.

Table 4-221 HACTLR bit assignments

Bits	Name	Function
[31:6]	-	Reserved, RES0.
[5]	L2ECTLR access control	L2ECTLR write access control. The possible values are: `0` The register is not write accessible from Non-secure EL1. This is the reset value. `1` The register is write accessible from Non-secure EL1. Write access from Non-secure EL1 also requires ACTLR(S)[5] to be set.

Table 4-221 HACTLR bit assignments (continued)

Bits	Name	Function
[4]	L2CTLR access control	L2CTLR write access control. The possible values are: `0` The register is not write accessible from Non-secure EL1. This is the reset value. `1` The register is write accessible from Non-secure EL1. Write access from Non-secure EL1 also requires ACTLR(S)[4] to be set.
[3:2]	-	Reserved, RES0.
[1]	ECTLR access control	ECTLR write access control. The possible values are: `0` The register is not write accessible from Non-secure EL1. This is the reset value. `1` The register is write accessible from Non-secure EL1. Write access from Non-secure EL1 also requires ACTLR(S)[1] to be set.
[0]		Reserved, RES0.

To access the HACTLR:

```
MRC p15,4,<Rt>,c1,c0,1 ; Read HACTLR into Rt
MCR p15,4,<Rt>,c1,c0,1 ; Write Rt to HACTLR
```

4.5.37 Hyp System Control Register

The HSCTLR characteristics are:

Purpose Provides top level control of the system operation in Hyp mode. This register provides Hyp mode control of features controlled by the Banked SCTLR bits, and shows the values of the non-Banked SCTLR bits.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations HSCTLR is architecturally mapped to AArch64 register SCTLR_EL2. See *4.3.43 System Control Register, EL2* on page 4-129.

Attributes HSCTLR is a 32-bit register.

The following figure shows the HSCTLR bit assignments.

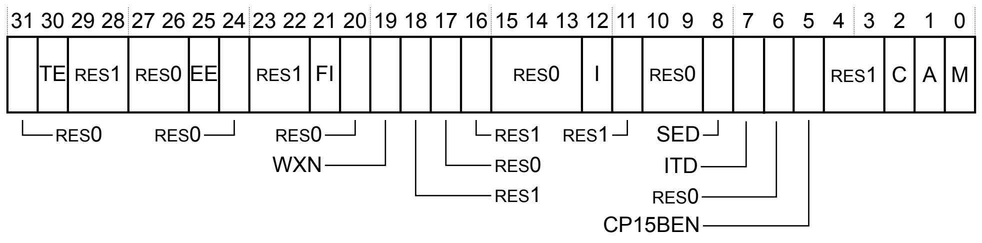

Figure 4-109 HSCTLR bit assignments

The following table shows the HSCTLR bit assignments.

Table 4-222 HSCTLR bit assignments

Bits	Name	Function
[31]	-	Reserved, RES0.
[30]	TE	Thumb Exception enable. This bit controls whether exceptions taken in Hyp mode are taken in A32 or T32 state: `0` Exceptions taken in A32 state. `1` Exceptions taken in T32 state.
[29:28]	-	Reserved, RES1.
[27:26]	-	Reserved, RES0.
[25]	EE	Exception Endianness. The value of this bit defines the value of the CPSR.E bit on entry to an exception vector, including reset. This value also indicates the endianness of the translation table data for translation table lookups: `0` Little endian. `1` Big endian. The reset value is UNKNOWN.
[24]	-	Reserved, RES0.
[23:22]	-	Reserved, RES1.
[21]	FI	Fast Interrupts configuration enable bit. This bit can be used to reduce interrupt latency by disabling implementation-defined performance features. This bit is not implemented, RES0.
[20]	-	Reserved, RES0.
[19]	WXN	Write permission implies *Execute Never* (XN). This bit can be used to require all memory regions with write permission to be treated as XN: `0` Regions with write permission are not forced to XN. `1` Regions with write permission are forced to XN. The WXN bit is permitted to be cached in a TLB. The reset value is UNKNOWN.
[18]	-	Reserved, RES1.
[17]	-	Reserved, RES0.

Table 4-222 HSCTLR bit assignments (continued)

Bits	Name	Function
[16]	-	Reserved, RES1.
[15:13]	-	Reserved, RES0.
[12]	I	Instruction cache enable. This is an enable bit for instruction caches at EL2: `0` Instruction caches disabled at EL2. If HSCTLR.M is set to 0, instruction accesses from stage 1 of the EL2 translation regime are to Normal memory, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable. `1` Instruction caches enabled at EL2. If HSCTLR.M is set to 0, instruction accesses from stage 1 of the EL2 translation regime are to Normal memory, Outer Shareable, Inner Write-Through, Outer Write-Through. When this bit is 0, all EL2 Normal memory instruction accesses are Non-cacheable.
[11]	-	Reserved, RES1.
[10:9]	-	Reserved, RES0.
[8]	SED	SETEND Disable: `0` The SETEND instruction is available. `1` The SETEND instruction is unallocated. The reset value is UNKNOWN.
[7]	ITD	Reserved, RES0. IT Disable. Disables some uses of IT instructions at PL1 and PL0.
[6]	-	Reserved, RES0.
[5]	CP15BEN	CP15 barrier enable: `0` CP15 barrier operations disabled. Their encodings are UNDEFINED. `1` CP15 barrier operations enabled. The reset value is UNKNOWN.
[4:3]	-	Reserved, RES1.
[2]	C	Cache enable. This is an enable bit for data and unified caches at EL2: `0` Data and unified caches disabled at EL2. `1` Data and unified caches enabled at EL2. When this bit is 0, all EL2 Normal memory data accesses and all accesses to the EL2 translation tables are Non-cacheable. Resets to UNKNOWN.

Table 4-222 HSCTLR bit assignments (continued)

Bits	Name	Function
[1]	A	Alignment check enable. This is the enable bit for Alignment fault checking: `0` Alignment fault checking disabled. `1` Alignment fault checking enabled. When this bit is 1, all instructions that load or store one or more registers, other than load/store exclusive and load-acquire/store-release, have an alignment check that the address being accessed is aligned to the size of the data element(s) being accessed. If this check fails it causes an Alignment fault, that is taken as a Data Abort exception. Load/store exclusive and load-acquire/store-release instructions have this alignment check regardless of the value of the A bit. Resets to UNKNOWN.
[0]	M	MMU enable. This is a global enable bit for the EL2 stage 1 MMU: `0` EL2 stage 1 MMU disabled. `1` EL2 stage 1 MMU enabled. Resets to UNKNOWN.

To access the HSCTLR:

```
MRC p15,4,<Rt>,c1,c0,0 ; Read HSCTLR into Rt
MCR p15,4,<Rt>,c1,c0,0 ; Write Rt to HSCTLR
```

4.5.38 Hyp Configuration Register

The HCR characteristics are:

Purpose Provides configuration controls for virtualization, including defining whether various Non-secure operations are trapped to Hyp mode.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations HCR is architecturally mapped to AArch64 register HCR_EL2[31:0]. See *4.3.44 Hypervisor Configuration Register* on page 4-131.

Attributes HCR is a 32-bit register.

The following figure shows the HCR bit assignments.

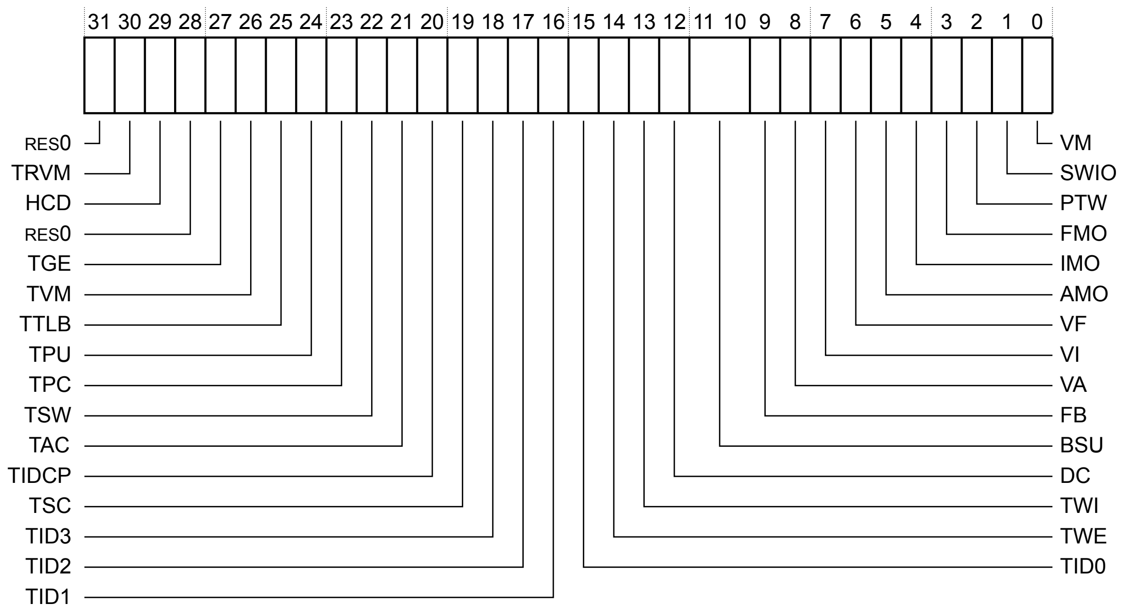

Figure 4-110 HCR bit assignments

The following table shows the HCR bit assignments.

Table 4-223 HCR bit assignments

Bits	Name	Function
[31]	-	Reserved, RES0.
[30]	TRVM	Trap Read of Virtual Memory controls. When 1, this causes Reads to the EL1 virtual memory control registers from EL1 to be trapped to EL2. This covers the following registers: SCTLR, TTBR0, TTBR1, TTBCR, DACR, DFSR, IFSR, DFAR, IFAR, ADFSR, AIFSR, PRRR/MAIR0, NMRR/MAIR1, AMAIR0, AMAIR1, and CONTEXTIDR. The reset value is 0.
[29]	HCD	Hyp Call Disable. The HCD value is: `0` HVC is enabled at EL1 or EL2. `1` HVC is UNDEFINED at all exception levels. The reset value is 0.
[28]	-	Reserved, RES0.

Table 4-223 HCR bit assignments (continued)

Bits	Name	Function
[27]	TGE	Trap 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.M bit is treated as 0 regardless of its actual state, other than for the purpose of reading the bit. • The HCR.FMO, IMO, and AMO bits are treated as 1 regardless of their actual state, other than for the purpose of 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. Additionally, if HCR.TGE is 1, the HDCR.{TDRA,TDOSA,TDA} bits are ignored and the processor behaves as if they are set to 1, other than for the value read back from HDCR. The reset value is 0.
[26]	TVM	Trap Virtual Memory controls. When 1, this causes Writes to the EL1 virtual memory control registers from EL1 to be trapped to EL2. This covers the following registers: SCTLR, TTBR0, TTBR1, TTBCR, DACR, DFSR, IFSR, DFAR, IFAR, ADFSR, AIFSR, PRRR/MAIR0, NMRR/MAIR1, AMAIR0, AMAIR1, and CONTEXTIDR. The reset value is 0.
[25]	TTLB	Trap TLB maintenance instructions. When `1`, this causes TLB maintenance instructions executed from EL1 that are not UNDEFINED to be trapped to EL2. This covers the following instructions: `TLBIALLIS`, `TLBIMVAIS`, `TLBIASIDIS`, `TLBIMVAAIS`, `TLBIALL`, `TLBIMVA`, `TLBIASID`, `TLBIMVAA`, `TLBIMVALIS`, `TLBIMVAALIS`, `TLBIMVAL`, and `TLBIMVAAL`. The reset value is 0.
[24]	TPU	Trap Cache maintenance instructions to Point of Unification. When 1, this causes Cache maintenance instructions to the point of unification executed from EL1 or EL0 that are not UNDEFINED to be trapped to EL2. This covers the following instructions: `ICIMVAU`, `ICIALLU`, `ICIALLUIS`, and `DCCMVAU`. The reset value is 0.
[23]	TPC	Trap Data/Unified Cache maintenance operations to Point of Coherency. When 1, this causes Data or Unified Cache maintenance instructions by address to the point of coherency executed from EL1 or EL0 that are not UNDEFINED to be trapped to EL2. This covers the following instructions: `DCIMVAC`, `DCCIMVAC`, and `DCCMVAC`. The reset value is 0.
[22]	TSW	Trap Data/Unified Cache maintenance operations by Set/Way. When 1, this causes Data or Unified Cache maintenance instructions by set/way executed from EL1 that are not UNDEFINED to be trapped to EL2. This covers the following instructions: `DCISW`, `DCCSW`, and `DCCISW`. The reset value is 0.
[21]	TAC	Trap ACTLR accesses. When this bit is set to 1, any valid Non-secure access to the ACTLR is trapped to Hyp mode. The reset value is 0.

Table 4-223 HCR bit assignments (continued)

Bits	Name	Function
[20]	TIDCP	Trap Implementation Dependent functionality. When 1, this causes accesses to all CP15 MCR and MRC instructions executed from EL1, to be trapped to EL2 as follows: • CRn is 9, Opcode1 is 0 to 7, CRm is c0, c1, c2, c5, c6, c7, c8, opcode2 is 0 to 7. • CRn is 10, Opcode1 is 0 to 7, CRm is c0, c1, c4, c8, opcode2 is 0 to 7. • CRn is 11, Opcode1 is 0 to 7, CRm is c0 to c8, or c15, opcode2 is 0 to 7. Accesses from EL0 are UNDEFINED. Resets to 0.
[19]	TSC	Trap SMC instruction. When this bit is set to 1, any attempt from a Non-secure EL1 state to execute an SMC instruction, that passes its condition check if it is conditional, is trapped to Hyp mode. The reset value is 0.
[18]	TID3	Trap ID Group 3. When 1, this causes reads to the following registers executed from EL1 to be trapped to EL2: ID_PFR0, ID_PFR1, ID_PFR2, ID_DFR0, ID_AFR0, ID_MMFR0, ID_MMFR1, ID_MMFR2, ID_MMFR3, ID_ISAR0, ID_ISAR1, ID_ISAR2, ID_ISAR3, ID_ISAR4, ID_ISAR5, MVFR0, MVFR1, and MVFR2. Also MRC instructions to any of the following encodings: • CP15, OPC1 is 0, CRn is 0, CRm is c3, c4, c5, c6, or c7, and Opc2 is 0 or 1. • CP15, Opc1 is 0, CRn is 0, CRm is c3, and Opc2 is 2. • CP15, Opc1 is 0, CRn is 0, CRm is 5, and Opc2 is 4 or 5. The reset value is 0.
[17]	TID2	Trap ID Group 2. When 1, this causes reads (or writes to CSSELR) to the following registers executed from EL1 or EL0 if not UNDEFINED to be trapped to EL2: CTR, CCSIDR, CLIDR, and CSSELR. The reset value is 0.
[16]	TID1	Trap ID Group 1. When 1, this causes reads to the following registers executed from EL1 to be trapped to EL2: TCMTR, TLBTR, AIDR, and REVIDR. The reset value is 0.
[15]	TID0	Trap ID Group 0. When 1, this causes reads to the following registers executed from EL1 or EL0 if not UNDEFINED to be trapped to EL2: FPSID and JIDR. The reset value is 0.
[14]	TWE	Trap WFE. When 1, this causes the `WFE` instruction executed from EL1 or EL0 to be trapped to EL2 if the instruction would otherwise cause suspension of execution. For example, if the event register is not set: The reset value is 0.
[13]	TWI	Trap WFI. When 1, this causes the `WFI` instruction executed from EL1 or EL0 to be trapped to EL2 if the instruction would otherwise cause suspension of execution. For example, if there is not a pending WFI wake-up event: The reset value is 0.

Table 4-223 HCR bit assignments (continued)

Bits	Name	Function
[12]	DC	Default cacheable. When this bit is set to 1, and the Non-secure EL1 and EL0 stage 1 MMU is disabled, the memory type and attributes determined by the stage 1 translation is Normal, Non-shareable, Inner Write-Back Write-Allocate, Outer Write-Back Write-Allocate. The reset value is 0.
[11:10]	BSU	Barrier Shareability upgrade. The value in this field determines the minimum shareability domain that is applied to any barrier executed from EL1 or EL0. The possible values are: `0b00` No effect. `0b01` Inner Shareable. `0b10` Outer Shareable. `0b11` Full System. The reset value is 0.
[9]	FB	Force Broadcast. When 1, this causes the following instructions to be broadcast within the Inner Shareable domain when executed from Non-secure EL1: `TLBIALL`, `TLBIMVA`, `TLBIASID`, `TLBIMVAA`, `BPIALL`, and `ICIALLU`. The reset value is 0.
[8]	VA	Virtual Asynchronous Abort exception. When the AMO bit is set to 1, setting this bit signals a virtual Asynchronous Abort exception to the Guest OS, when the processor is executing in Non-secure state at EL0 or EL1. The Guest OS cannot distinguish the virtual exception from the corresponding physical exception. The reset value is 0.
[7]	VI	Virtual IRQ exception. When the IMO bit is set to 1, setting this bit signals a virtual IRQ exception to the Guest OS, when the processor is executing in Non-secure state at EL0 or EL1. The Guest OS cannot distinguish the virtual exception from the corresponding physical exception. The reset value is 0.
[6]	VF	Virtual FIQ exception. When the FMO bit is set to 1, setting this bit signals a virtual FIQ exception to the Guest OS, when the processor is executing in Non-secure state at EL0 or EL1. The Guest OS cannot distinguish the virtual exception from the corresponding physical exception. The reset value is 0.
[5]	AMO	Asynchronous Abort Mask Override. When this is set to 1, it overrides the effect of CPSR.A, and enables virtual exception signaling by the VA bit. The reset value is 0.
[4]	IMO	IRQ Mask Override. When this is set to 1, it overrides the effect of CPSR.I, and enables virtual exception signaling by the VI bit. The reset value is 0.
[3]	FMO	FIQ Mask Override. When this is set to 1, it overrides the effect of CPSR.F, and enables virtual exception signaling by the VF bit. The reset value is 0.

Table 4-223 HCR bit assignments (continued)

Bits	Name	Function
[2]	PTW	Protected Table Walk. When 1, if the stage 2 translation of a translation table access made as part of a stage 1 translation table walk at EL0 or EL1 maps that translation table access to Device memory, the access is faulted as a stage 2 Permission fault. The reset value is 0.
[1]	SWIO	Set/Way Invalidation Override. When 1, this causes EL1 execution of the data cache invalidate by set/way instruction to be treated as data cache clean and invalidate by set/way. `DCISW` is executed as `DCCISW`. This bit is RES1.
[0]	VM	Second stage of Translation enable. When 1, this enables the second stage of translation for execution in EL1 and EL0. The reset value is 0.

To access the HCR:

```
MRC p15, 4, <Rt>, c1, c1, 0; Read Hyp Configuration Register
MCR p15, 4, <Rt>, c1, c1, 0; Write Hyp Configuration Register
```

4.5.39 Hyp Configuration Register 2

The HCR2 characteristics are:

Purpose Provides additional configuration controls for virtualization.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations HCR2 is architecturally mapped to AArch64 register HCR_EL2[63:32].

This register is accessible only at EL2 or EL3.

Attributes HCR2 is a 32-bit register.

The following figure shows the HCR2 bit assignments.

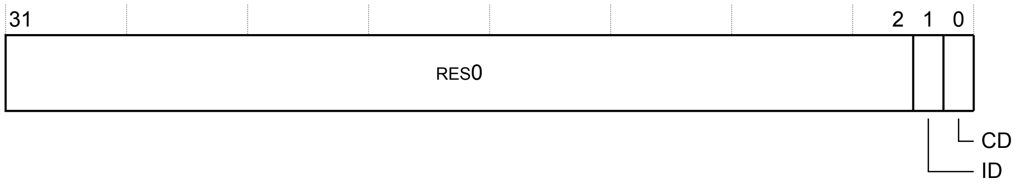

Figure 4-111 HCR2 bit assignments

The following table shows the HCR2 bit assignments.

Table 4-224 HCR2 bit assignments

Bits	Name	Function
[31:2]	-	Reserved, RES0.
[1]	ID	Stage 2 Instruction cache disable. When HCR.VM is 1, this forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable for the EL1/EL0 translation regime. The possible values are: `0` No effect on the stage 2 of the EL1/EL0 translation regime for instruction accesses. `1` Forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable for the EL0/EL1 translation regime. The reset value is 0.
[0]	CD	Stage 2 Data cache disable. When HCR.VM is 1, this forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable for the EL1/EL0 translation regime. The possible values are: `0` No effect on the stage 2 of the EL1/EL0 translation regime for data accesses and translation table walks. `1` Forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable for the EL0/EL1 translation regime. The reset value is 0.

To access the HCR2:

```
MRC p15,4,<Rt>,c1,c1,4 ; Read HCR2 into Rt
MCR p15,4,<Rt>,c1,c1,4 ; Write Rt to HCR2
```

4.5.40 Hyp Debug Control Register

The HDCR characteristics are:

Purpose Controls the trapping to Hyp mode of Non-secure accesses, at EL1 or lower, to functions provided by the debug and trace architectures and the Performance Monitor.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations
- HDCR is architecturally mapped to AArch64 register MDCR_EL2. See *4.3.45 Hyp Debug Control Register* on page 4-136.
- This register is accessible only at EL2 or EL3.

Attributes HDCR is a 32-bit register.

The following figure shows the HDCR bit assignments.

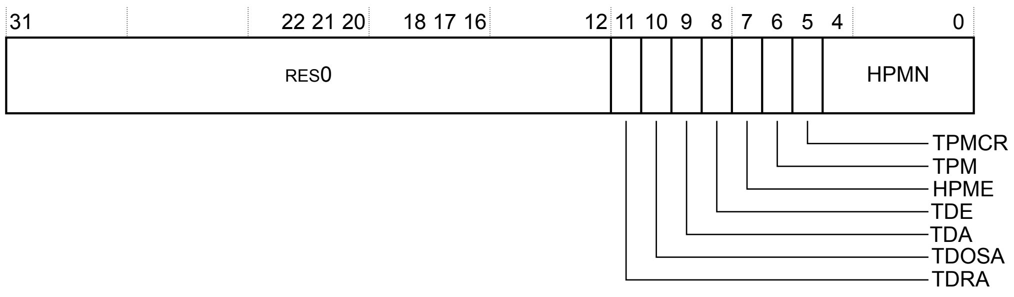

Figure 4-112 HDCR bit assignments

The following table shows the HDCR bit assignments.

Table 4-225 HDCR bit assignments

Bits	Name	Function
[31:12]	-	Reserved, RES0.
[11]	TDRA	Trap debug ROM address register access. `0` Has no effect on accesses to debug ROM address registers from EL1 and EL0. `1` Trap valid Non-secure EL1 and EL0 access to debug ROM address registers to Hyp mode. When this bit is set to 1, any valid Non-secure access to the following registers is trapped to Hyp mode: • DBGDRAR. • DBGDSAR. If HCR.TGE is 1 or HDCR.TDE is 1, then this bit is ignored and treated as though it is 1 other than for the value read back from HDCR. On Warm reset, the field resets to 0.
[10]	TDOSA	Trap Debug OS-related register access: `0` Has no effect on accesses to CP14 Debug registers. `1` Trap valid Non-secure accesses to CP14 OS-related Debug registers to Hyp mode. When this bit is set to 1, any valid Non-secure CP14 access to the following OS-related Debug registers is trapped to Hyp mode: • DBGOSLSR. • DBGOSLAR. • DBGOSDLR. • DBGPRCR. If HCR.TGE is 1 or HDCR.TDE is 1, then this bit is ignored and treated as though it is 1 other than for the value read back from HDCR. On Warm reset, the field resets to 0.

Table 4-225 HDCR bit assignments (continued)

Bits	Name	Function
[9]	TDA	Trap Debug Access: `0` Has no effect on accesses to CP14 Debug registers. `1` Trap valid Non-secure accesses to CP14 Debug registers to Hyp mode. When this bit is set to 1, any valid access to the CP14 Debug registers, other than the registers trapped by the TDRA and TDOSA bits, is trapped to Hyp mode. If HCR.TGE is 1 or HDCR.TDE is1, then this bit is ignored and treated as though it is 1 other than for the value read back from HDCR. On Warm reset, the field resets to 0.
[8]	TDE	Trap Debug Exceptions: `0` Has no effect on Debug exceptions. `1` Route Non-secure Debug exceptions to Hyp mode. When this bit is set to 1, any Debug exception taken in Non-secure state is trapped to Hyp mode. If HCR.TGE is 1, then this bit is ignored and treated as though it is 1 other than for the value read back from HDCR.This bit resets to 0.
[7]	HPME	Hypervisor Performance Monitor Enable: `0` Hyp mode performance monitor counters disabled. `1` Hyp mode performance monitor counters enabled. When this bit is set to 1, access to the performance monitors that are reserved for use from Hyp mode is enabled. For more information, see the description of the HPMN field. The reset value is 0.
[6]	TPM	Trap Performance Monitor accesses: `0` Has no effect on performance monitor accesses. `1` Trap valid Non-secure performance monitor accesses to Hyp mode. When this bit is set to 1, any valid Non-secure access to the Performance Monitor registers is trapped to Hyp mode. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information. The reset value is 0.

Table 4-225 HDCR bit assignments (continued)

Bits	Name	Function
[5]	TPMCR	Trap Performance Monitor Control Register accesses: `0` Has no effect on PMCR accesses. `1` Trap valid Non-secure PMCR accesses to Hyp mode. When this bit is set to 1, any valid Non-secure access to the PMCR is trapped to Hyp mode. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information. The reset value is 0.
[4:0]	HPMN	Hyp Performance Monitor count. Defines the number of Performance Monitors counters that are accessible from Non-secure EL1 and EL0 modes if unprivileged access is enabled. In Non-secure state, HPMN divides the Performance Monitors counters as follows: If PMnEVCNTR is performance monitor counter *n* then, in Non-secure state: • If *n* is in the range 0 ≤ *n* < HPMN, the counter is accessible from EL1 and EL2, and from EL0 if unprivileged access to the counters is enabled. • If *n* is in the range HPMN ≤ *n* <PMCR.N, the counter is accessible only from EL2. The HPME bit enables access to the counters in this range. If this field 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 processor must return a CONSTRAINED UNPREDICTABLE value being 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. This field resets to `0x6`.

To access the HDCR:

```
MRC p15,4,<Rt>,c1,c1,1 ; Read HDCR into Rt
MCR p15,4,<Rt>,c1,c1,1 ; Write Rt to HDCR
```

4.5.41 Hyp Architectural Feature Trap Register

The HCPTR characteristics are:

Purpose Controls trapping to Hyp mode of Non-secure access, at EL1 or lower, to coprocessors other than CP14 and CP15 and to floating-point and Advanced SIMD functionality. Also controls access from Hyp mode to this functionality.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

If a bit in the NSACR prohibits a Non-secure access, then the corresponding bit in the HCPTR behaves as RAO/WI for Non-secure accesses. See the bit description for TASE.

Configurations HCPTR is architecturally mapped to AArch64 register CPTR_EL2.

Attributes HCPTR is a 32-bit register.

The following figure shows the HCPTR bit assignments.

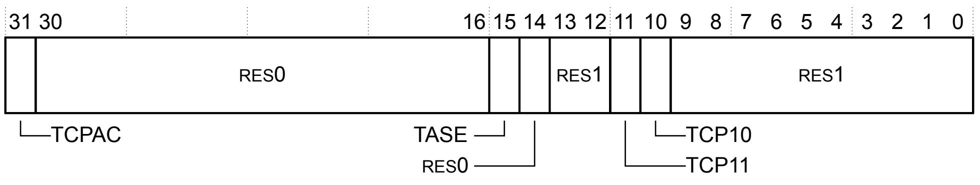

Figure 4-113 HCPTR bit assignments

The following table shows the HCPTR bit assignments.

Table 4-226 HCPTR bit assignments

Bits	Name	Function
[31]	TCPAC	Trap CPACR accesses. The possible values of this bit are: `0` Has no effect on CPACR accesses. `1` Trap valid Non-secure EL1 CPACR accesses to Hyp mode. When this bit is set to 1, any valid Non-secure EL1 access to the CPACR is trapped to Hyp mode. Resets to 0.
[30:16]	-	Reserved, RES0.
[15]	TASE	Trap Advanced SIMD use: `0` If the NSACR settings permit Non-secure use of the Advanced SIMD functionality then Hyp mode can access that functionality, regardless of any settings in the CPACR. This bit value has no effect on possible use of the Advanced SIMD functionality from Non-secure EL1 and EL0 modes. `1` Trap valid Non-secure accesses to Advanced SIMD functionality to Hyp mode. If NSACR.NSASEDIS is set to 1, then on Non-secure accesses to the HCPTR, the TASE bit behaves as RAO/WI. Resets to 0.
[14]	-	Reserved, RES0.
[13:12]	-	Reserved, RES1.

Table 4-226 HCPTR bit assignments (continued)

Bits	Name	Function
[11]	TCP11[bp]	Trap CP11. The possible values are: `0` If NSACR.cp11 is set to 1, then Hyp mode can access CP11, regardless of the value of CPACR.cp11. This bit value has no effect on possible use of CP11 from Non-secure EL1 and EL0 modes. `1` Trap valid Non-secure accesses to CP11 to Hyp mode. Any otherwise-valid access to CP11 from: • A Non-secure EL1 or EL0 state is trapped to Hyp mode. • Hyp mode generates an Undefined Instruction exception, taken in Hyp mode. Resets to 0.
[10]	TCP10[bp]	Trap CP10. The possible values are: `0` If NSACR.cp10 is set to 1, then Hyp mode can access CP10, regardless of the value of CPACR.cp10. This bit value has no effect on possible use of CP10 from Non-secure EL1 and EL0 modes. `1` Trap valid Non-secure accesses to CP10 to Hyp mode. Any otherwise-valid access to CP10 from: • A Non-secure EL1 or EL0 state is trapped to Hyp mode. • Hyp mode generates an Undefined Instruction exception, taken in Hyp mode. Resets to 0.
[9:0]	-	Reserved, RES1.

To access the HCPTR:

```
MRC p15,4,<Rt>,c1,c1,2 ; Read HCPTR into Rt
MCR p15,4,<Rt>,c1,c1,2 ; Write Rt to HCPTR
```

4.5.42 Translation Table Base Register 0

The TTBR0 characteristics are:

Purpose 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.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Used in conjunction with the TTBCR. When the 64-bit TTBR0 format is used, cacheability and shareability information is held in the TTBCR and not in TTBR0.

[bp] If the TCP11 and TCP10 fields are set to different values, the behavior is the same as if both fields were set to the value of TCP10 in all respects other than the value read back by explicitly reading TCP11.

Configurations	TTBR0 (NS) is architecturally mapped to AArch64 register TTBR0_EL1. See *4.3.52 Translation Table Base Register 0, EL1* on page 4-148. TTBR0 (S) is mapped to AArch64 register TTBR0_EL3. See *4.3.60 Translation Table Base Register 0, EL3* on page 4-161. If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.
Attributes	TTBR0 is: • A 32-bit register when TTBCR.EAE is 0. • A 64-bit register when TTBCR.EAE is 1.

There are different formats for this register. TTBCR.EAE determines which format of the register is used. This section describes:
- *TTBR0 format when using the Short-descriptor translation table format* on page 4-285.
- *TTBR0 format when using the Long-descriptor translation table format* on page 4-286.

TTBR0 format when using the Short-descriptor translation table format

The following figure shows the TTBR0 bit assignments when TTBCR.EAE is 0.

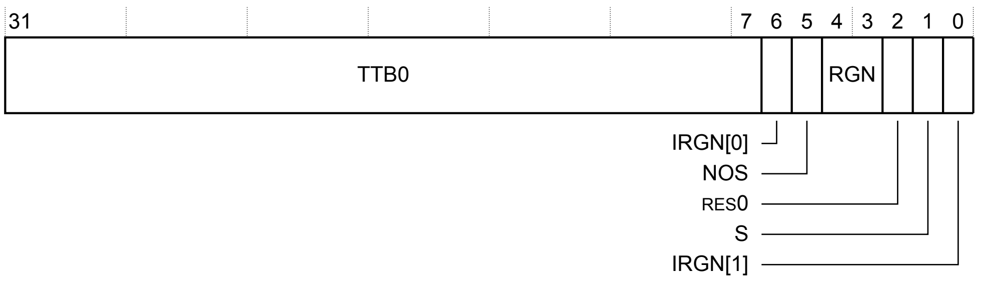

Figure 4-114 TTBR0 bit assignments, TTBCR.EAE is 0

The following table shows the TTBR0 bit assignments when TTBCR.EAE is 0.

Table 4-227 TTBR0 bit assignments, TTBCR.EAE is 0

Bits	Name	Function
[31:7]	TTB0	Translation table base 0 address, bits[31:x], where x is 14-(TTBCR.N). Bits [x-1:7] are RES0. The value of x determines the required alignment of the translation table, that must be aligned to 2^x bytes. If bits [x-1:7] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:7] are treated as if all the bits are zero. The value read back from those bits is the value written.
[6]	IRGN[0]	See bit[0] for a description of the IRGN field.
[5]	NOS	Not Outer Shareable bit. Indicates the Outer Shareable attribute for the memory associated with a translation table walk that has the Shareable attribute, indicated by TTBR0.S is 1. The possible values are: `0` Outer Shareable. `1` Inner Shareable. This bit is ignored when TTBR0.S is 0.

Table 4-227 TTBR0 bit assignments, TTBCR.EAE is 0 (continued)

Bits	Name	Function
[4:3]	RGN	Region bits. Indicates the Outer cacheability attributes for the memory associated with the translation table walks. The possible values are: `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.
[2]	-	Reserved, RES0.
[1]	S	Shareable bit. Indicates the Shareable attribute for the memory associated with the translation table walks. The possible values are: `0` Non-shareable. `1` Shareable.
[0]	IRGN[1]	Inner region bits. Indicates the Inner Cacheability attributes for the memory associated with the translation table walks. The possible values of IRGN[1:0] are: `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.

To access the TTBR0 when TTBCR.EAE is 0:

```
MRC p15,0,<Rt>,c2,c0,0 ; Read TTBR0 into Rt
MCR p15,0,<Rt>,c2,c0,0 ; Write Rt to TTBR0
```

TTBR0 format when using the Long-descriptor translation table format

The following figure shows the TTBR0 bit assignments when TTBCR.EAE is 1.

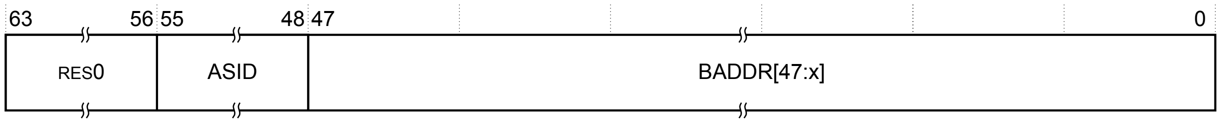

Figure 4-115 TTBR0 bit assignments, TTBCR.EAE is 1

The following table shows the TTBR0 bit assignments when TTBCR.EAE is 1.

Table 4-228 TTBR0 bit assignments, TTBCR.EAE is 1

Bits	Name	Function
[63:56]	-	Reserved, RES0.
[55:48]	ASID	An ASID for the translation table base address. The TTBCR.A1 field selects either TTBR0.ASID or TTBR1.ASID.
[47:0]	BADDR[47:x]	Translation table base address, bits[47:x]. Bits [x-1:0] are RES0. x is based on the value of TTBCR.T0SZ, and is calculated as follows: • If TTBCR.T0SZ is 0 or 1, x = 5 - TTBCR.T0SZ. • If TTBCR.T0SZ is greater than 1, x = 14 - TTBCR.T0SZ. The value of x determines the required alignment of the translation table, that must be aligned to 2x bytes. If bits [x-1:3] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:0] are treated as if all the bits are zero. The value read back from those bits is the value written.

To access the TTBR0 when TTBCR.EAE==1:

```
MRRC p15,0,<Rt>,<Rt2>,c2 ; Read 64-bit TTBR0 into Rt (low word) and Rt2 (high word)
MCRR p15,0,<Rt>,<Rt2>,c2 ; Write Rt (low word) and Rt2 (high word) to 64-bit TTBR0
```

4.5.43 Translation Table Base Register 1

The TTBR1 characteristics are:

Purpose 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 from modes other than Hyp mode.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Used in conjunction with the TTBCR. When the 64-bit TTBR1 format is used, cacheability and shareability information is held in the TTBCR and not in TTBR1. See *4.5.44 Translation Table Base Control Register* on page 4-289.

Configurations TTBR1 (NS) is architecturally mapped to AArch64 register TTBR0_EL1. See *4.3.53 Translation Table Base Register 1* on page 4-149.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

Attributes TTBR1 is:

- A 32-bit register when TTBCR.EAE is 0.
- A 64-bit register when TTBCR.EAE is 1.

There are two formats for this register. TTBCR.EAE determines which format of the register is used. This section describes:

- *TTBR1 format when using the Short-descriptor translation table format* on page 4-288.
- *TTBR1 format when using the Long-descriptor translation table format* on page 4-289.

TTBR1 format when using the Short-descriptor translation table format

The following figure shows the TTBR1 bit assignments when TTBCR.EAE is 0.

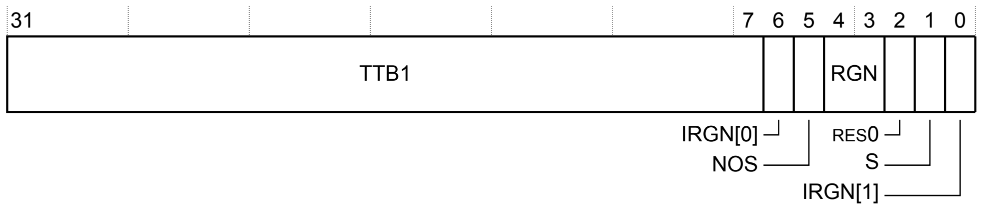

Figure 4-116 TTBR1 bit assignments, TTBCR.EAE is 0

The following table shows the TTBR1 bit assignments when TTBCR.EAE is 0.

Table 4-229 TTBR1 bit assignments, TTBCR.EAE is 0

Bits	Name	Function
[31:7]	TTB1	Translation table base 1 address, bits[31:x], where x is 14-(TTBCR.N). Bits [x-1:7] are RES0. The translation table must be aligned on a 16KByte boundary. If bits [x-1:7] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:7] are treated as if all the bits are zero. The value read back from those bits is the value written.
[6]	IRGN[0]	See bit[0] for description of the IRGN field.
[5]	NOS	Not Outer Shareable bit. Indicates the Outer Shareable attribute for the memory associated with a translation table walk that has the Shareable attribute, indicated by TTBR0.S is 1. The possible values are: `0` Outer Shareable. `1` Inner Shareable. This bit is ignored when TTBR0.S is 0.
[4:3]	RGN	Region bits. Indicates the Outer cacheability attributes for the memory associated with the translation table walks. The possible values are: `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.
[2]	-	Reserved, RES0.
[1]	S	Shareable bit. Indicates the Shareable attribute for the memory associated with the translation table walks. The possible values are: `0` Non-shareable. `1` Shareable.
[0]	IRGN[1]	Inner region bits. Indicates the Inner Cacheability attributes for the memory associated with the translation table walks. The possible values of IRGN[1:0] are: `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.

To access the TTBR1 when TTBCR.EAE is 0:

```
MRC p15, 0, <Rt>, c2, c0, 1 ; Read TTBR1 into Rt
MCR p15, 0, <Rt>, c2, c0, 1 ; Write Rt to TTBR1
```

TTBR1 format when using the Long-descriptor translation table format

The following figure shows the TTBR1 bit assignments when TTBCR.EAE is 1.

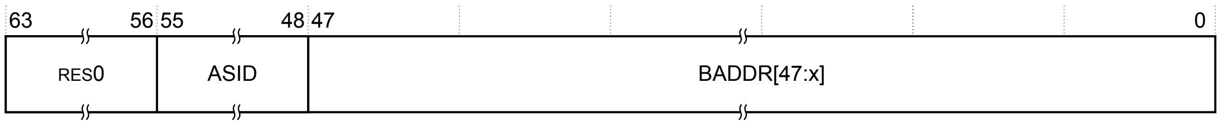

Figure 4-117 TTBR1 bit assignments, TTBCR.EAE is 1

The following table shows the TTBR1 bit assignments when TTBCR.EAE is 1.

Table 4-230 TTBR1 bit assignments, TTBCR.EAE is 1

Bits	Name	Function
[63:56]	-	Reserved, RES0.
[55:48]	ASID	An ASID for the translation table base address. The TTBCR.A1 field selects either TTBR0.ASID or TTBR1.ASID.
[47:0]	BADDR[47:x]	Translation table base address, bits[47:x]. Bits [x-1:0] are RES0. x is based on the value of TTBCR.T0SZ, and is calculated as follows: • If TTBCR.T0SZ is 0 or 1, x = 5 - TTBCR.T0SZ. • If TTBCR.T0SZ is greater than 1, x = 14 - TTBCR.T0SZ. The value of x determines the required alignment of the translation table, that must be aligned to 2x bytes. If bits [x-1:3] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:0] are treated as if all the bits are zero. The value read back from those bits is the value written.

To access the 64-bit TTBR1 when TTBCR.EAE = 1:

```
MRRC p15, 1, <Rt>, <Rt2>, c2 ; Read 64-bit TTBR1 into Rt (low word) and Rt2 (high word)
MCRR p15, 1, <Rt>, <Rt2>, c2 ; Write Rt (low word) and Rt2 (high word) to 64-bit TTBR1
```

4.5.44 Translation Table Base Control Register

The TTBCR characteristics are:

Purpose Determines which of the Translation Table Base Registers defines the base address for a translation table walk required for the stage 1 translation of a memory access from any mode other than Hyp mode. Also controls the translation table format and, when using the Long-descriptor translation table format, holds cacheability and shareability information.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

The processor does not use the implementation-defined bit, TTBCR[30], when using the Long-descriptor translation table format, so this bit is RES0.

Configurations TTBCR (NS) is architecturally mapped to AArch64 register TCR_EL1. See *4.3.56 Translation Control Register, EL1* on page 4-153.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

Attributes TTBCR is a 32-bit register.

There are two formats for this register. TTBCR.EAE determines which format of the register is used. This section describes:
- *TTBCR format when using the Short-descriptor translation table format* on page 4-290.
- *TTBCR format when using the Long-descriptor translation table format* on page 4-291.

TTBCR format when using the Short-descriptor translation table format

The following figure shows the TTBCR bit assignments when TTBCR.EAE is 0.

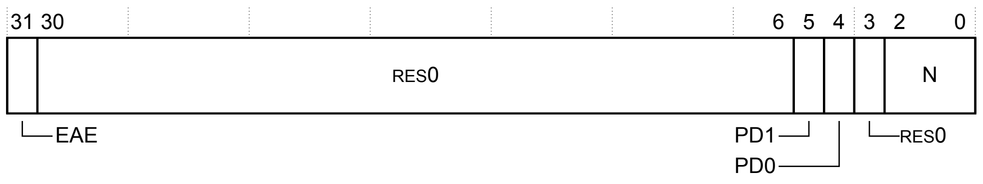

Figure 4-118 TTBCR bit assignments, TTBCR.EAE is 0

The following table shows the TTBCR bit assignments when TTBCR.EAE is 0.

Table 4-231 TTBCR bit assignments, TTBCR.EAE is 0

Bits	Name	Function
[31]	EAE	Extended Address Enable. `0` Use the 32-bit translation system, with the Short-descriptor translation table format.
[30:6]	-	Reserved, RES0.
[5]	PD1	Translation table walk disable for translations using TTBR1. This bit controls whether a translation table walk is performed on a TLB miss for an address that is translated using TTBR1. The possible values are: `0` Perform translation table walks using TTBR1. `1` A TLB miss on an address that is translated using TTBR1 generates a Translation fault. No translation table walk is performed.
[4]	PD0	Translation table walk disable for translations using TTBR0. This bit controls whether a translation table walk is performed on a TLB miss for an address that is translated using TTBR0. The possible values are: `0` Perform translation table walks using TTBR0. `1` A TLB miss on an address that is translated using TTBR0 generates a Translation fault. No translation table walk is performed.

Table 4-231 TTBCR bit assignments, TTBCR.EAE is 0 (continued)

Bits	Name	Function
[3]	-	Reserved, RES0.
[2:0]	N	Indicates the width of the base address held in TTBR0. In TTBR0, the base address field is bits[31:14-N]. The value of N also determines: • Whether TTBR0 or TTBR1 is used as the base address for translation table walks. • The size of the translation table pointed to by TTBR0. N can take any value from 0 to 7, that is, from 0b000 to 0b111. When N has its reset value of 0, the translation table base is compatible with Armv5 and Armv6. Resets to 0.

TTBCR format when using the Long-descriptor translation table format

The following figure shows the TTBCR bit assignments when TTBCR.EAE is 1.

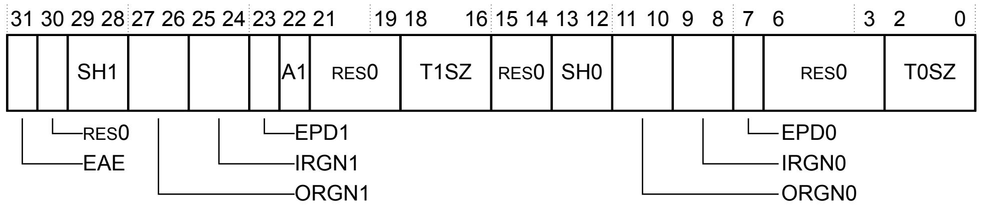

Figure 4-119 TTBCR bit assignments, TTBCR.EAE is 1

The following table shows the TTBCR bit assignments when TTBCR.EAE is 1.

Table 4-232 TTBCR bit assignments, TTBCR.EAE is 1

Bits	Name	Function
[31]	EAE	Extended Address Enable: `1` Use the 40-bit translation system, with the Long-descriptor translation table format.
[30]	-	Reserved, RES0.
[29:28]	SH1	Shareability attribute for memory associated with translation table walks using TTBR1: `0b00` Non-shareable. `0b10` Outer Shareable. `0b11` Inner Shareable. Other values are reserved. Resets to 0.
[27:26]	ORGN1	Outer cacheability attribute for memory associated with translation table walks using TTBR1: `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable. Resets to 0.

Table 4-232 TTBCR bit assignments, TTBCR.EAE is 1 (continued)

Bits	Name	Function
[25:24]	IRGN1	Inner cacheability attribute for memory associated with translation table walks using TTBR1: `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable. Resets to 0.
[23]	EPD1	Translation table walk disable for translations using TTBR1. This bit controls whether a translation table walk is performed on a TLB miss for an address that is translated using TTBR1: `0` Perform translation table walks using TTBR1. `1` A TLB miss on an address that is translated using TTBR1 generates a Translation fault. No translation table walk is performed.
[22]	A1	Selects whether TTBR0 or TTBR1 defines the ASID: `0` TTBR0.ASID defines the ASID. `1` TTBR1.ASID defines the ASID.
[21:19]	-	Reserved, RES0.
[18:16]	T1SZ	The size offset of the memory region addressed by TTBR1. The region size is $2^{32\text{-T1SZ}}$ bytes. Resets to 0.
[15:14]	-	Reserved, RES0.
[13:12]	SH0	Shareability attribute for memory associated with translation table walks using TTBR0: `0b00` Non-shareable. `0b10` Outer Shareable. `0b11` Inner Shareable. Other values are reserved. Resets to 0.
[11:10]	ORGN0	Outer cacheability attribute for memory associated with translation table walks using TTBR0: `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable. Resets to 0.

Table 4-232 TTBCR bit assignments, TTBCR.EAE is 1 (continued)

Bits	Name	Function
[9:8]	IRGN0	Inner cacheability attribute for memory associated with translation table walks using TTBR0: `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable. Resets to 0.
[7]	EPD0	Translation table walk disable for translations using TTBR0. This bit controls whether a translation table walk is performed on a TLB miss for an address that is translated using TTBR0: `0` Perform translation table walks using TTBR0. `1` A TLB miss on an address that is translated using TTBR0 generates a Translation fault. No translation table walk is performed.
[6:3]	-	Reserved, RES0.
[2:0]	T0SZ	The size offset of the memory region addressed by TTBR0. The region size is $2^{32\text{-T0SZ}}$ bytes. Resets to 0.

To access the TTBCR:

```
MRC p15,0,<Rt>,c2,c0,0 ; Read TTBCR into Rt
MCR p15,0,<Rt>,c2,c0,0 ; Write Rt to TTBCR
```

4.5.45 Hyp Translation Control Register

The HTCR characteristics are:

Purpose Controls translation table walks required for the stage 1 translation of memory accesses from Hyp mode, and holds cacheability and shareability information for the accesses.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations HTCR is architecturally mapped to AArch64 register TCR_EL2. See *4.3.57 Translation Control Register, EL2* on page 4-156.

Attributes HTCR is a 32-bit register.

The following figure shows the HTCR bit assignments.

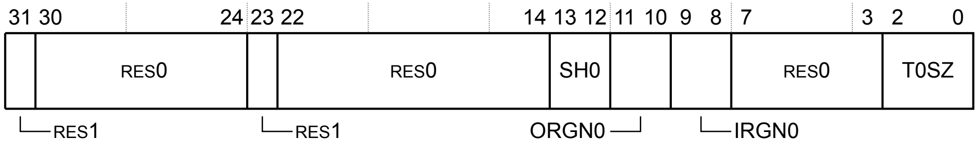

Figure 4-120 HTCR bit assignments

The following table shows the HTCR bit assignments.

Table 4-233 HTCR bit assignments

Bits	Name	Function
[31]	-	Reserved, RES1.
[30:24]	-	Reserved, RES0.
[23]	-	Reserved, RES1.
[22:14]	-	Reserved, RES0.
[13:12]	SH0	Shareability attribute for memory associated with translation table walks using TTBR0. The possible values are: `0b00` Non-shareable. `0b01` Reserved. `0b10` Outer shareable. `0b11` Inner shareable.
[11:10]	ORGN0	Outer cacheability attribute for memory associated with translation table walks using TTBR0. The possible values are: `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.
[9:8]	IRGN0	Inner cacheability attribute for memory associated with translation table walks using TTBR0. The possible values are: `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.
[7:3]	-	Reserved, RES0.
[2:0]	T0SZ	Size offset of the memory region addressed by TTBR0. The region size is $2^{(32\text{-TSIZE})}$ bytes.

The processor does not use the implementation-defined bit, HTCR[30], so this bit is RES0.

To access the HTCR:

```
MRC p15, 4, <Rt>, c2, c0, 2; Read HTCR into Rt
MCR p15, 4, <Rt>, c2, c0, 2; Write Rt to HTCR
```

4.5.46 Virtualization Translation Control Register

The VTCR characteristics are:

Purpose Controls the translation table walks required for the stage 2 translation of memory accesses from Non-secure modes other than Hyp mode, and holds cacheability and shareability information for the accesses.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Used in conjunction with VTTBR, that defines the translation table base address for the translations.

Configurations

VTCR is architecturally mapped to AArch64 register VTCR_EL2. See *4.3.58 Virtualization Translation Control Register, EL2* on page 4-158.

This register is accessible only at EL2 or EL3.

Attributes

VTCR is a 32-bit register.

The following figure shows the VTCR bit assignments.

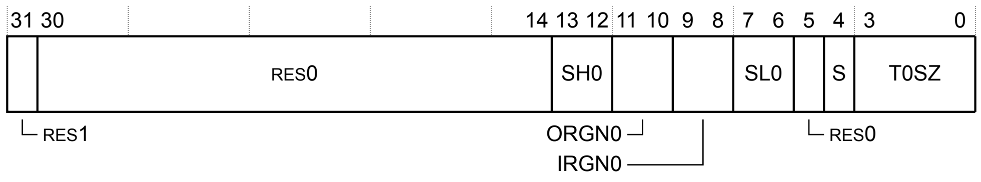

Figure 4-121 VTCR bit assignments

The following table shows the VTCR bit assignments.

Table 4-234 VTCR bit assignments

Bits	Name	Function
[31]	-	Reserved, RES1.
[30:14]	-	Reserved, RES0.
[13:12]	SH0	Shareability attribute for memory associated with translation table walks using TTBR0. `0b00` Non-shareable. `0b01` Reserved. `0b10` Outer Shareable. `0b11` Inner Shareable.
[11:10]	ORGN0	Outer cacheability attribute for memory associated with translation table walks using TTBR0. `0b00` Normal memory, Outer Non-cacheable. `0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Outer Write-Through Cacheable. `0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.
[9:8]	IRGN0	Inner cacheability attribute for memory associated with translation table walks using TTBR0. `0b00` Normal memory, Inner Non-cacheable. `0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable. `0b10` Normal memory, Inner Write-Through Cacheable. `0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.

Table 4-234 VTCR bit assignments (continued)

Bits	Name	Function
[7:6]	SL0	Starting level for translation table walks using VTTBR: `0b00` Start at second level. `0b01` Start at first level.
[5]	-	Reserved, RES0.
[4]	S	Sign extension bit. This bit must be programmed to the value of T0SZ[3]. If it is not, then the stage 2 T0SZ value is treated as an UNKNOWN value within the legal range that can be programmed.
[3:0]	T0SZ	The size offset of the memory region addressed by TTBR0. The region size is $2^{32\text{-T0SZ}}$ bytes.

To access the VTCR:

```
MRC p15, 4, <Rt>, c2, c1, 2; Read VTCR into Rt
MCR p15, 4, <Rt>, c2, c1, 2; Write Rt to VTCR
```

4.5.47 Domain Access Control Register

The DACR characteristics are:

Purpose Defines the access permission for each of the sixteen memory domains.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Configurations DACR (NS) is architecturally mapped to AArch64 register DACR32_EL2. See *4.3.59 Domain Access Control Register* on page 4-160.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

DACR has no function when TTBCR.EAE is set to 1, to select the Long-descriptor translation table format.

Attributes DACR is a 32-bit register.

The following figure shows the DACR bit assignments.

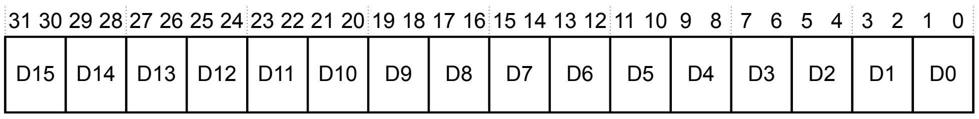

Figure 4-122 DACR bit assignments

The following table shows the DACR bit assignments.

Table 4-235 DACR bit assignments

Bits	Name	Function
[31:0]	D<*n*>, bits [2*n*+1:2*n*], for *n* = 0 to 15	Domain *n* access permission, where *n* = 0 to 15. Permitted values are: `0b00` No access. Any access to the domain generates a Domain fault. `0b01` Client. Accesses are checked against the permission bits in the translation tables. `0b11` Manager. Accesses are not checked against the permission bits in the translation tables. The value `0b10` is reserved.

To access the DACR:

```
MRC p15, 0, <Rt>, c3, c0, 0 ; Read DACR into Rt
MCR p15, 0, <Rt>, c3, c0, 0 ; Write Rt to DACR
```

4.5.48 Hyp System Trap Register

The HSTR characteristics are:

Purpose Controls trapping to Hyp mode of Non-secure accesses, at EL1 or lower, of use of T32EE, or the CP15 primary registers, {c0-c3,c5-c13,c15}.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations HSTR is architecturally mapped to AArch64 register HSTR_EL2.

This register is accessible only at EL2 or EL3.

Attributes HSTR is a 32-bit register.

The following figure shows the HSTR bit assignments.

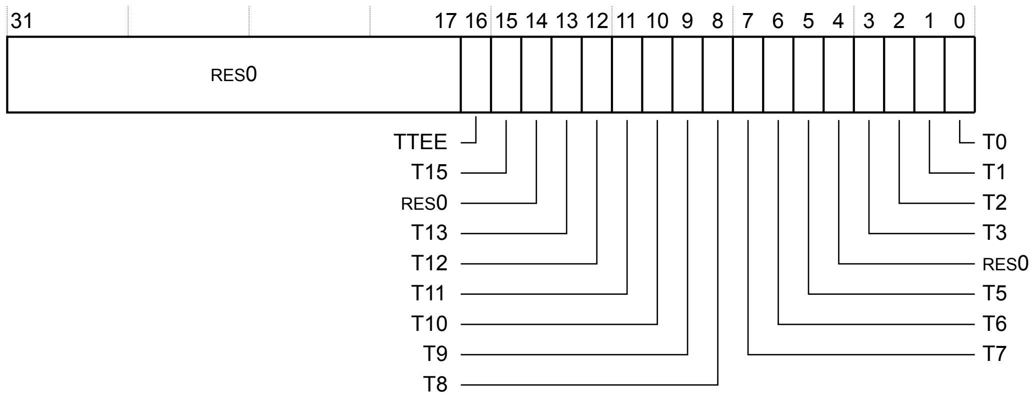

Figure 4-123 HSTR bit assignments

The following table shows the HSTR bit assignments.

Table 4-236 HSTR bit assignments

Bits	Name	Function
[31:17]	-	Reserved, RES0.
[16]	TTEE	Trap T32EE. This value is: `0` T32EE is not supported.
[15]	T15	Trap coprocessor primary register CRn = 15. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 15 to Hyp mode. The reset value is 0.
[14]	-	Reserved, RES0.
[13]	T13	Trap coprocessor primary register CRn = 13. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 13 to Hyp mode. The reset value is 0.
[12]	T12	Trap coprocessor primary register CRn = 12. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 12 to Hyp mode. The reset value is 0.
[11]	T11	Trap coprocessor primary register CRn = 11. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 11 to Hyp mode. The reset value is 0.
[10]	T10	Trap coprocessor primary register CRn = 10. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 10 to Hyp mode. The reset value is 0.
[9]	T9	Trap coprocessor primary register CRn = 9. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 9 to Hyp mode. The reset value is 0.
[8]	T8	Trap coprocessor primary register CRn = 8. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 8 to Hyp mode. The reset value is 0.

Table 4-236 HSTR bit assignments (continued)

Bits	Name	Function
[7]	T7	Trap coprocessor primary register CRn = 7. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 7 to Hyp mode. The reset value is 0.
[6]	T6	Trap coprocessor primary register CRn = 6. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 6 to Hyp mode. The reset value is 0.
[5]	T5	Trap coprocessor primary register CRn = 5. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 5 to Hyp mode. The reset value is 0.
[4]	-	Reserved, RES0.
[3]	T3	Trap coprocessor primary register CRn = 3. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 3 to Hyp mode. The reset value is 0.
[2]	T2	Trap coprocessor primary register CRn = 2. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 2 to Hyp mode. The reset value is 0.
[1]	T1	Trap coprocessor primary register CRn = 1. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 1 to Hyp mode. The reset value is 0.
[0]	T0	Trap coprocessor primary register CRn = 0. The possible values are: `0` Has no effect on Non-secure accesses to CP15 registers. `1` Trap valid Non-secure accesses to coprocessor primary register CRn = 0 to Hyp mode. The reset value is 0.

To access the HSTR:

```
MRC p15, 4, <Rt>, c1, c1, 3 ; Read HSTR into Rt
MCR p15, 4, <Rt>, c1, c1, 3 ; Write Rt to HSTR
```

4.5.49 Hyp Auxiliary Configuration Register

The processor does not implement HACR, so this register is always RES0.

4.5.50 Data Fault Status Register

The DFSR characteristics are:

Purpose Holds status information about the last data fault.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Configurations DFSR (NS) is architecturally mapped to AArch64 register ESR_EL1.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

There are two formats for this register. The current translation table format determines which format of the register is used.

Attributes DFSR is a 32-bit register.

This section describes:
- *DFSR when using the Short-descriptor translation table format* on page 4-300.
- *DFSR when using the Long-descriptor translation table format* on page 4-302.

DFSR when using the Short-descriptor translation table format

The following figure shows the DFSR bit assignments when using the Short-descriptor translation table format.

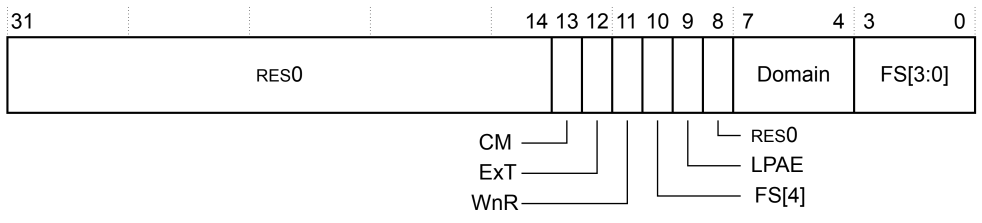

Figure 4-124 DFSR bit assignments for Short-descriptor translation table format

The following table shows the DFSR bit assignments when using the Short-descriptor translation table format.

Table 4-237 DFSR bit assignments for Short-descriptor translation table format

Bits	Name	Function
[31:14]	-	Reserved, RES0.
[13]	CM	Cache maintenance fault. For synchronous faults, this bit indicates whether a cache maintenance operation generated the fault: `0` Abort not caused by a cache maintenance operation. `1` Abort caused by a cache maintenance operation.

Table 4-237 DFSR bit assignments for Short-descriptor translation table format (continued)

Bits	Name	Function
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: `0` External abort marked as DECERR. `1` External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11]	WnR	Write not Read bit. This field indicates whether the abort was caused by a write or a read access: `0` Abort caused by a read access. `1` Abort caused by a write access. For faults on CP15 cache maintenance operations, including the VA to PA translation operations, this bit always returns a value of 1.
[10]	FS[4]	Part of the Fault Status field. See bits [3:0] in this table.
[9]	LPAE	On taking a Data Abort exception, this bit is set as follows: `0` Using the Short-descriptor translation table formats. `1` Using the Long-descriptor translation table formats. Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.
[8]	-	Reserved, RES0.

Table 4-237 DFSR bit assignments for Short-descriptor translation table format (continued)

Bits	Name	Function
[7:4]	Domain	Specifies which of the 16 domains, D15-D0, was being accessed when a data fault occurred. For permission faults that generate Data Abort exception, this field is UNKNOWN. Armv8 deprecates any use of the domain field in the DFSR.
[3:0]	FS[3:0]	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved: `0b00001` Alignment fault. `0b00010` Debug event. `0b00011` Access flag fault, section. `0b00100` Instruction cache maintenance fault. `0b00101` Translation fault, section. `0b00110` Access flag fault, page. `0b00111` Translation fault, page. `0b01000` Synchronous external abort, non-translation. `0b01001` Domain fault, section. `0b01011` Domain fault, page. `0b01100` Synchronous external abort on translation table walk, first level. `0b01101` Permission fault, section. `0b01110` Synchronous external abort on translation table walk, second level. `0b01111` Permission fault, second level. `0b10000` TLB conflict abort. `0b10101` `LDREX` or `STREX` abort. `0b10110` Asynchronous external abort. `0b11000` Asynchronous parity error on memory access. `0b11001` Synchronous parity error on memory access. `0b11100` Synchronous parity error on translation table walk, first level. `0b11110` Synchronous parity error on translation table walk, second level.

DFSR when using the Long-descriptor translation table format

The following figure shows the DFSR bit assignments when using the Long-descriptor translation table format.

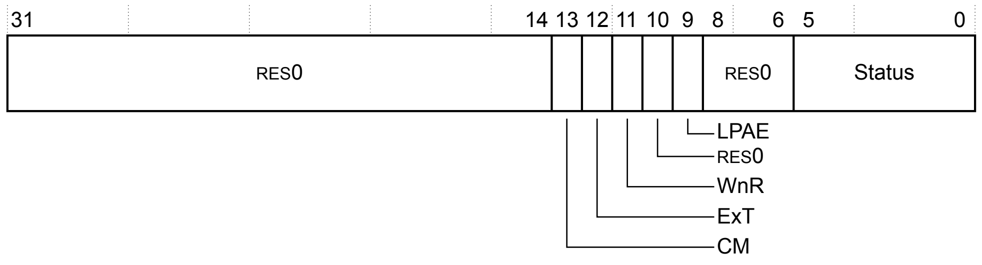

Figure 4-125 DFSR bit assignments for Long-descriptor translation table format

The following table shows the DFSR bit assignments when using the Long-descriptor translation table format.

Table 4-238 DFSR bit assignments for Long-descriptor translation table format

Bits	Name	Function
[31:14]	-	Reserved, RES0.
[13]	CM	Cache maintenance fault. For synchronous faults, this bit indicates whether a cache maintenance operation generated the fault: `0` Abort not caused by a cache maintenance operation. `1` Abort caused by a cache maintenance operation.
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: `0` External abort marked as DECERR. `1` External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11]	WnR	Write not Read bit. This field indicates whether the abort was caused by a write or a read access: `0` Abort caused by a read access. `1` Abort caused by a write access. For faults on CP15 cache maintenance operations, including the VA to PA translation operations, this bit always returns a value of 1.
[10]	-	Reserved, RES0.
[9]	LPAE	On taking a Data Abort exception, this bit is set as follows: `0` Using the Short-descriptor translation table formats. `1` Using the Long-descriptor translation table formats. Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.
[8:6]	-	Reserved, RES0.
[5:0]	Status	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved. `0b000000` Address size fault in TTBR0 or TTBR1. `0b0001LL` Translation fault, LL bits indicate level. `0b0010LL` Access fault flag, LL bits indicate level. `0b0011LL` Permission fault, LL bits indicate level. `0b010000` Synchronous external abort. `0b010001` Asynchronous external abort. `0b0101LL` Synchronous external abort on translation table walk, LL bits indicate level. `0b011000` Synchronous parity error on memory access. `0b011001` Asynchronous parity error on memory access (DFSR only). `0b0111LL` Synchronous parity error on memory access on translation table walk, first level, LL bits indicate level. `0b100001` Alignment fault. `0b100010` Debug event. `0b110000` TLB conflict abort. `0b110101` `LDREX` or `STREX` abort.

The following table shows how the LL bits in the Status field encode the lookup level associated with the MMU fault.

Table 4-239 Encodings of LL bits associated with the MMU fault

Bits	Meaning
`0b00`	Reserved
`0b01`	Level 1
`0b10`	Level 2
`0b11`	Level 3

To access the DFSR:

```
MRC p15, 0, <Rt>, c5, c0, 0; Read DFSR into Rt
MCR p15, 0, <Rt>, c5, c0, 0; Write Rt to DFSR
```

4.5.51 Instruction Fault Status Register

The IFSR characteristics are:

Purpose Holds status information about the last instruction fault.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Configurations IFSR (NS) is architecturally mapped to AArch64 register IFSR32_EL2. See *4.3.66 Instruction Fault Status Register, EL2* on page 4-166.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

Attributes IFSR is a 32-bit register.

There are two formats for this register. The current translation table format determines which format of the register is used. This section describes:

- *IFSR when using the Short-descriptor translation table format* on page 4-304.
- *IFSR when using the Long-descriptor translation table format* on page 4-305.

IFSR when using the Short-descriptor translation table format

The following figure shows the IFSR bit assignments when using the Short-descriptor translation table format.

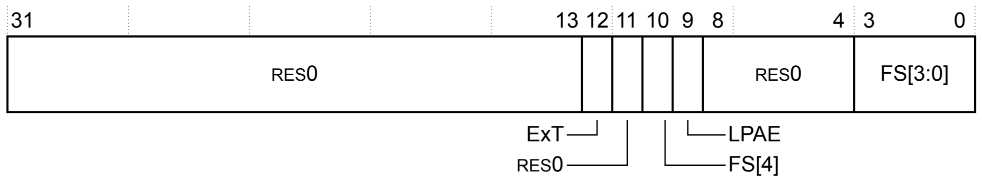

Figure 4-126 IFSR bit assignments for Short-descriptor translation table format

The following table shows the IFSR bit assignments when using the Short-descriptor translation table format.

Table 4-240 IFSR bit assignments for Short-descriptor translation table format

Bits	Name	Function
[31:13]	-	Reserved, RES0.
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: `0` External abort marked as DECERR. `1` External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11]	-	Reserved, RES0.
[10]	FS[4]	Part of the Fault Status field. See bits [3:0] in this table.
[9]	LPAE	On taking a Data Abort exception, this bit is set as follows: `0` Using the Short-descriptor translation table formats. `1` Using the Long-descriptor translation table formats. Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.
[8:4]	-	Reserved, RES0.
[3:0]	FS[3:0]	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved: `0b00010` Debug event. `0b00011` Access flag fault, section. `0b00101` Translation fault, section. `0b00110` Access flag fault, page. `0b00111` Translation fault, page. `0b01000` Synchronous external abort, non-translation. `0b01001` Domain fault, section. `0b01011` Domain fault, page. `0b01100` Synchronous external abort on translation table walk, first level. `0b01101` Permission Fault, section. `0b01110` Synchronous external abort on translation table walk, second level. `0b01111` Permission fault, page. `0b10000` TLB conflict abort. `0b11001` Synchronous parity error on memory access. `0b11100` Synchronous parity error on translation table walk, first level. `0b11110` Synchronous parity error on translation table walk, second level.

IFSR when using the Long-descriptor translation table format

The following figure shows the IFSR bit assignments when using the Long-descriptor translation table format.

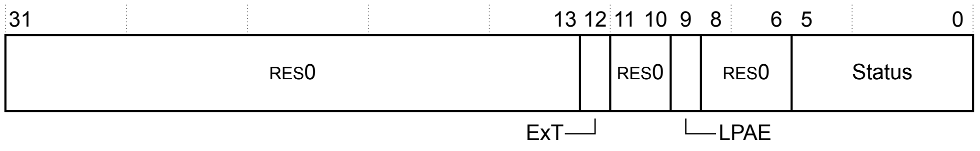

Figure 4-127 IFSR bit assignments for Long-descriptor translation table format

The following table shows the IFSR bit assignments when using the Long-descriptor translation table format.

Table 4-241 IFSR bit assignments for Long-descriptor translation table format

Bits	Name	Function
[31:13]	-	Reserved, RES0.
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: `0` External abort marked as DECERR. `1` External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11:10]	-	Reserved, RES0.
[9]	LPAE	On taking a Data Abort exception, this bit is set as follows: `0` Using the Short-descriptor translation table formats. `1` Using the Long-descriptor translation table formats. Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.
[8:6]	-	Reserved, RES0.
[5:0]	Status	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved: `0b000000` Address size fault in TTBR0 or TTBR1. `0b0001LL` Translation fault, LL bits indicate level. `0b0010LL` Access fault flag, LL bits indicate level. `0b0011LL` Permission fault, LL bits indicate level. `0b010000` Synchronous external abort. `0b0101LL` Synchronous external abort on translation table walk, LL bits indicate level. `0b011000` Synchronous parity error on memory access. `0b0111LL` Synchronous parity error on memory access on translation table walk, LL bits indicate level. `0b100001` Alignment fault. `0b100010` Debug event. `0b110000` TLB conflict abort.

