Arm® Cortex®-R52 Processor

Revision: r1p3

Technical Reference Manual

arm

100026_0103_00_en

Arm® Cortex®-R52 Processor

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-00	12 August 2016	Confidential	First release for r0p0.
0100-00	30 March 2017	Non-Confidential	First release for r1p0.
0101-00	13 September 2017	Non-Confidential	First release for r1p1.
0101-01	16 February 2018	Non-Confidential	Second release for r1p1.
0102-00	18 January 2019	Non-Confidential	First release for r1p2.
0103-00	28 July 2020	Non-Confidential	First release for r1p3.

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

developer.arm.com

Contents

Arm® Cortex®-R52 Processor Technical Reference Manual

Preface

This preface introduces the *Arm® Cortex®-R52 Processor Technical Reference Manual*.

It contains the following:

About this book

This book is for the Cortex-R52 processor.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, 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®-R52 processor.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction

This chapter gives an introduction to the Cortex-R52 processor.

Chapter 2 Programmers Model

This chapter describes the programmers model.

Chapter 3 System Control

This chapter describes the system registers and their structure and operation.

Chapter 4 Clocking and Resets

This chapter describes clocks and resets used within the processor.

Chapter 5 Power Management

This chapter describes the power management facilities provided by the Cortex-R52 processor.

Chapter 6 Initialization

This chapter describes considerations for initializing the Cortex-R52 processor.

Chapter 7 Memory System

This chapter describes the memory system.

Chapter 8 Memory Protection Unit

This chapter describes the MPU.

Chapter 9 Generic Interrupt Controller

This chapter describes the Cortex-R52 processor implementation of the *Generic Interrupt Controller* (GIC).

Chapter 10 Generic Timer

This chapter describes the Cortex-R52 processor implementation of the Arm Generic Timer.

Chapter 11 Debug

This chapter describes the Cortex-R52 processor debug registers and shows examples of how to use them.

Chapter 12 Performance Monitor Unit

This section describes the *Performance Monitor Unit* (PMU) features and the registers that it uses.

Chapter 13 Cross Trigger

This chapter describes the cross trigger logic for the Cortex-R52 processor.

Chapter 14 Embedded Trace Macrocell

This chapter describes the *Embedded Trace Macrocell* (ETM) for the Cortex-R52 processor.

Chapter 15 Advanced SIMD and floating-point support

This chapter describes the Advanced SIMD and floating-point features and registers the processor uses.

Appendix A Signal Descriptions

This appendix describes the Cortex-R52 processor signals.

Appendix B Cycle Timings and Interlock Behavior

This appendix describes the cycle timing and interlock behavior of instructions on the Cortex-R52 processor.

Appendix C Processor UNPREDICTABLE Behaviors

This appendix describes specific Cortex-R52 processor UNPREDICTABLE behaviors of particular interest. These UNPREDICTABLE behaviors differ from the Arm standard behavior.

Appendix D PMC-R52

This appendix describes the *Programmable MBIST Controller* (PMC) for the Cortex-R52 processor.

Appendix E Revisions

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

Glossary

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

See the *Arm® Glossary* for more information.

Typographic conventions

italic

Introduces special terminology, denotes cross-references, and citations.

bold

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

`monospace`

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

`monospace`

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

`monospace italic`

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

`monospace bold`

Denotes language keywords when used outside example code.

`<and>`

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

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

SMALL CAPITALS

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

Timing diagrams

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

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

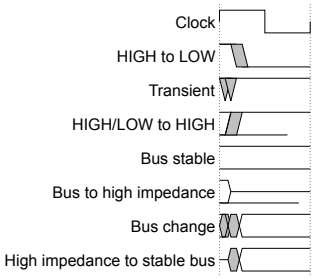

Figure 1 Key to timing diagram conventions

Signals

The signal conventions are:

Signal level

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

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

Lowercase n

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

Additional reading

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

Arm publications

- *AMBA® 3 APB Protocol v1.0 Specification* (IHI 0024B).
- *AMBA® 4 ATB Protocol Specification ATBv1.0 and ATBv1.1* (IHI 0032).
- *AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™, ACE and ACE-Lite™* (IHI 0022).
- *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* (DDI 0568).
- *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* (DDI 0487).
- *Arm® CoreSight™ Architecture Specification v2.0* (IHI 0029D).
- *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* (IHI 0064).
- *Arm® Generic Interrupt Controller Architecture Specification GIC architecture version 3.0 and version 4.0* (IHI 0069).
- *Low Power Interface Specification Arm® Q-Channel and P-Channel Interfaces* (IHI 0068).

The following confidential books are only available to licensees:

- *Arm® Cortex®-R52 Processor Configuration and Sign-off Guide* (100027).
- *Arm® Cortex®-R52 Processor Integration Manual* (100028).
- *Armv8 AArch32 UNPREDICTABLE behaviours* (PRD03-GENC-010544).

Other publications

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

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-R52 Processor Technical Reference Manual*.
- The number 100026_0103_00_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 gives an introduction to the Cortex-R52 processor.

It contains the following sections:

1.1 About the Cortex®-R52 processor

The Cortex-R52 processor is a mid-performance, in-order, superscalar processor primarily for use in automotive and industrial applications. It is also suited to a wide variety of other embedded applications such as communication and storage devices.

The Cortex-R52 processor has one to four cores, each implementing a single Armv8-R compliant *processing element* (PE). In the context of Cortex-R52, the PE and core are conceptually the same.

Multiple *Protected Memory System Architecture* (PMSA) contexts can execute on the same core using virtualization technology to contain them. The processor enables real-time performance of different contexts to be contained, which prevents one context from impacting the response time and determinism of a more critical context. The processor can have redundant copies of the logic and comparator instances for *Dual-Core Lock-Step* (DCLS) operation.

The following figure shows an example Cortex-R52 processor system.

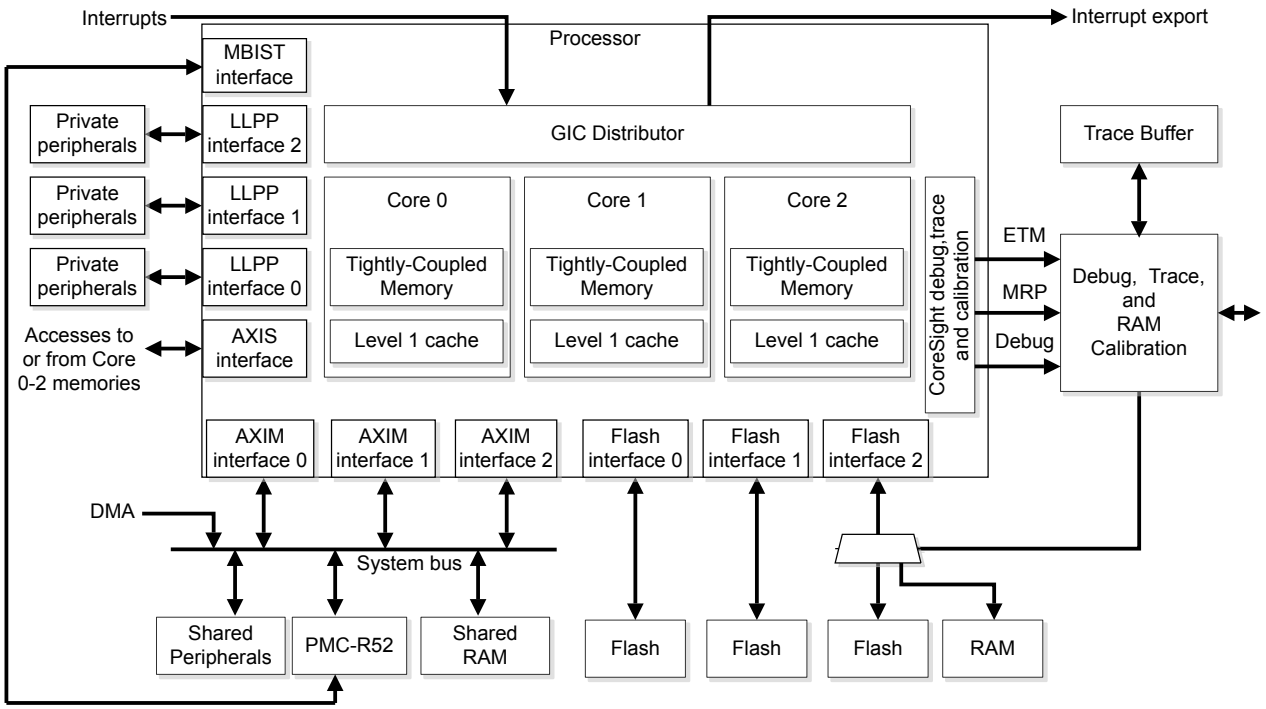

Figure 1-1 Example Cortex-R52 processor system

This section contains the following subsections:

- *1.1.1 Features* on page 1-14.
- *1.1.2 Interfaces* on page 1-15.
- *1.1.3 Configuration options* on page 1-15.

1.1.1 Features

The main features of the Cortex-R52 processor include:

- Up to four cores, each with an eight-stage in-order, superscalar pipeline with branch prediction.
- *Error Correcting Code* (ECC), *Single Error Correct Double Error Detect* (SECDED), protection for all of the instantiated cache tag and data, and *Tightly-Coupled Memory* (TCM) RAMs, and the connected flash memories.
- Optional signal integrity protection on the data, address, control and response payloads and handshake signals, and optional interconnect protection for the *main AXI Master* (AXIM) interface, *AXI Slave* (AXIS) interface, *Low-latency Peripheral Port* (LLPP) interface, and the Flash interface.
- Error reporting interface.

- Power management.
- Armv8-R debug providing hardware breakpoints and watchpoints, self-hosted, and external debug. It also enables communications between debug target and host.
- *Embedded Trace Macrocell* (ETM) for instruction and data trace.
- *Memory Reconstruction Port* (MRP) for use in emulation and calibration.
- *Performance Monitor Unit* (PMU) support based on the PMUv3 architecture.
- *Cross Trigger Interface* (CTI) for multiprocessor debugging.
- An integrated, fast-responding *Generic Interrupt Controller* (GIC) with virtualization.
- An online *Memory Built-In Self-Test* (MBIST) interface for testing memories, at boot time and at scheduled intervals after boot time.

1.1.2 Interfaces

The Cortex-R52 processor has several external interfaces.

The following figure shows the external interfaces of the Cortex-R52 processor. The arrows indicate the direction of signals in each interface.

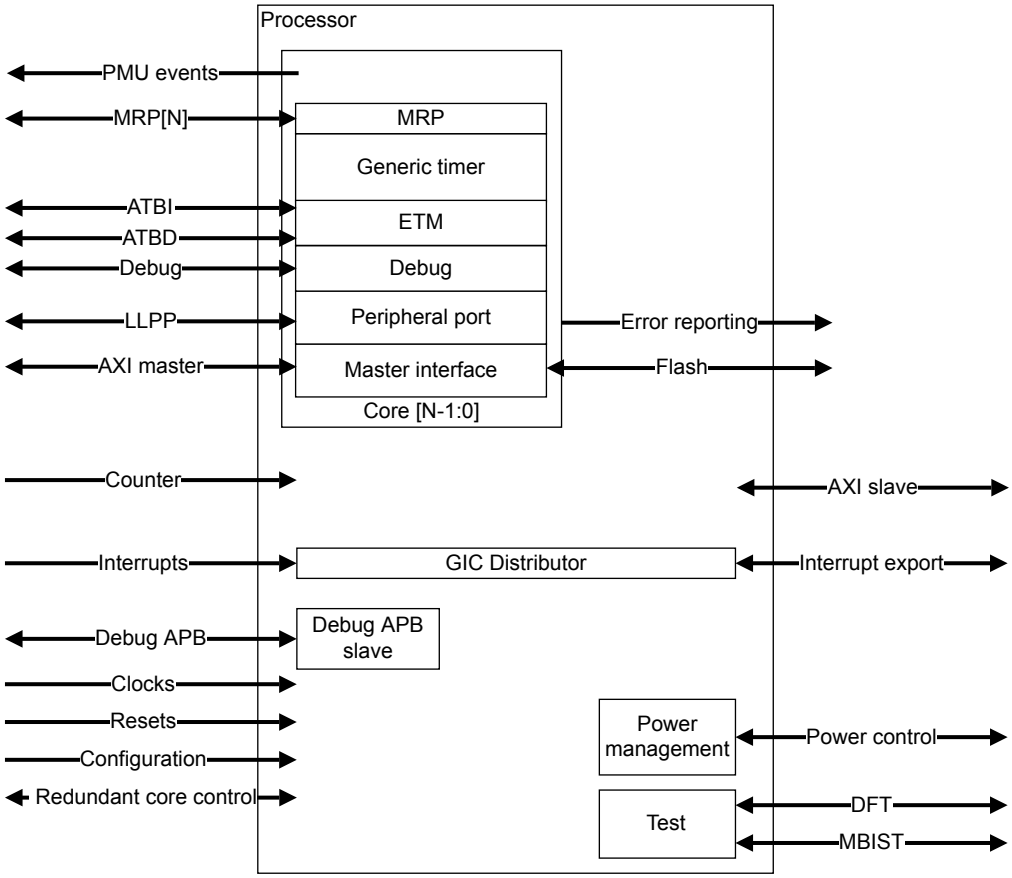

Figure 1-2 Cortex-R52 processor interfaces

1.1.3 Configuration options

The Cortex-R52 processor has options that you can configure during the implementation and integration stages to match your functional requirements.

The following table shows the configurable options of the processor.

Table 1-1 Configuration options

Feature	Options	Done at
Number of cores	1-4 cores.	Implementation
Lock-step	Redundant logic, flops, and comparators for DCLS included or not. DCLS configuration with 2, 4, 6, or 8 instances of the core logic, depending on the number of cores configured.	Implementation
EL1-controlled MPU	16, 20, or 24 programmable EL2-controlled *Memory Protection Unit* (MPU) regions per core.	Implementation
EL2-controlled MPU	0, 16, 20, or 24 programmable EL2-controlled MPU regions per core.	Implementation
AXI bus protection	No bus protection included, only signal integrity protection included, or signal integrity and interconnect protection are included.	Implementation
RAM protection	Included for all RAMs or not included.	Implementation
Number of interrupts (SPIs) into the interrupt controller	32-960 in multiples of 32, with a minimum of 32 interrupts per core.	Implementation
AXIS ID bits	Any non-zero number, but 5-16 preferred.	Implementation
Advanced SIMD and floating-point capabilities for each core	Single-precision floating-point only or single-precision and double-precision floating-point and Advanced SIMD.	Implementation
Size of each of the three TCMs on each core	0KB, 8KB-1MB (powers of 2).	Implementation
TCM wait states for each core	0 or 1 wait states.	Implementation
Instruction cache size for each core	4KB, 8KB, 16KB, 32KB, or excluded completely.	Implementation
Data cache size for each core	4KB, 8KB, 16KB, 32KB, or excluded completely.	Implementation
External device interfaces to GIC	0 or 1 external devices.	Implementation
Flash ECC scheme	Switches the flash memory integrity protection scheme between 64-bit ECC and 128-bit ECC.	Implementation
Reset all registers	Only required registers or all programmer-visible registers are reset in the hardware.	Integration
TCM boot	Boot with ATCM enabled and at address `0x0`, or disabled at reset.	Integration
Flash boot	Boot with flash memory enabled or disabled at reset.	Integration
Flash interface base address	Base address of Flash interface region.	Integration
Flash region present tie-off	Flash region access to AXIM interface or Flash interface.	Integration
LLPP interface base address and size	Base address and size (powers of 2) of LLPP region.	Integration
LLPP region present tie-off	LLPP region access to AXIM interface or LLPP.	Integration
TCMs base addresses	Base address of TCM regions as seen by AXIS interface.	Integration

Processor configurations

The Cortex-R52 processor can be configured to implement DCLS and Split/Lock configurations.

DCLS

In DCLS configurations, there is a second, redundant copy of the majority of the core logic for each core, and a redundant copy of the shared logic.

The redundant logic is driven by the same inputs as the functional logic. In particular, the redundant core logic shares the same cache RAMs and TCMs as the functional core. Therefore, only one set of cache RAMs and TCMs is required. The redundant logic operates in lock-step with the core, but does not directly affect the processor behavior in any way. The processor outputs to the rest of the system and the core outputs to the cache RAMs and TCMs are driven exclusively by the functional core.

During implementation, comparator logic can be included to compare the outputs of the redundant logic and the functional logic. These comparators can detect a single fault that occurs in either set of logic because of radiation or circuit failure. When used with RAM error detection schemes, the system can be protected from faults.

If you are implementing a DCLS configuration, contact Arm for more information.

Split/Lock

In Split/Lock configuration, there must be two or four complete redundant copies of each core. The following table shows how the cores are used in Lock mode and Split mode. In the following table:

- The number of physical cores is N.
- The number of cores used in Lock mode is LOCK_N.
- The number of cores used in Split mode is SPLIT_N.

Table 1-2 Split/Lock configuration

N	LOCK_N	SPLIT_N
2	1	2
4	2	4

In Lock mode, the higher order cores function as redundant copies of the lower order cores. For example, if N is 4, only the lower order cores are logically present, that is, core 0 and core 1. Core 2 and core 3 are the higher order cores which are logically not present, but function as redundant copies. Although present, the inputs and outputs, cache RAMs, and TCMs belonging to the higher order cores are disabled and must not be used in Lock mode.

In Split mode, all interfaces, cache RAMs, and TCMs associated to the number of physical cores selected are present and enabled but redundancy checking is not possible.

Similar to DCLS, comparator logic can be included to compare the outputs of the redundant logic and functional logic during Lock mode operation. Split mode operation disables the comparator logic.

For Split/Lock, a new input signal **CFGSLSPLIT** must be set to determine whether Split or Lock mode is configured. If Lock mode is selected, all the DCLS signals must be driven in addition to **CFGSLSPLIT**. If Split mode is selected, only **CLKINDCLS** must be driven in addition to **CFGSLSPLIT**. For more information on DCLS signals, see *A.15 DCLS signals* on page Appx-A-586.

If you are implementing Split/Lock configuration, contact Arm for more information.

Related references

1.2 Component blocks on page 1-18

1.2 Component blocks

The following figure shows the main component blocks of a processor in a two-core configuration.

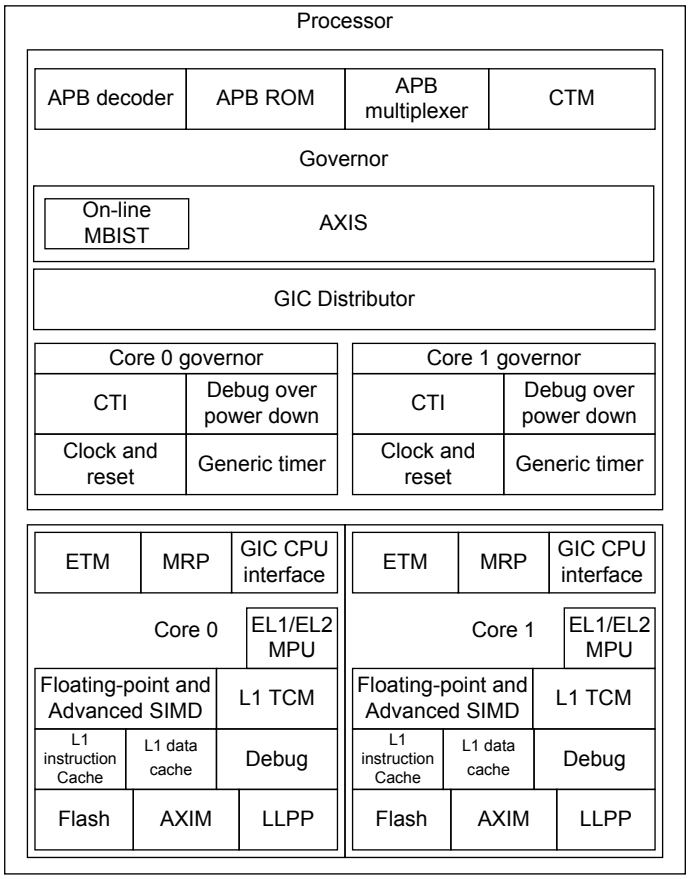

Figure 1-3 Functional block diagram

This section contains the following subsections:

1.2.1 Instruction Fetch

The *Prefetch Unit* (PFU) obtains instructions from the instruction cache, TCM, or from external memory and predicts the outcome of branches in the instruction stream and passes the instructions to the *Data Processing Unit* (DPU) for processing.

The PFU fetches 64 bits per cycle from memory.

Instruction fetches in the Cortex-R52 processor are always little-endian.

The instruction queue between the fetch and decode stages decouples instruction execution from instruction fetch. This allows instructions to continue to be executed while the fetch stages are being flushed due to a predicted branch.

The Cortex-R52 branch prediction mechanisms detect branches at an early stage in the pipeline. Also, they redirect instruction fetching to the appropriate address immediately, rather than waiting for the branch to reach the end of the pipeline. However, not all branches are predicted in this way.

Branch Target Address Cache
The PFU contains a 16-entry *Branch Target Address Cache* (BTAC) to predict the target address of indirect branches (except for subroutine returns). The BTAC implementation is architecturally transparent, so it does not have to be flushed on a context switch.

Branch predictor
The branch predictor is a global type that uses branch history registers and a 2048-entry pattern history prediction table.

Return stack
The PFU includes an 8-entry call-return stack to accelerate returns from subroutine calls. For each subroutine call, the return address is pushed onto a hardware stack. When a subroutine return is recognized, the address held in the return stack is popped, and the PFU uses it as the predicted return address. The return stack is architecturally transparent, so it does not have to be flushed on a context switch.

Exception Target Address Cache
The *Exception Target Address Cache* (ETAC) is a structure used to reduce the best case latency of IRQ and FIQ exceptions by caching the address of generic handler for these exceptions.
The ETAC is enabled out of reset. Writing 1 to the system register CPUACTLR.ETACDIS, disables the ETAC.
The ETAC supports caching of *Interrupt* (IRQ) and *Fast Interrupt* (FIQ) vector entries only. Other types of exceptions do not allocate entries into the ETAC. This is because a fast response to the IRQ and FIQ exceptions is most critical in real-time systems.
A vector is only cached in the ETAC if the vector is in a TCM. A vector located in any other type of memory never allocates or hits in the ETAC. This is because the TCMs are the only memories with a perfect response. Other memories can be subject to cache misses and in these cases the savings that the ETAC offers are minimal compared to the latency of the cache miss.
The ETAC only caches the vector corresponding to the IRQ or FIQ exception if the instruction in the vector table is a compatible instruction. Compatible instructions are all encodings of `B #immed`. If the exception vector is not a compatible instruction, the ETAC does not cache that exception.
The IRQ and FIQ exception can be taken to either Exception Level EL1 or EL2, depending on the exception level at the time of the interrupt and the values of HCR.IMO and HCR.FMO. The ETAC independently supports both IRQ and FIQ exceptions taken to both EL1 and EL2, which means that there are four independent entries for each of these cases.
For more information on:
- CPUACTLR, see *3.3.19 CPU Auxiliary Control Register* on page 3-90.
- HCR, see *3.3.39 Hyp Configuration Register* on page 3-111.

1.2.2 Advanced SIMD and floating-point support

The Advanced SIMD and floating-point that each core supports uses NEON™ technology, a SIMD architecture.

The Advanced SIMD and floating-point feature provides:
- Instructions for single-precision (C programming language `float` type) data-processing operations.
- Optional instructions for double-precision (C `double` type) data-processing operations.
- Combined Multiply and Accumulate instructions for increased precision (Fused MAC).
- Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root.
- Hardware support for denormals and all IEEE Standard 754-2008 rounding modes.
- For single-precision floating-point, there are 32 32-bit single-precision registers or 16 64-bit double-precision registers. If the optional instructions for the double-precision and Advanced SIMD are included, a total of 32 64-bit double-precision registers or 16 128-bit registers are available.

Related references

Chapter 15 Advanced SIMD and floating-point support on page 15-528

1.2.3 GIC Distributor

The GIC Distributor receives, prioritizes, and routes physical interrupts to the appropriate interrupt target.

The output of the GIC Distributor is the highest priority pending interrupt for each interrupt target. An interrupt target is either the GIC CPU interface for a core or an export port for connection to an external device such as a *Direct Memory Access* (DMA) controller.

Related references

Chapter 9 Generic Interrupt Controller on page 9-260

1.2.4 GIC CPU interface

The GIC CPU interfaces handle interrupt preemption for both physical and virtual interrupts for each core.

The virtual part of each GIC CPU interface is divided into hypervisor registers and guest OS registers. The hypervisor generates interrupts to the guest OS using the GIC CPU interface.

Related references

Chapter 9 Generic Interrupt Controller on page 9-260

1.2.5 Memory system

The Cortex-R52 memory system provides different memories and interfaces depending on your implementation.

Intended for use by contexts without strong real-time requirements, each Cortex-R52 core has a dedicated 128-bit AXIM interface for memory, instructions and data, and peripheral access.

Also, intended for use by real-time contexts, each Cortex-R52 core can have:

- Three unified TCMs, each 8KB-1MB providing lowest-latency access for instructions and data.
- Optionally, 32-bit AXI4 LLPP interface for device data accesses to private peripherals.
- 128-bit read-only Flash interface.
- ECC protection for all TCM and flash memories providing SECDED protection.
- TCM access for DMA through the AXIS interface.
- TCM testing using the MBIST interface.

Note

A real-time context is also able to access the AXIM, although such an access might not be desirable depending on the system design.

Each Cortex-R52 core has optional Harvard caches, which can be used to cache data from the Flash interface and the AXIM interface. The cache behavior depends on the memory attributes.

Each core has:

- Store buffer with merging and forwarding (as appropriate) for stores.
- 4-way instruction cache of 4-32KB.
- Instruction linefill buffering.
- 4-way data cache of 4-32KB with Write-Through behavior.
- Data read buffers.
- ECC protection for all cache memories (including tag RAM).
- 64-bit datapath for loads and stores to caches.
- Cache maintenance operations according to Arm architecture.
- Cache memory testing using the MBIST interface.

Related references

Chapter 7 Memory System on page 7-218

1.2.6 Memory management

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

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

Related references

Chapter 8 Memory Protection Unit on page 8-251

1.2.7 Debug, trace, and test

The processor has manufacturing test facilities and each core in the processor has both invasive debug and non-invasive debug features to support software debug.

The invasive debug features include hardware exception catching, breakpoints and watchpoints, data transfer through a Debug Communications Channel, and single-stepping. Invasive debug modes are self-hosted (or monitor) debug and external (or halting) debug. Invasive debug impacts the performance of the processor, although you can schedule self-hosted debug if time permits.

The non-invasive debug features include tracing instruction execution and tracing data transfers. Each core has its own ETM permitting all cores to be traced simultaneously subject to the available trace bandwidth. Trace information is exported on the ATB trace buses that can be connected to a CoreSight system for combining trace sources, buffering, and exporting them.

The processor includes a PMU that generates and can count events that occur in the core such as cache linefills, pipeline stalls, and ECC faults. These events are also exported for use by external hardware as appropriate.

Manufacturing test includes MBIST.

Related references

Chapter 11 Debug on page 11-350

Chapter 13 Cross Trigger on page 13-429

Chapter 14 Embedded Trace Macrocell on page 14-447

1.3 Interfaces

The Cortex-R52 processor has several external interfaces.

This section contains the following subsections:

1.3.1 Advanced Microcontroller Bus Architecture (AMBA) interfaces

The processor implements the following AMBA interfaces.

AXIM

Each core of the Cortex-R52 processor has a 128-bit AXIM interface that provides high-performance access to external memory and peripherals.

Related references

7.5 AXIM interface on page 7-228

A.8.1 AXIM interface signals on page Appx-A-560

AXIS

Each core in the Cortex-R52 processor is connected to a common 128-bit AXIS interface. This provides external access to the TCMs. The AXIS interface supports DMA access between an external controller and the internal memories.

Related references

7.8 AXIS interface on page 7-243

A.8.2 AXIS interface signals on page Appx-A-564

Advanced Peripheral Bus (APB) Debug interface

AMBA APBv3 interface is used for debugging purposes.

Related references

Debug APB interface signals on page Appx-A-576

LLPP

Each core in the Cortex-R52 processor has an interface to a dedicated 32-bit LLPP master interface. These ports are intended to be used for private peripherals requiring low-latency access.

Related references

7.6 Low-latency peripheral port on page 7-233

A.8.3 LLPP interface signals on page Appx-A-568

1.3.2 Flash interface

Each core in the Cortex-R52 processor has a dedicated 128-bit read-only Flash master interface, which can be used as an AXI4 interface. This interface is intended to provide low-latency access to flash memories attached to the processor, enabling the processor to execute real-time tasks from a flash ROM with caching.