The following table shows how the LL bits in the Status field encode the lookup level associated with the MMU fault.

Table 4-242 Encodings of LL bits associated with the MMU fault

Bits	Meaning
`0b00`	Reserved
`0b01`	Level 1
`0b10`	Level 2
`0b11`	Level 3

Note

If a Data Abort exception is generated by an instruction cache maintenance operation when the Long-descriptor translation table format is selected, the fault is reported as a Cache Maintenance fault in the DFSR or HSR with the appropriate Fault Status code. For such exceptions reported in the DFSR, the corresponding IFSR is UNKNOWN.

To access the IFSR:

```
MRC p15, 0, <Rt>, c5, c0, 1; Read IFSR into Rt
MCR p15, 0, <Rt>, c5, c0, 1; Write Rt to IFSR
```

Register access is encoded as follows:

Table 4-243 IFSR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0101	0000	001

4.5.52 Auxiliary Data Fault Status Register

The processor does not implement ADFSR, so this register is always RES0.

4.5.53 Auxiliary Instruction Fault Status Register

The processor does not implement AIFSR, so this register is always RES0.

4.5.54 Hyp Auxiliary Data Fault Status Syndrome Register

The processor does not implement HADFSR, so this register is always RES0.

4.5.55 Hyp Auxiliary Instruction Fault Status Syndrome Register

The processor does not implement HAIFSR, so this register is always RES0.

4.5.56 Hyp Syndrome Register

The HSR characteristics are:

Purpose Holds syndrome information for an exception taken to Hyp mode.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations HSR is architecturally mapped to AArch64 register ESR_EL2. See *4.3.67 Exception Syndrome Register, EL2* on page 4-169.

This register is accessible only at EL2 or EL3.

Attributes HSR is a 32-bit register.

The following figure shows the HSR bit assignments.

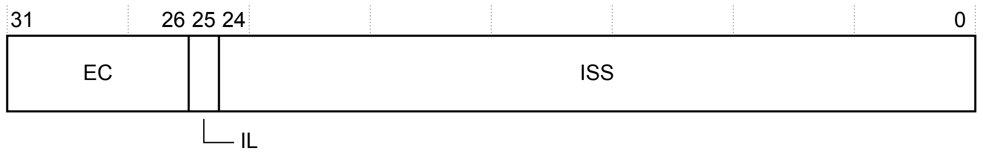

Figure 4-128 HSR bit assignments

The following table shows the HSR bit assignments.

Table 4-244 HSR bit assignments

Bits	Name	Function
[31:26]	EC	Exception Class. The exception class for the exception that is taken in Hyp mode. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[25]	IL	Instruction Length. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[24:0]	ISS	Instruction Specific Syndrome. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information. The interpretation of this field depends on the value of the EC field. See *Encoding of ISS[24:20] when HSR[31:30] is 0b00* on page 4-308.

Encoding of ISS[24:20] when HSR[31:30] is 0b00

For EC values that are nonzero and have the two most-significant bits 0b00, ISS[24:20] provides the condition field for the trapped instruction, together with a valid flag for this field.

The encoding of this part of the ISS field is:

CV, ISS[24] Condition valid. Possible values of this bit are:

`0`	The COND field is not valid.
`1`	The COND field is valid.

When an instruction is trapped, CV is set to 1.

COND, ISS[23:20] The Condition field for the trapped instruction. This field is valid only when CV is set to 1.

If CV is set to 0, this field is RES0.

When an instruction is trapped, the COND field is set to the condition the instruction was executed with.

4.5.57 Data Fault Address Register

The DFAR characteristics are:

Purpose Holds the virtual address of the faulting address that caused a synchronous Data Abort exception.

Usage constraints This register is accessible as follows:

	EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
DFAR(S)	-	-	-	RW	-	-	RW
DFAR(NS)	-	-	RW	-	RW	RW	-

Configurations DFAR(NS) is architecturally mapped to AArch64 register FAR_EL1[31:0]. See *4.3.69 Fault Address Register, EL1* on page 4-171.

DFAR(S) is architecturally mapped to AArch32 register HDFAR. See *4.5.59 Hyp Data Fault Address Register* on page 4-310.

DFAR(S) is architecturally mapped to AArch64 register FAR_EL2[31:0]. See *4.3.70 Fault Address Register, EL2* on page 4-172.

Attributes DFAR is a 32-bit register.

The following figure shows the DFAR bit assignments.

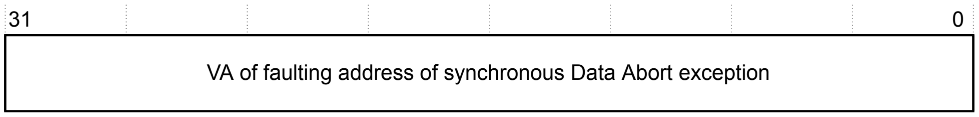

Figure 4-129 DFAR bit assignments

The following table shows the DFAR bit assignments.

Table 4-245 DFAR bit assignments

Bits	Name	Function
[31:0]	VA	The Virtual Address of faulting address of synchronous Data Abort exception

To access the DFAR:

```
MRC p15, 0, <Rt>, c6, c0, 0 ; Read DFAR into Rt
MCR p15, 0, <Rt>, c6, c0, 0 ; Write Rt to DFAR
```

4.5.58 Instruction Fault Address Register

The IFAR characteristics are:

Purpose Holds the virtual address of the faulting address that caused a synchronous Prefetch Abort exception.

Usage constraints This register is accessible as follows:

	EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
IFAR(S)	-	-	-	RW	-	-	RW
IFAR(NS)	-	-	RW	-	RW	RW	-

Configurations IFAR(NS) is architecturally mapped to AArch64 register FAR_EL1[63:32]. See *4.3.69 Fault Address Register, EL1* on page 4-171.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

IFAR(S) is architecturally mapped to AArch32 register HIFAR.

IFAR(S) is architecturally mapped to AArch64 register FAR_EL2[63:32]. See *4.3.70 Fault Address Register, EL2* on page 4-172.

Attributes IFAR is a 32-bit register.

The following figure shows the IFAR bit assignments.

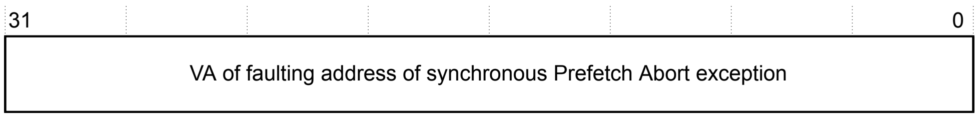
31
0
VA of faulting address of synchronous Prefetch Abort exception

Figure 4-130 IFAR bit assignments

The following table shows the IFAR bit assignments.

Table 4-246 IFAR bit assignments

Bits	Name	Function
[31:0]	VA	The Virtual Address of faulting address of synchronous Prefetch Abort exception

To access the IFAR:

```
MRC p15, 0, <Rt>, c6, c0, 2; Read IFAR into Rt
MCR p15, 0, <Rt>, c6, c0, 2; Write Rt to IFAR
```

4.5.59 Hyp Data Fault Address Register

The HDFAR characteristics are:

Purpose Holds the virtual address of the faulting address that caused a synchronous Data Abort exception that is taken to Hyp mode.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)
-	-	-	RW	RW	-

An execution in a Non-secure EL1 state, or in Secure state, makes the HDFAR UNKNOWN.

Configurations HDFAR is architecturally mapped to AArch64 register FAR_EL2[31:0] when EL3 is AArch64. See *4.3.70 Fault Address Register, EL2* on page 4-172.

HDFAR (S) is architecturally mapped to AArch32 register DFAR (S). See *4.5.57 Data Fault Address Register* on page 4-309.

Attributes HDFAR is a 32-bit register.

The following figure shows the HDFAR bit assignments.

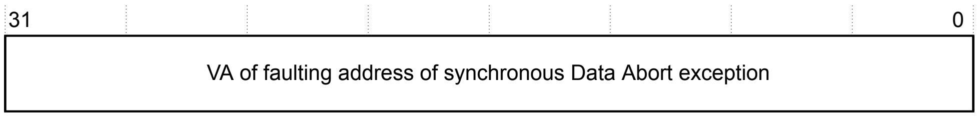

Figure 4-131 HDFAR bit assignments

The following table shows the HDFAR bit assignments.

Table 4-247 HDFAR bit assignments

Bits	Name	Function
[31:0]	VA	The Virtual Address of faulting address of synchronous Data Abort exception

To access the HDFAR:

```
MRC p15, 4, <Rt>, c6, c0, 0 ; Read HDFAR into Rt
MCR p15, 4, <Rt>, c6, c0, 0 ; Write Rt to HDFAR
```

4.5.60 Hyp Instruction Fault Address Register

The HIFAR characteristics are:

Purpose Holds the virtual address of the faulting address that caused a synchronous Prefetch Abort exception that is taken to Hyp mode.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Execution in any Non-secure mode other than Hyp mode makes HIFAR UNKNOWN.

Configurations HIFAR is architecturally mapped to AArch64 register FAR_EL2[63:32]. See *4.3.70 Fault Address Register, EL2* on page 4-172.

HIFAR is architecturally mapped to AArch32 register IFAR (S). See *4.5.58 Instruction Fault Address Register* on page 4-309.

Attributes HIFAR is a 32-bit register.

The following figure shows the HIFAR bit assignments.

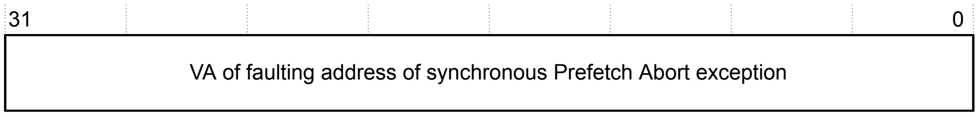
31
0
VA of faulting address of synchronous Prefetch Abort exception

Figure 4-132 HIFAR bit assignments

The following table shows the HIFAR bit assignments.

Table 4-248 HIFAR bit assignments

Bits	Name	Function
[31:0]	VA	The Virtual Address of faulting address of synchronous Prefetch Abort exception

To access the HIFAR:

```
MRC p15, 4, <Rt>, c6, c0, 2 ; Read HIFAR into Rt
MCR p15, 4, <Rt>, c6, c0, 2 ; Write Rt to HIFAR
```

4.5.61 Hyp IPA Fault Address Register

The HPFAR characteristics are:

Purpose Holds the faulting IPA for some aborts on a stage 2 translation that is taken to Hyp mode.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Execution in any Non-secure mode other than Hyp mode makes HPFAR UNKNOWN.

Configurations HPFAR is architecturally mapped to AArch64 register HPFAR_EL2[31:0]. See *4.3.71 Hypervisor IPA Fault Address Register, EL2* on page 4-173.

Attributes HPFAR is a 32-bit register.

The following figure shows the HPFAR bit assignments.

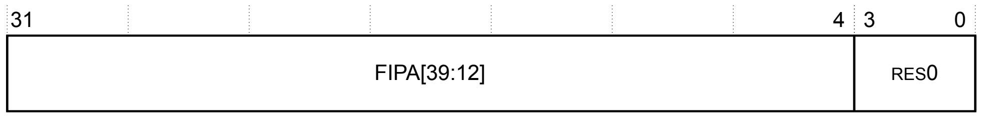

Figure 4-133 HPFAR bit assignments

The following table shows the HPFAR bit assignments.

Table 4-249 HPFAR bit assignments

Bits	Name	Function
[31:4]	FIPA[39:12]	Bits [39:12] of the faulting Intermediate Physical Address
[3:0]	-	Reserved, RES0

To access the HPFAR:

```
MRC p15, 4, <Rt>, c6, c0, 4 ; Read HPFAR into Rt
MCR p15, 4, <Rt>, c6, c0, 4 ; Write Rt to HPFAR
```

4.5.62 Physical Address Register

The processor does not use any implementation-defined bits in the 32-bit format or 64-bit format PAR.

Bit[8] is RES0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

4.5.63 L2 Control Register

The L2CTLR characteristics are:

Purpose Provides IMPLEMENTATION DEFINED control options for the L2 memory system.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Note

This register is write accessible in EL1 if:

- ACTLR_EL3.EN_L2CTLR is 1 and
- ACTLR_EL2.EN_L2CTLR is 1 or SCR_EL3.NS is 0

If write access is not possible, then L2CTLR is trapped to the lowest exception level that denied access (EL2 or EL3).

Configurations

L2CTLR is architecturally mapped to the AArch64 L2CTLR_EL1 register. See *4.3.72 L2 Control Register* on page 4-174.

There is only one L2CTLR for the Cortex-A73 processor.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

L2CTLR is a 32-bit register.

The following figure shows the L2CTLR bit assignments.

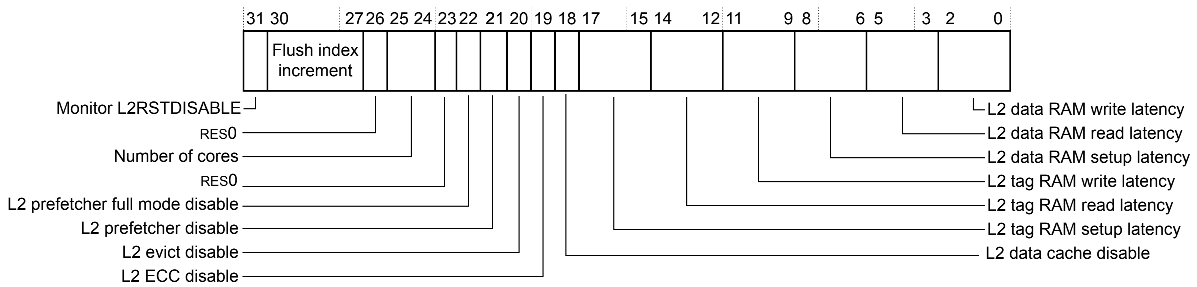

Figure 4-134 L2CTLR bit assignments

The following table shows the L2CTLR bit assignments.

Table 4-250 L2CTLR bit assignments

Bits	Name	Function
[31]	L2RSTDISABLE	Monitor L2RSTDISABLE.
[30:27]	Flush index increment	Flush index increment. The reset value is 0.
[26]	-	Reserved, RES0.

Table 4-250 L2CTLR bit assignments (continued)

Bits	Name	Function
[25:24]	Number of cores	Number of cores present: `0b00` One core, core 0. `0b01` Two cores, core 0 and core 1. `0b10` Three cores, cores 0 to 2. `0b11` Four cores, cores 0 to 3. These bits are read-only and the value of this field is set to the number of cores present in the configuration.
[23]	-	Reserved, RES0.
[22]	L2 prefetcher full mode disable	Disables the full mode of the L2 prefetcher: `0` Enables full mode. `1` Disables full mode. The reset value is 0. **Note** This bit has no effect when the L2 prefetcher disable bit is set to 1.
[21]	L2 prefetcher disable	Disables the L2 prefetcher: `0` Enables prefetcher. `1` Disables prefetcher. The reset value is 0.
[20]	L2 Evict disable	Disables evict information that is sent to interconnect: `0` Enables evict information. `1` Disables evict information.
[19]	L2 ECC disable	Disables L2 ECC protection: `0` Enables ECC protection. `1` Disables ECC protection. The reset value is 0.
[18]	L2 data cache disable[bq]	Disables L2 data cache: `0` Enables L2 data cache. `1` Disables L2 data cache. The reset value is 0.
[17:15]	L2 tag RAM setup latency	L2 tag RAM setup latency: `<n>` <n+1>-cycle setup delay from L2 tag RAMs.

[bq] The L2 cache must never be disabled in a multi-cluster configuration.

Table 4-250 L2CTLR bit assignments (continued)

Bits	Name	Function
[14:12]	L2 tag RAM read latency	L2 tag RAM read latency: `<n>` <n+1>-cycle read delay from L2 tag RAMs.
[11:9]	L2 tag RAM write Latency	L2 tag RAM write latency: `<n>` <n+1>-cycle write delay from L2 tag RAMs.
[8:6]	L2 data RAM setup Latency	L2 data RAM setup latency: `<n>` <n+1>-cycle setup delay from L2 data RAMs.
[5:3]	L2 data RAM read Latency	L2 data RAM read latency: `<n>` <n+1>-cycle read delay from L2 data RAMs.
[2:0]	L2 data RAM write Latency	L2 data RAM write latency: `<n>` <n+1>-cycle write delay from L2 data RAMs.

To access the L2CTLR:

```
MRC p15, 1, <Rt>, c9, c0, 2; Read L2CTLR into Rt
```

4.5.64 L2 Extended Control Register

The L2ECTLR characteristics are:

Purpose Provides additional IMPLEMENTATION DEFINED control options for the L2 memory system. This register is used for dynamically changing, but implementation specific, control bits.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

The L2ECTLR can be written dynamically.

Note

This register is write accessible in EL1 if:
- ACTLR_EL3.EN_L2CTLR is 1 and
- ACTLR_EL2.EN_EN_L2CTLR is 1 or ACTLR_EL3.EN_EN_L2CTLR is 1 and
- SCR.NS is 0.

If write access is not possible, then L2ECTLR is trapped to the lowest exception level that denied access (EL2 or EL3).

Configurations L2ECTLR is architecturally mapped o the AArch64 L2ECTLR_EL1 register. See *4.3.73 L2 Extended Control Register* on page 4-177.

There is one copy of this register that is used in both Secure and Non-secure states.

There is one L2ECTLR for the Cortex-A73 processor.

Attributes L2ECTLR is a 32-bit register.

The following figure shows the L2ECTLR bit assignments.

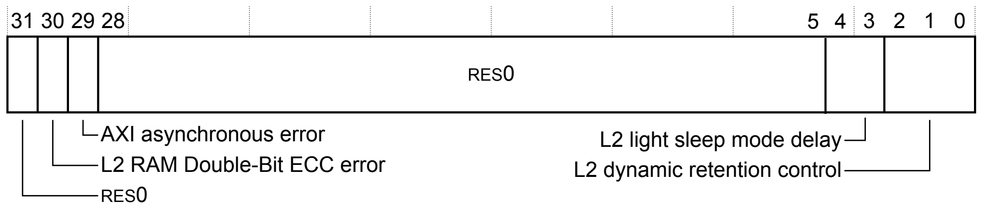
31 30 29 28
5 4 3 2 1 0
RES0
AXI asynchronous error
L2 RAM Double-Bit ECC error
RES0
L2 light sleep mode delay
L2 dynamic retention control

Figure 4-135 L2ECTLR bit assignments

The following table shows the L2ECTLR bit assignments.

Table 4-251 L2ECTLR bit assignments

Bits	Name	Function
[31]	-	Reserved, RES0.
[30]	L2 RAM Double-Bit ECC error	L2 RAM Double-Bit ECC error indication. The possible values are: `0` No pending Double-Bit ECC error. `1` A fatal Double-Bit ECC error has occurred. A write of 0 clears this bit. A write of 1 is ignored.
[29]	AXI asynchronous error	AXI asynchronous error indication. The possible values are: `0` No pending asynchronous error. `1` An asynchronous error has occurred. A write of 0 clears this bit. A write of 1 is ignored.
[28:5]	-	Reserved, RES0.
[4:3]	L2 light sleep mode delay	L2 light sleep mode delay. The possible values are: `0b00` 0 clock cycles before L2 RAM enters sleep mode. `0b01` 16 clock cycles before L2 RAM enters sleep mode. `0b10` 32 clock cycles before L2 RAM enters sleep mode. `0b11` 32 clock cycles before L2 RAM enters sleep mode.
[2:0]	L2 dynamic retention control	L2 Data dynamic retention control. The possible values are: `0b000` L2 dynamic retention disabled. This is the reset value. `0b001` 2 Generic Timer ticks required before retention entry. `0b010` 8 Generic Timer ticks required before retention entry. `0b011` 32 Generic Timer ticks required before retention entry. `0b100` 64 Generic Timer ticks required before retention entry. `0b101` 128 Generic Timer ticks required before retention entry. `0b110` 256 Generic Timer ticks required before retention entry. `0b111` 512 Generic Timer ticks required before retention entry.

To access the L2ECTLR:

```
MRC p15, 1, <Rt>, c9, c0, 3; Read L2ECTLR into Rt
MCR p15, 1, <Rt>, c9, c0, 3; Write Rt to L2ECTLR
```

4.5.65 Primary Region Remap Register

The PRRR characteristics are:

Purpose Controls the top level mapping of the TEX[0], C, and B memory region attributes.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

PRRR is not accessible when the Long-descriptor translation table format is in use. See, instead, *4.5.66 Memory Attribute Indirection Registers 0 and 1* on page 4-321.

Configurations PRRR(NS) is architecturally mapped to AArch64 register MAIR_EL1[31:0] when TTBCR.EAE is 0.

PRRR(S) is mapped to AArch64 register MAIR_EL3[31:0] when TTBCR.EAE is 0.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

Attributes PRRR is a 32-bit register when TTBCR.EAE==0.

The following figure shows the PRRR bit assignments.

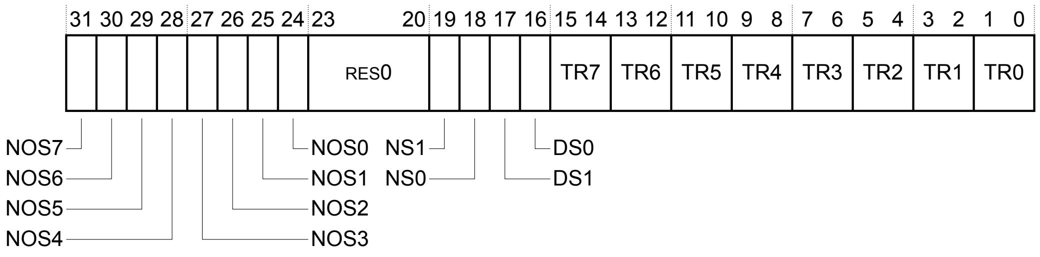

Figure 4-136 PRRR bit assignments

The following table shows the PRRR bit assignments.

Table 4-252 PRRR bit assignments

Bits	Name	Function
[24+*n*][br]	NOS*n*	Outer Shareable property mapping for memory attributes *n*, if the region is mapped as Normal Shareable. *n* is the value of the TEX[0], C and B bits concatenated. The possible values of each NOS*n* bit are: `0` Memory region is Outer Shareable. `1` Memory region is Inner Shareable. The value of this bit is ignored if the region is Normal or Device memory that is not Shareable.
[23:20]	-	Reserved, RES0.

[br] Where n is 0-7.

Table 4-252 PRRR bit assignments (continued)

Bits	Name	Function
[19]	NS1	Mapping of S = 1 attribute for Normal memory. This bit gives the mapped Shareable attribute for a region of memory that: • Is mapped as Normal memory. • Has the S bit set to 1. The possible values of the bit are: `0` Region is not Shareable. `1` Region is Shareable.
[18]	NS0	Mapping of S = 0 attribute for Normal memory. This bit gives the mapped Shareable attribute for a region of memory that: • Is mapped as Normal memory. • Has the S bit set to 0. The possible values of the bit are the same as those given for the NS1 bit, bit[19].
[17]	DS1	Mapping of S = 1 attribute for Device memory. This bit gives the mapped Shareable attribute for a region of memory that: • Is mapped as Device memory. • Has the S bit set to 1. **Note** This field has no significance in the processor.
[16]	DS0	Mapping of S = 0 attribute for Device memory. This bit gives the mapped Shareable attribute for a region of memory that: • Is mapped as Device memory. • Has the S bit set to 0. **Note** This field has no significance in the processor.
[2*n*+1:2*n*][bs]	TR*n*	Primary TEX mapping for memory attributes *n*. *n* is the value of the TEX[0], C and B bits, see *Table 4-253 Memory attributes and the n value for the PRRR field descriptions* on page 4-320. This field defines the mapped memory type for a region with attributes *n*. The possible values of the field are: `0b00` Device (nGnRnE). `0b01` Device (not nGnRnE). `0b10` Normal Memory. `0b11` Reserved, effect is UNPREDICTABLE.

The following table shows the mapping between the memory region attributes and the *n* value used in the PRRR.nOS*n* and PRRR.TR*n* field descriptions.

[bs] Where n is 0-7.

Table 4-253 Memory attributes and the *n* value for the PRRR field descriptions

Attributes			*n* value
TEX[0]	C	B	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Large physical address translations use Long-descriptor translation table formats and MAIR0 replaces the PRRR, and MAIR1 replaces the NMRR. For more information see *4.5.66 Memory Attribute Indirection Registers 0 and 1* on page 4-321.

To access the PRRR:

```
MRC p15, 0, <Rt>, c10, c2, 0    ; Read PRRR into Rt
MCR p15, 0, <Rt>, c10, c2, 0    ; Write Rt to PRRR
```

4.5.66 Memory Attribute Indirection Registers 0 and 1

The MAIR0 and MAIR1 characteristics are:

Purpose To provide the memory attribute encodings corresponding to the possible AttrIndx values in a Long-descriptor format translation table entry for stage 1 translations.

Usage constraints These registers are accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Accessible only when using the Long-descriptor translation table format. When using the Short-descriptor format see, instead, *4.5.65 Primary Region Remap Register* on page 4-318 and *4.5.67 Normal Memory Remap Register* on page 4-323.

AttrIndx[2], from the translation table descriptor, selects the appropriate MAIR: setting AttrIndx[2] to 0 selects MAIR0.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

The Secure instance of the register gives the value for memory accesses from Secure state.

The Non-secure instance of the register gives the value for memory accesses from Non-secure states other than Hyp mode.

Configurations MAIR0(NS) is architecturally mapped to AArch64 register MAIR_EL1[31:0] when TTBCR.EAE==1. See *4.3.76 Memory Attribute Indirection Register, EL1* on page 4-181.

MAIR0(S) is mapped to AArch64 register MAIR_EL3[31:0] when TTBCR.EAE==1. See *4.3.78 Memory Attribute Indirection Register, EL3* on page 4-184.

If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

MAIR0 has write access to the Secure instance of the register disabled when the **CP15SDISABLE** signal is asserted HIGH.

Attributes MAIR0 is a 32-bit register when TTBCR.EAE==1.

The following figure shows the MAIR0 and MAIR1 bit assignments.

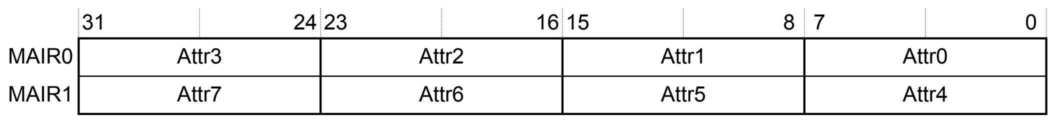

Figure 4-137 MAIR0 and MAIR1 bit assignments

The following table shows the MAIR0 and MAIR1 bit assignments.

Table 4-254 MAIR0 and MAIR1 bit assignments

Bits	Name	Description
[7:0]	Attr*m*[bt]	The memory attribute encoding for an AttrIndx[2:0] entry in a Long-descriptor format translation table entry, where: • AttrIndx[2] selects the appropriate MAIR: — Setting AttrIndx[2] to 0 selects MAIR0. — Setting AttrIndx[2] to 1 selects MAIR1. • AttrIndx[2:0] gives the value of <n> in Attr<n>.

The following table shows the Attr<n>[7:4] bit assignments.

Table 4-255 Attr<n>[7:4] bit assignments

Bits	Meaning
`0b0000`	Device memory. See The following table for the type of Device memory.
`0b00RW`, RW not `00`	Normal Memory, Outer Write-through transient.[bu]
`0b0100`	Normal Memory, Outer Non-Cacheable.
`0b01RW`, RW not `00`	Normal Memory, Outer Write-back transient.
`0b10RW`	Normal Memory, Outer Write-through non-transient.
`0b11RW`	Normal Memory, Outer Write-back non-transient.

The following table shows the Attr<n>[3:0] bit assignments. The encoding of Attr<n>[3:0] depends on the value of Attr<n>[7:4], as the following table shows.

Table 4-256 Attr<n>[3:0] bit assignments

Bits	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
`0b0000`	Device-nGnRnE memory	UNPREDICTABLE
`0b00RW`, RW not `00`	UNPREDICTABLE	Normal Memory, Inner Write-through transient
`0b0100`	Device-nGnRE memory	Normal memory, Inner Non-Cacheable
`0b01RW`, RW not `00`	UNPREDICTABLE	Normal Memory, Inner Write-back transient
`0b1000`	Device-nGRE memory	Normal Memory, Inner Write-through non-transient (RW=00)
`0b10RW`, RW not `00`	UNPREDICTABLE	Normal Memory, Inner Write-through non-transient
`0b1100`	Device-GRE memory	Normal Memory, Inner Write-back non-transient (RW=00)
`0b11RW`, RW not `00`	UNPREDICTABLE	Normal Memory, Inner Write-back non-transient

The following table shows the encoding of the R and W bits that are used, in some Attr<n> encodings in *Table 4-255 Attr<n>[7:4] bit assignments* on page 4-322 and *Table 4-256 Attr<n>[3:0] bit assignments* on page 4-322, to define the read-allocate and write-allocate policies:

[bt] Where m is 0-7.
[bu] The transient hint is ignored.

Table 4-257 Encoding of R and W bits in some Attr<n> fields

R or W	Meaning
0	Do not allocate
1	Allocate

To access the MAIR0:

```
MRC p15, 0, <Rt>, c10, c2, 0    ; Read MAIR0 into Rt
MCR p15, 0, <Rt>, c10, c2, 0    ; Write Rt to MAIR0
```

To access the MAIR1:

```
MRC p15, 0, <Rt>, c10, c2, 1    ; Read MAIR1 into Rt
MCR p15, 0, <Rt>, c10, c2, 1    ; Write Rt to MAIR1
```

4.5.67 Normal Memory Remap Register

The NMRR characteristics are:

Purpose Provides additional mapping controls for memory regions that are mapped as Normal memory by their entry in the PRRR.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

The register is:
- Used in conjunction with the PRRR.
- Not accessible when using the Long-descriptor translation table format.

Configurations If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

The Non-secure NMRR is architecturally mapped to the AArch64 MAIR_EL1[63:32] register when TTBCR.EAE==0.

The Secure NMRR is mapped to the AArch64 MAIR_EL3[63:32] register when TTBCR.EAE==0.

Attributes NMRR is a 32-bit register when TTBCR.EAE==0.

The following figure shows the NMRR bit assignments.

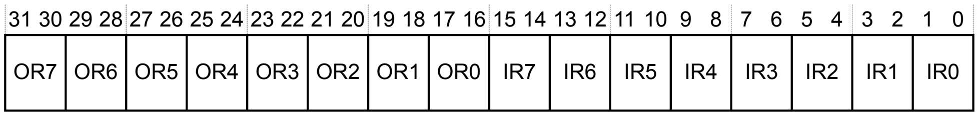

Figure 4-138 NMRR bit assignments

The following table shows the NMRR bit assignments.

Table 4-258 NMRR bit assignments

Bits	Name	Description
[2n+17:2n +16][bv]	ORn	Outer Cacheable property mapping for memory attributes *n*, if the region is mapped as Normal memory by the PRRR.TR*n* entry. *n* is the value of the TEX[0], C and B bits, see *Table 4-253 Memory attributes and the n value for the PRRR field descriptions* on page 4-320. The possible values of this field are: `0b00` Region is Non-cacheable. `0b01` Region is Write-Back, Write-Allocate. `0b10` Region is Write-Through, no Write-Allocate. `0b11` Region is Write-Back, no Write-Allocate.
[2n+1:2n][bv]	IRn	Inner Cacheable property mapping for memory attributes *n*, if the region is mapped as Normal Memory by the PRRR.TR*n* entry. *n* is the value of the TEX[0], C and B bits, see *Table 4-253 Memory attributes and the n value for the PRRR field descriptions* on page 4-320. The possible values of this field are the same as those given for the OR*n* field.

To access the NMRR:

```
MRC p15, 0, <Rt>, c10, c2, 1    ; Read NMRR into Rt
MCR p15, 0, <Rt>, c10, c2, 1    ; Write Rt to NMRR
```

4.5.68 Auxiliary Memory Attribute Indirection Register 0

The processor does not implement AMAIR0, so this register is always RES0.

4.5.69 Auxiliary Memory Attribute Indirection Register 1

The processor does not implement AMAIR1, so this register is always RES0.

4.5.70 Hyp Auxiliary Memory Attribute Indirection Register 0

The processor does not implement HAMAIR0, so this register is always RES0.

4.5.71 Hyp Auxiliary Memory Attribute Indirection Register 1

The processor does not implement HAMAIR1, so this register is always RES0.

4.5.72 Vector Base Address Register

The VBAR characteristics are:

Purpose Holds the exception base address for exceptions that are not taken to Monitor mode or to Hyp mode when high exception vectors are not selected.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Software must program the Non-secure instance of the register with the required initial value as part of the processor boot sequence.

[bv] Where n is 0-7.

Configurations If EL3 is using AArch32, there are separate Secure and Non-secure instances of this register.

The Non-secure VBAR is architecturally mapped to the AArch64 VBAR_EL1 register. See *4.3.79 Vector Base Address Register, EL1* on page 4-184.

The Secure VBAR is mapped to AArch64 register VBAR_EL3[31:0]. See *4.3.81 Vector Base Address Register, EL3* on page 4-186.

Attributes VBAR is a 32-bit register.

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

To access the VBAR:

```
MRC p15, 0, <Rt>, c12, c0, 0 ; Read VBAR into Rt
MCR p15, 0, <Rt>, c12, c0, 0 ; Write Rt to VBAR
```

4.5.73 Reset Management Register

The RMR characteristics are:

Purpose Controls the execution state that the processor boots into and allows request of a Warm reset.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	-	RW	RW

Configurations The RMR is architecturally mapped to the AArch64 RMR_EL3 register.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes RMR is a 32-bit register.

The following figure shows the RMR bit assignments.

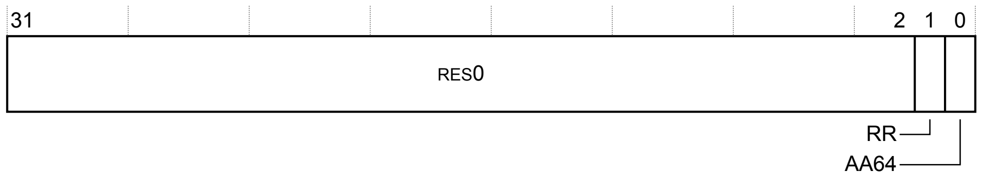

Figure 4-139 RMR bit assignments

The following table shows the RMR bit assignments.

Table 4-259 RMR bit assignments

Bits	Name	Function
[31:2]	-	Reserved, RES0.
[1]	RR	Reset Request. The possible values are: `0` This is the reset value. `1` Requests a Warm reset. This bit is set to 0 by either a cold or Warm reset. The bit is strictly a request. The RR bit drives the **WARMRSTREQ** output signal.
[0]	AA64	Determines which execution state the processor boots into after a Warm reset. The possible values are: `0` AArch32 Execution state. `1` AArch64 Execution state. The Cold reset value depends on the **AA64nAA32** signal.

To access the RMR:

```
MRC p15,0,<Rt>,c12,c0,2 ; Read RMR into Rt
MCR p15,0,<Rt>,c12,c0,2 ; Write Rt to RMR
```

Register access is encoded as follows:

Table 4-260 RMR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	1100	0000	010

4.5.74 Interrupt Status Register

The ISR characteristics are:

Purpose Shows whether an IRQ, FIQ, or external abort is pending. An indicated pending abort might be a physical abort or a virtual abort.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Configurations ISR is architecturally mapped to AArch64 register ISR_EL1. See *4.3.84 Interrupt Status Register* on page 4-188.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes ISR is a 32-bit register.

The following figure shows the ISR bit assignments.

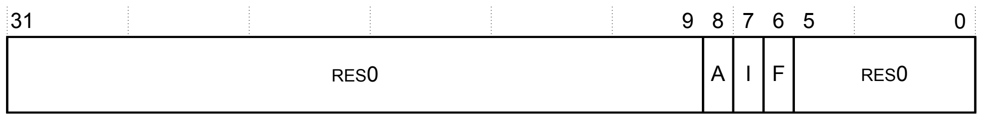

Figure 4-140 ISR bit assignments

The following table shows the ISR bit assignments.

Table 4-261 ISR bit assignments

Bits	Name	Function
[31:9]	-	Reserved, RES0.
[8]	A	External abort pending bit: `0` No pending external abort. `1` An external abort is pending.
[7]	I	IRQ pending bit. Indicates whether an IRQ interrupt is pending: `0` No pending IRQ. `1` An IRQ interrupt is pending.
[6]	F	FIQ pending bit. Indicates whether an FIQ interrupt is pending: `0` No pending FIQ. `1` An FIQ interrupt is pending.
[5:0]	-	Reserved, RES0.

To access the ISR:

```
MRC p15, 0, <Rt>, c12, c1, 1; Read ISR into Rt
```

Register access is encoded as follows:

Table 4-262 ISR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	1100	0001	000

4.5.75 Hyp Vector Base Address Register

The HVBAR characteristics are:

Purpose Holds the exception base address for any exception that is taken to Hyp mode.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	-	RW	RW	-

Configurations The HVBAR is:

- Architecturally mapped to the AArch64 VBAR_EL2[31:0]. See *4.3.80 Vector Base Address Register, EL2* on page 4-185.

Attributes HVBAR is a 32-bit register.

The following figure shows the HVBAR bit assignments.

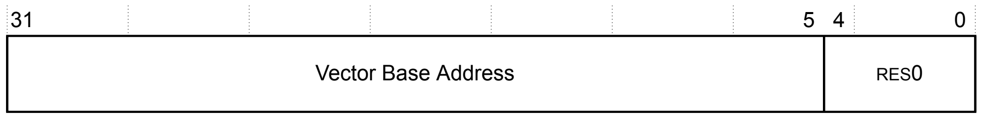

Figure 4-141 HVBAR bit assignments

The following table shows the HVBAR bit assignments.

Table 4-263 HVBAR bit assignments

Bits	Name	Function
[31:5]	Vector Base Address	Bits[31:5] of the base address of the exception vectors, for exceptions taken in this exception level. Bits[4:0] of an exception vector are the exception offset.
[4:0]	-	Reserved, RES0.

To access the HVBAR:

```
MRC p15, 4, <Rt>, c12, c0, 0 ; Read HVBAR into Rt
MCR p15, 4, <Rt>, c12, c0, 0 ; Write Rt to HVBAR
```

4.5.76 FCSE Process ID Register

The processor does not implement *Fast Context Switch Extension* (FCSE), so this register is always RES0.

4.5.77 Extended Control Register

The ECTLR characteristics are:

Purpose Provides configuration and control options for the L1 and L2 memory systems.

Usage constraints 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

The ECTLR:

- This register is write accessible in EL1 if ACTLR_S.ECTLR_EN is `1` and HACTLR.ECTLR_EN is `1`, or ACTLR_S.ECTLR_EN is `1` and SCR.NS is `0`.
- If write access not possible, then trap to the lowest exception level that denied access (EL2 or EL3).

Configurations The ECTLR is mapped to the AArch64 ECTLR_EL1 register. See *4.3.85 Extended Control Register, EL1* on page 4-189.

Attributes ECTLR is a 32-bit register.

The following figure shows the ECTLR bit assignments.

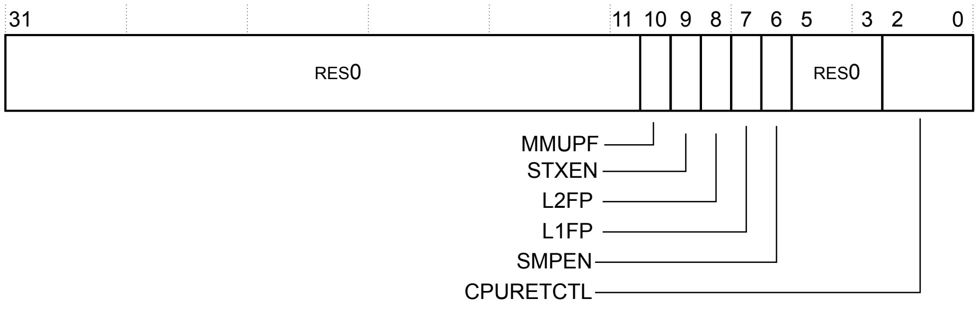

Figure 4-142 ECTLR bit assignments

The following table shows the ECTLR bit assignments.

Table 4-264 ECTLR bit assignments

Bits	Name	Function
[31:11]	-	Reserved, RES0.
[10]	MMUPF	Enables MMU prefetch. The reset value is `0b1`.
[9]	STXEN	Enables exclusive signaling on the ACE master interface for CleanUnique requests due to cacheable Exclusive stores. The reset value is 0.
[8]	L2PF	Enable L2 prefetch requests sent by the stride prefetcher. The reset value is `0b1`.
[7]	L1PF	Enable L1 prefetch requests sent by the stride prefetcher. The reset value is `0b1`.
[6]	SMPEN	Enable hardware management of data coherency with other cores in the cluster. The possible values are: `0` Disables data coherency with other cores in the cluster. This is the reset value. `1` Enables data coherency with other cores in the cluster. **Note** Set the SMPEN bit before enabling the caches, even if there is only one core in the system.
[5:3]	-	Reserved, RES0.
[2:0]	CPURETCTL	CPU retention control. The possible values are: `0b000` Disable the retention circuit. This is the reset value. `0b001` 2 Architectural Timer ticks are required before retention entry. `0b010` 8 Architectural Timer ticks are required before retention entry. `0b011` 32 Architectural Timer ticks are required before retention entry. `0b100` 64 Architectural Timer ticks are required before retention entry. `0b101` 128 Architectural Timer ticks are required before retention entry. `0b110` 256 Architectural Timer ticks are required before retention entry. `0b111` 512 Architectural Timer ticks are required before retention entry.

To access the ECTLR:

```
MRRC p15,1,<Rt>,<Rt2>,c15; Read ECTLR_EL1 into Rt
MCRR p15,1,<Rt>,<Rt2>,c15; Write Rt to ECTLR_EL1
```

or:

```
MRC p15,1,<Rt>,c15,c2,1 ; Read ECTLR[31:0] into Rt
MCR p15,1,<Rt>,c15,c2,1 ; Write Rt to ECTLR[31:0]
```

4.5.78 L2 Memory Error Syndrome Register

The L2MERRSR characteristics are:

Purpose Holds ECC errors on the:
- L2 data RAMs.
- L2 tag RAMs.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RW	RW	RW	RW	RW

Configurations The L2MERRSR is:
- Architecturally mapped to the AArch64 L2MERRSR_EL1 register. See *4.3.86 L2 Memory Error Syndrome Register* on page 4-191.
- There is one copy of this register that is used in both Secure and Non-secure states.
- A write of any value to the register updates the register to `0x0`.
- Accessing this register from an external debugger is not possible if EL3 is in AArch32 state (external debug mode).

Attributes L2MERRSR is a a 64-bit register.

The following figure shows the L2MERRSR bit assignments.

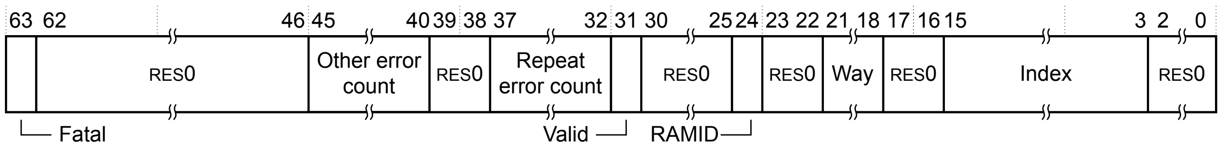

Figure 4-143 L2MERRSR bit assignments

The following table shows the L2MERRSR bit assignments.

Table 4-265 L2MERRSR bit assignments

Bits	Name	Function	
[63]	Fatal	Fatal bit. This bit is set to 1 on the first memory error that caused a data abort. It is a sticky bit so that after it is set, it remains set until the register is written. The reset value is 0.	
[62:46]	-	Reserved, RES0.	

Table 4-265 L2MERRSR bit assignments (continued)

Bits	Name	Function
[45:40]	Other error count	This field is set to 0 on the first memory error and is incremented on any memory error that does not match the RAMID and Bank/Way information in this register while the sticky Valid bit is set. The reset value is 0.
[39:38]	-	Reserved, RES0.
[37:32]	Repeat error count	This field is set to 0 on the first memory error and is incremented on any memory error that exactly matches the RAMID and Bank/Way information in this register while the sticky Valid bit is set. The reset value is 0.
[31]	Valid	Valid bit. This bit is set to 1 on the first memory error. It is a sticky bit so that after it is set, it remains set until the register is written. The reset value is 0.
[30:25]	-	Reserved, RES0.
[24]	RAMID	RAM Identifier. Indicates the RAM in which the first memory error occurred. The possible values are: `0x0` L2 tag RAM. `0x1` L2 data RAM. The reset value is 0.
[23:22]	-	Reserved, RES0.
[21:18]	Way	Indicates the RAM where the first memory error occurred. The reset value is 0. **L2 tag RAM**: `0x0` Way 0; `0x1` Way 1; ...; `0xE` Way 14; `0xF` Way 15. **L2 data RAM**: `0x0` Bank 0; `0x1` Bank 1; ...; `0x7` Bank 7; `0x8-0xF` Unused
[17:16]	-	Reserved, RES0.
[15:3]	Index	Indicates the index address of the first memory error. The reset value is 0.
[2:0]	-	Reserved, RES0.

Note

- A fatal error results in the RAMID, CPU ID/Way and RAM address recording the fatal error, even if the sticky bit was set.
- If two or more memory errors in the same RAM occur in the same cycle, only one error is reported.

- If two or more first memory error events from different RAMs occur in the same cycle, one of the errors is selected arbitrarily, while the Other error count field is incremented only by one.
- If two or more memory error events from different RAMs, that do not match the RAMID, bank, way, or index information in this register while the sticky Valid bit is set, occur in the same cycle, the Other error count field is incremented only by one.

To access the L2MERRSR:

```
MRRC p15, 3, <Rt>, <Rt2>, c15;  Read L2MERRSR into Rt and Rt2
MCRR p15, 3, <Rt>, <Rt2>, c15;  Write Rt and Rt2 to L2MERRSR
```

4.5.79 Configuration Base Address Register

The CBAR characteristics are:

Purpose
Holds the physical base address of the memory-mapped GIC CPU interface registers.

Usage constraints
This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Configurations
This register is available in all build configurations.

Attributes
CBAR is a 32-bit register.

The following figure shows the CBAR bit assignments.

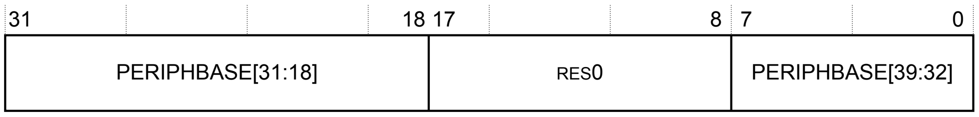

Figure 4-144 CBAR bit assignments

The following table shows the CBAR bit assignments.

Table 4-266 CBAR bit assignments

Bits	Name	Function
[31:18]	PERIPHBASE[31:18]	The input **PERIPHBASE[31:18]** determines the reset value.
[17:8]	-	Reserved, RES0.
[7:0]	PERIPHBASE[39:32]	The input **PERIPHBASE[39:32]** determines the reset value.

To access the CBAR:

```
MRC p15, 1, <Rt>, c15, c3, 0; Read CBAR into Rt
```

4.5.80 DCCIALL Clean Invalidate All

The DCCIALL characteristics are:

Purpose Cleans and invalidates all data caches or unified caches.

Usage constraints This operation can be performed at the following exception levels:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-		WO	WO	WO	WO	WO

Configurations DCCIALL performs the same function as AArch64 operation DC CIALL.

Attributes DCCIALL is a 32-bit system operation.

The following figure shows the DCCIALL input value bit assignments.

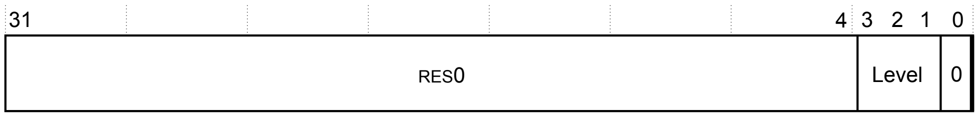

Figure 4-145 DCCIALL input value bit assignments

The DCCIALL input value bit assignments are:.

Table 4-267 DCCIALL bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:1]	Level	Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.
[0]	-	Reserved, RES0.

To perform the DCCIALL operation:

```
MCR p15,1,<Rt>,c15,c14,0 ; DCCIALL operation
```

The operation is encoded as follows:

Table 4-268 DCCIALL access encoding

coproc	opc1	CRn	CRm	opc2
1111	001	1111	1110	000

Chapter 5
Memory Management Unit

This chapter describes the *Memory Management Unit* (MMU).

It contains the following sections:

5.1 About the MMU

The MMU controls address translation, access permissions, and memory attribute determination and checking, for memory accesses made by the Cortex-A73 processor. The address translation process maps the *Virtual Addresses* (VAs) used by the processor onto the *Physical Addresses* (PAs) of the physical memory system. These translations are defined independently for different Exception levels and Security states. Each enabled stage of address translation uses a set of address translations and associated memory properties held in memory mapped tables called *translation tables*. Translation table entries can be cached into a *Translation Lookaside Buffer* (TLB).

The MMU in each core includes the following:

- 32 entries fully-associative instruction micro TLB.
- 48 entries fully-associative data micro TLB.
- A main TLB consisting of two cache RAMs, one with 1024 entries and the other with 128 entries.
  - 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.
- A page table prefetcher that detects access to contiguous page tables and prefetches the next one. This prefetcher can be disabled in ECTLR register.

The Cortex-A73 processor is an Armv8 compliant processor that supports execution in both the AArch64 and AArch32 states. In AArch32 state, the Armv8 address translation system resembles the Armv7 address translation system with LPAE and Virtualization Extensions. In AArch64 state, 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. For more information regarding the address translation formats, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. Key differences between the AArch64 and AArch32 address translation systems are that the AArch64 state provides:

- A translation granule of 4KB, 16KB, or 64KB for LPAE. AArch32 supports both VMSA and LPAE and the translation granule is limited to 4KB.
- A 16-bit ASID. In AArch32, the ASID is limited to an 8-bit value.

In AArch32 state, the only supported physical address size is 40 bits.

The AArch64 state supports a maximum of 40-bit physical addresses. This means that 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.

If stage 2 address translations are enabled, the initial stage 2 lookup begins at the next translation level and therefore the software must use concatenated page tables at the next translation level. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information on concatenated translation tables.

5.2 TLB organization

This section contains the following subsections:

- *5.2.1 Instruction micro TLB* on page 5-336.
- *5.2.2 Data micro TLB* on page 5-336.
- *5.2.3 Main TLB* on page 5-336.

5.2.1 Instruction micro TLB

The instruction micro TLB is implemented as a 32 entry fully-associative structure. This TLB caches entries at the 4KB, 16KB, 64KB, and 1MB granularity of *Virtual Address* (VA) to *Physical Address* (PA) mapping only.

A hit in the instruction micro TLB provides a single **CLK** cycle access to the translation, and returns the physical address to the instruction cache for comparison. It also checks the access permissions to signal a Prefetch Abort.

5.2.2 Data micro TLB

The data micro TLB is a 48 entry fully-associative TLB that is used for data loads and stores. The cache entries have 4KB, 16KB, 64KB, and 1MB granularity of VA to PA mappings only.

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

5.2.3 Main TLB

Misses from the instruction and data micro TLBs are handled by a main TLB which contains two cache RAMs accessed in parallel:

- A 4-way, set-associative, 1024 entry cache which stores VA to PA mappings for 4KB, 16KB, and 64KB page sizes.
- A 2-way, set-associative, 128 entry cache which stores:
  - VA to PA mappings for 1MB, 2MB, 16MB, 32MB, 512MB, and 1GB block sizes.
  - *Intermediate Physical Address* (IPA) to PA mappings for 2MB (in a 4KB translation granule and in AArch32), 16MB (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 table walk.

  Accesses to the main TLB take a variable number of cycles, based on:

- Competing requests from each of the micro TLBs.
- The TLB maintenance operations in flight.
- The different page or block size mappings in use.

5.3 TLB match process

The Armv8-A architecture provides for multiple maps from the VA space that are translated differently.

The TLB entries store all the required context information to facilitate a match and avoid the requirement for a TLB flush on a context or virtual machine switch. Each TLB entry contains a VA, block size, PA, and a set of memory properties that include the memory type and access permissions. Each entry is associated with a particular ASID, or is global for all application spaces. The TLB entry also contains a field to store the VMID in the entry, applicable to accesses made from the Non-secure EL0 and EL1 Exception levels. There is also a memory space identifier that records whether the request occurred at:

- EL3, if EL3 is executing in AArch64 state.
- Non-secure EL2.
- Secure EL0 or EL1, and EL3 when EL3 is executing in AArch32 state.
- Non-secure EL0 or EL1.

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

- Its VA, moderated by the page size such as the VA bits[47:N], where N is $\log_2$ of the block size for that translation stored in the TLB entry, matches that of the requested address.
- The memory space matches the memory space state of the requests.
- 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 register.

Note

- The ASID and VMID matches are ignored for a request originating from EL2 or AArch64 EL3.
- The VMID match is ignored for a request not originating from Non-secure EL0 or EL1.
- Lookups in TLB are only performed on block size already present for this translation regime in main TLB.

5.4 Memory access sequence

When the core generates a memory access, the MMU:

1. Performs a lookup for the requested VA, current ASID, current VMID, and memory space in the relevant instruction or data micro TLB.
2. Performs a lookup for the requested VA, current ASID, current VMID, and memory space in the main TLB if there is a miss in the relevant micro TLB.
3. Performs a hardware translation table walk if there is a miss in the main TLB.

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.

———— **Note** ————

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.

————————————

When executing in AArch32 state in a particular mode, you can configure the MMU to perform hardware translation table walks using either the Short-descriptor translation table format, or the Long-descriptor translation table format. This is controlled by programming the *Extended Address Enable* (EAE) bit in the appropriate Secure or Non-secure *Translation Table Base Control Register* (TTBCR). See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information on translation table formats.

———— **Note** ————

Translations in Hyp mode are always performed with the Long-descriptor translation table format.

————————————

You can configure the MMU to perform translation table walks in Cacheable regions, by programming the IRGN bits in the:

AArch32
- Translation table base registers (TTBR0/TTBR1_EL*x*) when using the Short-descriptor translation table format.
- TCR_EL*x* register when using the Long-descriptor translation table format.

AArch64 Appropriate TCR_EL*x* register.

For stage 2 translations, the IRGN bits must be programmed in the VTCR_EL2 register.

If the encoding of both the ORGN and IRGN bits is Write-Back, the data cache lookup is performed and data is read from the data cache. If the ORGN and IRGN bit contain different attributes or if the encoding of the ORGN and IRGN bits is Write-Through or Non-cacheable, an access to external memory is performed.

In the case of a main TLB miss, the hardware does a translation table walk provided the MMU is enabled, and the translation using the base register has not been disabled. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

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 and the domain 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:
  - A description of the various faults.
  - The fault codes.
  - Information regarding the registers where the fault codes are set.

  Note

  In AArch32 VMSA Short-descriptor format, the permission check includes the domain properties.

- The memory region attributes specified in the TLB entry determine if the access is:
  - Secure or Non-secure.
  - Inner or Outer.
  - Normal Memory or Device type, Strongly-ordered or Device type when using the Short-descriptor format in AArch32 state.
  - One of the four different device memory types defined for Armv8:

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

- The TLB translates the VA to a PA for the memory access.

5.5 MMU aborts

Certain faults and aborts can cause an exception to be taken because of a memory access.

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

5.5.1 External aborts

External aborts are defined as those that occur in the memory system rather than those that the MMU detects. Normally, external memory aborts are rare. External aborts are caused by errors flagged to the ACE interface.

When an external abort occurs on an access for a translation table walk access, the MMU returns a synchronous external abort. For a load multiple or store multiple operation, the address captured in the fault address register is that of the address that generated the synchronous external abort.

See *4.3.50 Secure Configuration Register* on page 4-144, *4.5.32 Secure Configuration Register* on page 4-263, or the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*, for more information.

5.5.2 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 defined for stage 1 by TxSZ for TTBx or for stage 2 by {SL0, T0SZ}.

The Cortex-A73 processor treats such a block, that might be a block within a contiguous set of blocks, as causing a Translation fault, even though the block is valid, and the address accessed within that block is within the size of the input address supported at a stage of translation. For more information, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Chapter 6
Level 1 Memory System

This chapter describes the *Level 1* (L1) memory system.

It contains the following sections:

6.1 About the L1 memory system

The L1 memory system consists of separate instruction and data caches. The size of the instruction cache is 64KB. The size of the data cache is configurable to either 32KB or 64KB.

The L1 instruction memory system has the following key features:

- *Virtually Indexed, Physically Tagged* (VIPT), four-way set-associative instruction cache.

  Note

  In the L1 instruction memory subsystems, aliases are not handled in hardware.

- Fixed cache line length of 64 bytes.
- Pseudo-random cache replacement policy.
- 128-bit read interface from the L2 memory system.

The L1 data memory system has the following features:

- VIPT, four-way set associative L1 data cache.

  Note

  In the L1 data memory subsystems, aliases are handled in hardware and from the programmer's point of view, the data cache behaves like an eight-way set associative PIPT cache (for 32KB configurations) and a 16-way set associative PIPT cache (for 64KB configurations).

- Fixed cache line length of 64 bytes.
- Pseudo-random cache replacement policy.
- 256-bit write interface to the L2 memory system.
- 128-bit read interface from the L2 memory system.
- Two 64-bit read paths from the data L1 memory system to the datapath.
- 64-bit write path from the datapath to the L1 memory system.

6.2 Cache behavior

The implementation-specific features of the instruction and data caches include:

- At reset the instruction and data caches are disabled.
- Both caches are automatically invalidated immediately after reset.

  Note

  When set HIGH, the **DBGL1RSTDISABLE** input signal disables the automatic hardware controlled invalidation of the L1 instruction and data caches after the processor is reset, using **nCORERESET** or **nCPUPORESET**.

- You can enable or disable each L1 (instruction or data) caches independently, but you cannot disable the L2 unified cache and L1 caches independently because they are controlled by the same enable bits. See *4.5.29 System Control Register* on page 4-256.
- Cache lockdown is not supported.
- On a cache miss, data for the cache linefill is requested in critical word-first order.

6.3 Support for v8 memory types

The Armv8-A architecture introduces several new memory types in place of the Armv7 Device and Strongly-Ordered memory types. These relate to the following attributes:

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

R Reordering. The capability to reorder transactions.

E Early-acknowledge. The capability to accept early acknowledge of transactions from the interconnect.

The following table describes the Armv8 memory types.

Table 6-1 Armv8 memory types

Memory type	Comment
GRE	Similar to *Normal* non-cacheable, but does not permit speculative accesses.
nGRE	Treated as nGnRE inside the Cortex-A73 processor, but can be reordered by the external interconnect.
nGnRE	Corresponds to *Device* in Armv7.
nGnRnE	Corresponds to *Strongly-Ordered* in Armv7. Treated the same as nGnRE inside a Cortex-A73 processor, but reported differently on **AxCACHE**.

As defined by the architecture, these bits apply only when the translation table is marked as Armv8 Device memory, they do not apply to Normal memory. If an Armv7 architecture operating system runs on a Cortex-A73 processor, the Device memory type matches the nGnRE encoding and the Strongly-Ordered memory type matches the nGnRnE memory type.

6.4 L1 instruction memory system

The L1 instruction side memory system is responsible for providing an instruction stream to the core. To increase overall performance and to reduce power consumption, it contains the following functionality:

- Branch prediction.
- Instruction caching.

This section contains the following subsections:

- *6.4.1 Instruction cache disabled behavior* on page 6-345.
- *6.4.2 Instruction cache speculative memory accesses* on page 6-345.
- *6.4.3 Program flow prediction* on page 6-345.

6.4.1 Instruction cache disabled behavior

If the instruction cache is disabled, fetches cannot access any of the instruction cache arrays. An exception to this rule is the CP15 instruction cache 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 means that instruction fetches might not be coherent with caches in other cores, and software must take account of this.

6.4.2 Instruction cache speculative memory accesses

Because there can be several unresolved branches in the pipeline, instruction fetches are speculative, meaning there is no guarantee that they are executed. A branch or exceptional instruction in the code stream can cause a pipeline flush, discarding the currently fetched instructions. Because of the aggressive prefetching behavior, you must not place read-sensitive devices in the same page as code. Pages with Device memory type attributes are treated as Non-Cacheable Normal Memory when accessed by instruction fetches. You must mark pages that contain read-sensitive devices with the translation table descriptor *Execute Never* (XN) attribute bit. To avoid speculative fetches to read-sensitive devices when address translation is disabled, these devices and code that are fetched must be separated in the physical memory map.

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

6.4.3 Program flow prediction

The Cortex-A73 processor 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 associated with flushing the pipeline.

To avoid this penalty, the branch prediction hardware predicts if a branch is to be taken. For conditional branches, the hardware predicts if the branch is to be taken as well as the address that the branch is to go to, known as the branch target address. For unconditional branches, only the target is predicted.

The hardware contains the following functionality:

- A BTAC holding the branch target address of previously taken branches.
- Dynamic branch predictor history stored in RAMs.
- The return stack, a stack of nested subroutine return addresses.
- A static branch predictor.
- An indirect branch predictor.

Predicted and non-predicted instructions

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

- Conditional branches.
- Unconditional branches.
- Indirect branches associated with procedure call and return instructions.
- Immediate branches, which are predicted by static prediction and BTAC.
- Branches that switch between A32 and T32 states.

Unless otherwise specified, this list applies to A64, A32, and T32 instructions.

However, some branch instructions are not predicted:

- Instructions with the S suffix are not predicted because they are typically used to return from exceptions and have side-effects that can change privilege mode and security state.
- All mode-changing instructions.

T32 state conditional branches

A T32 instruction set branch that is normally encoded as unconditional can be made conditional by inclusion in an *If-Then* (IT) block. Then it is treated as a conditional branch.

Return stack predictions

The return stack stores the address and, in AArch32, the A32 or T32 instruction set of the instruction after a procedure call type branch instruction. 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`
- `BLX` (immediate) in AArch32 state.
- `BLX` (register) in AArch32 state.
- `BLR` in AArch64 state.

In AArch32 state, the following instructions cause a return stack pop if predicted:

- `BX r14`
- `LDR pc, [r13], #imm`
- `LDM r13, {…,pc}`
- `LDM r13, {…,pc}!`
- `MOV ps, r14`
- `POP {…,pc}`

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

Because return-from-exception instructions can change Exception level, they are not predicted. This includes:

- `LDM (exception return)`
- `RFE`
- `SUBS pc, lr`
- `ERET`

Enabling program flow prediction

Program flow prediction is always enabled when the MMU is enabled by setting the M bit in the relevant system control register (SCTLR_ELn or SCTLR) to `1`.

6.5 L1 data memory system

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

——— **Note** ———

In the L1 data memory subsystems, aliases are handled in hardware and from the programmer's point of view, the data cache behaves like an eight-way set associative PIPT cache (for 32KB configurations) and a 16-way set associative PIPT cache (for 64KB configurations).

———————

This section contains the following subsections:

- *6.5.1 Data cache coherency* on page 6-347.
- *6.5.2 Data cache disabled behavior* on page 6-347.
- *6.5.3 Data cache maintenance considerations* on page 6-348.
- *6.5.4 Data cache zero* on page 6-348.
- *6.5.5 Internal exclusive monitor* on page 6-348.
- *6.5.6 ACE transactions* on page 6-349.

6.5.1 Data cache coherency

The Cortex-A73 processor uses the MOESI protocol to maintain data coherency between multiple cores.

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

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

The L1 memory subsystem stores the MOESI state of the cache line in the tag RAM.

——— **Note** ———

The names UniqueDirty, SharedDirty, UniqueClean, SharedClean, Invalid are equivalent to the AMBA names for the cache states.

———————

Data cache invalidate on reset

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

——— **Note** ———

The **DBGL1RSTDISABLE** pin must only be used in diagnostic mode as cache coherency cannot be ensured if the L1 cache is not invalidated on startup.

———————

6.5.2 Data cache disabled behavior

If the data cache is disabled, load and store instructions do not access any of the L2 or L1 data cache arrays. An exception to this rule is the CP15 data cache operations. If the data cache is disabled, the data cache maintenance operations can still execute normally.

If the data cache is disabled, all load and store instructions to cacheable memory are treated as if they were non-cacheable. This means that they are not coherent with the caches in this core or the caches in other cores, and software must take account of this.

6.5.3 Data cache maintenance considerations

DCIMVAC operations in AArch32 state are treated as DCCIMVAC.

DC IVAC operations in AArch64 state are treated as DC CIVAC except for permission checking and watchpoint matching as described in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*

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

6.5.4 Data cache zero

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

In Cortex-A73, this 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.*

6.5.5 Internal exclusive monitor

The Cortex-A73 processor L1 memory systems have an internal exclusive monitor. This is a two-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. The size of the tagged block is defined by CTR.ERG as 64 bytes, 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.

Treatment of intervening STR operations

In cases where there is an intervening store operation between an exclusive load and an exclusive store from the same core, the intermediate store does not produce any direct effect on the internal exclusive monitor. The local monitor is in the Exclusive Access state after the exclusive load, remains in the Exclusive Access state after the store, and returns to the Open Access state only after the exclusive store, a `CLREX` instruction, or an exception return.

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

6.5.6 ACE transactions

The following table shows the ACE transactions that each type of memory access generates.

Table 6-2 ACE transactions

Attributes		**ACE transaction**				
Memory type	**Shareability**	**Domain**	**Load**	**Store**	**Load exclusive**	**Store exclusive**
Device	-	System	ReadNoSnoop	WriteNoSnoop	ReadNoSnoop and **ARLOCKM** set to HIGH	WriteNoSnoop and **AWLOCKM** set to HIGH
Normal, inner Non-cacheable, outer Non-cacheable	Non-shared	System	ReadNoSnoop	WriteNoSnoop	ReadNoSnoop and **ARLOCKM** set to HIGH	WriteNoSnoop and **AWLOCKM** set to HIGH
	Inner-shared					
	Outer-shared					
Normal, inner Non-cacheable, outer Write-Back or Write-Through, or Normal, inner Write-Through, outer Write-Back, Write-Through or Non-cacheable, or Normal inner Write-Back outer Non-cacheable or Write-Through	Non-shared	System	ReadNoSnoop	WriteNoSnoop	ReadNoSnoop	ReadNoSnoop
	Inner-shared	System	ReadNoSnoop	WriteNoSnoop	ReadNoSnoop with **ARLOCKM** set to HIGH	WriteNoSnoop with **ARLOCKM** set to HIGH
	Outer-shared	System				
Normal, inner Write-Back, outer Write-Back	Outer-shared (Inner-shared)	Outer Shareable (Inner Shareable)	ReadShared for Linefills initiated by Loads/PTW/L1 prefetch/PLD. ReadClean for L2 prefetches	ReadUnique, CleanUnique, and also MakeUnique for streaming	ReadShared	CleanUnique with **ARLOCKM** set to HIGH if ECTLR.STXEN bit is set

See the *Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE, and ACE-Lite* for more information about ACE transactions.

6.6 Memory prefetching

This section describes memory prefetching.

This section contains the following subsections:

6.6.1 Preload instructions

In AArch64 state, the Cortex-A73 processor supports the `PRFM` (Prefetch Memory) instructions which signal to the memory system that memory accesses from a specified address are likely to occur in the near future. The memory system can respond by taking actions that are expected to reduce the latency of the memory access when they do occur. `PRFM` instructions perform a lookup in the cache, and start a linefill if they miss and are to a cacheable address. However, the `PRFM` instruction retires as soon as 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.

In AArch32 state or when present in a `PRFM` instruction, the memory system hint instructions `PLD` (Prefetch Data) and `PLDW` (Preload Data With Intent To Write), for cacheable data, perform a lookup in the cache and start a linefill if they miss. There are four `PLD` slots to allow further `PLD` instructions to execute if a preceding `PLD` causes a translation table walk. Any linefill started by a `PLDW` instruction causes the data to be invalidated in other cores, so that the line is ready to be written to.

The `PLI` (Preload Instruction) 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*.

6.6.2 Data prefetching and monitoring

The data cache implements an automatic prefetcher that monitors cache misses in the core.

When a pattern is detected, the automatic prefetcher starts linefills in the background. The prefetcher recognizes a sequence of data cache misses at a fixed stride pattern that lies in 32 cache lines, plus or minus. Any intervening stores or loads that hit in the data cache do not interfere with the recognition of the cache miss pattern. Up to eight independent streams can be handled by the prefetcher. This L1 prefetcher is coupled with a simpler L2 data prefetcher which receives hints from the L1 prefetcher.

The ECTLR and L2CTLR registers enable you to have some control over the prefetcher:

- The L1 stride prefetcher is enabled by setting bit [7] of the ECTLR register to `1`.
- The L2 stride prefetcher is enabled by setting bit [8] of the ECTLR register to `1`.

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

6.6.3 Non-temporal loads

In AArch64 state, cache requests made by a non-temporal load instruction (`LDNP`) are treated as normal loads.

6.7 Direct access to internal memory

The Cortex-A73 processor provides a mechanism to read the internal memory used by the L1 cache and TLB structures through IMPLEMENTATION DEFINED system registers. This functionality can be useful when investigating issues where the coherency between the data in the cache and data in system memory is broken.

When the processor is executing in AArch64 state, the appropriate memory block and location are selected using a number of write-only registers and the data is read from read-only registers as shown in the following table. These operations are available only in EL3. In all other modes, executing these instructions results in an Undefined Instruction exception.

Table 6-3 AArch64 registers used to access internal memory

Function	Access	Operation	Rd Data
Data Register 0	Read-only	`MRS <Xd>, S3_3_c15_c0_0`	Data
Data Register 1	Read-only	`MRS <Xd>, S3_3_c15_c0_1`	Data
Data Register 2	Read-only	`MRS <Xd>, S3_3_c15_c0_2`	Data
Data Register 3	Read-only	`MRS <Xd>, S3_3_c15_c0_3`	Data
Data Cache Tag Read Operation Register	Write-only	`MSR S3_3_c15_c2_0, <Xd>`	Set/Way
Instruction Cache Tag Read Operation Register	Write-only	`MSR S3_3_c15_c2_1, <Xd>`	Set/Way
Data Cache Data Read Operation Register	Write-only	`MSR S3_3_c15_c4_0, <Xd>`	Set/Way/Offset
Instruction Cache Data Read Operation Register	Write-only	`MSR S3_3_c15_c4_1, <Xd>`	Set/Way/Offset
TLB Data Read Operation Register	Write-only	`MSR S3_3_c15_c4_2, <Xd>`	Index/Way

When the core is executing in AArch32 state, the appropriate memory block and location are selected using a number of write-only CP15 registers and the data is read from read-only CP15 registers as shown in the following table. These operations are available only in EL3. In all other modes, executing the CP15 instruction results in an Undefined Instruction exception.

Note

Accessing these registers from an external debugger is only possible when EL3 is in AArch64 state.

Table 6-4 AArch32 CP15 registers used to access internal memory

Function	Access	CP15 operation	Rd Data
Data Register 0	Read-only	`MRC p15, 3, <Rd>, c15, c0, 0`	Data
Data Register 1	Read-only	`MRC p15, 3, <Rd>, c15, c0, 1`	Data
Data Register 2	Read-only	`MRC p15, 3, <Rd>, c15, c0, 2`	Data
Data Register 3	Read-only	`MRC p15, 3, <Rd>, c15, c0, 3`	Data
Data Cache Tag Read Operation Register	Write-only	`MCR p15, 3, <Rd>, c15, c2, 0`	Set/Way
Instruction Cache Tag Read Operation Register	Write-only	`MCR p15, 3, <Rd>, c15, c2, 1`	Set/Way
Data Cache Data Read Operation Register	Write-only	`MCR p15, 3, <Rd>, c15, c4, 0`	Set/Way/Offset
Instruction Cache Data Read Operation Register	Write-only	`MCR p15, 3, <Rd>, c15, c4, 1`	Set/Way/Offset
TLB Data Read Operation Register	Write-only	`MCR p15, 3, <Rd>, c15, c4, 2`	Index/Way

This section contains the following subsections:
- *6.7.1 Data cache tag and data encoding* on page 6-352.
- *6.7.2 Instruction cache tag and data encoding* on page 6-353.
- *6.7.3 TLB data encoding* on page 6-353.

6.7.1 Data cache tag and data encoding

The Cortex-A73 processor data cache consists of an 8-way or 16-way set-associative structure.

The number of sets in each way depends on the configured size of the cache. The encoding, set in Rd in the appropriate `MCR` instruction, used to locate the required cache data entry for tag and data memory is shown in the following table. It is very similar for both the tag and data RAM access. Data RAM access includes an additional field to locate the appropriate doubleword in the cache line.

Table 6-5 32KB data cache tag and data location encoding

Bit-field of Rd	Description
[31:29]	Cache way
[28:12]	Unused
[11:6]	Set index
[5:3]	Cache doubleword data offset, Data Register only
[2:0]	Unused

Table 6-6 64KB data cache tag and data location encoding

Bit-field of Rd	Description
[31:28]	Cache way
[27:12]	Unused
[11:6]	Set index
[5:3]	Cache doubleword data offset, Data Register only
[2:0]	Unused

Data cache reads return 64 bits of data in Data Register 0 and Data Register 1. The tag information, MOESI coherency state, outer attributes, and valid, for the selected cache line is returned using Data Register 0 and Data Register 1 using the format shown in the following table. The core encodes the 4-bit MOESI coherency state across two fields of Data Register 0 and Data Register 1.

Table 6-7 Data cache tag data format

Register	Bit-field	Description
Data Register 0	[31]	Partial MOESI state, Dirty
	[30]	Partial MOESI state, Exclusive
	[29]	Partial MOESI state, Valid
	[28]	Security state, NS
	[27:0]	Tag address [39:12]

Table 6-7 Data cache tag data format (continued)

Register	Bit-field	Description
Data Register 1	[31:4]	Unused
	[3]	Outer shareability
	[2:1]	Memory hint (read/write)
	[0]	Partial MOESI state, Globally shared

6.7.2 Instruction cache tag and data encoding

The Cortex-A73 processor instruction cache is significantly different from the data cache and this is shown in the encodings and data format used in the CP15 operations used to access the tag and data memories. The following table shows the encoding required to select a given cache line.

Table 6-8 Instruction cache tag and data location encoding

Bit-field of Rd	Description
[31:30]	Cache Way
[29:14]	Unused
[13:6]	Set index
[5:3]	Cache doubleword data offset, Data Register only
[2:0]	Unused

The following table shows the tag and valid bits format for the selected cache line using only Data Register 0.

Table 6-9 Instruction cache tag data format

Bits	Description
[31:30]	Unused
[29]	Valid
[28]	Non-secure state (NS)
[27:0]	Tag address

The CP15 Instruction Cache Data Read Operation returns a 64-bit entry from the cache in Data Register 0 and Data Register 1.

The CP15 Instruction Cache Tag Read Operation returns two entries from the cache in Data Register 0 and Data Register 1 corresponding to the 16-bit aligned offset in the cache line:

Data Register 0	Bits[31:0] data from cache offset + `0b000`.
Data Register 1	Bits[31:0] data from cache offset + `0b100`.

In T32 state, these two fields can represent any combination of 16-bit and partial or full 32-bit instructions.

6.7.3 TLB data encoding

The main TLB is built from two cache RAMs:

RAM0 This is a 4-way set-associative RAM array for 4KB, 16KB, and 64KB page translation entries.

RAM1 This is a 2-way set-associative RAM array for 1MB, 2MB, 16MB, 32MB, 512MB, and 1GB block sizes, walk cache, and IPA cache.

RAM0 holds regular format entries.

RAM1 holds:
- TLB regular format entries.
- TLB walk format entries.
- TLB IPA format entries.

The TLB walk format entry is identified by the Entry type field set to `1`.

The TLB IPA format entry is identified by the Entry type field set to `0` and the S1 level field set to `0b11`.

To read the individual entries into the data registers, software must write to the TLB Data Read Operation Register. The following table shows the write TLB Data Read Operation Register location encoding.

Table 6-10 TLB Data Read Operation Register location encoding

Bits	Description
[31:30][bw]	TLB way
[29:9]	Unused
[8]	Type: `0` RAM0. `1` RAM1.
[7:0][bx]	TLB index.

Both RAM0 and RAM1 are 117 bits wide. The data is returned in the data registers:

Data Register 0[31:0]	RAM data[31:0]
Data Register 1[31:0]	RAM data[63:32]
Data Register 2[31:0]	RAM data[95:64]
Data Register 3[20:0]	RAM data[116:96]

TLB regular format

The following table shows the TLB regular format encoding.

Table 6-11 TLB regular format encoding

Bits	Name	Description
[116]	Entry type	Indicates the entry type: `0` Regular page or IPA cache entry. `1` Walk cache entry.
[115]	Outer Shareable	Indicates the Outer Shareable memory attribute.

[bw] Bit [31] is unused if bit [8] = `0b1`.
[bx] Bits [7:6] are unused if bit [8] = `0b1`.

Table 6-11 TLB regular format encoding (continued)

Bits	Name	Description
[114:109]	MemAttr[7:2]	Indicates the memory attributes, which are a combination of stage 1 and stage 2 for this entry. (Inner read/write allocate hint ignored.)
[108:107]	HAP[1:0]	Indicates the hypervisor access permissions, from stage 2 translation.
[106:104]	S2 Translation Mode	Indicates the stage 2 translation mode and page size. `3b00_0` S2 off. `3b00_1` S2 LPAE 4K 1GB page. `3b10_1` S2 LPAE 4K 2MB page. `3b10_0` S2 LPAE 4K 4KB page. `3b01_1` S2 LPAE 16K 32MB page. `3b01_0` S2 LPAE 16K 16KB page. `3b11_0` S2 LPAE 64K 64KB page. `3b11_1` S2 LPAE 64K 512MB page.
[103]	PXN	Indicates the privileged execute-never attribute, from stage 1 translation.
[102]	ST1XN	Indicates the Execute-never attribute, from stage 1 translation.
[101]	ST2XN	Indicates the Execute-never attribute, from stage 2 translation.
[100:98]	AP[2:0]	Indicates the access permissions, from stage 1 translation.
[97]	Split	Asserted if the stage 1 page size is larger than stage 2 page size.
[96]	NS	Non-secure VA.
[95:68]	PA	Physical address.
[67]	Valid Entry	Indicates that the entry is valid: `0` Entry is not valid. `1` Entry is valid.
[66:34]	VA[48:16]	Indicates the virtual address.
[33:26]	VMID[7:0]	Indicates the virtual machine identifier.
[25:18]	ASID_HI[7:0]	In AArch64, indicates the most significant bits of the *Address Space Identifier* (ASID). In AArch32, VMSA short format bits [21:18] represent the domain index.
[17:10]	ASID_LO[7:0]	Indicates the least significant byte of the ASID.
[9]	nG	Indicates the Non-global bit.

Table 6-11 TLB regular format encoding (continued)

Bits	Name	Description
[8:6]	Size	Indicates the size of memory mapped by block or page allocated in the TLB entry: **In TLB RAM0:** `0b000` 4KB. `0b001` 16KB. `0b010` 64KB. **In TLB RAM1:** `0b000` 1MB. `0b001` 2MB. `0b010` 16MB. `0b011` 32MB. `0b100` 512MB. `0b101` 1GB.
[5:4]	Translation Regime	The possible values are: `0b00` In AArch64: EL0/1 NS. In AArch32: PL0/1 NS. `0b01` In AArch64: EL0/1 S. In AArch32: PL0/1 S. `0b10` In AArch64: EL2 (NS). In AArch32: PL2 (NS). `0b11` EL3 (S).

Table 6-11 TLB regular format encoding (continued)

Bits	Name	Description
[3:2]	S1 Translation Mode	Indicates the stage 1 translation mode. `0b00` VMSA short format. `0b10` LPAE 4K. `0b01` LPAE 16K. `0b11` LPAE 64K. If stage 1 is off encoding is `2'b11`
[1:0]	S1 Level	Indicates the stage 1 level that gave this translation: **VMSA** `0b00` Level 2. `0b01` Level 1. `0b10` Level 1 with contiguous hint. `0b11` Level 2 with contiguous hint. **LPAE 4K** `0b00` Level 3. `0b01` Level 2. `0b10` Level 1. `0b11` Level 3 with contiguous hint. **LPAE 16K** `0b00` Level 2. `0b01` Level 1. `0b10` Not used in regular format. `0b11` Level 2 with contiguous hint. **LPAE 64K** `0b00` Level 2. `0b01` Level 1. `0b10` Not used in regular format. `0b11` Level 2 with contiguous hint. If stage 1 is off this field is always `2'b10`.

TLB walk format

The following table shows the encoding for a translation table walk entry.

Table 6-12 TLB walk format encoding

Bits	Name	Description
[116]	Entry type	Indicates the entry type: `0` Regular page or IPA cache entry. `1` Walk cache entry.
[115]	Outer Shareable	Indicates the Outer Shareable memory attribute.
[114:109]	MemAttr[7:2]	Indicates the memory attributes, which are a combination of stage 1 and stage 2 for this entry. (Inner read/write allocate hint ignored.)
[108:107]	HAP[1:0]	Indicates the hypervisor access permissions, from stage 2 translation.
[106:104]	S2 Translation Mode	Indicates the stage 2 translation mode and page size. `3b00_0` S2 off. `3b00_1` S2 LPAE 4K 1GB page. `3b10_1` S2 LPAE 4K 2MB page. `3b10_0` S2 LPAE 4K 4KB page. `3b01_1` S2 LPAE 16K 32MB page. `3b01_0` S2 LPAE 16K 16KB page. `3b11_0` S2 LPAE 64K 64KB page. `3b11_1` S2 LPAE 64K 512MB page.
[103]	PXN	Indicates the privileged execute-never attribute, from stage 1 translation.
[102]	ST1XNTable	Indicates the Execute-never attribute, from stage 1 translation.
[101]	ST2XNTable	Indicates the Execute-never attribute, from stage 2 translation.
[100:98]	APTable[2:0]	Indicates the access permissions, from stage 1 translation.
[97]	Split	Asserted if the stage 1 page size is larger than stage 2 page size.
[96]	NS	Non-secure VA.
[95:68]	PA	Physical address.
[67]	Valid Entry	Indicates that the entry is valid: `0` Entry is not valid. `1` Entry is valid.
[66:34]	VA[48:16]	Indicates the virtual address.
[33:26]	VMID[7:0]	Indicates the virtual machine identifier.

Table 6-12 TLB walk format encoding (continued)

Bits	Name	Description
[25:18]	ASID_HI[7:0]	In AArch64, indicates the most significant bits of the *Address Space Identifier* (ASID). In AArch32 VMSA, bits [21:18] represent the domain index.
[17:10]	ASID_LO[7:0]	Indicates the least significant byte of the ASID.
[9]	nG	Indicates the Non-global bit. Always `0`.
[8:6]	Size	In this context, these bits are for the translation granule: `0b000` VMSA short format. `0b001` LPAE 4K. `0b011` LPAE 16K. `0b100` LPAE 64K.
[5:4]	Translation Regime	The possible values are: `0b00` In AArch64: EL0/1 NS. In AArch32: PL0/1 NS. `0b01` In AArch64: EL0/1 S. In AArch32: PL0/1 S. `0b10` In AArch64: EL2 (NS). In AArch32: PL2 (NS). `0b11` EL3 (S).

Table 6-12 TLB walk format encoding (continued)

Bits	Name	Description
[3:2]	S1 Translation Mode	Indicates the stage 1 translation mode. `0b00` VMSA short format. `0b10` LPAE 4K. `0b01` LPAE 16K. `0b11` LPAE 64K. If stage 1 is off encoding is `2'b11`
[1:0]	S1 Level	Indicates the stage 1 level that gave this translation: **VMSA** `0b00` Level 2. `0b01` Level 1. `0b10` Level 1 with contiguous hint. `0b11` Level 2 with contiguous hint. **LPAE 4K** `0b00` Level 3. `0b01` Level 2. `0b10` Level 1. `0b11` Level 3 with contiguous hint. **LPAE 16K** `0b00` Level 2. `0b01` Level 1. `0b10` Not used in regular format. `0b11` Level 2 with contiguous hint. **LPAE 64K** `0b00` Level 2. `0b01` Level 1. `0b10` Not used in regular format. `0b11` Level 2 with contiguous hint. If stage 1 is off this field is always `2'b10`.

TLB IPA format

The following table shows the encoding for an IPA translation entry.

Table 6-13 TLB IPA format encoding

Bits	Name	Description
[116]	Entry type	Indicates the entry type: `0` Regular page or IPA cache entry.
[115]	Outer Shareable	Indicates the Outer Shareable memory attribute.
[114:109]	MemAttr[7:2]	Indicates the memory attributes, which are a combination of stage 1 and stage 2 for this entry. (Inner read/write allocate hint ignored.)
[108:107]	HAP[1:0]	Indicates the hypervisor access permissions, from stage 2 translation.
[106:104]	S2 Translation Mode	Indicates the stage 2 translation mode and page size. `3b00_0` S2 off. `3b00_1` S2 LPAE 4K 1GB page. `3b10_1` S2 LPAE 4K 2MB page. `3b10_0` S2 LPAE 4K 4KB page. `3b01_1` S2 LPAE 16K 32MB page. `3b01_0` S2 LPAE 16K 16KB page. `3b11_0` S2 LPAE 64K 64KB page. `3b11_1` S2 LPAE 64K 512MB page.
[103]	PXN	Not applicable.
[102]	ST1XN	Not applicable.
[101]	ST2XN	Indicates the Execute-never attribute, from stage 2 translation.
[100:98]	AP[2:0]	Indicates the access permissions, from stage 1 translation.
[97]	Split	Not applicable.
[96]	NS	Non-secure IPA.
[95:68]	PA	Physical address.
[67]	Valid Entry	Indicates that the entry is valid: `0` Entry is not valid. `1` Entry is valid.
[66:58]	-	Reserved, RES0
[57:34]	IPA[39:16]	Indicates the intermediate physical address.
[33:26]	VMID[7:0]	Indicates the virtual machine identifier.
[25:18]	ASID_HI[7:0]	Not applicable.
[17:10]	ASID_LO[7:0]	Not applicable.
[9]	nG	Not applicable.

Table 6-13 TLB IPA format encoding (continued)

Bits	Name	Description
[8:6]	Size	In this context, these bits are for the translation granule: `0b001` LPAE 4K. `0b011` LPAE 16K. `0b100` LPAE 64K.
[5:4]	Translation Regime	The possible values are: `0b00` In AArch64: EL0/1 NS. In AArch32: PL0/1 NS.
[3:2]	S1 Translation Mode	Always `0b01`.
[1:0]	S1 Level	Always `0b10`.

Chapter 7
Level 2 Memory System

This chapter describes the *Level 2* (L2) memory system.

It contains the following sections:

7.1 About the L2 memory system

The L2 memory system consists of:

- An AMBA *AXI Coherency Extensions* (ACE) master interface 128-bit bus.
- An AMBA AXI4 *Accelerator Coherency Port* (ACP) slave interface 128-bit bus.
- An integrated *Snoop Control Unit* (SCU), connecting up to four cores within a cluster which:
  - — Maintains L1 data cache coherency between cores by snooping the L1 caches.
  - — Holds duplicate copies of the L1 data cache tags for efficient coherency support.
  - — Arbitrates requests from the cores and the ACP interface.

  ———— **Note** ————

  The requests from the instruction cache perform coherency requests in the L1 data cache in order to maintain coherency between the instruction cache and the data cache. However, the requests from the L1 data cache do not perform coherency requests in the L1 instruction cache.

  ————————————

- A tightly-integrated L2 cache with:
  - — A configurable size of 256KB, 512KB, 1MB, 2MB, 4MB, or 8MB.
  - — A 16-way, set-associative structure.
  - — A fixed line length of 64 bytes.
  - — Optional ECC protection.
  - — A *Physically Indexed, Physically Tagged* (PIPT) cache.
  - — A prefetcher.

The L2 memory system has an asynchronous error mechanism, see *7.7 External aborts and asynchronous errors* on page 7-374.

7.2 L2 cache

The integrated L2 cache is the Point of Unification for the Cortex-A73 processor. It handles both instruction and data requests from the I-Side and D-side of each core respectively.

Instructions are allocated to the L2 cache when fetched from the system and can be invalidated during maintenance operations.

There are 48 linefill buffers and 16 eviction buffers. All buffers handle 64-byte cache lines.

The L2 cache tags are looked up in parallel with the SCU duplicate tags. If both the L2 tag and SCU duplicate tag hit, the L1 cache data is used, because the L2 cache data might not be up to date if there has been a write since the last linefill.

L2 cache is invalidated automatically at reset unless the **L2RSTDISABLE** signal is set HIGH when the **nL2RESET** signal is deasserted.

This section contains the following subsections:
- *7.2.1 L2 ECC support* on page 7-365.
- *7.2.2 L2 cache RAM latency settings* on page 7-365.

7.2.1 L2 ECC support

The L2 cache can be configured to support ECC for the tag and data RAMs.

If implemented, *Single Error Correct, Double Error Detect* (SECDED) protection is offered with a 64-bit granularity.
- If an error is detected, the transfer is blocked and the line is invalidated from the L2 cache.
- If a single error is detected, the line is corrected and evicted to external memory if it is valid and dirty.
- If a double error is detected, an asynchronous error is generated by asserting **nECCERRIRQ**.
- After error correction has completed, access is restarted and the line is read again from external memory.
- Both single and double errors are recorded separately in the L2MERRSR, see the *4.3.86 L2 Memory Error Syndrome Register* on page 4-191 for more information.

7.2.2 L2 cache RAM latency settings

The L2 cache controller supports a range of data and tag RAM sizes. Each RAM size might have different timing requirements. To ensure that the L2 cache operates correctly with the implemented data and tag RAMs, you must set the appropriate latency values in L2CTLR. See *4.3.72 L2 Control Register* on page 4-174 for more information.

There are three latency settings which must be applied to the L2 cache data and tag RAMs:

Setup latency
The number of cycles that the driving signals are asserted before the L2 cache controller applies the rising edge of the clock to the RAM.

Write access latency
The time needed by the RAMs to internally compute the write control and data signals. It is the minimum number of cycles before the next rising clock edge is applied to the RAM, following a write access. The RAMs do not sample new data and control levels before this time has elapsed.

Read access latency
The minimum number of cycles before the next rising clock edge is applied to the RAM, following a read access. Read data is sampled in the last cycle.

7.3 Snoop Control Unit

The Cortex-A73 processor supports between one and four individual cores with L1 data cache coherency maintained by the SCU.

The SCU maintains coherency between the individual data caches in the processor using ACE modified equivalents of MOESI state.

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

Each core has a tag RAM that contains the state of the cache line. Rather than access these for each snoop request the SCU contains a set of duplicate tags that permit each coherent data request to be checked against the contents of the other caches in the cluster. The duplicate tags filter coherent requests from the system so that the cores and system can function efficiently even with a high volume of snoops from the system.

When an external snoop hits in the duplicate tags a request is made to the appropriate core.

This section contains the following subsection:
- *7.3.1 Snoop and maintenance requests* on page 7-366.

7.3.1 Snoop and maintenance requests

The SCU controls snoop and maintenance requests to the system using the external **BROADCASTINNER**, **BROADCASTOUTER**, **BROADCASTCACHEMAINT**, and **BROADCASTCACHEMAINTPOU** pins:

- When you set the **BROADCASTINNER** pin to 1, the inner shareability domain extends beyond the Cortex-A73 processor, and Inner Shareable snoop and maintenance operations are broadcast externally. When you set the **BROADCASTINNER** pin to 0, the inner shareability domain does not extend beyond the Cortex-A73 processor.
- When you set the **BROADCASTOUTER** pin to 1, the outer shareability domain extends beyond the Cortex-A73 processor, and outer shareable snoop and maintenance operations are broadcast externally. When you set the **BROADCASTOUTER** pin to 0, the outer shareability domain does not extend beyond the Cortex-A73 processor.

———— **Note** ————
- The Cortex-A73 processor does not differentiate the **BROADCASTINNER** and **BROADCASTOUTER** pins. It always broadcasts the cache maintenance operations externally when either the **BROADCASTINNER** or **BROADCASTOUTER** pins are equal to 1. That is, outer coherency setting is not supported.
- In a system that contains Cortex-A73 processors and other processors in a big.LITTLE™ configuration, you must ensure the **BROADCASTINNER** and **BROADCASTOUTER** pins on both processors are set to HIGH so that both processors are in the same Inner Shareable domain.

————————

- **BROADCASTCACHEMAINT** should always be set to 0 as the Cortex-A73 processor does not support broadcast of cache maintenance operations to downstream caches.
- If **BROADCASTCACHEMAINTPOU** is set to 1, then the cache maintenance to the PoU is broadcast to other cores in the same shareability domain. If all the cores in the system support coherency between the instruction cache and the data cache, this pin can be set to 0. The Cortex-A73 processor supports this coherency.

7.4 ACE master interface

This section describes the properties of the ACE master interface. The ACE interface to the system can be clocked at integer ratios of the **CLK** frequency.

This section contains the following subsections:

7.4.1 Memory interface attributes

The following table shows the ACE master interface attributes for the Cortex-A73 processor. The table lists the maximum possible values for the read and write issuing capabilities if the Cortex-A73 processor includes four cores, where *n* is the number of cores implemented, in the range 1 to 4.

Table 7-1 ACE master interface attributes

Attribute	Value	Comments
Write issuing capability	7n + 24	Each core can perform: • 7 Device write accesses. • 8 Non-cacheable write accesses. The whole cluster capability is: • 7n Device write accesses. • 8 Non-cacheable write accesses. • 16 cacheable write accesses.
Read issuing capability	14n + 48	Each core can perform: • 12 Strongly-ordered / Device data reads • 1 TLB Non-cacheable read • 1 Instruction Non-cacheable read The whole cluster capability is: • 48 cacheable reads.
Exclusive thread capability	n	Each core can have one exclusive access sequence in progress.
Write ID capability	11n + 16	The Write ID capability is made up of: • 1 for Device nGnRnE for each core • 1 for Device nGnRE write for each core. • 1 for exclusive Device or Non-cacheable for each core • 8 for Non-cacheable writes for each core. • 16 for cacheable L2 evictions.
Write ID width	8	The ID encodes the source of the memory transaction. See, *7.4.8 AXI ID core source encoding* on page 7-369

Table 7-1 ACE master interface attributes (continued)

Attribute	Value	Comments
Read ID capability	13n + 65	The Read ID capability is made up of: • 1 for Device or Non-cacheable data reads for each core. • 1 for Non-cacheable instruction reads for each core. • 1 for TLB Non-cacheable reads for each core. • 1 for DVM operation. • 10 for CleanUnique operation for each core. • 16 for MakeUnique operation that is used for the streaming store. • 48 for cacheable reads.
Read ID width	8	The ID encodes the source of the memory transaction. See, *7.4.8 AXI ID core source encoding* on page 7-369

7.4.2 ACE transfers

The Cortex-A73 processor does not generate any FIXED bursts and all WRAP bursts fetch a complete cache line starting with the critical word first. A burst does not cross a cache line boundary.

The cache linefill fetch length is always 64 bytes.

The Cortex-A73 processor generates only a subset of all possible AXI transactions on the master interface.

For Write-back (inner cacheable) transfers the supported transfers are:

- WRAP 4 128-bit for read transfers.
- INCR 4 128-bit for write transfers.

For Non-cacheable transactions:

- INCR N (N:1-4) 128-bit for write transfers.
- INCR 1 64-bit for read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, 128-bit for exclusive write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, 128-bit for exclusive read transfers.
- WRAP 4 128-bit for read transfers.

For Device transactions:

- INCR 1 8-bit, 16-bit, 32-bit, and 64-bit read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, and 64-bit write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, 128-bit exclusive read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, 128-bit exclusive write transfers.

For translation table walk transactions:

- INCR 1 64-bit read transfers.

The following points apply to AXI transactions:

- WRAP bursts are only 128-bit.
- INCR 1 can be any size up to 128-bits for read or write.
- INCR burst, more than one transfer, are only 128-bit.
- No transaction is marked as FIXED.
- Write transfers with none or some byte strobes LOW can occur.

7.4.3 Read response

The ACE master can delay accepting a read data channel transfer by holding **RREADY** LOW for an indeterminate number of cycles. **RREADY** can be deasserted LOW between read data channel transfers that form part of the same transaction.

7.4.4 Write response

The ACE master requires that the slave does not return a write response until it has received the write address.

7.4.5 Barriers

The Cortex-A73 processor never broadcasts barrier transactions to the interconnect.

They are always terminated within the cluster and therefore, **SYSBARDISABLE** must be set to HIGH.

When you terminate barriers within the cluster, ensure that your interconnect and any peripherals connected to it do not return a write response for a transaction until that transaction is considered complete by a later barrier. This means that the write must be observable to all other masters in the system. Arm expects the majority of peripherals to meet this requirement.

7.4.6 AXI3 compatibility mode

The Cortex-A73 processor implements an AXI3 compatibility mode that enables you to use the processor in a standalone environment where the AMBA 4 ACE interface is not required and the core does not propagate barriers outside of the cluster.

To enable this mode you must set the **SYSBARDISABLE** input pin to HIGH on the boundary of the core. You must also ensure that the **BROADCASTINNER**, **BROADCASTOUTER**, and **BROADCASTCACHEMAINT** input pins are set to LOW.

7.4.7 AXI privilege information

AXI provides information about the privilege level of an access on the **ARPROTM[2:0]** and **AWPROTM[2:0]** signals.

However, when accesses might be cached or merged together, the resulting transaction can have both privileged and unprivileged data combined. If this happens, the Cortex-A73 processor marks the transaction as privileged, even if it was initiated by an unprivileged process.

7.4.8 AXI ID core source encoding

When reading or writing to the memory-mapped interface of an external component, the component may need to determine which core is making the access. Cortex-A73 encodes the source core number in the ARIDM[6:5] and AWIDM[6:5] signals for Non-cacheable and Device transactions.

7.5 Additional memory attributes

The Cortex-A73 processor 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.

Note

When MMU stage 1 translation is disabled and the instruction cache is enabled, memory marked as Outer Write-Through/Inner Write-Through is cached in the L1 instruction cache but not in the L2 cache.

The attributes provided on **ARCACHE** or **AWCACHE** in ACE configurations are these downgraded attributes and indicate how the interconnect must treat the transaction.

Some interconnects or bus protocols might require more information about the memory type and, for these cases, the cluster exports the unaltered memory attribute information from the translation tables stored in the TLB. These signals are for information only, and do not form part of the ACE protocol.

There is a **RDMEMATTR** bus for the read channel and a **WRMEMATTR** bus for the write channel.

The following table describes the encodings on the memory attribute bus.

Table 7-2 Memory attribute bus encodings

Bits	Encoding
[7]	Outer shareable. Always set for device memory or memory that is both inner and outer non-cacheable.
[6:3]	Outer memory type, or device type. If bits[1:0] indicate Device, then: • Bit[3] is ignored. • Bits[6:4] are: `0b000` nGnRnE. `0b010` nGnRE. `0b100` nGRE. `0b110` GRE. If bits[1:0] indicate Normal, then: `0b0100` NC. `0b10RW` WT. `0b11RW` WB. Where `R` is read allocate hint, `W` is write allocate hint.

Table 7-2 Memory attribute bus encodings (continued)

Bits	Encoding
[2]	Inner shareable. Anything with bit[7] set must also have bit[2] set.
[1:0]	Inner memory type: `0b00` Device. `0b01` NC. `0b10` WT. `0b11` WB.

7.6 ACP

The *Accelerator Coherency Port* (ACP) is implemented as an AXI4 slave interface with 128-bit read and write buses.

This section contains the following subsections:
- *7.6.1 ACP interface restrictions* on page 7-372.
- *7.6.2 ACP requests* on page 7-372.

7.6.1 ACP interface restrictions

The requirements of the ACP interface are:

- All transactions have burst size equal to the data bus width, **AxSIZE** = `0b100`.
- All transaction burst addresses must be aligned to the burst length.
- All transactions have burst type INCR, **AxBURST** = `0b01`.
- Exclusive transactions are not supported, **AxLOCK** = `0b0`.
- All transactions are permitted to be Secure or Non-secure, **AxPROT** can take any value.
- *Quality of Service* (QoS) is not supported, **AxQOS** is not present and is effectively `0b0000`.
- Barrier transactions are not supported, **AxBAR** is not present and is effectively `0b00`.
- Multiple address region signaling is not supported, **AxREGION** is not present and is effectively `0b0000`.
- Cache maintenance transactions are not supported, **AxSNOOP** is not present and is effectively `0b00`.
- A transaction ID width of up to 5 bits is supported.
- Write interleaving is not supported.
- All bursts must be aligned and two burst lengths are supported:
  - 16 byte burst, **AxLEN** = `0b0000000` and **AxADDR[3:0]** = `0b0000`.
  - 64 byte burst, **AxLEN** = `0b0000011` and **AxADDR[5:0]** = `0b000000`.
- For write bursts:
  - 16-byte write bursts are permitted to have any combination of valid bytes. However, write bursts with no valid bytes are not supported.
  - 64-byte write bursts must have all bytes valid.

Note

- The ACP does not support non-cacheable transactions.

7.6.2 ACP requests

ARCACHE and AWCACHE are restricted to Normal, Write-Back, Read-Write-Allocate, Read-Allocate, Write-Allocate memory.

ARCACHE and AWCACHE are limited to the values `0b0111`, `0b1011`, and `0b1111`. Other values cause an SLVERR response on RRESP or BRESP.

Requests are treated as coherent requests.

ACP coherent read requests

In this case, the SCU enforces coherency.

When the data is present in one of the cores within the multiprocessor, the data is read directly from the relevant core and returned to the ACP.

When the data is not present in any of the cores, the read request is issued to the L2 cache. If data is present in the L2 cache, it is returned to the ACP. If the data is not present, the L2 cache performs a linefill before returning the data.

ACP coherent write requests

In this case, the SCU enforces coherency.

When the data is present in one of the cores within the multiprocessor the data is first cleaned and invalidated from the relevant core.

When the data is not present in any of the cores, or when it has been cleaned and invalidated, the write request is issued to the L2 cache. In the case of an L2 cache hit, the line is updated. On a miss, a line is allocated in the L2 cache.

7.7 External aborts and asynchronous errors

This section describes external aborts and asynchronous errors.

This section contains the following subsections:
- *7.7.1 External aborts* on page 7-374.
- *7.7.2 Asynchronous errors* on page 7-374.

7.7.1 External aborts

The L2 memory system handles external aborts depending on the instruction executed and attributes of the memory region of the access:

- External aborts on writes to Non-cacheable Normal memory or Device memory are indicated by asserting the **nAXIERRIRQ** signal.
- External aborts on evictions from the caches are indicated by asserting the **nAXIERRIRQ** signal.
- All other external aborts are reported to the core as asynchronous imprecise aborts.

7.7.2 Asynchronous errors

The L2 memory system has two outputs that indicate asynchronous error conditions. An asynchronous external error condition exists when either:

- The **nAXIERRIRQ** output is asserted LOW.
- The **nECCERRIRQ** output is asserted LOW.

If an asynchronous error condition is detected, the corresponding bit in the L2 Extended Control Register is asserted. The asynchronous error condition can be cleared by writing `0` to the corresponding bit of the L2ECTLR. Software can only clear the L2ECTLR. Any attempt to assert the error by writing the L2ECTLR is ignored. See *4.3.73 L2 Extended Control Register* on page 4-177 for more information.

External errors on the ACE write response channel or on the ACE read data channel, associated with a write-allocate memory transaction, cause the **nAXIERRIRQ** signal to be asserted LOW.

Double-bit ECC errors cause the **nECCERRIRQ** signal to be asserted LOW.

Any external error associated with a load instruction is reported back to the requester along with an error response and this might trigger an abort.

Chapter 8
Generic Interrupt Controller CPU Interface

This chapter describes the Cortex-A73 processor implementation of the Arm *Generic Interrupt Controller* (GIC) CPU interface.

It contains the following sections:

8.1 About the Generic Interrupt Controller CPU Interface

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

It provides:

- Registers for managing:
  - Interrupt sources.
  - Interrupt behavior.
  - Interrupt routing to one or more cores.

The Cortex-A73 processor implements the GIC CPU interface as described in the Generic Interrupt Controller (GICv4) architecture. This interfaces with an external GICv3 or GICv4 interrupt distributor component within the system.

The GICv4 architecture supports:

- Two security states.
- Interrupt virtualization.
- *Software-generated Interrupts* (SGIs).
- Message Based Interrupts.
- System register access for the CPU interface.
- Memory mapped register access for distributor registers, as well as CPU registers when using the GIC configured in legacy mode.
- 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 processor using either the IRQ or the FIQ exception request.
- Signaling Group 0 interrupts to the target processor 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-A73 processor implementation.

This section contains the following subsection:
- *8.1.1 Bypassing the CPU Interface* on page 8-376.

8.1.1 Bypassing the CPU Interface

The GIC CPU Interface is always implemented within the Cortex-A73 processor.

However, you can disable it if you assert the **GICCDISABLE** signal HIGH at reset. 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. If you disable the GIC CPU interface, the input pins **nVIRQ** and **nVFIQ** can be driven by an external GIC in the SoC.

You can disable the CPU Interface, when the Cortex-A73 processor is not being integrated with an external GICv3 or GICv4 distributor component in the system, by asserting the **GICCDISABLE** signal HIGH at reset.

Asserting the **GICCDISABLE** signal HIGH at reset removes access to the memory-mapped and system GIC CPU Interface registers.

8.2 GIC CPU interface programmers model

This section describes the GIC CPU interface programmers model for the Cortex-A73 processor.

This section contains the following subsections:

8.2.1 Memory map

The Cortex-A73 processor GIC CPU Interface implements a memory-mapped interface. The memory-mapped interface is offset from **PERIPHBASE**. The following table lists the address ranges.

Table 8-1 Memory Map

Address range	Functional block
`0x00000-0x01FFF`	CPU Interface
`0x02000-0x0FFFF`	Reserved
`0x10000-0x10FFF`	Virtual Interface Control
`0x11000-0x1FFFF`	Reserved
`0x20000-0x21FFF`	Virtual CPU Interface
`0x22000-0x2EFFF`	Reserved
`0x2F000-0x30FFF`	Alias of Virtual CPU Interface
`0x31000-0x3FFFF`	Reserved

Note

These registers are not available if **GICCDISABLE** is asserted.

8.2.2 CPU interface register summary

Each CPU interface block provides the interface for a Cortex-A73 processor that interfaces with a GIC distributor within the system. Each CPU interface provides a programming interface for:

- Enabling the signaling of interrupt requests by the CPU interface.
- Acknowledging an interrupt.
- Indicating completion of the processing of an interrupt.
- Setting an interrupt priority mask for the core.
- Defining the preemption policy for the core.
- Determining the highest priority pending interrupt for the core.
- Generating SGIs.

For more information on the CPU interface, see the *Arm® Generic Interrupt Controller Architecture Specification*.

The following table lists the registers for the CPU interface, all of which are word-accessible. Registers not described in this table are RES0. See the *Arm® Generic Interrupt Controller Architecture Specification* for more information.

Table 8-2 CPU interface register summary

Offset	Name	Type	Reset	Description
`0x0000`	GICC_CTLR	RW	`0x00000000`	CPU Interface Control Register
`0x0004`	GICC_PMR	RW	`0x00000000`	Interrupt Priority Mask Register
`0x0008`	GICC_BPR	RW	`0x00000002` (S)[by] `0x00000003` (NS)[bz]	Binary Point Register
`0x000C`	GICC_IAR	RO	-	Interrupt Acknowledge Register
`0x0010`	GICC_EOIR	WO	-	End Of Interrupt Register
`0x0014`	GICC_RPR	RO	`0x000000FF`	Running Priority Register
`0x0018`	GICC_HPPIR	RO	`0x000003FF`	Highest Priority Pending Interrupt Register
`0x001C`	GICC_ABPR	RW	`0x00000003`	Aliased Binary Point Register
`0x0020`	GICC_AIAR	RO	-	Aliased Interrupt Acknowledge Register
`0x0024`	GICC_AEOIR	WO	-	Aliased End of Interrupt Register
`0x0028`	GICC_AHPPIR	RO	`0x000003FF`	Aliased Highest Priority Pending Interrupt Register
`0x00D0`	GICC_APR0	RW	`0x00000000`	*Active Priority Register* on page 8-380
`0x00E0`	GICC_NSAPR0	RW	`0x00000000`	Non-secure Active Priority Register
`0x00FC`	GICC_IIDR	RO	`0x94043B`	*CPU Interface Identification Register* on page 8-380
`0x1000`	GICC_DIR	WO	-	Deactivate Interrupt Register

The following table shows the System register map for the CPU interface in AArch32. See the *Arm® Generic Interrupt Controller Architecture Specification, GICv3* for more information about the registers.

Table 8-3 AArch32 GIC CPU interface System register summary

Name	CRn	op1	CRm	op2	Type	Description
ICC_PMR	c4	0	c6	0	RW	Priority Mask Register
ICC_IAR0	c12	0	c8	0	RO	Group0 Interrupt Acknowledge Register
ICC_EOIR0				1	WO	Group0 End of Interrupt Register
ICC_HPPIR0				2	RO	Group0 Highest Priority Pending Interrupt Register
ICC_BPR0				3	RW	Group0 Binary Pointer Register
ICC_AP0R0				4	RW	Active Priorities 0 Register 0
ICC_AP1R0			c9	0	RW	Active Priorities 1 Register 0
ICC_DIR			c11	1	WO	Deactivate Register
ICC_RPR				3	RO	Running Priority Register

[by] S = Secure.
[bz] NS = Non-secure.

Table 8-3 AArch32 GIC CPU interface System register summary (continued)

Name	CRn	op1	CRm	op2	Type	Description
ICC_IAR1			c12	0	RO	Group1 Interrupt Acknowledge Register
ICC_EOIR1				1	WO	Group1 End of Interrupt Register
ICC_HPPIR1				2	RO	Group1 Highest Priority Pending Interrupt Register
ICC_BPR1				3	RW B[ca]	Group1 Binary Pointer Register
ICC_CTLR				4	RW B	Control Register
ICC_SRE				5	RW B	System Register Enable
ICC_IGRPEN0				6	RW	Group0 Interrupt Group Enable
ICC_IGRPEN1				7	RW B	Group1 Interrupt Group Enable
ICC_SGI1R[cb]				-	WO	Group1 Software Generated Interrupt Register
ICC_ASGI1R		0	c12	-	WO	Aliased Group1 Software Generated Interrupt Register
ICC_SGI0R		2	c12	-	WO	Group0 Software Generated Interrupt Register
ICC_MCTLR		6	c12	4	RW	Monitor Control Register
ICC_MSRE				5	RW	Monitor System Register Enable
ICC_MGRPEN1				7	RW	Monitor Group1 Interrupt Group Enable

The following table shows the System register map for the GIC CPU interface in AArch64. See the *Arm® Generic Interrupt Controller Architecture Specification, GICv3* for more information about the registers.

Table 8-4 AArch64 GIC CPU interface System register summary

Name	Type	Description
ICC_PMR_EL1	RW	Priority Mask Register
ICC_IAR0_EL1	RO	Group0 Interrupt Acknowledge Register
ICC_EOIR0_EL1	WO	Group0 End of Interrupt Register
ICC_HPPIR0_EL1	RO	Group0 Highest Priority Pending Interrupt Register
ICC_BPR0_EL1	RW	Group0 Binary Pointer Register
ICC_AP0R0_EL1	RW	Active Priorities 0 Register 0
ICC_AP1R0_EL1	RW	Active Priorities 1 Register 0
ICC_DIR_EL1	WO	Deactivate Register
ICC_RPR_EL1	RO	Running Priority Register
ICC_SGI1R_EL1	WO	Group1 Software Generated Interrupt Register
ICC_ASGI1R_EL1	WO	Aliased Group1 Software Generated Interrupt Register
ICC_SGI0R_EL1	WO	Group0 Software Generated Interrupt Register
ICC_IAR1_EL1	RO	Group1 Interrupt Acknowledge Register
ICC_EOIR1_EL1	WO	Group1 End of Interrupt Register

[ca] When operating in EL3, accesses to Banked EL1 registers access the copy designated by the current value of the SCR_EL3.NS. When EL3 is using AArch32, there is no Secure EL1 interrupt regime and accesses in any Secure EL3 mode, except Monitor mode, access the Secure copy.
[cb] Use `MCRR` instructions to access this register in AArch32 state.

Table 8-4 AArch64 GIC CPU interface System register summary (continued)

Name	Type	Description
ICC_HPPIR1_EL1	RO	Group1 Highest Priority Pending Interrupt Register
ICC_BPR1_EL1	RW B[cc]	Group1 Binary Pointer Register
ICC_CTLR_EL1	RW B	Control Register
ICC_SRE_EL1	RW B	System Register Enable
ICC_IGRPEN0_EL1	RW	Group0 Interrupt Group Enable Register
ICC_IGRPEN1_EL1	RW B	Group1 Interrupt Group Enable
ICC_CTLR_EL3	RW	EL3 Control Register
ICC_SRE_EL3	RW	EL3 System Register Enable
ICC_GRPEN1_EL3	RW	EL3 Group1 Interrupt Group Enable

8.2.3 CPU interface register descriptions

This section describes only registers whose implementation is specific to the Cortex-A73 processor.

All other registers are described in the *Arm® Generic Interrupt Controller Architecture Specification*.

Active Priority Register

The GICC_APR0 characteristics are:

Purpose	Provides support for preserving and restoring state in power management applications.
Usage constraints	This register is banked to provide Secure and Non-secure copies. This ensures that Non-secure accesses do not interfere with Secure operation.
Configurations	Available in all configurations.
Attributes	See the register summary in the following table.

The Cortex-A73 processor implements the GICC_APR0 according to the recommendations described in the *Arm® Generic Interrupt Controller Architecture Specification*.

The following table shows the Cortex-A73 MPCore GICC_APR0 implementation.

Table 8-5 Active Priority Register implementation

Number of group priority bits	Preemption levels	Minimum legal value of Secure GICC_BPR	Minimum legal value of Non-secure GICC_BPR	Active Priority Registers implemented	View of Active Priority Registers for Non-secure accesses
5	32	2	3	GICC_APR0[31:0]	GICC_NSAPR0[31:16] appears as GICC_APR0[15:0]

CPU Interface Identification Register

The GICC_IIDR characteristics are:

Purpose	Provides information about the implementer and revision of the CPU interface.
Usage constraints	There are no usage constraints.

[cc] When operating in EL3, accesses to Banked EL1 registers access the copy designated by the current value of the SCR_EL3.NS. When EL3 is using AArch32, there is no Secure EL1 interrupt regime and accesses in any Secure EL3 mode, except Monitor mode, access the Secure copy.

Configurations	Available in all configurations.
Attributes	GICC_IIDR is a 32-bit register.

The following figure shows the GICC_IIDR bit assignments.

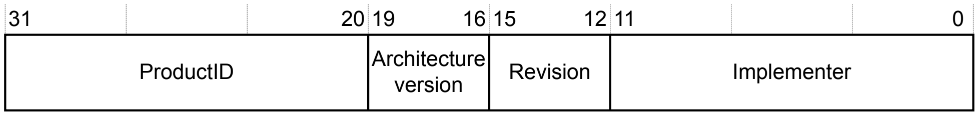

Figure 8-1 GICC_IIDR bit assignments

The following table shows the GICC_IIDR bit assignments.

Table 8-6 GICC_IIDR bit assignments

Bit	Name	Function
[31:20]	ProductID	Identifies the product: `0x9` Cortex-A73 processor.
[19:16]	Architecture version	Identifies the architecture version of the GICCPU Interface: `0x4` GICv4.
[15:12]	Revision	Identifies the revision number for the CPU interface: `0x0` r0p0.
[11:0]	Implementer	Contains the JEP106 code of the company that implements the CPU interface. For an Arm implementation, these values are: Bits[11:8] = `0x4` The JEP106 continuation code of the implementer. **Bit[7]** Always 0. Bits[6:0] = `0x3B` The JEP106 identity code of the implementer.

8.2.4 Virtual interface control register summary

The virtual interface control registers are management registers. Configuration software on the Cortex-A73 processor must ensure they are accessible only by a hypervisor, or similar software.

The following table describes the registers for the virtual interface control registers.

All the registers in the following table are word-accessible. Registers not described in this table are RES0. See the *Arm® Generic Interrupt Controller Architecture Specification* for more information.

Table 8-7 Virtual interface control register summary

Offset	Name	Type	Reset	Description
`0x000`	GICH_HCR	RW	`0x00000000`	Hypervisor Control Register
`0x004`	GICH_VTR	RO	`0x90000003`	*VGIC Type Register* on page 8-383
`0x008`	GICH_VMCR	RW	`0x004C0000`	Virtual Machine Control Register
`0x010`	GICH_MISR	RO	`0x00000000`	Maintenance Interrupt Status Register
`0x020`	GICH_EISR0	RO	`0x00000000`	End of Interrupt Status Registers

Table 8-7 Virtual interface control register summary (continued)

Offset	Name	Type	Reset	Description
0x030	GICH_ELRSR0	RO	0x0000000F	Empty List Register Status Registers
0x0F0	GICH_APR0	RW	0x00000000	Active Priorities Register
0x100	GICH_LR0	RW	0x00000000	List Register 0
0x104	GICH_LR1	RW	0x00000000	List Register 1
0x108	GICH_LR2	RW	0x00000000	List Register 2
0x10C	GICH_LR3	RW	0x00000000	List Register 3

The following table shows the register map for the AArch32 virtual interface System registers. The offsets in this table are relative to the virtual interface control registers block base address as shown in *Table 8-1 Memory Map* on page 8-377.

All the registers in the following table are word-accessible. Registers not described in this table are Reserved.

Table 8-8 AArch32 virtual interface System register summary

Name	CRn	op1	CRm	op2	Type	Description
ICH_APR0	c12	4	c8	0	RW	Hypervisor Active Priority Register 0
ICH_APR1			c9	0	RW	Hypervisor Active Priority Register 1
ICH_VSEIR				4	RW	Virtual System Error Interrupt Register
ICH_SRE				5	RW	Hypervisor System Register
ICH_HCR		4	c11	0	RW	Hypervisor Control Register
ICH_VTR				1	RO	VGIC Type Register
ICH_MISR				2	RO	Maintenance Interrupt Status Register
ICH_EISR				3	RO	End of Interrupt Status Register
ICH_ELRSR				5	RO	Empty List Register Status Register
ICH_VMCR				7	RW	Virtual Machine Control Register
ICH_LR0			c12	0	RW	List Register 0 to 3
ICH_LR1				1	RW	
ICH_LR2				2	RW	
ICH_LR3				3	RW	
ICH_LRC0			c14	0	RW	List Register Extension 0 to 3
ICH_LRC1				1	RW	
ICH_LRC2				2	RW	
ICH_LRC3				3	RW	

The following table shows the register map for the AArch64 virtual interface System registers. The offsets in this table are relative to the virtual interface control registers block base address as shown in *Table 8-1 Memory Map* on page 8-377.

All the registers in the following table are word-accessible. Registers not described in this table are Reserved.

Table 8-9 AArch64 virtual interface System register summary

Name	Type	Description
ICH_APR0_EL2	RW	Hypervisor Active Priority Register
ICH_VSEIR_EL2	RW	Virtual System Error Interrupt Register
ICH_HCR_EL2	RW	Hypervisor Control Register
ICH_VTR_EL2	RO	VGIC Type Register
ICC_SRE_EL2	RW	Hypervisor System Register Enable
ICH_MISR_EL2	RO	Maintenance Interrupt Status Register
ICH_EISR_EL2	RO	End of Interrupt Status Register
ICH_ELRSR_EL2	RO	Empty List Register Status Register
ICH_VMCR_EL2	RW	Virtual Machine Control Register
ICH_LR0_EL2	RW	List Register 0
ICH_LR1_EL2	RW	List Register 1
ICH_LR2_EL2	RW	List Register 2
ICH_LR3_EL2	RW	List Register 3

8.2.5 Virtual interface control register descriptions

This section describes only registers whose implementation is specific to the Cortex-A73 processor.

All other registers are described in the *Arm® Generic Interrupt Controller Architecture Specification*.

VGIC Type Register

The GICH_VTR characteristics are:

Purpose	Holds information on number of priority levels, number of preemption bits, and number of List registers implemented.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	GICH_VTR is a 32-bit register.

The following figure shows the GICH_VTR bit assignments.

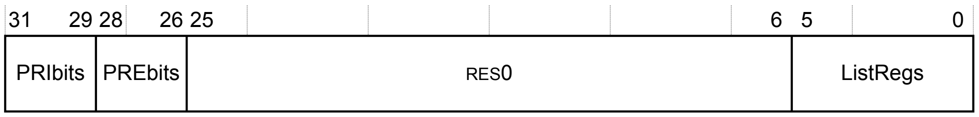

Figure 8-2 GICH_VTR bit assignments

The following table shows the GICH_VTR bit assignments.

Table 8-10 GICH_VTR bit assignments

Bit	Name	Description
[31:29]	PRIbits	Indicates the number of priority bits implemented, minus one: `0x4` Five bits of priority and 32 priority levels.
[28:26]	PREbits	Indicates the number of preemption bits implemented, minus one: `0x4` Five bits of preemption and 32 preemption levels.
[25:6]	-	Reserved, RES0.
[5:0]	ListRegs	Indicates the number of implemented List registers, minus one: `0x3` Four List registers.

8.2.6 Virtual CPU interface register summary

The virtual CPU interface forwards virtual interrupts to a connected Cortex-A73 processor, subject to the normal GIC handling and prioritization rules. The virtual interface control registers control virtual CPU interface operation, and in particular, the virtual CPU interface uses the contents of the List registers to determine when to signal virtual interrupts. When a core accesses the virtual CPU interface, the List registers are updated.

For more information on the virtual CPU interface, see the *Arm® Generic Interrupt Controller Architecture Specification*. The following table describes the registers for the virtual CPU interface.

All the registers in the following table are word-accessible. Registers not described in this table are RES0. See the *Arm® Generic Interrupt Controller Architecture Specification* for more information.

Table 8-11 Virtual CPU interface register summary

Name	Type	Reset	Description
GICV_CTLR	RW	`0x00000000`	VM Control Register
GICV_PMR	RW	`0x00000000`	VM Priority Mask Register
GICV_BPR	RW	`0x00000002`	VM Binary Point Register
GICV_IAR	RO	-	VM Interrupt Acknowledge Register
GICV_EOIR	WO	-	VM End Of Interrupt Register
GICV_RPR	RO	`0x000000FF`	VM Running Priority Register
GICV_HPPIR	RO	`0x000003FF`	VM Highest Priority Pending Interrupt Register
GICV_ABPR	RW	`0x00000003`	VM Aliased Binary Point Register
GICV_AIAR	RO	-	VM Aliased Interrupt Acknowledge Register
GICV_AEOIR	WO	-	VM Aliased End of Interrupt Register
GICV_AHPPIR	RO	`0x000003FF`	VM Aliased Highest Priority Pending Interrupt Register
GICV_APR0	RW	`0x00000000`	*VM Active Priority Register* on page 8-385
GICV_IIDR	RO	`0x0094443B`	*VM CPU Interface Identification Register* on page 8-385
GICV_DIR	WO	-	VM Deactivate Interrupt Register

8.2.7 Virtual CPU interface register descriptions

This section describes only registers whose implementation is specific to the Cortex-A73 processor.

All other registers are described in the *Arm® Generic Interrupt Controller Architecture Specification*.

VM Active Priority Register

The GICV_APR0 characteristics are:

Purpose	For software compatibility, this register is present in the virtual CPU interface. However, in a virtualized system, it is not used when preserving and restoring state.
Usage constraints	Reading the content of this register and then writing the same values must not change any state because there is no requirement to preserve and restore state during a powerdown.
Configurations	Available in all configurations.
Attributes	GICV_APR0 is a 32-bit register.

The Cortex-A73 processor implements the GICV_APR0 as an alias of GICH_APR0.

VM CPU Interface Identification Register

The GICV_IIDR characteristics are:

Purpose	Provides information about the implementer and revision of the virtual CPU interface.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	GICV_IIDR is a 32-bit register.

The bit assignments for the VM CPU Interface Identification Register are identical to the corresponding register in the CPU interface, see *CPU Interface Identification Register* on page 8-380.

Chapter 9
Generic Timer

This chapter describes the Generic Timer for the Cortex-A73 processor.

It contains the following sections:

9.1 About the Generic Timer

The Generic Timer can schedule events and trigger interrupts based on an incrementing counter value. It provides:

- Generation of timer events as interrupt outputs.
- Generation of event streams.

The Cortex-A73 Generic Timer is compliant with the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

This chapter describes only features that are specific to the Cortex-A73 processor implementation.

9.2 Generic Timer functional description

The Cortex-A73 processor provides a set of timer registers within each core of the cluster. The timers are:

- An EL1 Non-secure physical timer.
- An EL1 Secure physical timer.
- An EL2 physical timer.
- A virtual timer.

The Cortex-A73 processor does not include the system counter. This resides in the SoC. The system counter value is distributed to the Cortex-A73 processor with a synchronous binary encoded 64-bit bus, **CNTVALUEB[63:0]**.

Because **CNTVALUEB[63:0]** is generated from a system counter that typically operates at a slower frequency than **CLK**, the **CNTCLKEN** input is provided as a clock enable for the **CNTVALUEB[63:0]** bus. **CNTCLKEN** is registered inside the Cortex-A73 processor before being used as a clock enable for the **CNTVALUEB[63:0]** registers. This allows a multicycle path to be applied to the **CNTVALUEB[63:0]** bus. The following figure shows the interface.

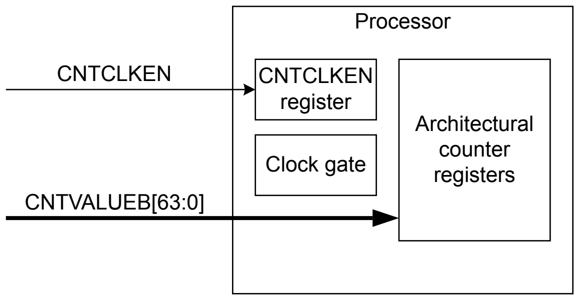

Figure 9-1 Architectural counter interface

The value on the **CNTVALUEB[63:0]** bus is required to be stable whenever the internally registered version of the **CNTCLKEN** clock enable is asserted. **CNTCLKEN** must be synchronous and balanced with **CLK** and must toggle at integer ratios of **CLK**.

See *2.3.1 Clocks* on page 2-33 for more information about **CNTCLKEN**.

Each timer provides an active-LOW interrupt output to the SoC.

The following table shows the signals that are the external interrupt output pins.

Table 9-1 Generic Timer signals

Signal[cd]	Description
nCNTPNSIRQ[n:0]	EL1 Non-secure physical timer event
nCNTPSIRQ[n:0]	EL1 Secure physical timer event
nCNTHPIRQ[n:0]	EL2 physical timer event
nCNTVIRQ[n:0]	Virtual timer event

[cd] **n** is the number of cores present in the cluster, minus one.

9.3 Generic Timer register summary

A set of Generic Timer registers are allocated within each core. The Generic Timer registers are either 32 bits wide or 64 bits wide and are accessible in the AArch32 and AArch64 execution states.

This section contains the following subsections:

- *9.3.1 AArch64 Generic Timer register summary* on page 9-389.
- *9.3.2 AArch32 Generic Timer register summary* on page 9-390.

9.3.1 AArch64 Generic Timer register summary

The following table shows the AArch64 Generic Timer registers.

Table 9-2 AArch64 Generic Timer registers

Name	Reset	Width	Description
CNTKCTL_EL1	-[ce]	32-bit	Counter-timer Kernel Control register
CNTFRQ_EL0	UNK	32-bit	Counter-timer Frequency register
CNTPCT_EL0	UNK	64-bit	Counter-timer Physical Count register
CNTVCT_EL0	UNK	64-bit	Counter-timer Virtual Count register
CNTP_TVAL_EL0	UNK	32-bit	Counter-timer Physical Timer TimerValue register
CNTP_CTL_EL0	-[cf]	32-bit	Counter-timer Physical Timer Control register
CNTP_CVAL_EL0	UNK	64-bit	Counter-timer Physical Timer CompareValue register
CNTV_TVAL_EL0	UNK	32-bit	Counter-timer Virtual Timer TimerValue register
CNTV_CTL_EL0	-[cf]	32-bit	Counter-timer Virtual Timer Control register
CNTV_CVAL_EL0	UNK	64-bit	Counter-timer Virtual Timer CompareValue register
CNTVOFF_EL2	UNK	64-bit	Counter-timer Virtual Offset register
CNTHCTL_EL2	-[cg]	32-bit	Counter-timer Hypervisor Control register
CNTHP_TVAL_EL2	UNK	32-bit	Counter-timer Hypervisor Physical Timer TimerValue register
CNTHP_CTL_EL2	-[cf]	32-bit	Counter-timer Hypervisor Physical Timer Control register
CNTHP_CVAL_EL2	UNK	64-bit	Counter-timer Hypervisor Physical Timer CompareValue register
CNTPS_TVAL_EL1	UNK	32-bit	Counter-timer Physical Secure Timer TimerValue register
CNTPS_CTL_EL1	-[cf]	32-bit	Counter-timer Physical Secure Timer Control register
CNTPS_CVAL_EL1	UNK	64-bit	Counter-timer Physical Secure Timer CompareValue register

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

[ce] The reset value for bits[9:8, 2:0] is `0b00000`.
[cf] The reset value for bit[0] and bit[1] is `0`.
[cg] The reset value for bit[2] is `0` and for bits[1:0] is `0b11`.

9.3.2 AArch32 Generic Timer register summary

The following table shows the AArch32 Generic Timer registers.

Table 9-3 AArch32 Generic Timer registers

Name	Reset	Width	Description
CNTFRQ	UNK	32-bit	Counter-timer Frequency register
CNTPCT	UNK	64-bit	Counter-timer Physical Count register
CNTKCTL	-[ch]	32-bit	Counter-timer Kernel Control register
CNTP_TVAL	UNK	32-bit	Counter-timer Physical Timer TimerValue register
CNTP_CTL	-[ci]	32-bit	Counter-timer Physical Timer Control register
CNTV_TVAL	UNK	32-bit	Counter-timer Virtual Timer TimerValue register
CNTV_CTL	-[ci]	32-bit	Counter-timer Virtual Timer Control register
CNTVCT	UNK	64-bit	Counter-timer Virtual Count register
CNTP_CVAL	UNK	64-bit	Counter-timer Physical Timer CompareValue register
CNTV_CVAL	UNK	64-bit	Counter-timer Virtual Timer CompareValue register
CNTVOFF	UNK	64-bit	Counter-timer Virtual Offset register
CNTHCTL	-[cj]	32-bit	Counter-timer Hyp Control register
CNTHP_TVAL	UNK	32-bit	Counter-timer Hyp Physical Timer TimerValue register
CNTHP_CTL	-[ci]	32-bit	Counter-timer Hyp Physical Timer Control register
CNTHP_CVAL	UNK	64-bit	Counter-timer Hyp Physical CompareValue register

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

[ch] The reset value for bits[9:8, 2:0] is `0b00000`.
[ci] The reset value for bit[0] and bit[1] is `0`.
[cj] The reset value for bit[2] is `0` and for bits[1:0] is `0b11`.

Chapter 10
Debug

This chapter describes the Cortex-A73 processor debug registers and shows examples of how to use them.

It contains the following sections:

10.1 About debug

This section gives an overview of debug and describes the debug components. The Cortex-A73 processor forms one component of a debug system.

You can use the following debug methods:

Conventional self-hosted debug (previously known as monitor debug)

For self-hosted debug, the debug target runs additional debug monitor software that runs on the Cortex-A73 processor itself, rather than requiring expensive interface hardware to connect a second host computer.

Conventional external debug (known as JTAG debug on SoC)

This is invasive debug with the core halted using:

- Breakpoints, watchpoints, exception catches, and various asynchronous debug events to halt the core on specific activities.
- A debug connection through a specific channel to examine and modify registers and memory, and provide single-step execution.

The following figure shows a typical external debug system.

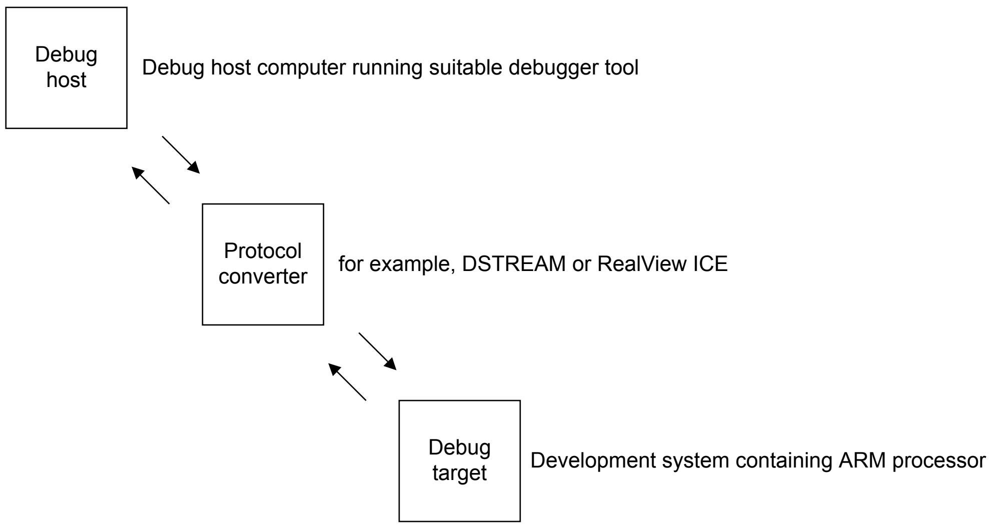
Debug host
Debug host computer running suitable debugger tool
Protocol converter
for example, DSTREAM or RealView ICE
Debug target
Development system containing ARM processor

Figure 10-1 Typical debug system

This typical system has several parts:

Debug host

The debug host is a computer, for example a personal computer, running a software debugger such as the DS-5 Debugger. The debug host enables you to issue high-level commands such as setting 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 debug target is the lowest level of the system. An example of a debug target is a development system with a test chip or a silicon part with a processor.

The debug target implements system support for the protocol converter to access the debug unit using the *Advanced Peripheral Bus* (APB) slave interface.

10.2 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 processor.
- An external debugger.

The Cortex-A73 processor implements the Armv8 Debug architecture and debug events as described in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

This section contains the following subsections:

- *10.2.1 Core interfaces* on page 10-394.
- *10.2.2 Breakpoints and watchpoints* on page 10-394
- *10.2.3 Effects of resets on debug registers* on page 10-394.
- *10.2.4 External access permissions* on page 10-395.

10.2.1 Core interfaces

System register access allows the core to directly access certain debug registers. The external debug interface enables both external and self-hosted debug agents to access debug registers.

See *10.10 External debug interface* on page 10-429 for more information.

Access to the debug registers is partitioned as follows:

Debug registers
This function is system register based and memory-mapped. You can access the debug register map using the APB slave port. See *10.10 External debug interface* on page 10-429.

Performance monitor
This function is system register based and memory-mapped. You can access the performance monitor registers using the APB slave port. See *10.10 External debug interface* on page 10-429.

Trace registers
This function is memory-mapped. See *10.10 External debug interface* on page 10-429.

Cross Trigger Interface registers
This function is memory-mapped. You can access the performance monitor registers using the APB slave port. See *Chapter 13 Cross Trigger* on page 13-545.

10.2.2 Breakpoints and watchpoints

The processor 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 to 3) match only to virtual address and the other two (BRP 4 and 5) match against either virtual address or context ID, or *Virtual Machine Identifier* (VMID). All the watchpoints can be linked to two breakpoints (BRP 4 and 5) to enable a memory request to be trapped in a given process context.

10.2.3 Effects of resets on debug registers

The Cortex-A73 processor has the following reset signals that affect the debug registers:

nCPUPORESET[CN:0]
This signal initializes the core logic, including the debug, *Embedded Trace Macrocell* (ETM) trace unit, breakpoint and watchpoint logic. This maps to a Cold reset that covers reset of the core logic and the integrated debug functionality.

nCORERESET[CN:0]

This signal resets some of the debug logic (excluding breakpoints and watchpoints) and performance monitor logic. This maps to a Warm reset that covers reset of the core logic.

nPRESETDBG

This signal initializes the shared debug APB, top debug, top ETM, top performance monitor logic, *Cross Trigger Interface* (CTI), and *Cross Trigger Matrix* (CTM) logic. This maps to an external debug reset that covers the resetting of the external debug interface and has no impact on the core functionality.

10.2.4 External access permissions

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 10-1 External register conditions

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. **Note** If core power is off, then all external debug and memory-mapped register accesses return an error.
DLK	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.
SLK	Memory-mapped interface only	Software lock is locked. For the external debug interface, ignore this column.
Default	-	None of the conditions apply, normal access.

The following table shows an example of external register condition codes for access to a debug 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 10-2 External register condition code example

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	RO/WI	RO

10.3 AArch64 debug register summary

The following table summarizes debug control registers that are accessible in the AArch64 execution state. These registers are accessed by the `MRS` and `MSR` instructions in the order of Op0, CRn, Op1, CRm, Op2.

See 10.7 *Memory-mapped debug register summary* on page 10-414 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 10-3 AArch64 debug register summary

Op0	CRn	Op1	CRm	Op2	Name	Type	Reset	Width	Description
2	c0	2	c0	0	OSDTRRX_EL1	RW	`0x00000000`	32-bit	OS Lock Data Transfer Register, Receive
2	c0	0	c0	4	DBGBVR0_EL1	RW	`0xXXXXXXXXXXXXXXXX`[ck]	64-bit	Debug Breakpoint Value Register 0
2	c0	0	c0	5	DBGBCR0_EL1	RW	`0x00XXXXXX`[cl]	32-bit	Debug Breakpoint Control Register 0 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
2	c0	0	c0	6	DBGWVR0_EL1	RW	`0xXXXXXXXXXXXXXXXX`[ck]	64-bit	Debug Watchpoint Value Register 0
2	c0	0	c0	7	DBGWCR0_EL1	RW	`0xXXXXXXXX`[cm]	32-bit	Watchpoint Control Register 0 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
2	c0	0	c1	4	DBGBVR1_EL1	RW	`0xXXXXXXXXXXXXXXXX`[ck]	64-bit	Debug Breakpoint Value Register 1
2	c0	0	c1	5	DBGBCR1_EL1	RW	`0x00XXXXXX`[cl]	32-bit	Debug Breakpoint Control Register 1 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
2	c0	0	c1	6	DBGWVR1_EL1	RW	`0xXXXXXXXXXXXXXXXX`[ck]	64-bit	Debug Watchpoint Value Register 1
2	c0	0	c1	7	DBGWCR1_EL1	RW	`0x00XXXXXX`[cm]	32-bit	Debug Watchpoint Control Register 1 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.