Related references

7.7 Flash interface on page 7-241

1.3.3 Memory Reconstruction Port

Each core of the Cortex-R52 processor provides an MRP for reporting write accesses so that an image of memory can be reconstructed.

The main features of the MRP interface for each core are:

- Trace information from the MRP is uncompressed.
- The MRP does not include any filtering.

Related references

A.7 MRP signals on page Appx-A-558

1.3.4 Interrupt interface

The interrupt interface for the Cortex-R52 processor has an input port for a configurable number of *Shared Peripheral Interrupts* (SPIs), from 32 to 960 (in multiples of 32), with a minimum of 32 for each core. All SPIs can be configured to be rising edge-triggered or level-sensitive active-HIGH.

The interrupt distributor has an optional interrupt export port for routing interrupts to an external device such as a DMA engine. The interrupt controller selects this port for routing SPIs in the same way as it routes interrupts to a core.

The interrupt controller also provides *Private Peripheral Interrupts* (PPIs) and *Software Generated Interrupts* (SGIs), which are private to each core. Some PPIs are exposed as primary inputs.

Related references

Chapter 9 Generic Interrupt Controller on page 9-260

1.3.5 MBIST interface

The MBIST interface is used for testing the RAMs during production test.

The Cortex-R52 processor allows the RAMs to be tested using the MBIST interface during normal execution. This is known as on-line MBIST.

Existing production test MBIST controllers do not support the requirements of on-line MBIST, therefore, it is necessary to use PMC-R52 which was specifically developed for this purpose. PMC-R52 is part of the Cortex-R52 RTL. It is an integration option and may not be present on every implementation. For more information on PMC-R52, see *Appendix D PMC-R52* on page Appx-D-629.

Contact your implementation team for more information about the MBIST interface and on-line MBIST.

1.3.6 Low Power Interface

The P-channel interfaces are used to signal power state information to an external power controller.

Related references

Chapter 5 Power Management on page 5-204

1.4 Supported standards

The processor complies with, or implements, various specifications defined by Arm.

1.4.1 Arm architecture

The Cortex-R52 processor implements the Armv8-R architecture. This includes:

- Support for AArch32 Execution state.
- Support for Exception levels, EL0, EL1, and EL2.
- Support for floating-point computation functionality that is compliant with the ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic.

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

1.4.2 AMBA

The Cortex-R52 processor complies with the:

- AMBA 4 *Advanced eXtensible Interface* (AXI) protocol for the main AXIM interface, AXIS interface, and LLPP ports. See the *AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™, ACE and ACE-Lite™*.
- AMBA 3 APB protocol. See the *AMBA® 3 APB Protocol v1.0 Specification*.
- AMBA 4 *Advanced Trace Bus* (ATB) protocol. See the *AMBA® 4 ATB Protocol Specification ATBv1.0 and ATBv1.1*.

1.4.3 Generic Interrupt Controller architecture

The Cortex-R52 processor supports a subset of GIC architecture version 3.

See the *Arm® Generic Interrupt Controller Architecture Specification GIC architecture version 3.0 and version 4.0*.

1.4.4 Generic Timer architecture

The processor implements the Arm Generic Timer architecture.

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

1.4.5 Debug architecture

The processor implements the Armv8-R debug architecture that complies with the CoreSight architecture.

See the following for more information:

- *Arm® CoreSight™ Architecture Specification v2.0*.
- *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

1.4.6 Embedded Trace Macrocell architecture

The processor implements the ETMv4.2 architecture.

See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

1.5 Documentation

The Cortex-R52 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 processor. It is required at all stages of the design flow. The choices made earlier in the design flow can mean that some behavior described in the TRM is not relevant. If you are programming the processor, additional information must be obtained from:

- The implementer to determine the build configuration of the implementation.
- The integrator to determine the pin configuration of the device that you are using.

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 validate the RTL.
- How to run test vectors.
- Considerations for floor-planning.
- 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. For EDA tool support, contact your EDA vendor.

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

Integration Manual

The *Integration Manual* (IM) describes how to integrate the processor into a SoC. It describes the signals that the integrator must tie off to configure the macrocell for the required integration. Some of the implementation options might affect which integration options are available.

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

Safety Manual

The *Safety Manual* (SM) and other associated documents describe in detail the capabilities of the fault detection and control features, the development process and assumptions of use as required to support product development incorporating the processor with defined functional safety requirements.
The SM is a confidential book that is only available to licensees.

1.6 Design process

The Arm processor is delivered as synthesizable RTL. Before the processor can be used in a product, it must go through the following process:

Implementation
The implementer configures and synthesizes the RTL to produce a hard macrocell. This might include integrating the cache RAMs into the design.

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

Programming
This is the last process. The system programmer develops:

- Software to configure the Arm processor.
- Software to initialize the Arm processor.
- Application software and the SoC tests.

Each process:

- Can be performed by a different party.
- Can involve making configuration choices that affect the behavior and features of the processor.

The operation of the final device depends on:

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

Configuration inputs
The integrator configures some features of the processor by tying inputs to specific values. These 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 processor by programming particular values into registers. This affects the behavior of the processor.

1.7 Product revisions

This section describes the differences in functionality between product revisions.

r0p0 First release.

r1p0 Second release. For more information, see *E.1 Revisions* on page Appx-E-688.

r1p1 Third release. For more information, see *E.1 Revisions* on page Appx-E-688.

r1p2 Fourth release. For more information, see *E.1 Revisions* on page Appx-E-688.

r1p3 Fifth release. For more information, see *E.1 Revisions* on page Appx-E-688.

Chapter 2
Programmers Model

This chapter describes the programmers model.

It contains the following sections:

2.1 About the programmers model

The Cortex-R52 processor implements the Armv8-R architecture. This includes:

- All the Exception levels, EL0-EL2.
- AArch32 execution state at each Exception level.
- T32 and A32 instruction sets that include:
  - Floating-point operations.
  - Optionally, Advanced SIMD operations.

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

This section contains the following subsections:

2.1.1 Advanced SIMD and Floating-point

Advanced SIMD is a media and signal processing architecture.

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 technology.

All Advanced SIMD and floating-point operations are part of the A32 and T32 instruction sets.

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

2.1.2 Generic Interrupt Controller

The Cortex-R52 processor does not support the following GIC version 3 features:

- 1of N interrupt distribution.
- *Locality-specific Peripheral Interrupts* (LPIs).
- Interrupt Translation Service (ITS).

2.1.3 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 processor does not accelerate the execution of any bytecodes, and the *Java Virtual Machine* (JVM) uses software routines to execute all bytecodes.

2.1.4 Instruction set states

The processor operates two instruction set states controlled by the CPSR.T bit:

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

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

2.1.5 Memory model

The Cortex-R52 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 access words in memory as either:

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

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

Note

Instructions are always little-endian.

2.1.6 Security state

The Cortex-R52 processor does not implement TrustZone® technology. It does not support the ability to distinguish between secure and non-secure physical memories.

2.2 Armv8-R architecture concepts

The programmers model for the Cortex-R52 processor is mostly defined by the architecture it implements. This manual does not include a duplicate description of the architectural programmers model. This manual describes features and behaviors that are specific to the Cortex-R52 processor implementation.

The following sections provide an introduction to the main architectural concepts and terminology used throughout the rest of this document. For more details, see the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

——— **Note** ———

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

———————————

This section contains the following subsections:

2.2.1 Execution state

The Armv8-R architecture has only one execution state, AArch32. The execution state defines the processor execution environment, including:

- Supported register widths.
- Supported instruction sets.
- Significant aspects of:
  - The execution model.
  - PMSA.
  - The programmers model.

2.2.2 Exception levels

The Armv8-R exception model defines Exception levels EL0-EL2, where:

- EL0 has the lowest software execution privilege, and execution at EL0 is called unprivileged execution.
- Increased Exception levels, from 1 to 2, indicate increased software execution privilege.
- EL2 provides support for processor virtualization.

Execution can move between Armv8-R 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 EL0.

2.2.3 Typical exception level usage model

The architecture does not specify what software uses at 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.

2.2.4 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 that 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 that the exception returns from. The processor state immediately after the execution of that instruction is the state the exception returns to.

Fast interrupts

When fast interrupts are enabled, when an interrupt is received, the processor abandons any started but incomplete restartable memory operations. Restartable memory operations are load and store instructions to Normal memory.

To minimize interrupt latency, Arm recommends that you do not perform multiword transfer operations to Device memory.

2.2.5 Instruction set state

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

The instruction sets supported in the AArch32 execution state are:

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.

2.2.6 AArch32 execution modes

When in AArch32 state the processor can execute in one of several modes. Each mode is associated with an Exception level. Some modes have private, banked copies of some of the general purpose registers. Exceptions cause the processor to switch to a specific mode.

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

Table 2-1 AArch32 processor modes and associated Exception levels

AArch32 processor mode	Exception level
User	EL0
System, FIQ, IRQ, Supervisor, Abort, Undefined	EL1
Hyp	EL2

2.2.7 Support for v8 memory types

Armv8 provides mutually exclusive memory types. Each address in the memory map has a memory type which is determined by the MPU.

The memory types are:

Normal This is generally used for bulk memory, both read/write and read-only.

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

The Armv8 architecture subdivides Device memory into several subtypes. These relate to the following attributes:

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

R Reordering. The capability to reorder transactions.

E Early-acknowledge. The capability to accept early acknowledgment of transactions from the interconnect.

The following table describes the Armv8 memory types.

Table 2-2 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-R52 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-R52 processor, but reported differently on **ARCACHE** or **AWCACHE**.

2.2.8 System registers

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

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

The System registers comprise of:

- ID registers.
- General system control registers.
- Debug registers.
- Generic Timer registers.

- Performance Monitor registers.
- GIC CPU interface registers.

Related references

Chapter 9 Generic Interrupt Controller on page 9-260
Chapter 10 Generic Timer on page 10-346
Chapter 11 Debug on page 11-350
Chapter 12 Performance Monitor Unit on page 12-398

2.2.9 General purpose registers

The Armv8-R architecture provides access to 15 32-bit general purpose registers, R0-R14.

Of these, two have specialized or restricted usage:

- R13 is usually identified as SP, the stack pointer.
- R14 is usually identified as LR, the link register.

There is also a 32-bit *Program Counter* (PC), R15.

Some of these registers are banked. These are multiple physical registers accessed using the same identifier and the execution mode determines which one is accessed. For example:

- In abort mode, an access to SP accesses the physical register SP_abt.
- In user mode, an access to SP accesses physical register SP_usr.

In Hypervisor mode, there is an additional, dedicated link register, ELR_hyp.

An additional set of registers is used by Advanced SIMD and floating-point instructions. These registers can be accessed as 32-bit single-precision registers S0-S31, or as 64-bit double-precision registers D0-D31, or 128-bit quad registers Q0-Q15, but these are different views of the same data. Data can be transferred between the Advanced SIMD and floating-point registers and the integer registers.

2.2.10 Program status registers

The program status registers are the *Current Program Status Register* (CPSR), the *Application Program Status Register* (APSR), and the *Saved Program Status Register* (SPSR).

The CPSR holds information including:

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

Many of the bits in the CPSR have restricted usage and can only be modified:

- In privileged modes.
- As a side-effect of an exception or exception return.

The view of the CPSR presented to applications running at Exception level EL0 is called the APSR.

Every mode that an exception can be taken to has its own SPSR. Each SPSR is used to save a copy of the CPSR when an exception is taken, allowing it to be restored on the exception return.

2.2.11 Data types

The Armv8-R architecture core supports the following integer data types:

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

The Armv8-R architecture also supports half-precision, single-precision, and double-precision floating-point data types.

2.2.12 Memory model

The Armv8-R architecture defines the PMSAv8 memory model that determines:

- Ordering rules and other restrictions on how memory accesses, for instructions and data, are performed.
- Permissions and other attributes for memory accesses, and how these are determined by the MPU.

Cortex-R52 supports PMSAv8 and has no MMU or address translation capabilities. Even though the MPU does not perform any actual translation, the function is still generically referred to as translation. This is because the physical address is always the same as the virtual address. Virtual and physical address refers to the address before the translation process, as generated by the instruction and after the translation process, as visible on the bus. The virtual and physical address can be treated as synonyms for Cortex-R52.

2.2.13 GIC Architecture

The Cortex-R52 processor supports three components of the GIC architecture: Distributor, Redistributor, and CPU interface.

The Distributor contains the registers supporting SPIs in addition to the prioritization logic which calculates the highest priority pending interrupt for each core. The Redistributor contains the registers supporting PPIs and SGIs. The CPU interface tracks the current running priority and virtual interrupts. It determines whether the core is interrupted.

Interrupts are configured to be either edge-triggered or level-sensitive. Each interrupt has programmable enable, priority (0-31), group (0 or 1), and routing (SPIs only) configuration and a current state.

When an interrupt signal is received, it is pended. According to its configuration, a pended interrupt might interrupt the execution of a core. Software can read the ID of the interrupt, which has the side effect of activating it. Software can also end an interrupt, which deactivates it.

Virtual interrupts are created by hypervisor software writing to a list register in the CPU interface, usually in response to a physical interrupt. Virtual interrupts are handled in a similar way to physical interrupts but can only interrupt a core when it is in EL0 or EL1. When a virtual interrupt is deactivated, it is possible to configure it in a way that when the virtual interrupt is deactivated, a corresponding physical deactivation message also send to the GIC Distributor.

Related references

1.4 Supported standards on page 1-24

Chapter 3
System Control

This chapter describes the system registers and their structure and operation.

It contains the following sections:

3.1 About system control

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

The main functions of the system registers are:

- Overall system control and configuration.
- MPU configuration and management.
- Cache configuration and management.
- System performance monitoring.
- GIC CPU interface configuration and management.

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

The terms RES0, RES1, RAZ, RAZ/WI, RAO/WI, and RAZ/SBZ are described in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

For information on the GIC registers, see *Chapter 9 Generic Interrupt Controller* on page 9-260 or the *Arm® Generic Interrupt Controller Architecture Specification GIC architecture version 3.0 and version 4.0*.

For information on debug related system registers (coproc==`0b1110`), see *11.3 System register summary* on page 11-356.

Related references

11.5 Memory-mapped register summary on page 11-363

3.2 Register summary

The system registers are accessed using `MCR`, `MRC`, `MRRC`, or `MCRR` instructions with (coproc==`0b1111`).

The system register space includes system registers and system operations. The description of the system register space describes the permitted access, *Read-Only* (RO), *Write-Only* (WO), or *Read-Write* (RW), to each register or operation.

The following table describes the column headings in the register summary tables used throughout this section.

Table 3-1 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.

This section contains the following subsections:

3.2.1 c0 registers

The following table shows the 32-bit system registers you can access when the value of CRn is c0.

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

Table 3-2 c0 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c0	0	c0	0	MIDR	RO	`0x411FD132`	*3.3.69 Main ID Register* on page 3-156
			1	CTR	RO	`0x8144c004`	*3.3.16 Cache Type Register* on page 3-87
			2	TCMTR	RO	-[a]	*3.3.96 TCM Type Register* on page 3-192
			3	TLBTR	RO	`0x00000000`	*3.3.99 TLB Type Register* on page 3-194
			4	MPUIR	RO	-[b]	*3.3.77 MPU Type Register* on page 3-166
			5	MPIDR	RO	-[c]	*3.3.78 Multiprocessor Affinity Register* on page 3-167
			6	REVIDR	RO	`0x00000000`	*3.3.90 Revision ID Register* on page 3-182
			7	MIDR	RO	`0x411FD132`	Alias of the Main ID Register, *3.3.69 Main ID Register* on page 3-156
c0	0	c1	0	ID_PFR0	RO	`0x00000131`	*3.3.83 Processor Feature Register 0* on page 3-174
			1	ID_PFR1	RO	`0x10111001`	*3.3.84 Processor Feature Register 1* on page 3-175
			2	ID_DFR0	RO	`0x03010006`	*3.3.24 Debug Feature Register 0* on page 3-99
			3	ID_AFR0	RO	`0x00000000`	*3.3.5 Auxiliary Feature Register 0* on page 3-78
			4	ID_MMFR0	RO	`0x00211040`	*3.3.71 Memory Model Feature Register 0* on page 3-159
			5	ID_MMFR1	RO	`0x40000000`	*3.3.72 Memory Model Feature Register 1* on page 3-160
			6	ID_MMFR2	RO	`0x01200000`	*3.3.73 Memory Model Feature Register 2* on page 3-161
			7	ID_MMFR3	RO	`0xF0102211`	*3.3.74 Memory Model Feature Register 3* on page 3-163

a The reset value depends on how many TCMs are implemented. See *3.3.96 TCM Type Register* on page 3-192.

b The reset value depends on the number of EL1-controlled MPU regions implemented.
- 16 EL1-controlled MPU regions=`0x00001000`
- 20 EL1-controlled MPU regions=`0x00001400`
- 24 EL1-controlled MPU regions=`0x00001800`

Bits [15:8] contain the number of regions.

c The reset value depends on the cluster and processor ID. See *3.3.78 Multiprocessor Affinity Register* on page 3-167.

d The reset value is the value of the Main ID Register.

Table 3-2 c0 register summary (continued)

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c0	0	c2	0	ID_ISAR0	RO	0x02101110	*3.3.60 Instruction Set Attribute Register 0* on page 3-144
			1	ID_ISAR1	RO	0x13112111	*3.3.61 Instruction Set Attribute Register 1* on page 3-146
			2	ID_ISAR2	RO	0x21232142	*3.3.62 Instruction Set Attribute Register 2* on page 3-147
			3	ID_ISAR3	RO	0x01112131	*3.3.63 Instruction Set Attribute Register 3* on page 3-149
			4	ID_ISAR4	RO	0x00010142	*3.3.64 Instruction Set Attribute Register 4* on page 3-151
			5	ID_ISAR5	RO	0x00010001	*3.3.65 Instruction Set Attribute Register 5* on page 3-152
			6	ID_MMFR4	RO	0x00000010	*3.3.75 Memory Model Feature Register 4* on page 3-164
c0	1	c0	0	CCSIDR	RO	UNK	*3.3.20 Current Cache Size ID Register* on page 3-93
			1	CLIDR	RO	-[g]	*3.3.13 Cache Level ID Register* on page 3-84
			7	AIDR	RO	0x00000000	*3.3.6 Auxiliary ID Register* on page 3-78
c0	2	c0	0	CSSELR	RW	0x00000000	*3.3.15 Cache Size Selection Register* on page 3-86
c0	4	c0	0	VPIDR	RW	-[d]	*3.3.102 Virtualization Processor ID Register* on page 3-196
			4	HMPUIR	RO	-[e]	*3.3.47 Hyp MPU Type Register* on page 3-125
			5	VMPIDR	RW	-[f]	*3.3.101 Virtualization Multiprocessor ID Register* on page 3-195

3.2.2 c1 registers

The following table shows the 32-bit system registers you can access when the value of CRn is c1.

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

[e] The reset value depends on the number of EL2-controlled MPU regions.
- No El2-controlled MPU regions=0x00000000.
- 16 El2-controlled MPU regions=0x00000010.
- 20 El2-controlled MPU regions=0x00000014.
- 24 El2-controlled MPU regions=0x00000018.

[f] Bits [7:0] contain the number of regions.
The reset value is the value of the Multiprocessor Affinity Register.

[g] The reset value depends on if the Instruction and Data caches are implemented. See register description.

Table 3-3 c1 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c1	0	c0	0	SCTLR	RW	-[i]	*3.3.92 System Control Register* on page 3-184
			1	ACTLR	RW	0x00000000	*3.3.2 Auxiliary Control Register* on page 3-77
			2	CPACR	RW	0x00000000	*3.3.1 Architectural Feature Access Control Register* on page 3-76
			3	ACTLR2	RW	0x00000000	*3.3.3 Auxiliary Control Register 2* on page 3-77
		c1	2	NSACR	RO	0x00000C00	*3.3.79 Non-Secure Access Control Register* on page 3-168
	4	c0	0	HSCTLR	RW	-[i]	*3.3.53 Hyp System Control Register* on page 3-133
			1	HACTLR	RW	0x00000000	*3.3.33 Hyp Auxiliary Control Register* on page 3-107
			3	HACTLR2	RW	0x00000000	*3.3.34 Hyp Auxiliary Control Register 2* on page 3-108
		c1	0	HCR	RW	0x00000002	*3.3.39 Hyp Configuration Register* on page 3-111
			1	HDCR	RW	0x00000004	*3.3.42 Hyp Debug Control Register* on page 3-117
			2	HCPTR	RW	-[h]	*3.3.31 Hyp Architectural Feature Trap Register* on page 3-105
			3	HSTR	RW	0x00000000	*3.3.54 Hyp System Trap Register* on page 3-137
			4	HCR2	RW	0x00000000	*3.3.40 Hyp Configuration Register 2* on page 3-116
			7	HACR	RW	0x00000000	*3.3.32 Hyp Auxiliary Configuration Register* on page 3-106

3.2.3 c2 registers

The following table shows the 32-bit system register you can access when the value of CRn is c2.

Table 3-4 c2 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c2	4	c0	0	VSCTLR	RW	0x00000000	*3.3.103 Virtualization System Control Register* on page 3-196

3.2.4 c3 registers

There are no system registers with the value of CRn as c3.

3.2.5 c4 registers

The following table shows the 32-bit system registers you can access when the value of CRn is c4.

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

[h] The reset value depends on the Advanced SIMD and floating-point configuration.
[i] The reset value depends on inputs, **CFGTHUMBEXCEPTION** and **CFGENDIANESS**.

Table 3-5 c4 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c4	0	c6	0	ICC_PMR, ICV_PMR	RW	0x00000000	*Interrupt Controller Interrupt Priority Mask Register* on page 9-321 *Interrupt Controller Virtual Interrupt Priority Mask Register* on page 9-340
	3	c5	0	DSPSR	RW	UNK	Debug Saved Program Status Register
			1	DLR	RW	-[j]	Debug Link Register

3.2.6 c5 registers

The following table shows the 32-bit system registers you can access when the value of CRn is c5.

Table 3-6 c5 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c5	0	c0	0	DFSR	RW	UNK	*3.3.23 Data Fault Status Register* on page 3-97
			1	IFSR	RW	UNK	*3.3.59 Instruction Fault Status Register* on page 3-143
		c1	0	ADFSR	RW	UNK	*3.3.4 Auxiliary Data Fault Status Register* on page 3-77
			1	AIFSR	RW	UNK	*3.3.7 Auxiliary Instruction Fault Status Register* on page 3-78
	4	c1	0	HADFSR	RW	UNK	*3.3.35 Hyp Auxiliary Data Fault Status Register* on page 3-108
			1	HAIFSR	RW	UNK	*3.3.36 Hyp Auxiliary Instruction Fault Status Register* on page 3-110
		c2	0	HSR	RW	UNK	*3.3.52 Hyp Syndrome Register* on page 3-131

3.2.7 c6 registers

The following table shows the 32-bit system registers you can access when the value of CRn is c6.

[j] The reset value is the value of the PC in Halting Debug state.

Table 3-7 c6 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c6	0	c0	0	DFAR	RW	UNK	*3.3.22 Data Fault Address Register* on page 3-96
			2	IFAR	RW	UNK	*3.3.58 Instruction Fault Address Register* on page 3-142
		c2	1	PRSELR	RW	UNK	*3.3.87 Protection Region Selection Register* on page 3-179
		c3	0	PRBAR	RW	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
			1	PRLAR	RW	UNK[k]	*3.3.86 Protection Region Limit Address Register* on page 3-178
		c8-15	0	PRBAR0-14 (even)	RW	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
			1	PRLAR0-14 (even)	RW	UNK[k]	*3.3.86 Protection Region Limit Address Register* on page 3-178
			4	PRBAR1-15 (odd)	RW	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
			5	PRLAR1-15 (odd)	RW	UNK[k]	*3.3.86 Protection Region Limit Address Register* on page 3-178
c6	4	c0	0	HDFAR	RW	UNK	*3.3.41 Hyp Data Fault Address Register* on page 3-116
			2	HIFAR	RW	UNK	*3.3.43 Hyp Instruction Fault Address Register* on page 3-120
			4	HPFAR	RW	UNK	*3.3.44 Hyp IPA Fault Address Register* on page 3-121
		c1	1	HPRENR	RW	0x00000000	*3.3.46 Hyp MPU Region Enable Register* on page 3-124
		c2	1	HPRSELR	RW	UNK	*3.3.50 Hyp Protection Region Selection Register* on page 3-129
		c3	0	HPRBAR	RW	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
			1	HPRLAR	RW	UNK[k]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128
		c8-15	0	HPRBAR0-14 (even)	RW	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
			1	HPRLAR0-14 (even)	RW	UNK[k]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128
			4	HPRBAR1-15 (odd)	RW	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
			5	HPRLAR1-15 (odd)	RW	UNK[k]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128

3.2.8 c7 registers

The following table shows the 32-bit system register you can access when the value of CRn is c7.

[k] The reset value for bit[0] is 0.

Table 3-8 c7 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c7	0	c4	0	PAR	RW	UNK	*3.3.81 Physical Address Register* on page 3-170

3.2.9 c7 System operations

The following table shows the System operations when CRn is c7.

The following operations are fully documented in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*, that is, there are no Cortex-R52-specific details.

Table 3-9 c7 System operation summary

op1	CRm	op2	Name	Description
0	c1	0	ICIALLUIS	Invalidate all instruction caches Inner Shareable to *Point of Unification* (PoU).
		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-R
		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[1]
		2	DCISW	Invalidate data cache line by set/way
	c8	0	ATS1CPR	Address Translate Stage 1 current state EL1 read
		1	ATS1CPW	Address Translate Stage 1 current state EL1 write
		2	ATS1CUR	Address Translate Stage 1 current state unprivileged read
		3	ATS1CUW	Address Translate Stage 1 current state unprivileged write
		4	ATS12NSOPR	Address Translate Stages 1 and 2 Non-secure only EL1 read
		5	ATS12NSOPW	Address Translate Stages 1 and 2 Non-secure only EL1 write
		6	ATS12NSOUR	Address Translate Stages 1 and 2 Non-secure only unprivileged read
		7	ATS12NSOUW	Address Translate 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-R
		5	CP15DMB	Data Memory Barrier operation, this operation is deprecated in Armv8-R
	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	Address Translate Stage 1 Hyp mode read
		1	ATS1HW	Address Translate Stage 1 Hyp mode write

[1] *Point of Coherence* (PoC) is always outside of the processor and depends on the external memory system.

3.2.10 c8 System operations

System operations when CRn is c8 excute as a `NOP`.

3.2.11 c9 registers

The following table shows the 32-bit system registers you can access when the value of CRn is c9.

c9 includes the Performance Monitor registers. With the exception of ID fields in PMCR, PMCEID0, and PMCEID1, the event selection values used by PMEVNTYPER, and the number of writable bits in various registers, these registers are fully documented by the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*. This implies that there are no Cortex-R52-specific details.

Table 3-10 c9 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c9	0	c1	0	IMP_ATCMREGIONR	RW	-[m]	*3.3.94 TCM Region Registers A, B, and C* on page 3-189
			1	IMP_BTCMREGIONR	RW	-[m]	*3.3.94 TCM Region Registers A, B, and C* on page 3-189
			2	IMP_CTCMREGIONR	RW	-[m]	*3.3.94 TCM Region Registers A, B, and C* on page 3-189
		c12	0	PMCR	RW	0x41132000	*12.3.1 Performance Monitors Control Register* on page 12-403
			1	PMCNTENSET	RW	UNK	Performance Monitors Count Enable Set Register
			2	PMCNTENCLR	RW	UNK	Performance Monitors Count Enable Clear Register
			3	PMOVSR	RW	UNK	Performance Monitor Overflow Flag Status Clear Register
			4	PMSWINC	WO	UNK	Performance Monitors Software Increment Register
			5	PMSELR	RW	UNK	Performance Monitors Event Counter Selection Register
			6	PMCEID0	RO	0x6E1FFFDB	*12.3.2 Performance Monitors Common Event Identification Register 0* on page 12-405
			7	PMCEID1	RO	0x0000001E	*12.3.3 Performance Monitors Common Event Identification Register 1* on page 12-406
		c13	0	PMCCNTR	RW	UNK	Performance Monitors Cycle Count Register
			1	PMXEVTYPER	RW	0x00000000	Performance Monitors Selected Event Type Register
			2	PMXEVCNTR	RW	UNK	Performance Monitors Selected Event Count Register
		c14	0	PMUSERENR	RW	0x00000000	Performance Monitors User Enable Register
			1	PMINTENSET	RW	UNK	Performance Monitors Interrupt Enable Set Register
			2	PMINTENCLR	RW	UNK	Performance Monitors Interrupt Enable Clear Register
			3	PMOVSSET	RW	UNK	Performance Monitor Overflow Flag Status Set Register
c9	1	c1	0	IMP_CSCTLR	RW	0x00000000	*3.3.14 Cache Segregation Control Register* on page 3-85
			1	IMP_BPCTLR	RW	0x00000000	*3.3.10 Branch Predictor Control Register* on page 3-80
			2	IMP_MEMPROTCTLR	RW	-[n]	*3.3.76 Memory Protection Control Register* on page 3-165

Related references

Chapter 12 Performance Monitor Unit on page 12-398

[m] The reset value depends on implementation (TCM size), and input pins. See register description.
[n] The reset value depends on the signals **CFGFLASHPROTEN** and **CFGRAMPROTEN**, see register description.

3.2.12 c10 registers

The following table shows the 32-bit system registers you can access when the value of CRn is c10.

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

Table 3-11 c10 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c10	0	c2	0	MAIR0	RW	0x00098AA4	*3.3.70 Memory Attribute Indirection Registers 0 and 1* on page 3-157
			1	MAIR1	RW	0x44E048E0	*3.3.70 Memory Attribute Indirection Registers 0 and 1* on page 3-157
		c3	0	AMAIR0	RW	0x00000000	*3.3.8 Auxiliary Memory Attribute Indirection Register 0* on page 3-80
			1	AMAIR1	RW	0x00000000	*3.3.9 Auxiliary Memory Attribute Indirection Register 1* on page 3-80
	4	c2	0	HMAIR0	RW	0x00000000	*3.3.45 Hyp Memory Attribute Indirection Register 0 and 1* on page 3-122
			1	HMAIR1	RW	0x00000000	*3.3.45 Hyp Memory Attribute Indirection Register 0 and 1* on page 3-122
		c3	0	HAMAIR0	RW	0x00000000	*3.3.37 Hyp Auxiliary Memory Attribute Indirection Register 0* on page 3-111
			1	HAMAIR1	RW	0x00000000	*3.3.38 Hyp Auxiliary Memory Attribute Indirection Register 1* on page 3-111

3.2.13 c11 registers

The following table shows the 32-bit system registers you can access when the value of CRn is c11.

Table 3-12 c11 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c11	0	c0	0	IMP_SLAVEPCTLR	RW	0x00000001	*3.3.91 Slave Port Control Register* on page 3-183

3.2.14 c12 registers

The following table shows the 32-bit system registers you can access when the value of CRn is c12.

See the *Arm® Generic Interrupt Controller Architecture Specification GIC architecture version 3.0 and version 4.0* for more information.

Table 3-13 c12 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c12	0	c0	0	VBAR	RW	0x00000000	*3.3.100 Vector Base Address Register* on page 3-194
			1	RVBAR	RO	-[p]	*3.3.89 Reset Vector Base Address Register* on page 3-181
		c1	0	ISR	RO	-[o]	*3.3.67 Interrupt Status Register* on page 3-155
		c8	0	ICC_IAR0	RO	0x000003FF	*Interrupt Controller Interrupt Acknowledge Register 0* on page 9-314
				ICV_IAR0			*Interrupt Controller Virtual Interrupt Acknowledge Register 0* on page 9-333
			1	ICC_EOIR0	WO	UNK	*Interrupt Controller End Of Interrupt Register 0* on page 9-316
				ICV_EOIR0			*Interrupt Controller Virtual End Of Interrupt Register 0* on page 9-335
			2	ICC_HPPIR0	RO	0x000003FF	*Interrupt Controller Highest Priority Pending Interrupt Register 0* on page 9-317
				ICV_HPPIR0			*Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0* on page 9-336
			3	ICC_BPR0	RW	0x00000002	*Interrupt Controller Binary Point Register 0* on page 9-318
				ICV_BPR0		0x00000002	*Interrupt Controller Virtual Binary Point Register 0* on page 9-337
			4	ICC_AP0R0	RW	0x00000000	*Interrupt Controller Active Priorities Group 0 Register* on page 9-330
				ICV_AP0R0			*Interrupt Controller Virtual Active Priorities Group 0 Register* on page 9-344
		c9	0	ICC_AP1R0	RW	0x00000000	*Interrupt Controller Active Priorities Group 1 Register* on page 9-331
				ICV_AP1R0			*Interrupt Controller Virtual Active Priorities Group 1 Register* on page 9-344
		c11	1	ICC_DIR	WO	UNK	*Interrupt Controller Deactivate Interrupt Register* on page 9-320
				ICV_DIR			*Interrupt Controller Deactivate Virtual Interrupt Register* on page 9-339
			3	ICC_RPR	RO	0x000000FF	*Interrupt Controller Running Priority Register* on page 9-322
				ICV_RPR			*Interrupt Controller Virtual Running Priority Register* on page 9-341

[o] The reset value depends on if there are interrupts pending.

Table 3-13 c12 register summary (continued)

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c12	0	c12	0	ICC_IAR1	RO	0x000003FF	*Interrupt Controller Interrupt Acknowledge Register 1* on page 9-315
				ICV_IAR1			*Interrupt Controller Virtual Interrupt Acknowledge Register 1* on page 9-334
			1	ICC_EOIR1	WO	UNK	*Interrupt Controller End Of Interrupt Register 1* on page 9-316
				ICV_EOIR1			*Interrupt Controller Virtual End Of Interrupt Register 1* on page 9-335
			2	ICC_HPPIR1	RO	0x000003FF	*Interrupt Controller Highest Priority Pending Interrupt Register 1* on page 9-318
				ICV_HPPIR1			*Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1* on page 9-337
			3	ICC_BPR1	RW	0x00000003	*Interrupt Controller Binary Point Register 1* on page 9-319
				ICV_BPR1		0x00000003	*Interrupt Controller Virtual Binary Point Register 1* on page 9-338
			4	ICC_CTLR	RW	0x00000400	*Interrupt Controller Control Register (EL1)* on page 9-322
				ICV_CTLR		0x00000400	*Interrupt Controller Virtual Control Register* on page 9-341
			5	ICC_SRE	RO	0x00000007	*Interrupt Controller System Register Enable Register (EL1)* on page 9-323
			6	ICC_IGRPEN0	RW	0x00000000	*Interrupt Controller Interrupt Group 0 Enable Register* on page 9-325
				ICV_IGRPEN0			*Interrupt Controller Virtual Interrupt Group 0 Enable Register* on page 9-342
			7	ICC_IGRPEN1	RW	0x00000000	*Interrupt Controller Interrupt Group 1 Enable Register* on page 9-326
				ICV_IGRPEN1			*Interrupt Controller Virtual Interrupt Group 1 Enable Register* on page 9-343
	4	c0	0	HVBAR	RW	-[p]	*3.3.55 Hyp Vector Base Address Register* on page 3-139.
			2	HRMR	RW	0x00000000	*3.3.56 Hypervisor Reset Management Register* on page 3-140
		c8	0	ICH_AP0R0	RW	0x00000000	*Interrupt Controller Hyp Active Priorities Group 0 Register 0* on page 9-312

Table 3-13 c12 register summary (continued)

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c12	4	c9	0	ICH_AP1R0	RW	0x00000000	*Interrupt Controller Hyp Active Priorities Group 1 Register 0* on page 9-312
			5	ICC_HSRE	RO	0x0000000F	*Interrupt Controller System Register Enable Register (EL2)* on page 9-324
		c11	0	ICH_HCR	RW	0x00000000	*Interrupt Controller Hyp Control Register* on page 9-300
			1	ICH_VTR	RO	0x90180003	*Interrupt Controller Hyp Control VGIC Type Register* on page 9-303
			2	ICH_MISR	RO	0x00000000	*Interrupt Controller Hyp Maintenance Interrupt Status Register EL2* on page 9-304
			3	ICH_EISR	RO	0x00000000	*Interrupt Controller End of Interrupt Status Register* on page 9-306
			5	ICH_ELRSR	RO	0x0000000F	*Interrupt Controller Empty List Register Status Register* on page 9-307
			7	ICH_VMCR	RW	0x004C0008	*Interrupt Controller Virtual Machine Control Register* on page 9-308
		c12	0	ICH_LR0	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-309
			1	ICH_LR1	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-309
			2	ICH_LR2	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-309
			3	ICH_LR3	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-309
		c14	0	ICH_LRC0	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-310
			1	ICH_LRC1	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-310
			2	ICH_LRC2	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-310
			3	ICH_LRC3	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-310

3.2.15 c13 registers

The following table shows the 32-bit wide system registers you can access when the value of CRn is c13.

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

[p] **CFGVECTABLEx[31:5]** determines the vector base address out of reset, see register description.

Table 3-14 c13 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c13	0	c0	0	FCSEIDR	RW	0x00000000	*3.3.28 FCSE Process ID Register* on page 3-102
			1	CONTEXTIDR	RW	0x00000000	*3.3.18 Context ID Register* on page 3-89
			2	TPIDRURW	RW	UNK	*3.3.25 EL0 Read/Write Software Thread ID Register* on page 3-100
			3	TPIDRURO	RW	UNK	*3.3.26 EL0 Read-Only Software Thread ID Register* on page 3-101
			4	TPIDRPRW	RW	UNK	*3.3.27 EL1 Software Thread ID Register* on page 3-102
	4	c0	2	HTPIDR	RW	UNK	*3.3.51 Hyp Software Thread ID Register* on page 3-130

3.2.16 c14 registers

The following table shows the 32-bit system registers when the value of CRn is c14.

With the exception of the event selection values used by PMEVCNTR*, these registers are fully documented in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*. This implies that there are no Cortex-R52-specific details.

Table 3-15 c14 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c14	0	c0	0	CNTFRQ	-[q]	0x00000000	Counter-timer Frequency Register
		c1	0	CNTKCTL	RW	0x00000000	Counter-timer Kernel Control Register
		c2	0	CNTP_TVAL	RW	UNK	Counter-timer Physical Timer RimerValue Register
			1	CNTP_CTL	RW	-	Counter-timer Physical Timer Control Register
		c3	0	CNTV_TVAL	RW	UNK	Counter-timer Virtual Timer TimerValue Register
			1	CNTV_CTL	RW	-[r]	Counter-timer Virtual Timer Control Register
		c8	0	PMEVCNTR0	RW	UNK	Performance Monitors Event Count Registers
			1	PMEVCNTR1	RW	UNK	
			2	PMEVCNTR2	RW	UNK	
			3	PMEVCNTR3	RW	UNK	
		c12	0	PMEVTYPER0	RW	UNK	Performance Monitors Event Type Registers
			1	PMEVTYPER1	RW	UNK	
			2	PMEVTYPER2	RW	UNK	
			3	PMEVTYPER3	RW	UNK	
		c15	7	PMCCFILTR	RW	0x00000000	Performance Monitors Cycle Count Filter Register
	4	c1	0	CNTHCTL	RW	-[s]	Counter-timer Hyp Control Register
		c2	0	CNTHP_TVAL	RW	UNK	Counter-timer Physical Timer TimerValue Register
			1	CNTHP_CTL	RW	-[r]	Counter-timer Hyp Physical Timer Control Register (EL2)

Related references

Chapter 12 Performance Monitor Unit on page 12-398

3.2.17 c15 registers

The following table shows the 32-bit system registers you can access when the value of CRn is c15.

[q] CNTKCTL[0] or CNTKCTL[1] determine user read accessibility of CNTFRQ.
[r] The reset value for bit[0] is 0.
[s] The reset value for bit[2] is 0 and for bits[1:0] is 0b11.

Table 3-16 c15 register summary

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c15	0	c0	0	IMP_PERIPHPREGIONR	RW	- [t]	*3.3.80 Peripheral Port Region Register* on page 3-169
			1	IMP_FLASHIFREGIONR	RW	- [u]	*3.3.30 Flash Interface Region Register* on page 3-104
		c2	0	IMP_BUILDOPTR	RO	-[v]	*3.3.11 Build Options Register* on page 3-81
			7	IMP_PINOPTR	RO	-[w]	*3.3.82 Pin Options Register* on page 3-173
	1	c3	0	IMP_CBAR	RO	-[x]	*3.3.17 Configuration Base Address Register* on page 3-88
			1	IMP_QOSR	RW	0x00000000	*3.3.88 Quality Of Service Register* on page 3-180
			2	IMP_BUSTIMEOUTR	RW	0x00000000	*3.3.12 Bus Timeout Register* on page 3-82
			4	IMP_INTMONR	RW	0x00000000	*3.3.66 Interrupt Monitoring Register* on page 3-154
		c14	0	IMP_CDBGDCI	WO	UNK	*3.3.68 Invalidate All Register* on page 3-156
	2	c0	0	IMP_ICERR0	RW	0x00000000	*3.3.57 Instruction Cache Error Record Registers 0 and 1* on page 3-141
			1	IMP_ICERR1	RW	0x00000000	*3.3.57 Instruction Cache Error Record Registers 0 and 1* on page 3-141
		c1	0	IMP_DCERR0	RW	0x00000000	*3.3.21 Data Cache Error Record Registers 0 and 1* on page 3-95
			1	IMP_DCERR1	RW	0x00000000	*3.3.21 Data Cache Error Record Registers 0 and 1* on page 3-95
		c2	0	IMP_TCMERR0	RW	0x00000000	*3.3.93 TCM Error Record Register 0 and 1* on page 3-188
			1	IMP_TCMERR1	RW	0x00000000	*3.3.93 TCM Error Record Register 0 and 1* on page 3-188
			2	IMP_TCMSYNDR0	RO	0x00000000	*3.3.95 TCM Syndrome Register 0 and 1* on page 3-190
			3	IMP_TCMSYNDR1	RO	0x00000000	*3.3.95 TCM Syndrome Register 0 and 1* on page 3-190
		c3	0	IMP_FLASHERR0	RW	0x00000000	*3.3.29 Flash Error Record Registers 0 and 1* on page 3-102
			1	IMP_FLASHERR1	RW	0x00000000	*3.3.29 Flash Error Record Registers 0 and 1* on page 3-102

[t] The reset value depends on **CFGLLPPBASEADDR[31:12]** and **CFGLLPPSIZE[3:0]**. See register description.
[u] The reset value depends on **CFGFLASHBASEADDR[31:27]**. See register description.
[v] The reset value depends on hardware configuration. See *3.3.11 Build Options Register* on page 3-81.
[w] The reset value depends on hardware configuration. See *3.3.82 Pin Options Register* on page 3-173.
[x] The reset value for bits [31:21] is the value of the **CFGPERIPHBASE** signal. The reset value for bits [20:0] is 0b00000000000000000000.

Table 3-16 c15 register summary (continued)

CRn	Op1	CRm	Op2	Name	Access	Reset	Description
c15	3	c0	0	IMP_CDBGDR0	RO	UNK	Cache Debug Data Register 0. See *7.4 Direct access to internal memory* on page 7-225 for more information.
			1	IMP_CDBGDR1	RO	UNK	Cache Debug Data Register 1. See *7.4 Direct access to internal memory* on page 7-225 for more information.
		c2	0	IMP_CDBGDCT	WO	UNK	Data Cache Tag Read Operation. See *7.4 Direct access to internal memory* on page 7-225 for more information.
			1	IMP_CDBGICT	WO	UNK	Instruction Cache Tag Read Operation. See *7.4 Direct access to internal memory* on page 7-225 for more information.
		c4	0	IMP_CDBGDCD	WO	UNK	Data Cache Data Read Operation. See *7.4 Direct access to internal memory* on page 7-225 for more information.
			1	IMP_CDBGICD	WO	UNK	Instruction Cache Data Read Operation. See *7.4 Direct access to internal memory* on page 7-225 for more information.
c15	4	c0	0	IMP_TESTR0	RO	-[y]	*3.3.97 Test Register 0* on page 3-193
			1	IMP_TESTR1	WO	-	*3.3.98 Test Register 1* on page 3-194

3.2.18 64-bit registers

The following table shows the 64-bit non-debug system registers (coproc==`0b1111`), accessed by the `MCRR` and `MRRC` instructions.

The Performance Monitor and Counter-timer registers are fully documented in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*. This implies that there are no Cortex-R52-specific details.

Table 3-17 64-bit register summary

Op1	CRm	Name	Access	Reset	Description
0	c7	PAR	RW	UNK	*3.3.81 Physical Address Register* on page 3-170
	c9	PMCCNTR	RW	UNK	Performance Monitors Cycle Count Register
	c12	ICC_SGI1R	WO	UNK	*Interrupt Controller Software Generated Interrupt Group 1 Register* on page 9-328
	c14	CNTPCT	RO	UNK	Counter-timer Physical Count Register
	c15	CPUACTLR		`0x8A800`	*3.3.19 CPU Auxiliary Control Register* on page 3-90
1	c12	ICC_ASGI1R	WO	UNK	*Interrupt Controller Alias Software Generated Interrupt Group 1 Register* on page 9-329
	c14	CNTVCT	RO	UNK	Counter-timer Virtual Count Register

[y] The reset value depends on the current interrupts. See *3.3.97 Test Register 0* on page 3-193.

Table 3-17 64-bit register summary (continued)

Op1	CRm	Name	Access	Reset	Description
2	c12	ICC_SGI0R	WO	UNK	*Interrupt Controller Software Generated Interrupt Group 0 Register* on page 9-326
	c14	CNTP_CVAL	RW	UNK	Counter-timer Physical Timer CompareValue Register
3	c14	CNTV_CVAL	RW	UNK	Counter-timer Virtual Timer CompareValue Register
4	c14	CNTVOFF	RW	UNK	Counter-timer Virtual Offset Register
6	c14	CNTHP_CVAL	RW	UNK	Counter-timer Hyp Physical CompareValue Register

3.2.19 AArch32 Identification registers

The following table shows the identification registers.

Table 3-18 Identification registers

Name	CRn	Op1	CRm	Op2	Reset	Access	Description
MIDR	c0	0	c0	0	0x411FD132	RO	*3.3.69 Main ID Register* on page 3-156
CTR				1	0x8144c004	RO	*3.3.16 Cache Type Register* on page 3-87
TCMTR				2	-[z]	RO	*3.3.96 TCM Type Register* on page 3-192
TLBTR				3	0x00000000	RO	*3.3.99 TLB Type Register* on page 3-194
MPIDR				5	-[aa]	RO	*3.3.78 Multiprocessor Affinity Register* on page 3-167
REVIDR				6	0x00000000	RO	*3.3.90 Revision ID Register* on page 3-182
MIDR				7	0x411FD132	RO	Alias of the Main ID Register, *3.3.69 Main ID Register* on page 3-156
ID_PFR0	c0	0	c1	0	0x00000131	RO	*3.3.83 Processor Feature Register 0* on page 3-174
ID_PFR1				1	0x10111001	RO	*3.3.84 Processor Feature Register 1* on page 3-175
ID_DFR0				2	0x03010006	RO	*3.3.24 Debug Feature Register 0* on page 3-99
ID_AFR0				3	0x00000000	RO	*3.3.5 Auxiliary Feature Register 0* on page 3-78
ID_MMFR0	c0	0	c1	4	0x00211040	RO	*3.3.71 Memory Model Feature Register 0* on page 3-159
ID_MMFR1				5	0x40000000	RO	*3.3.72 Memory Model Feature Register 1* on page 3-160
ID_MMFR2				6	0x01200000	RO	*3.3.73 Memory Model Feature Register 2* on page 3-161
ID_MMFR3				7	0xF0102211	RO	*3.3.74 Memory Model Feature Register 3* on page 3-163

[z] The reset value depends on how many TCMs are implemented. See *3.3.96 TCM Type Register* on page 3-192.
[aa] The reset value depends on the cluster and processor ID. See *3.3.78 Multiprocessor Affinity Register* on page 3-167.

Table 3-18 Identification registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Access	Description
ID_ISAR0	c0	0	c2	0	0x02101110	RO	*3.3.60 Instruction Set Attribute Register 0* on page 3-144
ID_ISAR1				1	0x13112111	RO	*3.3.61 Instruction Set Attribute Register 1* on page 3-146
ID_ISAR2				2	0x21232142	RO	*3.3.62 Instruction Set Attribute Register 2* on page 3-147
ID_ISAR3				3	0x01112131	RO	*3.3.63 Instruction Set Attribute Register 3* on page 3-149
ID_ISAR4				4	0x00010142	RO	*3.3.64 Instruction Set Attribute Register 4* on page 3-151
ID_ISAR5				5	0x00010001	RO	*3.3.65 Instruction Set Attribute Register 5* on page 3-152
ID_MMFR4				6	0x00000010	RO	*3.3.75 Memory Model Feature Register 4* on page 3-164
CCSIDR	c0	1	c0	0	UNK	RO	*3.3.20 Current Cache Size ID Register* on page 3-93
CLIDR				1	-[ab]	RO	*3.3.13 Cache Level ID Register* on page 3-84
AIDR				7	0x00000000	RO	*3.3.6 Auxiliary ID Register* on page 3-78
CSSELR	c0	2	c0	0	UNK	RW	*3.3.15 Cache Size Selection Register* on page 3-86
VPIDR	c0	4	c0	0	-[ac]	RW	*3.3.102 Virtualization Processor ID Register* on page 3-196
VMPIDR				5	-[ad]	RW	*3.3.101 Virtualization Multiprocessor ID Register* on page 3-195

3.2.20 AArch32 Memory control registers

The following table shows the memory control registers.

[ab] The reset value depends on if the Instruction and Data caches are implemented. See register description.
[ac] The reset value is the value of the Main ID Register.
[ad] The reset value is the value of the Multiprocessor Affinity Register.

Table 3-19 Virtual memory control registers

Name	CRn	Op1	CRm	Op2	Reset	Access	Width	Description
MAIR0	c10	0	c2	0	0x00098AA4	RW	32-bit	*3.3.70 Memory Attribute Indirection Registers 0 and 1* on page 3-157
MAIR1				1	0x44E048E0	RW	32-bit	*3.3.70 Memory Attribute Indirection Registers 0 and 1* on page 3-157
AMAIR0			c3	0	0x00000000	RW	32-bit	*3.3.8 Auxiliary Memory Attribute Indirection Register 0* on page 3-80
AMAIR1				1	0x00000000	RW	32-bit	*3.3.9 Auxiliary Memory Attribute Indirection Register 1* on page 3-80
HMAIR0		4	c2	0	0x00000000	RW	32-bit	*3.3.45 Hyp Memory Attribute Indirection Register 0 and 1* on page 3-122
HMAIR1				1	0x00000000	RW	32-bit	*3.3.45 Hyp Memory Attribute Indirection Register 0 and 1* on page 3-122
HAMAIR0			c3	0	0x00000000	RW	32-bit	*3.3.37 Hyp Auxiliary Memory Attribute Indirection Register 0* on page 3-111
HAMAIR1				1	0x00000000	RW	32-bit	*3.3.38 Hyp Auxiliary Memory Attribute Indirection Register 1* on page 3-111
CONTEXTIDR	c13	0	c0	1	0x00000000	RW	32-bit	*3.3.18 Context ID Register* on page 3-89

3.2.21 AArch32 Exception and fault handling registers

The following table shows the fault handling registers.

Table 3-20 Fault handling registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
DFSR	c5	0	c0	0	RW	UNK	*3.3.23 Data Fault Status Register* on page 3-97
IFSR				1	RW	UNK	*3.3.59 Instruction Fault Status Register* on page 3-143
ADFSR			c1	0	RW	UNK	*3.3.4 Auxiliary Data Fault Status Register* on page 3-77
AIFSR				1	RW	UNK	*3.3.7 Auxiliary Instruction Fault Status Register* on page 3-78
HADFSR		4	c1	0	RW	UNK	*3.3.35 Hyp Auxiliary Data Fault Status Register* on page 3-108
HAIFSR				1	RW	UNK	*3.3.36 Hyp Auxiliary Instruction Fault Status Register* on page 3-110
HSR			c2	0	RW	UNK	*3.3.52 Hyp Syndrome Register* on page 3-131
DFAR	c6	0	c0	0	RW	UNK	*3.3.22 Data Fault Address Register* on page 3-96
IFAR				2	RW	UNK	*3.3.58 Instruction Fault Address Register* on page 3-142
HDFAR		4	c0	0	RW	UNK	*3.3.41 Hyp Data Fault Address Register* on page 3-116
HIFAR				2	RW	UNK	*3.3.43 Hyp Instruction Fault Address Register* on page 3-120
HPFAR				4	RW	UNK	*3.3.44 Hyp IPA Fault Address Register* on page 3-121

Table 3-20 Fault handling registers (continued)

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
VBAR	c12	0	c0	0	RW	0x00000000	*3.3.100 Vector Base Address Register* on page 3-194
ISR			c1	0	RO	-[ae]	*3.3.67 Interrupt Status Register* on page 3-155
HVBAR		4	c0	0	RW	-[af]	*3.3.55 Hyp Vector Base Address Register* on page 3-139.

The Virtualization registers include additional fault handling registers. For more information, see *3.2.26 AArch32 Virtualization registers* on page 3-60.

3.2.22 AArch32 Other system control registers

The following table shows the other system control registers.

Table 3-21 Other system control registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
SCTLR	c1	0	c0	0	RW	-[ag]	*3.3.92 System Control Register* on page 3-184
ACTLR				1	RW	0x00000000	*3.3.2 Auxiliary Control Register* on page 3-77
CPACR				2	RW	0x00000000	*3.3.1 Architectural Feature Access Control Register* on page 3-76
ACTLR2				3	RW	0x00000000	*3.3.3 Auxiliary Control Register 2* on page 3-77
HSCTLR		4	c0	0	RW	-[ag]	*3.3.53 Hyp System Control Register* on page 3-133
HACTLR				1	RW	0x00000000	*3.3.33 Hyp Auxiliary Control Register* on page 3-107
HACTLR2				3	RW	0x00000000	*3.3.34 Hyp Auxiliary Control Register 2* on page 3-108

3.2.23 AArch32 Address registers

The following table shows the address translation register and operations.

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

Table 3-22 Address translation operations

Name	CRn	Op1	CRm	Op2	Access	Reset	Width	Description
PAR	c7	0	c4	0	RW	UNK	32-bit	*3.3.81 Physical Address Register* on page 3-170
	-	0	c7	-	RW		64-bit	

[ae] The reset value depends on if there are interrupts pending.
[af] **CFGVECTABLEx[31:5]** determines the vector base address out of reset, see register description.
[ag] The reset value depends on inputs, **CFGTHUMBEXCEPTION** and **CFGENDIANESS**. The value shown in *Table 3-2 c0 register summary* on page 3-39 assumes that these signals are set to LOW.

Table 3-22 Address translation operations (continued)

Name	CRn	Op1	CRm	Op2	Access	Reset	Width	Description
ATS1CPR	c7	0	c8	0	WO	-	32-bit	Address Translate Stage 1 current state EL1 read
ATS1CPW				1	WO	-	32-bit	Address Translate Stage 1 current state EL1 write
ATS1CUR				2	WO	-	32-bit	Address Translate Stage 1 current state unprivileged read
ATS1CUW				3	WO	-	32-bit	Address Translate Stage 1 current state unprivileged write
ATS12NSOPR				4	WO	-	32-bit	Address Translate Stages 1 and 2 Non-secure only EL1 read
ATS12NSOPW				5	WO	-	32-bit	Address Translate Stages 1 and 2 Non-secure only EL1 write
ATS12NSOUR				6	WO	-	32-bit	Address Translate Stages 1 and 2 Non-secure only unprivileged read
ATS12NSOUW				7	WO	-	32-bit	Address Translate Stages 1 and 2 Non-secure only unprivileged write
ATS1HR		4	c8	0	WO	-	32-bit	Address Translate Stage 1 Hyp mode read
ATS1HW				1	WO	-	32-bit	Address Translate Stage 1 Hyp mode write

3.2.24 AArch32 Thread registers

The following table shows the miscellaneous operations.

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

Table 3-23 Miscellaneous System instructions

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
TPIDRURW	c13	0	c0	2	RW	UNK	*3.3.25 EL0 Read/Write Software Thread ID Register* on page 3-100
TPIDRURO				3	RW	UNK	*3.3.26 EL0 Read-Only Software Thread ID Register* on page 3-101
TPIDRPRW				4	RW	UNK	*3.3.27 EL1 Software Thread ID Register* on page 3-102
HTPIDR		4	c0	2	RW	UNK	*3.3.51 Hyp Software Thread ID Register* on page 3-130

3.2.25 AArch32 Performance monitor registers

Information about the performance monitor registers can be found in the PMU register summary.

See *Table 12-1 Performance monitor registers* on page 12-401.

3.2.26 AArch32 Virtualization registers

The following table shows the Virtualization registers.