Table 10-3 AArch64 debug register summary (continued)

Op0	CRn	Op1	CRm	Op2	Name	Type	Reset	Width	Description
2	c0	0	c2	0	MDCCINT_EL1	RW	`0x00000000`	32-bit	Monitor Debug Comms Channel Interrupt Enable Register
2	c0	0	c2	2	MDSCR_EL1	RW	`0x00000000`	32-bit	Monitor Debug System Register
2	c0	0	c2	4	DBGBVR2_EL1	RW	`0xXXXXXXXXXXXXXXXX`[ck]	64-bit	Debug Breakpoint Value Register 2
2	c0	0	c2	5	DBGBCR2_EL1	RW	`0x00XXXXXX`[cl]	32-bit	Debug Breakpoint Control Register 2 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
2	c0	0	c2	6	DBGWVR2_EL1	RW	`0xXXXXXXXXXXXXXXXX`[ck]	64-bit	Debug Watchpoint Value Register 2
2	c0	0	c2	7	DBGWCR2_EL1	RW	`0x00XXXXXX`[cm]	32-bit	Debug Watchpoint Control Register 2 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
2	c0	0	c3	2	OSDTRTX_EL1	RW	`0x00000000`	32-bit	OS Lock Data Transfer Register, Transmit
2	c0	0	c3	4	DBGBVR3_EL1	RW	`0xXXXXXXXXXXXXXXXX`[ck]	64-bit	Debug Breakpoint Value Register 3
2	c0	0	c3	5	DBGBCR3_EL1	RW	`0x00XXXXXX`[cl]	32-bit	Debug Breakpoint Control Register 3 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
2	c0	0	c3	6	DBGWVR3_EL1	RW	`0xXXXXXXXXXXXXXXXX`[ck]	64-bit	Debug Watchpoint Value Register 3
2	c0	0	c3	7	DBGWCR3_EL1	RW	`0x00XXXXXX`[cl]	32-bit	Debug Watchpoint Control Register 3 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
2	c0	0	c4	4	DBGBVR4_EL1	RW	`0xXXXXXXXX00000000`	64-bit	Debug Breakpoint Value Register 4

Table 10-3 AArch64 debug register summary (continued)

Op0	CRn	Op1	CRm	Op2	Name	Type	Reset	Width	Description
2	c0	0	c4	5	DBGBCR4_EL1	RW	`0x00XXXXXX`[cn]	32-bit	Debug Breakpoint Control Register 4 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
2	c0	0	c5	4	DBGBVR5_EL1	RW	`0xXXXXXXXX00000000`	64-bit	Debug Breakpoint Value Register 5
2	c0	0	c5	5	DBGBCR5_EL1	RW	`0x00XXXXXX`[cn]	32-bit	Debug Breakpoint Control Register 5 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
2	c0	0	c6	2	OSECCR_EL1	RW	`0x00000000`	32-bit	OS Lock Exception Catch Control Register
2	c0	3	c1	0	MDCCSR_EL0	RO	`0x00000000`	32-bit	Monitor Debug Comms Channel Status Register
2	c0	3	c4	0	DBGDTR_EL0	RW	`0x0000000000000000`	64-bit	Debug Data Transfer Register, half-duplex
2	c0	3	c5	0	DBGDTRTX_EL0	WO	-	32-bit	Debug Data Transfer Register, Transmit, Internal View
2	c0	3	c5	0	DBGDTRRX_EL0	RO	`0x00000000`	32-bit	Debug Data Transfer Register, Receive, Internal View
2	c0	4	c7	0	DBGVCR32_EL2	RW	`0x00000000`	32-bit	Debug Vector Catch Register
2	c1	0	c0	0	MDRAR_EL1	RO	[co]	64-bit	Debug ROM Address Register
2	c1	0	c0	4	OSLAR_EL1	WO	-	32-bit	Debug OS Lock Access Register
2	c1	0	c1	4	OSLSR_EL1	RO	`0x0000000A`	32-bit	Debug OS Lock Status Register
2	c1	0	c3	4	OSDLR_EL1	RW	`0x00000000`	32-bit	Debug OS Double Lock Register
2	c1	0	c4	4	DBGPRCR_EL1	RW	[cp]	32-bit	Debug Power/Reset Control Register
2	c7	0	c8	6	DBGCLAIMSET_EL1	RW	`0x000000FF`	32-bit	Debug Claim Tag Set Register
2	c7	0	c9	6	DBGCLAIMCLR_EL1	RW	`0x00000000`	32-bit	Debug Claim Tag Clear Register
2	c7	0	c14	6	DBGAUTHSTATUS_EL1	RO	`0x000000AA`[cq]	32-bit	Debug Authentication Status Register

ck The actual reset value is {62{1'bx}},2'b0.
cl The actual reset value is 32'b000000000x0x0x0xxxx0000xxxx00xx0.
cm The actual reset value is 32'b000xxxxx000x0x0xxxxxxxxxxxxxxxx0.
cn The actual reset value is 32'b00000000xxxx0x0xxxx0000xxxx00xx0.
co Resets to the physical address of the ROM table +3.
cp The actual reset value is 31'b0000000000000000000000000000000,EDPRCR.COREPURQ.
cq The actual reset value is 24'h000000,1'b1,(DBGEN ¦ NIDEN) & (SPIDEN ¦ SPNIDEN),1'b1,DBGEN & SPIDEN,1'b1,DBGEN ¦ NIDEN,1'b1,DBGEN.

10.4 AArch64 debug register descriptions

This section describes the debug registers in AArch64 state.

10.3 AArch64 debug register summary on page 10-396 provides cross-references to the individual registers.

This section contains the following subsections:
- *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
- *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.

10.4.1 Debug Breakpoint Control Registers, EL1

The DBGBCR*n*_EL1 characteristics are:

Purpose Holds control information for a breakpoint. Each DBGBVR_EL1 is associated with a DBGBCR_EL1 to form a *Breakpoint Register Pair* (BRP). DBGBVR*n*_EL1 is associated with DBGBCR*n*_EL1 to form BRP*n*.

Note

The range of breakpoint number *n*, for DBGBCR*n*_EL1 is 0 to 5.

Usage constraints These registers are accessible as follows:

EL0	**EL1 (NS)**	**EL1 (S)**	**EL2**	**EL3 (SCR.NS = 1)**	**EL3 (SCR.NS = 0)**
-	RW	RW	RW	RW	RW

Configurations DBGBCR*n*_EL1 are architecturally mapped to:
- The AArch32 DBGBCR*n* registers.
- The external DBGBCR*n*_EL1 registers.

Attributes DBGBCR*n*_EL1 is a 32-bit register.

The debug logic reset value of a DBGBCR*n*_EL1 is UNKNOWN.

The following figure shows the DBGBCR*n*_EL1 bit assignments.

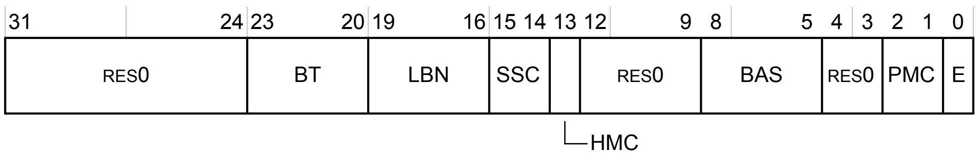

Figure 10-2 DBGBCR*n*_EL1 bit assignments.

The following table shows the DBGBCR*n*_EL1 bit assignments.

Table 10-4 DBGBCR*n*_EL1 bit assignments

Bits	Name	Function
[31:24]	-	Reserved, RES0.
[23:20]	BT	Breakpoint Type. This field controls the behavior of breakpoint debug event generation. This includes the meaning of the value held in the associated DBGBVR, 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 ContextIDR match. `0b0011` Linked ContextIDR match. `0b0100` Unlinked instruction address mismatch. `0b0101` Linked instruction address mismatch. `0b1000` Unlinked VMID match. `0b1001` Linked VMID match. `0b1010` Unlinked VMID + CONTEXTIDR match. `0b1011` Linked VMID + CONTEXTIDR match. All other values are reserved. The field break down is: • BT[2] and BT[0] are available for all BRPs and are read/write. —— **Note** —— BT[0], (BCR[20]) sets LBN[2] (BCR[18]) ———— • BT[3] and BT[1] are only available for context-aware breakpoints (BRP 4 and 5)
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. It is only possible to link to context-aware breakpoints. • LBN[0] is read/write. • LBN[1] and LBN[3] are RES0. • LBN[2] is mapped to BT[0] which is read/write.
[15:14]	SSC	Security State Control. Determines the Security states under which a breakpoint debug event for a breakpoint *n* is generated. This field must be interpreted with the HMC 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 fields.
[13]	HMC	Higher Mode Control. 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 fields.
[12:9]	-	Reserved, RES0.

Table 10-4 DBGBCR*n*_EL1 bit assignments (continued)

Bits	Name	Function
[8:5]	BAS[cr]	Byte Address Select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. A debugger must program this field as follows: `0x3` Match the T32 instruction at DBGBVRn. `0xC` Match the T32 instruction at DBGBVRn+2. `0xF` Match the A64 or A32 instruction at DBGBVRn, or context match. All other values are reserved. **Note** 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.
[4:3]	-	Reserved, RES0.
[2:1]	PMC	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 fields. **Note** Bits[2:1] have no effect for accesses made in Hyp mode.
[0]	E	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. **Note** DBGBCR.E is reset on cold reset, using **nCPUPORESET**.

To access the DBGBCR*n*_EL1 in AArch64 state, read or write the register with:

```
MRS <Xt>, DBGBCRn_EL1; Read Debug Breakpoint Control Register nMSR DBGBCRn_EL1, <Xt>; Write
Debug Breakpoint Control Register n
```

To access the DBGBCR*n* in AArch32 state, read or write the CP14 register with:

```
MRC p14, 0, <Rt>, c0, cn, 4; Read Debug Breakpoint Control Register nMCR p14, 0, <Rt>, c0,
cn, 4; Write Debug Breakpoint Control Register n
```

The DBGBCR*n*_EL1 can be accessed through the external debug interface, offset `0x4n8`.

10.4.2 Debug Watchpoint Control Registers, EL1

The DBGWCR*n*_EL1 characteristics are:

[cr] See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information on how the BAS field is interpreted by hardware.

Purpose Holds control information for a watchpoint. Each DBGWCR_EL1 is associated with a DBGWVR_EL1 to form a *Watchpoint Register Pair* (WRP). DBGWCR*n*_EL1 is associated with DBGWVR*n*_EL1 to form WRP*n*.

Note

The range of watchpoint number *n* for DBGWCR*n*_EL1 is 0 to 3.

Usage constraints These registers are accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	RW	RW	RW	RW	RW

Configurations The DBGWCR*n*_EL1 is architecturally mapped to:
- The AArch32 DBGWCR*n* registers.
- The external DBGWCR*n*_EL1 registers.

Attributes DBGWCR*n*_EL1 is a 32-bit register.

The debug logic reset value of a DBGWCR_EL1 is UNKNOWN.

The following figure shows the DBGWCR*n*_EL1 bit assignments.

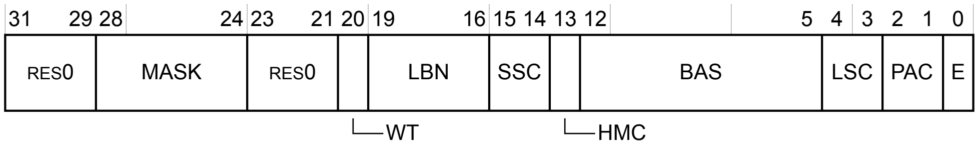

Figure 10-3 DBGWCR*n*_EL1 bit assignments

The following table shows the DBGWCR*n*_EL1 bit assignments.

Table 10-5 DBGWCR*n*_EL1 bit assignments

Bits	Name	Function
[31:29]	-	Reserved, RES0.
[28:24]	MASK	Address Mask. Only objects up to 2GB can be watched using a single mask. `0b00000` No mask. `0b00001` Reserved. `0b00010` Reserved. Other values mask the corresponding number of address bits, from `0b00011` masking three address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).
[23:21]	-	Reserved, RES0.
[20]	WT	Watchpoint Type. Possible values are: `0` Unlinked data address match. `1` Linked data address match.

Table 10-5 DBGWCR*n*_EL1 bit assignments (continued)

Bits	Name	Function
[19:16]	LBN	Linked Breakpoint Number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to. It is only possible to link to context-aware breakpoints. • LBN[0] is read/write. • LBN[1] and LBN[3] are RES0. • LBN[2] is mapped to WT[0] which is read/write. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[15:14]	SSC	Security State Control. Determines the Security states under which a watchpoint debug event for watchpoint *n* is generated. This field must be interpreted along with the HMC and PAC fields. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[13]	HMC	Higher Mode Control. Determines the debug perspective for deciding when a watchpoint debug event for watchpoint *n* is generated. This field must be interpreted along with the SSC and PAC fields. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[12:5]	BAS	Byte Address Select. Each bit of this field selects whether a byte from within the word or doubleword addressed by DBGWVR*n*_EL1 is being watched. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[4:3]	LSC	Load/Store Control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: `0b01` Match instructions that load from a watchpointed address. `0b10` Match instructions that store to a watchpointed address. `0b11` Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled.
[2:1]	PAC	Privilege of Access Control. Determines the Exception level or levels at which a watchpoint debug event for watchpoint *n* is generated. This field must be interpreted together with the SSC and HMC fields. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
[0]	E	Enable Watchpoint *n*. Possible values are: `0` Watchpoint disabled. `1` Watchpoint enabled. ——— **Note** ——— DBGBCR.E is reset on cold reset, using **nCPUPORESET**. ———————

To access the DBGWCR*n*_EL1 in AArch64 state, read or write the register with:

```
MRS <Xt>, DBGWCRn_EL1; Read Debug Watchpoint Control Register nMSR DBGWCRn_EL1, <Xt>; Write
Debug Watchpoint Control Register n
```

To access the DBGWCR*n* in AArch32 state, read or write the CP14 register with:

```
MRC p14, 0, <Rt>, c0, cn, 7; Read Debug Watchpoint Control Register nMCR p14, 0, <Rt>, c0,
cn, 7; Write Debug Watchpoint Control Register n
```

The DBGWCR*n*_EL1 can be accessed through the external debug interface, offset `0x8n8`.

10.5 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 Armv8, for Armv8-A architecture profile*. See the *10.7 Memory-mapped debug register summary* on page 10-414 for a complete list of registers accessible from the external debug interface.

Table 10-6 AArch32 debug register summary

CRn	Op2	CRm	Op1	Name	Type	Reset	Description
c0	0	c0	0	DBGDIDR	RO	`0x3516D000`	*10.6.1 Debug ID Register* on page 10-410
c0	0	c1	0	DBGDSCRint	RO	`0x000X0000`[cs]	Debug Status and Control Register, Internal View
c0	0	c2	0	DBGDCCINT	RW	`0x00000000`	Debug Comms Channel Interrupt Enable Register
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
c0	0	c6	0	DBGWFAR[ct]	RW	-	Watchpoint Fault Address Register, RES0
c0	0	c7	0	DBGVCR	RW	`0x00000000`	Debug Vector Catch Register
c0	2	c0	0	DBGDTRRXext	RW	`0x00000000`	Debug Data Transfer Register, Receive, External View
c0	2	c2	0	DBGDSCRext	RW	`0x000X0000`[cs]	Debug Status and Control Register, External View
c0	2	c3	0	DBGDTRTXext	RW	`0x00000000`	Debug Data Transfer Register, Transmit, External View
c0	2	c6	0	DBGOSECCR	RW	`0x00000000`	Debug OS Lock Exception Catch Control Register
c0	4	c0	0	DBGBVR0	RW	`0xXXXXXXXX`[cu]	Debug Breakpoint Value Register 0
c0	4	c1	0	DBGBVR1	RW	`0xXXXXXXXX`[cu]	Debug Breakpoint Value Register 1
c0	4	c2	0	DBGBVR2	RW	`0xXXXXXXXX`[cu]	Debug Breakpoint Value Register 2

Table 10-6 AArch32 debug register summary (continued)

CRn	Op2	CRm	Op1	Name	Type	Reset	Description
c0	4	c3	0	DBGBVR3	RW	`0xXXXXXXXX`[cu]	Debug Breakpoint Value Register 3
c0	4	c4	0	DBGBVR4	RW	`0xXXXXXXXX`[cu]	Debug Breakpoint Value Register 4
c0	4	c5	0	DBGBVR5	RW	`0xXXXXXXXX`[cu]	Debug Breakpoint Value Register 5
c0	5	c0	0	DBGBCR0	RW	`0x00XXXXXX`[cv]	Debug Breakpoint Control Register 0 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
c0	5	c1	0	DBGBCR1	RW	`0x00XXXXXX`[cv]	Debug Breakpoint Control Register 1 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
c0	5	c2	0	DBGBCR2	RW	`0x00XXXXXX`[cv]	Debug Breakpoint Control Register 2 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
c0	5	c3	0	DBGBCR3	RW	`0x00XXXXXX`[cv]	Debug Breakpoint Control Register 3 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
c0	5	c4	0	DBGBCR4	RW	`0x00XXXXXX`[cw]	Debug Breakpoint Control Register 4 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
c0	5	c5	0	DBGBCR5	RW	`0x00XXXXXX`[cw]	Debug Breakpoint Control Register 5 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.

Table 10-6 AArch32 debug register summary (continued)

CRn	Op2	CRm	Op1	Name	Type	Reset	Description
c0	6	c0	0	DBGWVR0	RW	`0xXXXXXXX`[cu]	Debug Watchpoint Value Register 0
c0	6	c1	0	DBGWVR1	RW	`0xXXXXXXX`[cu]	Debug Watchpoint Value Register 1
c0	6	c2	0	DBGWVR2	RW	`0xXXXXXXX`[cu]	Debug Watchpoint Value Register 2
c0	6	c3	0	DBGWVR3	RW	`0xXXXXXXX`[cu]	Debug Watchpoint Value Register 3
c0	7	c0	0	DBGWCR0	RW	`0xXXXXXXX`[cx]	Watchpoint Control Register 0 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
c0	7	c1	0	DBGWCR1	RW	`0xXXXXXXX`[cx]	Watchpoint Control Register 1 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
c0	7	c2	0	DBGWCR2	RW	`0xXXXXXXX`[cx]	Watchpoint Control Register 2 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
c0	7	c3	0	DBGWCR3	RW	`0xXXXXXXX`[cx]	Watchpoint Control Register 3 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
c1	0	c0	0	DBGDRAR[31:0]	RO	[cy]	Debug ROM Address Register
-	-	c1	-	DBGDRAR[63:0]	RO	[cz]	
c1	1	c4	0	DBGBXVR4	RW	`0xXXXXXXX`[da]	Debug Breakpoint Extended Value Register 4
c1	1	c5	0	DBGBXVR5	RW	`0xXXXXXXX`[da]	Debug Breakpoint Extended Value Register 5
c1	4	c0	0	DBGOSLAR	WO	-	Debug OS Lock Access Register

Table 10-6 AArch32 debug register summary (continued)

CRn	Op2	CRm	Op1	Name	Type	Reset	Description
c1	4	c1	0	DBGOSLSR	RO	`0x0000000A`	Debug OS Lock Status Register
c1	4	c3	0	DBGOSDLR	RW	`0x00000000`	Debug OS Double Lock Register
c1	4	c4	0	DBGPRCR	RW	[db]	Debug Power/Reset Control Register
c2	2	c0	0	DBGDSAR[31:0]	RO	-	Debug Self Address Register RES0
-	0	c2	-	DBGDSAR[63:0][dc]	RO	-	
c7	7	c0	0	DBGDEVID2	RO	`0x00000000`	Debug Device ID Register 2, RES0
c7	7	c1	0	DBGDEVID1	RO	`0x00000002`	*10.6.3 Debug Device ID Register 1* on page 10-412
c7	7	c2	0	DBGDEVID	RO	`0x00110F13`	*10.6.2 Debug Device ID Register* on page 10-411
c7	6	c8	0	DBGCLAIMSET	RW	`0x000000FF`	Debug Claim Tag Set Register
c7	6	c9	0	DBGCLAIMCLR	RW	`0x00000000`	Debug Claim Tag Clear Register
c7	6	c14	0	DBGAUTHSTATUS	RO	`0x000000AA`[dd]	Debug Authentication Status Register

cs The actual reset value is 32'b0000000000000xxx0000000000000000.
ct Previously returned information about the address of the instruction that accessed a watchpoint address. This register is now deprecated and is RES0.
cu The actual reset value is {30{1'bx}},2'b0
cv The actual reset value is 32'b000000000x0x0x0xxxx0000xxxx00xx0.
cw The actual reset value is 32'b00000000xxxx0x0xxxx0000xxxx00xx0.
cx The actual reset value is 32'b000xxxxx000x0x0xxxxxxxxxxxxxxxx0.
cy The actual reset value is ROMADDR[31:12],10'b0000000000,{2{ROMADDRV}}.
cz The actual reset value is 0x000000,ROMADDR[39:12],10'b0000000000,{2{ROMADDRV}}.
da The actual reset value is 32'hxxxxxxxxx.
db The actual reset value is 31'b0000000000000000000000000000000,EDPRCR.COREPURQ.
dc Previously defined the offset from the base address defined in DBGDRAR of the physical base address of the debug registers for the processor. This register is now deprecated and RES0.
dd The actual reset value is 24'h000000,1'b1,(DBGEN ¦ NIDEN) & (SPIDEN ¦ SPNIDEN),1'b1,DBGEN & SPIDEN,1'b1,DBGEN ¦ NIDEN,1'b1,DBGEN.

10.6 AArch32 debug register descriptions

This section describes the debug registers in AArch32 state.

10.5 AArch32 debug register summary on page 10-406 provides cross-references to the individual registers.

This section contains the following subsections:
- *10.6.1 Debug ID Register* on page 10-410.
- *10.6.2 Debug Device ID Register* on page 10-411.
- *10.6.3 Debug Device ID Register 1* on page 10-412.

10.6.1 Debug ID Register

The DBGDIDR characteristics are:

Purpose Specifies:
- The version of the Debug architecture.
- Some features of the debug implementation.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
RO	RO	RO	RO	RO	RO	RO

Configurations There is one copy of this register that is used in both Secure and Non-secure states.

Attributes DBGDIDR is a 32-bit register.

The following figure shows the DBGDIDR bit assignments.

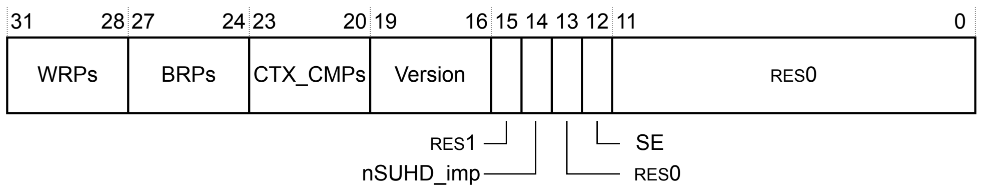

Figure 10-4 DBGDIDR bit assignments

The following table shows the DBGDIDR bit assignments.

Table 10-7 DBGDIDR bit assignments

Bits	Name	Function
[31:28]	WRPs	The number of *Watchpoint Register Pairs* (WRPs) implemented, minus one. This value is: `0x3` Four WRPs are implemented. This field has the same value as ID_AA64DFR0_EL1.WRPs.
[27:24]	BRPs	The number of *Breakpoint Register Pairs* (BRPs) implemented, minus one. This value is: `0x5` Six BRPs are implemented. This field has the same value as ID_AA64DFR0_EL1.BRPs.

Table 10-7 DBGDIDR bit assignments (continued)

Bits	Name	Function
[23:20]	CTX_CMPs	The number of BRPs that can be used for Context matching, minus one. This value is: `0x1` Two Context matching breakpoints, breakpoints 4 and 5, are implemented. This field has the same value as ID_AA64DFR0_EL1.CTX_CMPs.
[19:16]	Version	The Debug architecture version. `0x6` Armv8, v8 Debug architecture is implemented.
[15]	-	Reserved, RES1.
[14]	nSUHD_imp	The value of this bit must match the value of SE_imp. This value is: `1` The value of SE_imp is `1`.
[13]	-	Reserved, RES0.
[12]	SE_imp	EL3 implemented. The value is: `1` EL3 is implemented.
[11:0]	-	Reserved, RES0.

To access the DBGDIDR in AArch32 state, read the CP14 register with:

```
MRC p14, 0, <Rt>, c0, c0, 0; Read Debug ID Register
```

10.6.2 Debug Device ID Register

The DBGDEVID characteristics are:

Purpose Adds to the information given by the DBGDIDR by describing other features of the debug implementation.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Configurations There is one copy of this register that is used in both Secure and Non-secure states.

Attributes DBGDEVID is a 32-bit register.

The following figure shows the DBGDEVID bit assignments.

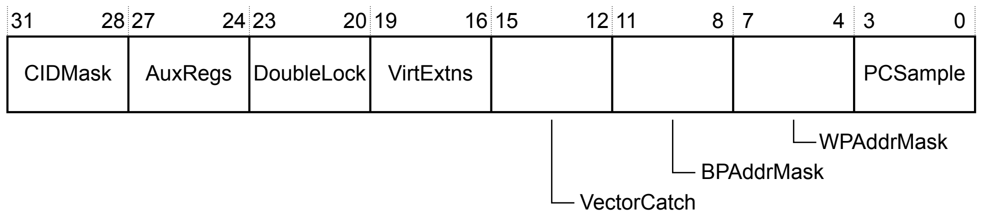

Figure 10-5 DBGDEVID bit assignments

The following table shows the DBGDEVID bit assignments.

Table 10-8 DBGDEVID bit assignments

Bits	Name	Function
[31:28]	CIDMask	Indicates the level of support for the Context ID matching breakpoint masking capability. This value is: `0x0` Context ID masking is not implemented.
[27:24]	AuxRegs	Indicates support for Auxiliary registers. This value is: `0x0` None supported.
[23:20]	DoubleLock	Indicates the presence of the DBGOSDLR, OS Double Lock Register. This value is: `0x1` The DBGOSDLR is present.
[19:16]	VirtExtns	Indicates whether EL2 is implemented. This value is: `0x1` EL2 is implemented.
[15:12]	VectorCatch	Defines the form of Vector catch debug event implemented. This value is: `0x0` Address matching vector catch debug event implemented.
[11:8]	BPAddrMask	Indicates the level of support for the *Immediate Virtual Address* (IVA) matching breakpoint masking capability. This value is: `0xF` Breakpoint address masking is not implemented. DBGBCR*n*[28:24] is RES0.
[7:4]	WPAddrMask	Indicates the level of support for the data VA matching watchpoint masking capability. This value is: `0x1` Watchpoint address mask implemented.
[3:0]	PCSample	Indicates the level of Sample-based profiling support using external debug registers 40 through 43. This value is: `0x3` EDPCSR, EDCIDSR and EDVIDSR are implemented as debug registers 40, 41, and 42.

To access the DBGDEVID in AArch32 state, read the CP14 register with:

```
MRC p14, 0, <Rt>, c7, c2, 7; Read Debug Device ID Register 0
```

10.6.3 Debug Device ID Register 1

The DBGDEVID1 characteristics are:

Purpose Adds to the information given by the DBGDIDR by describing other features of the debug implementation.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Configurations There is one copy of this register that is used in both Secure and Non-secure states.

Attributes DBGDEVID1 is a 32-bit register.

The following figure shows the DBGDEVID1 register bit assignments.

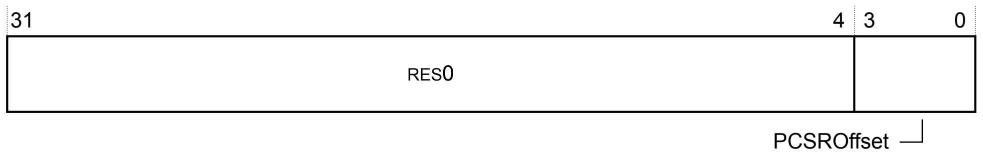

Figure 10-6 DBGDEVID1 bit assignments

The following table shows the DBGDEVID1 register bit assignments.

Table 10-9 DBGDEVID1 bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	PCSROffset	Indicates the offset applied to PC samples returned by reads of EDPCSR. The value is: `0x2` EDPCSR is implemented and samples have no offset applied and do not sample the instruction set state in AArch32 state.

To access the DBGDEVID1 register in AArch32 state, read the CP14 register with:

```
MRC p14, 0, <Rt>, c7, c1, 47 Read Debug Device ID Register 1
```

10.7 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 10-10 Memory-mapped debug register summary

Offset	Name	Type	Width	Description
`0x000-0x01C`	-	-	-	Reserved
`0x020`	EDESR	RW	32-bit	External Debug Event Status Register
`0x024`	EDECR	RW	32-bit	External Debug Execution Control Register
`0x028-0x02C`	-	-	-	Reserved
`0x030`	EDWAR[31:0]	RO	64-bit	External Debug Watchpoint Address Register
`0x034`	EDWAR[63:32]			
`0x038-0x07C`	-	-	-	Reserved
`0x080`	DBGDTRRX_EL0	RW	32-bit	Debug Data Transfer Register, Receive
`0x084`	EDITR	WO	32-bit	External Debug Instruction Transfer Register
`0x088`	EDSCR	RW	32-bit	External Debug Status and Control Register
`0x08C`	DBGDTRTX_EL0	RW	32-bit	Debug Data Transfer Register, Transmit
`0x090`	EDRCR	WO	32-bit	*10.8.1 External Debug Reserve Control Register* on page 10-418
`0x098`	EDECCR	RW	32-bit	External Debug Exception Catch Control Register
`0x09C`	-	-	32-bit	Reserved
`0x0A0`	EDPCSRlo	RO	32-bit	External Debug Program Counter Sample Register, low word
`0x0A4`	EDCIDSR	RO	32-bit	External Debug Context ID Sample Register
`0x0A8`	EDVIDSR	RO	32-bit	External Debug Virtual Context Sample Register
`0x0AC`	EDPCSRhi	RO	32-bit	External Debug Program Counter Sample Register, high word
`0x0B0-0x2FC`	-	-	-	Reserved
`0x300`	OSLAR_EL1	WO	32-bit	OS Lock Access Register
`0x304-0x30C`	-	-	-	Reserved
`0x310`	EDPRCR	RW	32-bit	External Debug Power/Reset Control Register
`0x314`	EDPRSR	RO	32-bit	External Debug Processor Status Register
`0x318-0x3FC`	-	-	-	Reserved
`0x400`	DBGBVR0_EL1[31:0]	RW	64-bit	Debug Breakpoint Value Register 0
`0x404`	DBGBVR0_EL1[63:32]			
`0x408`	DBGBCR0_EL1	RW	32-bit	Debug Breakpoint Control Register 0 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.

Table 10-10 Memory-mapped debug register summary (continued)

Offset	Name	Type	Width	Description
0x40C	-	-	-	Reserved
0x410	DBGBVR1_EL1[31:0]	RW	64-bit	Debug Breakpoint Value Register 1
0x414	DBGBVR1_EL1[63:32]			
0x418	DBGBCR1_EL1	RW	32-bit	Debug Breakpoint Control Register 1 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
0x41C	-	-	-	Reserved
0x420	DBGBVR2_EL1[31:0]	RW	64-bit	Debug Breakpoint Value Register 2
0x424	DBGBVR2_EL1[63:32]			
0x428	DBGBCR2_EL1	RW	32-bit	Debug Breakpoint Control Register 2 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
0x42C	-	-	-	Reserved
0x430	DBGBVR3_EL1[31:0]	RW	64-bit	Debug Breakpoint Value Register 3
0x434	DBGBVR3_EL1[63:32]			
0x438	DBGBCR3_EL1	RW	32-bit	Debug Breakpoint Control Register 3 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
0x43C	-	-	-	Reserved
0x440	DBGBVR4_EL1[31:0]	RW	64-bit	Debug Breakpoint Value Register 4
0x444	DBGBVR4_EL1[63:32]			
0x448	DBGBCR4_EL1	RW	32-bit	Debug Breakpoint Control Register 4 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
0x44C	-	-	-	Reserved
0x450	DBGBVR5_EL1[31:0]	RW	64-bit	Debug Breakpoint Value Register 5
0x454	DBGBVR5_EL1[63:32]			
0x458	DBGBCR5_EL1	RW	32-bit	Debug Breakpoint Control Register 5 See *10.4.1 Debug Breakpoint Control Registers, EL1* on page 10-400.
0x45C-0x7FC	-	-	-	Reserved
0x800	DBGWVR0_EL1[31:0]	RW	64-bit	Debug Watchpoint Value Register 0
0x804	DBGWVR0_EL1[63:32]			

Table 10-10 Memory-mapped debug register summary (continued)

Offset	Name	Type	Width	Description
0x808	DBGWCR0_EL1	RW	32-bit	Debug Watchpoint Control Register 0 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
0x80C	-	-	-	Reserved
0x810	DBGWVR1_EL1[31:0]	RW	64-bit	Debug Watchpoint Value Register 1
0x814	DBGWVR1_EL1[63:32]			
0x818	DBGWCR1_EL1	RW	32-bit	Debug Watchpoint Control Register 1 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
0x81C	-	-	-	Reserved
0x820	DBGWVR2_EL1[31:0]	RW	64-bit	Debug Watchpoint Value Register 2
0x824	DBGWVR2_EL1[63:32]			
0x828	DBGWCR2_EL1	RW	32-bit	Debug Watchpoint Control Register 2 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
0x82C	-	-	-	Reserved
0x830	DBGWVR3_EL1[31:0]	RW	64-bit	Debug Watchpoint Value Register 3
0x834	DBGWVR3_EL1[63:32]			
0x838	DBGWCR3_EL1	RW	32-bit	Debug Watchpoint Control Register 3 See *10.4.2 Debug Watchpoint Control Registers, EL1* on page 10-402.
0x83C-0xCFC	-	-	-	Reserved
0xD00	MIDR	RO	32-bit	*4.3.1 Main ID Register, EL1* on page 4-83
0xD04-0xD1C	-	-	-	Reserved
0xD20	ID_AA64PFR0_EL1[31:0]	RO	64-bit	*4.3.20 AArch64 Processor Feature Register 0* on page 4-107
0xD24	ID_AA64PFR0_EL1[63:32]			
0xD28	ID_AA64DFR0_EL1[31:0]	RO	64-bit	*4.3.22 AArch64 Debug Feature Register 0, EL1* on page 4-109
0xD2C	ID_AA64DFR0_EL1[63:32]			
0xD30	ID_AA64ISAR0_EL1[31:0]	RO	64-bit	*4.3.26 AArch64 Instruction Set Attribute Register 0, EL1* on page 4-111
0xD34	ID_AA64ISAR0_EL1[63:32]			
0xD38	ID_AA64MMFR0_EL1[31:0]	RO	64-bit	*4.3.28 AArch64 Memory Model Feature Register 0, EL1* on page 4-112
0xD3C	ID_AA64MMFR0_EL1[63:32]			
0xD40	ID_AA64PFR1_EL1[31:0]	RO	64-bit	Processor Feature Register 1, RES0
0xD44	ID_AA64PFR1_EL1[63:32]			

Table 10-10 Memory-mapped debug register summary (continued)

Offset	Name	Type	Width	Description
`0xD48`	ID_AA64DFR1_EL1[31:0]	RO	64-bit	Debug Feature Register 1, RES0
`0xD4C`	ID_AA64DFR1_EL1[63:32]			
`0xD50`	ID_AA64ISAR1_EL1[31:0]	RO	64-bit	Instruction Set Attribute Register 1 low word, RES0
`0xD54`	ID_AA64ISAR1_EL1[63:32]			
`0xD58`	ID_AA64MMFR1_EL1[31:0]	RO	64-bit	Memory Model Feature Register 1 low word, RES0
`0xD5C`	ID_AA64MMFR1_EL1[63:32]			
`0xD60-0xEFC`	-	-	-	Reserved
`0xF04-0xF9C`	-	-	-	Reserved
`0xFA0`	DBGCLAIMSET_EL1	RW	32-bit	Debug Claim Tag Set Register
`0xFA4`	DBGCLAIMCLR_EL1	RW	32-bit	Debug Claim Tag Clear Register
`0xFA8`	EDDEVAFF0	RO	32-bit	External Debug Device Affinity Register 0
`0xFAC`	EDDEVAFF1	RO	32-bit	External Debug Device Affinity Register 1, RES0
`0xFB0`	EDLAR	WO	32-bit	External Debug Lock Access Register
`0xFB4`	EDLSR	RO	32-bit	External Debug Lock Status Register
`0xFB8`	DBGAUTHSTATUS_EL1	RO	32-bit	Debug Authentication Status Register
`0xFBC`	EDDEVARCH	RO	32-bit	External Debug Device Architecture Register
`0xFC0`	EDDEVID2	RO	32-bit	External Debug Device ID Register 2, RES0
`0xFC4`	EDDEVID1	RO	32-bit	*10.8.3 External Debug Device ID Register 1* on page 10-419
`0xFC8`	EDDEVID	RO	32-bit	*10.8.2 External Debug Device ID Register 0* on page 10-419
`0xFCC`	EDDEVTYPE	RO	32-bit	External Debug Device Type Register
`0xFD0`	EDPIDR4	RO	32-bit	*External Debug Peripheral Identification Register 4* on page 10-423
`0xFD4-0xFDC`	EDPIDR5-7	RO	32-bit	*External Debug Peripheral Identification Register 5-7* on page 10-424
`0xFE0`	EDPIDR0	RO	32-bit	*External Debug Peripheral Identification Register 0* on page 10-421
`0xFE4`	EDPIDR1	RO	32-bit	*External Debug Peripheral Identification Register 1* on page 10-421
`0xFE8`	EDPIDR2	RO	32-bit	*External Debug Peripheral Identification Register 2* on page 10-422
`0xFEC`	EDPIDR3	RO	32-bit	*External Debug Peripheral Identification Register 3* on page 10-423
`0xFF0`	EDCIDR0	RO	32-bit	*External Debug Component Identification Register 0* on page 10-424
`0xFF4`	EDCIDR1	RO	32-bit	*External Debug Component Identification Register 1* on page 10-425
`0xFF8`	EDCIDR2	RO	32-bit	*External Debug Component Identification Register 2* on page 10-425
`0xFFC`	EDCIDR3	RO	32-bit	*External Debug Component Identification Register 3* on page 10-426

10.8 Memory-mapped debug register descriptions

This section describes the Cortex-A73 processor debug registers.

10.7 Memory-mapped debug register summary on page 10-414 provides cross-references to the individual registers.

This section contains the following subsections:

- *10.8.1 External Debug Reserve Control Register* on page 10-418.
- *10.8.2 External Debug Device ID Register 0* on page 10-419.
- *10.8.3 External Debug Device ID Register 1* on page 10-419.
- *10.8.4 External Debug Peripheral Identification Registers* on page 10-420.
- *10.8.5 External Debug Component Identification Registers* on page 10-424.

10.8.1 External Debug Reserve Control Register

The EDRCR characteristics are:

Purpose This register is used to allow imprecise entry to Debug state and clear sticky bits in EDSCR.

This register is part of the Debug registers functional group.

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.

Attributes EDRCR is a 32-bit register.

The following figure shows the EDRCR bit assignments.

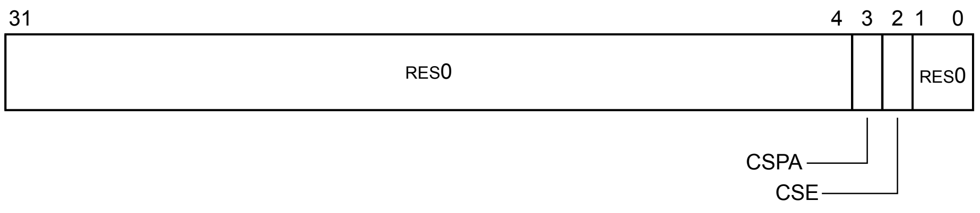

Figure 10-7 EDRCR bit assignments

The following table shows the EDRCR bit assignments.

Table 10-11 EDRCR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3]	CSPA	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`.

Table 10-11 EDRCR bit assignments (continued)

Bits	Name	Function
[2]	CSE	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`.
[1:0]	-	Reserved, RES0.

The EDRCR can be accessed through the external debug interface, offset `0x090`.

10.8.2 External Debug Device ID Register 0

The EDDEVID register characteristics are:

Purpose Provides extra information for external debuggers about features of the debug implementation.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDDEVID register is in the Debug power domain.

Attributes EDDEVID is a 32-bit register.

The following figure shows the EDDEVID bit assignments.

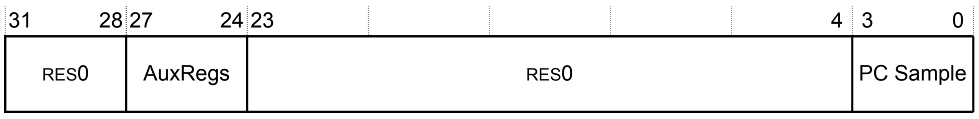

Figure 10-8 EDDEVID bit assignments

The following table shows the EDDEVID bit assignments.

Table 10-12 EDDEVID bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RES0.
[27:24]	AuxRegs	Indicates support for Auxiliary registers. This value is: `0x0` None supported.
[23:4]	-	Reserved, RES0.
[3:0]	PC Sample	Indicates the level of Sample-based profiling support using external debug registers 40 through 43. This value is: `0x3` EDPCSR, EDCIDSR and EDVIDSR are implemented.

The EDDEVID register can be accessed through the external debug interface, offset `0xFC8`.

10.8.3 External Debug Device ID Register 1

The EDDEVID1 register characteristics are:

Purpose Provides extra information for external debuggers about features of the debug implementation.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDDEVID1 register is in the Debug power domain.

Attributes EDDEVID1 is a 32-bit register.

The following figure shows the EDDEVID1 bit assignments.

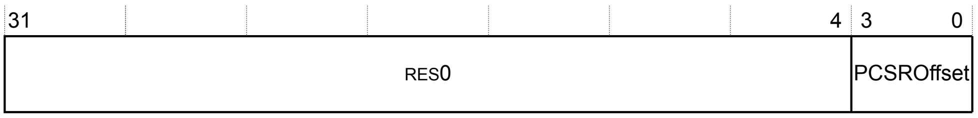

Figure 10-9 EDDEVID1 bit assignments

The following table shows the EDDEVID1 bit assignments.

Table 10-13 EDDEVID1 bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	PCSROffset	Indicates the offset applied to PC samples returned by reads of EDPCSR. This value is: `0x2` EDPCSR is implemented, and samples have no offset applied and do not sample the instruction set state in AArch32 state.

The EDDEVID1 register can be accessed through the external debug interface, offset `0xFC4`.

10.8.4 External Debug Peripheral Identification Registers

The External Debug Peripheral Identification Registers provide standard information required for all components that conform to the *Arm® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. They are a set of eight registers, listed in register number order in the following table.

Table 10-14 Summary of the External Debug Peripheral Identification Registers

Register	Value	Offset
EDPIDR4	`0x04`	`0xFD0`
EDPIDR5	`0x00`	`0xFD4`
EDPIDR6	`0x00`	`0xFD8`
EDPIDR7	`0x00`	`0xFDC`
EDPIDR0	`0x09`	`0xFE0`
EDPIDR1	`0xBD`	`0xFE4`
EDPIDR2	`0x0B`	`0xFE8`
EDPIDR3	`0x00`	`0xFEC`

Only bits[7:0] of each External Debug Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight External Debug Peripheral ID Registers define a single 64-bit Peripheral ID.

The External Debug Peripheral ID registers are:

- *External Debug Peripheral Identification Register 0* on page 10-421.
- *External Debug Peripheral Identification Register 1* on page 10-421.
- *External Debug Peripheral Identification Register 2* on page 10-422.
- *External Debug Peripheral Identification Register 3* on page 10-423.
- *External Debug Peripheral Identification Register 4* on page 10-423.
- *External Debug Peripheral Identification Register 5-7* on page 10-424.

External Debug Peripheral Identification Register 0

The EDPIDR0 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDPIDR0 is in the Debug power domain.

Attributes EDPIDR0 is a 32-bit register.

The following figure shows the EDPIDR0 bit assignments.

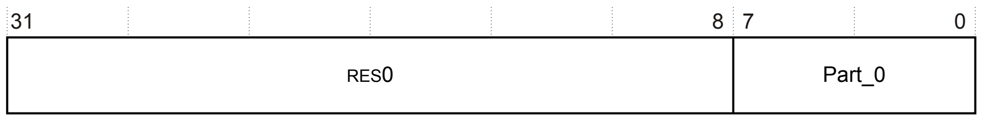

Figure 10-10 EDPIDR0 bit assignments

The following table shows the EDPIDR0 bit assignments.

Table 10-15 EDPIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	Part_0	`0x09` Least significant byte of the debug part number.

The EDPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

External Debug Peripheral Identification Register 1

The EDPIDR1 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDPIDR1 is in the Debug power domain.

Attributes EDPIDR1 is a 32-bit register.

The following figure shows the EDPIDR1 bit assignments.

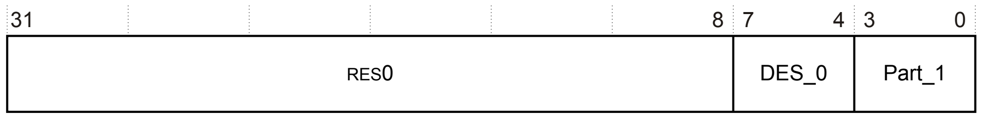

Figure 10-11 EDPIDR1 bit assignments

The following table shows the EDPIDR1 bit assignments.

Table 10-16 EDPIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	DES_0	`0xB` Arm Limited. This is the least significant nibble of JEP106 ID code.
[3:0]	Part_1	`0xD` Most significant nibble of the debug part number.

The EDPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

External Debug Peripheral Identification Register 2

The EDPIDR2 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDPIDR2 is in the Debug power domain.

Attributes EDPIDR2 is a 32-bit register.

The following figure shows the EDPIDR2 bit assignments.

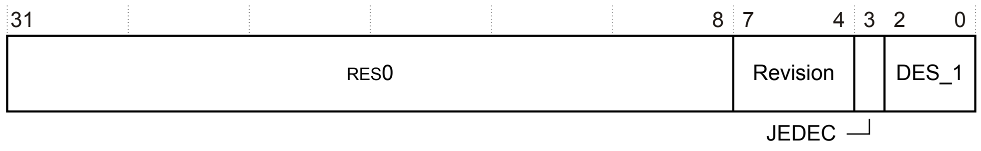

Figure 10-12 EDPIDR2 bit assignments

The following table shows the EDPIDR2 bit assignments.

Table 10-17 EDPIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Revision	`0x0` r0p0.
[3]	JEDEC	RES1 Indicates a JEP106 identity code is used.
[2:0]	DES_1	`0b011` Arm Limited. These are the most significant bits of JEP106 ID code.

The EDPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

External Debug Peripheral Identification Register 3

The EDPIDR3 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDPIDR3 is in the Debug power domain.

Attributes EDPIDR3 is a 32-bit register.

The following figure shows the EDPIDR3 bit assignments.

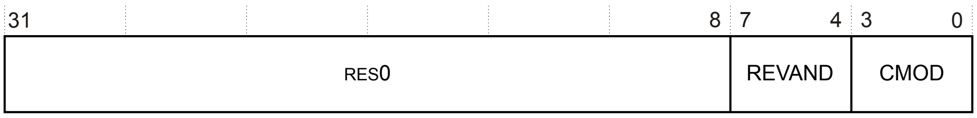

Figure 10-13 EDPIDR3 bit assignments

The following table shows the EDPIDR3 bit assignments.

Table 10-18 EDPIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	REVAND	`0x0` Part minor revision.
[3:0]	CMOD	`0x0` Customer modified. Indicates someone other than Arm Limited has modified the component.

The EDPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

External Debug Peripheral Identification Register 4

The EDPIDR4 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDPIDR4 is in the Debug power domain.

Attributes EDPIDR4 is a 32-bit register.

The following figure shows the EDPIDR4 bit assignments.

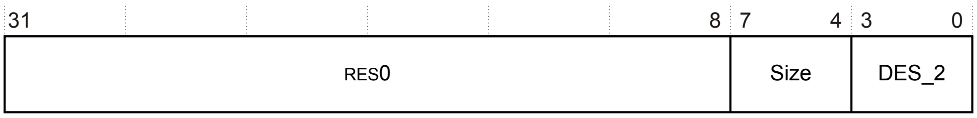

Figure 10-14 EDPIDR4 bit assignments

The following table shows the EDPIDR4 bit assignments.

Table 10-19 EDPIDR4 bit assignments

Bits	Name	Function	
[31:8]	-	Reserved, RES0.	
[7:4]	Size	`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.
[3:0]	DES_2	`0x4`	Arm Limited. This is the least significant nibble of the JEP106 continuation code.

The EDPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

External Debug Peripheral Identification Register 5-7

No information is held in EDPIDR5, EDPIDR6, and EDPIDR7. They are reserved for future use and are RES0.

10.8.5 External Debug Component Identification Registers

There are four read-only External Debug Component Identification Registers. The following table shows these registers.

Table 10-20 Summary of the External Debug Component Identification Registers

Register	Value	Offset
EDCIDR0	`0x0D`	`0xFF0`
EDCIDR1	`0x90`	`0xFF4`
EDCIDR2	`0x05`	`0xFF8`
EDCIDR3	`0xB1`	`0xFFC`

The External Debug Component Identification Registers identify Debug as an *Arm® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2* component. The External Debug Component ID registers are:

- *External Debug Component Identification Register 0* on page 10-424.
- *External Debug Component Identification Register 1* on page 10-425.
- *External Debug Component Identification Register 2* on page 10-425.
- *External Debug Component Identification Register 3* on page 10-426.

External Debug Component Identification Register 0

The EDCIDR0 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDCIDR0 is in the Debug power domain.

Attributes EDCIDR0 is a 32-bit register.

The following figure shows the EDCIDR0 bit assignments.

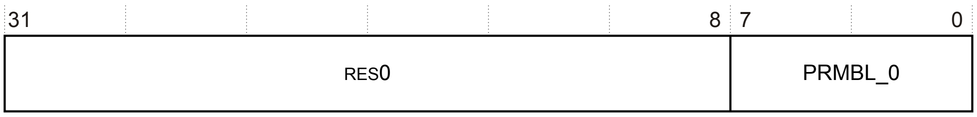

Figure 10-15 EDCIDR0 bit assignments

The following table shows the EDCIDR0 bit assignments.

Table 10-21 EDCIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_0	`0x0D` Preamble byte 0.

The EDCIDR0 can be accessed through the external debug interface, offset `0xFF0`.

External Debug Component Identification Register 1

The EDCIDR1 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDCIDR1 is in the Debug power domain.

Attributes EDCIDR1 is a 32-bit register.

The following figure shows the EDCIDR1 bit assignments.

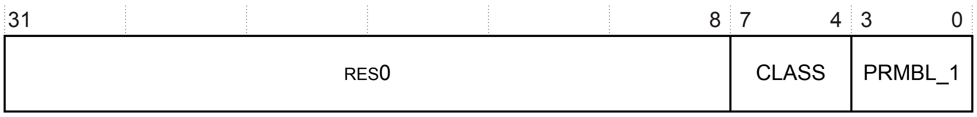

Figure 10-16 EDCIDR1 bit assignments

The following table shows the EDCIDR1 bit assignments.

Table 10-22 EDCIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	CLASS	`0x9` Debug component.
[3:0]	PRMBL_1	`0x0` Preamble byte 1.

The EDCIDR1 can be accessed through the external debug interface, offset `0xFF4`.

External Debug Component Identification Register 2

The EDCIDR2 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDCIDR2 is in the Debug power domain.

Attributes EDCIDR2 is a 32-bit register.

The following figure shows the EDCIDR2 bit assignments.

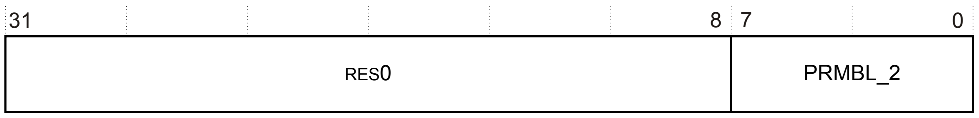

Figure 10-17 EDCIDR2 bit assignments

The following table shows the EDCIDR2 bit assignments.

Table 10-23 EDCIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_2	`0x05` Preamble byte 2.

The EDCIDR2 can be accessed through the external debug interface, offset `0xFF8`.

External Debug Component Identification Register 3

The EDCIDR3 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The EDCIDR3 is in the Debug power domain.

Attributes EDCIDR3 is a 32-bit register.

The following figure shows the EDCIDR3 bit assignments.

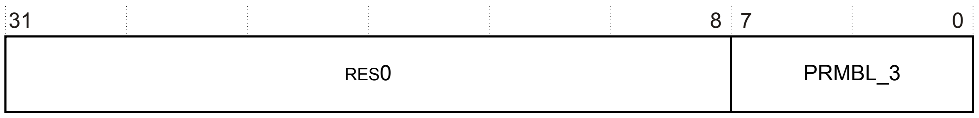

Figure 10-18 EDCIDR3 bit assignments

The following table shows the EDCIDR3 bit assignments.

Table 10-24 EDCIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_3	`0xB1` Preamble byte 3.

The EDCIDR3 can be accessed through the external debug interface, offset `0xFFC`.

10.9 Debug events

A debug event can be either:

- A software debug event.
- 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 on debug events.

This section contains the following subsections:

- *10.9.1 Watchpoint debug events* on page 10-428.
- *10.9.2 Debug OS Lock* on page 10-428.

10.9.1 Watchpoint debug events

In the Cortex-A73 processor, watchpoint debug events are always synchronous. Memory hint instructions and cache clean operations, except `DC ZVA`, `DC IVAC`, and `DC IMVAC`, 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.

For watchpoint debug events, except those resulting from cache maintenance operations, the value reported in DFAR is guaranteed to be no lower than the address of the watchpointed location rounded down to a multiple of 16 bytes.

10.9.2 Debug OS Lock

Debug OS Lock is set by the powerup reset, **nCPUPORESET[CN:0]**. For normal behavior of debug events and debug register accesses, Debug OS Lock must be cleared.

See *2.3.3 Resets* on page 2-37 and the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

10.10 External debug interface

The system can access memory-mapped debug registers through the APB interface. The APB interface is compliant with the AMBA 3 APB v1.0 interface.

The following figure shows the debug interface implemented in the Cortex-A73 processor. For more information on these signals, see the *Arm® CoreSight™ Architecture Specification*.

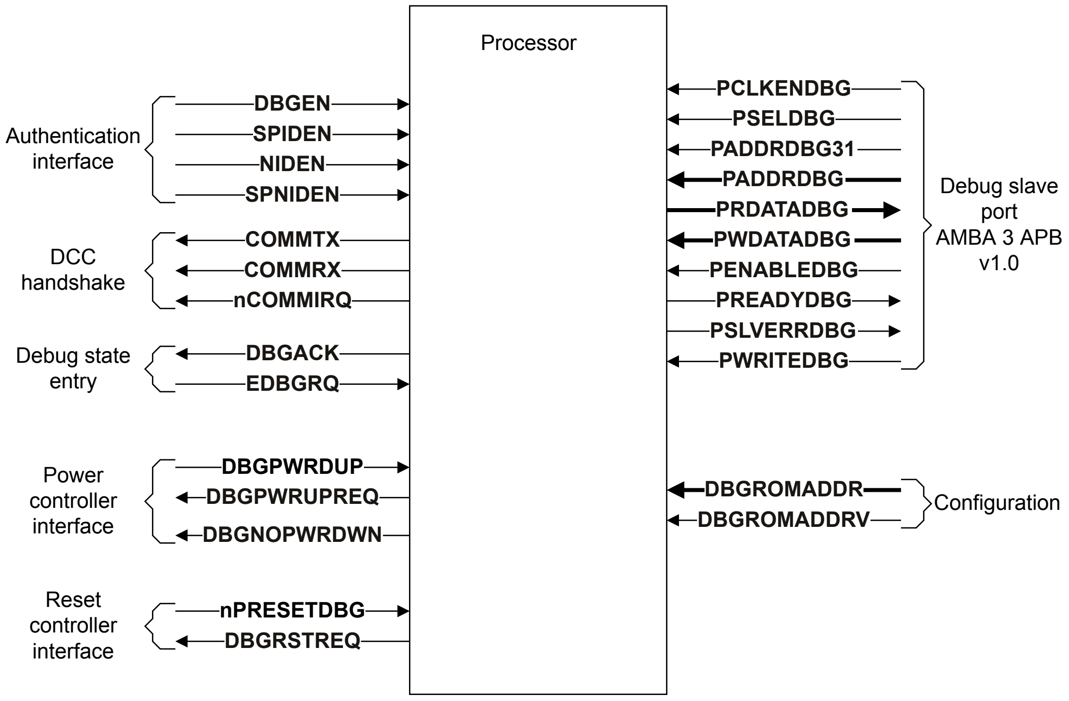

Figure 10-19 External debug interface

This section contains the following subsections:
- *10.10.1 Debug memory map* on page 10-429.
- *10.10.2 DBGPWRDUP debug signal* on page 10-430.
- *10.10.3 DBGL1RSTDISABLE debug signal* on page 10-430.
- *10.10.4 Changing the authentication signals* on page 10-431.

10.10.1 Debug memory map

The basic memory map supports up to four cores in the cluster.

The following table shows the address mapping for the Cortex-A73 processor debug APB components. Each component in the table requires 4KB, and uses the bottom 4KB of each 64KB region. The remaining 60KB of each region is reserved.

Table 10-25 Address mapping for APB components

Address offset [21:0]	Component[de]
`0x000000` - `0x000FFF`	Cortex-A73 APB ROM table
`0x010000` - `0x010FFF`	Core 0 Debug
`0x020000` - `0x020FFF`	Core 0 CTI

[de] Indicates the mapped component if present, otherwise reserved.

Table 10-25 Address mapping for APB components (continued)

Address offset [21:0]	Component[de]
`0x030000` - `0x030FFF`	Core 0 PMU
`0x040000` - `0x040FFF`	Core 0 Trace
`0x041000` - `0x10FFFF`	Reserved
`0x110000` - `0x110FFF`	Core 1 Debug
`0x120000` - `0x120FFF`	Core 1 CTI
`0x130000` - `0x130FFF`	Core 1 PMU
`0x140000` - `0x140FFF`	Core 1 Trace
`0x141000` - `0x20FFFF`	Reserved
`0x210000` - `0x210FFF`	Core 2 Debug
`0x220000` - `0x220FFF`	Core 2 CTI
`0x230000` - `0x230FFF`	Core 2 PMU
`0x240000` - `0x240FFF`	Core 2 Trace
`0x241000` - `0x30FFFF`	Reserved
`0x310000` - `0x310FFF`	Core 3 Debug
`0x320000` - `0x320FFF`	Core 3 CTI
`0x330000` - `0x330FFF`	Core 3 PMU
`0x340000` - `0x340FFF`	Core 3 Trace
`0x341000` - `0x3FFFFF`	Reserved

10.10.2 DBGPWRDUP debug signal

You must set the **DBGPWRDUP[n]** signal LOW before removing power to the core domain. After power is restored to the core domain, the **DBGPWRDUP[n]** signal must be asserted HIGH. The EDPRSR.PU bit reflects the value of this **DBGPWRDUP[n]** signal.

Note

DBGPWRDUP[n] must be tied HIGH if the particular implementation does not support separate core and SCU power domains.

10.10.3 DBGL1RSTDISABLE debug signal

When set HIGH, the DBGL1RSTDISABLE input signal disables the automatic hardware controlled invalidation of the L1 instruction and data caches after the processor is reset, using nCORERESET or nCPUPORESET.

The **DBGL1RSTDISABLE** must be used only to assist debug of an external watchdog triggered reset by allowing the contents of the L1 data cache prior to the reset to be observable after the reset. If reset is asserted, while an L1 data cache eviction or L1 data cache fetch is performed, the accuracy of those cache entries is not guaranteed.

You must not use the **DBGL1RSTDISABLE** signal to disable automatic hardware controlled invalidation of the L1 data cache in normal processor powerup sequences. This is because

[de] Indicates the mapped component if present, otherwise reserved.

synchronization of the L1 data cache invalidation sequence with the duplicate L1 tags in the SCU is not guaranteed.

The **DBGL1RSTDISABLE** signal applies to all cores in the cluster. Each core samples the signal when **nCORERESET** or **nCPUPORESET** is asserted.

If the functionality offered by the **DBGL1RSTDISABLE** input signal is not required, the input must be tied to LOW.

10.10.4 Changing the authentication signals

The **NIDEN[n]**, **DBGEN[n]**, **SPIDEN[n]**, and **SPNIDEN[n]** input signals are either tied off to some fixed value or controlled by some external device.

If software running on the processor has control over an external device that drives the authentication signals, it must make the change using a safe sequence:

1. Execute an implementation-specific sequence of instructions to change the signal value. For example, this might be a single `STR` instruction that writes certain values to a control register in a system peripheral.
2. If the previous step involves any memory operation, issue a `DSB` instruction.
3. Poll the DBGAUTHSTATUS_EL1 to check whether the processor has already detected the changed value of these signals. This is required because the system might not issue the signal change to the processor until several cycles after the `DSB` instruction completes.
4. Issue an `ISB` instruction, an exception entry, or an exception return.

The software cannot perform debug or analysis operations that depend on the new value of the authentication signals until this procedure is complete. The same rules apply when the debugger has control of the processor through the External Debug Instruction Transfer Register, EDITR, while in Debug state. The relevant combinations of the **DBGEN[n]**, **NIDEN[n]**, **SPIDEN[n]**, and **SPNIDEN[n]** values can be determined by polling DBGAUTHSTATUS_EL1.

10.11 ROM table

Debuggers can use the ROM table to determine which components are implemented.

The Cortex-A73 processor includes a ROM table that complies with the *Arm® CoreSight™ Architecture Specification*. This table contains a list of components such as debug units, *Cross Trigger Interfaces* (CTIs), *Performance Monitoring Units* (PMUs) and *Embedded Trace Macrocell* (ETM) units.

If a component is not included in your configuration of the Cortex-A73 processor, the ROMENTRY registers for its debug, CTI, PMU and ETM trace unit components are `0x00000000`.

The interface to the ROM table entries is the APB slave port. See *10.10 External debug interface* on page 10-429 for more information.

This section contains the following subsections:

- *10.11.1 ROM table register summary* on page 10-432.
- *10.11.2 ROM table register descriptions* on page 10-433.
- *10.11.3 ROM table Debug Peripheral Identification Registers* on page 10-435.
- *10.11.4 ROM tables Debug Component Identification Registers* on page 10-438.

10.11.1 ROM table register summary

The following table shows the offsets from the physical base address of the ROM table.

Table 10-26 ROM table registers

Offset	Name	Type	Description
`0x000`	ROMENTRY0	RO	Core 0 Debug, see *ROM entry registers* on page 10-433
`0x004`	ROMENTRY1	RO	Core 0 CTI, see *ROM entry registers* on page 10-433
`0x008`	ROMENTRY2	RO	Core 0 PMU, see *ROM entry registers* on page 10-433
`0x00C`	ROMENTRY3	RO	Core 0 ETM, see *ROM entry registers* on page 10-433
`0x010`	ROMENTRY4	RO	Core 1 Debug, see *ROM entry registers* on page 10-433
`0x014`	ROMENTRY5	RO	Core 1 CTI, see *ROM entry registers* on page 10-433
`0x018`	ROMENTRY6	RO	Core 1 PMU, see *ROM entry registers* on page 10-433
`0x01C`	ROMENTRY7	RO	Core 1 ETM, see *ROM entry registers* on page 10-433
`0x020`	ROMENTRY8	RO	Core 2 Debug, see *ROM entry registers* on page 10-433
`0x024`	ROMENTRY9	RO	Core 2 CTI, see *ROM entry registers* on page 10-433
`0x028`	ROMENTRY10	RO	Core 2 PMU, see *ROM entry registers* on page 10-433
`0x02C`	ROMENTRY11	RO	Core 2 ETM, see *ROM entry registers* on page 10-433
`0x030`	ROMENTRY12	RO	Core 3 Debug, see *ROM entry registers* on page 10-433
`0x034`	ROMENTRY13	RO	Core 3 CTI, see *ROM entry registers* on page 10-433
`0x038`	ROMENTRY14	RO	Core 3 PMU, see *ROM entry registers* on page 10-433
`0x03C`	ROMENTRY15	RO	Core 3 ETM, see *ROM entry registers* on page 10-433
`0x040-0xFCC`	-	RO	Reserved, RES0
`0xFD0`	ROMPIDR4	RO	*ROM table Debug Peripheral Identification Register 4* on page 10-438
`0xFD4`	ROMPIDR5	RO	*ROM table Debug Peripheral Identification Register 5-7* on page 10-438
`0xFD8`	ROMPIDR6	RO	*ROM table Debug Peripheral Identification Register 5-7* on page 10-438

Table 10-26 ROM table registers (continued)

Offset	Name	Type	Description
0xFDC	ROMPIDR7	RO	*ROM table Debug Peripheral Identification Register 5-7* on page 10-438
0xFE0	ROMPIDR0	RO	*ROM table Debug Peripheral Identification Register 0* on page 10-435
0xFE4	ROMPIDR1	RO	*ROM table Debug Peripheral Identification Register 1* on page 10-436
0xFE8	ROMPIDR2	RO	*ROM table Debug Peripheral Identification Register 2* on page 10-436
0xFEC	ROMPIDR3	RO	*ROM table Debug Peripheral Identification Register 3* on page 10-437
0xFF0	ROMCIDR0	RO	*ROM table Debug Component Identification Register 0* on page 10-439
0xFF4	ROMCIDR1	RO	*ROM table Debug Component Identification Register 1* on page 10-439
0xFF8	ROMCIDR2	RO	*ROM table Debug Component Identification Register 2* on page 10-440
0xFFC	ROMCIDR3	RO	*ROM table Debug Component Identification Register 3* on page 10-440

10.11.2 ROM table register descriptions

This section describes the ROM table registers.

10.11.1 ROM table register summary on page 10-432 provides cross-references to individual registers.

ROM entry registers

The characteristics of the ROMENTRY*n* registers are:

Purpose Indicates to a debugger whether the debug component is present in the core debug logic. There are 16 ROMENTRY registers in the Cortex-A73 processor.

Usage constraints These registers are accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations There is one copy of these registers that is used in both Secure and Non-secure states.

Attributes ROMENTRY*n* are 32-bit registers.

The following figure shows the bit assignments for a ROMENTRY register.

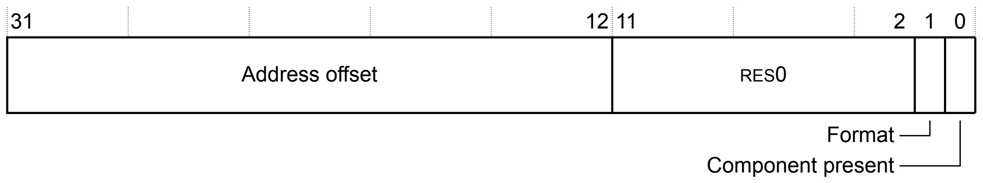

Figure 10-20 ROMENTRY bit assignments

The following table shows the bit assignments for a ROMENTRY register.

Table 10-27 ROMENTRY bit assignments

Bits	Name	Function
[31:12]	Address offset	Address offset for the debug component. **Note** Negative values of address offsets are permitted using the two's complement of the offset.
[11:2]	-	Reserved, RES0.
[1]	Format	Format of the ROM table entry. The value for all ROMENTRY registers is: `0` End marker. `1` 32-bit format.
[0]	Component present[df]	Indicates whether the component is present: `0` Component is not present. `1` Component is present.

The Physical Address of a debug component is determined by shifting the address offset 12 places to the left and adding the result to the Physical Address of the Cortex-A73 processor ROM table.

If a core is not implemented, the ROMENTRY registers for its debug, CTI, PMU and ETM trace unit components are `0x00000000` when a v8 memory map is implemented.

Table 10-28 v8 ROMENTRY values

Name	Debug component	Address offset[31:12]	ROMENTRY value
ROMENTRY0	Core 0 Debug	`0x00010`	`0x00010003`
ROMENTRY1	Core 0 CTI	`0x00020`	`0x00020003`
ROMENTRY2	Core 0 PMU	`0x00030`	`0x00030003`
ROMENTRY3	Core 0 ETM trace unit	`0x00040`	`0x00040003`
ROMENTRY4	Core 1 Debug	`0x00110`	`0x00110003`[dg]
ROMENTRY5	Core 1 CTI	`0x00120`	`0x00120003`[dg]
ROMENTRY6	Core 1 PMU	`0x00130`	`0x00130003`[dg]
ROMENTRY7	Core 1 ETM trace unit	`0x00140`	`0x00140003`[dg]
ROMENTRY8	Core 2 Debug	`0x00210`	`0x00210003`[dg]
ROMENTRY9	Core 2 CTI	`0x00220`	`0x00220003`[dg]
ROMENTRY10	Core 2 PMU	`0x00230`	`0x00230003`[dg]
ROMENTRY11	Core 2 ETM trace unit	`0x00240`	`0x00240003`[dg]
ROMENTRY12	Core 3 Debug	`0x00310`	`0x00310003`[dg]
ROMENTRY13	Core 3 CTI	`0x00320`	`0x00320003`[dg]
ROMENTRY14	Core 3 PMU	`0x00330`	`0x00330003`[dg]
ROMENTRY15	Core 3 ETM trace unit	`0x00340`	`0x00340003`[dg]

[df] The components for Core 0 are always present. The entries for Core 1, 2, and 3 components depend on your configuration.

10.11.3 ROM table Debug Peripheral Identification Registers

The ROM table Debug Peripheral Identification Registers provide standard information required for all components that conform to the *Arm® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2*. There is a set of eight registers, listed in register number order in the following table.

Table 10-29 Summary of the ROM table Debug Peripheral Identification Registers

Register	Value	Offset
ROMPIDR4	`0x04`	`0xFD0`
ROMPIDR5	`0x00`	`0xFD4`
ROMPIDR6	`0x00`	`0xFD8`
ROMPIDR7	`0x00`	`0xFDC`
ROMPIDR0	`0xA6`	`0xFE0`
ROMPIDR1	`0xB4`	`0xFE4`
ROMPIDR2	`0x0B`	`0xFE8`
ROMPIDR3	`0x00`	`0xFEC`

Only bits[7:0] of each ROM table Debug Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight ROM table Debug Peripheral ID Registers define a single 64-bit Peripheral ID.

The ROM table Debug Peripheral ID registers are:
- *ROM table Debug Peripheral Identification Register 0* on page 10-435.
- *ROM table Debug Peripheral Identification Register 1* on page 10-436.
- *ROM table Debug Peripheral Identification Register 2* on page 10-436.
- *ROM table Debug Peripheral Identification Register 3* on page 10-437.
- *ROM table Debug Peripheral Identification Register 4* on page 10-438.
- *ROM table Debug Peripheral Identification Register 5-7* on page 10-438.

ROM table Debug Peripheral Identification Register 0

The ROMPIDR0 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The ROMPIDR0 is in the Debug power domain.

Attributes ROMPIDR0 is a 32-bit register.

The following figure shows the ROMPIDR0 bit assignments.

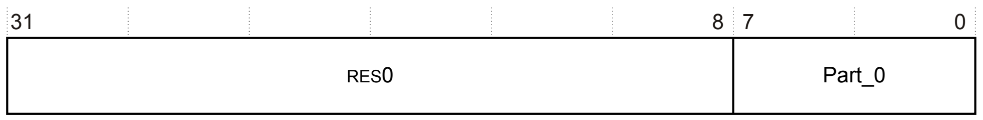

Figure 10-21 ROMPIDR0 bit assignments

The following table shows the ROMPIDR0 bit assignments.

dg If the component is present.