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

Table 3-24 Virtualization registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Width	Description
VPIDR	c0	4	c0	0	RW	-[ah]	32-bit	*3.3.102 Virtualization Processor ID Register* on page 3-196
VMPIDR				5	RW	-[ai]	32-bit	*3.3.101 Virtualization Multiprocessor ID Register* on page 3-195
HSCTLR	c1	4	c0	0	RW	-[aj]	32-bit	*3.3.53 Hyp System Control Register* on page 3-133
HACTLR				1	RW	0x00000000		*3.3.33 Hyp Auxiliary Control Register* on page 3-107
HACTLR2				3	RW	0x00000000	32-bit	*3.3.34 Hyp Auxiliary Control Register 2* on page 3-108
HCR			c1	0	RW	0x00000002	32-bit	*3.3.39 Hyp Configuration Register* on page 3-111
HDCR				1	RW	0x00000004	32-bit	*3.3.42 Hyp Debug Control Register* on page 3-117
HCPTR				2	RW	-[ak]	32-bit	*3.3.31 Hyp Architectural Feature Trap Register* on page 3-105
HSTR				3	RW	0x00000000	32-bit	*3.3.54 Hyp System Trap Register* on page 3-137
HCR2				4	RW	0x00000000	32-bit	*3.3.40 Hyp Configuration Register 2* on page 3-116
HACR				7	RW	0x00000000	32-bit	*3.3.32 Hyp Auxiliary Configuration Register* on page 3-106
VSCTLR	c2	4	c0	0	RW	0x00000000	32-bit	*3.3.103 Virtualization System Control Register* on page 3-196
HADFSR	c5	4	c1	0	RW	UNK	32-bit	*3.3.35 Hyp Auxiliary Data Fault Status Register* on page 3-108
HAIFSR				1	RW	UNK	32-bit	*3.3.36 Hyp Auxiliary Instruction Fault Status Register* on page 3-110
HSR			c2	0	RW	UNK	32-bit	*3.3.52 Hyp Syndrome Register* on page 3-131
HDFAR	c6	4	c0	0	RW	UNK	32-bit	*3.3.41 Hyp Data Fault Address Register* on page 3-116
HIFAR				2	RW	UNK	32-bit	*3.3.43 Hyp Instruction Fault Address Register* on page 3-120
HPFAR				4	RW	UNK	32-bit	*3.3.44 Hyp IPA Fault Address Register* on page 3-121

[ah] The reset value is the value of the Main ID Register.
[ai] The reset value is the value of the Multiprocessor Affinity Register.
[aj] The reset value depends on **CFGENDIANESSx** and **CFGTHUMBEXCEPTIONSx**.
[ak] The reset value depends on the floating-point and Advanced SIMD configuration.

Table 3-24 Virtualization registers (continued)

Name	CRn	Op1	CRm	Op2	Access	Reset	Width	Description
HMAIR0	c10	4	c2	0	RW	0x00000000	32-bit	*3.3.45 Hyp Memory Attribute Indirection Register 0 and 1* on page 3-122
HMAIR1				1	RW	0x00000000	32-bit	*3.3.45 Hyp Memory Attribute Indirection Register 0 and 1* on page 3-122
HAMAIR0			c3	0	RW	0x00000000	32-bit	*3.3.37 Hyp Auxiliary Memory Attribute Indirection Register 0* on page 3-111
HAMAIR1				1	RW	0x00000000	32-bit	*3.3.38 Hyp Auxiliary Memory Attribute Indirection Register 1* on page 3-111
HVBAR	c12	4	c0	0	RW	-[an]	32-bit	*3.3.55 Hyp Vector Base Address Register* on page 3-139
HRMR				2	RW	0x00000000	32-bit	*3.3.56 Hypervisor Reset Management Register* on page 3-140
ICH_AP0R0			c8	0	RW	0x00000000	32-bit	*Interrupt Controller Hyp Active Priorities Group 0 Register 0* on page 9-312
ICH_AP1R0	c12	4	c9	0	RW	0x00000000	32-bit	*Interrupt Controller Hyp Active Priorities Group 1 Register 0* on page 9-312
ICC_HSRE				5	RW	0x0000000F	32-bit	*Interrupt Controller System Register Enable Register (EL2)* on page 9-324
ICH_HCR			c11	0	RW	0x00000000	32-bit	*Interrupt Controller Hyp Control Register* on page 9-300
ICH_VTR				1	RO	0x90180003	32-bit	*Interrupt Controller Hyp Control VGIC Type Register* on page 9-303
ICH_MISR				2	RO	0x00000000	32-bit	*Interrupt Controller Hyp Maintenance Interrupt Status Register EL2* on page 9-304
ICH_EISR				3	RO	0x00000000	32-bit	*Interrupt Controller End of Interrupt Status Register* on page 9-306
ICH_ELRSR				5	RO	0x0000000F	32-bit	*Interrupt Controller Empty List Register Status Register* on page 9-307
ICH_VMCR				7	RW	0x004C0008	32-bit	*Interrupt Controller Virtual Machine Control Register* on page 9-308
ICH_LR0			c12	0	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LR1				1	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LR2				2	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LR3	c12	4	c12	3	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LRC0			c14	0	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-310
ICH_LRC1				1	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-310
ICH_LRC2				2	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-310
ICH_LRC3				3	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-310
HTPIDR	c13	4	c0	2	RW	UNK	32-bit	*3.3.51 Hyp Software Thread ID Register* on page 3-130

Table 3-24 Virtualization registers (continued)

Name	CRn	Op1	CRm	Op2	Access	Reset	Width	Description
CNTHCTL	c14	4	c1	0	RW	-[al]	32-bit	Counter-timer Hyp Control Register
CNTHP_TVAL			c2	0	RW	UNK	32-bit	Counter-timer Physical Timer TimerValue Register
CNTHP_CTL				1	RW	-[am]	32-bit	Counter-timer Hyp Physical Timer Control Register (EL2)
CNTVOFF	-	4	c14	-	RW	UNK	64-bit	Counter-timer Virtual Offset Register
CNTHP_CVAL		6			RW	UNK	64-bit	Counter-timer Hyp Physical CompareValue Register

The following table shows Hyp mode (coproc==`0b1111`) operations that are part of this functional group.

Table 3-25 Hyp mode (coproc==0b1111) operations

Name	CRn	Op1	CRm	Op2	Access	Reset	Width	Description
ATS1HR	c7	4	c8	0	WO	-	32-bit	Address Translate Stage 1 Hyp mode read
ATS1HW				1	WO	-	32-bit	Address Translate Stage 1 Hyp mode write

3.2.27 AArch32 GIC system registers

The following table shows the GIC system registers.

Table 3-26 GIC system registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Width	Description
ICC_SGI1R	-	0	c12	-	WO	UNK	64-bit	*Interrupt Controller System Register Enable Register (EL1)* on page 9-323
ICC_ASGI1R		1	c12	-	WO	UNK	64-bit	*Interrupt Controller Alias Software Generated Interrupt Group 1 Register* on page 9-329
ICC_SGI0R		2	c12	-	WO	UNK	64-bit	*Interrupt Controller Software Generated Interrupt Group 0 Register* on page 9-326
ICC_PMR	c4	0	c6	0	RW	0x00000000	32-bit	*Interrupt Controller Interrupt Priority Mask Register* on page 9-321
ICV_PMR								*Interrupt Controller Virtual Interrupt Priority Mask Register* on page 9-340

[al] The reset value for bit[2] is `0` and for bits[1:0] is `0b11`.
[am] The reset value for bit[0] is `0`.
[an] **CFGVECTABLEx[31:5]** determines the vector base address out of reset, see register description.

Table 3-26 GIC system registers (continued)

Name	CRn	Op1	CRm	Op2	Access	Reset	Width	Description
ICC_IAR0	c12	0	c8	0	RO	0x000003FF	32-bit	*Interrupt Controller Interrupt Acknowledge Register 0* on page 9-314
ICV_IAR0								*Interrupt Controller Virtual Interrupt Acknowledge Register 0* on page 9-333
ICC_EOIR0				1	WO	UNK	32-bit	*Interrupt Controller End Of Interrupt Register 0* on page 9-316
ICV_EOIR0								*Interrupt Controller Virtual End Of Interrupt Register 0* on page 9-335

Table 3-26 GIC system registers (continued)

Name	CRn	Op1	CRm	Op2	Access	Reset	Width	Description
ICC_HPPIR0	c12	0	c8	2	RO	0x000003FF	32-bit	*Interrupt Controller Highest Priority Pending Interrupt Register 0* on page 9-317
ICV_HPPIR0								*Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0* on page 9-336
ICC_BPR0				3	RW	0x00000003	32-bit	*Interrupt Controller Binary Point Register 0* on page 9-318
ICV_BPR0						0x00000002		*Interrupt Controller Virtual Binary Point Register 0* on page 9-337
ICC_AP0R0				4	RW	0x00000000	32-bit	*Interrupt Controller Active Priorities Group 0 Register* on page 9-330
ICV_AP0R0								*Interrupt Controller Virtual Active Priorities Group 0 Register* on page 9-344
ICC_AP1R0			c9	0	RW	0x00000000	32-bit	*Interrupt Controller Active Priorities Group 1 Register* on page 9-331
ICV_AP1R0								*Interrupt Controller Virtual Active Priorities Group 1 Register* on page 9-344
ICC_DIR			c11	1	WO	UNK	32-bit	*Interrupt Controller Deactivate Interrupt Register* on page 9-320
ICV_DIR								*Interrupt Controller Deactivate Virtual Interrupt Register* on page 9-339
ICC_RPR				3	RO	0x000000FF	32-bit	*Interrupt Controller Running Priority Register* on page 9-322
ICV_RPR								*Interrupt Controller Virtual Running Priority Register* on page 9-341
ICC_IAR1			c12	0	RO	0x000003FF	32-bit	*Interrupt Controller Interrupt Acknowledge Register 1* on page 9-315
ICV_IAR1								*Interrupt Controller Virtual Interrupt Acknowledge Register 1* on page 9-334
ICC_EOIR1				1	WO	UNK	32-bit	*Interrupt Controller End Of Interrupt Register 1* on page 9-316
ICV_EOIR1								*Interrupt Controller Virtual End Of Interrupt Register 1* on page 9-335
ICC_HPPIR1				2	RO	0x000003FF	32-bit	*Interrupt Controller Highest Priority Pending Interrupt Register 1* on page 9-318
ICV_HPPIR1								*Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1* on page 9-337
ICC_BPR1	c12	0	c12	3	RW	0x00000004	32-bit	*Interrupt Controller Binary Point Register 1* on page 9-319
ICV_BPR1						0x00000003		*Interrupt Controller Virtual Binary Point Register 1* on page 9-338

Table 3-26 GIC system registers (continued)

Name	CRn	Op1	CRm	Op2	Access	Reset	Width	Description
ICC_CTLR	c12	0	c12	4	RW	0x00000400	32-bit	*Interrupt Controller Control Register (EL1)* on page 9-322
ICV_CTLR						0x00000400		*Interrupt Controller Virtual Control Register* on page 9-341
ICC_SRE				5	RW	0x00000007	32-bit	*Interrupt Controller System Register Enable Register (EL1)* on page 9-323
ICC_IGRPEN0				6	RW	0x00000000	32-bit	*Interrupt Controller Interrupt Group 0 Enable Register* on page 9-325
ICV_IGRPEN0								*Interrupt Controller Virtual Interrupt Group 0 Enable Register* on page 9-342
ICC_IGRPEN1				7	RW	0x00000000	32-bit	*Interrupt Controller Interrupt Group 1 Enable Register* on page 9-326
ICV_IGRPEN1								*Interrupt Controller Virtual Interrupt Group 1 Enable Register* on page 9-343
ICC_HSRE		4	c9	5	RW	0x0000000F	32-bit	*Interrupt Controller System Register Enable Register (EL2)* on page 9-324
ICH_AP0R0		4	c8	0	RW	0x00000000	32-bit	*Interrupt Controller Hyp Active Priorities Group 0 Register 0* on page 9-312
ICH_AP1R0			c9	0	RW	0x00000000	32-bit	*Interrupt Controller Hyp Active Priorities Group 1 Register 0* on page 9-312
ICH_HCR			c11	0	RW	0x00000000	32-bit	*Interrupt Controller Hyp Control Register* on page 9-300
ICH_VTR				1	RO	0x90180003	32-bit	*Interrupt Controller Hyp Control VGIC Type Register* on page 9-303
ICH_MISR				2	RO	0x00000000	32-bit	*Interrupt Controller Hyp Maintenance Interrupt Status Register EL2* on page 9-304
ICH_EISR				3	RO	0x00000000	32-bit	*Interrupt Controller End of Interrupt Status Register* on page 9-306
ICH_ELRSR				5	RO	0x0000000F	32-bit	*Interrupt Controller Empty List Register Status Register* on page 9-307
ICH_VMCR				7	RW	0x004C0008	32-bit	*Interrupt Controller Virtual Machine Control Register* on page 9-308 EL2
ICH_LR0			c12	0	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LR1				1	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LR2				2	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LR3				3	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LRC0			c14	0	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-310
ICH_LRC1				1	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-310
ICH_LRC2				2	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-310
ICH_LRC3				3	RW	0x00000000	32-bit	*Interrupt Controller List Registers 0-3* on page 9-310

3.2.28 AArch32 Generic Timer registers

For more information on generic timer registers:

See *10.3.1 AArch32 Generic Timer register summary* on page 10-349.

3.2.29 AArch32 Implementation defined registers

These registers provide test features and any required configuration options specific to the Cortex-R52 processor.

In this section implementation defined is an architectural term that is used to indicate registers and register fields which are specific to the Cortex-R52 processor.

The following table shows 32-bit and 64-bit wide implementation defined registers.

Table 3-27 Implementation defined registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
CPUACTLR	-	0	c15	-	RW	0x8A800	*3.3.19 CPU Auxiliary Control Register* on page 3-90
MPUIR	c0	0	c0	4	RO	-[b]	*3.3.77 MPU Type Register* on page 3-166
AIDR		1	c0	7	RO	0x00000000	*3.3.6 Auxiliary ID Register* on page 3-78
HMPUIR		4	c0	4	RO	-[e]	*3.3.47 Hyp MPU Type Register* on page 3-125
ACTLR	c1	0	c0	1	RW	0x00000000	*3.3.2 Auxiliary Control Register* on page 3-77
ACTLR2				3	RW	0x00000000	*3.3.3 Auxiliary Control Register 2* on page 3-77
HACTLR		4	c0	1	RW	0x00000000	*3.3.33 Hyp Auxiliary Control Register* on page 3-107
HACTLR2				3	RW	0x00000000	*3.3.34 Hyp Auxiliary Control Register 2* on page 3-108
HACR			c1	7	RW	0x00000000	*3.3.32 Hyp Auxiliary Configuration Register* on page 3-106
ADFSR	c5	0	c1	0	RW	UNK	*3.3.4 Auxiliary Data Fault Status Register* on page 3-77
AIFSR				1	RW	UNK	*3.3.7 Auxiliary Instruction Fault Status Register* on page 3-78
HADFSR		4	c1	0	RW	UNK	*3.3.35 Hyp Auxiliary Data Fault Status Register* on page 3-108
HAIFSR				1	RW	UNK	*3.3.36 Hyp Auxiliary Instruction Fault Status Register* on page 3-110

Table 3-27 Implementation defined registers (continued)

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
IMP_ATCMREGIONR	c9	0	c1	0	RW	-[av]	*3.3.94 TCM Region Registers A, B, and C* on page 3-189
IMP_BTCMREGIONR				1	RW	-[av]	*3.3.94 TCM Region Registers A, B, and C* on page 3-189
IMP_CTCMREGIONR				2	RW	-[av]	*3.3.94 TCM Region Registers A, B, and C* on page 3-189
IMP_CSCTLR		1	c1	0	RW	0x00000000	*3.3.14 Cache Segregation Control Register* on page 3-85
IMP_BPCTLR				1	RW	0x00000000	*3.3.10 Branch Predictor Control Register* on page 3-80
IMP_MEMPROTCLR				2	RW	-[ao]	*3.3.76 Memory Protection Control Register* on page 3-165
AMAIR0	c10	0	c3	0	RW	0x00000000	*3.3.8 Auxiliary Memory Attribute Indirection Register 0* on page 3-80
AMAIR1				1	RW	0x00000000	*3.3.9 Auxiliary Memory Attribute Indirection Register 1* on page 3-80
HAMAIR0		4	c3	0	RW	0x00000000	*3.3.37 Hyp Auxiliary Memory Attribute Indirection Register 0* on page 3-111
HAMAIR1				1	RW	0x00000000	*3.3.38 Hyp Auxiliary Memory Attribute Indirection Register 1* on page 3-111
IMP_SLAVEPCTLR	c11	0	c0	0	RW	0x00000001	*3.3.91 Slave Port Control Register* on page 3-183

[ao] The reset value depends on the signals **CFGFLASHPROTEN** and **CFGRAMPROTEN**, see register description.

Table 3-27 Implementation defined registers (continued)

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
IMP_PERIPHPREGIONR	c15	0	c0	0	RW	- [ap]	*3.3.80 Peripheral Port Region Register* on page 3-169
IMP_FLASHIFREGIONR				1	RW	- [aq]	*3.3.30 Flash Interface Region Register* on page 3-104
IMP_BUILDOPTR			c2	0	RO	-[ar]	*3.3.11 Build Options Register* on page 3-81
IMP_PINOPTR				7	RO	-[as]	*3.3.82 Pin Options Register* on page 3-173
IMP_CBAR		1	c3	0	RO	- [at]	*3.3.17 Configuration Base Address Register* on page 3-88
IMP_QOSR				1	RW	0x00000000	*3.3.88 Quality Of Service Register* on page 3-180
IMP_BUSTIMEOUTR				2	RW	0x00000000	*3.3.12 Bus Timeout Register* on page 3-82
IMP_INTMONR				4	RW	0x00000000	*3.3.66 Interrupt Monitoring Register* on page 3-154
IMP_ICERR0		2	c0	0	RW	0x00000000	*3.3.57 Instruction Cache Error Record Registers 0 and 1* on page 3-141
IMP_ICERR1				1	RW	0x00000000	*3.3.57 Instruction Cache Error Record Registers 0 and 1* on page 3-141
IMP_DCERR0			c1	0	RW	0x00000000	*3.3.21 Data Cache Error Record Registers 0 and 1* on page 3-95
IMP_DCERR1				1	RW	0x00000000	*3.3.21 Data Cache Error Record Registers 0 and 1* on page 3-95
IMP_TCMERR0	c15		c2	0	RW	0x00000000	*3.3.93 TCM Error Record Register 0 and 1* on page 3-188
IMP_TCMERR1				1	RW	0x00000000	*3.3.93 TCM Error Record Register 0 and 1* on page 3-188
IMP_TCMSYNDR0				2	RO	0x00000000	*3.3.95 TCM Syndrome Register 0 and 1* on page 3-190
IMP_TCMSYNDR1				3	RO	0x00000000	*3.3.95 TCM Syndrome Register 0 and 1* on page 3-190
IMP_FLASHERR0			c3	0	RW	0x00000000	*3.3.29 Flash Error Record Registers 0 and 1* on page 3-102
IMP_FLASHERR1				1	RW	0x00000000	*3.3.29 Flash Error Record Registers 0 and 1* on page 3-102

[ap] The reset value depends on **CFGLLPPBASEADDR[31:12]** and **CFGLLPPSIZE[3:0]**. See register description.
[aq] The reset value depends on **CFGFLASHBASEADDR[31:27]**. See register description.
[ar] The reset value depends on hardware configuration. See *3.3.11 Build Options Register* on page 3-81.
[as] The reset value depends on hardware configuration. See *3.3.82 Pin Options Register* on page 3-173.
[at] The reset value for bits [31:21] is the value of the **CFGPERIPHBASE** signal. The reset value for bits [20:0] is `0b00000000000000000000`.

Table 3-27 Implementation defined registers (continued)

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
IMP_CDBGDR0	c15	3	c0	0	RO	UNK	Cache Debug Data Register 0. See *7.4 Direct access to internal memory* on page 7-225.
IMP_CDBGDR1				1	RO	UNK	Cache Debug Data Register 1. See *7.4 Direct access to internal memory* on page 7-225.
IMP_TESTR0		4	c0	0	RO	-[au]	*3.3.97 Test Register 0* on page 3-193
IMP_TESTR1				1	WO	-	This register is only for testing purposes.

3.2.30 AArch32 Implementation defined operations

These registers provide operations specific to the Cortex-R52 processor.

Table 3-28 Implementation defined registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
IMP_CDBGDCI	c15	1	c14	0	WO	UNK	*3.3.68 Invalidate All Register* on page 3-156.
IMP_CDBGDCT		3	c2	0	WO	UNK	Data Cache Tag Read Operation Register. See *7.4 Direct access to internal memory* on page 7-225.
IMP_CDBGICT				1	WO	UNK	Instruction Cache Tag Read Operation Register. See *7.4 Direct access to internal memory* on page 7-225.
IMP_CDBGDCD			c4	0	WO	UNK	Data Cache Data Read Operation Register. See *7.4 Direct access to internal memory* on page 7-225.
IMP_CDBGICD				1	WO	UNK	Instruction Cache Data Read Operation Register. See *7.4 Direct access to internal memory* on page 7-225.

3.2.31 AArch32 Debug registers

The following table shows 32-bit wide debug registers accessed with (coproc==`0b1111`).

Debug registers accessed with (coproc==`0b1110`) are listed in *11.3 System register summary* on page 11-356.

Table 3-29 Debug registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
HDCR	c1	4	c1	1	RW	`0x00000004`	*3.3.42 Hyp Debug Control Register* on page 3-117
DSPSR	c4	3	c5	0	RW	UNK	Debug Saved Program Status Register
DLR				1	RW	-[aw]	Debug Link Register

3.2.32 AArch32 Reset management registers

The following table shows 32-bit wide reset management registers.

au The reset value for bits [31:6] is `0b00000000000000000000000000`. The reset value of bits [5:0] gives the value of the current interrupts. See *3.3.97 Test Register 0* on page 3-193.
av The reset value depends on implementation (TCM size),and input pins. See register description.
aw The reset value is the value of the PC in Halting Debug state.

Table 3-30 Reset management registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
RVBAR	c12	0	c0	1	RO	-[ax]	*3.3.89 Reset Vector Base Address Register* on page 3-181
HRMR		4	c0	2	RW	0x00000000	*3.3.56 Hypervisor Reset Management Register* on page 3-140

3.2.33 AArch32 Legacy feature registers

The following table shows legacy feature registers and operations.

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

Table 3-31 Implementation defined registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
CP15ISB	c7	0	c5	4	WO	-	Instruction Synchronization Barrier operation, this operation is deprecated in Armv8-R
CP15DSB			c10	4	WO	-	Data Synchronization Barrier operation, this operation is deprecated in Armv8-R
CP15DMB				5	WO	-	Data Memory Barrier operation, this operation is deprecated in Armv8-R
FCSEIDR	c13	0	c0	0	RW	0x00000000	*3.3.28 FCSE Process ID Register* on page 3-102

3.2.34 AArch32 Cache maintenance instructions

The following table shows 32-bit wide cache maintenance instructions.

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

[ax] CFGVECTABLEx[31:5] determines the vector base address out of reset, see register description.

Table 3-32 Address translation operations

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
ICIALLUIS	c7	0	c1	0	WO	-	Invalidate all instruction caches Inner Shareable to *Point of Unification* (PoU).
BPIALLIS				6	WO	-	Invalidate all entries from branch predictors Inner Shareable
ICIALLU			c5	0	WO	-	Invalidate all Instruction Caches to PoU
ICIMVAU				1	WO	-	Invalidate Instruction Caches by VA to PoU
BPIALL				6	WO	-	Invalidate all entries from branch predictors
BPIMVA				7	WO	-	Invalidate VA from branch predictors
DCIMVAC			c6	1	WO	-	Invalidate data cache line by VA to PoC[ay]
DCISW				2	WO	-	Invalidate data cache line by set/way
DCCMVAC			c10	1	WO	-	Clean data cache line by VA to PoC
DCCSW				2	WO	-	Clean data cache line by set/way
DCCMVAU			c11	1	WO	-	Clean data cache line by VA to PoU
DCCIMVAC			c14	1	WO	-	Clean and invalidate data cache line by VA to PoC
DCCISW				2	WO	-	Clean and invalidate data cache line by set/way

3.2.35 AArch32 Security registers

The following table shows 32-bit wide security registers.

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

Table 3-33 Security registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
NSACR	c1	0	c1	2	RO	0x00000C00	*3.3.79 Non-Secure Access Control Register* on page 3-168

The Cortex-R52 processor does not support the ability to distinguish between secure and non-secure physical memories. The NSACR register is included, and assigned to a functional group, for completeness.

3.2.36 AArch 32 PMSA-specific registers

The following table shows 32-bit wide PMSA-specific registers.

[ay] *Point of Coherence* (PoC) is always outside of the processor and depends on the external memory system.

Table 3-34 PMSA-specific registers

Name	CRn	Op1	CRm	Op2	Access	Reset	Description
PRSELR	c6	0	c2	1	RW	UNK	*3.3.87 Protection Region Selection Register* on page 3-179
PRBAR			c3	0	RW	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
PRLAR				1	RW	UNK[az]	*3.3.86 Protection Region Limit Address Register* on page 3-178
PRBAR0-14 (even)			c8-15	0	RW	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
PRLAR0-14 (even)				1	RW	UNK[az]	*3.3.86 Protection Region Limit Address Register* on page 3-178
PRBAR1-15 (odd)				4	RW	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
PRLAR1-15 (odd)				5	RW	UNK[az]	*3.3.86 Protection Region Limit Address Register* on page 3-178
HPRENR		4	c1	1	RW	0x00000000	*3.3.46 Hyp MPU Region Enable Register* on page 3-124
HPRSELR			c2	1	RW	UNK	*3.3.50 Hyp Protection Region Selection Register* on page 3-129
HPRBAR			c3	0	RW	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
HPRLAR				1	RW	UNK[az]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128
HPRBAR0-14 (even)			c8-15	0	RW	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
HPRLAR0-14 (even)				1	RW	UNK[az]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128
HPRBAR1-15 (odd)				4	RW	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
HPRLAR1-15 (odd)				5	RW	UNK[az]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128

[az] The reset value for bit[0] is 0.

3.3 AArch32 register descriptions

This section describes system registers in alphabetical order based on their name.

3.2.1 c0 registers on page 3-39 to *3.2.18 64-bit registers* on page 3-55 provide cross-references to individual registers arranged by register number.

This section contains the following subsections:

3.3.1 Architectural Feature Access Control Register

The CPACR controls access to the Advanced SIMD and floating-point functionality and indicates that CP0 to CP13 are not implemented.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

The CPACR has no effect on instructions executed at EL2.

Traps and enables

If HCPTR.TCPAC is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If HSTR.T1 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations

This register is available in all build configurations.

Attributes

CPACR is a 32-bit register.

The following figure shows the CPACR bit assignments.

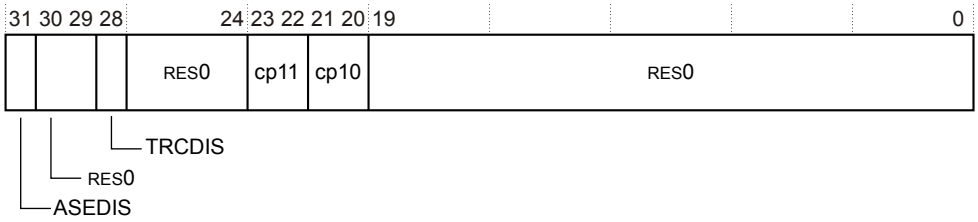

Figure 3-1 CPACR bit assignments

The following table shows the CPACR bit assignments.

Table 3-35 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.
[30:29]	-	Reserved, RES0.
[28]	TRCDIS	This field is RES0. The Cortex-R52 processor does not include a System register interface to the trace macrocell registers.
[27:24]	-	Reserved, RES0.