Table 10-30 ROMPIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	Part_0	`0xA6` Least significant byte of the ROM table part number.

The ROMPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

ROM table Debug Peripheral Identification Register 1

The ROMPIDR1 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The ROMPIDR1 is in the Debug power domain.

Attributes ROMPIDR1 is a 32-bit register.

The following figure shows the ROMPIDR1 bit assignments.

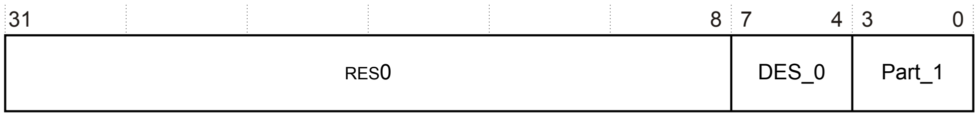

Figure 10-22 ROMPIDR1 bit assignments

The following table shows the ROMPIDR1 bit assignments.

Table 10-31 ROMPIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	DES_0	`0xB` Arm Limited. Least significant nibble of JEP106 ID code.
[3:0]	Part_1	`0x4` Most significant nibble of the ROM table part number.

The ROMPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

ROM table Debug Peripheral Identification Register 2

The ROMPIDR2 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The ROMPIDR2 is in the Debug power domain.

Attributes ROMPIDR2 is a 32-bit register.

The following figure shows the ROMPIDR2 bit assignments.

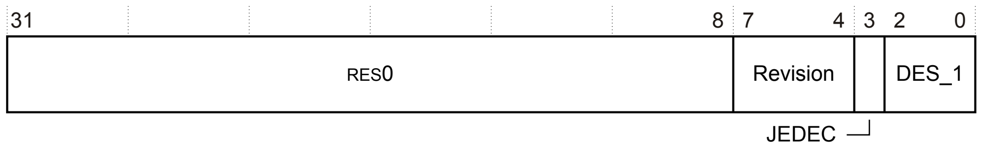

Figure 10-23 ROMPIDR2 bit assignments

The following table shows the ROMPIDR2 bit assignments.

Table 10-32 ROMPIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Revision	`0x0` r0p0.
[3]	JEDEC	`0b1` Indicates a JEP106 identity code is used.
[2:0]	DES_1	`0b011` Arm Limited. Most significant bits of JEP106 ID code.

The ROMPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

ROM table Debug Peripheral Identification Register 3

The ROMPIDR3 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The ROMPIDR3 is in the Debug power domain.

Attributes ROMPIDR3 is a 32-bit register.

The following figure shows the ROMPIDR3 bit assignments.

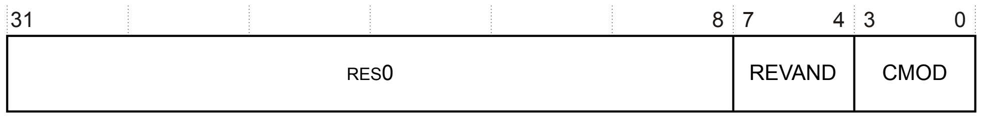

Figure 10-24 ROMPIDR3 bit assignments

The following table shows the ROMPIDR3 bit assignments.

Table 10-33 ROMPIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	REVAND	`0x0` Part minor revision.
[3:0]	CMOD	`0x0` Customer modified.

The ROMPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

ROM table Debug Peripheral Identification Register 4

The ROMPIDR4 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The ROMPIDR4 is in the Debug power domain.

Attributes ROMPIDR4 is a 32-bit register.

The following figure shows the ROMPIDR4 bit assignments.

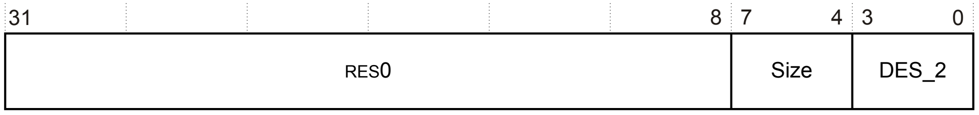

Figure 10-25 ROMPIDR4 bit assignments

The following table shows the ROMPIDR4 bit assignments.

Table 10-34 ROMPIDR4 bit assignments

Bits	Name	Function	
[31:8]	-	Reserved, RES0.	
[7:4]	Size	`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.
[3:0]	DES_2	`0x4`	Arm Limited. Least significant nibble of the JEP106 continuation code.

The ROMPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

ROM table Debug Peripheral Identification Register 5-7

No information is held in ROMPIDR5, ROMPIDR6, and ROMPIDR7. They are reserved for future use and are RES0.

10.11.4 ROM tables Debug Component Identification Registers

There are four read-only ROM table Debug Component Identification Registers. The following table shows these registers.

Table 10-35 Summary of the ROM table Debug Component Identification registers

Register	Value	Offset
ROMCIDR0	`0x0D`	`0xFF0`
ROMCIDR1	`0x10`	`0xFF4`
ROMCIDR2	`0x05`	`0xFF8`
ROMCIDR3	`0xB1`	`0xFFC`

The ROM table Debug Component Identification Registers identify Debug as an *Arm® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2* component.

ROM table Debug Component Identification Register 0

The ROMCIDR0 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The ROMCIDR0 is in the Debug power domain.

Attributes ROMCIDR0 is a 32-bit register.

The following figure shows the ROMCIDR0 bit assignments.

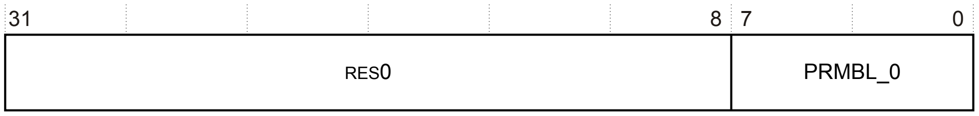

Figure 10-26 ROMCIDR0 bit assignments

The following table shows the ROMCIDR0 bit assignments.

Table 10-36 ROMCIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_0	`0x0D` Preamble byte 0.

The ROMCIDR0 can be accessed through the external debug interface, offset `0xFF0`.

ROM table Debug Component Identification Register 1

The ROMCIDR1 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The ROMCIDR1 is in the Debug power domain.

Attributes ROMCIDR1 is a 32-bit register.

The following figure shows the ROMCIDR1 bit assignments.

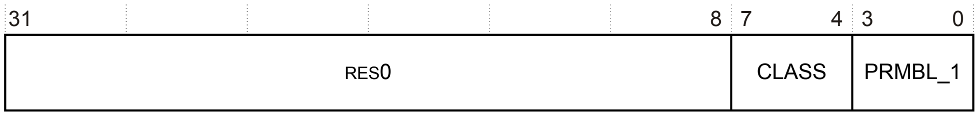

Figure 10-27 ROMCIDR1 bit assignments

The following table shows the ROMCIDR1 bit assignments.

Table 10-37 ROMCIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	CLASS	`0x1` Component Class. For a ROM table.
[3:0]	PRMBL_1	`0x0` Preamble byte 1.

The ROMCIDR1 can be accessed through the external debug interface, offset `0xFF4`.

ROM table Debug Component Identification Register 2

The ROMCIDR2 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The ROMCIDR2 is in the Debug power domain.

Attributes ROMCIDR2 is a 32-bit register.

The following figure shows the ROMCIDR2 bit assignments.

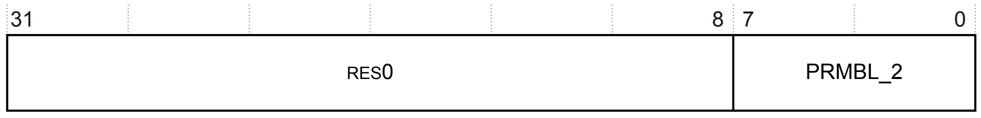

Figure 10-28 ROMCIDR2 bit assignments

The following table shows the ROMCIDR2 bit assignments.

Table 10-38 ROMCIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_2	`0x05` Preamble byte 2.

The ROMCIDR2 can be accessed through the external debug interface, offset `0xFF8`.

ROM table Debug Component Identification Register 3

The ROMCIDR3 characteristics are:

Purpose Provides information to identify an external debug component.

Usage constraints This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Configurations The ROMCIDR3 is in the Debug power domain.

Attributes ROMCIDR3 is a 32-bit register.

The following figure shows the ROMCIDR3 bit assignments.

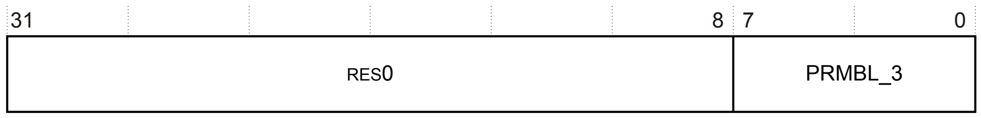

Figure 10-29 ROMCIDR3 bit assignments

The following table shows the ROMCIDR3 bit assignments.

Table 10-39 ROMCIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_3	`0xB1` Preamble byte 3.

The ROMCIDR3 can be accessed through the external debug interface, offset `0xFFC`.

Chapter 11
Performance Monitor Unit

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

It contains the following sections:

11.1 About the PMU

The Cortex-A73 processor includes performance monitors that implement the Arm PMUv3 architecture. These 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 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.

11.2 PMU functional description

The following figure shows the major blocks inside the PMU.

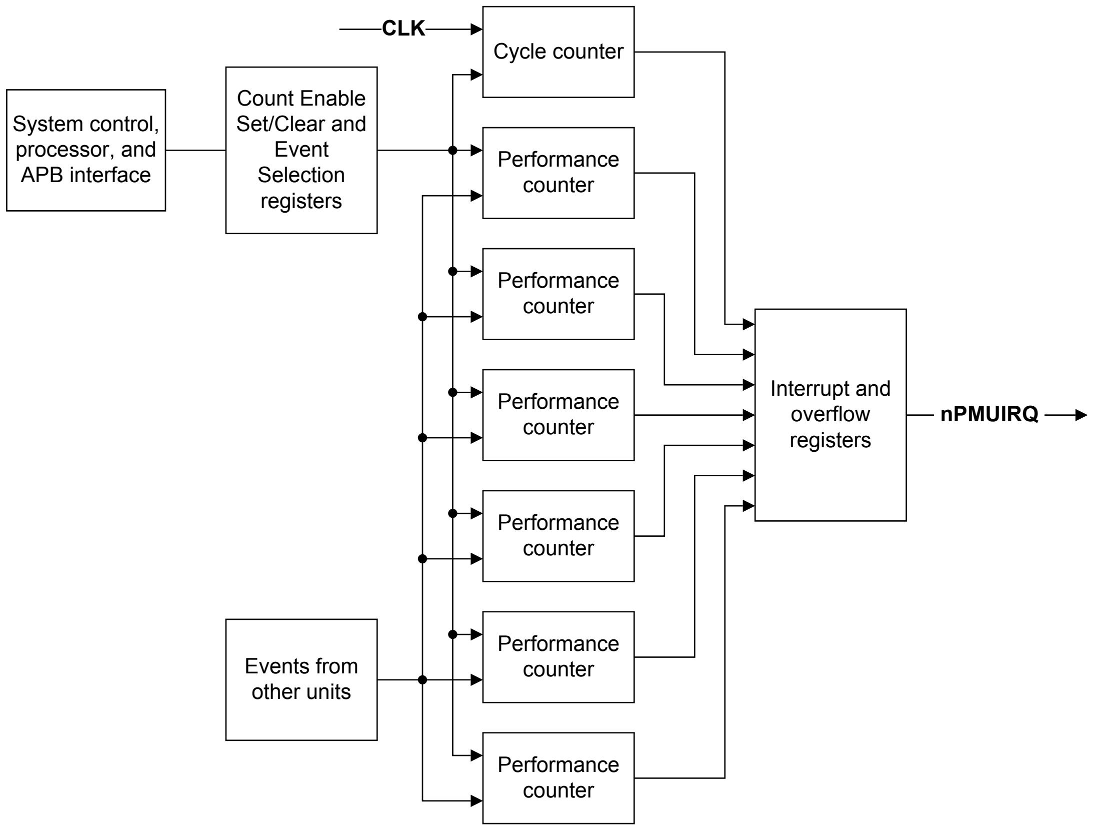

Figure 11-1 PMU block diagram

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 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-A73 processor supports access to the performance monitor registers from the internal system register. External access to the performance monitor registers is also provided with the APB slave port or external debug interface. See *10.10 External debug interface* on page 10-429.

This section contains the following subsection:
- *11.2.1 External register access permissions* on page 11-444.

11.2.1 External register access permissions

Whether or not access is permitted to a register depends on:

- If the processor is powered up.
- The state of the OS Lock, OS Double Lock, and Software Lock.
- The state of the debug authentication inputs to the processor.

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 Armv8, for Armv8-A architecture profile*.

The register descriptions provided in this section does describe whether each register is read/write or read-only.

11.3 AArch64 PMU register summary

The 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-A73 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 11-1 PMU register summary in AArch64 state

Name	Type	Width	Description
PMCR_EL0	RW	32-bit	*11.4.1 Performance Monitors Control Register* on page 11-448
PMCNTENSET_EL0	RW	32-bit	Performance Monitors Count Enable Set Register
PMCNTENCLR_EL0	RW	32-bit	Performance Monitors Count Enable Clear Register
PMOVSCLR_EL0	RW	32-bit	Performance Monitors Overflow Flag Status Register
PMSWINC_EL0	WO	32-bit	Performance Monitors Software Increment Register
PMSELR_EL0	RW	32-bit	Performance Monitors Event Counter Selection Register
PMCEID0_EL0	RO	32-bit	*11.4.2 Performance Monitors Common Event Identification Register 0* on page 11-450
PMCEID1_EL0	RO	32-bit	*11.4.3 Performance Monitors Common Event Identification Register 1* on page 11-453
PMCCNTR_EL0	RW	64-bit	Performance Monitors Cycle Count Register
PMXEVTYPER_EL0	RW	32-bit	Performance Monitors Selected Event Type and Filter Register
PMCCFILTR_EL0	RW	32-bit	Performance Monitors Cycle Count Filter Register
PMXEVCNTR0_EL0	RW	32-bit	Performance Monitors Selected Event Count Register
PMUSERENR_EL0	RW	32-bit	Performance Monitors User Enable Register
PMINTENSET_EL1	RW	32-bit	Performance Monitors Interrupt Enable Set Register
PMINTENCLR_EL1	RW	32-bit	Performance Monitors Interrupt Enable Clear Register

Table 11-1 PMU register summary in AArch64 state (continued)

Name	Type	Width	Description
PMOVSSET_EL0	RW	32-bit	Performance Monitors Overflow Flag Status Set Register
PMEVCNTR0_EL0	RW	32-bit	Performance Monitors Event Count Registers
PMEVCNTR1_EL0	RW	32-bit	
PMEVCNTR2_EL0	RW	32-bit	
PMEVCNTR3_EL0	RW	32-bit	
PMEVCNTR4_EL0	RW	32-bit	
PMEVCNTR5_EL0	RW	32-bit	
PMEVTYPER0_EL0	RW	32-bit	Performance Monitors Event Type Registers
PMEVTYPER1_EL0	RW	32-bit	
PMEVTYPER2_EL0	RW	32-bit	
PMEVTYPER3_EL0	RW	32-bit	
PMEVTYPER4_EL0	RW	32-bit	
PMEVTYPER5_EL0	RW	32-bit	
PMCCFILTR_EL0	RW	32-bit	Performance Monitors Cycle Count Filter Register

11.4 AArch64 PMU register descriptions

This section describes the Cortex-A73 processor PMU registers in AArch64 state.

11.3 AArch64 PMU register summary on page 11-446 provides cross-references to individual registers.

This section contains the following subsections:

11.4.1 Performance Monitors Control Register

The PMCR_EL0 characteristics are:

Purpose Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Usage constraints 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

This register is accessible at EL0 when PMUSERENR_EL0.EN is set to 1.

Configurations PMCR_EL0 is architecturally mapped to the AArch32 register PMCR. See *11.6.1 Performance Monitors Control Register* on page 11-457.

Attributes PMCR_EL0 is a 32-bit register.

The following figure shows the PMCR_EL0 bit assignments.

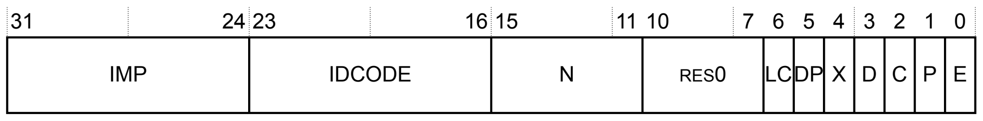

Figure 11-2 PMCR_EL0 bit assignments

The following table shows the PMCR_EL0 bit assignments.

Table 11-2 PMCR_EL0 bit assignments

Bits	Name	Function
[31:24]	IMP	Implementer code. This value is: `0x41` Arm Limited. This is a read-only field.
[23:16]	IDCODE	Identification code. This value is: `0x04` Cortex-A73 processor. This is a read-only field.
[15:11]	N	Number of event counters. This value is: `0x6` Six counters are implemented.
[10:7]	-	Reserved, RES0.

Table 11-2 PMCR_EL0 bit assignments (continued)

Bits	Name	Function
[6]	LC	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`. This bit resets to 0.
[5]	DP	Disable cycle counter when event counting is prohibited. The possible values of this bit are: `0` PMCCNTR_EL0, if enabled, counts when event counting is prohibited. `1` PMCCNTR_EL0 does not count when event counting is prohibited. This bit is read/write and resets to 0.
[4]	X	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 of this bit are: `0` Do not export events. `1` Export events where not prohibited. This bit is read/write and does not affect the generation of Performance Monitors interrupts on the **nPMUIRQ** pin. This bit resets to 0.
[3]	D	Clock divider. The possible values of this bit are: `0` When enabled, PMCCNTR_EL0 counts every clock cycle. `1` When enabled, PMCCNTR_EL0 counts every 64 clock cycles. This bit is read/write and resets to 0.
[2]	C	Clock counter reset. This bit is WO. The effects of writing to this bit are: `0` No action. `1` Reset PMCCNTR_EL0 to `0`. This bit is always RES0 and resets to 0. **Note** Resetting PMCCNTR 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.

Table 11-2 PMCR_EL0 bit assignments (continued)

Bits	Name	Function
[1]	P	Event counter reset. This bit is WO. The effects of writing to this bit are: `0` No action. `1` Reset all event counters, not including PMCCNTR_EL0, to `0`. This bit is always RES0. 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`. This bit resets to 0.
[0]	E	Enable. The possible values of this bit are: `0` All counters, including PMCCNTR_EL0, are disabled. `1` All counters are enabled by PMCNTENSET_EL0. 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. This bit resets to 0.

To access the PMCR_EL0:

```
MRS <Xt>, PMCR_EL0 ; Read PMCR_EL0 into Xt
MSR PMCR_EL0, <Xt> ; Write Xt to PMCR_EL0
```

To access the PMCR in AArch32 state, read or write the CP15 registers with:

```
MRC p15, 0, <Rt>, c9, c12, 0; Read Performance Monitor Control Register
MCR p15, 0, <Rt>, c9, c12, 0; Write Performance Monitor Control Register
```

The PMCR_EL0 can be accessed through the external debug interface, offset `0xE04`.

11.4.2 Performance Monitors Common Event Identification Register 0

The PMCEID0_EL0 register characteristics are:

Purpose Defines which common architectural and common microarchitectural feature events are implemented.

Usage constraints This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
Config	RO	RO	RO	RO	RO

This register is accessible at EL0 when PMUSERENR_EL0.EN is set to `1`.

Configurations The PMCEID0_EL0 register is architecturally mapped to:

- The AArch32 register PMCEID0. See *11.6.2 Performance Monitors Common Event Identification Register 0* on page 11-459.
- The external register PMCEID0_EL0.

Attributes PMCEID0_EL0 is a 32-bit register.

The following figure shows the PMCEID0_EL0 bit assignments.

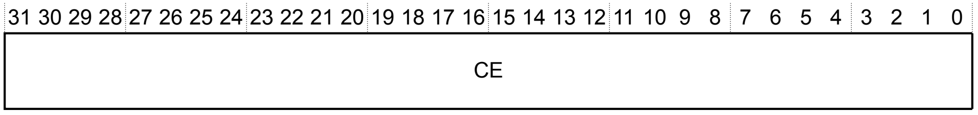

Figure 11-3 PMCEID0_EL0 bit assignments

The following table shows the PMCEID0_EL0 bit assignments

Table 11-3 PMCEID0_EL0 bit assignments

Bits	Name	Function
[31:0]	CE[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 11-4 PMU common events

Bit	Event number	Event mnemonic	Description
[31]	`0x1F`	L1D_CACHE_ALLOCATE	L1 data cache allocate: `0` This event is not implemented.
[30]	`0x1E`	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]	`0x1D`	BUS_CYCLES	Bus cycle: `1` This event is implemented.
[28]	`0x1C`	TTBR_WRITE_RETIRED	TTBR write, architecturally executed, condition check pass - write to translation table base: `1` This event is implemented.
[27]	`0x1B`	INST_SPEC	Instruction speculatively executed: `1` This event is implemented.
[26]	`0x1A`	MEMORY_ERROR	Local memory error: `0` This event is not implemented.
[25]	`0x19`	BUS_ACCESS	Bus access: `1` This event is implemented.
[24]	`0x18`	L2D_CACHE_WB	L2 data cache Write-Back: `1` This event is implemented.

Table 11-4 PMU common events (continued)

Bit	Event number	Event mnemonic	Description
[23]	`0x17`	L2D_CACHE_REFILL	L2 data cache refill: `1` This event is implemented.
[22]	`0x16`	L2D_CACHE	L2 data cache access: `1` This event is implemented.
[21]	`0x15`	L1D_CACHE_WB	L1 data cache Write-Back: `1` This event is implemented.
[20]	`0x14`	L1I_CACHE	L1 instruction cache access: `1` This event is implemented.
[19]	`0x13`	MEM_ACCESS	Data memory access: `1` This event is implemented.
[18]	`0x12`	BR_PRED	Predictable branch speculatively executed: `1` This event is implemented.
[17]	`0x11`	CPU_CYCLES	Cycle: `1` This event is implemented.
[16]	`0x10`	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed: `1` This event is implemented.
[15]	`0x0F`	UNALIGNED_LDST_RETIRED	Instruction architecturally executed, condition check pass - unaligned load or store: `0` This event is not implemented.
[14]	`0x0E`	BR_RETURN_RETIRED	Instruction architecturally executed, condition check pass - procedure return: `1` This event is implemented.
[13]	`0x0D`	BR_IMMED_RETIRED	Instruction architecturally executed - immediate branch: `1` This event is implemented.
[12]	`0x0C`	PC_WRITE_RETIRED	Instruction architecturally executed, condition check pass - software change of the PC: `1` This event is implemented.
[11]	`0x0B`	CID_WRITE_RETIRED	Instruction architecturally executed, condition check pass - write to CONTEXTIDR: `1` This event is implemented.

Table 11-4 PMU common events (continued)

Bit	Event number	Event mnemonic	Description
[10]	`0x0A`	EXC_RETURN	Instruction architecturally executed, condition check pass - exception return: `1` This event is implemented.
[9]	`0x09`	EXC_TAKEN	Exception taken: `1` This event is implemented.
[8]	`0x08`	INST_RETIRED	Instruction architecturally executed: `1` This event is implemented.
[7]	`0x07`	ST_RETIRED	Instruction architecturally executed, condition check pass - store: `0` This event is not implemented.
[6]	`0x06`	LD_RETIRED	Instruction architecturally executed, condition check pass - load: `0` This event is not implemented.
[5]	`0x05`	L1D_TLB_REFILL	L1 Data TLB refill: `1` This event is implemented.
[4]	`0x04`	L1D_CACHE	L1 data cache access: `1` This event is implemented.
[3]	`0x03`	L1D_CACHE_REFILL	L1 data cache refill: `1` This event is implemented.
[2]	`0x02`	L1I_TLB_REFILL	L1 instruction TLB refill: `1` This event is implemented.
[1]	`0x01`	L1I_CACHE_REFILL	L1 instruction cache refill: `1` This event is implemented.
[0]	`0x00`	SW_INCR	Instruction architecturally executed, condition check pass - software increment: `1` This event is implemented.

To access the PMCEID0_EL0 register in AArch64 state, read or write the register with:

```
MRS <Xt>, PMCEID0_EL0; Read Performance Monitor Common Event Identification Register 0
```

To access the PMCEID0 register in AArch32 state, read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c9, c12, 6; Read Performance Monitor Common Event Identification Register 0
```

The PMCEID0_EL0 register can be accessed through the external debug interface, offset `0xE20`.

11.4.3 Performance Monitors Common Event Identification Register 1

The PMCEID1_EL0 register characteristics are:

Purpose
Defines which common architectural and common microarchitectural feature events are implemented.

Usage constraints
This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
Config	RO	RO	RO	RO	RO

This register is accessible at EL0 when PMUSERENR_EL0.EN is set to `1`.

Configurations
The PMCEID1_EL0 register is architecturally mapped to:
- The AArch32 register PMCEID1. See *11.6.3 Performance Monitors Common Event Identification Register 1* on page 11-462.
- The external register PMCEID1_EL0.

Attributes
PMCEID1_EL0 is a 32-bit register.

The following figure shows the PMCEID1_EL0 bit assignments.

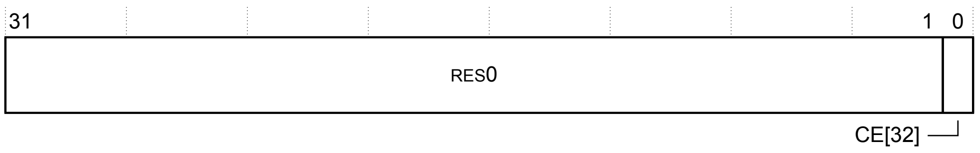

Figure 11-4 PMCEID1_EL0 bit assignments

The following table shows the PMCEID1_EL0 bit assignments.

Table 11-5 PMCEID1 bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	CE[32]	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 11-6 PMU common events

Bit	Event number	Event mnemonic	Description
[0]	`0x20`	L2D_CACHE_ALLOCATE	`0` This event is not implemented.

To access the PMCEID1_EL0 register:

```
MRS <Xt>, PMCEID1_EL0; Read Performance Monitor Common Event Identification Register 1
```

The PMCEID1_EL0 register can be accessed through the external debug interface, offset `0xE24`.

11.5 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-A73 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*.

See the *Table 11-13 Memory-mapped PMU register summary* on page 11-464 for a complete list of registers that are accessible from the external debug interface.

Table 11-7 PMU register summary in AArch32 state

CRn	Op1	CRm	Op2	Name	Type	Width	Description
c9	0	c12	0	PMCR	RW	32-bit	*11.6.1 Performance Monitors Control Register* on page 11-457
c9	0	c12	1	PMCNTENSET	RW	32-bit	Performance Monitors Count Enable Set Register
c9	0	c12	2	PMCNTENCLR	RW	32-bit	Performance Monitors Count Enable Clear Register
c9	0	c12	3	PMOVSR	RW	32-bit	Performance Monitors Overflow Flag Status Register
c9	0	c12	4	PMSWINC	WO	32-bit	Performance Monitors Software Increment Register
c9	0	c12	5	PMSELR	RW	32-bit	Performance Monitors Event Counter Selection Register
c9	0	c12	6	PMCEID0	RO	32-bit	*11.6.2 Performance Monitors Common Event Identification Register 0* on page 11-459
c9	0	c12	7	PMCEID1	RO	32-bit	*11.6.3 Performance Monitors Common Event Identification Register 1* on page 11-462
c9	0	c13	0	PMCCNTR[31:0]	RW	32-bit	Performance Monitors Cycle Count Register
-	0	c9	-	PMCCNTR[63:0]	RW	64-bit	
c9	0	c13	1	PMXEVTYPER	RW	32-bit	Performance Monitors Selected Event Type Register
				PMCCFILTR	RW	32-bit	Performance Monitors Cycle Count Filter Register
c9	0	c13	2	PMXEVCNTR	RW	32-bit	Performance Monitors Selected Event Count Register
c9	0	c14	0	PMUSERENR	RW	32-bit	Performance Monitors User Enable Register
c9	0	c14	1	PMINTENSET	RW	32-bit	Performance Monitors Interrupt Enable Set Register
c9	0	c14	2	PMINTENCLR	RW	32-bit	Performance Monitors Interrupt Enable Clear Register
c9	0	c14	3	PMOVSSET	RW	32-bit	Performance Monitor Overflow Flag Status Set Register
c14	0	c8	0	PMEVCNTR0	RW	32-bit	Performance Monitor Event Count Registers
c14	0	c8	1	PMEVCNTR1	RW	32-bit	
c14	0	c8	2	PMEVCNTR2	RW	32-bit	
c14	0	c8	3	PMEVCNTR3	RW	32-bit	
c14	0	c8	4	PMEVCNTR4	RW	32-bit	
c14	0	c8	5	PMEVCNTR5	RW	32-bit	

Table 11-7 PMU register summary in AArch32 state (continued)

CRn	Op1	CRm	Op2	Name	Type	Width	Description
c14	0	c12	0	PMEVTYPER0	RW	32-bit	Performance Monitors Event Type Registers
c14	0	c12	1	PMEVTYPER1	RW	32-bit	
c14	0	c12	2	PMEVTYPER2	RW	32-bit	
c14	0	c12	3	PMEVTYPER3	RW	32-bit	
c14	0	c12	4	PMEVTYPER4	RW	32-bit	
c14	0	c12	5	PMEVTYPER5	RW	32-bit	
c14	0	c15	7	PMCCFILTR	RW	32-bit	Performance Monitors Cycle Count Filter Register

11.6 AArch32 PMU register descriptions

This section describes the processor PMU registers in AArch32 state.

11.5 AArch32 PMU register summary on page 11-455 provides cross-references to individual registers.

This section contains the following subsections:

- *11.6.1 Performance Monitors Control Register* on page 11-457.
- *11.6.2 Performance Monitors Common Event Identification Register 0* on page 11-459.
- *11.6.3 Performance Monitors Common Event Identification Register 1* on page 11-462.

11.6.1 Performance Monitors Control Register

The PMCR characteristics are:

Purpose Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Usage constraints 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	Config-RW	RW	RW	RW	RW	RW

This register is accessible at EL0 when PMUSERENR_EL0.EN is set to 1.

Configurations PMCR is architecturally mapped to:

- The AArch64 register PMCR_EL0. See *11.4.1 Performance Monitors Control Register* on page 11-448.
- The external PMCR_EL0 register.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes PMCR is a 32-bit register.

The following figure shows the PMCR bit assignments.

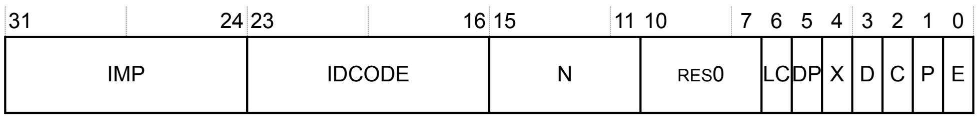

Figure 11-5 PMCR bit assignments

The following table shows the PMCR bit assignments.

Table 11-8 PMCR bit assignments

Bits	Name	Function
[31:24]	IMP	Implementer code. This value is: `0x41` Arm Limited. This is a read-only field.
[23:16]	IDCODE	Identification code. This value is: `0x04` Cortex-A73 processor. This is a read-only field.

Table 11-8 PMCR bit assignments (continued)

Bits	Name	Function
[15:11]	N	Number of event counters. `0x6` Six counters are implemented.
[10:7]	-	Reserved, RES0.
[6]	LC	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`. This bit is read/write.
[5]	DP	Disable cycle counter when event counting is prohibited. The possible values of this bit are: `0` PMCCNTR, if enabled, counts when event counting is prohibited. `1` PMCCNTR does not count when event counting is prohibited. This bit is read/write.
[4]	X	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: `0` Do not export events. `1` Export events where not prohibited. This bit is read/write and does not affect the generation of Performance Monitors interrupts on the **nPMUIRQ** pin.
[3]	D	Clock divider. The possible values of this bit are: `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.
[2]	C	Clock counter reset. This bit is WO. The effects of writing to this bit are: `0` No action. `1` Reset PMCCNTR_EL0 to 0. This bit is always RES0. ——— **Note** ——— Resetting PMCCNTR 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. ———————

Table 11-8 PMCR bit assignments (continued)

Bits	Name	Function
[1]	P	Event counter reset. This bit is WO. The effects of writing to this bit are: `0` No action. `1` Reset all event counters, not including PMCCNTR_EL0, to `0`. This bit is always RES0. 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`.
[0]	E	Enable. The possible values of this bit are: `0` All counters, including PMCCNTR_EL0, are disabled. `1` All counters are enabled. This bit is read/write. In Non-secure EL0 and EL1, this bit does not affect the operation of event counters that MDCR_EL2.HPMN reserves for EL2 use.

To access the PMCR:

```
MRC p15, 4, <Rt>, c12, c0, 0 ; Read PMCR into Rt
MCR p15, 4, <Rt>, c12, c0, 0 ; Write Rt to PMCR
```

The PMCR can be accessed through the external debug interface, offset `0xE04`.

11.6.2 Performance Monitors Common Event Identification Register 0

The PMCEID0 register characteristics are:

Purpose Defines which common architectural and common microarchitectural feature events are implemented.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
Config	Config	RO	RO	RO	RO	RO

This register is accessible at EL0 when PMUSERENR_EL0.EN is set to `1`.

Configurations The PMCEID0 register is architecturally mapped to:

- The AArch64 register PMCEID0_EL0. See *11.4.2 Performance Monitors Common Event Identification Register 0* on page 11-450.
- The external register PMCEID0_EL0.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes PMCEID0 is a 32-bit register.

The following figure shows the PMCEID0 bit assignments.

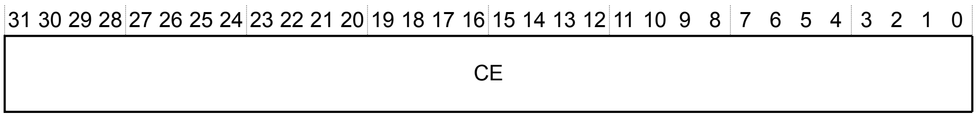

Figure 11-6 PMCEID0 bit assignments

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 11-9 PMCEID0 bit assignments

Bits	Name	Function
[31:0]	CE[31:0]	Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

Table 11-10 PMU common events

Bit	Event number	Event mnemonic	Description
[31]	`0x1F`	L1D_CACHE_ALLOCATE	L1 data cache allocate: `0` This event is not implemented.
[30]	`0x1E`	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]	`0x1D`	BUS_CYCLES	Bus cycle: `1` This event is implemented.
[28]	`0x1C`	TTBR_WRITE_RETIRED	TTBR write, architecturally executed, condition check pass - write to translation table base: `1` This event is implemented.
[27]	`0x1B`	INST_SPEC	Instruction speculatively executed: `1` This event is implemented.
[26]	`0x1A`	MEMORY_ERROR	Local memory error: `0` This event is not implemented.
[25]	`0x19`	BUS_ACCESS	Bus access: `1` This event is implemented.
[24]	`0x18`	L2D_CACHE_WB	L2 data cache Write-Back: `1` This event is implemented.
[23]	`0x17`	L2D_CACHE_REFILL	L2 data cache refill: `1` This event is implemented.

Table 11-10 PMU common events (continued)

Bit	Event number	Event mnemonic	Description
[22]	`0x16`	L2D_CACHE	L2 data cache access: `1` This event is implemented.
[21]	`0x15`	L1D_CACHE_WB	L1 data cache Write-Back: `1` This event is implemented.
[20]	`0x14`	L1I_CACHE	L1 instruction cache access: `1` This event is implemented.
[19]	`0x13`	MEM_ACCESS	Data memory access: `1` This event is implemented.
[18]	`0x12`	BR_PRED	Predictable branch speculatively executed: `1` This event is implemented.
[17]	`0x11`	CPU_CYCLES	Cycle: `1` This event is implemented.
[16]	`0x10`	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed: `1` This event is implemented.
[15]	`0x0F`	UNALIGNED_LDST_RETIRED	Instruction architecturally executed, condition check pass - unaligned load or store: `0` This event is not implemented.
[14]	`0x0E`	BR_RETURN_RETIRED	Instruction architecturally executed, condition check pass - procedure return: `1` This event is implemented.
[13]	`0x0D`	BR_IMMED_RETIRED	Instruction architecturally executed - immediate branch: `1` This event is implemented.
[12]	`0x0C`	PC_WRITE_RETIRED	Instruction architecturally executed, condition check pass - software change of the PC: `1` This event is implemented.
[11]	`0x0B`	CID_WRITE_RETIRED	Instruction architecturally executed, condition check pass - write to CONTEXTIDR: `1` This event is implemented.
[10]	`0x0A`	EXC_RETURN	Instruction architecturally executed, condition check pass - exception return: `1` This event is implemented.

Table 11-10 PMU common events (continued)

Bit	Event number	Event mnemonic	Description
[9]	`0x09`	EXC_TAKEN	Exception taken: `1` This event is implemented.
[8]	`0x08`	INST_RETIRED	Instruction architecturally executed: `1` This event is implemented.
[7]	`0x07`	ST_RETIRED	Instruction architecturally executed, condition check pass - store: `0` This event is not implemented.
[6]	`0x06`	LD_RETIRED	Instruction architecturally executed, condition check pass - load: `0` This event is not implemented.
[5]	`0x05`	L1D_TLB_REFILL	L1 Data TLB refill: `1` This event is implemented.
[4]	`0x04`	L1D_CACHE	L1 data cache access: `1` This event is implemented.
[3]	`0x03`	L1D_CACHE_REFILL	L1 data cache refill: `1` This event is implemented.
[2]	`0x02`	L1I_TLB_REFILL	L1 instruction TLB refill: `1` This event is implemented.
[1]	`0x01`	L1I_CACHE_REFILL	L1 instruction cache refill: `1` This event is implemented.
[0]	`0x00`	SW_INCR	Instruction architecturally executed, condition check pass - software increment: `1` This event is implemented.

To access the PMCEID0:

```
MRC p15,0,<Rt>,c9,c12,6 ; Read PMCEID0 into Rt
```

The PMCEID0 register can be accessed through and the external debug interface, offset `0xE20`.

11.6.3 Performance Monitors Common Event Identification Register 1

The PMCEID1 register characteristics are:

Purpose Defines which common architectural and common microarchitectural feature events are implemented.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL1 (NS)	EL2
Config	RO	RO

This register is accessible at EL0 when PMUSERENR_EL0.EN is set to `1`.

Configurations PMCEID1 is architecturally mapped to:

- The AArch32 register PMCEID1_EL0. See *11.4.3 Performance Monitors Common Event Identification Register 1* on page 11-453.
- The external register PMCEID1_EL0.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes PMCEID1 is a 32-bit register.

The following figure shows the PMCEID1 bit assignments

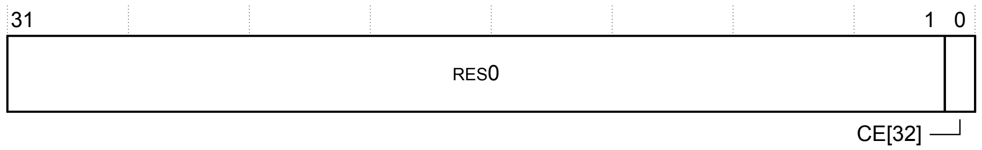

Figure 11-7 PMCEID1 bit assignments

The following table shows the PMCEID1 bit assignments.

Table 11-11 PMCEID1 bit assignments

Bits	Name	Function
[31:1]	RES0	
[0]	CE[32]	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 11-12 PMU common events

Bit	Event number	Event mnemonic	Description
[0]	`0x20`	L2D_CACHE_ALLOCATE	`0` This event is not implemented.

To access the PMCEID1 register:

```
MRC p15,0,<Rt>,c9,c12,7 ; Read PMCEID1 into Rt
```

The PMCEID1 register can be accessed through the external debug interface, offset `0xE24`.

11.7 Memory-mapped PMU register summary

The following table shows the PMU registers that are accessible through 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 11-13 Memory-mapped PMU register summary

Offset	Name	Type	Description
`0x000`	PMEVCNTR0	RW	Performance Monitors Event Count Register 0
`0x004`	-	-	Reserved
`0x008`	PMEVCNTR1	RW	Performance Monitors Event Count Register 1
`0x00C`	-	-	Reserved
`0x010`	PMEVCNTR2	RW	Performance Monitors Event Count Register 2
`0x014`	-	-	Reserved
`0x018`	PMEVCNTR3	RW	Performance Monitors Event Count Register 3
`0x01C`	-	-	Reserved
`0x020`	PMEVCNTR4	RW	Performance Monitors Event Count Register 4
`0x024`	-	-	Reserved
`0x028`	PMEVCNTR5	RW	Performance Monitors Event Count Register 5
`0x02C-0xF4`	-	-	Reserved
`0x0F8`	PMCCNTR[31:0]	RW	Performance Monitors Cycle Count Register
`0x0FC`	PMCCNTR[63:32]	RW	
`0x100-0x3FC`	-	-	Reserved
`0x400`	PMEVTYPER0	RW	Performance Monitors Event Type Register
`0x404`	PMEVTYPER1	RW	
`0x408`	PMEVTYPER2	RW	
`0x40C`	PMEVTYPER3	RW	
`0x410`	PMEVTYPER4	RW	
`0x414`	PMEVTYPER5	RW	
`0x418-0x478`	-	-	Reserved
`0x47C`	PMCCFILTR	RW	Performance Monitors Cycle Count Filter Register
`0x480-0xBFC`	-	-	Reserved
`0xC00`	PMCNTENSET	RW	Performance Monitors Count Enable Set Register
`0xC04-0xC1C`	-	-	Reserved
`0xC20`	PMCNTENCLR	RW	Performance Monitors Count Enable Clear Register
`0xC24-0xC7C`	-	-	Reserved

Table 11-13 Memory-mapped PMU register summary (continued)

Offset	Name	Type	Description
`0xC80`	PMOVSCLR	RW	Performance Monitors Overflow Flag Status Register
`0xC84-0xC9C`	-	-	Reserved
`0xCA0`	PMSWINC	WO	Performance Monitors Software Increment Register
`0xCA4-0xCBC`	-	-	Reserved
`0xCC0`	PMOVSSET	RW	Performance Monitors Overflow Flag Status Set Register
`0xCC4-0xDFC`	-	-	Reserved
`0xE00`	PMCFGR	RO	*11.8.1 Performance Monitors Configuration Register* on page 11-467
`0xE04`	PMCR[dh]	RW	Performance Monitors Control Register
`0xE08-0xE1C`	-	-	Reserved
`0xE20`	PMCEID0	RO	Performance Monitors Common Event Identification Register 0
`0xE24`	PMCEID1	RO	Performance Monitors Common Event Identification Register 1
`0xE28-0xFA4`	-	-	Reserved
`0xFA8`	PMDEVAFF0	RO	Performance Monitors Device Affinity Register 0, see *4.3.2 Multiprocessor Affinity Register* on page 4-84
`0xFAC`	PMDEVAFF1	RO	Performance Monitors Device Affinity Register 1, RES0
`0xFB0`	PMLAR	WO	Performance Monitors Lock Access Register
`0xFB4`	PMLSR	RO	Performance Monitors Lock Status Register
`0xFB8`	PMAUTHSTATUS	RO	Performance Monitors Authentication Status Register
`0xFBC`	PMDEVARCH	RO	Performance Monitors Device Architecture Register
`0xFC0-0xFC8`	-	-	Reserved
`0xFCC`	PMDEVTYPE	RO	Performance Monitors Device Type Register
`0xFD0`	PMPIDR4	RO	*Performance Monitors Peripheral Identification Register 4* on page 11-471
`0xFD4`	PMPIDR5	RO	*Performance Monitors Peripheral Identification Register 5-7* on page 11-471
`0xFD8`	PMPIDR6	RO	
`0xFDC`	PMPIDR7	RO	
`0xFE0`	PMPIDR0	RO	*Performance Monitors Peripheral Identification Register 0* on page 11-468
`0xFE4`	PMPIDR1	RO	*Performance Monitors Peripheral Identification Register 1* on page 11-469
`0xFE8`	PMPIDR2	RO	*Performance Monitors Peripheral Identification Register 2* on page 11-469
`0xFEC`	PMPIDR3	RO	*Performance Monitors Peripheral Identification Register 3* on page 11-470
`0xFF0`	PMCIDR0	RO	*Performance Monitors Component Identification Register 0* on page 11-472
`0xFF4`	PMCIDR1	RO	*Performance Monitors Component Identification Register 1* on page 11-472

[dh] This register is distinct from the PMCR system register. It does not have the same value.

Table 11-13 Memory-mapped PMU register summary (continued)

Offset	Name	Type	Description
`0xFF8`	PMCIDR2	RO	*Performance Monitors Component Identification Register 2* on page 11-473
`0xFFC`	PMCIDR3	RO	*Performance Monitors Component Identification Register 3* on page 11-474

11.8 Memory-mapped PMU register descriptions

This section describes the Cortex-A73 processor PMU registers accessible through the memory-mapped and debug interfaces.

Table 11-13 Memory-mapped PMU register summary on page 11-464 provides cross-references to individual registers.

This section contains the following subsections:
- *11.8.1 Performance Monitors Configuration Register* on page 11-467.
- *11.8.2 Performance Monitors Peripheral Identification Registers* on page 11-468.
- *11.8.3 Performance Monitors Component Identification Registers* on page 11-472.

11.8.1 Performance Monitors Configuration Register

The PMCFGR characteristics are:

Purpose Contains PMU specific configuration data.

Usage constraints The accessibility to the PMCFGR by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
Error	Error	Error	Error	RO	RO

Configurations The PMCFGR is in the processor power domain.

Attributes PMCFGR is a 32-bit register.

The following figure shows the PMCFGR bit assignments.

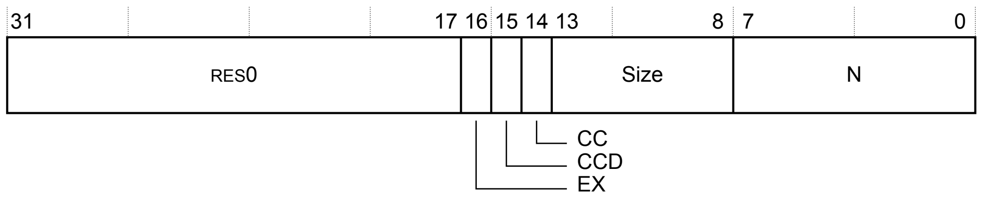

Figure 11-8 PMCFGR bit assignments

The following table shows the PMCFGR bit assignments.

Table 11-14 PMCFGR bit assignments

Bits	Name	Function
[31:17]	-	Reserved, RES0.
[16]	EX	Export supported. The value is: 1 Export is supported. PMCR.EX is read/write.
[15]	CCD	Cycle counter has prescale. The value is: 1 PMCR.D is read/write.
[14]	CC	Dedicated cycle counter supported. This bit is RES1.

Table 11-14 PMCFGR bit assignments (continued)

Bits	Name	Function
[13:8]	Size	Counter size. The value is: `0b111111` 64-bit counters.
[7:0]	N	Number of event counters. The value is: `0x06` Six counters.

The PMCFGR can be accessed through the external debug interface, offset `0xE00`.

11.8.2 Performance Monitors Peripheral Identification Registers

The Performance Monitors Peripheral Identification Registers provide standard information required for all components that conform to the Arm PMUv3 architecture.

There is a set of eight registers, listed in the following table.

Table 11-15 Summary of the Performance Monitors Peripheral Identification Registers

Register	Value	Offset
PMPIDR4	`0x04`	`0xFD0`
PMPIDR5	`0x00`	`0xFD4`
PMPIDR6	`0x00`	`0xFD8`
PMPIDR7	`0x00`	`0xFDC`
PMPIDR0	`0xD9`	`0xFE0`
PMPIDR1	`0xB9`	`0xFE4`
PMPIDR2	`0x0B`	`0xFE8`
PMPIDR3	`0x00`	`0xFEC`

Only bits[7:0] of each Performance Monitors Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Performance Monitors Peripheral ID Registers define a single 64-bit Peripheral ID.

Performance Monitors Peripheral Identification Register 0

The PMPIDR0 characteristics are:

Purpose Provides information to identify a Performance Monitor component.

Usage constraints The PMPIDR0 can be accessed through the external debug interface.

The accessibility to the PMPIDR0 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Configurations The PMPIDR0 is in the Debug power domain.

Attributes PMPIDR0 is a 32-bit register.

The following figure shows the PMPIDR0 bit assignments.

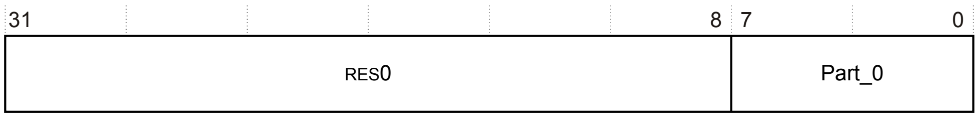

Figure 11-9 PMPIDR0 bit assignments

The following table shows the PMPIDR0 bit assignments.

Table 11-16 PMPIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	Part_0	`0xD9` Least significant byte of the performance monitor part number.

The PMPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

Performance Monitors Peripheral Identification Register 1

The PMPIDR1 characteristics are:

Purpose Provides information to identify a Performance Monitor component.

Usage constraints The PMPIDR1 can be accessed through the external debug interface.

The accessibility to the PMPIDR1 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Configurations The PMPIDR1 is in the Debug power domain.

Attributes PMPIDR1 is a 32-bit register.

The following figure shows the PMPIDR1 bit assignments.

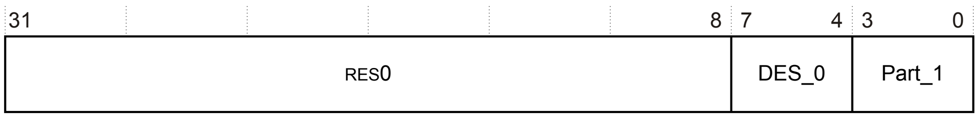

Figure 11-10 PMPIDR1 bit assignments

The following table shows the PMPIDR1 bit assignments.

Table 11-17 PMPIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	DES_0	`0xB` Arm Limited. This is the least significant nibble of JEP106 ID code.
[3:0]	Part_1	`0x9` Most significant nibble of the performance monitor part number.

The PMPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

Performance Monitors Peripheral Identification Register 2

The PMPIDR2 characteristics are:

Purpose	Provides information to identify a Performance Monitor component.
Usage constraints	The accessibility to the PMPIDR2 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

The PMPIDR2 can be accessed through the external debug interface.

Configurations	The PMPIDR2 is in the Debug power domain.
Attributes	PMPIDR2 is a 32-bit register.

The following figure shows the PMPIDR2 bit assignments.

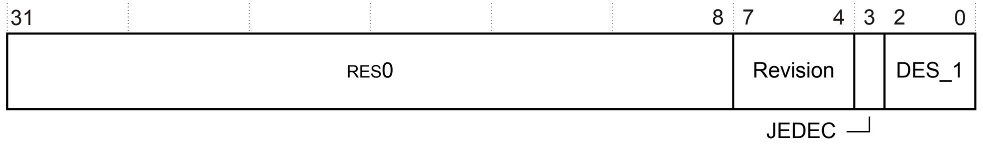

Figure 11-11 PMPIDR2 bit assignments

The following table shows the PMPIDR2 bit assignments.

Table 11-18 PMPIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Revision	`0x0` r0p0.
[3]	JEDEC	`0b1` Indicates a JEP106 identity code is used.
[2:0]	DES_1	`0b011` Arm Limited. The most significant bits of JEP106 ID code.

The PMPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

Performance Monitors Peripheral Identification Register 3

The PMPIDR3 characteristics are:

Purpose	Provides information to identify a Performance Monitor component.
Usage constraints	The PMPIDR3 can be accessed through the external debug interface.

The accessibility to the PMPIDR3 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Configurations	The PMPIDR3 is in the Debug power domain.
Attributes	PMPIDR3 is a 32-bit register.

The following figure shows the PMPIDR3 bit assignments.

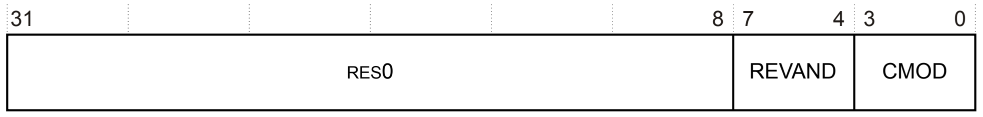

Figure 11-12 PMPIDR3 bit assignments

The following table shows the PMPIDR3 bit assignments.

Table 11-19 PMPIDR3 bit assignments

Bits	Name	Function	
[31:8]	-	Reserved, RES0.	
[7:4]	REVAND	`0x0`	Part minor revision.
[3:0]	CMOD	`0x0`	Customer modified.

The PMPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

Performance Monitors Peripheral Identification Register 4

The PMPIDR4 characteristics are:

Purpose Provides information to identify a Performance Monitor component.

Usage constraints The PMPIDR4 can be accessed through the external debug interface.

The accessibility to the PMPIDR4 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Configurations The PMPIDR4 is in the Debug power domain.

Attributes PMPIDR4 is a 32-bit register.

The following figure shows the PMPIDR4 bit assignments.

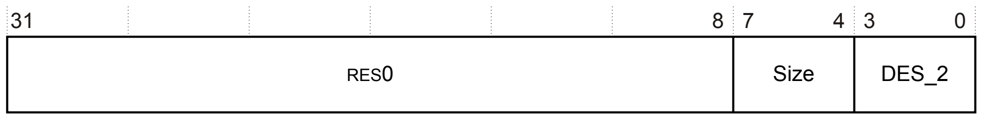

Figure 11-13 PMPIDR4 bit assignments

The following table shows the PMPIDR4 bit assignments.

Table 11-20 PMPIDR4 bit assignments

Bits	Name	Function	
[31:8]	-	Reserved, RES0.	
[7:4]	Size	`0x0`	Size of the component. RES0. Log_2 the number of 4KB pages from the start of the component to the end of the component ID registers.
[3:0]	DES_2	`0x4`	Arm Limited. This is the least significant nibble of the JEP106 continuation code.

The PMPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

Performance Monitors Peripheral Identification Register 5-7

No information is held in PMPIDR5, PMPIDR6, and PMPIDR7. They are reserved for future use and are RES0.

11.8.3 Performance Monitors Component Identification Registers

There are four read-only Performance Monitors Component Identification Registers. The following table shows these registers.

Table 11-21 Summary of the Performance Monitors Component Identification Registers

Register	Value	Offset
PMCIDR0	`0x0D`	`0xFF0`
PMCIDR1	`0x90`	`0xFF4`
PMCIDR2	`0x05`	`0xFF8`
PMCIDR3	`0xB1`	`0xFFC`

The Performance Monitors Component Identification Registers identify Performance Monitor as Arm PMUv3 architecture.

Performance Monitors Component Identification Register 0

The PMCIDR0 characteristics are:

Purpose Provides information to identify a Performance Monitor component.

Usage constraints The PMCIDR0 can be accessed through the external debug interface.

The accessibility to the PMCIDR0 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Configurations The PMCIDR0 is in the Debug power domain.

Attributes PMCIDR0 is a 32-bit register.

The following figure shows the PMCIDR0 bit assignments.

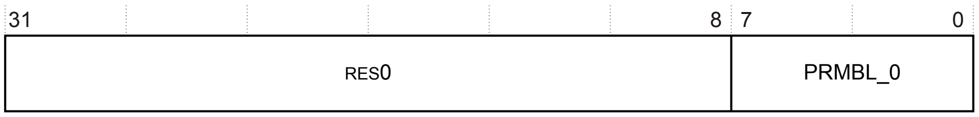

Figure 11-14 PMCIDR0 bit assignments

The following table shows the PMCIDR0 bit assignments.

Table 11-22 PMCIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_0	`0x0D` Preamble byte 0.

The PMCIDR0 can be accessed through the external debug interface, offset `0xFF0`.

Performance Monitors Component Identification Register 1

The PMCIDR1 characteristics are:

Purpose Provides information to identify a Performance Monitor component.

Usage constraints The PMCIDR1 can be accessed through the external debug interface.

The accessibility to the PMCIDR1 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Configurations The PMCIDR1 is in the Debug power domain.

Attributes PMCIDR1 is a 32-bit register.

The following figure shows the PMCIDR1 bit assignments.

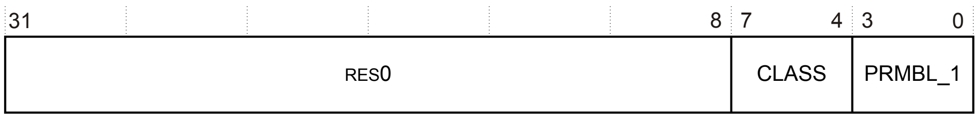

Figure 11-15 PMCIDR1 bit assignments

The following table shows the PMCIDR1 bit assignments.

Table 11-23 PMCIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	CLASS	`0x9` Debug component.
[3:0]	PRMBL_1	`0x0` Preamble byte 1, RES0.

The PMCIDR1 can be accessed through the external debug interface, offset `0xFF4`.

Performance Monitors Component Identification Register 2

The PMCIDR2 characteristics are:

Purpose Provides information to identify a Performance Monitor component.

Usage constraints The PMCIDR2 can be accessed through the external debug interface.

The accessibility to the PMCIDR2 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Configurations The PMCIDR2 is in the Debug power domain.

Attributes PMCIDR2 is a 32-bit register.

The following figure shows the PMCIDR2 bit assignments.

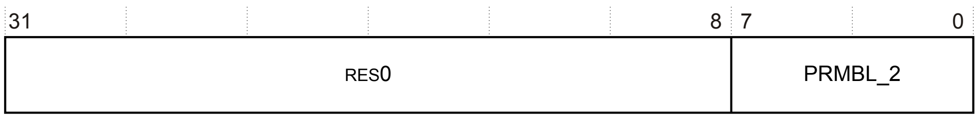

Figure 11-16 PMCIDR2 bit assignments

The following table shows the PMCIDR2 bit assignments.

Table 11-24 PMCIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_2	`0x05` Preamble byte 2.

The PMCIDR2 can be accessed through the external debug interface, offset `0xFF8`.

Performance Monitors Component Identification Register 3

The PMCIDR3 characteristics are:

Purpose Provides information to identify a Performance Monitor component.

Usage constraints The PMCIDR3 can be accessed through the external debug interface.

The accessibility to the PMCIDR3 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Configurations The PMCIDR3 is in the Debug power domain.

Attributes PMCIDR3 is a 32-bit register.

The following figure shows the PMCIDR3 bit assignments.

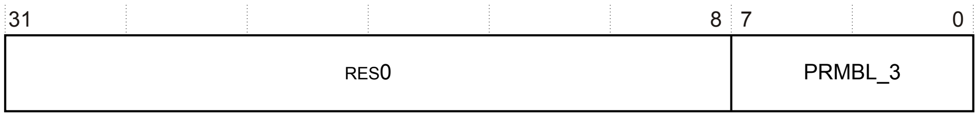

Figure 11-17 PMCIDR3 bit assignments

The following table shows the PMCIDR3 bit assignments.

Table 11-25 PMCIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_3	`0xB1` Preamble byte 3.

The PMCIDR3 can be accessed through the external debug interface, offset `0xFFC`.

11.9 Events

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

Table 11-26 PMU events

Event number	Event mnemonic	PMU event bus (to external)	PMU event bus (to trace)	Event name
`0x00`	SW_INCR	-	-	Software increment. Instruction architecturally executed (condition check pass).
`0x01`	L1I_CACHE_REFILL	[0]	[0]	L1 instruction cache refill.
`0x02`	L1I_TLB_REFILL	[1]	[1]	L1 instruction TLB refill.
`0x03`	L1D_CACHE_REFILL	[2]	[2]	L1 data cache refill.
`0x04`	L1D_CACHE	[4:3]	[4:3]	L1 data cache access.
`0x05`	L1D_TLB_REFILL	[5]	[5]	L1 data TLB refill.
`0x08`	INST_RETIRED	[16:10]	[16:10][di]	Instruction architecturally executed.
`0x09`	EXC_TAKEN	[17]	[17]	Exception taken.
`0x0A`	EXC_RETURN	[18]	[18]	Instruction architecturally executed, condition code check pass, exception return.
`0x0B`	CID_WRITE_RETIRED	[19]	[19]	Instruction architecturally executed, condition code check pass, write to CONTEXTIDR.
`0x0C`	PC_WRITE_RETIRED	[21]	[21]	Instruction architecturally executed, condition check pass, software change of the PC.
`0x0D`	BR_IMMED_RETIRED	[22]	[22]	Instruction architecturally executed, immediate branch.
`0x0E`	BR_RETURN_RETIRED	[23]	[23]	Instruction architecturally executed, condition code check pass, procedure return.[dj]
`0x10`	BR_MIS_PRED	[26]	-	Mispredicted or not predicted branch speculatively executed.
`0x11`	CPU_CYCLES	-	-	Cycle.
`0x12`	BR_PRED	[27]	-	Predictable branch speculatively executed.
`0x13`	MEM_ACCESS	[30:28]	-	Data memory access.
`0x14`	L1I_CACHE	[31]	-	L1 instruction cache access.
`0x15`	L1D_CACHE_WB	[32]	-	L1 data cache Write-Back.
`0x16`	L2D_CACHE	-	-	L2 data cache access.
`0x17`	L2D_CACHE_REFILL	-	-	L2 data cache refill.

[di] This event is encoded with 7 bits so that it can indicate that up to 127 instructions have been retired.
[dj] In addition to the instructions that are counted in this event as specified in the Armv8 architecture, the Cortex-A73 processor also counts the instruction `LDR PC, [SP, #offset]`.

Table 11-26 PMU events (continued)

Event number	Event mnemonic	PMU event bus (to external)	PMU event bus (to trace)	Event name
`0x18`	L2D_CACHE_WB	-	-	L2 data cache Write-Back.
`0x19`	BUS_ACCESS	-	-	Bus access.
`0x1B`	INT_SPEC	-	-	Operation speculatively executed.
`0x1C`	TTBR_WRITE_RETIRED	[20]	[20]	Instruction architecturally executed (condition check pass) - Write to TTBR.
`0x1D`	BUS_CYCLES	-	-	Bus cycle.
`0x1E`	CHAIN	-	-	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.
`0x40`	L1D_CACHE_RD	-	-	L1 data cache access, read.
`0x41`	L1D_CACHE_WR	-	-	L1 data cache access, write.
`0x50`	L2D_CACHE_RD	-	-	L2 data cache access, read.
`0x51`	L2D_CACHE_WR	-	-	L2 data cache access, write.
`0x56`	L2D_CACHE_WB_VICTIM	-	-	L2 data cache write-back, victim.
`0x57`	L2D_CACHE_WB_CLEAN	-	-	L2 data cache write-back, cleaning and coherency.
`0x58`	L2D_CACHE_INVAL	-	-	L2 data cache invalidate.
`0x62`	BUS_ACCESS_SHARED	-	-	Bus access, Normal, Cacheable, Shareable.
`0x63`	BUS_ACCESS_NOT_SHARED	-	-	Bus access, not Normal, Cacheable, or Shareable.
`0x64`	BUS_ACCESS_NORMAL	-	-	Bus access, Normal.
`0x65`	BUS_ACCESS_SO_DIV	-	-	Bus access, Device.
`0x66`	MEM_ACCESS_RD	-	-	Data memory access, read.
`0x67`	MEM_ACCESS_WR	-	-	Data memory access, write.
`0x6A`	UNALIGNED_LDST_SPEC	[25:24]	[25:24]	Unaligned access.
`0x6C`	LDREX_SPEC	-	-	Exclusive operation speculatively executed, LDREX, or LDX.
`0x6E`	STREX_FAIL_SPEC	-	-	Exclusive operation speculatively executed, STREX, or STX pass.
`0x70`	LD_SPEC	[7:6]	[7:6]	Operation speculatively executed, load.
`0x71`	ST_SPEC	[9:8]	[9:8]	Operation speculatively executed, store.
`0x72`	LDST_SPEC	-	-	Operation speculatively executed, load or store.
`0x73`	DP_SPEC	-	-	Operation speculatively executed, integer data processing.

Table 11-26 PMU events (continued)

Event number	Event mnemonic	PMU event bus (to external)	PMU event bus (to trace)	Event name
`0x74`	ASE_SPEC	-	-	Operation speculatively executed, Advanced SIMD instruction.
`0x75`	VFP_SPEC	-	-	Operation speculatively executed, floating-point instruction.
`0x77`	CRYPTO_SPEC	-	-	Operation speculatively executed, Cryptographic instruction.
`0x7A`	BR_INDIRECT_SPEC	-	-	Branch speculatively executed - Indirect branch.
`0x7C`	ISB_SPEC	-	-	Barrier speculatively executed, ISB.
`0x7D`	DSB_SPEC	-	-	Barrier speculatively executed, DSB.
`0x7E`	DMB_SPEC	-	-	Barrier speculatively executed, DMB.
`0x8A`	EXC_HVC	-	-	Exception taken, Hypervisor Call.
`0xC0`	LF_STALL	-	-	A linefill caused an instruction side stall.
`0xC1`	PTW_STALL	-	-	A translation table walk caused an instruction side stall.
`0xC2`	I_TAG_RAM_RD	-	-	Number of ways read in the instruction cache - Tag RAM.
`0xC3`	I_DATA_RAM_RD	-	-	Number of ways read in the instruction cache - Data RAM.
`0xC4`	I_BTAC_RAM_RD	-	-	Number of ways read in the instruction BTAC RAM.
`0xD3`	D_LSU_SLOT_FULL	-	-	Duration for which all slots in the Load-Store Unit are busy.
`0xD8`	LS_IQ_FULL	-	-	Duration for which all slots in the load-store issue queue are busy.
`0xD9`	DP_IQ_FULL	-	-	Duration for which all slots in the data processing issue queue are busy.
`0xDA`	DE_IQ_FULL	-	-	Duration for which all slots in the Data Engine issue queue are busy.
`0xDC`	EXC_TRAP_HYP	-	-	Number of Traps to hypervisor.
`0xDE`	ETM_EXT_OUT0	-	-	ETM trace unit output 0
`0xDF`	ETM_EXT_OUT1	-	-	ETM trace unit output 1
`0xE0`	MMU_PTW	-	-	Duration of a translation table walk handled by the MMU.
`0xE1`	MMU_PTW_ST1	-	-	Duration of a Stage 1 translation table walk handled by the MMU.
`0xE2`	MMU_PTW_ST2	-	-	Duration of a Stage 2 translation table walk handled by the MMU.

Table 11-26 PMU events (continued)

Event number	Event mnemonic	PMU event bus (to external)	PMU event bus (to trace)	Event name
`0xE3`	MMU_PTW_LSU	-	-	Duration of a translation table walk requested by the LSU.
`0xE4`	MMU_PTW_ISIDE	-	-	Duration of a translation table walk requested by the Instruction Side.
`0xE5`	MMU_PTW_PLD	-	-	Duration of a translation table walk requested by a Preload instruction or Prefetch request.
`0xE6`	MMU_PTW_CP15	-	-	Duration of a translation table walk requested by a CP15 operation (maintenance by MVA and VA to PA operations).
`0xE7`	PLD_UTLB_REFILL	-	-	Level 1 PLD TLB refill.
`0xE8`	CP15_UTLB_REFILL	-	-	Level 1 CP15 TLB refill.
`0xE9`	UTLB_FLUSH	-	-	Level 1 TLB flush.
`0xEA`	TLB_ACCESS	-	-	Level 2 TLB access.
`0xEB`	TLB_MISS	-	-	Level 2 TLB miss.
`0xEC`	DCACHE_SELF_HIT_VIPT	-	-	Data cache hit in itself due to VIPT aliasing

11.10 Interrupts

The Cortex-A73 processor asserts the **nPMUIRQ** signal when an interrupt is generated by the PMU. 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 CTI. See *Chapter 13 Cross Trigger* on page 13-545 for more information.

11.11 Exporting PMU events

This section describes exporting of PMU events.

This section contains the following subsections:

- *11.11.1 External hardware* on page 11-480.
- *11.11.2 Debug trace hardware* on page 11-480.

11.11.1 External hardware

In addition to the counters in the core, some events are exported on the **PMUEVENT** bus and can be connected to external hardware.

These events are described in *Table 11-26 PMU events* on page 11-475.

11.11.2 Debug trace hardware

Some events are exported to the ETM trace unit or to the *Cross Trigger Interface* (CTI), to enable the events to be monitored.

These events are described in *Table 11-26 PMU events* on page 11-475. See *Chapter 12 Embedded Trace Macrocell* on page 12-481 and *Chapter 13 Cross Trigger* on page 13-545 for more information.

Chapter 12
Embedded Trace Macrocell

This chapter describes the *Embedded Trace Macrocell* (ETM) for the Cortex-A73 processor.

It contains the following sections:

12.1 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, DS-5 Development Studio.

See the *Arm® Embedded Trace Macrocell Architecture Specification, ETMv4* for more information.

12.2 ETM trace unit generation options and resources

The following table shows the trace generation options implemented in the Cortex-A73 ETM trace unit.

Table 12-1 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	1
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
Exception Levels implemented in Non-secure state	EL2, EL1, EL0
Exception Levels implemented in Secure state	EL3, EL1, EL0
Number of events supported in the trace	4
Return stack support	Implemented
Tracing of SError exception support	Implemented
Instruction trace cycle counting minimum threshold	4
Size of Trace ID	7 bits
Synchronization period support	Read-write
Global timestamp size	64 bits
Number of cores available for tracing	1
ATB trigger support	Implemented
Low power behavior override	Implemented
Stall control support	Implemented
Support for no overflows in the trace	Not implemented

The following table shows the resources implemented in the Cortex-A73 ETM trace unit.

Table 12-2 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	30, 4 CTI + 26 PMU
Number of counters implemented	2

Table 12-2 ETM trace unit resources implemented (continued)

Description	Configuration
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 processor comparator inputs implemented	0
Data address comparisons implemented	Not implemented
Number of data value comparators implemented	0

12.3 ETM trace unit functional description

The following figure shows the main functional blocks of the ETM trace unit.

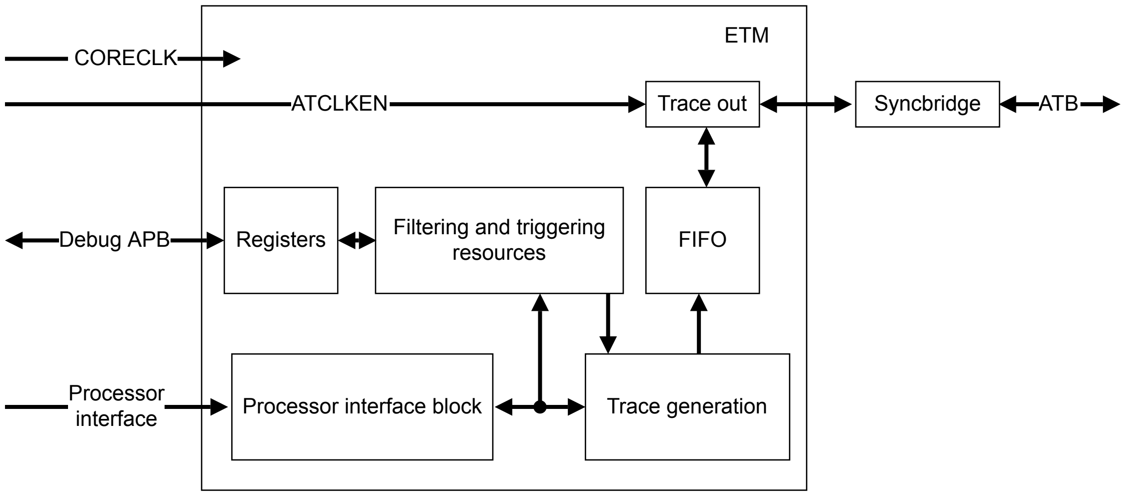

Figure 12-1 ETM functional blocks

Processor interface block

This block monitors the behavior of the processor and generates P0 elements that are essentially executed instructions 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 the FIFO is output on the synchronous AMBA ATB interface.

Syncbridge

The ATB interface from the trace out block goes through an ATB synchronous bridge. See the *Arm® AMBA® 4 ATB Protocol Specification ATBv1.0 and ATBv1.1* for information about the ATB protocol.

12.4 Reset

The reset for the ETM trace unit is the same as a Cold reset for the processor. The ETM trace unit is not reset when Warm reset is applied to the processor so that tracing through warm processor reset is possible.

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

12.5 Modes of operation and execution

This section describes how to control the ETM trace unit programming.

This section contains the following subsections:
- *12.5.1 Controlling ETM trace unit programming* on page 12-487.
- *12.5.2 Programming and reading ETM trace unit registers* on page 12-488.

12.5.1 Controlling ETM trace unit programming

When programming the ETM trace unit registers, you must enable all the changes at the same time. For example, if the counter is reprogrammed, it might start to count based on incorrect events, before the trigger condition has been correctly set up.

You must use the ETM trace unit main enable in the TRCPRGCTLR to disable all trace operations during programming. See the *Arm® Embedded Trace Macrocell Architecture Specification, ETMv4* for more information. The following figure shows the procedure that must be used when programming the ETM trace unit registers.

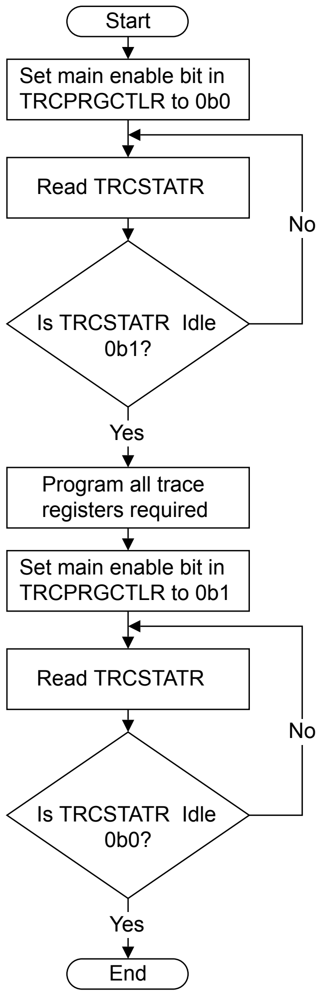
Start
Set main enable bit in TRCPRGCTLR to 0b0
Read TRCSTATR
Is TRCSTATR Idle 0b1?
No
Yes
Program all trace registers required
Set main enable bit in TRCPRGCTLR to 0b1
Read TRCSTATR
Is TRCSTATR Idle 0b0?
No
Yes
End

Figure 12-2 Programming ETM trace unit registers

Note

The Cortex-A73 processor does not have to be in the Debug state while you program the ETM trace unit registers.

12.5.2 Programming and reading ETM trace unit registers

You program and read the ETM trace unit registers using the Debug APB interface. This provides a direct method of programming the ETM trace unit.

12.6 ETM trace unit register interfaces

The Cortex-A73 processor only supports the memory-mapped interface for trace registers.

For more information see *10.10 External debug interface* on page 10-429.

This section contains the following subsection:
- *12.6.1 Access permissions* on page 12-489.

12.6.1 Access permissions

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.

12.7 ETM register summary

This section summarizes the ETM trace unit registers.

For full descriptions of the ETM trace unit registers, see:

- *12.8 ETM register descriptions* on page 12-494, for the IMPLEMENTATION DEFINED registers and the *Arm® Embedded Trace Macrocell Architecture Specification, ETMv4*, for the other registers.

Note

Access type is described as follows:

RW	Read and write.
RO	Read only.
WO	Write only.

The following table lists all of the ETM trace unit registers and their offsets 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.

Table 12-3 ETM trace unit register summary

Offset	Name	Type	Description
`0x000`	-	-	Reserved
`0x004`	TRCPRGCTLR	RW	Programming Control Register
`0x008`	-	-	Reserved
`0x00C`	TRCSTATR	RO	Status Register
`0x010`	TRCCONFIGR	RW	*12.8.1 Trace Configuration Register* on page 12-495
`0x014`	-	-	Reserved
`0x018`	TRCAUXCTLR	RW	*12.8.2 Auxiliary Control Register* on page 12-496
`0x01C`	-	-	Reserved
`0x020`	TRCEVENTCTL0R	RW	*12.8.3 Event Control 0 Register* on page 12-498
`0x024`	TRCEVENTCTL1R	RW	*12.8.4 Event Control 1 Register* on page 12-499
`0x028`	-	-	Reserved
`0x02C`	TRCSTALLCTLR	RW	*12.8.5 Stall Control Register* on page 12-500
`0x030`	TRCTSCTLR	RW	*12.8.6 Global Timestamp Control Register* on page 12-501
`0x034`	TRCSYNCPR	RW	*12.8.7 Synchronization Period Register* on page 12-502
`0x038`	TRCCCCTLR	RW	*12.8.8 Cycle Count Control Register* on page 12-502
`0x03C`	TRCBBCTLR	RW	*12.8.9 Branch Broadcast Control Register* on page 12-503
`0x040`	TRCTRACEIDR	RW	*12.8.10 Trace ID Register* on page 12-504
`0x044-0x07C`	-	-	Reserved
`0x080`	TRCVICTLR	RW	*12.8.11 ViewInst Main Control Register* on page 12-504
`0x084`	TRCVIIECTLR	RW	*12.8.12 ViewInst Include-Exclude Control Register* on page 12-506
`0x088`	TRCVISSCTLR	RW	*12.8.13 ViewInst Start-Stop Control Register* on page 12-507

Table 12-3 ETM trace unit register summary (continued)

Offset	Name	Type	Description
`0x08C-0x0FC`	-	-	Reserved
`0x100`	TRCSEQEVR0	RW	*12.8.14 Sequencer State Transition Control Registers 0-2* on page 12-508
`0x104`	TRCSEQEVR1	RW	*12.8.14 Sequencer State Transition Control Registers 0-2* on page 12-508
`0x108`	TRCSEQEVR2	RW	*12.8.14 Sequencer State Transition Control Registers 0-2* on page 12-508
`0x10C-0x114`	-	-	Reserved
`0x118`	TRCSEQRSTEVR	RW	*12.8.15 Sequencer Reset Control Register* on page 12-509
`0x11C`	TRCSEQSTR	RW	*12.8.16 Sequencer State Register* on page 12-510
`0x120`	TRCEXTINSELR	RW	*12.8.17 External Input Select Register* on page 12-510
`0x124-0x13C`	-	-	Reserved
`0x140`	TRCCNTRLDVR0	RW	*12.8.18 Counter Reload Value Registers 0-1* on page 12-511
`0x144`	TRCCNTRLDVR1	RW	*12.8.18 Counter Reload Value Registers 0-1* on page 12-511
`0x148-0x14C`	-	-	Reserved
`0x150`	TRCCNTCTLR0	RW	*12.8.19 Counter Control Register 0* on page 12-512
`0x154`	TRCCNTCTLR1	RW	*12.8.20 Counter Control Register 1* on page 12-513
`0x158-0x15C`	-	-	Reserved
`0x160`	TRCCNTVR0	RW	*12.8.21 Counter Value Registers 0-1* on page 12-514
`0x164`	TRCCNTVR1	RW	*12.8.21 Counter Value Registers 0-1* on page 12-514
`0x168-0x17C`	-	-	Reserved
`0x180`	TRCIDR8	RO	*12.8.22 ID Register 8* on page 12-514
`0x184`	TRCIDR9	RO	*12.8.23 ID Register 9* on page 12-515
`0x188`	TRCIDR10	RO	*12.8.24 ID Register 10* on page 12-515
`0x18C`	TRCIDR11	RO	*12.8.25 ID Register 11* on page 12-516
`0x190`	TRCIDR12	RO	*12.8.26 ID Register 12* on page 12-516
`0x194`	TRCIDR13	RO	*12.8.27 ID Register 13* on page 12-517
`0x198-0x1BC`	-	-	Reserved
`0x1C0`	TCRIMSPEC0	RW	*12.8.28 Implementation Specific Register 0* on page 12-517
`0x1C4-0x1DC`	-	-	Reserved
`0x1E0`	TRCIDR0	RO	*12.8.29 ID Register 0* on page 12-518
`0x1E4`	TRCIDR1	RO	*12.8.30 ID Register 1* on page 12-520
`0x1E8`	TRCIDR2	RO	*12.8.31 ID Register 2* on page 12-521
`0x1EC`	TRCIDR3	RO	*12.8.32 ID Register 3* on page 12-522
`0x1F0`	TRCIDR4	RO	*12.8.33 ID Register 4* on page 12-523
`0x1F4`	TRCIDR5	RO	*12.8.34 ID Register 5* on page 12-524
`0x1F8-0x204`	-	-	Reserved

Table 12-3 ETM trace unit register summary (continued)

Offset	Name	Type	Description
`0x208-0x23C`	TRCRSCTLRn	RW	*12.8.35 Resource Selection Control Registers 2-16* on page 12-526, n is 2-15
`0x240-0x27C`	-	-	Reserved
`0x280`	TRCSSCCR0	RW	*12.8.36 Single-Shot Comparator Control Register 0* on page 12-527
`0x284-0x29C`	-	-	Reserved
`0x2A0`	TRCSSCSR0	RW	*12.8.37 Single-Shot Comparator Status Register 0* on page 12-528
`0x2A4-0x2FC`	-	-	Reserved
`0x300`	TRCOSLAR	WO	OS Lock Access Register
`0x304`	TRCOSLSR	RO	OS Lock Status Register
`0x308-0x30C`	-	-	Reserved
`0x310`	TRCPDCR	RW	Power Down Control Register
`0x314`	TRCPDSR	RO	Power Down Status Register
`0x318-0x3FC`	-	-	Reserved
`0x400-0x438`	TRCACVRn	RW	*12.8.38 Address Comparator Value Registers 0-7* on page 12-528[n=0-7]
`0x440-0x47C`	-	-	Reserved
`0x480-0x4B8`	TRCACATRn	RW	*12.8.39 Address Comparator Access Type Registers 0-7* on page 12-529
`0x4C0-0x5FC`	-	-	Reserved
`0x600`	TRCCIDCVR0	RW	*12.8.40 Context ID Comparator Value Register 0* on page 12-531
`0x604-0x63F`	-	-	Reserved
`0x640`	TRCVMIDCVR0	RW	*12.8.41 VMID Comparator Value Register 0* on page 12-531
`0x644-0x67F`	-	-	Reserved
`0x680`	TRCCIDCCTLR0	RW	*12.8.42 Context ID Comparator Control Register 0* on page 12-532
`0x684-0xEE0`	-	-	Reserved
`0xEE4`	TRCITATBIDR	RW	*12.8.43 Integration ATB Identification Register* on page 12-532
`0xEE8`	-	-	Reserved
`0xEEC`	TRCITIDATAR	WO	*12.8.44 Integration Instruction ATB Data Register* on page 12-533
`0xEF0`	-	-	Reserved
`0xEF4`	TRCITIATBINR	RO	*12.8.45 Integration Instruction ATB In Register* on page 12-535.
`0xEF8`	-	-	Reserved
`0xEFC`	TRCITIATBOUTR	WO	*12.8.46 Integration Instruction ATB Out Register* on page 12-536
`0xF00`	TRCITCTRL	RW	*12.8.47 Integration Mode Control Register* on page 12-536
`0xF04-0xF9C`	-	-	Reserved
`0xFA0`	TRCCLAIMSET	RW	*12.8.48 Claim Tag Set Register* on page 12-537
`0xFA4`	TRCCLAIMCLR	RW	*12.8.49 Claim Tag Clear Register* on page 12-538
`0xFA8`	TRCDEVAFF0	RO	*12.8.50 Device Affinity Register 0* on page 12-538

Table 12-3 ETM trace unit register summary (continued)

Offset	Name	Type	Description
`0xFAC`	TRCDEVAFF1	RO	*12.8.51 Device Affinity Register 1* on page 12-539
`0xFB0`	TRCLAR	WO	Software Lock Access Register
`0xFB4`	TRCLSR	RO	Software Lock Status Register
`0xFB8`	TRCAUTHSTATUS	RO	Authentication Status Register
`0xFBC`	TRCDEVARCH	RO	Device Architecture Register
`0xFC0-0xFC4`	-	-	Reserved
`0xFC8`	TRCDEVID	RO	Device ID Register
`0xFCC`	TRCDEVTYPE	RO	Device Type Register
`0xFD0`	TRCPIDR4	RO	*Peripheral Identification Register 4* on page 12-543
`0xFD4`	TRCPIDR5	RO	*Peripheral Identification Register 5-7* on page 12-543
`0xFD8`	TRCPIDR6	RO	
`0xFDC`	TRCPIDR7	RO	
`0xFE0`	TRCPIDR0	RO	*Peripheral Identification Register 0* on page 12-540
`0xFE4`	TRCPIDR1	RO	*Peripheral Identification Register 1* on page 12-541
`0xFE8`	TRCPIDR2	RO	*Peripheral Identification Register 2* on page 12-541
`0xFEC`	TRCPIDR3	RO	*Peripheral Identification Register 3* on page 12-542
`0xFF0`	TRCCIDR0	RO	Component Identification Register 0
`0xFF4`	TRCCIDR1	RO	Component Identification Register 1
`0xFF8`	TRCCIDR2	RO	Component Identification Register 2
`0xFFC`	TRCCIDR3	RO	Component Identification Register 3

12.8 ETM register descriptions

This section describes the implementation-specific ETM trace unit registers in the Cortex-A73 processor.

Table 12-3 ETM trace unit register summary on page 12-490 provides cross-references to individual registers. The *Arm® Embedded Trace Macrocell Architecture Specification, ETMv4* describes the other ETM trace unit registers.

This section contains the following subsections:

- *12.8.48 Claim Tag Set Register* on page 12-537.
- *12.8.49 Claim Tag Clear Register* on page 12-538.
- *12.8.50 Device Affinity Register 0* on page 12-538.
- *12.8.51 Device Affinity Register 1* on page 12-539.
- *12.8.52 Peripheral Identification Registers* on page 12-540.

12.8.1 Trace Configuration Register

The TRCCONFIGR characteristics are:

Purpose	Controls the tracing options.
Usage constraints	• You must always program this register as part of trace unit initialization. • This register only accepts writes when the trace unit is disabled.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCCONFIGR bit assignments.

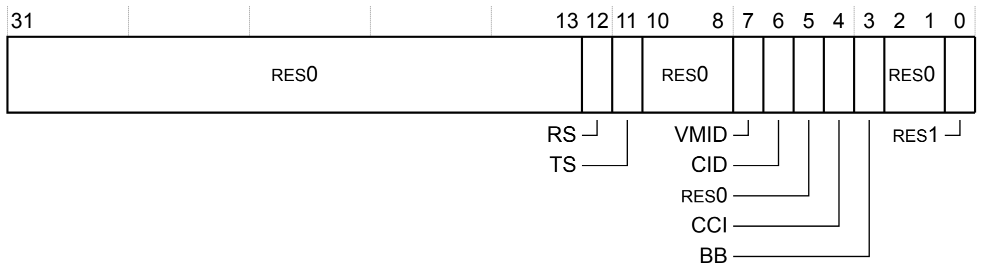

Figure 12-3 TRCCONFIGR bit assignments

The following table shows the TRCCONFIGR bit assignments.

Table 12-4 TRCCONFIGR bit assignments

Bits	Name	Function
[31:13]	-	Reserved, RES0.
[12]	RS	Return stack enable bit. The possible values are: `0` Return stack is disabled. `1` Return stack is enabled.
[11]	TS	Global timestamp tracing enable bit. The possible values are: `0` Global timestamp tracing is disabled. `1` Global timestamp tracing is enabled. TRCTSCTLR controls the insertion of timestamps in the trace.
[10:8]	-	Reserved, RES0.
[7]	VMID	VMID tracing enable bit. The possible values are: `0` VMID tracing is disabled. `1` VMID tracing is enabled.

Table 12-4 TRCCONFIGR bit assignments (continued)

Bits	Name	Function
[6]	CID	Context ID tracing enable bit. The possible values are: `0` Context ID tracing is disabled. `1` Context ID tracing is enabled.
[5]	-	Reserved, RES0.
[4]	CCI	Cycle counting in the instruction trace enable bit. The possible values are: `0` Cycle counting in the instruction trace is disabled. `1` Cycle counting in the instruction trace is enabled. TRCCCCTLR controls the threshold value for cycle counting.
[3]	BB	Branch broadcast mode enable bit. The possible values are: `0` Branch broadcast mode is disabled. `1` Branch broadcast mode is enabled. TRCBBCTLR controls which regions of memory are enabled to use branch broadcasting.
[2:1]	-	Reserved, RES0.
[0]	-	Reserved, RES1.

The TRCCONFIGR can be accessed through the external debug interface with offset `0x010`.

12.8.2 Auxiliary Control Register

The TRCAUXCTLR characteristics are:

Purpose The function of this register is to provide IMPLEMENTATION DEFINED configuration and control options.

Usage constraints This register only accepts writes when the trace unit is disabled.

Note

If trace debug tools set the value of this register to nonzero, then it might cause the behavior of a trace unit to contradict this architecture specification.

Configurations Available in all configurations.

Attributes A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

Note

This register is set to zero on a trace unit reset. Resetting this register to zero ensures that none of the IMPLEMENTATION DEFINED features are enabled by default, and that the trace unit resets to a known state (an ETMv4 trace unit with no IMPLEMENTATION DEFINED features enabled).

The following figure shows the TRCAUXCTLR bit assignments.

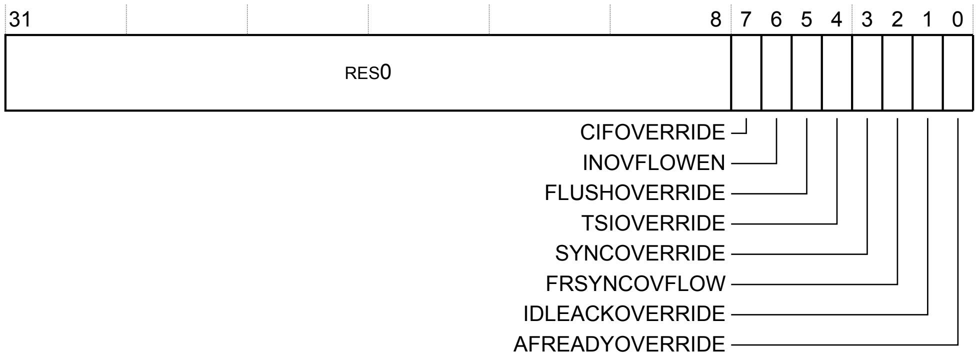

Figure 12-4 TRCAUXCTLR bit assignments

The following table shows the TRCAUXCTLR bit assignments.

Table 12-5 TRCAUXCTLR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7]	CIFOVERRIDE	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.
[6]	INOVFLOWEN	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.
[5]	FLUSHOVERRIDE	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.
[4]	TSIOVERRIDE	Override TS packet insertion behavior. The possible values are: `0` Timestamp packets are inserted into FIFO when trace activity is LOW. `1` Timestamp packets are inserted into FIFO irrespective of trace activity.
[3]	SYNCOVERRIDE	Override SYNC packet insertion behavior. The possible values are: `0` SYNC packets are inserted into FIFO when trace activity is LOW. `1` SYNC packets are inserted into FIFO irrespective of trace activity.

Table 12-5 TRCAUXCTLR bit assignments (continued)

Bits	Name	Function
[2]	FRSYNCOVFLOW	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.
[1]	IDLEACKOVERRIDE	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.
[0]	AFREADYOVERRIDE	Force assertion of **AFREADYM** output. The possible values are: `0` ETM trace unit **AFREADYM** output is asserted HIGH 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`, the trace unit behavior deviates from architecturally-specified behavior.

The TRCAUXCTLR can be accessed through the external debug interface with offset `0x018`.

12.8.3 Event Control 0 Register

The TRCEVENTCTL0R characteristics are:

Purpose	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.
Usage constraints	• You must always program this register as part of trace unit initialization. • This register accepts writes only when the trace unit is disabled.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCEVENTCTL0R bit assignments.

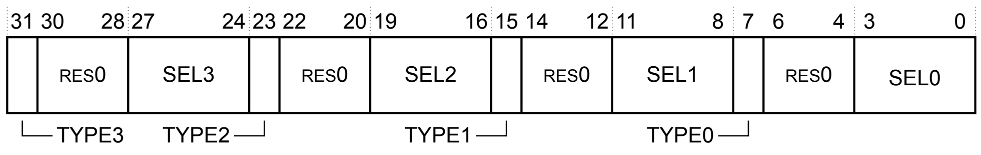

Figure 12-5 TRCEVENTCTL0R bit assignments

The following table shows the TRCEVENTCTL0R bit assignments.

Table 12-6 TRCEVENTCTL0R bit assignments

Bits	Name	Function
[31]	TYPE3	Selects the resource type for trace event 3: `0` Single selected resource. `1` Boolean combined resource pair.
[30:28]	-	Reserved, RES0.
[27:24]	SEL3	Selects the resource number, based on the value of TYPE3: When TYPE3 is `0`, SEL3 selects a single selected resource from 0-15 defined by bits[3:0]. When TYPE3 is `1`, SEL3 selects a Boolean combined resource pair from 0-7 defined by bits[2:0].
[23]	TYPE2	Selects the resource type for trace event 2: `0` Single selected resource. `1` Boolean combined resource pair.
[22:20]	-	Reserved, RES0.
[19:16]	SEL2	Selects the resource number, based on the value of TYPE2: When TYPE2 is `0`, SEL2 selects a single selected resource from 0-15 defined by bits[3:0]. When TYPE2 is `1`, SEL2 selects a Boolean combined resource pair from 0-7 defined by bits[2:0].
[15]	TYPE1	Selects the resource type for trace event 1: `0` Single selected resource. `1` Boolean combined resource pair.
[14:12]	-	Reserved, RES0.
[11:8]	SEL1	Selects the resource number, based on the value of TYPE1: When TYPE1 is `0`, SEL1 selects a single selected resource from 0-15 defined by bits[3:0]. When TYPE1 is `1`, SEL1 selects a Boolean combined resource pair from 0-7 defined by bits[2:0].
[7]	TYPE0	Selects the resource type for trace event 0: `0` Single selected resource. `1` Boolean combined resource pair.
[6:4]	-	Reserved, RES0.
[3:0]	SEL0	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].

The TRCEVENTCTL0R can be accessed through the external debug interface, offset `0x020`.

12.8.4 Event Control 1 Register

The TRCEVENTCTL1R characteristics are:

Purpose Controls the behavior of the events that TRCEVENTCTL0R selects.

Usage constraints
- You must always program this register as part of trace unit initialization.
- Accepts writes only when the trace unit is disabled.

Configurations Available in all configurations.

Attributes A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCEVENTCTL1R bit assignments.

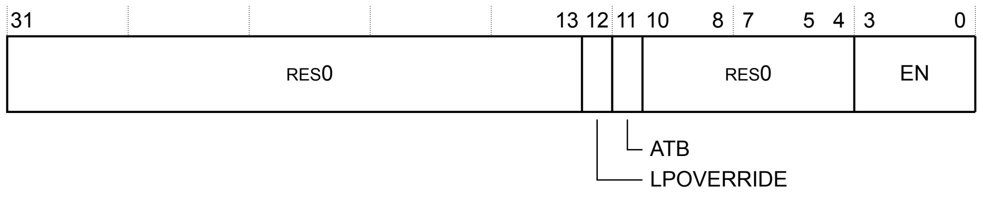

Figure 12-6 TRCEVENTCTL1R bit assignments

The following table shows the TRCEVENTCTL1R bit assignments.

Table 12-7 TRCEVENTCTL1R bit assignments

Bits	Name	Function
[31:13]	-	Reserved, RES0.
[12]	LPOVERRIDE	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.
[11]	ATB	ATB trigger enable: `0` ATB trigger disabled. `1` ATB trigger enabled.
[10:4]	-	Reserved, RES0.
[3:0]	EN	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.

The TRCEVENTCTL1R can be accessed through the external debug interface, offset `0x024`.

12.8.5 Stall Control Register

The TRCSTALLCTLR characteristics are:

Purpose Enables the ETM trace unit to stall the Cortex-A73 processor if the ETM trace unit FIFO overflows.

Usage constraints
- You must always program this register as part of trace unit initialization.
- This register accepts writes only when the trace unit is disabled.

Configurations Available in all configurations.

Attributes A 32-bit RW register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCSTALLCTLR bit assignments.

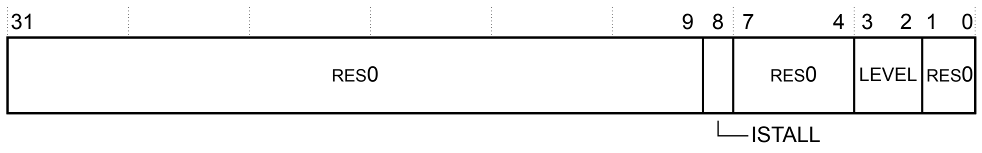

Figure 12-7 TRCSTALLCTLR bit assignments

The following table shows the TRCSTALLCTLR bit assignments.

Table 12-8 TRCSTALLCTLR bit assignments

Bits	Name	Function
[31:9]	-	Reserved, RES0.
[8]	ISTALL	Instruction stall bit. Controls if the trace unit can stall the processor when the instruction trace buffer space is less than LEVEL: `0` The trace unit does not stall the processor. `1` The trace unit can stall the processor.
[7:4]	-	Reserved, RES0.
[3:2]	LEVEL	Threshold level field. The field can support four monotonic levels from `0b00` to `0b11`, where: `0b00` Zero invasion. This setting has a greater risk of an ETM trace unit FIFO overflow. `0b11` Maximum invasion occurs but there is less risk of a FIFO overflow.
[1:0]	-	Reserved, RES0.

The TRCSTALLCTLR can be accessed through the external debug interface, offset `0x02C`.

12.8.6 Global Timestamp Control Register

The TRCTSCTLR characteristics are:

Purpose	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.
Usage constraints	• This register accepts writes only when the trace unit is disabled. • Must be programmed if TRCCONFIGR.TS == `1`.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCTSCTLR bit assignments:

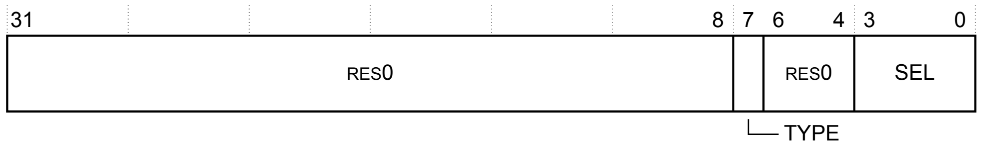

Figure 12-8 TRCTSCTLR bit assignments

The following table shows the TRCTSCTLR bit assignments:

Table 12-9 TRCTSCTLR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7]	TYPE	Single or combined resource selector.
[6:4]	-	Reserved, RES0.
[3:0]	SEL	Identifies the resource selector to use.

The TRCTSCTLR can be accessed through the external debug interface, offset `0x030`.

12.8.7 Synchronization Period Register

The TRCSYNCPR characteristics are:

Purpose Controls how often periodic trace synchronization requests occur.

Usage constraints
- You must always program this register as part of trace unit initialization.
- This register accepts writes only when the trace unit is disabled.

Configurations Available in all configurations.

Attributes A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCSYNCPR bit assignments.

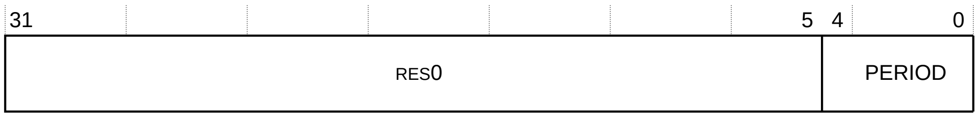

Figure 12-9 TRCSYNCPR bit assignments

The following table shows the TRCSYNCPR bit assignments.

Table 12-10 TRCSYNCPR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0.
[4:0]	PERIOD	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 `0b00000`, is `0b01000`, providing a minimum synchronization period of 256 bytes. • The maximum value is `0b10100`, providing a maximum synchronization period of 2^{20} bytes.

The TRCSYNCPR can be accessed through the external debug interface, offset `0x034`.

12.8.8 Cycle Count Control Register

The TRCCCCTLR characteristics are:

Purpose Sets the threshold value for cycle counting.

Usage constraints
- This register accepts writes only when the trace unit is disabled.
- You must program this register if TRCCONFIGR.CCI == `1`.

Configurations Available in all configurations.

Attributes A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCCCCTLR bit assignments.

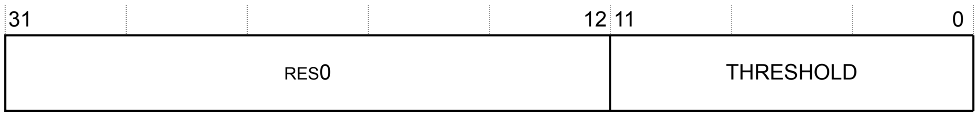

Figure 12-10 TRCCCCTLR bit assignments

The following table shows the TRCCCCTLR bit assignments.

Table 12-11 TRCCCCTLR bit assignments

Bits	Name	Function
[31:12]	-	Reserved, RES0.
[11:0]	THRESHOLD	Sets the threshold value for instruction trace cycle counting. The minimum threshold value that can be programmed into THRESHOLD is 4, as given in TRCIDR3.CCITMIN. See *Arm® ETM Architecture Specification, ETMv4* for more information.

The TRCCCCTLR can be accessed through the external debug interface, offset `0x038`.

12.8.9 Branch Broadcast Control Register

The TRCBBCTLR characteristics are:

Purpose Controls which regions in the memory map are enabled to use branch broadcasting.

Usage constraints
- This register only accepts writes when the trace unit is disabled.
- You must program this register if TRCCONFIGR.BB == `1`.

Configurations Available in all configurations.

Attributes A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCBBCTLR bit assignments.

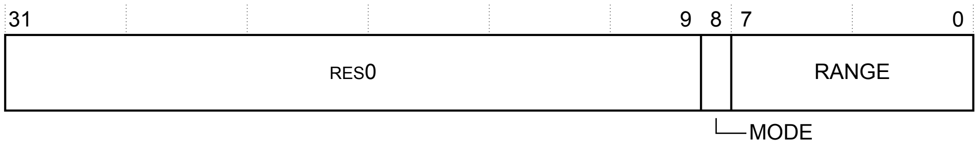

Figure 12-11 TRCBBCTLR bit assignments

The following table shows the TRCBBCTLR bit assignments.

Table 12-12 TRCBBCTLR bit assignments

Bits	Name	Function
[31:9]	-	Reserved, RES0.
[8]	MODE	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.
[7:0]	RANGE	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.

The TRCBBCTLR can be accessed through the external debug interface, offset `0x03C`.

12.8.10 Trace ID Register

The TRCTRACEIDR characteristics are:

Purpose Sets the trace ID for instruction trace.

Usage constraints
- You must always program this register as part of trace unit initialization.
- This register accepts writes only when the trace unit is disabled.

Configurations Available in all configurations.

Attributes A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCTRACEIDR bit assignments.

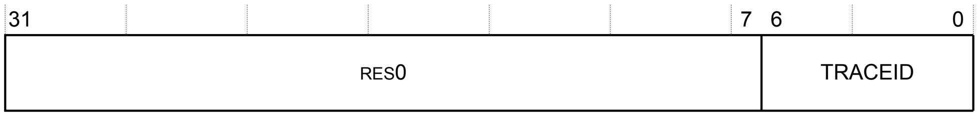

Figure 12-12 TRCTRACEIDR bit Assignments

The following table shows the TRCTRACEIDR bit assignments.

Table 12-13 TRCTRACEIDR bit assignments

Bits	Name	Function
[31:7]	-	Reserved, RES0.
[6:0]	TRACEID	Trace ID field. Sets the trace ID value for instruction trace.

The TRCTRACEIDR can be accessed through the external debug interface, offset `0x040`.

12.8.11 ViewInst Main Control Register

The TRCVICTLR characteristics are:

Purpose	Controls instruction trace filtering.
Usage constraints	This register: • Accepts writes only when the trace unit is disabled. • Returns stable data only when TRCSTATR.PMSTABLE == `1`. • Must be programmed, particularly to set the value of the SSSTATUS bit, that sets the state of the start-stop logic.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCVICTLR bit assignments.

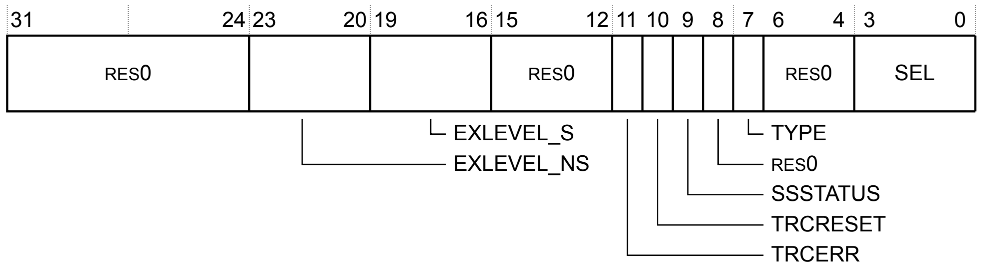

Figure 12-13 TRCVICTLR bit assignments

The following table shows the TRCVICTLR bit assignments.

Table 12-14 TRCVICTLR bit assignments

Bits	Name	Function
[31:24]	-	Reserved, RES0.
[23:20]	EXLEVEL_NS	In Non-secure state, each bit controls whether instruction tracing is enabled for the corresponding Exception level: `0` The trace unit generates instruction trace, in Non-secure state, for Exception level *n*. `1` The trace unit does not generate instruction trace, in Non-secure state, for Exception level *n*. **Note** The Exception levels are: **Bit[20]** Exception level 0. **Bit[21]** Exception level 1. **Bit[22]** Exception level 2. **Bit[23]** RES0. Instruction tracing is not implemented for Exception level 3.

Table 12-14 TRCVICTLR bit assignments (continued)

Bits	Name	Function
[19:16]	EXLEVEL_S	In Secure state, each bit controls whether instruction tracing is enabled for the corresponding Exception level: `0` The trace unit generates instruction trace, in Secure state, for Exception level *n*. `1` The trace unit does not generate instruction trace, in Secure state, for Exception level *n*. **Note** The Exception levels are: **Bit[16]** Exception level 0. **Bit[17]** Exception level 1. **Bit[18]** RES0. Instruction tracing is not implemented for Exception level 2. **Bit[19]** Exception level 3.
[15:12]	-	Reserved, RES0.
[11]	TRCERR	Controls whether a trace unit must trace a System error exception: `0` The trace unit does not trace a System error unless it traces the exception or instruction immediately prior to the System error exception. `1` The trace unit always traces a System error, regardless of the value of ViewInst.
[10]	TRCRESET	Controls whether a trace unit must trace a Reset exception: `0` The trace unit does not trace a Reset exception unless it traces the exception or instruction immediately prior to the Reset exception. `1` The trace unit always traces a Reset exception.
[9]	SSSTATUS	Indicates the current status of the start-stop logic: `0` The start-stop logic is in the stopped state. `1` The start-stop logic is in the started state.
[8]	-	Reserved, RES0.
[7]	TYPE	Selects the resource type for the ViewInst event: `0` Single selected resource. `1` Boolean combined resource pair.
[6:4]	-	Reserved, RES0.
[3:0]	SEL	Selects the resource number to use for the ViewInst event, based on the value of TYPE: • When TYPE is `0`, SEL selects a single selected resource from 0-15 defined by bits[3:0]. • When TYPE is `1`, SEL selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

The TRCVICTLR can be accessed through the external debug interface, offset `0x080`.

12.8.12 ViewInst Include-Exclude Control Register

The TRCVIIECTLR characteristics are:

Purpose Defines the address range comparators that control the ViewInst Include/Exclude control.

Usage constraints
- You must always program this register as part of trace unit initialization.
- This register accepts writes only when the trace unit is disabled.

Configurations Available in all configurations.

Attributes A 32-bit RW register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCVIIECTLR bit assignments.

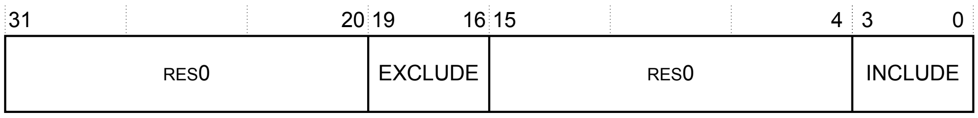

Figure 12-14 TRCVIIECTLR bit assignments

The following table shows the TRCVIIECTLR bit assignments.

Table 12-15 TRCVIIECTLR bit assignments

Bits	Name	Function
[31:20]	-	Reserved, RES0.
[19:16]	EXCLUDE	Defines the address range comparators for ViewInst exclude control. One bit is provided for each implemented address range comparator.
[15:4]	-	Reserved, RES0.
[3:0]	INCLUDE	Defines the address range comparators for ViewInst include control. Not selecting any 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.

The TRCVIIECTLR can be accessed through the external debug interface, offset `0x084`.

12.8.13 ViewInst Start-Stop Control Register

The TRCVISSCTLR characteristics are:

Purpose Defines the single address comparators that control the ViewInst Start-Stop logic.

Usage constraints
- You must always program this register as part of trace unit initialization.
- This register accepts writes only when the trace unit is disabled.

Configurations Available in all configurations.

Attributes A 32-bit RW register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCVISSCTLR bit assignments.

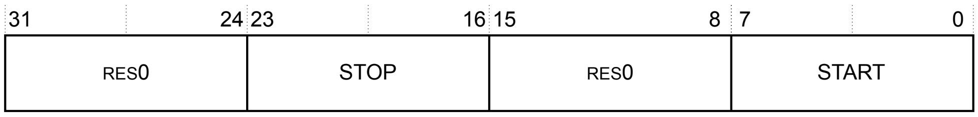

Figure 12-15 TRCVISSCTLR bit assignments

The following table shows the TRCVISSCTLR bit assignments.

Table 12-16 TRCVISSCTLR bit assignments

Bits	Name	Function
[31:24]	-	Reserved, RES0.
[23:16]	STOP	Defines the single address comparators to stop trace with the ViewInst Start-Stop control. One bit is provided for each implemented single address comparator.
[15:8]	-	Reserved, RES0.
[7:0]	START	Defines the single address comparators to start trace with the ViewInst Start-Stop control. One bit is provided for each implemented single address comparator.

The TRCVISSCTLR can be accessed through the external debug interface, offset `0x088`.

12.8.14 Sequencer State Transition Control Registers 0-2

The TRCSEQEVRn characteristics are:

Purpose	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.
Usage constraints	• These registers accept writes only when the trace unit is disabled. • Return stable data only when TRCSTATR.PMSTABLE == `1`. • Software must use these registers to set the initial state of the sequencer before the sequencer is used.
Configurations	Available in all configurations.
Attributes	32-bit RW registers. See also *Table 12-3 ETM trace unit register summary on page 12-490*.

The following figure shows the TRCSEQEVRn bit assignments.

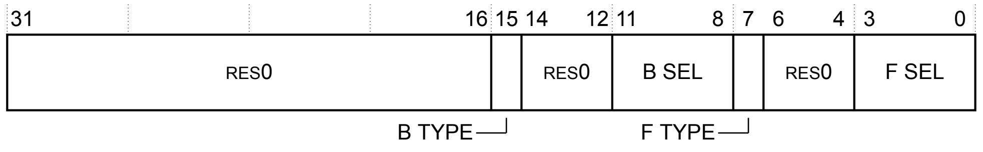

Figure 12-16 TRCSEQEVRn bit assignments

The following table shows the TRCSEQEVRn bit assignments.

Table 12-17 TRCSEQEVRn bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15]	B TYPE	Selects the resource type to move backwards to this state from the next state: `0` Single selected resource. `1` Boolean combined resource pair.
[14:12]	-	Reserved, RES0

Table 12-17 TRCSEQEVRn bit assignments (continued)

Bits	Name	Function
[11:8]	B SEL	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].
[7]	F TYPE	Selects the resource type to move forwards from this state to the next state: `0` Single selected resource. `1` Boolean combined resource pair.
[6:4]	-	Reserved, RES0.
[3:0]	F SEL	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].

The TRCSEQEVRn registers can be accessed through the external debug interface with offset `0x100` for TRCSEQEVR0, `0x104` for TRCSEQEVR1, and `0x108` for TRCSEQEVR2.

12.8.15 Sequencer Reset Control Register

The TRCSEQRSTEVR characteristics are:

Purpose	Moves the sequencer to state 0 when a programmed event occurs.
Usage constraints	• This register accepts writes only when the trace unit is disabled. • If the sequencer is used, you must program all sequencer state transitions with a valid event.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCSEQRSTEVR bit assignments.

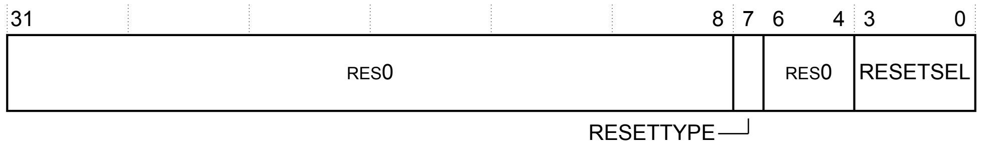

Figure 12-17 TRCSEQRSTEVR bit assignments

The following table shows the TRCSEQRSTEVR bit assignments.

Table 12-18 TRCSEQRSTEVR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7]	RESETTYPE	Selects the resource type to move back to state 0: `0` Single selected resource. `1` Boolean combined resource pair.

Table 12-18 TRCSEQRSTEVR bit assignments (continued)

Bits	Name	Function
[6:4]	-	Reserved, RES0.
[3:0]	RESETSEL	Selects the resource number, based on the value of RESETTYPE: When RESETTYPE is `0`, RESETSEL selects a single selected resource from 0-15 defined by bits[3:0]. When RESETTYPE is `1`, RESETSEL selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

The TRCSEQRSTEVR can be accessed through the external debug interface, offset `0x118`.

12.8.16 Sequencer State Register

The TRCSEQSTR characteristics are:

Purpose Use this register to set or read the sequencer state.

Usage constraints
- This register accepts writes only when the trace unit is disabled.
- Returns stable data only when TRCSTATR.PMSTABLE == `1`.
- Software must use this register to set the initial state of the sequencer before the sequencer is used.

Configurations Available in all configurations.

Attributes A 32-bit RW register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCSEQSTR bit assignments.

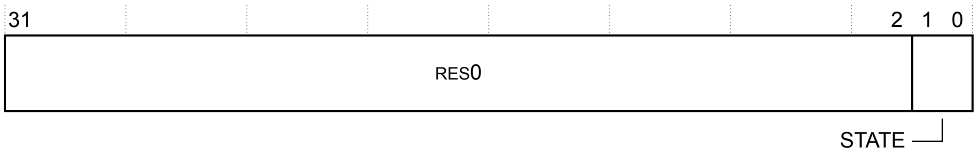

Figure 12-18 TRCSEQSTR bit assignments

The following table shows the TRCSEQSTR bit assignments.

Table 12-19 TRCSEQSTR bit assignments

Bits	Name	Function
[31:2]	-	Reserved, RES0.
[1:0]	STATE	Sets or returns the current sequencer state: `0b00` State 0. `0b01` State 1. `0b10` State 2. `0b11` State 3.

The TRCSEQSTR can be accessed through the external debug interface, offset `0x11c`.

12.8.17 External Input Select Register

The TRCEXTINSELR characteristics are:

Purpose Use this register to set, or read, which external inputs are resources to the trace unit.

Usage constraints This register accepts writes only when the trace unit is disabled.

Configurations Available in all configurations.

Attributes A 32-bit RW register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCEXTINSELR bit assignments.

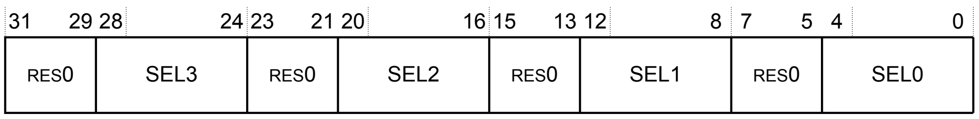

Figure 12-19 TRCEXTINSELR bit assignments

The following table shows the TRCEXTINSELR bit assignments.

Table 12-20 TRCEXTINSELR bit assignments

Bits	Name	Function
[31:29]	-	Reserved, RES0.
[28:24]	SEL3	Selects an event from the external input bus for External Input Resource 3.
[23:21]	-	Reserved, RES0.
[20:16]	SEL2	Selects an event from the external input bus for External Input Resource 2.
[15:13]	-	Reserved, RES0.
[12:8]	SEL1	Selects an event from the external input bus for External Input Resource 1.
[7:5]	-	Reserved, RES0.
[4:0]	SEL0	Selects an event from the external input bus for External Input Resource 0.

The TRCEXTINSELR can be accessed through the external debug interface, offset `0x120`.

12.8.18 Counter Reload Value Registers 0-1

The TRCCNTRLDVRn characteristics are:

Purpose Defines the reload value for the counter.

Usage constraints These registers accept writes only when the trace unit is disabled.

Configurations Available in all configurations.

Attributes Both are 32-bit RW trace registers. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCCNTRLDVRn bit assignments.

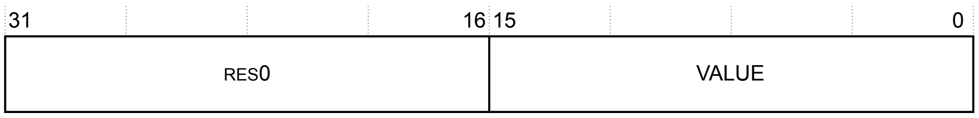

Figure 12-20 TRCCNTRLDVRn bit assignments

The following table shows the TRCCNTRLDVRn bit assignments.

Table 12-21 TRCCNTRLDVRn bit assignments

Bits	Value	Function
[31:16]	-	Reserved, RES0.
[15:0]	VALUE	Defines the reload value for the counter. This value is loaded into the counter each time the reload event occurs.

The TRCCNTRLDVRn registers can be accessed through external debug interface with offset `0x140` for TRCCNTRLDVR0 and `0x144` for TRCCNTRLDVR1.

12.8.19 Counter Control Register 0

The TRCCNTCTLR0 characteristics are:

Purpose	Controls the operation of counter 0.
Usage constraints	This register accepts writes only when the trace unit is disabled.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCCNTCTLR0 bit assignments.

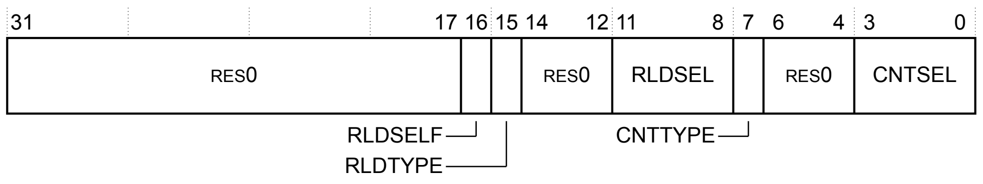

Figure 12-21 TRCCNTCTLR0 bit assignments

The following table shows the TRCCNTCTLR0 bit assignments.

Table 12-22 TRCCNTCTLR0 bit assignments

Bits	Name	Function
[31:17]	-	Reserved, RES0.
[16]	RLDSELF	Controls whether a reload event occurs for counter 0, when counter 0 reaches zero: `0` Counter 0 is in Normal mode. `1` Counter 0 is in Self-reload mode.
[15]	RLDTYPE	Selects the resource type for the reload: `0` Single selected resource. `1` Boolean combined resource pair.
[14:12]	-	Reserved, RES0.
[11:8]	RLDSEL	Selects the resource number, based on the value of RLDTYPE: When RLDTYPE is `0`, RLDSEL selects a single selected resource from 0-15 defined by bits[3:0]. When RLDTYPE is `1`, RLDSEL selects a Boolean combined resource pair from 0-7 defined by bits[2:0].
[7]	CNTTYPE	Selects the resource type for the counter: `0` Single selected resource. `1` Boolean combined resource pair.
[6:4]	-	Reserved, RES0.
[3:0]	CNTSEL	Selects the resource number, based on the value of CNTTYPE: When CNTTYPE is `0`, CNTSEL selects a single selected resource from 0-15 defined by bits[3:0]. When CNTTYPE is `1`, CNTSEL selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

The TRCCNTCTLR0 can be accessed through the external debug interface, offset `0x150`.

12.8.20 Counter Control Register 1

The TRCCNTCTLR1 characteristics are:

Purpose	Controls the operation of counter 1.
Usage constraints	This register accepts writes only when the trace unit is disabled.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCCNTCTLR1 bit assignments.

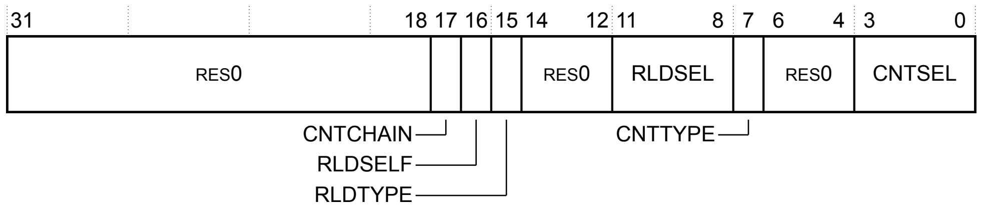

Figure 12-22 TRCCNTCTLR1 bit assignments

The following table shows the TRCCNTCTLR1 bit assignments.

Table 12-23 TRCCNTCTLR1 bit assignments

Bits	Name	Function
[31:18]	-	Reserved, RES0.
[17]	CNTCHAIN	Controls whether counter 1 decrements when a reload event occurs for counter 1. This enables two counters to be used in combination to provide a larger counter: `0` Counter 1 does not decrement when a reload event for counter 0 occurs. Counter 1 operates independently from counter 0. `1` Counter 0 decrements when a reload event for counter 0 occurs. This concatenates counter 0 and counter 1 to provide a larger count value.
[16]	RLDSELF	Controls whether a reload event occurs for counter 1, when counter 1 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.
[15]	RLDTYPE	Selects the resource type for the reload: `0` Single selected resource. `1` Boolean combined resource pair.
[14:12]	-	Reserved, RES0.
[11:8]	RLDSEL	Selects the resource number, based on the value of RLDTYPE: When RLDTYPE is `0`, RLDSEL selects a single selected resource from 0-15 defined by bits[3:0]. When RLDTYPE is `1`, RLDSEL selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Table 12-23 TRCCNTCTLR1 bit assignments (continued)

Bits	Name	Function
[7]	CNTTYPE	Selects the resource type for the counter: `0` Single selected resource. `1` Boolean combined resource pair.
[6:4]	-	Reserved, RES0.
[3:0]	CNTSEL	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].

The TRCCNTCTLR1 can be accessed through the external debug interface, offset `0x154`.

12.8.21 Counter Value Registers 0-1

The TRCCNTVRn characteristics are:

Purpose	Contains the current counter value.
Usage constraints	• These registers only accept writes when the ETM trace unit is disabled. • The count value is stable only when TRCSTATR.PMSTABLE == `1`. • If software uses counter 0 or 1, then it must write to this register to set the initial counter value.
Configurations	Available in all configurations.
Attributes	Both are 32-bit RW registers. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCCNTVRn bit assignments.

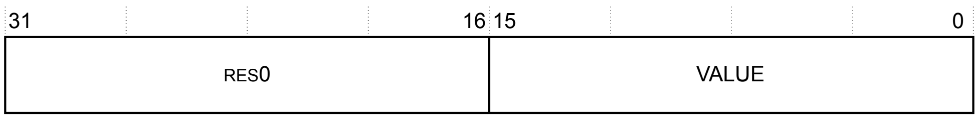

Figure 12-23 TRCCNTVRn bit assignments

The following table shows the TRCCNTVRn bit assignments.

Table 12-24 TRCCNTVRn bit assignments

Bits	Value	Function
[31:16]	-	Reserved, RES0.
[15:0]	VALUE	Contains the current counter value.

The TRCCNTVRn registers can be accessed through the external debug interface with offset `0x160` for TRCCNTVR0 and `0x164` for TRCCNTVR1.

12.8.22 ID Register 8

The TRCIDR8 characteristics are:

Purpose	Returns the maximum speculation depth of the instruction trace stream.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.

Attributes A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCIDR8 bit assignments.

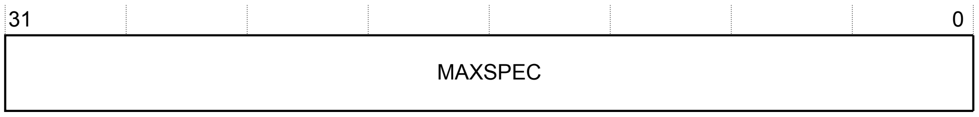

Figure 12-24 TRCIDR8 bit assignments

The following table shows the TRCIDR8 bit assignments.

Table 12-25 TRCIDR8 bit assignments

Bits	Name	Function
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction stream. This is the maximum number of P0 elements in the trace stream that can be speculative at any time. This value is: `0` No P0 elements can be speculative at any time.

The TRCIDR8 can be accessed through the external debug interface, offset `0x180`.

12.8.23 ID Register 9

The TRCIDR9 characteristics are:

Purpose Returns the number of P0 right-hand keys that the trace unit can use.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCIDR9 bit assignments.

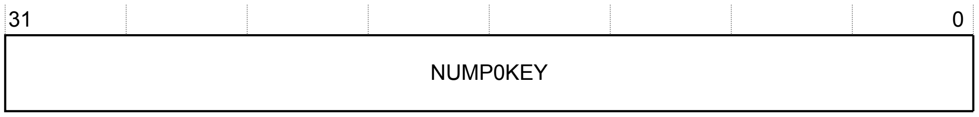

Figure 12-25 TRCIDR9 bit assignments

The following table shows the TRCIDR9 bit assignments.

Table 12-26 TRCIDR9 bit assignments

Bits	Name	Function
[31:0]	NUMP0KEY	Indicates the number of P0 right-hand keys that the trace unit can use. This value is: `0` The trace unit cannot use any P0 right-hand keys.

The TRCIDR9 can be accessed through the external debug interface, offset `0x184`.

12.8.24 ID Register 10

The TRCIDR10 characteristics are:

Purpose Returns the number of P1 right-hand keys that the trace unit can use.

Usage constraints There are no usage constraints.

Configurations	Available in all configurations.
Attributes	A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCIDR10 bit assignments.

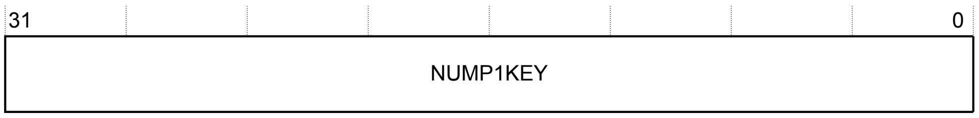

Figure 12-26 TRCIDR10 bit assignments

The following table shows the TRCIDR10 bit assignments.

Table 12-27 TRCIDR10 bit assignments

Bits	Name	Function
[31:0]	NUMP1KEY	Indicates the number of P1 right-hand keys that the trace unit can use. This value is: `0` The trace unit cannot use any P1 right-hand keys.

The TRCIDR10 can be accessed through the external debug interface, offset `0x188`.

12.8.25 ID Register 11

The TRCIDR11 characteristics are:

Purpose	Returns the number of special P1 right-hand keys that the trace unit can use.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCIDR11 bit assignments.

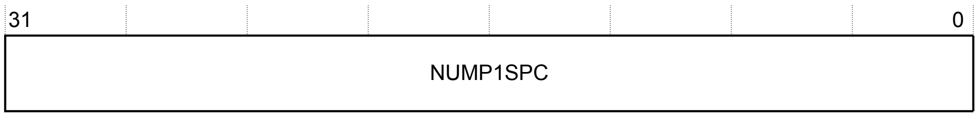

Figure 12-27 TRCIDR11 bit assignments

The following table shows the TRCIDR11 bit assignments.

Table 12-28 TRCIDR11 bit assignments

Bits	Name	Function
[31:0]	NUMP1SPC	Indicates the number of special P1 right-hand keys that the trace unit can use. This value is: `0` The trace unit cannot use any special P1 right-hand keys.

The TRCIDR11 can be accessed through the external debug interface, offset `0x18C`.

12.8.26 ID Register 12

The TRCIDR12 characteristics are:

Purpose	Returns the number of conditional instruction right-hand keys that the trace unit can use.

Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCIDR12 bit assignments.

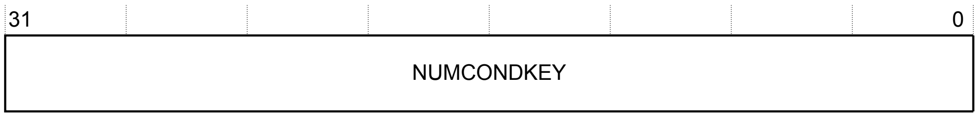

Figure 12-28 TRCIDR12 bit assignments

The following table shows the TRCIDR12 bit assignments.

Table 12-29 TRCIDR12 bit assignments

Bits	Name	Function
[31:0]	NUMCONDKEY	Indicates the number of conditional instruction right-hand keys that the trace unit can use. The number includes normal and special keys. This value is: `0` The trace unit cannot use any conditional instruction right-hand keys.

The TRCIDR12 can be accessed through the external debug interface, offset `0x190`.

12.8.27 ID Register 13

The TRCIDR13 characteristics are:

Purpose	Returns the number of special conditional instruction right-hand keys that the trace unit can use.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCIDR13 bit assignments.

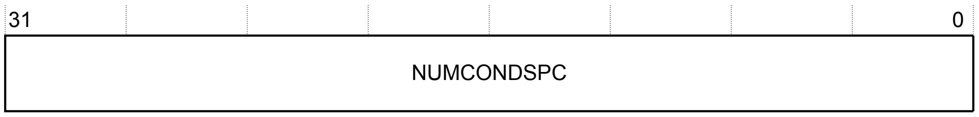

Figure 12-29 TRCIDR13 bit assignments

The following table shows the TRCIDR13 bit assignments.

Table 12-30 TRCIDR13 bit assignments

Bits	Name	Function
[31:0]	NUMCONDSPC	Indicates the number of special conditional instruction right-hand keys that the trace unit can use. `0` The trace unit cannot use any special conditional instruction right-hand keys.

The TRCIDR13 can be accessed through the external debug interface, offset `0x194`.

12.8.28 Implementation Specific Register 0

The TRCIMSPEC0 characteristics are:

Purpose	Shows the presence of any implementation specific features, and enables any features that are provided.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary on page 12-490*.

The following figure shows the TRCIMSPEC0 bit assignments.

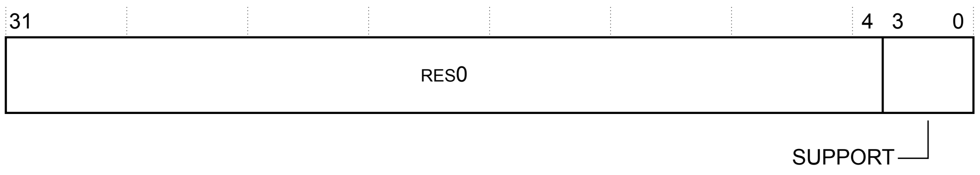

Figure 12-30 TRCIMSPEC0 bit assignments

The following table shows the TRCIMSPEC0 bit assignments.

Table 12-31 TRCIMSPEC0 bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	SUPPORT	`0` No implementation specific extensions are supported.

The TRCIMSPEC0 register can be accessed through the external debug interface, offset `0x1C0`.

12.8.29 ID Register 0

The TRCIDR0 characteristics are:

Purpose	Returns the tracing capabilities of the ETM trace unit.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary on page 12-490*.

The following figure shows the TRCIDR0 bit assignments.

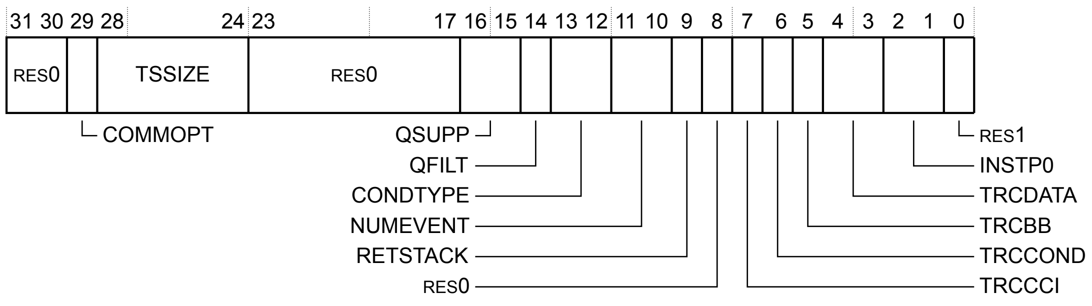

Figure 12-31 TRCIDR0 bit assignments

The following table shows the TRCIDR0 bit assignments.

Table 12-32 TRCIDR0 bit assignments

Bits	Name	Function
[31:30]	-	Reserved, RES0.
[29]	COMMOPT	Commit mode field. This value is: `1` Commit mode 1. See the *Arm® ETM Architecture Specification, ETMv4* for more information on how this affects certain packets.
[28:24]	TSSIZE	Global timestamp size. This value is: `0b01000` Implementation supports a maximum global timestamp of 64 bits.
[23:17]	-	Reserved, RES0.
[16:15]	QSUPP	Indicates Q element support. This value is: `0b00` Q element support is not implemented. TRCCONFIGR.QE is RES0.
[14]	QFILT	QFILT is RES0 when QSUPP==`0x00`.
[13:12]	CONDTYPE	Conditional tracing field. This value is: `0b00` The ETM trace unit indicates only if a conditional instruction passes or fails its condition code check.
[11:10]	NUMEVENT	Number of events supported in the trace, minus 1. This value is: `0b11` The ETM trace unit supports four events. This field controls how many fields are supported in TRCEVENTCTL0R and indicates the size of TRCEVENTCTRL1R.INSTEN.
[9]	RETSTACK	Return stack support. This value is: `1` Return stack is implemented, so TRCCONFIGR.RS is supported.
[8]	-	Reserved, RES0.
[7]	TRCCCI	Cycle counting instruction bit. Indicates whether the trace unit supports cycle counting for instructions. This value is: `1` Cycle counting in the instruction trace is implemented, therefore: • TRCCONFIGR.CCI is supported. • TRCCCCTLR is supported.
[6]	TRCCOND	Conditional instruction tracing support bit. This value is: `0` Conditional instruction tracing is not supported.
[5]	TRCBB	Branch broadcasting tracing support bit. Indicates whether the trace unit supports branch broadcast tracing. This value is: `1` Branch broadcast tracing is supported, therefore: • TRCCONFIGR.BB is supported. • TRCBBCTLR is supported.

Table 12-32 TRCIDR0 bit assignments (continued)

Bits	Name	Function
[4:3]	TRCDATA	Conditional tracing field. This value is: `0b00` Data tracing is not supported.
[2:1]	INSTP0	P0 tracing support field. Indicates support for tracing of load and store instructions as P0 elements. This value is: `0b00` Tracing of load and store instructions as P0 elements is not supported.
[0]	-	Reserved, RES1.

The TRCIDR0 can be accessed through the external debug interface with offset `0x1E0`.

12.8.30 ID Register 1

The TRCIDR1 characteristics are:

Purpose	Returns the base architecture of the trace unit.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCIDR1 bit assignments.

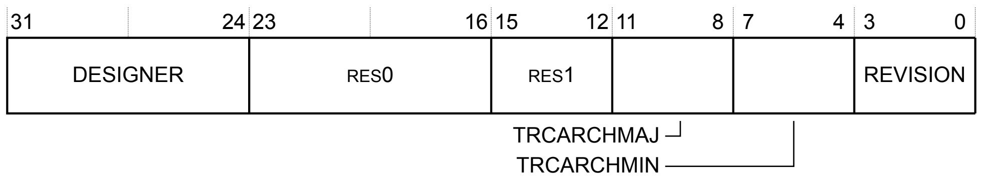

Figure 12-32 TRCIDR1 bit assignments

The following table shows the TRCIDR1 bit assignments.

Table 12-33 TRCIDR1 bit assignments

Bits	Name	Function
[31:24]	DESIGNER	Indicates which company designed the trace unit. This value is: `0x41` Arm Limited.
[23:16]	-	Reserved, RES0.
[15:12]	-	Reserved, RES1.
[11:8]	TRCARCHMAJ	Indicates the major version number of the trace unit architecture. This value is: `0b0100` ETMv4.

Table 12-33 TRCIDR1 bit assignments (continued)

Bits	Name	Function
[7:4]	TRCARCHMIN	Indicates the minor version number of the trace unit architecture. This value is: `0b0000` ETMv4 minor revision 0.
[3:0]	REVISION	Indicates the revision numbers of the trace registers and the OS Lock registers. This value is: `0b0000` r0p0.

The TRCIDR1 can be accessed through the external debug interface, offset `0x1E4`.

12.8.31 ID Register 2

The TRCIDR2 characteristics are:

Purpose Returns the maximum size of the following parameters in the trace unit:
- Cycle counter.
- Data value.
- Data address.
- VMID.
- Context ID.
- Instruction address.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary on page 12-490*.

The following figure shows the TRCIDR2 bit assignments.

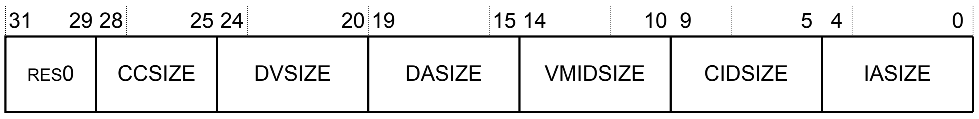

Figure 12-33 TRCIDR2 bit assignments

The following table shows the TRCIDR2 bit assignments.

Table 12-34 TRCIDR2 bit assignments

Bits	Name	Function
[31:29]	-	Reserved, RES0.
[28:25]	CCSIZE	Indicates the size of the cycle counter in bits minus 12. This value is: `0x0` The cycle counter is 12 bits in length.
[24:20]	DVSIZE	Indicates the data value size in bytes. This value is: `0x0` Data value tracing is not supported, as indicated by TRCIDR0.TRCDATA.
[19:15]	DASIZE	Indicates the data address size in bytes. This value is: `0x0` Data address tracing is not supported, as indicated by TRCIDR0.TRCDATA.

Table 12-34 TRCIDR2 bit assignments (continued)

Bits	Name	Function
[14:10]	VMIDSIZE	Indicates the Virtual Machine ID size. This value is: `0x1` Maximum of 8-bit VMID size, therefore TRCCONFIGR.VMID is supported.
[9:5]	CIDSIZE	Indicates the Context ID size in bytes. This value is: `0x4` Maximum of 32-bit Context ID size, therefore TRCCONFIGR.CID is supported.
[4:0]	IASIZE	Indicates the instruction address size in bytes. This value is: `0x8` Maximum of 64-bit address size.

The TRCIDR2 can be accessed through the external debug interface with offset `0x1E8`.

12.8.32 ID Register 3

The TRCIDR3 characteristics are:

Purpose Indicates:
- Whether TRCVICTLR.TRCERR is supported.
- The number of cores available for tracing.
- Whether 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 processor.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCIDR3 bit assignments.

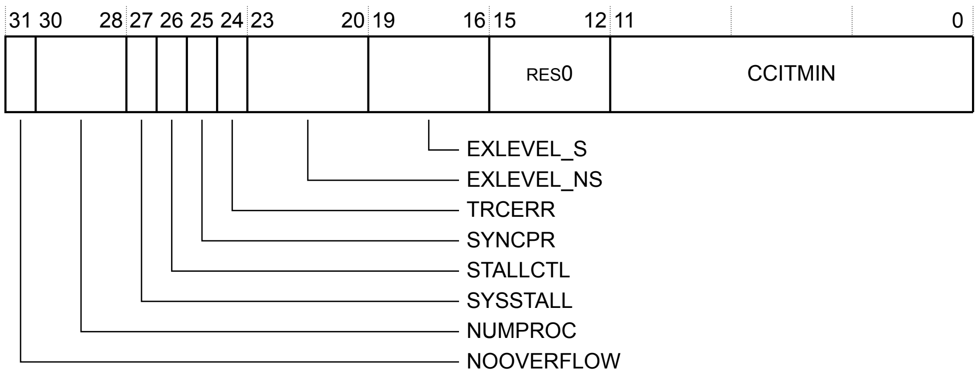

Figure 12-34 TRCIDR3 bit assignments

The following table shows the TRCIDR3 bit assignments.

Table 12-35 TRCIDR3 bit assignments

Bits	Name	Function
[31]	NOOVERFLOW	Indicates whether TRCSTALLCTLR.NOOVERFLOW is implemented. This value is: `0` TRCSTALLCTLR.NOOVERFLOW is not implemented.
[30:28]	NUMPROC	Indicates the number of cores available for tracing, minus one. This value is: `0b000` The trace unit can trace one core. ETM trace unit sharing is not supported.
[27]	SYSSTALL	Indicates whether stall control is implemented. This value is: `1` The system supports core stall control.
[26]	STALLCTL	Indicates whether TRCSTALLCTLR is implemented. This value is: `1` TRCSTALLCTLR is implemented. This field is used in conjunction with SYSSTALL.
[25]	SYNCPR	Indicates whether there is a fixed synchronization period. This value is: `0` TRCSYNCPR is read-write so software can change the synchronization period.
[24]	TRCERR	Indicates whether TRCVICTLR.TRCERR is supported. This value is: `1` TRCVICTLR.TRCERR is supported.
[23:20]	EXLEVEL_NS	In Non-secure state, each bit indicates whether instruction tracing is supported for the corresponding Exception level. This value is: `0b0111` Instruction tracing is supported for Non-secure EL0, EL1, and EL2 Exception levels.
[19:16]	EXLEVEL_S	In Secure state, each bit indicates whether instruction tracing is supported for the corresponding Exception level. This value is: `0b1011` Instruction tracing is supported for Secure EL0, EL1, and EL3 exception levels.
[15:12]	-	Reserved, RES0.
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in TRCCCCTLR.THRESHOLD. This value is: `0x004` The minimum instruction trace cycle counting threshold is 4.

The TRCIDR3 can be accessed through the external debug interface, offset `0x1EC`.

12.8.33 ID Register 4

The TRCIDR4 characteristics are:

Purpose	Returns how many resources the trace unit supports.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	A 32-bit RO register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCIDR4 bit assignments.

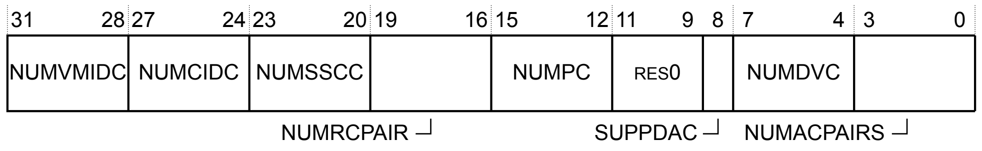

Figure 12-35 TRCIDR4 bit assignments

The following table shows the TRCIDR4 bit assignments.

Table 12-36 TRCIDR4 bit assignments

Bits	Name	Function
[31:28]	NUMVMIDC	Indicates the number of VMID comparators that are available for tracing. This value is: `0b0001` One VMID comparator is available.
[27:24]	NUMCIDC	Indicates the number of Context ID comparators that are available for tracing. This value is: `0b0001` One Context ID comparator is available.
[23:20]	NUMSSCC	Indicates the number of single-shot comparator controls that are available for tracing. This value is: `0b0001` One single-shot comparator control is available.
[19:16]	NUMRSPAIR	Indicates the number of resource selection pairs that are available for tracing, minus 1. This value is: `0b0111` Eight resource selection pairs are available.
[15:12]	NUMPC	Indicates the number of processor comparator inputs that are available for tracing. This value is: `0b0000` No processor comparator inputs are available.
[11:9]	-	Reserved, RES0.
[8]	SUPPDAC	Indicates data address comparison support. This value is: `0` Data address comparisons are not supported.
[7:4]	NUMDVC	Indicates the number of data value comparators that are available for tracing. This value is: `0b0000` No data value comparators are available.
[3:0]	NUMACPAIRS	Indicates the number of address comparator pairs that are available for tracing. This value is: `0b0100` Four address comparator pairs are available.

The TRCIDR4 can be accessed through the external debug interface with offset `0x1F0`.

12.8.34 ID Register 5

The TRCIDR5 characteristics are:

Purpose	Returns how many resources the trace unit supports.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	A 32-bit RO trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCIDR5 bit assignments.

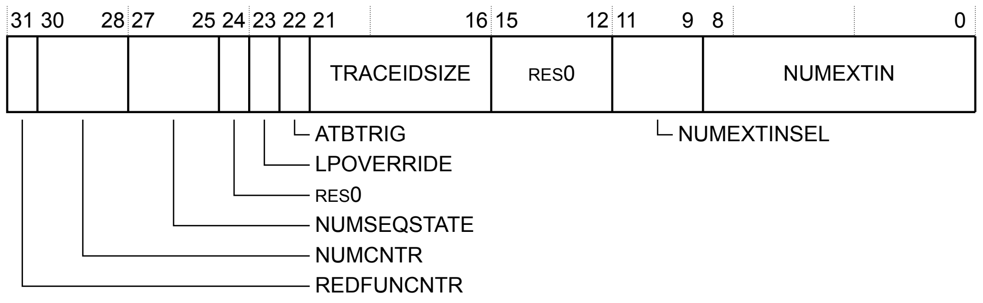

Figure 12-36 TRCIDR5 bit assignments

The following table shows the TRCIDR5 bit assignments.

Table 12-37 TRCIDR5 bit assignments

Bits	Name	Function
[31]	REDFUNCNTR	Indicates whether the reduced function counter is supported. This value is: `0` The reduced function counter is not supported.
[30:28]	NUMCNTR	Indicates the number of counters that are available for tracing. This value is: `0b010` Two counters are available.
[27:25]	NUMSEQSTATE	Indicates the number of sequencer states that are implemented. This value is: `0b100` Four sequencer states are implemented.
[24]	-	Reserved, RES0.
[23]	LPOVERRIDE	Indicates whether low power state override is supported. This value is: `1` Low power state override support is supported.
[22]	ATBTRIG	Indicates whether ATB triggers are supported. This value is: `1` ATB triggers are supported and TRCEVENTCTL1R.ATB is implemented.
[21:16]	TRACEIDSIZE	Indicates trace ID width. This value is: `0x07` A 7-bit trace ID is supported. This defines the width of the TRCTRACEIDR.TRACEID field. **Note** The CoreSight ATB requires a 7-bit trace ID width.
[15:12]	-	Reserved, RES0.
[11:9]	NUMEXTINSEL	Indicates the number of external input selectors that are implemented. This value is: `0b100` Four external input selectors are implemented.
[8:0]	NUMEXTIN	Indicates the number of external inputs that are implemented. This value is: `0b000011110` 30 external inputs implemented.

The TRCIDR5 can be accessed through the external debug interface with offset `0x1F4`.

12.8.35 Resource Selection Control Registers 2-16

The TRCRSCTLRn characteristics are:

Purpose Controls the trace resources.

There are eight resource pairs, the first pair is predefined as {0,1,pair=0} and reserves select registers. This leaves seven pairs to be implemented as programmable selectors.

Usage constraints
- This register accepts writes only when the trace unit is disabled.
- If software selects a non-implemented resource then CONSTRAINED UNPREDICTABLE behavior of the resource selector occurs. The resource selector might activate unexpectedly or might not activate. Reads of the TRCRSCTLRn might return UNKNOWN.

Configurations Resource selectors are implemented in pairs and there are eight pairs of TRCRSCTLR registers implemented, set by TRCIDR4.NUMRSPAIR. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Resource selector pair 0 is always implemented and is reserved. Resource selector zero always returns FALSE, and resource selector one always returns TRUE.

Attributes A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary on page 12-490*.

The following figure shows the TRCRSCTLRn bit assignments.

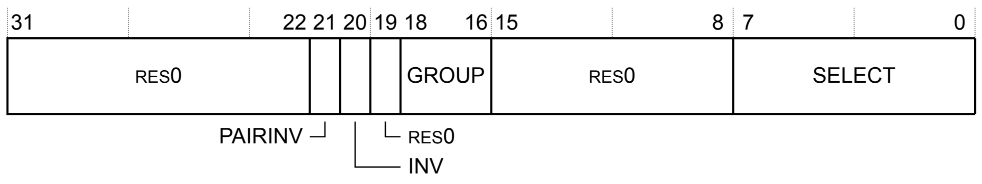

Figure 12-37 TRCRSCTLRn bit assignments

The following table shows the TRCRSCTLRn bit assignments.

Table 12-38 TRCRSCTLRn bit assignments

Bits	Name	Function
[31:22]	-	Reserved, RES0.
[21]	PAIRINV	Controls whether the combined result from a resource pair is inverted when *n* is even. The possible values are: `0` The combined result is not inverted. `1` The combined result is inverted. PAIRINV is RES0 when n is odd.
[20]	INV	Controls whether the resource that GROUP and SELECT selects is inverted. The possible values are: `0` The selected resource is not inverted. `1` The selected resource is inverted.
[19]	-	Reserved, RES0.

Table 12-38 TRCRSCTLRn bit assignments (continued)

Bits	Name	Function
[18:16]	GROUP	Selects a group of resources. See the *Arm® ETM Architecture Specification, ETMv4* for more information.
[15:8]	-	Reserved, RES0.
[7:0]	SELECT	Selects one or more resources from the group that the GROUP field selects. Each bit represents a resource from the selected group. See the *Arm® ETM Architecture Specification, ETMv4* for more information.

The TRCRSCTLRn can be accessed through the external debug interface, offset `0x208-023C`.

12.8.36 Single-Shot Comparator Control Register 0

The TRCSSCCR0 characteristics are:

Purpose	Controls the single-shot comparator resource.
Usage constraints	This register accepts writes only when the trace unit is disabled.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCSSCCR0 bit assignments.

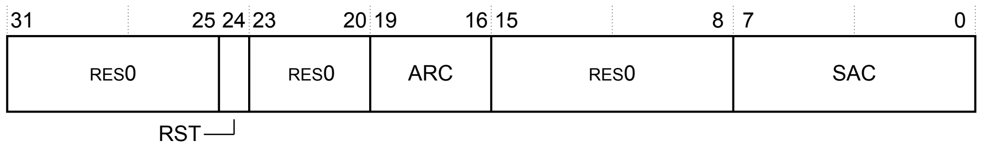

Figure 12-38 TRCSSCCR0 bit assignments

The following table shows the TRCSSCCR0 bit assignments.

Table 12-39 TRCSSCCR0 bit assignments

Bits	Name	Function
[31:25]	-	Reserved, RES0.
[24]	RST	Controls whether the single-shot comparator resource is reset when it fires. The possible values are: `0` The single-shot comparator resource is not reset when it fires. `1` The single-shot comparator resource is reset when it fires. This enables the single-shot comparator resource to fire multiple times.
[23:20]	-	Reserved, RES0.
[19:16]	ARC	Selects one or more address range comparators for single-shot control. One bit is provided for each implemented address range comparator.
[15:8]	-	Reserved, RES0.
[7:0]	SAC	Selects one or more single address comparators for single-shot control. One bit is provided for each implemented single address comparator.

The TRCSSCCR0 can be accessed through the external debug interface, offset `0x280`.

12.8.37 Single-Shot Comparator Status Register 0

The TRCSSCSR0 characteristics are:

Purpose Indicates the status of the single-shot comparator:
- TRCSSCSR0 is sensitive to instruction addresses.

Usage constraints
- Accepts writes only when the trace unit is disabled.
- The STATUS bit value is stable only when TRCSTATR.PMSTABLE == `1`.

Configurations Available in all configurations.

Attributes A 32-bit register, some fields are RW and others are RO. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCSSCSR0 bit assignments.

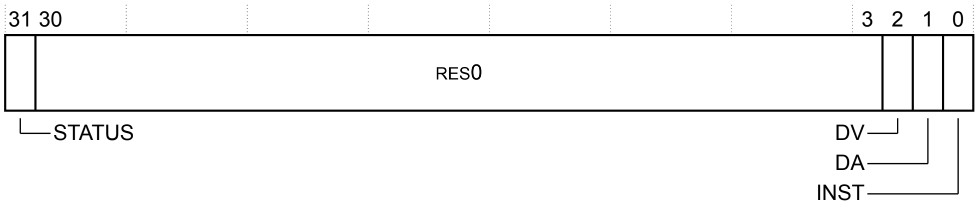

Figure 12-39 TRCSSCSR0 bit assignments

The following table shows the TRCSSCSR0 bit assignments.

Table 12-40 TRCSSCSR0 bit assignments

Bits	Name	Function
[31]	STATUS	Single-shot status. This indicates whether any of the selected comparators, that TRCSSCCR0.SAC or TRCSSCCR0.ARC selects, have matched: `0` No match has occurred. When the first match occurs, this field takes a value of `0b1`. It remains at `0b1` until explicitly modified by a write to this register. `1` One or more matches has occurred. If TRCSSCCRn.RST == `0` then: • There is only one match and no more matches are possible. • You must reset this bit to `0` to re-enable single-shot control.
[30:3]	-	Reserved, RES0.
[2]	DV	Indicates if the trace unit supports data address with data value comparisons. This field is read-only. This value is: `0` Single-shot data address with data value comparisons are not supported.
[1]	DA	Indicates if the trace unit supports data address comparisons. This field is read-only. This value is: `0` Single-shot data address comparisons are not supported.
[0]	INST	Indicates if the trace unit supports instruction address comparisons. This field is read-only. This value is: `1` Single-shot instruction address comparisons are supported.

The TRCSSCSR0 can be accessed through the external debug interface, offset `0x2A0`.

12.8.38 Address Comparator Value Registers 0-7

The TRCACVRn characteristics are:

Purpose	Indicates the address for the address comparators.
Usage constraints	This register accepts writes only when the trace unit is disabled.
Configurations	Available in all configurations.
Attributes	A 64-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCACVRn bit assignments.

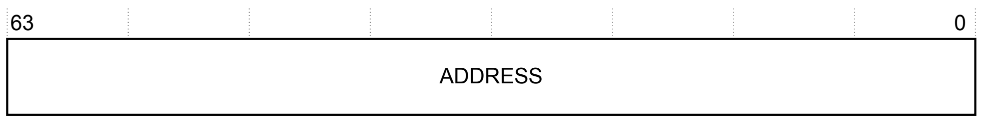

Figure 12-40 TRCACVRn bit assignments

The following table shows the TRCACVRn bit assignments.

Table 12-41 TRCACVRn bit assignments

Bits	Name	Function
[63:0]	ADDRESS	The address value to compare against. See the *Arm® ETM Architecture Specification, ETMv4* for more information.

The TRCACVRn can be accessed through the external debug interface, offset `0x400-0x43C`.

12.8.39 Address Comparator Access Type Registers 0-7

The TRCACATRn characteristics are:

Purpose Defines the type of access for the corresponding TRCACVRn. This register configures the context type, Exception levels, alignment and masking that is applied by the address comparator, together with how the address comparator behaves when it is one half of an address range comparator.

Usage constraints
- This register accepts writes only when the trace unit is disabled.
- If software uses two single address comparators as an address range comparator then it must program the corresponding TRCACATR registers with identical values in the following fields:
  - — TYPE
  - — CONTEXTTYPE
  - — EXLEVEL_S
  - — EXLEVEL_NS

Configurations Available in all configurations.

Attributes A 64-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCACATRn bit assignments.

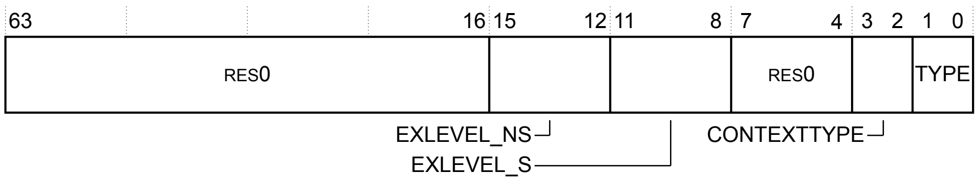

Figure 12-41 TRCACATRn bit assignments

The following table shows the TRCACATRn bit assignments.

Table 12-42 TRCACATRn bit assignments

Bits	Name	Function
[63:16]	-	Reserved, RES0.
[15:12]	EXLEVEL_NS	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*. —— **Note** —— The exception levels are: **Bit[12]** Exception level 0. **Bit[13]** Exception level 1. **Bit[14]** Exception level 2. **Bit[15]** Always RES0.
[11:8]	EXLEVEL_S	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*. —— **Note** —— The Exception levels are: **Bit[8]** Exception level 0. **Bit[9]** Exception level 1. **Bit[10]** Always RES0. **Bit[11]** Exception level 3.
[7:4]	-	Reserved, RES0.
[3:2]	CONTEXTTYPE	Controls whether the trace unit performs a Context ID comparison, a VMID comparison, or both comparisons. The possible values are: `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.
[1:0]	TYPE	Controls what type of comparison the trace unit performs. This value is: `0b00` Instruction address, RES0.

The TRCACATR*n* can be accessed through the external debug interface, offset `0x480-0x4B8`.

12.8.40 Context ID Comparator Value Register 0

The TRCCIDCVR0 characteristics are:

Purpose	Contains a Context ID value.
Usage constraints	This register accepts writes only when the trace unit is disabled.
Configurations	Available in all configurations.
Attributes	A 64-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCCIDCVR0 bit assignments.

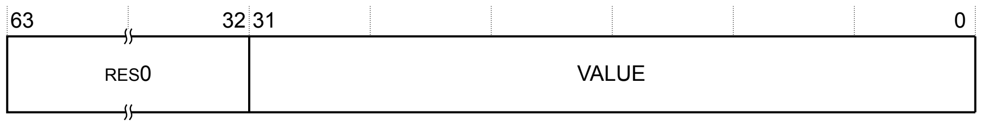

Figure 12-42 TRCCIDCVR0 bit assignments

The following table shows the TRCCIDCVR0 bit assignments.

Table 12-43 TRCCIDCVR0 bit assignments

Bits	Name	Function
[63:32]	-	Reserved, RES0.
[31:0]	VALUE	Context ID value.

The TRCCIDCVR0 can be accessed through the external debug interface, offset `0x600`.

12.8.41 VMID Comparator Value Register 0

The TRCVMIDCVR0 characteristics are:

Purpose	Contains a VMID value.
Usage constraints	This register accepts writes only when the trace unit is disabled.
Configurations	Available in all configurations.
Attributes	A 64-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCVMIDCVR0 bit assignments.

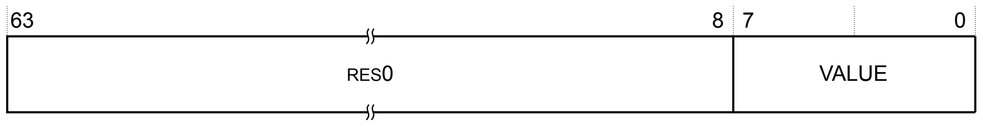

Figure 12-43 TRCVMIDCVR0 bit assignments

The following table shows the TRCVMIDCVR0 bit assignments.

Table 12-44 TRCVMIDCVR0 bit assignments

Bits	Name	Function
[63:8]	-	Reserved, RES0
[7:0]	VALUE	The VMID value.

The TRCVMIDCVR0 can be accessed through the external debug interface, offset `0x640`.

12.8.42 Context ID Comparator Control Register 0

The TRCCIDCCTLR0 characteristics are:

Purpose Contains Context ID mask values for TRCCIDCVR0.

Usage constraints
- This register accepts writes only when the trace unit is disabled.
- If software uses TRCCIDCVR0, then it must program this register.
- If software sets a mask bit to `1` then it must program the relevant byte in TRCCIDCVR0 to `0x00`.

Configurations Available in all configurations.

Attributes A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCCIDCCTLR0 bit assignments.

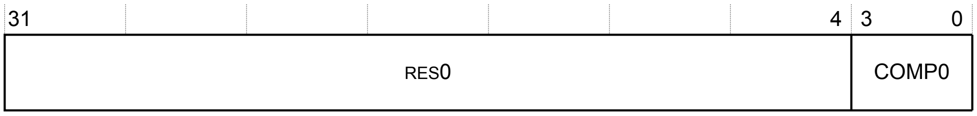

Figure 12-44 TRCCIDCCTLR0 bit assignments

The following table shows the TRCCIDCCTLR0 bit assignments.

Table 12-45 TRCCIDCCTLR0 bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	COMP0	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.

The TRCCIDCCTLR0 can be accessed through the external debug interface, offset `0x680`.

12.8.43 Integration ATB Identification Register

The TRCITATBIDR characteristics are:

Purpose Sets the state of output pins.

Usage constraints
- Available when bit[0] of TRCITCTRL is set to `1`.
- The value of the register sets the signals on the output pins when the register is written.

Configurations Available in all configurations.

Attributes A 32-bit RW register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCITATBIDR bit assignments.

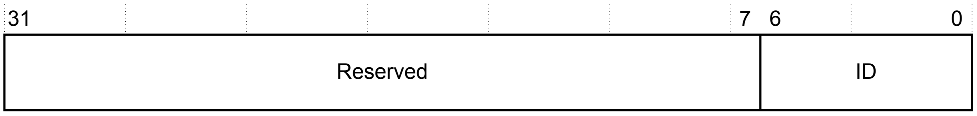

Figure 12-45 TRCITATBIDR bit assignments

The following table shows the TRCITATBIDR bit assignments.

Table 12-46 TRCITATBIDR bit assignments

Bits	Name	Function
[31:7]	-	Reserved. Read undefined.
[6:0]	ID	Drives the **ATIDMn[6:0]** output pins[dk].

The TRCITATBIDR can be accessed through the external debug interface, offset `0xEE4`.

12.8.44 Integration Instruction ATB Data Register

The TRCITIDATAR characteristics are:

Purpose	Sets the state of the **ATDATAMn** output pins.
Usage constraints	• Available when bit[0] of TRCITCTRL is set to `1`. • The value of the register sets the signals on the output pins when the register is written.
Configurations	Available in all configurations.
Attributes	A 32-bit RW register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCITIDATAR bit assignments.

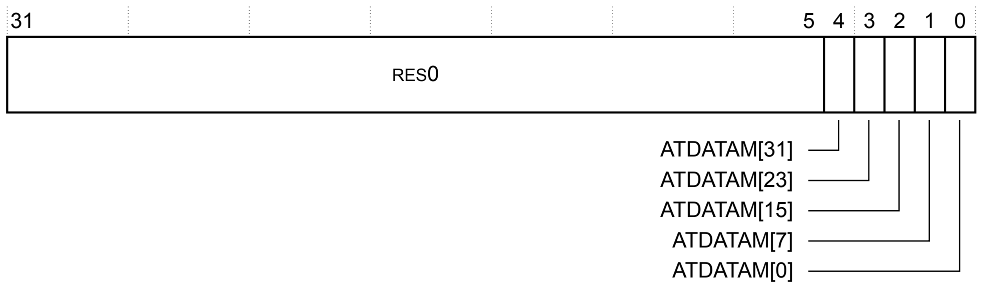

Figure 12-46 TRCITIDATAR bit assignments

The following table shows the TRCITIDATAR bit assignments.

Table 12-47 TRCITIDATAR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0
[4]	ATDATAM[31]	Drives the **ATDATAM[31]** output[dl]
[3]	ATDATAM[23]	Drives the **ATDATAM[23]** output[dl]
[2]	ATDATAM[15]	Drives the **ATDATAM[15]** output[dl]
[1]	ATDATAM[7]	Drives the **ATDATAM[7]** output[dl]
[0]	ATDATAM[0]	Drives the **ATDATAM[0]** output[dl]

dk When a bit is set to `0`, the corresponding output pin is LOW. When a bit is set to `1`, the corresponding output pin is HIGH. The TRCITATBIDR bit values correspond to the physical state of the output pins.

The TRCITIDATAR can be accessed through the external debug interface, offset `0xEEC`.

[dl] When a bit is set to `0`, the corresponding output pin is LOW. When a bit is set to `1`, the corresponding output pin is HIGH. The TRCITIDATAR bit values correspond to the physical state of the output pins.

12.8.45 Integration Instruction ATB In Register

The TRCITIATBINR characteristics are:

Purpose	Reads the state of the input pins.
Usage constraints	• Available when bit[0] of TRCITCTRL is set to `1`. • The values of the register bits depend on the signals on the input pins when the register is read.
Configurations	Available in all configurations.
Attributes	A 32-bit RW register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCITIATBINR bit assignments.

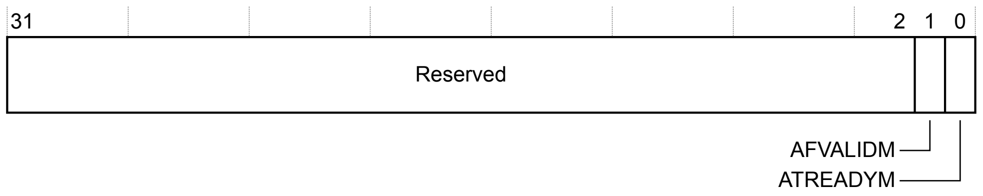

Figure 12-47 TRCITIATBINR bit assignments

The following table shows the TRCITIATBINR bit assignments.

Table 12-48 TRCITIATBINR bit assignments

Bits	Name	Function
[31:2]	-	Reserved. Read undefined.
[1]	AFVALIDM	Returns the value of the **AFVALIDMn** input pin[dm].
[0]	ATREADYM	Returns the value of the **ATREADYMn** input pin[dm].

The TRCITIATBINR can be accessed through the external debug interface, offset `0xEF4`.

[dm] When an input pin is LOW, the corresponding register bit is `0`. When an input pin is HIGH, the corresponding register bit is `1`. The TRCITIATBINR bit values always correspond to the physical state of the input pins.

12.8.46 Integration Instruction ATB Out Register

The TRCITIATBOUTR characteristics are:

Purpose	Sets the state of the output pins.
Usage constraints	• Available when bit[0] of TRCITCTRL is set to `1`. • The value of the register sets the signals on the output pins when the register is written.
Configurations	Available in all configurations.
Attributes	A 32-bit RW register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCITIATBOUTR bit assignments.

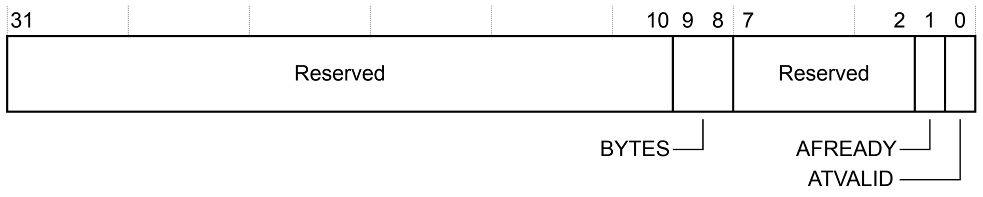

Figure 12-48 TRCITIATBOUTR bit assignments

The following table shows the TRCITIATBOUTR bit assignments.

Table 12-49 TRCITIATBOUTR bit assignments

Bits	Name	Function
[31:10]	-	Reserved. Read undefined.
[9:8]	BYTES	Drives the **ATBYTESMn[1:0]** output pins[dn].
[7:2]	-	Reserved. Read undefined.
[1]	AFREADY	Drives the **AFREADYMn** output pin[dn].
[0]	ATVALID	Drives the **ATVALIDMn** output pin[dn].

The TRCITIATBOUTR can be accessed through the external debug interface, offset `0xEFC`.

12.8.47 Integration Mode Control Register

The TRCITCTRL characteristics are:

Purpose	Controls whether the trace unit is in integration mode.
Usage constraints	• Accessible only from the memory-mapped interface or from an external agent such as a debugger. • Arm recommends that you perform a debug reset after using integration mode.
Configurations	Available in all configurations.
Attributes	A 32-bit management register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCITCTRL bit assignments.

[dn] When a bit is set to `0`, the corresponding output pin is LOW. When a bit is set to `1`, the corresponding output pin is HIGH. The TRCITIATBOUTR bit values always correspond to the physical state of the output pins.

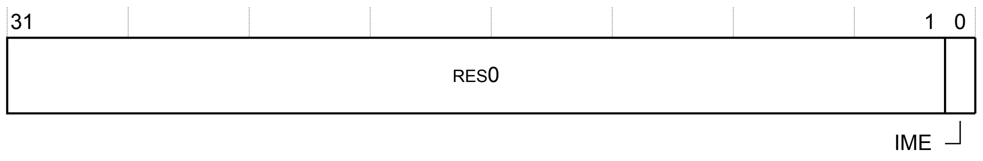

Figure 12-49 TRCITCTRL bit assignments

The following table shows the TRCITCTRL bit assignments.

Table 12-50 TRCITCTRL bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	IME	Integration mode enable bit. The possible values are: `0` The trace unit is not in integration mode. `1` The trace unit is in integration mode. This mode enables: • A debug agent to perform topology detection. • SoC test software to perform integration testing.

The TRCITCTRL register can be accessed through the external debug interface, offset `0xF00`.

12.8.48 Claim Tag Set Register

The TRCCLAIMSET characteristics are:

Purpose	Sets bits in the claim tag to `1` and determines the number of bits that the claim tag supports.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary on page 12-490*.

The following figure shows the TRCCLAIMSET bit assignments.

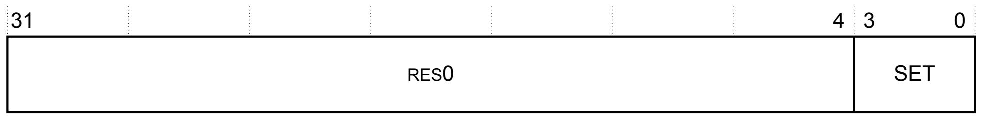

Figure 12-50 TRCCLAIMSET bit assignments

The following table shows the TRCCLAIMSET bit assignments.

Table 12-51 TRCCLAIMSET bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	SET	When a write to one of these bits occurs, with the value: `0` The register ignores the write. `1` If the bit is supported, then it is set to `1`. When a read occurs, the supported bits in the SET field are RES0 and therefore the value the register returns indicates how many SET bits are supported.

The TRCCLAIMSET can be accessed through the the external debug interface, offset `0xFA0`.

12.8.49 Claim Tag Clear Register

The TRCCLAIMCLR characteristics are:

Purpose	Clears bits in the claim tag to `0` and determines the current value of the claim tag.
Usage constraints	There are no usage constraints.
Configurations	Available in all configurations.
Attributes	A 32-bit RW trace register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCCLAIMCLR bit assignments.

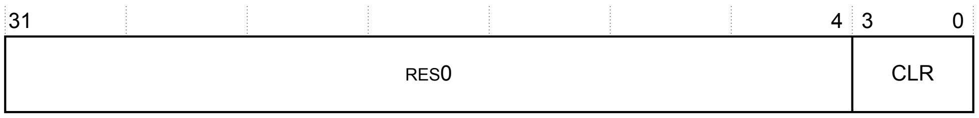

Figure 12-51 TRCCLAIMCLR bit assignments

The following table shows the TRCCLAIMCLR bit assignments.

Table 12-52 TRCCLAIMCLR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	CLR	When a write to one of these bits occurs, with the value: `0` The register ignores the write. `1` If the bit is supported, then it is set to `0`. A read returns the value of the claim tag.

The TRCCLAIMCLR can be accessed through the external debug interface, offset `0xFA4`.

12.8.50 Device Affinity Register 0

The TRCDEVAFF0 characteristics are:

Purpose TRCDEVAFF0 returns the lower 32 bits of the processor MPIDR_EL1, that is, MPIDR_EL1[31:0]. This enables a debugger to determine which processor in a multiprocessor system the trace unit relates to.

TRCDEVAFF0 is a read-only copy of MPIDR_EL1 accessible from the external debug interface.

Usage constraints This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	RO	RO	RO	RO	RO

Configurations Available in all configurations.

Attributes A 32-bit RO management register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCDEVAFF0 bit assignments.

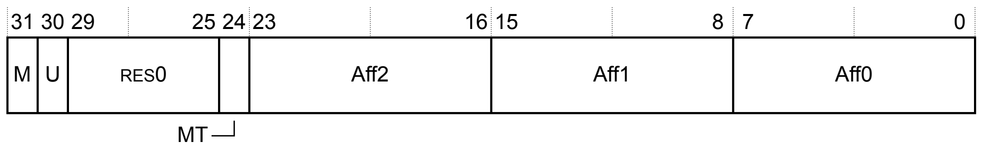

Figure 12-52 TRCDEVAFF0 bit assignments

The following table shows the TRCDEVAFF0 bit assignments.

Table 12-53 TRCDEVAFF0 bit assignments

Bits	Name	Function
[31]	M	RES1. Indicates support for the Multiprocessing Extensions.
[30]	U	Indicates a single core system, as distinct from core 0 in a cluster. This value is: `0` Core is part of a cluster.
[29:25]	-	Reserved, RES0.
[24]	MT	Indicates whether the lowest level of affinity consists of logical cores that are implemented using a multi-threading type approach. This value is: `0` The performance of cores at the lowest affinity level is largely independent.
[23:16]	Aff2	Affinity level 2. The least significant affinity level field. Indicates the value read in the **CLUSTERIDAFF2** configuration signal.
[15:8]	Aff1	Affinity level 1. The intermediate affinity level for this core in the cluster. Indicates the value read in the **CLUSTERIDAFF1[7:0]** signal.
[7:0]	Aff0	Affinity level 0. The most significant affinity level field for this core in the cluster. Indicates the core number in the Cortex-A73 processor: `0x0` Core is CPU0. `0x1` Core is CPU1. `0x2` Core is CPU2. `0x3` Core is CPU3.

12.8.51 Device Affinity Register 1

The TRCDEVAFF1 characteristics are:

Purpose	Returns the upper 32 bits of MPIDR_EL1, that is, MPIDR_EL1[63:32]. This enables a debugger to determine which core in a multiprocessor system the trace unit relates to.
Usage constraints	Accessible only from the external debugger interface.
Configurations	Available in all configurations.
Attributes	A 32-bit RO management register. See also *Table 12-3 ETM trace unit register summary* on page 12-490. For the Cortex-A73 processor, MPIDR_EL1[63:32] is RES0 and so TRCDEVAFF1 is RES0.

The TRCDEVAFF1 can be accessed through the external debug interface, offset `0xFAC`.

12.8.52 Peripheral Identification Registers

The Peripheral Identification Registers provide standard information required for all CoreSight components. They are a set of eight registers, listed in register number order in the following table.

Table 12-54 Summary of the Peripheral ID Registers

Register	Value	Offset
Peripheral ID4	`0x04`	`0xFD0`
Peripheral ID5	`0x00`	`0xFD4`
Peripheral ID6	`0x00`	`0xFD8`
Peripheral ID7	`0x00`	`0xFDC`
Peripheral ID0	`0x59`	`0xFE0`
Peripheral ID1	`0xB9`	`0xFE4`
Peripheral ID2	`0x0B`	`0xFE8`
Peripheral ID3	`0x00`	`0xFEC`

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

The Peripheral ID registers are:
- *Peripheral Identification Register 0* on page 12-540.
- *Peripheral Identification Register 1* on page 12-541.
- *Peripheral Identification Register 2* on page 12-541.
- *Peripheral Identification Register 3* on page 12-542.
- *Peripheral Identification Register 4* on page 12-543.
- *Peripheral Identification Register 5-7* on page 12-543.

Peripheral Identification Register 0

The TRCPIDR0 characteristics are:

Purpose	Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the trace unit.
Usage constraints	• Only bits[7:0] are valid. • Accessible only from the external debugger interface, offset `0xFE0`.
Configurations	Available in all configurations.
Attributes	A 32-bit RO management register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCPIDR0 bit assignments.

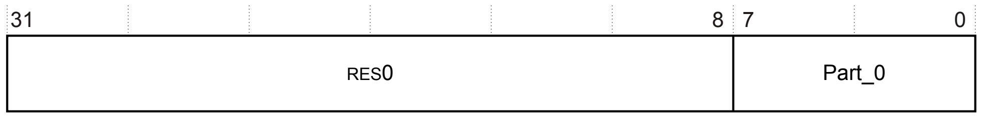

Figure 12-53 TRCPIDR0 bit assignments

The following table shows the TRCPIDR0 bit assignments.

Table 12-55 TRCPIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	Part_0	`0x59` Least significant byte of the ETM trace unit part number.

Peripheral Identification Register 1

The TRCPIDR1 characteristics are:

Purpose Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the trace unit.

Usage constraints
- Only bits[7:0] are valid.
- Accessible only from the memory-mapped interface or the external debugger interface.

Configurations Available in all configurations.

Attributes A 32-bit RO management register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCPIDR1 bit assignments.

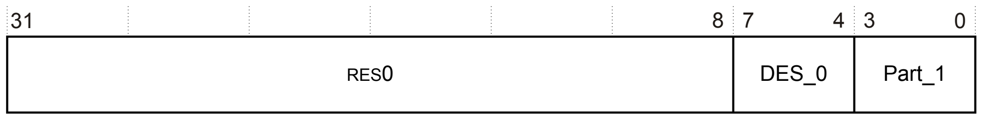

Figure 12-54 TRCPIDR1 bit assignments

The following table shows the TRCPIDR1 bit assignments.

Table 12-56 TRCPIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	DES_0	`0xB` Arm Limited. This is bits[3:0] of JEP106 ID code.
[3:0]	Part_1	`0x9` Most significant four bits of the ETM trace unit part number.

The TRCPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

Peripheral Identification Register 2

The TRCPIDR2 characteristics are:

Purpose Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the trace unit.

Usage constraints
- Only bits[7:0] are valid.
- Accessible only from the memory-mapped interface or the external debugger interface.

Configurations Available in all configurations.

Attributes A 32-bit RO management register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCPIDR2 bit assignments.

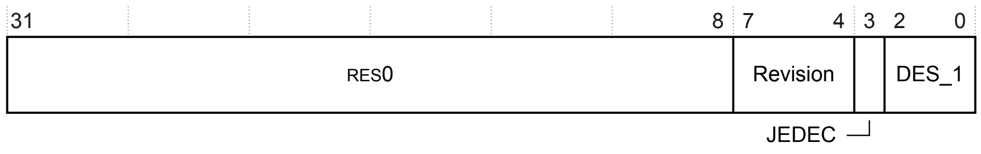

Figure 12-55 TRCPIDR2 bit assignments

The following table shows the TRCPIDR2 bit assignments.

Table 12-57 TRCPIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Revision	`0x0` r0p0.
[3]	JEDEC	`0b1` RES1. Indicates a JEP106 identity code is used.
[2:0]	DES_1	`0b011` Arm Limited. This is bits[6:4] of JEP106 ID code.

The TRCPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

Peripheral Identification Register 3

The TRCPIDR3 characteristics are:

Purpose Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the trace unit.

Usage constraints
- Only bits[7:0] are valid.
- Accessible only from the memory-mapped interface or the external debugger interface.

Configurations Available in all configurations.

Attributes A 32-bit RO management register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCPIDR3 bit assignments.

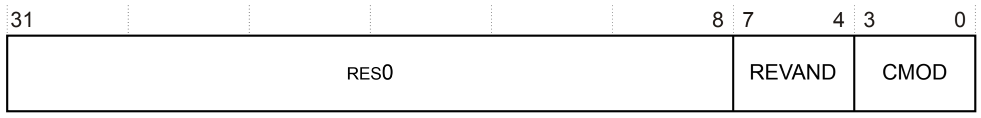

Figure 12-56 TRCPIDR3 bit assignments

The following table shows the TRCPIDR3 bit assignments.

Table 12-58 TRCPIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	REVAND	`0x0` Part minor revision.
[3:0]	CMOD	`0x0` Not customer modified.

The TRCPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

Peripheral Identification Register 4

The TRCPIDR4 characteristics are:

Purpose	Returns information that helps identify the peripheral. If software reads the TRCPIDR[7:0] register group then it can determine the 64-bit CoreSight Peripheral ID for the trace unit.
Usage constraints	• Only bits[7:0] are valid. • Accessible only from the memory-mapped interface or the external debugger interface.
Configurations	Available in all configurations.
Attributes	A 32-bit RO management register. See also *Table 12-3 ETM trace unit register summary* on page 12-490.

The following figure shows the TRCPIDR4 bit assignments.

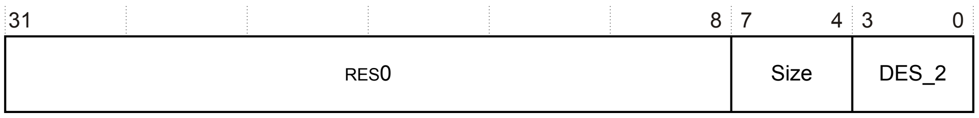

Figure 12-57 TRCPIDR4 bit assignments

The following table shows the TRCPIDR4 bit assignments.

Table 12-59 TRCPIDR4 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Size	`0x0` This indicates that the trace unit memory map occupies 4KB.
[3:0]	DES_2	`0x4` Arm Limited. This is bits[3:0] of the JEP106 continuation code.

The TRCPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

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.

12.9 Interaction with Debug and Performance Monitoring Unit

This section describes the interaction of the ETM with the PMU and Debug.

This section contains the following subsections:

- *12.9.1 Interaction with the Performance Monitoring Unit* on page 12-544.
- *12.9.2 Effect of Debug double lock on trace register access* on page 12-544.

12.9.1 Interaction with the Performance Monitoring Unit

The Cortex-A73 processor includes a *Performance Monitoring Unit* (PMU) that enables events, such as cache misses and instructions executed, to be counted over a period of time.

See *Chapter 11 Performance Monitor Unit* on page 11-442 for more information.

Use of PMU events by the ETM trace unit

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

The ETM trace unit uses four extended external input selectors to access the PMU events. Each selector can independently select one of the PMU events, that are then active for the cycles where the relevant events occur. These selected events can then be accessed by any of the event registers within the ETM trace unit. *11.9 Events* on page 11-475 describes the PMU events. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information on PMU events.

12.9.2 Effect of Debug double lock on trace register access

All trace register accesses through the memory-mapped and external debug interfaces behave as if the processor power domain is powered down when Debug double lock is set.

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

Chapter 13
Cross Trigger

This chapter describes the cross trigger interface for the Cortex-A73 processor.

It contains the following sections:

13.1 About the cross trigger

The Cortex-A73 processor has a single external cross trigger channel interface. This external interface is connected to the CoreSight *Cross Trigger Interface* (CTI) corresponding to each core through a *Cross Trigger Matrix* (CTM). A number of *Embedded Cross Trigger* (ECT) trigger inputs and trigger outputs are connected between debug components in the Cortex-A73 processor and CoreSight CTI blocks.

The CTI enables the debug logic, ETM trace unit, and PMU to interact with each other and with other CoreSight components. This is called cross triggering. For example, you configure the CTI to generate an interrupt when the ETM trace unit trigger event occurs.

The following figure shows the debug system components and the available trigger inputs and trigger outputs.

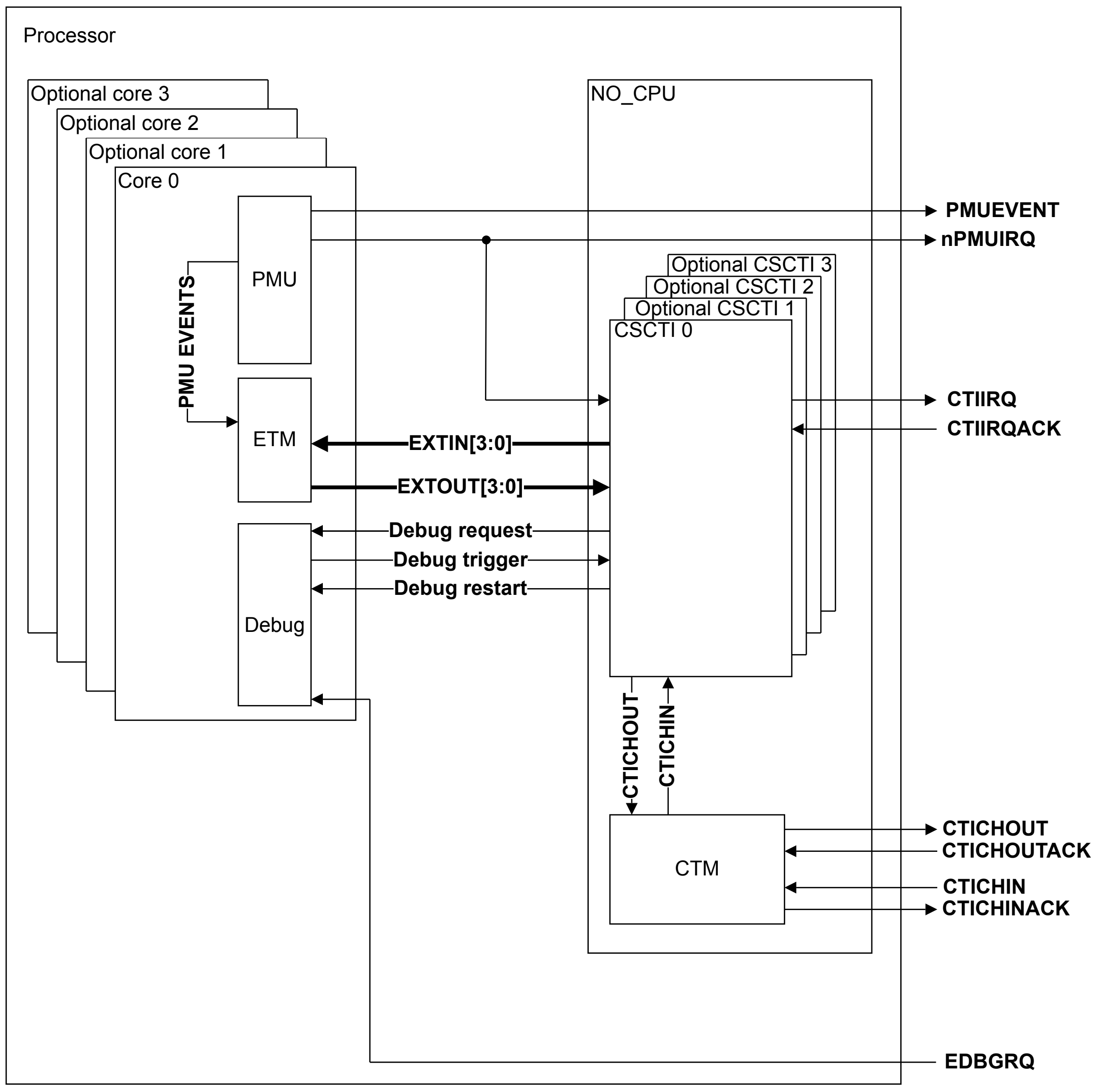

Figure 13-1 Debug system components

13.2 Trigger inputs and outputs

This section describes the trigger inputs and trigger outputs that are available to the CTI.

The following table shows the CTI inputs.

Table 13-1 Trigger inputs

CTI input	Name	Description
0	**DBGTRIGGER**, pulsed	Pulsed on entry to Debug state
1	**PMUIRQ**[do]	PMU generated interrupt
2	-	-
3	-	-
4	**EXTOUT[0]**	ETM trace unit external output
5	**EXTOUT[1]**	ETM trace unit external output
6	**EXTOUT[2]**	ETM trace unit external output
7	**EXTOUT[3]**	ETM trace unit external output

The following table shows the CTI outputs.

Table 13-2 Trigger outputs

CTI output	Name	Description
0	**EDBGRQ**	Causes the processor to enter Debug state
1	**DBGRESTART**	Causes the processor to exit Debug state
2	**CTIIRQ**	CTI interrupt
3	-	-
4	**EXTIN[0]**	ETM trace unit external input
5	**EXTIN[1]**	ETM trace unit external input
6	**EXTIN[2]**	ETM trace unit external input
7	**EXTIN[3]**	ETM trace unit external input

[do] This signal is the same as **nPMUIRQ** with inverted polarity.

13.3 Cortex-A73 CTM

The CoreSight CTI channel signals from all the cores are combined using a *Cross Trigger Matrix* (CTM) block so that a single cross trigger channel interface is presented in the Cortex-A73 processor. This module can combine up to four internal channel interfaces corresponding to each core along with one external channel interface.

In the Cortex-A73 processor CTM, the external channel output is driven by the OR output of all internal channel outputs. Each internal channel input is driven by the OR output of internal channel outputs of all other CTIs in addition to the external channel input.

13.4 Cross trigger register summary

This section describes the cross trigger registers in the Cortex-A73 processor. These registers are accessed through the external debug interface.

The following table gives a summary of the Cortex-A73 cross trigger registers. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table 13-3 Cross trigger register summary

Offset	Name	Type	Description
`0x000`	CTICONTROL	RW	CTI Control Register
`0x000-0x00C`	-	-	Reserved
`0x010`	CTIINTACK	WO	CTI Output Trigger Acknowledge Register
`0x014`	CTIAPPSET	RW	CTI Application Trigger Set Register
`0x018`	CTIAPPCLEAR	WO	CTI Application Trigger Clear Register
`0x01C`	CTIAPPPULSE	WO	CTI Application Pulse Register
`0x020`	CTIINEN0	RW	CTI Input Trigger to Output Channel Enable Registers
`0x024`	CTIINEN1	RW	
`0x028`	CTIINEN2	RW	
`0x02C`	CTIINEN3	RW	
`0x030`	CTIINEN4	RW	
`0x034`	CTIINEN5	RW	
`0x038`	CTIINEN6	RW	
`0x03C`	CTIINEN7	RW	
`0x040-0x09C`	-	-	Reserved
`0x0A0`	CTIOUTEN0	RW	CTI Input Channel to Output Trigger Enable Registers
`0x0A4`	CTIOUTEN1	RW	
`0x0A8`	CTIOUTEN2	RW	
`0x0AC`	CTIOUTEN3	RW	
`0x0B0`	CTIOUTEN4	RW	
`0x0B4`	CTIOUTEN5	RW	
`0x0B8`	CTIOUTEN6	RW	
`0x0BC`	CTIOUTEN7	RW	
`0x0C0-0x12C`	-	-	Reserved
`0x130`	CTITRIGINSTATUS	RO	CTI Trigger In Status Register
`0x134`	CTITRIGOUTSTATUS	RO	CTI Trigger Out Status Register
`0x138`	CTICHINSTATUS	RO	CTI Channel In Status Register
`0x13C`	CTICHOUTSTATUS	RO	CTI Channel Out Status Register
`0x140`	CTIGATE	RW	CTI Channel Gate Enable Register

Table 13-3 Cross trigger register summary (continued)

Offset	Name	Type	Description
0x144-0xF7C	-	-	Reserved
0xF00	CTIITCTRL	RW	*13.5.2 CTI Integration Mode Control Register* on page 13-553
0xF04-0xFA4	-	-	Reserved
0xFA0	CTICLAIMSET	RW	CTI Claim Tag Set Register
0xFA4	CTICLAIMCLR	RW	CTI Claim Tag Clear Register
0xFA8	CTIDEVAFF0	RO	CTI Device Affinity Register 0
0xFAC	CTIDEVAFF1	RO	CTI Device Affinity Register 1
0xFB0	CTILAR	WO	CTI Lock Access Register
0xFB4	CTILSR	RO	CTI Lock Status Register
0xFB8	CTIAUTHSTATUS	RO	CTI Authentication Status Register
0xFBC	CTIDEVARCH	RO	CTI Device Architecture Register
0xFC0	CTIDEVID2	RO	CTI Device ID Register 2
0xFC4	CTIDEVID1	RO	CTI Device ID Register 1
0xFC8	CTIDEVID	RO	*13.5.1 CTI Device Identification Register* on page 13-552
0xFCC	CTIDEVTYPE	RO	CTI Device Type Register
0xFD0	CTIPIDR4	RO	*Peripheral Identification Register 4* on page 13-557
0xFD4	CTIPIDR5	RO	*Peripheral Identification Register 5-7* on page 13-558
0xFD8	CTIPIDR6	RO	
0xFDC	CTIPIDR7	RO	
0xFE0	CTIPIDR0	RO	*Peripheral Identification Register 0* on page 13-554
0xFE4	CTIPIDR1	RO	*Peripheral Identification Register 1* on page 13-555
0xFE8	CTIPIDR2	RO	*Peripheral Identification Register 2* on page 13-555
0xFEC	CTIPIDR3	RO	*Peripheral Identification Register 3* on page 13-556
0xFF0	CTICIDR0	RO	*Component Identification Register 0* on page 13-558
0xFF4	CTICIDR1	RO	*Component Identification Register 1* on page 13-559
0xFF8	CTICIDR2	RO	*Component Identification Register 2* on page 13-559
0xFFC	CTICIDR3	RO	*Component Identification Register 3* on page 13-560

13.4.1 External register access permissions

External access permission to the cross trigger registers is subject to the conditions at the time of the access. The following table describes the processor response to accesses through the external debug and memory-mapped interfaces.

Table 13-4 External register conditions

Name	Condition	Description
Off	EDPRSR.PU is `0`	Processor power domain is completely off, or in a low-power state where the processor power domain registers cannot be accessed.
DLK	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 EDPRSR.SDAD is unchanged.
SLK	Memory-mapped interface only	Software lock is locked. For the external debug interface, ignore this row.
Default	-	None of the conditions apply, normal access.

The following table shows an example of external register condition codes for access to a cross trigger 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 13-5 External register condition code example

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	RO/WI	RO

13.5 Cross trigger register descriptions

This section contains the following subsections:
- *13.5.1 CTI Device Identification Register* on page 13-552.
- *13.5.2 CTI Integration Mode Control Register* on page 13-553.
- *13.5.3 CTI Peripheral Identification Registers* on page 13-553.
- *13.5.4 Component Identification Registers* on page 13-558.

13.5.1 CTI Device Identification Register

The CTIDEVID characteristics are:

Purpose Describes the CTI component to the debugger.

Usage constraints The accessibility of the CTIDEVID register by condition code is:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	RO	RO

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTIDEVID register is in the Debug power domain.

Attributes The CTIDEVID is a 32-bit register.

The following figure shows the CTIDEVID bit assignments.

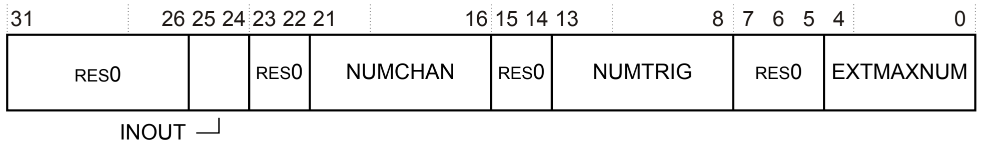

Figure 13-2 CTIDEVID bit assignments

The following table shows the CTIDEVID bit assignments.

Table 13-6 CTIDEVID bit assignments

Bits	Name	Function
[31:26]	-	Reserved, RES0.
[25:24]	INOUT	Input and output options. Indicates the presence of an input gate. The possible values are: `0b00` CTIGATE does not mask propagation of input events from external channels. `0b01` CTIGATE masks propagation of input events from external channels.
[23:22]	-	Reserved, RES0.
[21:16]	NUMCHAN	Number of channels implemented. This value is: `0b000100` Four channels implemented.
[15:14]	-	Reserved, RES0.
[13:8]	NUMTRIG	Number of triggers implemented. This value is: `0b001000` Eight triggers implemented.

Table 13-6 CTIDEVID bit assignments (continued)

Bits	Name	Function
[7:5]	-	Reserved, RES0.
[4:0]	EXTMAXNUM	Maximum number of external triggers implemented. This value is: `0b00000` No external triggers implemented.

The CTIDEVID register can be accessed through the external debug interface with offset `0xFC8`.

13.5.2 CTI Integration Mode Control Register

The CTIITCTRL characteristics are:

Purpose The CTIITCTRL register shows that the Cortex-A73 processor does not implement an integration mode.

Usage constraints The accessibility of the CTIITCTRL register by condition code is:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	RO/WI	RW

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTIITCTRL register is in the Debug power domain.

Attributes The CTIITCTRL is a 32-bit register.

The following figure shows the CTIITCTRL bit assignments.

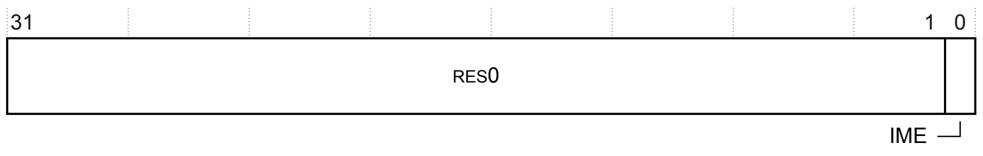

Figure 13-3 CTIITCTRL bit assignments

The following table shows the CTIITCTRL bit assignments.

Table 13-7 CTIITCTRL bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	IME	Integration mode enable. The possible value is: `0` Normal operation.

The CTIITCTRL register can be accessed through external debug interface with offset `0xF00`.

13.5.3 CTI Peripheral Identification Registers

The Peripheral Identification Registers provide standard information required for all components that conform to the Arm CoreSight architecture. There is a set of eight registers, listed in register number order in the following table.

Table 13-8 Summary of the Peripheral Identification Registers

Register	Value	Offset
Peripheral ID4	`0x04`	`0xFD0`
Peripheral ID5	`0x00`	`0xFD4`
Peripheral ID6	`0x00`	`0xFD8`
Peripheral ID7	`0x00`	`0xFDC`
Peripheral ID0	`0xAA`	`0xFE0`
Peripheral ID1	`0xB9`	`0xFE4`
Peripheral ID2	`0x0B`	`0xFE8`
Peripheral ID3	`0x00`	`0xFEC`

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

The Peripheral ID registers are:
- *Peripheral Identification Register 0* on page 13-554.
- *Peripheral Identification Register 1* on page 13-555.
- *Peripheral Identification Register 2* on page 13-555.
- *Peripheral Identification Register 3* on page 13-556.
- *Peripheral Identification Register 4* on page 13-557.
- *Peripheral Identification Register 5-7* on page 13-558.

Peripheral Identification Register 0

The CTIPIDR0 characteristics are:

Purpose Provides information to identify a CTI component.

Usage constraints The accessibility of the CTIPIDR0 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTIPIDR0 is in the Debug power domain and is optional to implement in the external register interface.

Attributes The CTIPIDR0 is a 32-bit register.

The following figure shows the CTIPIDR0 bit assignments.

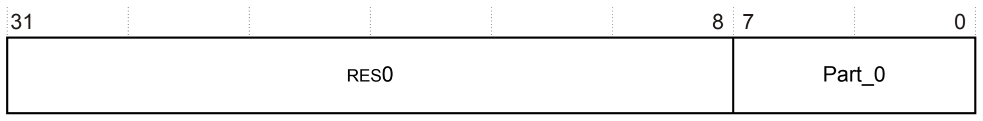

Figure 13-4 CTIPIDR0 bit assignments

The following table shows the CTIPIDR0 bit assignments.

Table 13-9 CTIPIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	Part_0	`0xAA` Least significant byte of the cross trigger part number.

The CTIPIDR0 can be accessed through the external debug interface with offset `0xFE0`.

Peripheral Identification Register 1

The CTIPIDR1 characteristics are:

Purpose Provides information to identify a CTI component.

Usage constraints The accessibility of the CTIPIDR1 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTIPIDR1 is in the Debug power domain and is optional to implement in the external register interface.

Attributes The CTIPIDR1 is a 32-bit register.

The following figure shows the CTIPIDR1 bit assignments.

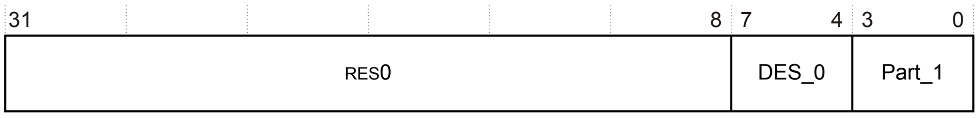

Figure 13-5 CTIPIDR1 bit assignments

The following table shows the CTIPIDR1 bit assignments.

Table 13-10 CTIPIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	DES_0	`0xB` Arm Limited. This is the least significant nibble of JEP106 ID code.
[3:0]	Part_1	`0x9` Most significant nibble of the CTI part number.

The CTIPIDR1 can be accessed through the external debug interface with offset `0xFE4`.

Peripheral Identification Register 2

The CTIPIDR2 characteristics are:

Purpose Provides information to identify a CTI component.

Usage constraints The accessibility of the CTIPIDR2 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTIPIDR2 is in the Debug power domain and is optional to implement in the external register interface.

Attributes The CTIPIDR2 is a 32-bit register.

The following figure shows the CTIPIDR2 bit assignments.

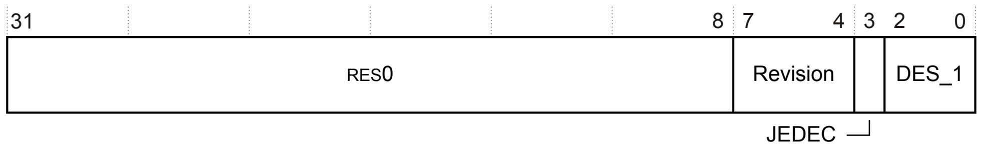

Figure 13-6 CTIPIDR2 bit assignments

The following table shows the CTIPIDR2 bit assignments.

Table 13-11 CTI PIDR2 bit assignments

Bits	Name	Function	
[31:8]	-	Reserved, RES0.	
[7:4]	Revision	`0x0`	r0p0.
[3]	JEDEC	`0b1`	RES1. Indicates a JEP106 identity code is used.
[2:0]	DES_1	`0b011`	Arm Limited. This is the most significant nibble of JEP106 ID code.

CTIPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

Peripheral Identification Register 3

The CTIPIDR3 characteristics are:

Purpose Provides information to identify a CTI component.

Usage constraints The accessibility of the CTIPIDR3 register by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTIPIDR3 register is in the Debug power domain and is optional to implement in the external register interface.

Attributes The CTIPIDR3 is a 32-bit register.

The following figure shows the CTIPIDR3 bit assignments.

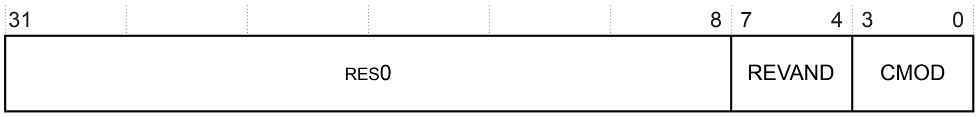

Figure 13-7 CTIPIDR3 bit assignments

The following table shows the CTIPIDR3 bit assignments.

Table 13-12 CTIPIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	REVAND	`0x0` Part minor revision.
[3:0]	CMOD	`0x0` Customer modified.

The CTIPIDR3 register can be accessed through the external debug interface with offset `0xFEC`.

Peripheral Identification Register 4

The CTIPIDR4 characteristics are:

Purpose Provides information to identify a CTI component.

Usage constraints The accessibility of the CTIPIDR4 register by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTIPIDR4 register is in the Debug power domain and is optional to implement in the external register interface.

Attributes The CTIPIDR4 is a 32-bit register.

The following figure shows the CTIPIDR4 bit assignments.

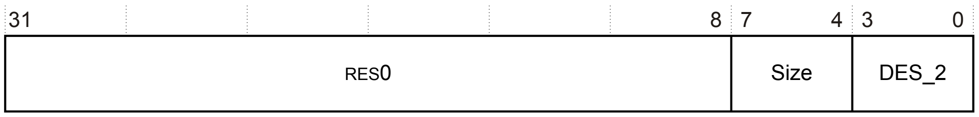

Figure 13-8 CTIPIDR4 bit assignments

The following table shows the CTIPIDR4 bit assignments.

Table 13-13 CTIPIDR4 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Size	`0x0` Size of the component. Log_2 of the number of 4KB pages from the start of the component to the end of the component ID registers.
[3:0]	DES_2	`0x4` Arm Limited. This is the least significant nibble of the JEP106 continuation code.

The CTIPIDR4 register can be accessed through the external debug interface with offset `0xFD0`.

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.

13.5.4 Component Identification Registers

There are four read-only Component Identification Registers, Component ID0 through Component ID3. The following table shows these registers.

Table 13-14 Summary of the Component Identification Registers

Register	Value	Offset
Component ID0	`0x0D`	`0xFF0`
Component ID1	`0x90`	`0xFF4`
Component ID2	`0x05`	`0xFF8`
Component ID3	`0xB1`	`0xFFC`

The Component ID registers are:
- *Component Identification Register 0* on page 13-558.
- *Component Identification Register 1* on page 13-559.
- *Component Identification Register 2* on page 13-559.
- *Component Identification Register 3* on page 13-560.

Component Identification Register 0

The CTICIDR0 characteristics are:

Purpose Provides information to identify a CTI component.

Usage constraints The accessibility of the CTICIDR0 register by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTICIDR0 register is in the Debug power domain and is optional to implement in the external register interface.

Attributes The CTICIDR0 is a 32-bit register.

The following figure shows the CTICIDR0 bit assignments.

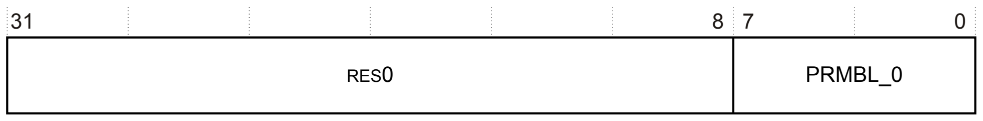

Figure 13-9 CTICIDR0 bit assignments

The following table shows the CTICIDR0 bit assignments.

Table 13-15 CTICIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_0	`0x0D` Preamble byte 0.

The CTICIDR0 register can be accessed through the external debug interface with offset `0xFF0`.

Component Identification Register 1

The CTICIDR1 characteristics are:

Purpose Provides information to identify a CTI component.

Usage constraints The accessibility of the CTICIDR1 register by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTICIDR1 register is in the Debug power domain and is optional to implement in the external register interface.

Attributes The CTICIDR1 is a 32-bit register.

The following figure shows the CTICIDR1 bit assignments.

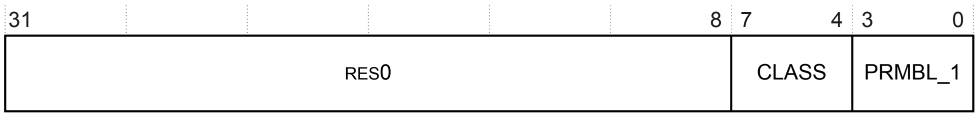

Figure 13-10 CTICIDR1 bit assignments

The following table shows the CTICIDR1 bit assignments.

Table 13-16 CTICIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	CLASS	`0x9` Debug component.
[3:0]	PRMBL_1	`0x0` Preamble byte 1.

The CTICIDR1 register can be accessed through external debug interface with offset `0xFF4`.

Component Identification Register 2

The CTICIDR2 characteristics are:

Purpose Provides information to identify a CTI component.

Usage constraints The accessibility of the CTICIDR2 register by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTICIDR2 register is in the Debug power domain and is optional to implement in the external register interface.

Attributes The CTICIDR2 is a 32-bit register.

The following figure shows the CTICIDR2 bit assignments.

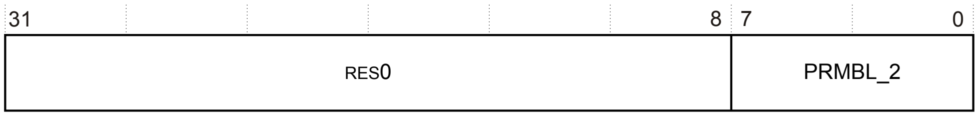

Figure 13-11 CTICIDR2 bit assignments

The following table shows the CTICIDR2 bit assignments.

Table 13-17 CTICIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_2	`0x05` Preamble byte 2.

The CTICIDR2 register can be accessed through the external debug interface with offset `0xFF8`.

Component Identification Register 3

The CTICIDR3 characteristics are:

Purpose Provides information to identify a CTI component.

Usage constraints The accessibility of the CTICIDR3 register by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

Table 13-4 External register conditions on page 13-551 describes the condition codes.

Configurations The CTICIDR3 register is in the Debug power domain and is optional to implement in the external register interface.

Attributes The CTICIDR3 is a 32-bit register.

The following figure shows the CTICIDR3 bit assignments.

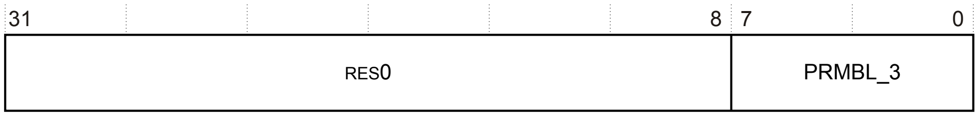

Figure 13-12 CTICIDR3 bit assignments

The following table shows the CTICIDR3 bit assignments.

Table 13-18 CTICIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_3	`0xB1` Preamble byte 3.

The CTICIDR3 register can be accessed through the external debug interface with offset `0xFFC`.

Chapter 14
Advanced SIMD and Floating-point Support

This chapter describes the Advanced SIMD and floating-point features and registers in the Cortex-A73 processor.

It contains the following sections:

14.1 About the Advanced SIMD and Floating-point support

The Cortex-A73 processor supports the Advanced SIMD and floating-point instructions in the A64, A32, and T32 instruction sets.

The Armv8 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.

14.2 Floating-point support

The Cortex-A73 floating-point implementation:

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

14.3 Accessing the feature identification registers

Software can identify the Advanced SIMD and floating-point features using the feature identification registers in the AArch64 and AArch32 Execution states.

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