Table 3-35 CPACR bit assignments (continued)

Bits	Name	Function
[23:22]	cp11[bb]	The value of this field is ignored. If this field is programmed with a different value to the cp10 field, then this field is UNKNOWN on a direct read of the CPACR.
[21:20]	cp10[ba]	Defines the access rights for the Advanced SIMD and floating-point features. Possible values of the fields are: `0b00` EL0 and EL1 accesses to Advanced SIMD and floating-point registers or instructions are UNDEFINED. 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` This control permits full access to Advanced SIMD and floating-point functionality from EL0 and EL1.
[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
```

3.3.2 Auxiliary Control Register

This register is always RES0.

3.3.3 Auxiliary Control Register 2

This register is always RES0.

3.3.4 Auxiliary Data Fault Status Register

The ADFSR provides fault status information for Data Abort exceptions taken to EL1 modes. This fault status information can be used together with the DFSR to further classify faults.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

If HCR.TVM is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.
If HCR.TRVM is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.
If HSTR.T5 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations

This register is available in all build configurations.

[ba] The Advanced SIMD and floating-point features controlled by this field 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.

[bb] cp10 and cp11 must be programmed to the same value.

Attributes

ADFSR is a 32-bit register.

The following figure shows the ADFSR bit assignments.

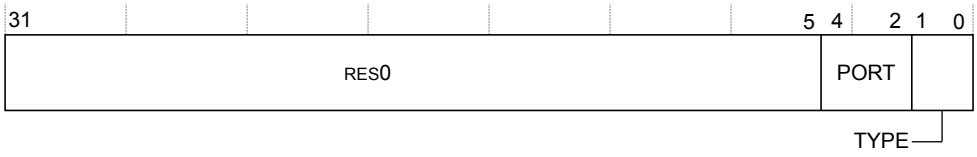

Figure 3-2 ADFSR bit assignments

The following table shows the ADFSR bit assignments.

Table 3-36 ADFSR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0.
[4:2]	PORT	Memory or port that caused the fault: `0b000` AXIM. `0b001` Flash. `0b010` LLPP. `0b011` Internal peripheral interface. `0b100` ATCM. `0b101` BTCM. `0b110` CTCM. `0b111` Overlap. This port is used if the memory port is ambiguous.
[1:0]	TYPE	Fault type: `0b00` No auxiliary error classification. `0b01` Error on external bus control, response signal. `0b10` Error on TCM, Cache, or bus data. `0b11` Error due to bus timeout.

To access the ADFSR:

```
MRC p15,0,<Rt>,c5,c1,0 ; Read ADFSR into Rt
MCR p15,0,<Rt>,c5,c1,0 ; Write Rt to ADFSR
```

3.3.5 Auxiliary Feature Register 0

ID_AFR0 is always RES0.

3.3.6 Auxiliary ID Register

AIDR is always RES0.

3.3.7 Auxiliary Instruction Fault Status Register

The AIFSR provides fault status information for Prefetch Abort exceptions taken to EL1 modes.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

If HCR.TVM is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.
If HCR.TRVM is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.
If HSTR.T5 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations

This register is available in all build configurations.

Attributes

AIFSR is a 32-bit register.

The following figure shows the AIFSR bit assignments.

Figure 3-3 AIFSR bit assignments

The following table shows the AIFSR bit assignments.

Table 3-37 AIFSR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0.
[4:2]	PORT	Memory port that caused the fault: `0b000` AXIM. `0b001` Flash. `0b010` LLPP. `0b100` ATCM. `0b101` BTCM. `0b110` CTCM. `0b111` Overlap. This port is used if the memory port is ambiguous.
[1:0]	TYPE	Fault type: `0b00` No auxiliary error classification. `0b01` Error on external bus control, response signal. `0b10` Error on TCM, Cache, or bus data. `0b11` Error due to bus timeout.

To access the AIFSR:

```
MRC p15,0,<Rt>,c5,c1,1 ; Read AIFSR into Rt
MCR p15,0,<Rt>,c5,c1,1 ; Write Rt to AIFSR
```

3.3.8 Auxiliary Memory Attribute Indirection Register 0

AMAIR0 is always RES0.

3.3.9 Auxiliary Memory Attribute Indirection Register 1

AMAIR1 is always RES0.

3.3.10 Branch Predictor Control Register

The IMP_BPCTLR provides software controls for disabling the dynamic branch prediction.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

If HCR.TIDCP is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations

This register is available in all build configurations.

Attributes

IMP_BPCTLR is a 32-bit register.

The following figure shows the IMP_BPCTLR bit assignments.

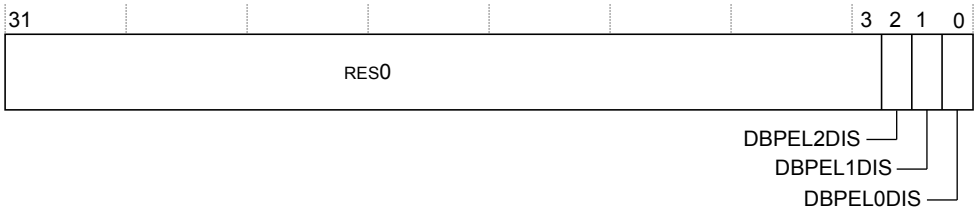

Figure 3-4 IMP_BPCTLR bit assignments

The following table shows the IMP_BPCTLR bit assignments.

Table 3-38 IMP_BPCTLR bit assignments

Bits	Name	Function
[31:3]	-	Reserved, RES0.
[2]	DBPEL2DIS	Disable dynamic branch predictor when running at EL2.
[1]	DBPEL1DIS	Disable dynamic branch predictor when running at EL1.
[0]	DBPEL0DIS	Disable dynamic branch predictor when running at EL0.

To access the IMP_BPCTLR:

```
MRC p15, 1, <Rt>, c9, c1, 1 ; Read IMP_BPCTLR into Rt
MCR p15, 1, <Rt>, c9, c1, 1 ; Write Rt to IMP_BPCTLR
```

3.3.11 Build Options Register

The IMP_BUILDOPTR shows the values of the implementation-time build configuration options used to configure the processor.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

IMP_BUILDOPTR is a 32-bit register.

The following figure shows the IMP_BUILDOPTR bit assignments.

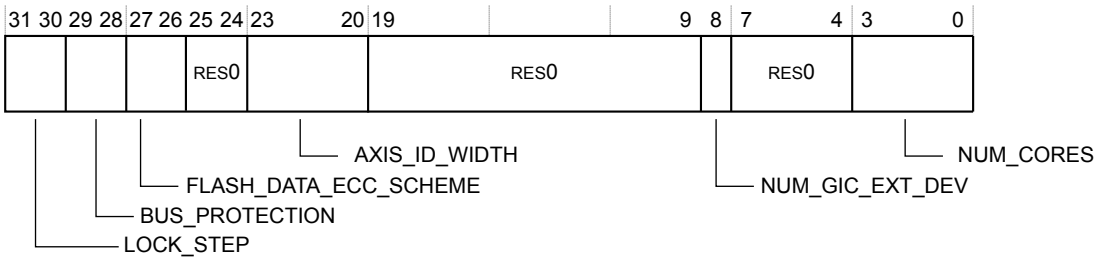

Figure 3-5 IMP_BUILDOPTR bit assignments

The following table shows the IMP_BUILDOPTR bit assignments.

Table 3-39 IMP_BUILDOPTR bit assignments

Bits	Name	Function
[31:30]	LOCK_STEP	Include redundant logic, flops, and comparators, for DCLS: 0 No redundant logic included. 1 DCLS configuration. 2 Split/Lock configuration.
[29:28]	BUS_PROTECTION	Indicates what bus protection scheme is implemented: 0 Signal integrity protection and interconnect protection not included. 1 Include signal integrity protection but do not include interconnect protection. 2 Include signal integrity protection and interconnect protection.

Table 3-39 IMP_BUILDOPTR bit assignments (continued)

Bits	Name	Function
[27:26]	FLASH_DATA_ECC_SCHEME	Flash memory interface data ECC chunk size: **1** 64-bit chunks. **2** 128-bit chunks.
[25:24]	-	Reserved, RES0.
[23:20]	AXIS_ID_WIDTH	Width of AXIS interface ID signals, encoded as the width minus one.
[19:9]	-	Reserved, RES0.
[8]	NUM_GIC_EXT_DEV	Indicates the number of external device interfaces to the GIC. This can be `0` or `1`.
[7:4]	-	Reserved, RES0.
[3:0]	NUM_CORES	Indicates the number of cores in the Cortex-R52 processor, encoded as the number of cores minus one.

To access the IMP_BUILDOPTR:

```
MRC p15, 0, <Rt>, c15, c2, 0 ; Read IMP_BUILDOPTR into Rt
```

3.3.12 Bus Timeout Register

The IMP_BUSTIMEOUTR provides programmable time limits for accesses to AXIM, Flash interface, and LLPP.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW/RO	RW

Accesses to IMP_BUSTIMEOUTR at EL1 are read-only (RO) when HACTLR.BUSTIMEOUTR = 0.

Traps and enables

If HACTLR.BUSTIMEOUTR is set to 0, then write accesses to this register from EL1 are trapped to EL2.

Configurations

This register is available in all build configurations.

Attributes

IMP_BUSTIMEOUTR is a 32-bit register.

The following figure shows the BUSTIMEOUTR bit assignments.

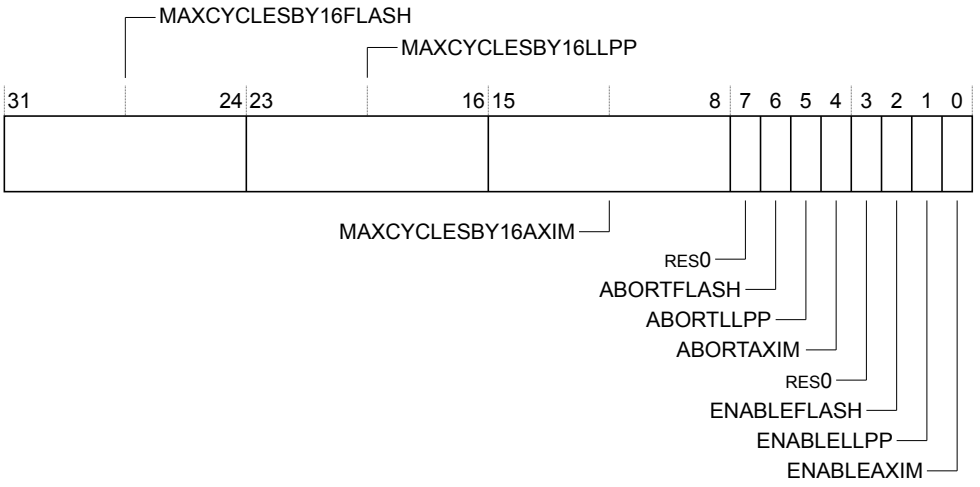

Figure 3-6 IMP_BUSTIMEOUTR bit assignments

The following table shows the IMP_BUSTIMEOUTR bit assignments.

Table 3-40 IMP_BUSTIMEOUTR bit assignments

Bits	Name	Function
[31:24]	MAXCYCLESBY16FLASH	Flash interface timeout value in cycles divided by 16
[23:16]	MAXCYCLESBY16LLPP	LLPP timeout value in cycles divided by 16
[15:8]	MAXCYCLESBY16AXIM	AXIM interface timeout value in cycles divided by 16
7	-	Reserved, RES0
6	ABORTFLASH	When Flash timeout detected, abort current and future flash accesses
5	ABORTLLPP	When LLPP timeout detected, abort current and future LLPP accesses
4	ABORTAXIM	When AXIM timeout detected, abort current and future AXIM accesses
3	-	Reserved, RES0
2	ENABLEFLASH	Timeout counter enable for Flash interface
1	ENABLELLPP	Timeout counter enable for LLPP
0	ENABLEAXIM	Timeout counter enable for AXIM interface

To access the IMP_BUSTIMEOUTR:

```
MRC p15,1,<Rt>,c15,c3,2 ; Read IMP_BUSTIMEOUTR into Rt
MCR p15,1,<Rt>,c15,c3,2 ; Write Rt to IMP_BUSTIMEOUTR
```

Note

The Flash, LLPP and AXIM interface timeout value in cycles (MAXCYCLESBYFLASH16FLASH, MAXCYCLESBYFLASH16LLPP, and MAXCYCLESBYFLASH16AXIM) cannot be zero when the timeout counter is enabled.

3.3.13 Cache Level ID Register

The CLIDR identifies the type of cache or caches that are 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	EL2
-	RO	RO

Traps and enables

If HCR.TID2 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If HSTR.T0 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

Configurations

The register is available in all build configurations.

Attributes

CLIDR is a 32-bit register.

The following figure shows the CLIDR bit assignments.

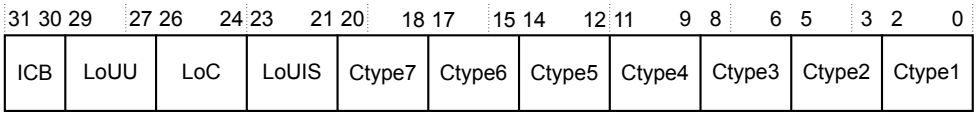

Figure 3-7 CLIDR bit assignments

The following table shows the CLIDR bit assignments.

Table 3-41 CLIDR bit assignments

Bits	Name	Function
[31:30]	ICB	Inner cache boundary: `0b00` Not disclosed in this mechanism.
[29:27]	LoUU	Indicates the Level of Unification Uniprocessor for the cache hierarchy: `0b001` Level 1, if either cache is implemented. `0b000` Level 0, if both instruction and data caches are not implemented.
[26:24]	LoC	Indicates the Level of Coherency for the cache hierarchy: `0b001` Level 1, if either cache is implemented. `0b000` Level 0, if both instruction and data caches are not implemented.
[23:21]	LoUIS	Indicates the Level of Unification Inner Shareable for the cache hierarchy: `0b001` Level 1, if either cache is implemented. `0b000` Level 0, if both instruction and data caches are not implemented.
[20:18]	Ctype7	Indicates the type of cache if the processor implements L7 cache: `0b000` L7 cache not implemented.

Table 3-41 CLIDR bit assignments (continued)

Bits	Name	Function
[17:15]	Ctype6	Indicates the type of cache if the processor implements L6 cache: `0b000` L6 cache not implemented.
[14:12]	Ctype5	Indicates the type of cache if the processor implements L5 cache: `0b000` L5 cache not implemented.
[11:9]	Ctype4	Indicates the type of cache if the processor implements L4 cache: `0b000` L4 cache not implemented.
[8:6]	Ctype3	Indicates the type of cache if the processor implements L3 cache: `0b000` L3 cache not implemented.
[5:3]	Ctype2	Indicates the type of cache if the processor implements L2 cache: `0b000` L2 cache is not implemented.
[2:0]	Ctype1	Indicates the type of cache implemented at L1: `0b000` No cache. `0b001` Instruction cache only. `0b010` Data cache only. `0b011` Separate instruction and data caches.

To access the CLIDR:

```
MRC p15,1,<Rt>,c0,c0,1 ; Read CLIDR into Rt
```

3.3.14 Cache Segregation Control Register

The IMP_CSCTLR controls segregation of instruction and data cache ways between Flash and AXIM.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Writes to IMP_CSCTLR are only permitted before the caches have been enabled, following a system reset. This ensures that the cache segregation controls do not change after either the data cache or the instruction cache has been enabled even if they are subsequently disabled.

Traps and enables

If HCR.TIDCP is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations

This register is available in all build configurations.

Attributes

IMP_CSCTLR is a 32-bit register.

The following figure shows the IMP_CSCTLR bit assignments.

Figure 3-8 IMP_CSCTLR bit assignments

The following table shows the IMP_CSCTLR bit assignments.

Table 3-42 IMP_CSCTLR bit assignments

Bits	Name	Function
[31:11]	-	Reserved, RES0
[10:8]	IFLW	Instruction cache Flash ways
[7:3]	-	Reserved, RES0
[2:0]	DFLW	Data cache Flash ways

The following table shows the IFLW and DFLW field encoding.

Table 3-43 IFLW and DFLW field encoding

Value	Flash ways	AXIM ways
0b000	-	0-3
0b001	0	1-3
0b010	0-1	2-3
0b011	0-2	3
0b100	0-3	-

To access the IMP_CSCTLR:

```
MRC p15,1,<Rt>,c9,c1,0 ; Read IMP_CSCTLR into Rt
MCR p15,1,<Rt>,c9,c1,0 ; Write Rt to IMP_CSCTLR
```

3.3.15 Cache Size Selection Register

The CSSELR selects the current CCSIDR by specifying the required cache level and the cache type, either instruction or data cache.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

If HCR.TID2 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.
If HSTR.T0 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations

This register is available in all build configurations.

Attributes

CSSELR is a 32-bit register.

The following figure shows the CSSELR bit assignments.

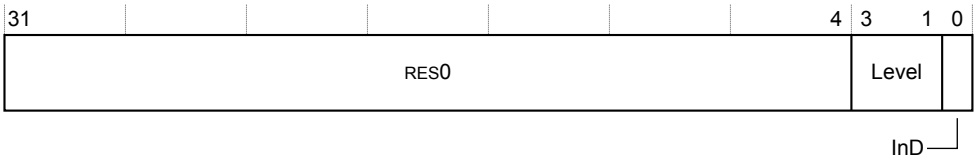

Figure 3-9 CSSELR bit assignments

The following table shows the CSSELR bit assignments.

Table 3-44 CSSELR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:1]	Level	Cache level of required cache. This field is RO: `0b000` L1.
[0]	InD	Instruction not Data bit: `0` Data 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
```

3.3.16 Cache Type Register

The CTR provides information about the architecture of the caches.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and enables

If HCR.TID2 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If HSTR.T0 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

Configurations

This register is available in all build configurations.

Attributes

CTR is a 32-bit register.

The following figure shows the CTR bit assignments.

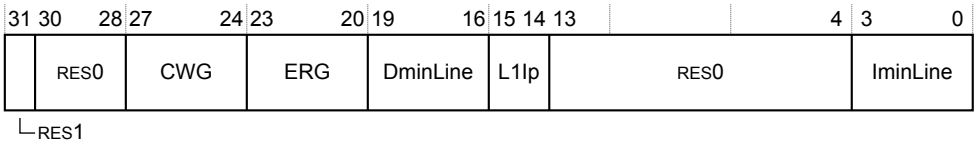

Figure 3-10 CTR bit assignments

The following table shows the CTR bit assignments.

Table 3-45 CTR bit assignments

Bits	Name	Function
[31]	-	Reserved, RES1.
[30:28]	-	Reserved, RES0.
[27:24]	CWG	Cache Write-Back granule. `0x1` Cache Write-Back granule is 2 words[bc].
[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 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: `0b11` *Physically Indexed Physically Tagged* (PIPT).
[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:

```
MRC p15,0,<Rt>,c0,c0,1 ; Read CTR into Rt
```

3.3.17 Configuration Base Address Register

The IMP_CBAR holds the physical base address of the memory-mapped GIC Distributor registers.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and enables

There are no traps and enables affecting this register.

[bc] In the Cortex-R52 processor caches are Write-Through and do not support modified cache lines. If software uses this field to determine DMA buffers alignment, this value indicates that 64-bit aligned DMA buffers can be used without incurring a cache performance penalty.

Configurations

This register is available in all build configurations.

Attributes

IMP_CBAR is a 32-bit register.

The following figure shows the IMP_CBAR bit assignments.

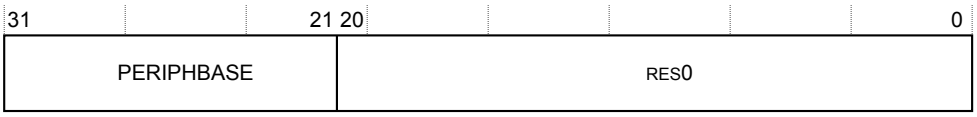

Figure 3-11 IMP_CBAR bit assignments

The following table shows the IMP_CBAR bit assignments.

Table 3-46 IMP_CBAR bit assignments

Bits	Name	Function
[31:21]	PERIPHBASE	Upper bits of base physical address of memory-mapped peripherals. These are the top bits of the base physical address of the memory mapped peripherals memory region. The input **CFGPERIPHBASE[31:21]** determines the reset value.
[20:0]	-	Reserved, RES0.

To access the IMP_CBAR:

```
MRC p15, 1, <Rt>, c15, c3, 0 ; Read IMP_CBAR into Rt
```

3.3.18 Context ID Register

The CONTEXTIDR identifies the current Process Identifier.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

If HCR.TVM is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.
If HCR.TRVM is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.
If HSTR.T13 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations

This register is available in all build configurations.

Attributes

CONTEXTIDR is a 32-bit register.

The following figure shows the CONTEXTIDR bit assignments.

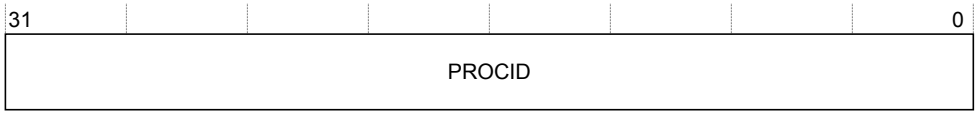

Figure 3-12 CONTEXTIDR bit assignments

The following table shows the CONTEXTIDR bit assignments.

Table 3-47 CONTEXTIDR bit assignments

Bits	Name	Function
[31:0]	PROCID	Process Identifier. This field must be programmed with a unique value that identifies the current process.

This register resets to 0.

To access the CONTEXTIDR:

```
MRC p15, 0, <Rt>, c13, c0, 1 ; Read CONTEXTIDR into Rt
MCR p15, 0, <Rt>, c13, c0, 1 ; Write Rt to CONTEXTIDR
```

3.3.19 CPU Auxiliary Control Register

The CPUACTLR provides the implemented configuration and control options for the processor.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW/RO	RW

CPUACTLR is RW from EL2, RW from EL1 when HACTLR.CPUACTLR is set, RO from EL1 when HACTLR.CPUACTLR is clear and inaccessible from EL0.

The CPU Auxiliary Control Register can be written only when the system is idle. *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* recommends that you write to this register after a powerup reset, before the MPU is enabled.

Note

Setting many of these bits can cause significantly lower performance on your code. Therefore, it is suggested that you do not modify this register unless directed by Arm.

Traps and enables

If HACTLR.CPUACTLR is set to 0, then write accesses to this register from EL1 are trapped to EL2.

Configurations

This register is available in all build configurations.

Attributes

See the register summary in *Table 3-17 64-bit register summary* on page 3-55.

The following figure shows the CPUACTLR bit assignments.

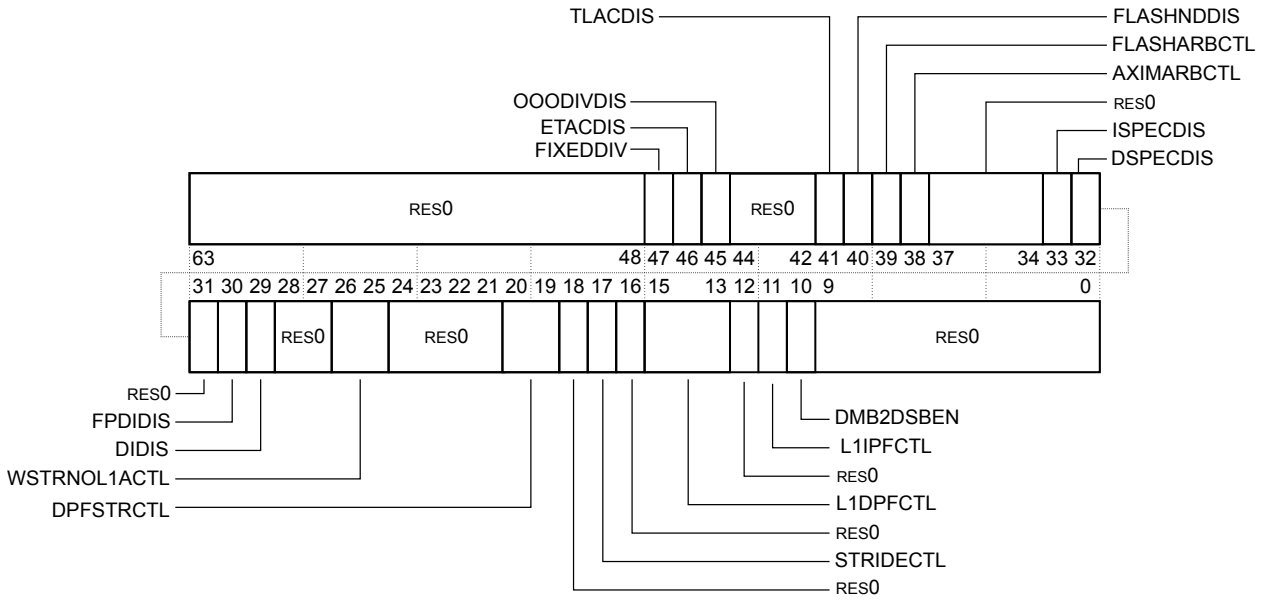

Figure 3-13 CPUACTLR bit assignments

The following table shows the CPUACTLR bit assignments.

Table 3-48 CPUACTLR bit assignments

Bits	Name	Function
[63:48]	-	Reserved, RES0.
[47]	FIXEDDIV	Enable fixed latency for integer divide instructions. The possible values: **0** Disable fixed latency for integer divide instructions. This is the reset value. **1** Enable fixed latency for integer divide instructions.
[46]	ETACDIS	Disable PFU exception target address cache. The possible values are: 0 Enable PFU exception target address cache This is the reset value. 1 Disable PFU exception target address cache.
[45]	OOODIVDIS	Disable out-of-order completion of divide instructions. The possible values are: 0 Enable out-of-order completion of divide instructions. This is the reset value. 1 Disable out-of-order completion of divide instructions.
[44:42]	-	Reserved, RES0.
[41]	TLACDIS	Disable the *Store Unit* (STU) tag lookup avoidance cache. 0 Enable STU tag lookup avoidance cache. This is the reset value. 1 Disable STU tag lookup avoidance cache.
[40]	FLASHNDDIS	Disable flash accesses use of non-flash-dedicated resources. The possible values are: 0 Enable flash accesses use of non-flash-dedicated resources. This is the reset value. 1 Disable flash accesses use of non-flash-dedicated resources.

Table 3-48 CPUACTLR bit assignments (continued)

Bits	Name	Function
[39]	FLASHARBCTL	Flash interface arbitration control: `0` Data side accesses have higher priority. This is the reset value. `1` Instruction side accesses have higher priority.
[38]	AXIMARBCTL	AXIM interface arbitration control. The possible values are: `0` Data side accesses have higher priority. This is the reset value. `1` Instruction side accesses have higher priority.
[37:34]	-	Reserved, RES0.
[33]	ISPECDIS	Disable instruction side speculative access. The possible values are: `0` Enable instruction side speculative access. This is the reset value. `1` Disable instruction side speculative access.
[32]	DSPECDIS	Disable data side speculative access. The possible values are: `0` Enable data side speculative access. This is the reset value. `1` Disable data side speculative access.
[31]	-	Reserved, RES0.
[30]	FPDIDIS	Disable floating-point dual issue. The possible values are: `0` Enable dual issue of Advanced SIMD and floating-point instructions. This is the reset value. `1` Disable dual issue of Advanced SIMD and floating-point instructions.
[29]	DIDIS	Disable Dual Issue. The possible values are: `0` Enable Dual Issue of instructions. This is the reset value. `1` Disable Dual Issue of all instructions.
[28:27]	-	Reserved, RES0.
[26:25]	WSTRNOL1ACTL	Write streaming no-L1-allocate threshold. The possible values are: `0b00` 4th consecutive streaming cache line does not allocate in the L1 cache. This is the reset value. `0b01` 64th consecutive streaming cache line does not allocate in the L1 cache. `0b10` 128th consecutive streaming cache line does not allocate in the L1 cache. `0b11` Disables streaming. All Write-Allocate lines allocate in the L1 cache.
[24:21]	-	Reserved, RES0.
[20:19]	DPFSTRCTL	Number of independent data prefetch streams. The possible values are: `0b00` 1 stream. `0b01` 2 streams. This is the reset value. `0b10` 3 streams. `0b11` 4 streams.

Table 3-48 CPUACTLR bit assignments (continued)

Bits	Name	Function
[18]	-	Reserved, RES0.
[17]	STRIDECTL	Enable stride detection. The possible values are: `0` 2 consecutive strides to trigger prefetch. This is the reset value. `1` 3 consecutive strides to trigger prefetch.
[16]	-	Reserved, RES0.
[15:13]	L1DPFCTL	L1 Data prefetch control. The value of this field determines the maximum number of outstanding data prefetches allowed in the L1 memory system, excluding those generated by software load or PLD instructions. The possible values are: `0b000` Prefetch disabled. `0b001` 1 outstanding prefetch allowed. `0b010` 2 outstanding prefetches allowed. `0b011` 3 outstanding prefetches allowed. `0b100` 4 outstanding prefetches allowed. `0b101` 5 outstanding prefetches allowed. This is the reset value. `0b110` 6 outstanding prefetches allowed. `0b111` 8 outstanding prefetches allowed.
[12]	-	Reserved, RES0.
[11]	L1IPFCTL	L1 Instruction prefetch control. The possible values are: `0` Prefetch disabled. `1` Prefetch enabled. This is the reset value.
[10]	DMB2DSBEN	Enable Data Memory Barrier behaving as Data Synchronization Barrier. The possible values are: `0` Disable Data Memory Barrier behaving as Data Synchronization Barrier. This is the reset value. `1` Enable Data Memory Barrier behaving as Data Synchronization Barrier
[9:0]	-	Reserved, RES0.

To access the CPUACTLR:

```
MRRC p15, 0, <Rt>, <Rt2>, c15 ; Read CPU Auxiliary Control Register
MCRR p15, 0, <Rt>, <Rt2>, c15 ; Write CPU Auxiliary Control Register
```

3.3.20 Current Cache Size ID Register

The CCSIDR provides information about the architecture of the caches.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and enables

EL1 read accesses to CCSIDR are trapped to EL2 when HCR.TID2 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

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.

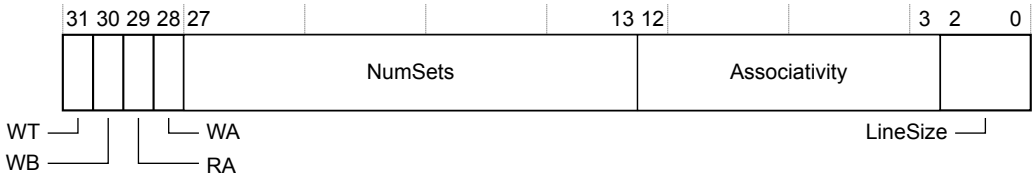

Figure 3-14 CCSIDR bit assignments

The following table shows the CCSIDR bit assignments.

Table 3-49 CCSIDR bit assignments

Bits	Name	Function
[31]	WT	Indicates support for Write-Through: `0` Cache level does not support Write-Through. `1` Cache level supports Write-Through. **Note** Value `0` is returned when the instruction cache is selected.
[30]	WB	Indicates support for Write-Back: `0` Cache level does not support Write-Back.
[29]	RA	Indicates support for 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. **Note** Value `0` is returned when the instruction cache is selected.
[27:13]	NumSets	Indicates the number of sets in cache - 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.[bd]

[bd] For more information about encoding see *Table 3-50 CCSIDR encodings* on page 3-95.

Table 3-49 CCSIDR bit assignments (continued)

Bits	Name	Function
[12:3]	Associativity	Indicates the associativity of cache: `0x3` The cache has four ways.
[2:0]	LineSize	Indicates the number of words in cache line: `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 3-50 CCSIDR encodings

CSSELR	Cache	Size	Complete register encoding	Register bit field encoding						
				WT	WB	RA	WA	NumSets	Associativity	LineSize
`0x0`	L1 Data cache	4KB	`0xB001E01A`	1	0	1	1	`0x000F`	`0x003`	`0x2`
		8KB	`0xB003E01A`					`0x001F`		
		16KB	`0xB007E01A`					`0x003F`		
		32KB	`0xB00FE01A`					`0x007F`		
`0x1`	L1 Instruction cache	4KB	`0x2001E01A`	0	0	1	0	`0x000F`	`0x003`	`0x2`
		8KB	`0x2003E01A`					`0x001F`		
		16KB	`0x2007E01A`					`0x003F`		
		32KB	`0x201FE01A`					`0x007F`		

To access the CCSIDR:

```
MRC p15, 1, <Rt>, c0, c0, 0 ; Read CCSIDR into Rt
```

3.3.21 Data Cache Error Record Registers 0 and 1

The IMP_DCERR0 and IMP_DCERR1 registers indicate the data cache RAM and the cache index of a detected data cache error.

Usage constraints

These registers are accessible as follows:

EL0	EL1	EL2
-	RW/RO	RW

These registers are accessible as RW from EL1 when HACTLR.ERR is set, RO from EL1 when HACTLR.ERR is clear.

Traps and enables

EL1 write accesses to these registers are trapped to EL2 when HACTLR.ERR is 0.

Configurations

These registers are available on all cores configured with a data cache.

Attributes

IMP_DCERR0 and IMP_DCERR1 are 32-bit registers.

The following figure shows the IMP_DCERR0 and IMP_DCERR1 bit assignments.

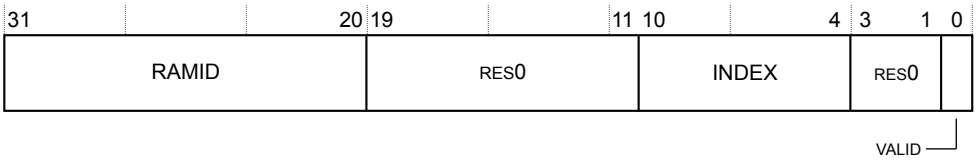

Figure 3-15 IMP_DCERR0 and IMP_DCERR1 bit assignments

The following table shows the IMP_DCERR0 and IMP_DCERR1 bit assignments.

Table 3-51 IMP_DCERR0 and IMP_DCERR1 bit assignments

Bits	Name	Function	Notes
[31:20]	RAMID	Indicates the RAM bank: **RAMID[11]** Tag RAM, bank 3. **RAMID[10]** Tag RAM, bank 2. **RAMID[9]** Tag RAM, bank 1. **RAMID[8]** Tag RAM, bank 0. **RAMID[7]** Data RAM, bank 7. **RAMID[6]** Data RAM, bank 6. **RAMID[5]** Data RAM, bank 5. **RAMID[4]** Data RAM, bank 4. **RAMID[3]** Data RAM, bank 3. **RAMID[2]** Data RAM, bank 2. **RAMID[1]** Data RAM, bank 1. **RAMID[0]** Data RAM, bank 0.	-
[19:11]	-	Reserved, RES0.	-
[10:4]	INDEX	Data cache index. This is address bits [12:6] appropriately masked with the cache size.	-
[3:1]	-	Reserved, RES0.	Writes to these registers are unique. If bit[1] and bit[0] are set to 0, and if the register holds information about a correctable error, then the register is cleared. If bit[1] is set to 1 and bit[0] is set to 0, then the register is always cleared.
[0]	VALID	Register contents are valid.	

To access the IMP_DCERR0:

```
MRC p15,2,<Rt>,c15,c1,0 ; Read IMP_DCERR0 into Rt
MCR p15,2,<Rt>,c15,c1,0 ; Write Rt to IMP_DCERR0
```

To access the IMP_DCERR1:

```
MRC p15,2,<Rt>,c15,c1,1 ; Read IMP_DCERR1 into Rt
MCR p15,2,<Rt>,c15,c1,1 ; Write Rt to IMP_DCERR1
```

3.3.22 Data Fault Address Register

The DFAR holds the faulting address that caused the last data fault (synchronous only) or a watchpoint hit that was not taken to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

DFAR reads and writes from EL1 are trapped to EL2 according to HCR.TRVM and HCR.TVM respectively.

Configurations

This register is available in all build configurations.

Attributes

DFAR is a 32-bit register.

The following figure shows the DFAR bit assignments.

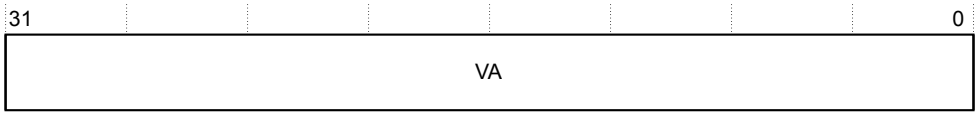

Figure 3-16 DFAR bit assignments

The following table shows the DFAR bit assignments.

Table 3-52 DFAR bit assignments

Bits	Name	Function
[31:0]	VA	The 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
```

3.3.23 Data Fault Status Register

The DFSR holds status information about the last data fault (synchronous or asynchronous) or a watchpoint hit that was not taken to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

DFSR reads and writes from EL1 are trapped to EL2 according to HCR.TRVM and HCR.TVM respectively.

Configurations

This register is available in all build configurations.

Attributes

DFSR is a 32-bit register.

The following figure shows the DFSR bit assignments.

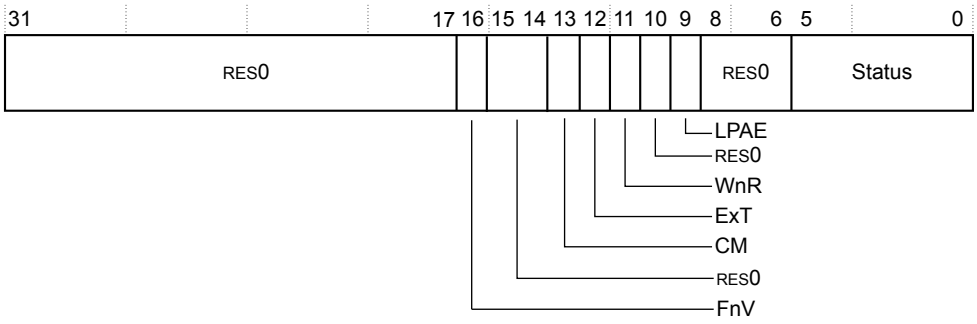

Figure 3-17 DFSR bit assignments

The following table shows the DFSR bit assignments.

Table 3-53 DFSR bit assignments

Bits	Name	Function
[31:17]	-	Reserved, RES0.
[16]	FnV	This field is RES0.
[15:14]	-	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.
[11]	WnR	Write not Read bit. This field indicates if a write or a read access caused the abort: `0` Abort caused by a read access. `1` Abort caused by a write access. For faults on cache maintenance operations with (coproc==`0b1111`), including the translation operations, this bit always returns a value of 1.
[10]	-	Reserved, RES0.
[9]	LPAE	This field is RES1.

Table 3-53 DFSR bit assignments (continued)

Bits	Name	Function
[8:6]	-	Reserved, RES0.
[5:0]	Status	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved. 0b000100 Translation fault, level 0. 0b001100 Permission fault, level 0. 0b010000 Synchronous external abort. 0b010001 Asynchronous external abort. 0b011000 Synchronous parity error on memory access. [be] 0b011001 Asynchronous parity error on memory access.[be] 0b100001 Alignment fault. 0b100010 Debug event. 0b110101 Unsupported Exclusive access fault.

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
```

3.3.24 Debug Feature Register 0

The ID_DFR0 provides top-level information about the debug system.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Must be interpreted with the Main ID Register, MIDR.

Traps and enables

EL1 read accesses to ID_DFR0 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_DFR0 is a 32-bit register.

The following figure shows the ID_DFR0 bit assignments.

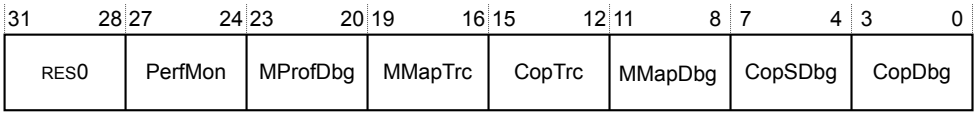

Figure 3-18 ID_DFR0 bit assignments

The following table shows the ID_DFR0 bit assignments.

[be] Parity is the generic architectural term. This is the value used for memory ECC and bus protection errors which generate aborts.

Table 3-54 ID_DFR0 bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RES0.
[27:24]	PerfMon	Indicates support for performance monitor model: `0b0011` Support for *Performance Monitor Unit version 3* (PMUv3) system registers.
[23:20]	MProfDbg	Indicates support for memory-mapped debug model for M profile processors: `0b0000` Processor does not support M profile Debug architecture.
[19:16]	MMapTrc	Indicates support for memory-mapped trace model: `0b0001` Support for Arm trace architecture, with memory-mapped access. In the Trace registers, the TRCIDR0-13 give more information about the implementation.
[15:12]	CopTrc	Indicates support for system register based trace model: `0b0000` Processor does not support Arm trace architecture, with (coproc==`0b1110`) access.
[11:8]	MMapDbg	This field is RES0.
[7:4]	CopSDbg	This field is RES0.
[3:0]	CopDbg	Indicates support for system register based debug model: `0b0110` Processor supports v8 Debug architecture, with (coproc==`0b1110`) access.

To access the ID_DFR0:

```
MRC p15,0,<Rt>,c0,c1,2 ; Read ID_DFR0 into Rt
```

3.3.25 EL0 Read/Write Software Thread ID Register

The TPIDRURW register provides a location where software executing at EL0 can store thread identifying information, for OS management purposes.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
RW	RW	RW

Traps and enables

EL0 and EL1 accesses to TPIDRURW are trapped to EL2 when HSTR.T13 is set.

Configurations

This register is available in all build configurations.

Attributes

TPIDRURW is a 32-bit registers.

The following figure shows the TPIDRURW bit assignments.

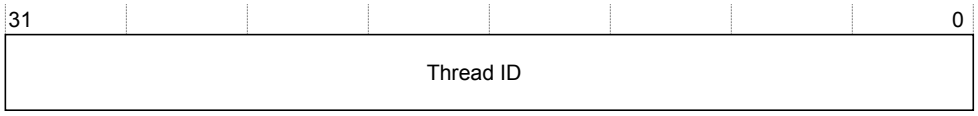

Figure 3-19 TPIDRURW bit assignments

The following table shows the TPIDRURW bit assignments.

Table 3-55 TPIDRURW bit assignments

Bits	Name	Function
[31:0]	Thread ID	Thread identifying information stored by software running at EL0.

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

To access the TPIDRURW:

```
MRC p15, 0, <Rt>, c13, c0, 2 ; Read TPIDRURW into Rt
MCR p15, 0, <Rt>, c13, c0, 2 ; Write Rt to TPIDRURW
```

3.3.26 EL0 Read-Only Software Thread ID Register

The TPIDRURO register provides a location where software executing at EL1 or higher can store thread identifying information that is visible to software executing at EL0, for OS management purposes.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
RO	RW	RW

Traps and enables

EL0 and EL1 accesses to TPIDRURO are trapped to EL2 when HSTR.T13 is set.

Configurations

This register is available in all build configurations.

Attributes

TPIDRURO is a 32-bit registers.

The following figure shows the TPIDRURO bit assignments.

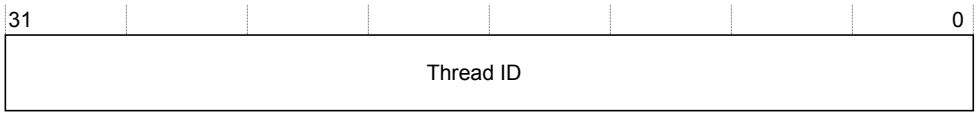

Figure 3-20 TPIDRURO bit assignments

The following table shows the TPIDRURO bit assignments.

Table 3-56 TPIDRURO bit assignments

Bits	Name	Function
[31:0]	Thread ID	Thread identifying information stored by software running at EL1.

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

To access the TPIDRURO:

```
MRC p15, 0, <Rt>, c13, c0, 3 ; Read TPIDRURO into Rt
MCR p15, 0, <Rt>, c13, c0, 3 ; Write Rt to TPIDRURO
```

3.3.27 EL1 Software Thread ID Register

The TPIDRPRW register provides a location where software executing at EL1 or higher can store thread identifying information that is not visible to software execution at EL0. This is for OS management purposes.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

EL1 accesses to TPIDRPRW are trapped to EL2 when HSTR.T13 is set.

Configurations

This register is available in all build configurations.

Attributes

TPIDRPRW is a 32-bit registers.

The following figure shows the TPIDRPRW bit assignments.

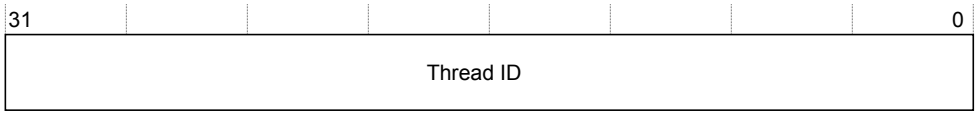

Figure 3-21 TPIDRPRW bit assignments

The following table shows the TPIDRPRW bit assignments.

Table 3-57 TPIDRPRW bit assignments

Bits	Name	Function
[31:0]	Thread ID	Thread identifying information stored by software running at EL1.

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

To access the TPIDRPRW:

```
MRC p15, 0, <Rt>, c13, c0, 4 ; Read TPIDRPRW into Rt
MCR p15, 0, <Rt>, c13, c0, 4 ; Write Rt to TPIDRPRW
```

3.3.28 FCSE Process ID Register

Armv8-R obsoletes the FCSE functionality. The FCSEIDR behaves as RAZ/WI in Armv8-R.

3.3.29 Flash Error Record Registers 0 and 1

The IMP_FLASHERR0 and IMP_FLASHERR1 registers indicate the type of error and address index of a detected flash data error.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW/RO	RW

Register is accessible as RW from EL1 when HACTLR.ERR is set, RO from EL1 when HACTLR.ERR is clear.

Traps and enables

EL1 write accesses to these registers are trapped to EL2 when HACTLR.ERR is 0.

Configurations

This register is available in all build configurations.

Attributes

IMP_FLASHERR0 and IMP_FLASHERR1 are 32-bit registers.

The following figure shows the IMP_FLASHERR0 and IMP_FLASHERR1 bit assignments.

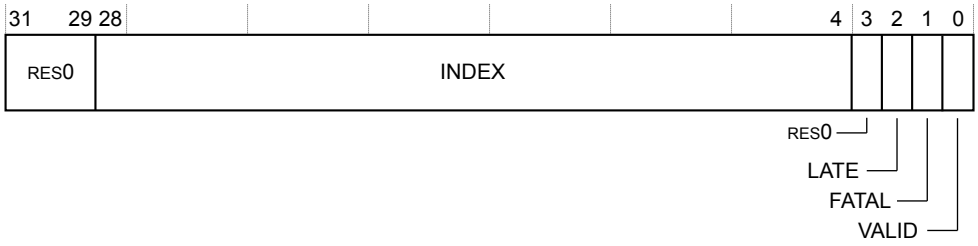

Figure 3-22 IMP_FLASHERR0 and IMP_FLASHERR1 bit assignments

The following table shows the IMP_FLASHERR0 and IMP_FLASHERR1 bit assignments.

Table 3-58 IMP_FLASHERR0 and IMP_FLASHERR1 bit assignments

Bits	Name	Function
[31:29]	-	Reserved, RES0.
[28:4]	INDEX	Index into memory. This corresponds to bits [25:1] of the access address.
[3]	-	Reserved, RES0.
[2]	LATE	Recorded error was a late error.
[1]	FATAL	Recorded error is fatal or correctable: **0** Correctable. **1** Fatal.
[0]	VALID	Register contents are valid.

To access the IMP_FLASHERR0:

```
MRC p15,2,<Rt>,c15,c3,0 ; Read IMP_FLASHERR0 into Rt
MCR p15,2,<Rt>,c15,c3,0 ; Write Rt to IMP_FLASHERR0
```

To access the IMP_FLASHERR1:

```
MRC p15,2,<Rt>,c15,c3,1 ; Read IMP_FLASHERR1 into Rt
MCR p15,2,<Rt>,c15,c3,1 ; Write Rt to IMP_FLASHERR1
```

Note

Writes to these registers are unique. If bit[1] and bit[0] are set to 0, and if the register holds information about a correctable error, then the register is cleared. If bit[1] is set to 1 and bit[0] is set to 0, then the register is always cleared.

Related references

7.9.4 Bus protection on page 7-247

3.3.30 Flash Interface Region Register

The IMP_FLASHIFREGIONR indicates the base address of the Flash interface region. It provides control to enable and disable the Flash interface.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW/RO	RW

Accesses to IMP_FLASHIFREGIONR from EL1 are RW when HACTLR.FLASHIFREGIONR is 1, and RO when HACTLR.FLASHIFREGIONR is 0.

Traps and enables

EL1 accesses to IMP_FLASHIFREGIONR are trapped to EL2 when HCR.TIDCP is set.

Configurations

This register is available in all build configurations.

Attributes

IMP_FLASHIFREGIONR is a 32-bit register.

The following figure shows the IMP_FLASHIFREGIONR bit assignments.

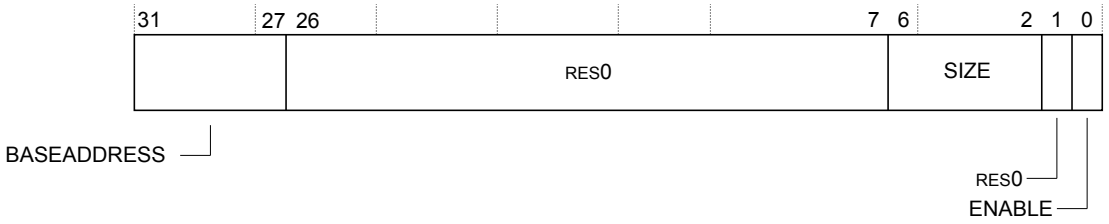

Figure 3-23 IMP_FLASHIFREGIONR bit assignments

The following table shows the IMP_FLASHIFREGIONR bit assignments.

Table 3-59 IMP_FLASHIFREGIONR bit assignments

Bits	Name	Function
[31:27]	BASEADDRESS	Indicates upper bits of the Flash interface region base address on input signals **CFGFLASHBASEADDR[31:27]**.
[26:7]	-	Reserved, RES0.
[6:2]	SIZE	Flash interface region size. See *Table 3-60 IMP_FLASHIFREGION.SIZE supported values* on page 3-105.
[1]	-	Reserved, RES0.
[0]	ENABLE	Enable the Flash interface. At reset, set to input signal **CFGFLASHENx**.

The following table shows the supported Flash interface region sizes.

Table 3-60 IMP_FLASHIFREGION.SIZE supported values

SIZE value	Flash interface region size
0b00000	0KB (no Flash region).
0b10010	128MB

To access the IMP_FLASHIFREGIONR:

```
MRC p15,0,<Rt>,c15,c0,1 ; Read IMP_FLASHIFREGIONR into Rt
MCR p15,0,<Rt>,c15,c0,1 ; Write Rt to IMP_FLASHIFREGIONR
```

3.3.31 Hyp Architectural Feature Trap Register

The HCPTR controls trapping to EL2 of access, at EL1 or lower, to Advanced SIMD and floating-point functionality. It also controls access from EL2 to this functionality.

Usage constraints This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables EL1 accesses to HCPTR are trapped to EL2 when HSTR.T1 is set.

Configurations This register is available in all build configurations.

Attributes HCPTR is a 32-bit register.

The following figure shows the HCPTR bit assignments.

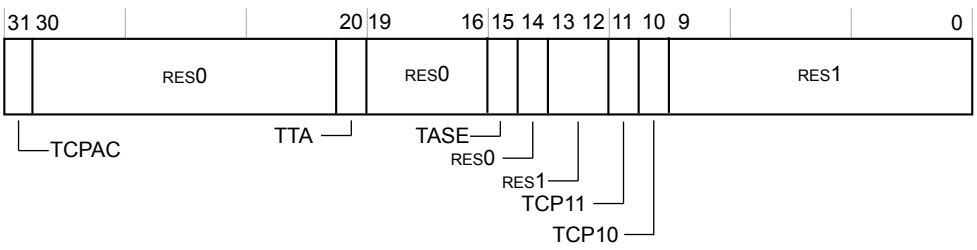

Figure 3-24 HCPTR bit assignments

The following table shows the HCPTR bit assignments.

Table 3-61 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 EL1 CPACR accesses to EL2. The reset value is 0.
[30:21]	-	Reserved, RES0.
[20]	TTA	Traps System register accesses to all implemented trace registers to EL2. The implementation does not include a System register interface to the trace macrocell registers, therefore the value is RES0.
[19:16]	-	Reserved, RES0.
[15]	TASE	Trap Advanced SIMD instructions to EL2 when the value of HCPTR.TCP10 is 0. The possible values of this bit are: 0 This control has no effect on execution of Advanced SIMD instructions. 1 When the value of HCPTR.TCP10 is 0, any attempt to execute an Advanced SIMD instruction is trapped to EL2, unless it is trapped to EL1 by a CPACR or NSACR control. A trapped instruction generates: • A Hyp Trap exception, if the exception is taken from EL0 to EL1. • An Undefined Instruction exception taken to EL2, if the exception is taken from EL2. When the value of HCPTR.TCP10 is 1, the value of this field is ignored. The reset value is 0.
[14]	-	Reserved, RES0.
[13:12]	-	Reserved, RES1.
[11]	TCP11	The value of this field is ignored. If this field is programmed with a different value to the TCP10 bit, then this field is UNKNOWN on a direct read of the HCPTR.
[10]	TCP10	Trap accesses to Advanced SIMD and floating-point functionality to EL2. The possible values of this bit are: 0 This control has no effect on accesses to Advanced SIMD and floating-point functionality. 1 Any attempted access to Advanced SIMD and floating-point functionality is trapped to EL2, unless it is trapped to EL1 by a CPACR or NSACR control. A trapped instruction generates: • A Hyp Trap exception, if the exception is taken from EL0 to EL1. • An Undefined Instruction exception taken to EL2, if the exception is taken from EL2. The reset value is 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
```

3.3.32 Hyp Auxiliary Configuration Register

HACR register is always RES0.

3.3.33 Hyp Auxiliary Control Register

The HACTLR controls trapping to EL2 of Cortex-R52-specific system register accesses performed at EL1.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HACTLR are trapped to EL2 when HSTR.T1 is set.

Configurations

This register is available in all build configurations.

Attributes

HACTLR is a 32-bit register.

The following figure shows the HACTLR bit assignments.

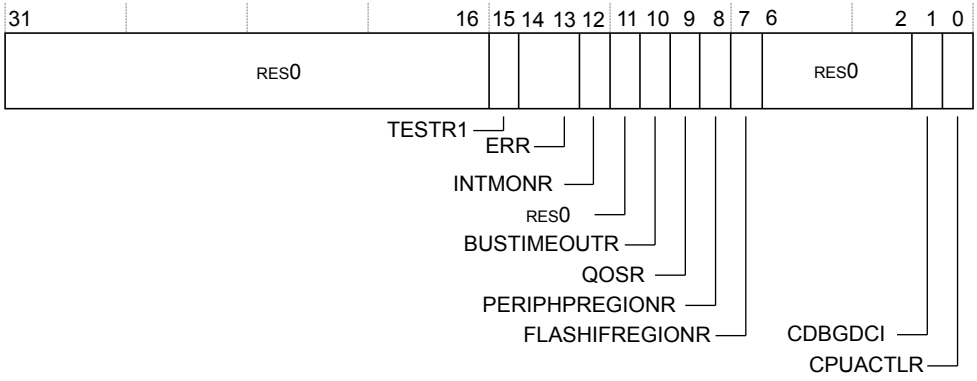

Figure 3-25 HACTLR bit assignments

The following table shows the HACTLR bit assignments.

Table 3-62 HACTLR bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15]	TESTR1	Controls access to IMP_TESTR1 at EL0 and EL1: 0 Traps write access at EL0 and EL1. 1 Enables access at EL0 and EL1.
[14]	-	Reserved, RES0.
[13]	ERR	Controls access to IMP_DCERR0, IMP_DCERR1, IMP_ICERR0, IMP_ICERR1, IMP_TCMERR0, IMP_TCMERR1, IMP_FLASHERR0, and IMP_FLASHERR1 registers: 0 Traps write accesses to registers at EL1. 1 Enables access to registers at EL1.

Table 3-62 HACTLR bit assignments (continued)

Bits	Name	Function
[12]	INTMONR	Controls access to IMP_INTMONR at EL1: 0 Traps write access at EL1. 1 Enables access at EL1.
[11]	-	Reserved, RES0.
[10]	BUSTIMEOUTR	Controls access to IMP_BUSTIMEOUTR at EL1: 0 Traps write access at EL1. 1 Enables access at EL1.
[9]	QOSR	Controls access to QOSR at EL1: 0 Traps write access at EL1. 1 Enables access at EL1.
[8]	PERIPHPREGIONR	Controls access to IMP_PERIPHPREGIONR at EL1: 0 Traps write access at EL1. 1 Enables access at EL1.
[7]	FLASHIFREGIONR	Controls access to IMP_FLASHIFREGIONR at EL1: 0 Traps write access at EL1. 1 Enables access at EL1.
[6:2]	-	Reserved, RES0.
[1]	CDBGDCI	Controls access to CDBGDCI at EL1: 0 Traps write access at EL1. 1 Enables access at EL1.
[0]	CPUACTLR	IMP_CPUACTLR write access control. The possible values are: 0 Traps write access at EL1. This is the reset value. 1 The register is write accessible from EL1.

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
```

3.3.34 Hyp Auxiliary Control Register 2

This register is always RAZ/WI.

3.3.35 Hyp Auxiliary Data Fault Status Register

The HADFSR provides additional syndrome information for Data Abort exceptions taken to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

HADFSR reads and writes from EL1 are trapped to EL2 according to HCR.TRVM and HCR.TVM respectively.

Configurations

This register is available in all build configurations.

Attributes

HADFSR is a 32-bit register.

The following figure shows the HADFSR bit assignments.

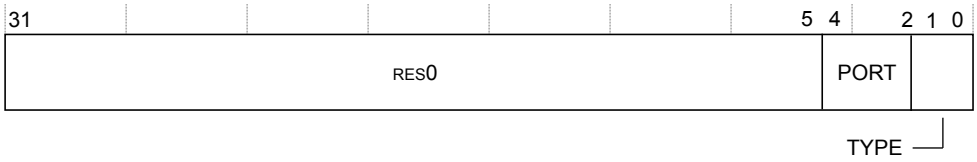

Figure 3-26 HADFSR bit assignments

The following table shows the HADFSR bit assignments.