```
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
```

You can access the feature identification registers in the AArch32 Execution state using the `VMRS` instructions, for example:

```
VMRS <Rt>, FPSID ; Read FPSID into Rt
VMRS <Rt>, MVFR0 ; Read MVFR0 into Rt
VMRS <Rt>, MVFR1 ; Read MFFR1 into Rt
VMRS <Rt>, MVFR2 ; Read MVFR2 into Rt
```

The following table lists the feature identification registers for Advanced SIMD and floating-point support.

Table 14-1 Advanced SIMD and floating-point feature identification registers

AArch64 name	AArch32 name	Description
-	FPSID	See *14.7.1 Floating-point System ID Register* on page 14-576.
MVFR0_EL1	MVFR0	See: • *14.5.3 Media and VFP Feature Register 0, EL1* on page 14-570 (AArch64). • *14.7.3 Media and Floating-point Feature Register 0* on page 14-579 (AArch32).
MVFR1_EL1	MVFR1	See: • *14.5.4 Media and VFP Feature Register 1, EL1* on page 14-571 (AArch64). • *14.7.4 Media and Floating-point Feature Register 1* on page 14-581 (AArch32).
MVFR2_EL1	MVFR2	See: • *14.5.5 Media and VFP Feature Register 2, EL1* on page 14-572 (AArch64). • *14.7.5 Media and Floating-point Feature Register 2* on page 14-582 (AArch32).

14.4 AArch64 register summary

The following table gives a summary of the Cortex-A73 processor Advanced SIMD and floating-point system registers in the AArch64 execution state.

Table 14-2 AArch64 Advanced SIMD and floating-point system registers

Name	Type	Reset	Description
FPCR	RW	`0x00000000`	See *14.5.1 Floating-point Control Register* on page 14-567.
FPSR	RW	`0x00000000`	See *14.5.2 Floating-point Status Register* on page 14-568.
MVFR0_EL1	RO	`0x10110222`	See *14.5.3 Media and VFP Feature Register 0, EL1* on page 14-570.
MVFR1_EL1	RO	`0x12111111`	See *14.5.4 Media and VFP Feature Register 1, EL1* on page 14-571.
MVFR2_EL1	RW	`0x00000043`	See *14.5.5 Media and VFP Feature Register 2, EL1* on page 14-572.
FPEXC32_EL2	RW	`0x00000700`	See *14.5.6 Floating-point Exception Control Register 32, EL2* on page 14-573.

14.5 AArch64 register descriptions

This section describes the AArch64 Advanced SIMD and floating-point system registers in the Cortex-A73 processor.

Table 14-2 AArch64 Advanced SIMD and floating-point system registers on page 14-566 provides cross-references to individual registers.

This section contains the following subsections:
- *14.5.1 Floating-point Control Register* on page 14-567.
- *14.5.2 Floating-point Status Register* on page 14-568.
- *14.5.3 Media and VFP Feature Register 0, EL1* on page 14-570.
- *14.5.4 Media and VFP Feature Register 1, EL1* on page 14-571.
- *14.5.5 Media and VFP Feature Register 2, EL1* on page 14-572.
- *14.5.6 Floating-point Exception Control Register 32, EL2* on page 14-573.

14.5.1 Floating-point Control Register

The FPCR characteristics are:

Purpose Controls floating-point support behavior.

Usage constraints The accessibility to the FPCR by Exception level is:

EL0	EL1(NS)	EL1(S)	EL2	EL3(SCR.NS = 1)	EL3(SCR.NS = 0)
RW	RW	RW	RW	RW	RW

Configurations The named fields in this register map to the equivalent fields in the AArch32 FPSCR. See *14.7.2 Floating-point Status and Control Register* on page 14-577.

Attributes The FPCR is a 32-bit register.

The following figure shows the FPCR bit assignments.

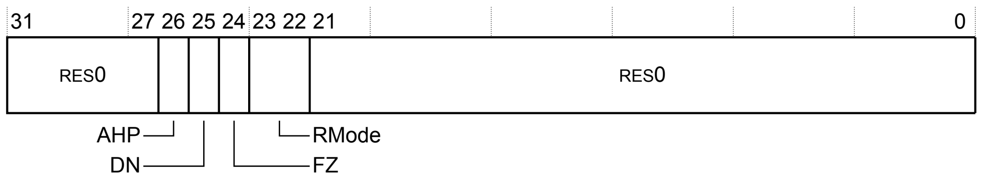

Figure 14-1 FPCR bit assignments

The following table shows the FPCR bit assignments.

Table 14-3 FPCR bit assignments

Bits	Name	Function
[31:27]	-	Reserved, RES0.
[26]	AHP	Alternative half-precision control bit. The possible values are: `0` IEEE half-precision format selected. `1` Alternative half-precision format selected.
[25]	DN	Default NaN mode control bit. The possible values are: `0` NaN operands propagate through to the output of a floating-point operation. `1` Any operation involving one or more NaNs returns the Default NaN.

Table 14-3 FPCR bit assignments (continued)

Bits	Name	Function
[24]	FZ	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. `1` Flush-to-zero mode enabled.
[23:22]	RMode	Rounding Mode control field. The encoding of this field is: `0b00` *Round to Nearest* (RN) mode. `0b01` *Round towards Plus Infinity* (RP) mode. `0b10` *Round towards Minus Infinity* (RM) mode. `0b11` *Round towards Zero* (RZ) mode.
[21:0]	-	Reserved, RES0.

To access the FPCR in AArch64 state, read or write the register with:

```
MRS <Xt>, FPCR; Read FPCR into Xt
MSR FPCR, <Xt>; Write Xt to FPCR
```

The following table shows the register access encoding.

Table 14-4 FPCR access encoding

op0	op1	CRn	CRm	op2
11	011	0100	0100	000

14.5.2 Floating-point Status Register

The FPSR characteristics are:

Purpose Provides floating-point system status information.

Usage constraints The accessibility to the FPSR by Exception level is:

EL0	EL1(NS)	EL1(S)	EL2	EL3(SCR.NS = 1)	EL3(SCR.NS = 0)
RW	RW	RW	RW	RW	RW

Configurations The FPSR is part of the floating-point functional group.

The named fields in this register map to the equivalent fields in the AArch32 FPSCR. See *14.7.2 Floating-point Status and Control Register* on page 14-577.

Attributes The FPSR is a 32-bit register.

Note

AArch64 floating-point comparisons set flags in the PSTATE register instead.

The following figure shows the FPSR bit assignments.

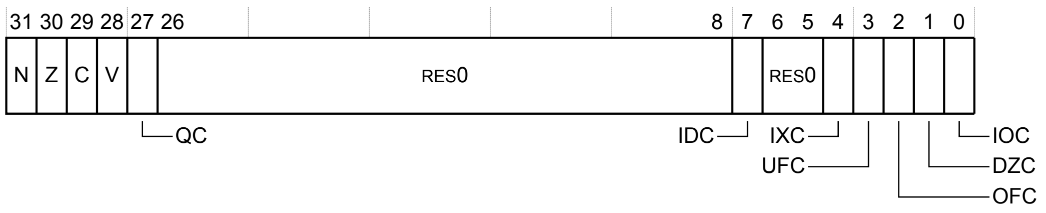

Figure 14-2 FPSR bit assignments

The following table shows the FPSR bit assignments.

Table 14-5 FPSR bit assignments

Bits	Name	Function
[31]	N	Negative condition flag for AArch32 floating-point comparison operations.
[30]	Z	Zero condition flag for AArch32 floating-point comparison operations.
[29]	C	Carry condition flag for AArch32 floating-point comparison operations.
[28]	V	Overflow condition flag for AArch32 floating-point comparison operations.
[27]	QC	Cumulative saturation bit, Advanced SIMD only. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since 0 was last written to this bit.
[26:8]	-	Reserved, RES0.
[7]	IDC	Input Denormal cumulative exception bit. This bit is set to 1 to indicate that the Input Denormal exception has occurred since a 0 was last written to this bit.
[6:5]	-	Reserved, RES0.
[4]	IXC	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.
[3]	UFC	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.
[2]	OFC	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.
[1]	DZC	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.
[0]	IOC	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.

To access the FPSR in AArch64 state, read or write the register with:

```
MRS <Xt>, FPSR; Read FPSR into Xt
MSR FPSR, <Xt>; Write Xt to FPSR
```

The following table shows the register access encoding.

Table 14-6 FPSR access encoding

op0	op1	CRn	CRm	op2
11	011	0100	0100	001

14.5.3 Media and VFP Feature Register 0, EL1

The MVFR0_EL1 characteristics are:

Purpose The MVFR0_EL1 must be interpreted with the MVFR1_EL1 and the MVFR2_EL1 to describe the features provided by Advanced SIMD and floating-point support.

Usage constraints The accessibility to the MVFR0_EL1 in AArch64 state by Exception level is:

EL0	EL1(NS)	EL1(S)	EL2	EL3(SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

Configurations The MVFR0_EL1 is architecturally mapped to AArch32 register MVFR0. See *14.7.3 Media and Floating-point Feature Register 0* on page 14-579.

Attributes The MVFR0_EL1 is a 32-bit register.

The following figure shows the MVFR0_EL1 bit assignments.

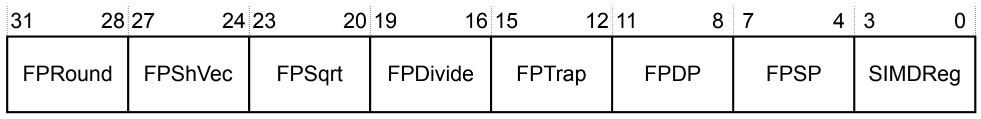

Figure 14-3 MVFR0_EL1 bit assignments

The following table shows the MVFR0_EL1 bit assignments.

Table 14-7 MVFR0_EL1 bit assignments

Bits	Name	Function
[31:28]	FPRound	Indicates the rounding modes supported by the floating-point hardware: `0x1` All rounding modes supported.
[27:24]	FPShVec	Indicates the hardware support for floating-point short vectors: `0x0` Not supported.
[23:20]	FPSqrt	Indicates the hardware support for floating-point square root operations: `0x1` Supported.
[19:16]	FPDivide	Indicates the hardware support for floating-point divide operations: `0x1` Supported.
[15:12]	FPTrap	Indicates whether the floating-point hardware implementation supports exception trapping: `0x0` Not supported.
[11:8]	FPDP	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.

Table 14-7 MVFR0_EL1 bit assignments (continued)

Bits	Name	Function
[7:4]	FPSP	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.
[3:0]	SIMDReg	Indicates support for the Advanced SIMD register bank: `0x2` 32×64-bit registers supported. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

To access the MVFR0_EL1:

```
MRS <Xt>, MVFR0_EL1 ; Read MVFR0_EL1 into Xt
```

The following table shows the register access encoding.

Table 14-8 MVFR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	000

14.5.4 Media and VFP Feature Register 1, EL1

The MVFR1_EL1 characteristics are:

Purpose The MVFR1_EL1 must be interpreted with the MVFR0_EL1 and the MVFR2_EL1 to describe the features provided by Advanced SIMD and floating-point support.

Usage constraints The accessibility to the MVFR1_EL1 in AArch64 state by Exception level is:

EL0	EL1(NS)	EL1(S)	EL2	EL3(SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

Configurations The MVFR1_EL1 is architecturally mapped to AArch32 register MVFR1. See *14.7.4 Media and Floating-point Feature Register 1* on page 14-581.

Attributes The MVFR1_EL1 is a 32-bit register.

The following figure shows the MVFR1_EL1 bit assignments.

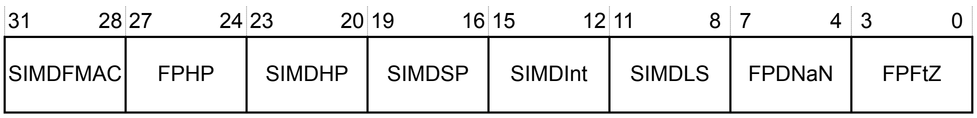

Figure 14-4 MVFR1_EL1 bit assignments

The following table shows the MVFR1_EL1 bit assignments.

Table 14-9 MVFR1_EL1 bit assignments

Bits	Name	Function
[31:28]	SIMDFMAC	Indicates whether Advanced SIMD or floating-point supports fused multiply accumulate operations: `0x1` Implemented.
[27:24]	FPHP	Indicates whether floating-point supports half-precision floating-point conversion operations: `0x2` Instructions to convert between half-precision and single-precision, and between half-precision and double-precision are implemented.
[23:20]	SIMDHP	Indicates whether Advanced SIMD supports half-precision floating-point conversion operations: `0x1` Implemented.
[19:16]	SIMDSP	Indicates whether Advanced SIMD supports single-precision floating-point operations: `0x1` Implemented.
[15:12]	SIMDInt	Indicates whether Advanced SIMD supports integer operations: `0x1` Implemented.
[11:8]	SIMDLS	Indicates whether Advanced SIMD supports load/store instructions: `0x1` Implemented.
[7:4]	FPDNaN	Indicates whether the floating-point hardware implementation supports only the Default NaN mode: `0x1` Hardware supports propagation of NaN values.
[3:0]	FPFtZ	Indicates whether the floating-point hardware implementation supports only the Flush-to-Zero mode of operation: `0x1` Hardware supports full denormalized number arithmetic.

To access the MVFR1_EL1:

```
MRS <Xt>, MVFR1_EL1 ; Read MVFR1_EL1 into Xt
```

The following table shows the register access encoding.

Table 14-10 MVFR1_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	001

14.5.5 Media and VFP Feature Register 2, EL1

The MVFR2_EL1 characteristics are:

Purpose The MVFR2_EL1 must be interpreted with the MVFR0_EL1 and the MVFR1_EL1 to describe the features provided by Advanced SIMD and floating-point support.

Usage constraints The accessibility to the MVFR2_EL1 in AArch64 state by Exception level is:

EL0	EL1(NS)	EL1(S)	EL2	EL3(SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

Configurations The MVFR2_EL1 is architecturally mapped to AArch32 register MVFR2. See *14.7.5 Media and Floating-point Feature Register 2* on page 14-582.

Attributes The MVFR2_EL1 is a 32-bit register.

The following figure shows the MVFR2_EL1 bit assignments.

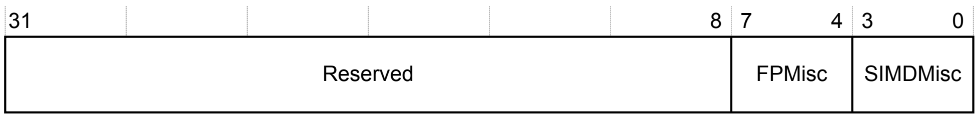

Figure 14-5 MVFR2_EL1 bit assignments

The following table shows the MVFR2_EL1 bit assignments.

Table 14-11 MVFR2_EL1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	FPMisc	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.
[3:0]	SIMDMisc	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.

To access the MVFR2_EL1:

```
MRS <Xt>, MVFR2_EL1 ; Read MVFR2_EL1 into Xt
```

The following table shows the register access encoding.

Table 14-12 MVFR2_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	010

14.5.6 Floating-point Exception Control Register 32, EL2

The FPEXC32_EL2 characteristics are:

Purpose Provides access to the AArch32 register FPEXC from AArch64 state only. Its value has no effect on execution in AArch64 state.

Usage constraints The accessibility to the FPEXC32_EL2 in AArch64 state by Exception level is:

EL0	EL1(NS)	EL1(S)	EL2	EL3(SCR.NS = 1)	EL3(SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations FPEXC32_EL2 is architecturally mapped to AArch32 register FPEXC. See *14.7.6 Floating-Point Exception Control Register* on page 14-583.

Attributes FPEXC32_EL2 is a 32-bit register.

The following figure shows the FPEXC32_EL2 bit assignments.

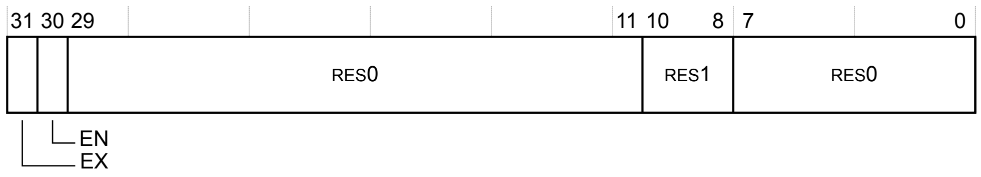

Figure 14-6 FPEXC32_EL2 bit assignments

The following table shows the FPEXC32_EL2 bit assignments.

Table 14-13 FPEXC32_EL2 bit assignments

Bits	Name	Function
[31]	EX	Exception bit. RES0 The Cortex-A73 processor implementation does not generate asynchronous floating-point exceptions.
[30]	EN	Enable bit. A global enable for Advanced SIMD and floating-point support: `0` Advanced SIMD and floating-point support is disabled. `1` Advanced SIMD and floating-point support is enabled and operates normally. This bit applies only to AArch32 execution, and only when EL1 is not AArch64.
[29:11]	-	Reserved, RES0.
[10:8]	-	Reserved, RES1.
[7:0]	-	Reserved, RES0.

To access the FPEXC_EL2:

```
MRS <Xt>, FPEXC32_EL2 ; Read FPEXC32_EL2 into Xt
MSR FPEXC32_EL2, <Xt> ; Write Xt to FPEXC32_EL2
```

See also *14.3 Accessing the feature identification registers* on page 14-565.

The following table shows the register access encoding.

Table 14-14 FPEXC_EL2 access encoding

op0	op1	CRn	CRm	op2
11	100	0101	0011	000

14.6 AArch32 register summary

The following table gives a summary of the Cortex-A73 processor Advanced SIMD and floating-point system registers in the AArch32 Execution state.

Table 14-15 AArch32 Advanced SIMD and floating-point system registers

Name	Type	Reset	Description
FPSID	RO	`0x41034091`	See *14.7.1 Floating-point System ID Register* on page 14-576.
FPSCR	RW	`0x00000000`	See *14.7.2 Floating-point Status and Control Register* on page 14-577.
MVFR0	RO	`0x10110222`	See *14.7.3 Media and Floating-point Feature Register 0* on page 14-579.
MVFR1	RO	`0x12111111`	See *14.7.4 Media and Floating-point Feature Register 1* on page 14-581.
MVFR2	RW	`0x00000043`	See *14.7.5 Media and Floating-point Feature Register 2* on page 14-582.
FPEXC	RW	`0x00000700`	See *14.7.6 Floating-Point Exception Control Register* on page 14-583.

Note

The floating-point Instruction Registers, FPINST and FPINST2, are not implemented and any attempt to access them is UNDEFINED.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information about permitted accesses to the Advanced SIMD and floating-point system registers.

14.7 AArch32 register descriptions

This section describes the AArch32 Advanced SIMD and floating-point system registers in the Cortex-A73 processor.

Table 14-15 AArch32 Advanced SIMD and floating-point system registers on page 14-575 provides cross-references to individual registers.

This section contains the following subsections:
- *14.7.1 Floating-point System ID Register* on page 14-576.
- *14.7.2 Floating-point Status and Control Register* on page 14-577.
- *14.7.3 Media and Floating-point Feature Register 0* on page 14-579.
- *14.7.4 Media and Floating-point Feature Register 1* on page 14-581.
- *14.7.5 Media and Floating-point Feature Register 2* on page 14-582.
- *14.7.6 Floating-Point Exception Control Register* on page 14-583.

14.7.1 Floating-point System ID Register

The FPSID characteristics are:

Purpose Provides top-level information about the floating-point implementation.

Usage constraints The accessibility to the FPSID by Exception level is:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Access to this register depends on the values of CPACR.{CP10,CP11}, NSACR.{CP10,CP11}, and HCPTR.{TCP10,TCP11}. For details of which field values permit access at specific Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

This register largely duplicates information held in the MIDR. Arm deprecates use of it.

Configurations There is one copy of this register that is used in both Secure and Non-secure states.

Attributes The FPSID is a 32-bit register.

The following figure shows the FPSID bit assignments.

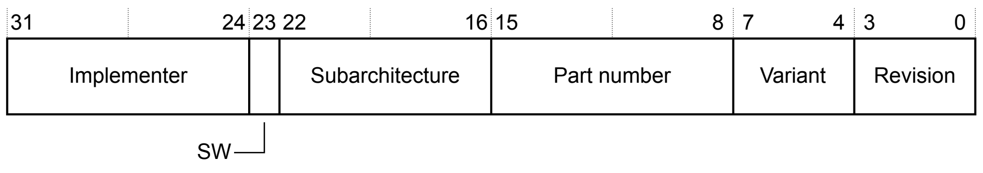

Figure 14-7 FPSID bit assignments

The following table shows the FPSID bit assignments.

Table 14-16 FPSID bit assignments

Bits	Name	Function
[31:24]	Implementer	Indicates the implementer: `0x41` Arm Limited.
[23]	SW	Software bit. This bit indicates whether a system provides only software emulation of the floating-point instructions: `0` The system includes hardware support for floating-point operations.
[22:16]	Subarchitecture	Subarchitecture version number: `0x03` VFPv3 architecture, or later, with no subarchitecture. The entire floating-point implementation is in hardware, and no software support code is required. The MVFR0, MVFR1, and MVFR2 registers indicate the VFP architecture version.
[15:8]	Part Number	Indicates the part number for the floating-point implementation: `0x40` Cortex-A73 processor.
[7:4]	Variant	Indicates the variant number: `0x9` Cortex-A73 processor.
[3:0]	Revision	Indicates the revision number for the floating-point implementation: `0x1` r1p0.

To access the FPSID register:

```
VMRS <Rt>, FPSID ; Read FPSID into Rt
```

14.7.2 Floating-point Status and Control Register

The FPSCR characteristics are:

Purpose Provides floating-point system status information and control.

Usage constraints The accessibility to the FPSCR by Exception level is:

EL0 (NS)	EL0 (S)	EL1(NS)	EL1(S)	EL2	EL3(SCR.NS = 1)	EL3(SCR.NS = 0)
Config	RW	Config	RW	Config	Config	RW

Access to this register depends on the values of CPACR.{CP10,CP11}, NSACR.{CP10,CP11}, HCPTR.{TCP10,TCP11} and FPEXC.EN. For details of which field values allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 *14.5.1 Floating-point Control Register* on page 14-567 and *14.5.2 Floating-point Status Register* on page 14-568.

Attributes The FPSCR is a 32-bit register.

The following figure shows the FPSCR bit assignments.

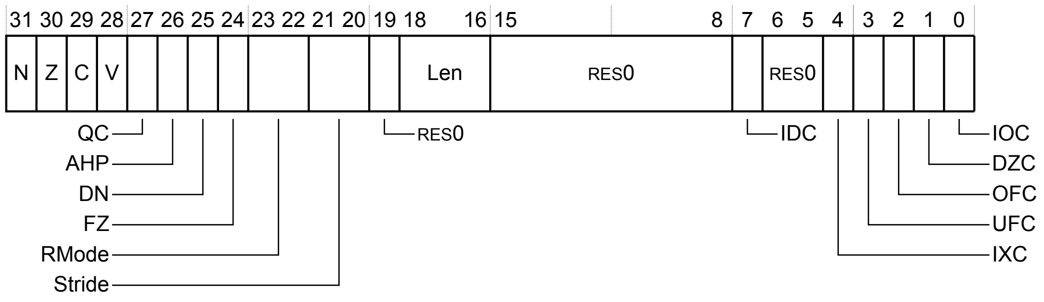

Figure 14-8 FPSCR bit assignments

The following table shows the FPSCR bit assignments.

Table 14-17 FPSCR bit assignments

Bits	Field	Function
[31]	N	Floating-point Negative condition code flag. Set to 1 if a floating-point comparison operation produces a less than result.
[30]	Z	Floating-point Zero condition code flag. Set to 1 if a floating-point comparison operation produces an equal result.
[29]	C	Floating-point Carry condition code flag. Set to 1 if a floating-point comparison operation produces an equal, greater than, or unordered result.
[28]	V	Floating-point Overflow condition code flag. Set to 1 if a floating-point comparison operation produces an unordered result.
[27]	QC	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.
[26]	AHP	Alternative Half-Precision control bit: `0` IEEE half-precision format selected. `1` Alternative half-precision format selected.
[25]	DN	Default NaN mode control bit: `0` NaN operands propagate through to the output of a floating-point operation. `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.
[24]	FZ	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. `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.

Table 14-17 FPSCR bit assignments (continued)

Bits	Field	Function
[23:22]	RMode	Rounding Mode control field: `0b00` *Round to Nearest* (RN) mode. `0b01` *Round towards Plus Infinity* (RP) mode. `0b10` *Round towards Minus Infinity* (RM) mode. `0b11` *Round towards Zero* (RZ) mode. The specified rounding mode is used by almost all scalar floating-point instructions. AArch32 Advanced SIMD arithmetic always uses the Round to Nearest setting, regardless of the value of the RMode bits.
[21:20]	Stride	RES0.
[19]	-	Reserved, RES0.
[18:16]	Len	RES0.
[15:8]	-	Reserved, RES0.
[7]	IDC	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.
[6:5]	-	Reserved, RES0.
[4]	IXC	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.
[3]	UFC	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.
[2]	OFC	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.
[1]	DZC	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.
[0]	IOC	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.

To access the FPSCR:

```
VMRS <Rt>, FPSCR ; Read FPSCR into Rt
VMSR FPSCR, <Rt> ; Write Rt to FPSCR
```

Note

The Cortex-A73 processor implementation does not support the deprecated floating-point short vector feature.

14.7.3 Media and Floating-point Feature Register 0

The MVFR0 characteristics are:

Purpose The MVFR0 must be interpreted with the MVFR1 and the MVFR2 to describe the features provided by Advanced SIMD and floating-point support.

Usage constraints The accessibility to the MVFR0 by Exception level is:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Access to this register depends on the values of CPACR.{CP10,CP11}, NSACR.{CP10,CP11}, and HCPTR.{TCP10,TCP11}. For details of which field values allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations MVFR0 is architecturally mapped to AArch64 register MVFR0_EL1. See *14.5.3 Media and VFP Feature Register 0, EL1* on page 14-570.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes The MVFR0 is a 32-bit register.

The following figure shows the MVFR0 bit assignments.

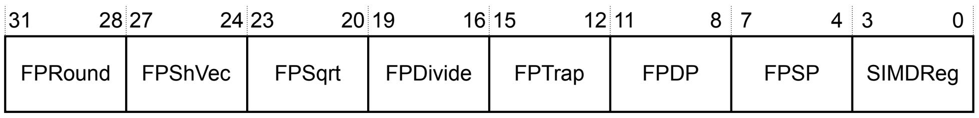

Figure 14-9 MVFR0 bit assignments

The following table shows the MVFR0 bit assignments.

Table 14-18 MVFR0 bit assignments

Bits	Name	Function
[31:28]	FPRound	Indicates the rounding modes supported by the floating-point hardware: `0x1` All rounding modes supported.
[27:24]	FPShVec	Indicates the hardware support for floating-point short vectors: `0x0` Not supported.
[23:20]	FPSqrt	Indicates the hardware support for floating-point square root operations: `0x1` Supported.
[19:16]	FPDivide	Indicates the hardware support for floating-point divide operations: `0x1` Supported.
[15:12]	FPTrap	Indicates whether the floating-point hardware implementation supports exception trapping: `0x0` Not supported.
[11:8]	FPDP	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.

Table 14-18 MVFR0 bit assignments (continued)

Bits	Name	Function
[7:4]	FPSP	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.
[3:0]	SIMDReg	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.

To access the MVFR0:

```
VMRS <Rt>, MVFR0 ; Read MVFR0 into Rt
```

14.7.4 Media and Floating-point Feature Register 1

The MVFR1 characteristics are:

Purpose The MVFR1 must be interpreted with the MVFR0 and the MVFR2 to describe the features provided by Advanced SIMD and floating-point support.

Usage constraints The accessibility to the MVFR1 by Exception level is:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Access to this register depends on the values of CPACR.{CP10,CP11}, NSACR.{CP10,CP11}, HCPTR.{TCP10,TCP11}. For details of which field values allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations MVFR1 is architecturally mapped to AArch64 register MVFR1_EL1. See *14.5.4 Media and VFP Feature Register 1, EL1* on page 14-571.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes The MVFR1 is a 32-bit register.

The following figure shows the MVFR1 bit assignments.

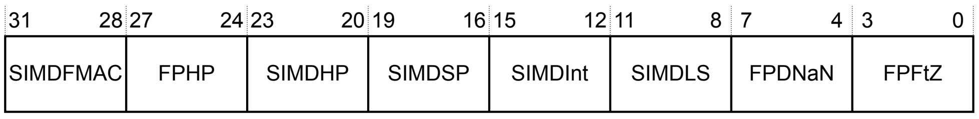

Figure 14-10 MVFR1 bit assignments

The following table shows the MVFR1 bit assignments.

Table 14-19 MVFR1 bit assignments

Bits	Name	Function
[31:28]	SIMDFMAC	Indicates whether Advanced SIMD and floating-point supports fused multiply accumulate operations: `0x1` Implemented.
[27:24]	FPHP	Indicates whether Advanced SIMD and floating-point supports half-precision floating-point conversion instructions: `0x2` Instructions to convert between half-precision and single-precision, and between half-precision and double-precision are implemented.
[23:20]	SIMDHP	Indicates whether Advanced SIMD and floating-point supports half-precision floating-point conversion operations: `0x1` Implemented.
[19:16]	SIMDSP	Indicates whether Advanced SIMD and floating-point supports single-precision floating-point operations: `0x1` Implemented.
[15:12]	SIMDInt	Indicates whether Advanced SIMD and floating-point supports integer operations: `0x1` Implemented.
[11:8]	SIMDLS	Indicates whether Advanced SIMD and floating-point supports load/store instructions: `0x1` Implemented.
[7:4]	FPDNaN	Indicates whether the floating-point hardware implementation supports only the Default NaN mode: `0x1` Hardware supports propagation of NaN values.
[3:0]	FPFtZ	Indicates whether the floating-point hardware implementation supports only the Flush-to-Zero mode of operation: `0x1` Hardware supports full denormalized number arithmetic.

To access the MVFR1:

```
VMRS <Rt>, MVFR1 ; Read MVFR1 into Rt
```

14.7.5 Media and Floating-point Feature Register 2

The MVFR2 characteristics are:

Purpose The MVFR2 must be interpreted with the MVFR0 and the MVFR1 to describe the features provided by Advanced SIMD and floating-point support.

Usage constraints The accessibility to the MVFR2 by Exception level is:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Access to this register depends on the values of CPACR.{CP10,CP11}, NSACR.{CP10,CP11}, HCPTR.{TCP10,TCP11}. For details of which field values allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations MVFR2 is architecturally mapped to AArch64 register MVFR2_EL1. See *14.5.5 Media and VFP Feature Register 2, EL1* on page 14-572.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes The MVFR2 is a 32-bit register.

The following figure shows the MVFR2 bit assignments.

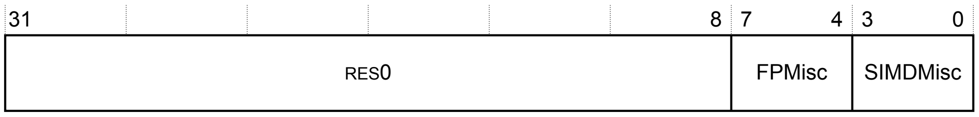

Figure 14-11 MVFR2 bit assignments

The following table shows the MVFR2 bit assignments.

Table 14-20 MVFR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	FPMisc	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.
[3:0]	SIMDMisc	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.

To access the MVFR2:

```
VMRS <Rt>, MVFR2 ; Read MVFR2 into Rt
```

14.7.6 Floating-Point Exception Control Register

The FPEXC characteristics are:

Purpose Provides a global enable for Advanced SIMD and floating-point support, and indicates how the state of this support is recorded.

Usage constraints The accessibility to the FPEXC by Exception level is:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RW	Config	Config	RW

Access to this register depends on the values of CPACR.{CP10,CP11}, NSACR.{CP10,CP11}, and HCPTR.{TCP10,TCP11}. For details of which field values allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations FPEXC is architecturally mapped to AArch64 register FPEXC32_EL2. See *14.5.6 Floating-point Exception Control Register 32, EL2* on page 14-573.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes FPEXC is a 32-bit register.

The following figure shows the FPEXC bit assignments.

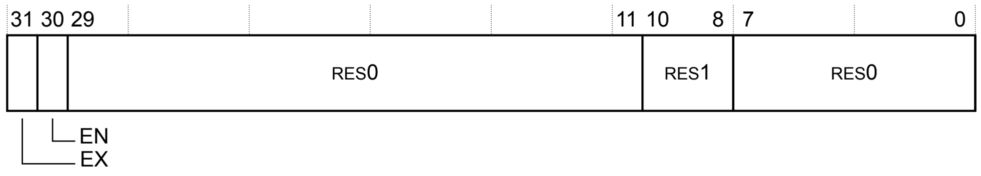

Figure 14-12 FPEXC bit assignments

The following table shows the FPEXC Register bit assignments.

Table 14-21 FPEXC bit assignments

Bits	Name	Function
[31]	EX	Exception bit. The Cortex-A73 processor implementation does not generate asynchronous floating-point exceptions, therefore this bit is RES0.
[30]	EN	Enable bit. A global enable for Advanced SIMD and floating-point support: `0` Advanced SIMD and floating-point support is disabled. `1` Advanced SIMD and floating-point support is enabled and operates normally. The EN bit is cleared at reset. This bit applies only to AArch32 execution, and only when EL1 is not AArch64.
[29:11]	-	Reserved, RES0.
[10:8]	-	Reserved, RES1.
[7:0]	-	Reserved, RES0.

To access the FPEXC register:

```
VMRS <Rt>, FPEXC ; Read FPEXC into Rt
```

See also *14.3 Accessing the feature identification registers* on page 14-565.

Appendix A
Signal Descriptions

This appendix describes the Cortex-A73 processor signals.

It contains the following sections:

A.1 About the signal descriptions

The tables in this appendix list the Cortex-A73 processor signals, along with their direction, input or output, and a high-level description.

Some of the buses include a configurable width field, <Signal>[CN:0], where CN = 0, 1, 2, or 3, to encode up to four cores. For example:

- **nIRQ[0]** represents a core 0 interrupt request.
- **nIRQ[2]** represents a core 2 interrupt request.

Some signals are specified in the form **<signal>x** where x = 0, 1, 2 or 3 to reference core 0, core 1, core 2, core 3. If a core is not present, the corresponding pin is removed. For example:

- **PMUEVENT0[24:0]** represents the core 0 PMU event bus.
- **PMUEVENT3[24:0]** represents the core 3 PMU event bus.

The number of signals changes depending on the configuration. For example, the ACP interface signals are not present if the ACP interface is not implemented.

A.2 Clock signals

The following table shows the clock signals.

Table A-1 Clock signals

Signal	Direction	Description
CLK	Input	Global clock
CORECLKEN[CN:0]	Input	Core clock enable to generate internal clocks from **CLK**
PCLKENDBG	Input	Clock enable signal to allow the external APB to be clocked at a lower frequency than **CLK**

A.3 Reset signals

The following table shows the reset and reset control signals.

Table A-2 Reset and reset control signals

Signal	Direction	Description
nCPUPORESET[CN:0]	Input	Processor powerup reset: `0` Apply reset to all processor logic. Processor logic includes Advanced SIMD and floating-point, Debug, ETM trace unit, breakpoint and watchpoint logic. `1` Do not apply reset to all processor logic.
nCORERESET[CN:0]	Input	Individual core resets excluding Debug and ETM trace unit: `0` Apply reset to processor logic. Processor logic includes Advanced SIMD and floating-point, but excludes Debug, ETM trace unit, breakpoint and watchpoint logic. `1` Do not apply reset to processor logic.
nL2RESET	Input	L2 memory system reset: `0` Apply reset to shared L2 memory system controller. `1` Do not apply reset to shared L2 memory system controller.
L2RSTDISABLE	Input	Disable automatic L2 cache invalidate at reset: `0` Hardware resets L2 cache. `1` Hardware does not reset L2 cache.
WARMRSTREQ[CN:0]	Output	Processor Warm reset request: `0` Do not apply Warm reset. `1` Apply Warm reset.

Note

- See **nPRESETDBG** in *Table A-25 APB interface signals* on page Appx-A-606.
- See **nMBISTRESET** in *Table A-32 MBIST interface signals* on page Appx-A-613.

A.4 Configuration signals

The following table shows the configuration signals.

Note

The configurations pins have to be stable during the reset.

Table A-3 Configuration signals

Signal	Direction	Description
AA64nAA32[CN:0]	Input	Register width state: This pin is sampled only during reset of the processor. `0` AArch32. `1` AArch64.
CFGEND[CN:0]	Input	Endianness configuration at reset. It sets the initial value of the EE bits in the CP15 SCTLR_EL3 and SCTR_S registers: `0` EE bit is LOW. `1` EE bit is HIGH. This pin is sampled only during reset of the processor.
CFGTE[CN:0]	Input	Enable T32 exceptions. It sets the initial value of the TE bit in the CP15 SCTLR register: `0` TE bit is LOW. `1` TE bit is HIGH. This pin is sampled only during reset of the processor.
CLUSTERIDAFF1[7:0]	Input	Value read in the Cluster ID Affinity Level 1 field, MPIDR bits[15:8], of the CP15 MPIDR register. These pins are sampled only during reset of the processor.
CLUSTERIDAFF2[7:0]	Input	Value read in the Cluster ID Affinity Level 2 field, MPIDR bits[23:16], of the CP15 MPIDR register. These pins are sampled only during reset of the processor.
CRYPTODISABLE[CN:0]	Input	Disables the Cryptographic Extension. This pin is sampled only during reset of the processor.
DBGL1RSTDISABLE	Input	Disables automatic hardware invalidation of Level 1 instruction and data caches after leaving reset. This pin is sampled only during reset of the processor.
L2RSTDISABLE	Input	Disables automatic hardware invalidation of Level 2 cache after leaving reset. This pin is sampled only during reset of the processor.

Table A-3 Configuration signals (continued)

Signal	Direction	Description
RVBARADDRx[39:2]	Input	Reset Vector Base Address for executing in 64-bit state. These pins are sampled only during reset of the processor.
VINITHI[CN:0]	Input	Location of the exception vectors at reset. It sets the initial value of the V bit in the CP15 SCTLR register: `0` Exception vectors start at address `0x00000000`. `1` Exception vectors start at address `0xFFFF0000`. This pin is sampled only during reset of the processor.

A.5 Generic Interrupt Controller signals

The following table shows the *Generic Interrupt Controller* (GIC) signals.

Table A-4 GIC signals

Signal	Direction	Description
nFIQ[CN:0]	Input	FIQ request. Active-LOW, level sensitive, asynchronous FIQ interrupt request: `0` Activate FIQ interrupt. `1` Do not activate FIQ interrupt. The processor treats the **nFIQ** input as level-sensitive. The **nFIQ** input must be asserted until the processor acknowledges the interrupt.
nIRQ[CN:0]	Input	IRQ request input lines. Active-LOW, level sensitive, asynchronous interrupt request: `0` Activate interrupt. `1` Do not activate interrupt. The processor treats the **nIRQ** input as level-sensitive. The **nIRQ** input must be asserted until the processor acknowledges the interrupt.
nSEI[CN:0]	Input	System Error Interrupt request. Active-LOW, edge sensitive: `0` Activate SEI request. `1` Do not activate SEI request. Asserting the **nSEI** input causes one of the following to occur: • Asynchronous Data Abort, if taken to AArch32. The DFSR.FS field is set to indicate an Asynchronous External Abort. • SError interrupt, if taken to AArch64. The ESR_ELx.ISS field is set, see *Table 4-102 ISS field contents for the Cortex-A73 processor* on page 4-165.
nVFIQ[CN:0]	Input	Virtual FIQ request. Active-LOW, level sensitive, asynchronous FIQ interrupt request: `0` Activate FIQ interrupt. `1` Do not activate FIQ interrupt. The processor treats the **nVFIQ** input as level-sensitive. The **nVFIQ** input must be asserted until the processor acknowledges the interrupt. If the GIC is enabled by tying the **GICCDISABLE** input pin LOW, the **nVFIQ** input pin must be tied off to HIGH. If the GIC is disabled by tying the **GICCDISABLE** input pin HIGH, the **nVFIQ** input pin can be driven by an external GIC in the SoC.
nVIRQ[CN:0]	Input	Virtual IRQ request. Active-LOW, level sensitive, asynchronous interrupt request: `0` Activate interrupt. `1` Do not activate interrupt. The processor treats the **nVIRQ** input as level-sensitive. The **nVIRQ** input must be asserted until the processor acknowledges the interrupt. If the GIC is enabled by tying the **GICCDISABLE** input pin LOW, the **nVIRQ** input pin must be tied off to HIGH. If the GIC is disabled by tying the **GICCDISABLE** input pin HIGH, the **nVIRQ** input pin can be driven by an external GIC in the SoC.

Table A-4 GIC signals (continued)

Signal	Direction	Description
nVSEI[CN:0]	Input	Virtual System Error Interrupt request. Active-LOW, edge sensitive: `0` Activate virtual SEI request. `1` Do not activate virtual SEI request. Asserting the **nVSEI** input causes one of the following to occur: • Asynchronous Data Abort, if taken to AArch32. The DFSR.FS field is set to indicate an Asynchronous External Abort. • SError interrupt, if taken to AArch64. The ESR_EL1.ISS field is set, see *Table 4-102 ISS field contents for the Cortex-A73 processor* on page 4-165.
nREI[CN:0]	Input	RAM Error Interrupt request. Active-LOW, edge sensitive: `0` Activate REI request. Reports an asynchronous RAM error in the system. `1` Do not activate REI request. Asserting the **nREI** input causes one of the following to occur: • Asynchronous Data Abort, if taken to AArch32. The DFSR.FS field is set to indicate an Asynchronous parity error on memory access. • SError interrupt, if taken to AArch64. The ESR_ELx.ISS field is set, see *Table 4-102 ISS field contents for the Cortex-A73 processor* on page 4-165.
nVCPUMNTIRQ[CN:0]	Output	Virtual CPU interface maintenance interrupt PPI output.
PERIPHBASE[39:18]	Input	Specifies the base address for the GIC registers. This value is sampled into the CP15 *Configuration Base Address Register* (CBAR) at reset.
GICCDISABLE	Input	Globally disables the GIC CPU interface logic and routes the “External” signals directly to the processor: `0` Enable the GIC CPU interface logic. `1` Disable the GIC CPU interface logic and route the legacy **nIRQ**, **nFIQ**, **nVIRQ**, and **nVFIQ** signals directly to the processor. Drive this signal HIGH when using a legacy interrupt controller such as GIC-400 which does not support GICv3 or GICv4.
ICDTVALID	Input	AXI4 Stream Protocol signal. Distributor to GIC CPU Interface messages. **TVALID** indicates that the master is driving a valid transfer.
ICDTREADY	Output	AXI4 Stream Protocol signal. Distributor to GIC CPU Interface messages. **TREADY** indicates that the slave can accept a transfer in the current cycle.
ICDTDATA[15:0]	Input	AXI4 Stream Protocol signal. Distributor to GIC CPU Interface messages. **TDATA** is the primary payload that is used to provide the data that is passing across the interface.
ICDTLAST	Input	AXI4 Stream Protocol signal. Distributor to GIC CPU Interface messages. **TLAST** indicates the boundary of a packet.
ICDTDEST[1:0]	Input	AXI4 Stream Protocol signal. Distributor to GIC CPU Interface messages. **TDEST** provides routing information for the data stream.
ICCTVALID	Output	AXI4 Stream Protocol signal. GIC CPU Interface to Distributor messages. **TVALID** indicates that the master is driving a valid transfer.
ICCTREADY	Input	AXI4 Stream Protocol signal. GIC CPU Interface to Distributor messages. **TREADY** indicates that the slave can accept a transfer in the current cycle.

Table A-4 GIC signals (continued)

Signal	Direction	Description
ICCTDATA[15:0]	Output	AXI4 Stream Protocol signal. GIC CPU Interface to Distributor messages. **TDATA** is the primary payload that is used to provide the data that is passing across the interface.
ICCTLAST	Output	AXI4 Stream Protocol signal. GIC CPU Interface to Distributor messages. **TLAST** indicates the boundary of a packet.
ICCTID[1:0]	Output	AXI4 Stream Protocol signal. GIC CPU Interface to Distributor. **TID** is the data stream identifier that indicates different streams of data.

A.6 Generic Timer signals

The following table shows the Generic Timer signals.

Table A-5 Generic Timer signals

Signal	Direction	Description
nCNTHPIRQ[CN:0]	Output	Hypervisor physical timer event.
nCNTPNSIRQ[CN:0]	Output	Non-secure physical timer event.
nCNTPSIRQ[CN:0]	Output	Secure physical timer event.
nCNTVIRQ[CN:0]	Output	Virtual physical timer event.
CNTCLKEN	Input	Counter clock enable. This clock enable must be asserted one cycle before the **CNTVALUEB** bus.
CNTVALUEB[63:0]	Input	Global system counter value in binary format.

A.7 Power management signals

The following table shows the non-Retention power management signals.

Table A-6 Non-Retention power management signals

Signal	Direction	Description
CLREXMONREQ	Input	Clearing of the external global exclusive monitor request. When this signal is asserted, it acts as a WFE wake-up event to all the cores in the MPCore device. See *CLREXMON request and acknowledge signaling* on page 2-45 for more information.
CLREXMONACK	Output	Clearing of the external global exclusive monitor acknowledge. See *CLREXMON request and acknowledge signaling* on page 2-45 for more information.
EVENTI	Input	Event input for processor wake-up from WFE state. See *Event communication using WFE or SEV* on page 2-44 for more information.
EVENTO	Output	Event output. Active when a `SEV` instruction is executed. See *Event communication using WFE or SEV* on page 2-44 for more information.
STANDBYWFI[CN:0]	Output	Indicates whether a core is in WFI low-power state: `0` Core not in WFI low-power state. `1` Core in WFI low-power state. This is the reset condition.
STANDBYWFE[CN:0]	Output	Indicates whether a core is in WFE low-power state: `0` Core not in WFE low-power state. `1` Core in WFE low-power state.
STANDBYWFIL2	Output	Indicates whether the L2 memory system is in WFI low-power state. This signal is active when the following conditions are met: • All cores are in WFI low-power state, held in reset, or **nL2RESET** is asserted LOW. • In an ACE configuration, **ACINACTM** is asserted HIGH. • If ACP has been configured, **AINACTS** is asserted HIGH. • L2 memory system is idle.
L2FLUSHREQ	Input	L2 hardware flush request.
L2FLUSHDONE	Output	L2 hardware flush complete.
SMPEN[CN:0]	Output	Indicates whether a core is taking part in coherency.
DBGNOPWRDWN[CN:0]	Output	Core no powerdown request: `0` Do not request that the core stays powered up. `1` Request that the core stays powered up.
DBGPWRUPREQ[CN:0]	Output	Core powerup request: `0` Do not request that the core is powered up. `1` Request that the core is powered up.

Table A-6 Non-Retention power management signals (continued)

Signal	Direction	Description
DBGPWRDUP[CN:0]	Input	Core powered up: `0` Core is powered down. `1` Core is powered up.
AWAKEUPM	Output	Can be used with CCI-500 to provide transfer-level clock wake-up signaling when the Cortex-A73 processor performs an AXI access. This signal is HIGH when **ARVALID**, **AWVALID**, or **DWVALID** are set, otherwise it is `0`.

The following table shows the Retention power management signals.

Table A-7 Retention power management signals

Signal	Direction	Description
CPUQACTIVE[CN:0]	Output	Indicates whether the referenced core is active
CPUQREQn[CN:0]	Input	Indicates that the power controller is ready to enter or exit retention for the referenced core
CPUQDENY[CN:0]	Output	Indicates that the referenced core denies the power controller retention request
CPUQACCEPTn[CN:0]	Output	Indicates that the referenced core accepts the power controller retention request
L2QACTIVE	Output	Indicates whether the L2 data RAMs are active
L2QREQn	Input	Indicates that the power controller is ready to enter or exit retention for the L2 data RAMs
L2QDENY	Output	Indicates that the L2 data RAMs deny the power controller retention request
L2QACCEPTn	Output	Indicates that the L2 data RAMs accept the power controller retention request

A.8 L2 error signals

The following table shows the L2 error signals.

Table A-8 L2 error signals

Signal	Direction	Description
nAXIERRIRQ	Output	Error indicator for AXI transactions with a write response error condition. See *7.7 External aborts and asynchronous errors* on page 7-374 for more information.
nECCERRIRQ	Output	Error indicator for L2 RAM double-bit ECC error.

A.9 ACE interface signals

This section describes the ACE master interface signals:

For a complete description of the ACE interface signals, see the *Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite*.

Note

- This interface exists only if the Cortex-A73 processor is configured to have the ACE interface.
- All ACE channels must be balanced with respect to **CLK** and timed relative to **ACLKENM**.

A.9.1 Clock and configuration signals

The following table shows the clock and configuration signals for the ACE interface.

Table A-9 Clock and configuration signals

Signal	Direction	Description
ACLKENM	Input	ACE Master bus clock enable. See *2.3.1 Clocks* on page 2-33 for more information.
ACINACTM	Input	Snoop interface is inactive and not participating in coherency: `0` Snoop interface is active. `1` Snoop interface is inactive.
BROADCASTCACHEMAINT	Input	Enable broadcasting of cache maintenance operations to downstream caches: `0` Cache maintenance operations are not broadcast to downstream caches. `1` Cache maintenance operations are broadcast to downstream caches. This pin is sampled only during reset of the Cortex-A73 processor. In a bigLITTLE system with Cortex-A73, this pin must be tied to 0 because Cortex-A73 does not support this feature.
BROADCASTCACHEMAINTPOU	Input	Enable broadcasting of cache maintenance by Modified Virtual Address (MVA) to the Point of Unification (PoU): `0` Cache maintenance operations by MVA to PoU are not broadcast to other clusters. `1` Cache maintenance operations by MVA to PoU are broadcast to other clusters. This pin is sampled only during reset of the Cortex-A73 processor.

Table A-9 Clock and configuration signals (continued)

Signal	Direction	Description
BROADCASTINNER	Input	Enable broadcasting of Inner Shareable transactions: `0` Inner Shareable transactions are not broadcast externally. `1` Inner Shareable transactions are broadcast externally. If **BROADCASTINNER** is tied HIGH, you must also tie **BROADCASTOUTER** HIGH. This pin is sampled only during reset of the Cortex-A73 processor.
BROADCASTOUTER	Input	Enable broadcasting of outer shareable transactions: `0` Outer Shareable transactions are not broadcast externally. `1` Outer Shareable transactions are broadcast externally. This pin is sampled only during reset of the Cortex-A73 processor.
SYSBARDISABLE	Input	Disable broadcasting of barriers onto the system bus: `0` Barriers are broadcast onto the system bus. This requires an AMBA 4 ACE. `1` Barriers are not broadcast onto the system bus. This is compatible with an AXI3 interconnect and most AMBA 4 interconnects. See *7.4.5 Barriers* on page 7-369. This pin is sampled only during reset of the Cortex-A73 processor.
RDMEMATTR[7:0]	Output	Read request memory attributes.
WRMEMATTR[7:0]	Output	Write request memory attributes.

A.9.2 Write address channel signals

The following table shows the write address channel signals for the ACE interface.

Table A-10 Write address channel signals

Signal	Direction	Description
AWADDRM[39:0]	Output	Write address.
AWBARM[1:0]	Output	Write barrier type.
AWBURSTM[1:0]	Output	Write burst type.
AWCACHEM[3:0]	Output	Write cache type.
AWDOMAINM[1:0]	Output	Write shareability domain type.
AWIDM[7:0]	Output	Write address ID.
AWLENM[7:0]	Output	Write burst length.
AWLOCKM	Output	Write lock type.
AWPROTM[2:0]	Output	Write protection type.
AWREADYM	Input	Write address ready.

Table A-10 Write address channel signals (continued)

Signal	Direction	Description
AWSIZEM[2:0]	Output	Write burst size.
AWSNOOPM[2:0]	Output	Write snoop request type.
AWVALIDM	Output	Write address valid.

A.9.3 Write data channel signals

The following table shows the write data channel signals for the ACE interface.

Table A-11 Write data channel signals

Signal	Direction	Description
WDATAM[127:0]	Output	Write data
WIDM[7:0]	Output	Write data ID
WLASTM	Output	Write data last transfer indication
WREADYM	Input	Write data ready
WSTRBM[15:0]	Output	Write byte-lane strobes
WVALIDM	Output	Write data valid

A.9.4 Write data response channel signals

The following table shows the write data response channel signals for the ACE master interface.

Table A-12 Write data response channel signals

Signal	Direction	Description
BIDM[7:0]	Input	Write response ID
BREADYM	Output	Write response ready
BRESPM[1:0]	Input	Write response
BVALIDM	Input	Write response valid

A.9.5 Read address channel signals

The following table shows the read address channel signals for the ACE master interface.

Table A-13 Read address channel signals

Signal	Direction	Description
ARADDRM[43:0]	Output	Read address. The top 4 bits communicate only the ACE virtual address for DVM messages. The top 4 bits are Read-as-Zero if a DVM message is not being broadcast.
ARBARM[1:0]	Output	Read barrier type.
ARBURSTM[1:0]	Output	Read burst type.

Table A-13 Read address channel signals (continued)

Signal	Direction	Description
ARCACHEM[3:0]	Output	Read cache type.
ARDOMAINM[1:0]	Output	Read shareability domain type.
ARIDM[7:0]	Output	Read address ID.
ARLENM[7:0]	Output	Read burst length.
ARLOCKM	Output	Read lock type.
ARPROTM[2:0]	Output	Read protection type.
ARREADYM	Input	Read address ready.
ARSIZEM[2:0]	Output	Read burst size.
ARSNOOPM[3:0]	Output	Read snoop request type.
ARVALIDM	Output	Read address valid.

A.9.6 Read data channel signals

The following table shows the read data channel signals for the ACE master interface.

Table A-14 Read data channel signals

Signal	Direction	Description
RDATAM[127:0]	Input	Read data
RIDM[7:0]	Input	Read data ID
RLASTM	Input	Read data last transfer indication
RREADYM	Output	Read data ready
RRESPM[3:0]	Input	Read data response
RVALIDM	Input	Read data valid

A.9.7 Coherency address channel signals

The following table shows the coherency address channel signals for the ACE master interface.

Table A-15 Coherency address channel signals

Signal	Direction	Description
ACADDRM[43:0]	Input	Snoop address. The top 4 bits communicate only the ACE virtual address for DVM messages.
ACPROTM[2:0]	Input	Snoop protection type.
ACREADYM	Output	Master ready to accept snoop address.

Table A-15 Coherency address channel signals (continued)

Signal	Direction	Description
ACSNOOPM[3:0]	Input	Snoop request type.
ACVALIDM	Input	Snoop address valid.

A.9.8 Coherency response channel signals

The following table shows the coherency response channel signals for the ACE master interface.

Table A-16 Coherency response channel signals

Signal	Direction	Description
CRREADYM	Input	Slave ready to accept snoop response
CRVALIDM	Output	Snoop response valid
CRRESPM[4:0]	Output	Snoop response

A.9.9 Coherency data channel handshake signals

The following table shows the coherency data channel handshake signals for the ACE master interface.

Table A-17 Coherency data channel handshake signals

Signal	Direction	Description
CDDATAM[127:0]	Output	Snoop data
CDLASTM	Output	Snoop data last transform
CDREADYM	Input	Slave ready to accept snoop data
CDVALIDM	Output	Snoop data valid

A.9.10 Read and write acknowledge signals

The following table shows the read/write acknowledge signals for the ACE master interface.

Table A-18 Read and write acknowledge signals

Signal	Direction	Description
RACKM	Output	Read acknowledge
WACKM	Output	Write acknowledge

A.10 ACP interface signals

The following ACP interface signals are described:

- *A.10.1 Clock and configuration signals* on page Appx-A-603.
- *A.10.2 Write address channel signals* on page Appx-A-603.
- *A.10.3 Write data channel signals* on page Appx-A-604.
- *A.10.4 Write response channel signals* on page Appx-A-604.
- *A.10.5 Read address channel signals* on page Appx-A-604.
- *A.10.6 Read data channel signals* on page Appx-A-605.

Note

- This interface exists only if the Cortex-A73 processor is configured to have the ACP interface.
- All ACP channels must be balanced with respect to **CLK** and timed relative to **ACLKENS**.

A.10.1 Clock and configuration signals

The following table shows the clock and configuration signals for the ACP interface.

Table A-19 Clock and configuration signals

Signal	Direction	Description
ACLKENS	Input	AXI slave bus clock enable.
AINACTS	Input	ACP master is inactive and is not participating in coherency. There must be no outstanding transactions when the master asserts this signal, and while it is asserted the master must not send any new transactions: `0` ACP Master is active. `1` ACP Master is inactive. **Note** This signal must be asserted before the processor enters the low power L2 WFI state.

A.10.2 Write address channel signals

The following table shows the write address channel signals for the ACP interface.

Table A-20 Write address channel signals

Signal	Direction	Description
AWREADYS	Output	Write address ready
AWVALIDS	Input	Write address valid
AWIDS[4:0]	Input	Write address ID
AWADDRS[39:0]	Input	Write address
AWLENS[7:0]	Input	Write burst length
AWSIZES[2:0]	Input	Write burst size
AWLOCKS	Input	Write lock type
AWBURSTS[1:0]	Input	Write burst type

Table A-20 Write address channel signals (continued)

Signal	Direction	Description
AWCACHES[3:0]	Input	Write cache type
AWPROTS[2:0]	Input	Write protection type

A.10.3 Write data channel signals

The following table shows the write data channel signals for the ACP interface.

Table A-21 Write data channel signals

Signal	Direction	Description
WREADYS	Output	Write data ready
WVALIDS	Input	Write data valid
WDATAS[127:0]	Input	Write data
WSTRBS[15:0]	Input	Write byte-lane strobes
WLASTS	Input	Write data last transfer indication

A.10.4 Write response channel signals

The following table shows the write response channel signals for the ACP interface.

Table A-22 Write response channel signals

Signal	Direction	Description
BREADYS	Input	Write response ready
BVALIDS	Output	Write response valid
BIDS[4:0]	Output	Write response ID
BRESPS[1:0]	Output	Write response

A.10.5 Read address channel signals

The following table shows the read address channel signals for the ACP interface.

Table A-23 Read address channel signals

Signal	Direction	Description
ARREADYS	Output	Read address ready
ARVALIDS	Input	Read address valid
ARIDS[4:0]	Input	Read address ID
ARADDRS[39:0]	Input	Read address
ARLENS[7:0]	Input	Read burst length
ARSIZES[2:0]	Input	Read burst size
ARBURSTS[1:0]	Input	Read burst type
ARLOCKS	Input	Read lock type

Table A-23 Read address channel signals (continued)

Signal	Direction	Description
ARCACHES[3:0]	Input	Read cache type
ARPROTS[2:0]	Input	Read protection type

A.10.6 Read data channel signals

The following table shows the read data channel signals for the ACP interface.

Table A-24 Read data channel signals

Signal	Direction	Description
RREADYS	Input	Read data ready
RVALIDS	Output	Read data valid
RIDS[4:0]	Output	Read data ID
RDATAS[127:0]	Output	Read data
RRESPS[1:0]	Output	Read response
RLASTS	Output	Read data last transfer indication

A.11 External debug interface

The following external debug interface signals are described:

- *A.11.1 APB interface signals* on page Appx-A-606.
- *A.11.2 Miscellaneous debug signals* on page Appx-A-606.

A.11.1 APB interface signals

The following table shows the APB interface signals.

Note

You must balance all APB interface signals with respect to **CLK** and time them relative to **PCLKENDBG**.

Table A-25 APB interface signals

Signal	Direction	Description
nPRESETDBG	Input	APB reset, active-LOW: `0` Apply reset to APB interface. `1` Do not apply reset to APB interface.
PADDRDBG[21:2]	Input	APB address bus.
PADDRDBG31	Input	APB address bus bit[31]: `0` Not an external debugger access. `1` External debugger access.
PCLKENDBG	Input	APB clock enable.
PENABLEDBG	Input	Indicates the second and subsequent cycles of an APB transfer.
PRDATADBG[31:0]	Output	APB read data.
PREADYDBG	Output	APB slave ready. An APB slave can deassert **PREADYDBG** to extend a transfer by inserting wait states.
PSELDBG	Input	Debug bus access.
PSLVERRDBG	Output	APB slave transfer error: `0` No transfer error. `1` Transfer error.
PWDATADBG[31:0]	Input	APB write data.
PWRITEDBG	Input	APB read or write signal: `0` Reads from APB. `1` Writes to APB.

A.11.2 Miscellaneous debug signals

The following table shows the miscellaneous Debug signals.

Table A-26 Miscellaneous Debug signals

Signal	Direction	Description
DBGROMADDR[39:12]	Input	Debug ROM base address. Specifies bits[39:12] of the ROM table physical address. If the address cannot be determined, tie this signal LOW. This pin is sampled only during reset of the processor.
DBGROMADDRV	Input	Debug ROM base address valid. If the debug ROM address cannot be determined, tie this signal LOW. This pin is sampled only during reset of the processor.
DBGACK[CN:0]	Output	Debug acknowledge: `0` External debug request not acknowledged. `1` External debug request acknowledged.
nCOMMIRQ[CN:0]	Output	Communications channel receive or transmit interrupt request: `0` Request interrupt. `1` No interrupt request.
COMMRX[CN:0]	Output	Communications channel receive. Receive portion of Data Transfer Register full flag: `0` Empty. `1` Full.
COMMTX[CN:0]	Output	Communication transmit channel. Transmit portion of Data Transfer Register empty flag: `0` Full. `1` Empty.
EDBGRQ[CN:0]	Input	External debug request: `0` No external debug request. `1` External debug request. The processor treats the **EDBGRQ** input as level-sensitive. The **EDBGRQ** input must be asserted until the processor asserts **DBGACK.**
DBGEN[CN:0]	Input	Invasive debug enable: `0` Not enabled. `1` Enabled.
NIDEN[CN:0]	Input	Non-invasive debug enable: `0` Not enabled. `1` Enabled.
SPIDEN[CN:0]	Input	Secure privileged invasive debug enable: `0` Not enabled. `1` Enabled.

Table A-26 Miscellaneous Debug signals (continued)

Signal	Direction	Description
SPNIDEN[CN:0]	Input	Secure privileged non-invasive debug enable: `0` Not enabled. `1` Enabled.
DBGRSTREQ[CN:0]	Output	Warm reset request.
DBGNOPWRDWN[CN:0]	Output	Core no powerdown request: `0` Do not request that the core stays powered up. `1` Request that the core stays powered up.
DBGPWRUPREQ[CN:0]	Output	Core powerup request: `0` Do not request that the core is powered up. `1` Request that the core is powered up.
DBGPWRDUP[CN:0]	Input	Core powered up: `0` Core is powered down. `1` Core is powered up.
DBGL1RSTDISABLE	Input	Disable L1 instruction and data cache automatic invalidate on reset functionality: `0` Enable automatic invalidation of L1 instruction and data caches on reset. `1` Disable automatic invalidation of L1 instruction and data caches on reset. This pin is sampled only during reset of the processor.

A.12 ATB interface signals

The following table shows the ATB interface signals.

Note

- You must balance all ATB interface signals with respect to **CLK** and time them relative to **ATCLKEN**.

Table A-27 ATB interface signals

Signal	Direction	Description
ATCLKEN	Input	ATB clock enable
ATREADYMx	Input	ATB device ready
AFVALIDMx	Input	FIFO flush request
ATDATAMx[31:0]	Output	Data
ATVALIDMx	Output	Data valid
ATBYTESMx[1:0]	Output	Data size
AFREADYMx	Output	FIFO flush finished
ATIDMx[6:0]	Output	Trace source ID

A.13 Miscellaneous ETM trace unit signals

The following table shows the miscellaneous ETM trace unit signals.

Table A-28 Miscellaneous ETM trace unit signals

Signal	Direction	Description
SYNCREQx	Input	Synchronization request from trace sink
TSVALUEB[63:0]	Input	Timestamp in binary encoding

A.14 CTI interface signals

The following table shows the CTI interface signals.

Table A-29 CTI interface signals

Signal	Direction	Description
CTICHIN[3:0]	Input	Channel In
CTICHOUTACK[3:0]	Input	Channel Out acknowledge
CTICHOUT[3:0]	Output	Channel Out
CTICHINACK[3:0]	Output	Channel In acknowledge
CISBYPASS	Input	Channel interface sync bypass
CIHSBYPASS[3:0]	Input	Channel interface handshake bypass
CTIIRQ[CN:0]	Output	CTI interrupt (active-HIGH)
CTIIRQACK[CN:0]	Input	CTI interrupt acknowledge

A.15 PMU interface signals

The following table shows the PMU interface signals.

Table A-30 PMU interface signals

Signal	Direction	Description
PMUEVENTx[37:0]	Output	PMU event bus
nPMUIRQ[CN:0]	Output	PMU interrupt request

A.16 DFT and MBIST interface signals

The following interfaces are described:

- *A.16.1 DFT interface* on page Appx-A-613.
- *A.16.2 MBIST interface* on page Appx-A-613.

A.16.1 DFT interface

The following table shows the DFT interface signals.

Table A-31 DFT interface signals

Signal	Direction	Description
DFTCGEN	Input	Scan shift enable, forces on the clock grids during scan shift.
DFTRAMHOLD	Input	Disable the RAM chip select during scan testing.
DFTRSTDISABLE[1:0]	Input	Disable internal synchronized reset during scan shift.
DFTMCPHOLD	Input	Disable multicycle paths on RAM interfaces.
DFTCLKBYPASS	Input	Bypass internal stretched clock for test purposes if **L2HALFDATARAMCLK** configuration is used.

A.16.2 MBIST interface

The following table describes the signals for the MBIST interface.

Table A-32 MBIST interface signals

Signal	Direction	Description
nMBISTRESET	Input	MBIST reset.
MBISTREQx[3:0]	Input	MBIST L1 test request.

Note

Apart from **nMBISTRESET** and **MBISTREQx**, all MBIST signals are internal to `artemis_nocpu.v` module and connected to `artemis_topmbmux.v` module. The MBIST controller is expected to be directly connected to `artemis_topmbmux.v` module.

Appendix B
Revisions

This appendix describes the technical changes between released issues of this document.

It contains the following section:

B.1 Revisions

This section describes the technical changes between released issues of this document.

Table B-1 Issue 0000-01

Change	Location	Affects
First release	-	-

Table B-2 Differences between Issue 0000-01 and Issue 0001-02

Change	Location	Affects
Added note about integer clock ratio.	*CORECLKEN[CN:0] clock enable signal* on page 2-36	All Versions
Updated description of CLIDR_EL1 LoUIS bits.	*4.3.31 Cache Level ID Register* on page 4-115	All Versions
Updated CCSIDR encodings.	*4.5.22 Cache Size ID Register* on page 4-248	All Versions
Added note about aliases in the L1 data memory subsystems.	*6.5 L1 data memory system* on page 6-347	All Versions
Updated ACE transactions.	*6.5.6 ACE transactions* on page 6-349	All Versions
Updated some reset values in the debug registers.	• *10.3 AArch64 debug register summary* on page 10-396 • *10.3 AArch64 debug register summary* on page 10-396	All Versions

Table B-3 Differences between Issue 0001-02 and Issue 0002-03

Change	Location	Affects
Updated signals for exit from L2 WFI standby state.	*L2 Wait for Interrupt* on page 2-45	All Versions
Added note about cache disable.	*4.3.72 L2 Control Register* on page 4-174	All Versions
Updated description of attribute fields.	*4.3.75 Physical Address Register, EL1* on page 4-179	All Versions
Added note about access from an external debugger.	*6.7 Direct access to internal memory* on page 6-351	All Versions
Updated read issuing capability.	*7.4.1 Memory interface attributes* on page 7-367	All Versions
Added note.	*7.5 Additional memory attributes* on page 7-370	All Versions

Table B-4 Differences between Issue 0002-03 and Issue 0002-04

Change	Location	Affects
Updated the Input synchronization list.	*2.3.2 Input synchronization* on page 2-36	All Versions
Updated the Individual core shutdown mode section.	*Individual core shutdown mode* on page 2-50	All Versions
Updated the Cluster shutdown mode without system driven L2 flush section.	*Cluster shutdown mode without system driven L2 flush* on page 2-51	All Versions
Updated the LoUIS field of the CLIDR_EL1 register	*4.3.31 Cache Level ID Register* on page 4-115	All Versions
Updated the Flush index increment field of the L2CTLR_EL1 register.	*4.3.72 L2 Control Register* on page 4-174	All Versions
Updated the Cache behavior description	*6.2 Cache behavior* on page 6-343	All Versions

Table B-4 Differences between Issue 0002-03 and Issue 0002-04 (continued)

Change	Location	Affects
Updated bit-field of Data Register 1.	*6.7.1 Data cache tag and data encoding* on page 6-352	All Versions
Updated note.	*7.3.1 Snoop and maintenance requests* on page 7-366	All Versions

Table B-5 Differences between Issue 0002-04 and 0002-05

Change	Location	Affects
Updated Events table	*11.9 Events* on page 11-475	All Versions
Changed number of CLK cycles	*Event communication using WFE or SEV* on page 2-44	All Versions

Table B-6 Differences between Issue 0002-05 and 0100-06

Change	Location	Affects
Updated TLB RAM description for direct RAM access decoding.	*6.7.3 TLB data encoding* on page 6-353	All Versions
Updated TLB RAM description for direct RAM access decoding.	*6.7.3 TLB data encoding* on page 6-353	All Versions
Updated **DBGL1RSTDISABLE** signal to indicate L1 instruction cache reset is disabled when set HIGH.	*A.11.2 Miscellaneous debug signals* on page Appx-A-606, *A.4 Configuration signals* on page Appx-A-589, *10.10.3 DBGL1RSTDISABLE debug signal* on page 10-430, *6.2 Cache behavior* on page 6-343, *2.3.3 Resets* on page 2-37.	All Versions
Updated L2 Memory Error Syndrome Register.	*4.3.86 L2 Memory Error Syndrome Register* on page 4-191	All Versions
Added requirements for warm reset.	*Warm reset* on page 2-40	All Versions
Added a description of WARMRSTREQ and DBGRSTREQ, a mechanism to configure whether a processor uses AArch32 or AArch64 at EL3 as a result of a Warm reset.	*WARMRSTREQ and DBGRSTREQ* on page 2-40	All Versions
Changed PMXVTYPER2_EL0 to PMEVTYPER2_EL0.	*11.3 AArch64 PMU register summary* on page 11-446	All Versions
Updated Translation Control Registers EL1, EL2 and EL3 to show support for 16KB translation granule.	*4.3.56 Translation Control Register, EL1* on page 4-153, *4.3.57 Translation Control Register, EL2* on page 4-156, *4.3.61 Translation Control Register, EL3* on page 4-162	All Versions
Added clarification to L2 hardware cache flush regarding pending evictions due to HW flush request when L2FLUSHDONE is raised.	*L2 hardware cache flush* on page 2-45	All Versions
Updated description of bits [1:0] in TLB regular format encoding table and TLB walk format encoding table.	*TLB regular format* on page 6-354 and *TLB walk format* on page 6-357	All Versions
Updated Core dynamic retention.	*Core dynamic retention* on page 2-46	All Versions
Added AXI ID core source encoding.	*7.4.8 AXI ID core source encoding* on page 7-369	All Versions

Table B-6 Differences between Issue 0002-05 and 0100-06 (continued)

Change	Location	Affects
New CSV3 field in ID_AA64PFR0_EL1 and ID_PFR2(_EL1) registers, indicating that data loaded under speculation with a permission or domain fault, if used as an address in a speculative load, cannot cause cache allocation.	*4.3.20 AArch64 Processor Feature Register 0* on page 4-107, *4.5.8 Processor Feature Register 2* on page 4-229	r1p0
Inter-exception level isolation of branch predictor structures so that an exception level cannot train branch prediction for a different exception level to reliably hit in these trained prediction entries. New CSV2 field in ID_AA64PFR0_EL1 and ID_PFR0(_EL1) registers indicates the feature is present in this release.	*4.3.20 AArch64 Processor Feature Register 0* on page 4-107, *4.5.6 Processor Feature Register 0* on page 4-226	r1p0