Table 3-63 HADFSR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0.
[4:2]	PORT	Memory port that caused the fault: `0b000` AXIM. `0b001` Flash. `0b010` LLPP. `0b011` Internal peripheral interface. `0b100` ATCM. `0b101` BTCM. `0b110` CTCM. `0b111` UNKNOWN (memory port is ambiguous).
[1:0]	TYPE	Fault type: `0b00` Other error, not because of response, ECC, or timeout. `0b01` Error on response. `0b10` ECC error on data. `0b11` Bus timeout error.

To access the HADFSR:

```
MRC p15,4,<Rt>,c5,c1,0 ; Read HADFSR into Rt
MCR p15,4,<Rt>,c5,c1,0 ; Write Rt to HADFSR
```

3.3.36 Hyp Auxiliary Instruction Fault Status Register

The HAIFSR provides additional syndrome information for Prefetch Abort exceptions taken to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

HAIFSR reads and writes from EL1 are trapped to EL2 according to HCR.TRVM and HCR.TVM respectively.

Configurations

This register is available in all build configurations.

Attributes

HAIFSR is a 32-bit register.

The following figure shows the HAIFSR bit assignments.

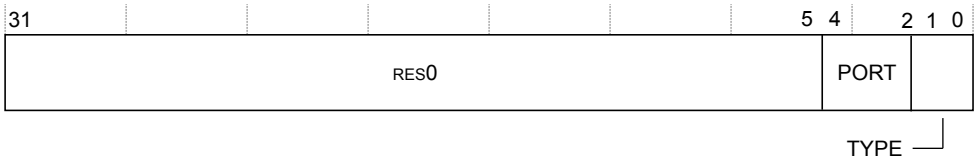

Figure 3-27 HAIFSR bit assignments

The following table shows the HAIFSR bit assignments.

Table 3-64 HAIFSR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0.
[4:2]	PORT	Memory port that caused the fault: 0b000 AXIM. 0b001 Flash. 0b010 LLPP. 0b100 ATCM. 0b101 BTCM. 0b110 CTCM. 0b111 UNKNOWN (memory port is ambiguous).
[1:0]	TYPE	Fault type: 0b00 Other error, not because of response, ECC, or timeout. 0b01 Error on response. 0b10 ECC error on data. 0b11 Bus timeout error.

To access the HAIFSR:

```
MRC p15,4,<Rt>,c5,c1,1 ; Read HAIFSR into Rt
MCR p15,4,<Rt>,c5,c1,1 ; Write Rt to HAIFSR
```

3.3.37 Hyp Auxiliary Memory Attribute Indirection Register 0

The processor does not implement HAMAIR0, so this register is always RES0.

3.3.38 Hyp Auxiliary Memory Attribute Indirection Register 1

The processor does not implement HAMAIR1, so this register is always RES0.

3.3.39 Hyp Configuration Register

The HCR provides configuration controls for virtualization, including defining whether various operations are trapped to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HCR are trapped to EL2 when HSTR.T1 is set.

Configurations

This register is available in all build configurations.

Attributes

HCR is a 32-bit register.

The following figure shows the HCR bit assignments.

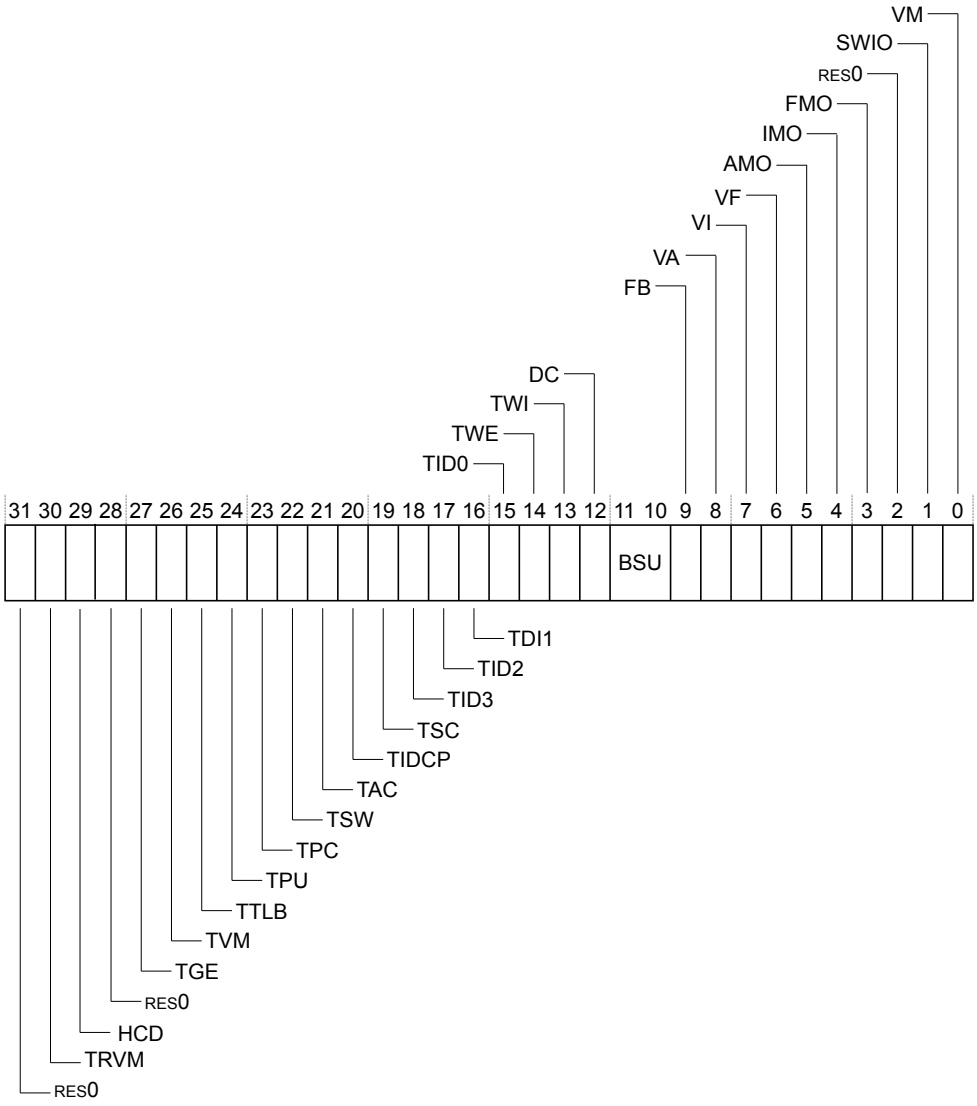

Figure 3-28 HCR bit assignments

The following table shows the HCR bit assignments.

Table 3-65 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. The registers for which read accesses are trapped are as follows: SCTLR, DFSR, IFSR, DFAR, IFAR, ADFSR, AIFSR, MAIR0, MAIR1, AMAIR0, AMAIR1, CONTEXTIDR, PRSELR, PRBAR, PRBARn, PRLAR, and PRLARn. The reset value is 0.

Table 3-65 HCR bit assignments (continued)

Bits	Name	Function
[29]	HCD	HVC instruction disable. The options are: **0** HVC instruction execution is enabled at EL2 and EL1. **1** HVC instructions are UNDEFINED at EL2 and EL1. The Undefined Instruction exception is taken to the Exception level at which the HVC instruction is executed. The reset value is 0.
[28]	-	Reserved, RES0.
[27]	TGE	Trap General Exceptions. If this bit 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. • An exception return to EL1 is treated as an illegal exception return. Also, 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. The registers for which write accesses are trapped are as follows: SCTLR, DFSR, IFSR, DFAR, IFAR, ADFSR, AIFSR, MAIR0, MAIR1, AMAIR0, AMAIR1, CONTEXTIDR, PRSELR, PRBAR, PRBARn, PRLAR, and PRLARn. The reset value is 0.
[25]	TTLB	This field is RES0.
[24]	TPU	Trap cache maintenance instructions that operate to the point of unification. When 1, this causes those instructions 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 Cache maintenance operations that operate to the point of coherency. When 1, this causes those instructions 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 Cache maintenance operations by Set/Way. When 1, this causes those instructions 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 access to the ACTLR is trapped to Hyp mode. The reset value is 0.

Table 3-65 HCR bit assignments (continued)

Bits	Name	Function
[20]	TIDCP	Trap the following registers. When 1, this causes accesses to all MCR and MRC instructions with (coproc==`0b1111`) 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 to these registers from EL0 are UNDEFINED. The reset value is 0.
[19]	TSC	This field is RES0.
[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_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: • (coproc==`0b1111`), OPC1 is 0, CRn is 0, CRm is c3, c4, c5, c6, or c7, and Opc2 is 0 or 1. • (coproc==`0b1111`), Opc1 is 0, CRn is 0, CRm is c3, and Opc2 is 2. • (coproc==`0b1111`), 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, AIDR, MPUIR, TLBTR, 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 Jazelle ID Register (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. For more information about the dependency of the WFE instruction on the state of the Event Register, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.
[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 3-65 HCR bit assignments (continued)

Bits	Name	Function
[12]	DC	Default cacheable. This bit is extended to enable default cacheability in PMSA mode for EL1 and EL0. When set to 1, the default memory map of a PMSA guest is substituted for Normal, Non-shareable, Inner Write-Back Write-Allocate, Outer Write-Back Write-Allocate. Setting this bit to one does not force SCTLR.M to zero for PMSA contexts. 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 `0b00`.
[9]	FB	Force broadcast. Has no effect on Cortex-R52 because the Inner Shareable domain only contains one core. 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 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 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 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 on physical asynchronous aborts when the core is in EL1 or EL0 exception state, and enables virtual exception signaling by the VA bit and **VSEI** input signal. The reset value is 0.
[4]	IMO	IRQ Mask Override. When this is set to 1, it overrides the masking effect of CPSR.I on physical IRQ exceptions when the core is in EL1 or EL0 exception state, and enables virtual exception signaling by the VI bit and interrupt controller. The reset value is 0.
[3]	FMO	FIQ Mask Override. When this is set to 1, it overrides the masking effect of CPSR.I on physical FIQ exceptions when the core is in EL1 or EL0 exception state, and enables virtual exception signaling by the VF bit and interrupt controller. The reset value is 0.

Table 3-65 HCR bit assignments (continued)

Bits	Name	Function
[2]	PTW	This field is RES0.
[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	Enables the EL2-controlled MPU protection for EL1 and EL0 accesses. `0` EL1 and EL0 stage 2 address protection disabled, unless the HCR.DC bit is 1. `1` EL1 and EL0 stage 2 address protection and attribute coalescing enabled. 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
```

3.3.40 Hyp Configuration Register 2

HCR2 register is always RES0.

3.3.41 Hyp Data Fault Address Register

The HDFAR holds the faulting address that causes a synchronous Data Abort exception to be taken to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HDFAR are trapped to EL2 when HSTR.T6 is set.

Configurations

This register is available in all build configurations.

Attributes

HDFAR is a 32-bit register.

Traps and Enables

EL1 accesses to HDFAR are trapped to EL2 if HSTR.T6 is set.

The following figure shows the HDFAR bit assignments.

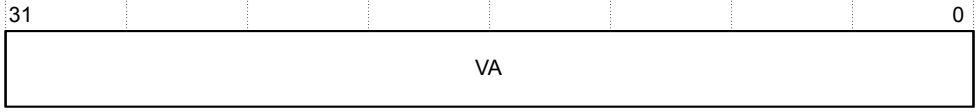

Figure 3-29 HDFAR bit assignments

The following table shows the HDFAR bit assignments.

Table 3-66 HDFAR bit assignments

Bits	Name	Function
[31:0]	VA	The 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
```

3.3.42 Hyp Debug Control Register

The HDCR controls the trapping to Hyp mode of 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	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HDCR are trapped to EL2 when HSTR.T1 is set.

Configurations

This register is available in all build configurations.

Attributes

HDCR is a 32-bit register.

The following figure shows the HDCR bit assignments.

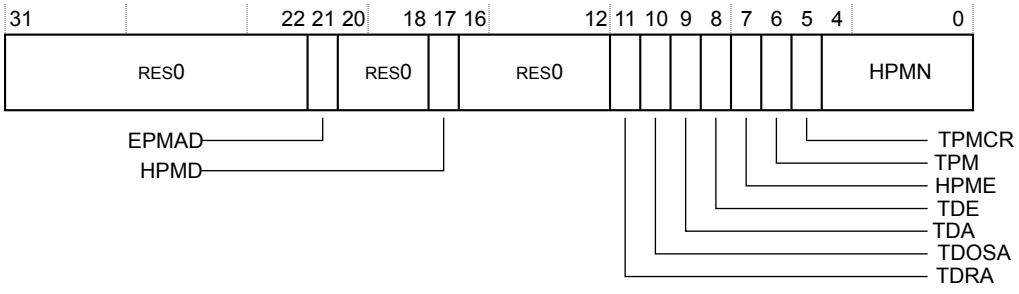

Figure 3-30 HDCR bit assignments

The following table shows the HDCR bit assignments.

Table 3-67 HDCR bit assignments

Bits	Name	Function
[31:22]	-	Reserved, RES0.
[21]	EPMAD	Disables external debugger access to Performance Monitor registers. 0 Access to Performance Monitors registers from external debugger is permitted. 1 Access to Performance Monitors registers from external debugger is disabled. The reset value is 0.
[20:18]	-	Reserved, RES0.
[17]	HPMD	Hyp performance monitors disable. This prohibits event counting in Hyp mode. 0 Event counting allowed in Hyp mode. 1 Event counting prohibited in Hyp mode. This control applies to: • The counters in the range between 0 to HPMN. • If PMCR.DP is set to 1, PMCCNTR. The higher numbered event counters are unaffected. If PMCR.DP is set to 0, PMCCNTR is unaffected. On warm reset, the field reset to 0.
[16:12]	-	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 EL1 and EL0 access to debug ROM address registers, DBGDRAR and DBGDSAR, to Hyp mode. 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 debug system registers. 1 Trap valid accesses to OS-related debug system registers to Hyp mode. Registers for which accesses are trapped are as follows: • 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 3-67 HDCR bit assignments (continued)

Bits	Name	Function
[9]	TDA	Trap Debug Access: 0 Has no effect on accesses to debug system registers. 1 Trap valid accesses to debug system registers, other than the registers trapped by HDCR.TDRA and HDCR.TDOSA, to Hyp mode. 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.
[8]	TDE	Trap Debug Exceptions: 0 Has no effect on Debug exceptions. 1 Route Non-secure Debug exceptions 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 is enabled to the performance monitors that are reserved for use from Hyp mode. For more information, see the description of the HPMN field. On warm reset the field resets to 0.
[6]	TPM	Trap Performance Monitor accesses: 0 Has no effect on performance monitor accesses. 1 Trap valid performance monitor accesses to Hyp mode. The reset value is 0. See the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* for more information.

Table 3-67 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 PMCR accesses to Hyp mode. The reset value is 0. See the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* for more information.
[4:0]	HPMN	Hyp Performance Monitor count. It defines the number of Performance Monitor counters that are accessible from EL1 and EL0 modes if unprivileged access is enabled. If software is accessing Performance Monitors counter *n*, then: • If *n* is in the range $0 \le n <$ 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 $\le n <$PMCR.N, the counter is accessible only from EL2, and only if enabled by HPME. If this field is set to 0, then there is no access to any counters from EL0 or EL1. If this field is set to a value larger than PMCR.N, all counters are available in EL0 or EL1. EL2 reads of HDCR.HPMN always return the value written to HDCR.HPMN. This is regardless of whether the value is 0 or larger than PMCR.N. The reset value is `0x04`.

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
```

3.3.43 Hyp Instruction Fault Address Register

The HIFAR holds the faulting address that causes a synchronous Prefetch Abort exception to be taken to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HIFAR are trapped to EL2 when HSTR.T6 is set.

Configurations

This register is available in all build configurations.

Attributes

HIFAR is a 32-bit register.

The following figure shows the HIFAR bit assignments.

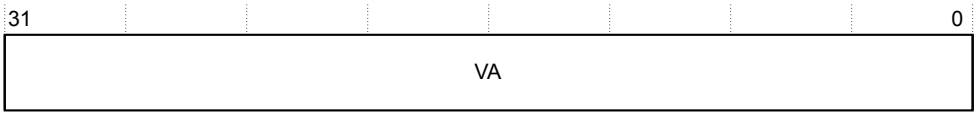

Figure 3-31 HIFAR bit assignments

The following table shows the HIFAR bit assignments.

Table 3-68 HIFAR bit assignments

Bits	Name	Function
[31:0]	VA	The 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
```

3.3.44 Hyp IPA Fault Address Register

The HPFAR holds the faulting address for prefetch and data aborts on a stage 2 translation taken to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HPFAR are trapped to EL2 when HSTR.T6 is set.

Configurations

This register is available in all build configurations.

Attributes

HPFAR is a 32-bit register.

The following figure shows the HPFAR bit assignments.

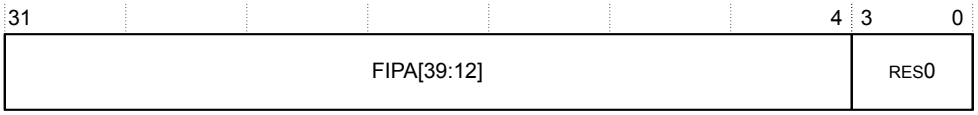

Figure 3-32 HPFAR bit assignments

The following table shows the HPFAR bit assignments.

Table 3-69 HPFAR bit assignments

Bits	Name	Function
[31:4]	FIPA[39:12]	Bits [31:4] of the faulting 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
```

3.3.45 Hyp Memory Attribute Indirection Register 0 and 1

The HMAIR0 and HMAIR1 provide memory attribute encodings corresponding to the possible AttrIndex values in a Long-descriptor format translation table entry. This is for the EL2-controlled MPU.

Usage constraints

These registers are accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HMAIR0 and HMAIR1 are trapped to EL2 if HSTR.T10 is set.

Configurations

These registers are available in all build configurations.

Attributes

HMAIR0 and HMAIR1 are 32-bit registers.

The following figure shows the HMAIR0 and HMAIR1 bit assignments.

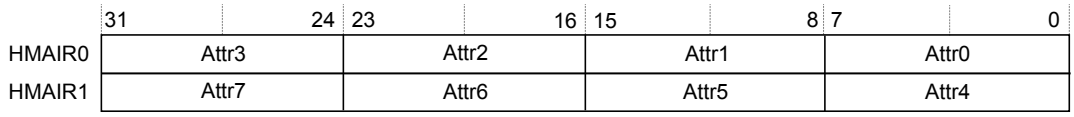

Figure 3-33 HMAIR0 and HMAIR1 bit assignments

The following table shows HMAIR0 bit assignments.

Table 3-70 HMAIR0 bit assignments

Bits	Name	Definition
[31:24]	Attr3	The memory attribute encoding for EL2-controlled MPU regions, which have AttrIndex set to 3.
[23:16]	Attr2	The memory attribute encoding for EL2-controlled MPU regions, which have AttrIndex set to 2.
[15:8]	Attr1	The memory attribute encoding for EL2-controlled MPU regions, which have AttrIndex set to 1.
[7:0]	Attr0	The memory attribute encoding for EL2-controlled MPU regions, which have AttrIndex set to 0.

The following table shows HMAIR1 bit assignments.

Table 3-71 HMAIR1 bit assignments

Bits	Name	Definition
[31:24]	Attr7	The memory attribute encoding for EL2-controlled MPU regions, which have AttrIndex set to 7.
[23:16]	Attr6	The memory attribute encoding for EL2-controlled MPU regions, which have AttrIndex set to 6.
[15:8]	Attr5	The memory attribute encoding for EL2-controlled MPU regions, which have AttrIndex set to 5.
[7:0]	Attr4	The memory attribute encoding for EL2-controlled MPU regions, which have AttrIndex set to 4.

The following table shows the Attr<n>[7:4] bit encodings:

Table 3-72 Attr<n>[7:4] bit encodings

Attr<n>[7:4]	Name
0000	Device memory. See encoding of Attr<n>[3:0] for the type of Device memory.
00RW, RW not 00	Normal Memory, Outer Write-Through transient.
0100	Normal Memory, Outer Non-cacheable.
01RW, RW not 00	Normal Memory, Outer Write-Back transient.
10RW	Normal Memory, Outer Write-Through non-transient.
11RW	Normal Memory, Outer Write-Back non-transient.

R = Outer Read Allocate Policy, W = Outer Write Allocate Policy.

The meaning of bits[3:0] depends on the value of bits[7:4]. The following table shows the Attr<n>[3:0] bit encodings

Table 3-73 Attr<n>[3:0] bit encodings

Attr<n>[3:0]	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
0000	Device-nGnRnE memory	UNPREDICTABLE
00RW, RW not 00	UNPREDICTABLE	Normal Memory, Inner Write-Through transient
0100	Device-nGnRE memory	Normal memory, Inner Non-cacheable
01RW, RW not 00	UNPREDICTABLE	Normal Memory, Inner Write-Back transient
1000	Device-nGRE memory	Normal Memory, Inner Write-Through non-transient (RW=00)
10RW, RW not 00	UNPREDICTABLE	Normal Memory, Inner Write-Through non-transient
1100	Device-GRE memory	Normal Memory, Inner Write-Back non-transient (RW=00)
11RW, RW not 00	UNPREDICTABLE	Normal Memory, Inner Write-Back non-transient

R = Inner Read Allocate Policy, W = Inner Write Allocate Policy.

The R and W bits in some Attr<n> fields have the following meanings:

Table 3-74 R and W bit definitions

R or W	Meaning
0	Do not allocate
1	Allocate

To access the HMAIR0:

```
MRC p15, 4, <Rt>, c10, c2, 0 ; Read HMAIR0 into Rt
MCR p15, 4, <Rt>, c10, c2, 0 ; Write Rt to HMAIR0
```

To access the HMAIR1:

```
MRC p15, 4, <Rt>, c10, c2, 1 ; Read HMAIR1 into Rt
MCR p15, 4, <Rt>, c10, c2, 1 ; Write Rt to HMAIR1
```

3.3.46 Hyp MPU Region Enable Register

The HPRENR provides direct access to the region enable (HPRLAR.EN) for regions 0 to 15 of the EL2-controlled MPU.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

HPRENR, is a 32-bit register.

The following figure shows the HPRENR bit assignments when there are no EL2-controlled MPU regions implemented.

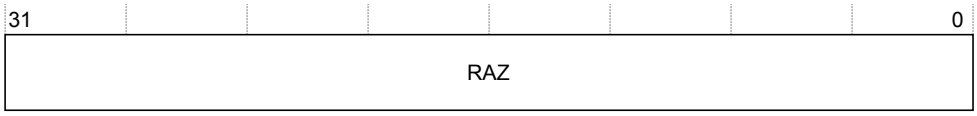

Figure 3-34 HPRENR, bit assignments

The following table shows the HPRENR bit assignments.

Table 3-75 HPRENR bit assignments

Bits	Name	Function
[31:16]	-	RAZ.

The following figure shows the HPRENR bit assignments when there are 16 EL2-controlled MPU regions implemented.

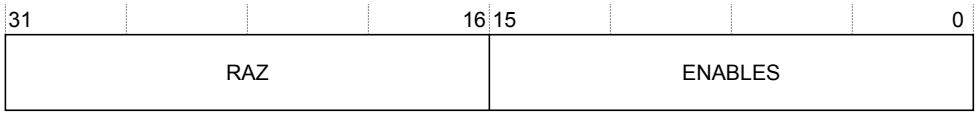

Figure 3-35 HPRENR, bit assignments

The following table shows the HPRENR bit assignments.

Table 3-76 HPRENR bit assignments

Bits	Name	Function
[31:16]	-	RAZ.
[15:0]	ENABLES	Each bit, N, enables or disables the respective EL2-controlled MPU N region.

The following figure shows the HPRENR bit assignments when there are 20 EL2-controlled MPU regions implemented.

Figure 3-36 HPRENR, bit assignments

The following table shows the HPRENR bit assignments.

Table 3-77 HPRENR bit assignments

Bits	Name	Function
[31:20]	-	RAZ.
[19:0]	ENABLES	Each bit, N, enables or disables the respective EL2-controlled MPU N region.

The following figure shows the HPRENR bit assignments when there are 24 EL2-controlled MPU regions implemented.

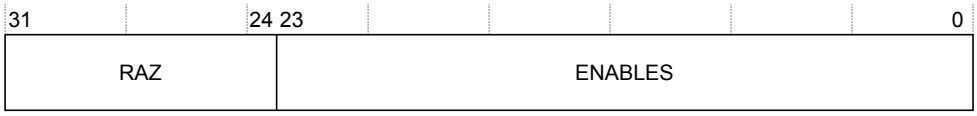

Figure 3-37 HPRENR, bit assignments

The following table shows the HPRENR bit assignments.

Table 3-78 HPRENR bit assignments

Bits	Name	Function
[31:24]	-	RAZ.
[23:0]	ENABLES	Each bit, N, enables or disables the respective EL2-controlled MPU N region.

To access the HPRENR:

```
MRC p15,4,<Rt>,c6,c1,1 ; Read HPRENR, into Rt
MCR p15,4,<Rt>,c6,c1,1 ; Write Rt to HPRENR
```

3.3.47 Hyp MPU Type Register

The HMPUIR indicates the number of programmable memory regions implemented by the EL2-controlled MPU.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RO

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

HMPUIR is a 32-bit register.

The following figure shows the HMPUIR bit assignments.

Figure 3-38 HMPUIR bit assignments

The following table shows the HMPUIR bit assignments.

Table 3-79 HMPUIR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	REGION	Indicates the number of programmable memory regions implemented by the EL2-controlled MPU, either 0, 16, 20, or 24.

To access the HMPUIR:

```
MRC p15,4,<Rt>,c0,c0,4 ; Read HMPUIR into Rt
```

3.3.48 Hyp Protection Region Base Address Register

The Hyp Protection Region Base Address Register indicates the base address of the EL2-controlled MPU region, and provides two mechanisms for accessing it, direct (HPRBAR<n>) and indirect (HPRBAR). For the indirect access also see HPRSELR.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

HPRBAR and HPRBARn are 32-bit registers.

The following figure shows the HPRBAR bit assignments.

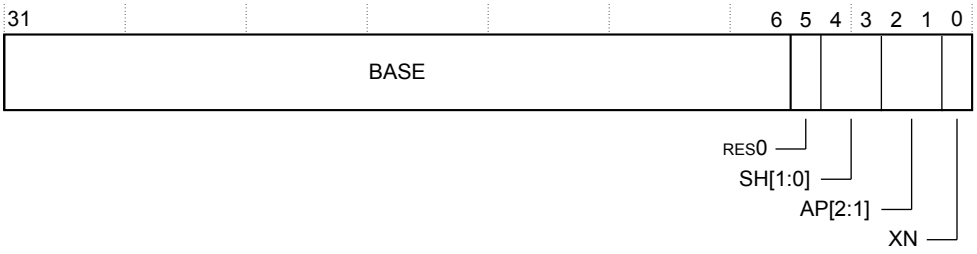

Figure 3-39 HPRBAR bit assignments

The following table shows the HPRBAR bit assignments.

Table 3-80 HPRBAR bit assignments

Bits	Name	Function
[31:6]	BASE	Contains bits[31:6] of the lower inclusive limit of the selected EL2-controlled MPU memory region. This value is zero extended to provide the base address to be checked against.
[5]	-	Reserved, RES0.
[4:3]	SH[1:0]	Shareability field.
[2:1]	AP[2:1]	Access Permission bits.
[0]	XN	Execute-never.

The following table shows the SH[1:0] field encoding for Normal memory.

Table 3-81 SH[1:0] field encoding for Normal memory

SH[1:0]	Normal memory
0b00	Non-shareable
0b01	UNPREDICTABLE
0b10	Outer Shareable
0b11	Inner Shareable

The following table shows the AP[2:1] Data access permissions for EL2-controlled MPU.

Table 3-82 AP[2:1] Data access permissions for EL2-controlled MPU

AP[2:1]	Access from EL2	Access from EL0 and EL1	Description
0b00	Read/write	None	Read/write, only at EL2
0b01	Read/write	Read/write	Read/write, at any privilege level
0b10	Read-only	None	Read-only, only at EL2
0b11	Read-only	Read-only	Read-only, at any privilege level

This register resets to an UNKNOWN value.

To access the HPRBAR:

```
MRC p15,4,<Rt>,c6,c3,0 ; Read HPRBAR into Rt
MCR p15,4,<Rt>,c6,c3,0 ; Write Rt to HPRBAR
```

Direct access is provided to HPRBAR0 to HPRBAR15. To access HPRBARn, where n is referenced as a binary number:

```
MRC p15, 4, <Rt>, c6, c8+n[3:1], 4*n[0] ; Read PRBARn into Rt
MCR p15, 4, <Rt>, c6, c8+n[3:1], 4*n[0] ; Write Rt into PRBARn
```

3.3.49 Hyp Protection Region Limit Address Register

The Hyp Protection Region Limit Address Register indicates the limit address of the EL2-controlled MPU region, and provides two mechanisms for accessing it, direct (HPRLARn) and indirect (HPRLAR). For the indirect access also see HPRSELR.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

HPRLAR and HPRLARn are 32-bit registers.

The following figure shows the HPRLAR bit assignments.

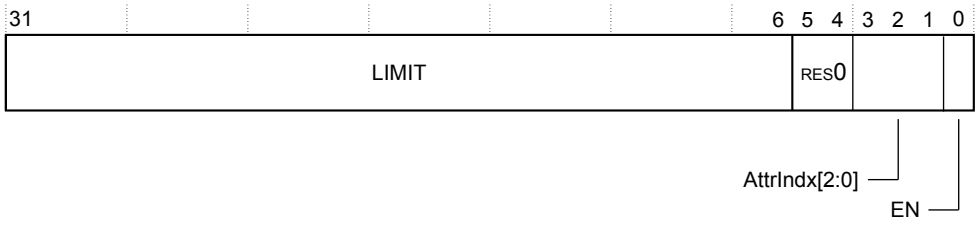

Figure 3-40 HPRLAR bit assignments

The following table shows the HPRLAR bit assignments.

Table 3-83 HPRLAR bit assignments

Bits	Name	Function
[31:6]	LIMIT	Contains bits[31:6] of the upper inclusive limit of the selected EL2-controlled MPU memory region. This value is postfixed with `0x3F` to provide the limit address to be checked against. Resets to an UNKNOWN value.
[5:4]	-	Reserved, RES0.

Table 3-83 HPRLAR bit assignments (continued)

Bits	Name	Function
[3:1]	AttrIndx[2:0]	Indexes a set of attributes in one of the HMAIRx registers. Resets to an UNKNOWN value.
[0]	EN	Region enable. 0 Region disabled. 1 Region enabled. This field resets to zero.

To access the HPRLAR:

```
MRC p15,4,<Rt>,c6,c3,1 ; Read HPRLAR into Rt
MCR p15,4,<Rt>,c6,c3,1 ; Write Rt to HPRLAR
```

Direct access is provided to HPRLAR0 to HPRLAR15. To access HPRLARn, where n is referenced as a binary number:

```
MRC p15, 4, <Rt>, c6, c8+n[3:1], 4*n[0]+1 ; Read PRLARn into Rt
MCR p15, 4, <Rt>, c6, c8+n[3:1], 4*n[0]+1 ; Write Rt into PRLARn
```

3.3.50 Hyp Protection Region Selection Register

The HPRSELR indicates, and selects the current EL2-controlled MPU region, which can be indirectly accessed using the HPRBAR and HPRLAR registers.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

HPRSELR is a 32-bit register.

0 or 16 EL2-controlled MPU regions

The following figure shows the HPRSELR bit assignments if 0 or 16 EL2-controlled MPU regions are implemented.

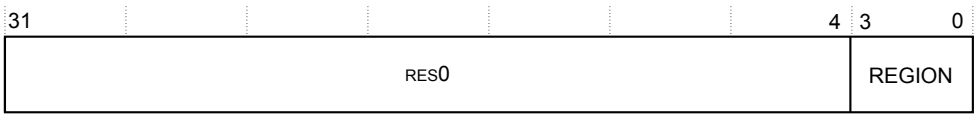

Figure 3-41 HPRSELR bit assignments for 0 or 16 EL2-controlled MPU regions

The following table shows the HPRSELR bit assignments if 0 or 16 EL2-controlled MPU regions are implemented.

Table 3-84 HPRSELR bit assignments for 0 or 16 EL2-controlled MPU regions

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	REGION	The number of the current region visible in HPRBAR and HPRLAR. If a 0 region MPU is implemented, writing a value to this register has UNPREDICTABLE results. If a 16 region MPU is implemented, writing a value to this register greater than or equal to 16 has UNPREDICTABLE results.

20 or 24 EL2-controlled MPU regions

The following figure shows the HPRSELR bit assignments if 20 or 24 EL2-controlled MPU regions are implemented.

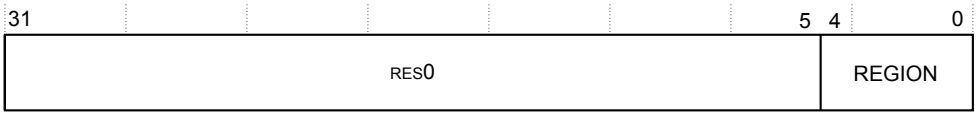

Figure 3-42 HPRSELR bit assignments for 20 or 24 EL2-controlled MPU regions

The following table shows the HPRSELR bit assignments if 20 or 24 EL2-controlled MPU regions are implemented.

Table 3-85 HPRSELR bit assignments for 20 or 24 EL2-controlled MPU regions

Bits	Name	Function
[31:5]	-	Reserved, RES0.
[4:0]	REGION	The number of the current region visible in HPRBAR and HPRLAR. If a 20 region MPU is implemented, writing a value to this register greater than or equal to 20 has UNPREDICTABLE results. If a 24 region MPU is implemented, writing a value to this register greater than or equal to 24 has UNPREDICTABLE results.

To access the HPRSELR:

```
MRC p15,4,<Rt>,c6,c2,1 ; Read HPRSELR into Rt
MCR p15,4,<Rt>,c6,c2,1 ; Write Rt to HPRSELR
```

3.3.51 Hyp Software Thread ID Register

The HTPIDR register provides a location where software running in Hyp mode can store thread identifying information, which is visible to software executing at EL0 or EL1. This is for hypervisor management purposes.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HTPIDR are trapped to EL2 when HSTR.T13 is set.

Configurations

This register is available in all build configurations.

Attributes

HTPIDR is a 32-bit registers.

This register is UNKNOWN on reset.

The following figure shows the HTPIDR bit assignments.

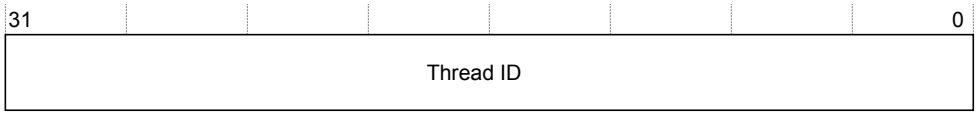

Figure 3-43 HTPIDR bit assignments

The following table shows the HTPIDR bit assignments.

Table 3-86 HTPIDR bit assignments

Bits	Name	Function
[31:0]	Thread ID	Thread identifying information stored by software running at this Exception level.

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

To access the HTPIDR:

```
MRC p15, 4, <Rt>, c13, c0, 2 ; Read HTPIDR into Rt
MCR p15, 4, <Rt>, c13, c0, 2 ; Write Rt to HTPIDR
```

3.3.52 Hyp Syndrome Register

The HSR holds syndrome information for an exception taken to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HSR are trapped to EL2 when HSTR.T5 is set.

Configurations

This register is available in all build configurations.

Attributes

HSR is a 32-bit register.

The following figure shows the HSR bit assignments.

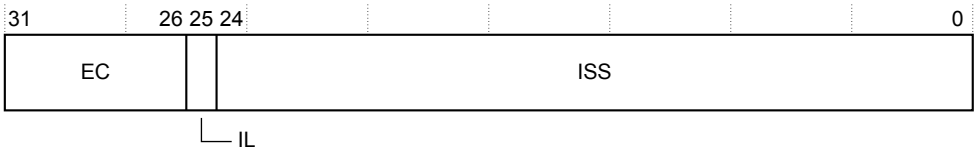

Figure 3-44 HSR bit assignments

The following table shows the HSR bit assignments.

Table 3-87 HSR bit assignments

Bits	Name	Function
[31:26]	EC	Exception class. The exception class for the exception that is taken in Hyp mode. See *Table 3-88 Exception class encodings* on page 3-132.
[25]	IL	Instruction length bit. Indicates the size of the instruction that has been trapped to Hyp mode. When this bit is valid, possible values of this bit are: `0` 16-bit instruction trapped. `1` 32-bit instruction trapped. This field is RES1, and not valid for the following cases: • When the EC field is `0b000000`, indicating an exception with an unknown reason. • Prefetch Aborts. • Data Aborts that do not have valid ISS information, or for which the ISS is not valid. • When the EC value is `0b001110`, indicating an Illegal state exception. The IL field is not valid and is UNKNOWN on an exception from a PC alignment fault.
[24:0]	ISS	Instruction specific syndrome. The interpretation of this field depends on the value of the EC field. See the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* for more information.

Table 3-88 Exception class encodings

EC value	Exception class	Description
`0b000000`	Exceptions with an unknown reason	Unknown reason.
`0b000001`	Exception from a WFI or WFE instruction	Trapped WFI or WFE instruction. Conditional WFE and WFI instructions that fail their condition code check do not cause an exception.
`0b000011`	Exception from an MCR or MRC access	Trapped MCR or MRC access with (coproc==`0b1111`).
`0b000100`	Exception from an MCRR or MRRC access	Trapped MCRR or MRRC access with (coproc==`0b1111`).
`0b000101`	Exception from an MCR or MRC access	Trapped MCR or MRC access with (coproc==`0b1110`).
`0b000110`	Exception from an LDC or STC access with (coproc==`0b1110`).	Trapped LDC or STC access with (coproc==`0b1110`).
`0b000111`	Exception from an access to SIMD or floating-point functionality, resulting from HCPTR	Exception from an access to SIMD or floating-point functionality, as a result of HCPTR. Excludes exceptions resulting from when Advanced SIMD and floating-point are not implemented. These are reported with EC value `0b000000`.
`0b001000`	Exception from an MCR or MRC access	Trapped VMRS access for ID group traps.

Table 3-88 Exception class encodings (continued)

EC value	Exception class	Description
0b001100	Exception from an MCRR or MRRC access	Trapped MRRC or MCRR access with (coproc==0b1110).
0b001110	Exception from an Illegal state or PC alignment fault	Illegal state exception taken to AArch32 state.
0b010001	Exception from HVC or SVC instruction execution	SVC taken to Hyp mode.
0b010010	Exception from HVC or SVC instruction execution	HVC executed.
0b100000	Exception from a Prefetch abort	Prefetch Abort routed to Hyp mode.
0b100001	Exception from a Prefetch abort	Prefetch Abort taken from Hyp mode.
0b100010	Exception from an Illegal state or PC alignment fault	PC alignment fault exception.
0b100100	Exception from a Data abort	Data Abort routed to Hyp mode.
0b100101	Exception from a Data abort	Data Abort taken from Hyp mode.

To access the HSR:

```
MRC p15, 4, <Rt>, c5, c2, 0 ; Read HSR into Rt
MCR p15, 4, <Rt>, c5, c2, 0 ; Write Rt to HSR
```

3.3.53 Hyp System Control Register

The HSCTLR provides top level control of the system operation in Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HSCTLR are trapped to EL2 when HSTR.T1 is set.

Configurations

This register is available in all build configurations.

Attributes

HSCTLR is a 32-bit register.

The following figure shows the HSCTLR bit assignments.

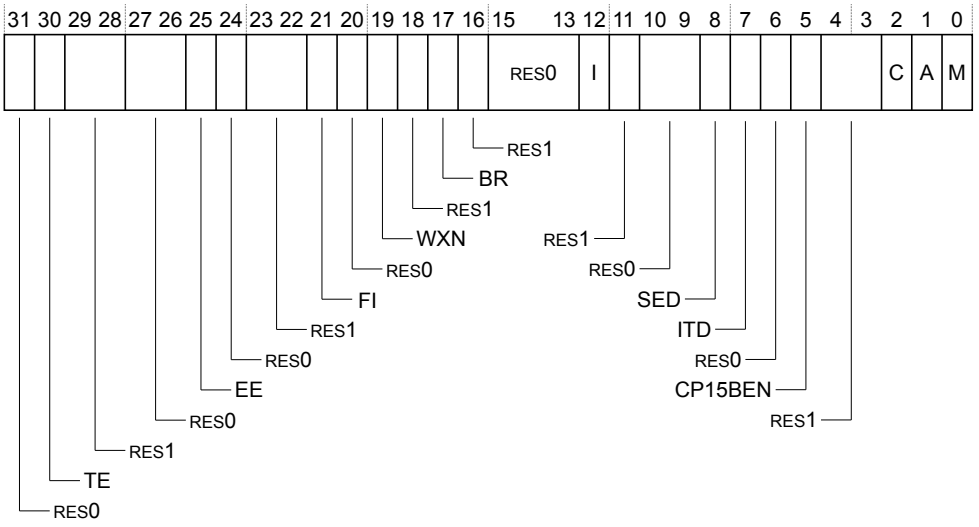

Figure 3-45 HSCTLR bit assignments

The following table shows the HSCTLR bit assignments.

Table 3-89 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. The input signal **CFGTHUMBEXCEPTIONS** defines the reset value of the TE bit.
[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 an exception taken to Hyp mode, including reset. 0 Little endian. 1 Big endian. The input signal **CFGENDIANESSx** defines the reset value of the EE bit.
[24]	-	Reserved, RES0.
[23:22]	-	Reserved, RES1.

Table 3-89 HSCTLR bit assignments (continued)

Bits	Name	Function
[21]	FI	Fast Interrupts configuration enable bit: 0 All performance features enabled. 1 Fast interrupts configuration. Some performance features disabled. Setting this bit to 1 can reduce interrupt latency in an implementation, by disabling some implementation defined performance features. See *Fast interrupts* on page 2-32. The resets value is 0.
[20]	-	Reserved, RES0.
[19]	WXN	Write permission implies XN. This bit can be used to require all memory regions with write permission from the EL2-controlled MPU to be treated as XN: 0 EL2-controlled MPU regions with write permission are not forced to XN. 1 EL2-controlled MPU regions with write permission are forced to XN. The reset value is 0.
[18]	-	Reserved, RES1.
[17]	BR	Background Region enable bit. When the EL2 MPU is enabled this controls how a EL2 access that does not map to any EL2 MPU region is handled: 0 Any EL2 transaction not matching a EL2 MPU region results in a translation fault. 1 Enables the background region for accesses from EL2 that do not match a programmable EL2 MPU region. See *8.2.2 EL2-controlled MPU background region* on page 8-254. The reset value is 0.
[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. 1 Instruction caches enabled at EL2. The field resets to 0.
[11]	-	Reserved, RES1.
[10:9]	-	Reserved, RES0.
[8]	SED	SETEND Disable: 0 The SETEND instruction is available at EL2. 1 The SETEND instruction is UNDEFINED. The reset value is 0.

Table 3-89 HSCTLR bit assignments (continued)

Bits	Name	Function

[7]	ITD	IT Disable:
		`0` The IT instruction functionality is available at EL2.
		`1` All encodings of the IT instruction with hw1[3:0]!=1000 are UNDEFINED at EL2. All encodings of the subsequent instruction with the following values for hw1 are UNDEFINED at EL2:
		`0x11xxxxxxxxxxxxxx` All 32-bit instructions, B(2), B(1), Undefined, SVC, Load/Store multiple
		`0x1x11xxxxxxxxxxx` Miscellaneous 16-bit instructions
		`0x1x10xxxxxxxxx` `ADD Rd, PC, #imm`
		`0x01001xxxxxxxxxxx` `LDR Rd, [PC, #imm]`
		`0x0101xxx1111xxx` `ADD(4), CMP(3), MOV, BX pc, BLX pc`
		`0x010001xx1xxxx111` `ADD(4), CMP(3), MOV`
		The reset value is 0.
		Though the Cortex-R52 processor supports this functionality, it is deprecated in Armv8.

[6]	-	Reserved, RES0.
[5]	CP15BEN	CP15* Barrier enable. Enables use of the CP15DMB, CP15DSB, and CP15ISB barrier operations at EL2: `0` CP15* barrier operations disabled at EL2. Their encodings are UNDEFINED. `1` CP15* barrier operations enabled at EL2. The reset value is `1`.
[4:3]	-	Reserved, RES1.
[2]	C	Cache enable. This is an enable bit for data cache at EL2: `0` Data cache disabled at EL2. `1` Data cache enabled at EL2. The field resets to 0.
[1]	A	Alignment check enable. This is the enable bit for Alignment fault checking: `0` Alignment fault checking disabled at EL2. `1` Alignment fault checking enabled at EL2. The field resets to 0.
[0]	M	MPU enable. This is a global enable bit for the EL2-controlled MPU: `0` EL2-controlled MPU disabled. `1` EL2-controlled MPU enabled. The field resets to 0.

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
```

3.3.54 Hyp System Trap Register

The HSTR controls trapping to EL2 of accesses at EL1 or lower, or the system register in the coproc== 1111 encoding space.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HSTR are trapped to EL2 when HSTR.T1 is set.

Configurations

This register is available in all build configurations.

Attributes

HSTR is a 32-bit register.

The following figure shows the HSTR bit assignments.

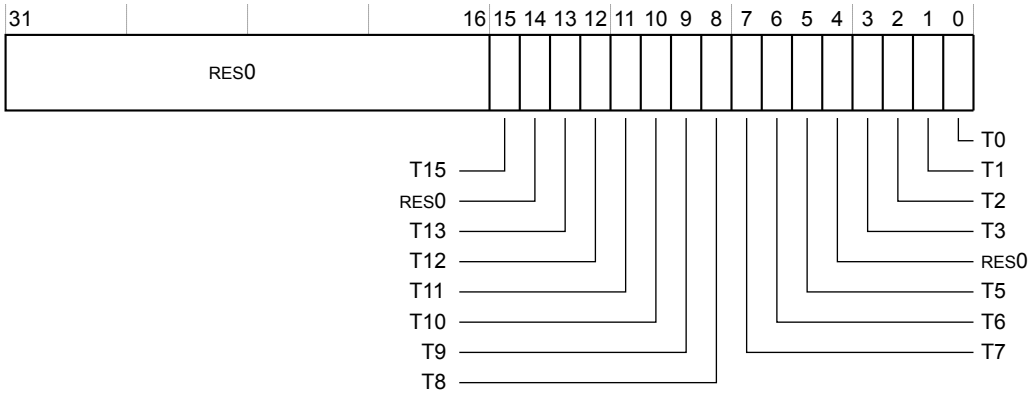

Figure 3-46 HSTR bit assignments

The following table shows the HSTR bit assignments.

Table 3-90 HSTR bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15]	T15	Trap primary register CRn = 15. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 15 to Hyp mode. The reset value is 0.
[14]	-	Reserved, RES0.

Table 3-90 HSTR bit assignments (continued)

Bits	Name	Function
[13]	T13	Trap primary register CRn = 13. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 13 to Hyp mode. The reset value is 0.
[12]	T12	Trap primary register CRn = 12. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 12 to Hyp mode. The reset value is 0.
[11]	T11	Trap primary register CRn = 11. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 11 to Hyp mode. The reset value is 0.
[10]	T10	Trap primary register CRn = 10. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 10 to Hyp mode. The reset value is 0.
[9]	T9	Trap primary register CRn = 9. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 9 to Hyp mode. The reset value is 0.
[8]	T8	Trap primary register CRn = 8. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 8 to Hyp mode. The reset value is 0.
[7]	T7	Trap primary register CRn = 7. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 7 to Hyp mode. The reset value is 0.
[6]	T6	Trap primary register CRn = 6. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 6 to Hyp mode. The reset value is 0.

Table 3-90 HSTR bit assignments (continued)

Bits	Name	Function
[5]	T5	Trap primary register CRn = 5. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 5 to Hyp mode. The reset value is 0.
[4]	-	Reserved, RES0.
[3]	T3	Trap primary register CRn = 3. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 3 to Hyp mode. The reset value is 0.
[2]	T2	Trap primary register CRn = 2. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 2 to Hyp mode. The reset value is 0.
[1]	T1	Trap primary register CRn = 1. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to primary register CRn = 1 to Hyp mode. The reset value is 0.
[0]	T0	Trap primary register CRn = 0. The possible values are: 0 Has no effect on accesses to non-debug system registers. 1 Trap valid EL1 or lower accesses to 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
```

3.3.55 Hyp Vector Base Address Register

The HVBAR holds the exception base address for any exception that is taken to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HVBAR are trapped to EL2 when HSTR.T12 is set.

Configurations

This register is available in all build configurations.

Attributes

HVBAR is a 32-bit register.

The following figure shows the HVBAR bit assignments.

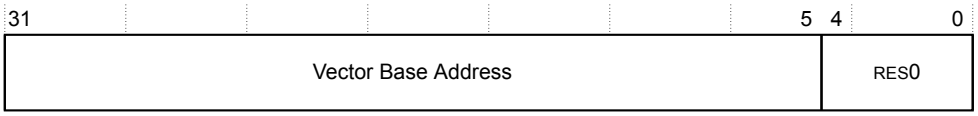

Figure 3-47 HVBAR bit assignments

The following table shows the HVBAR bit assignments.

Table 3-91 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 to EL2. Bits[4:0] of an exception vector are the exception offset. **CFGVECTABLEx[31:5]** determines the vector base address out of reset.
[4:0]	-	Reserved, RES0.

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

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
```

3.3.56 Hypervisor Reset Management Register

A write to the HRMR register requests a warm reset.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to HRMR are trapped to EL2 when HSTR.T12 is set.

Configurations

This register is available in all build configurations.

Attributes

HRMR is a 32-bit register.

The following figure shows the HRMR bit assignments.

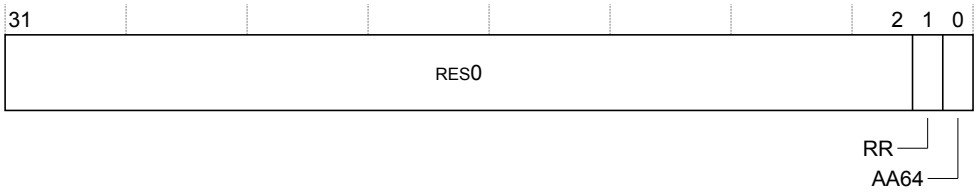

Figure 3-48 HRMR bit assignments

The following table shows the HRMR bit assignments.

Table 3-92 HRMR bit assignments

Bits	Name	Function
[31:2]	-	Reserved, RES0.
[1]	RR	Reset Request. Setting this bit to `1` requests a warm reset and asserts **WARMRSTREQx**.
[0]	AA64	RAZ/WI.

To access the HRMR:

```
MRC p15, 4, <Rt>, c12, c0, 2 ; Read HRMR into Rt
MCR p15, 4, <Rt>, c12, c0, 2 ; Write Rt to HRMR
```

3.3.57 Instruction Cache Error Record Registers 0 and 1

The IMP_ICERR0 and IMP_ICERR1 registers indicate the instruction cache RAM and the cache index of a detected instruction cache error.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW/RO	RW

The IMP_ICERR0 and IMP_ICERR1 registers are accessible as RW from EL1 when HACTLR.ERR is set, RO from EL1 when HACTLR.ERR is clear.

Traps and enables

EL1 write accesses to these registers are trapped to EL2 when HACTLR.ERR is 0.

Configurations

These registers are available in cores configured with an instruction cache.

Attributes

IMP_ICERR0 and IMP_ICERR1 are 32-bit registers.

The following figure shows IMP_ICERR0 and IMP_ICERR1 bit assignments.

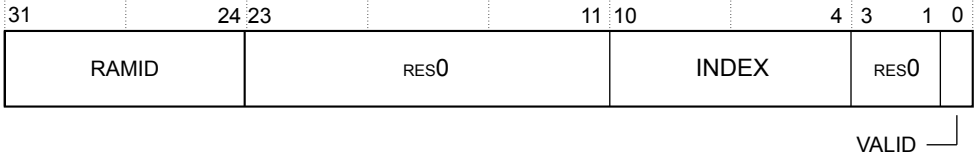

Figure 3-49 IMP_ICERR0 and IMP_ICERR1 bit assignments

The following table shows the IMP_ICERR0 and IMP_ICERR1 bit assignments.

Table 3-93 IMP_ICERR0 and IMP_ICERR1 bit assignments

Bits	Name	Function	Notes
[31:24]	RAMID	Indicates the RAM bank: **RAMID[7]** Tag RAM, bank 3. **RAMID[6]** Tag RAM, bank 2. **RAMID[5]** Tag RAM, bank 1. **RAMID[4]** Tag RAM, bank 0. **RAMID[3]** Data RAM, bank 3. **RAMID[2]** Data RAM, bank 2. **RAMID[1]** Data RAM, bank 1. **RAMID[0]** Data RAM, bank 0.	-
[23:11]	-	Reserved, RES0.	-
[10:4]	INDEX	Instruction cache index. This is address bits [12:6] appropriately masked with the cache size.	-
[3:1]	-	Reserved, RES0.	Writes to these registers are unique. If bit[1] and bit[0] are set to 0, and if the register holds information about a correctable error, then the register is cleared. If bit[1] is set to 1 and bit[0] is set to 0, then the register is always cleared.
[0]	VALID	Register contents are valid.	

To access the IMP_ICERR0:

```
MRC p15,2,<Rt>,c15,c0,0 ; Read IMP_ICERR0 into Rt
MCR p15,2,<Rt>,c15,c0,0 ; Write Rt to IMP_ICERR0
```

To access the IMP_ICERR1:

```
MRC p15,2,<Rt>,c15,c0,1 ; Read IMP_ICERR1 into Rt
MCR p15,2,<Rt>,c15,c0,1 ; Write Rt to IMP_ICERR1
```

3.3.58 Instruction Fault Address Register

The IFAR holds the faulting address that caused a synchronous Prefetch Abort exception.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

IFAR reads and writes from EL1 are trapped to EL2 according to HCR.TRVM and HCR.TVM respectively.

Configurations

This register is available in all build configurations.

Attributes

IFAR is a 32-bit register.

The following figure shows the IFAR bit assignments.

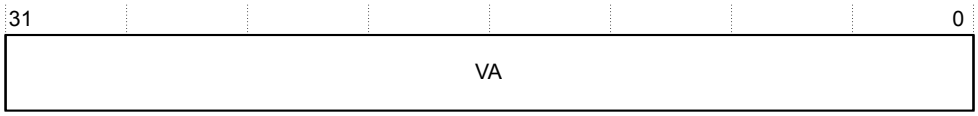

Figure 3-50 IFAR bit assignments

The following table shows the IFAR bit assignments.

Table 3-94 IFAR bit assignments

Bits	Name	Function
[31:0]	VA	The 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
```

3.3.59 Instruction Fault Status Register

The IFSR holds status information about the last instruction fault.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

IFSR reads and writes from EL1 are trapped to EL2 according to HCR.TRVM and HCR.TVM respectively.

Configurations

This register is available in all build configurations.

Attributes

IFSR is a 32-bit register.

The following figure shows the IFSR bit assignments.

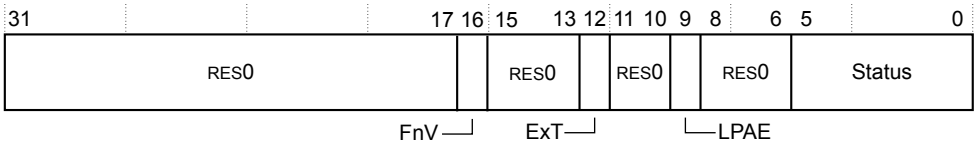

Figure 3-51 IFSR bit assignments

The following table shows the IFSR bit assignments.

Table 3-95 IFSR bit assignments

Bits	Name	Function
[31:17]	-	Reserved, RES0.
[16]	FnV	This field is RES0.
[15:13]	-	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	This field is RES1.
[8:6]	-	Reserved, RES0.
[5:0]	Status	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved. `0b000100` Translation fault, level 0. `0b001100` Permission fault, level 0. `0b010000` Synchronous external abort. `0b011000` Synchronous parity error on memory access.[be] `0b100001` Alignment fault. `0b100010` Debug event.

Note

If a Data Abort exception is generated by an instruction cache maintenance, 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
```

3.3.60 Instruction Set Attribute Register 0

The ID_ISAR0 provides information about the instruction sets implemented by the processor.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

The ID_ISAR0 must be interpreted with ID_ISAR1, ID_ISAR2, ID_ISAR3, ID_ISAR4, and ID_ISAR5. See:

- *3.3.61 Instruction Set Attribute Register 1* on page 3-146.
- *3.3.62 Instruction Set Attribute Register 2* on page 3-147.
- *3.3.63 Instruction Set Attribute Register 3* on page 3-149.
- *3.3.64 Instruction Set Attribute Register 4* on page 3-151.
- *3.3.65 Instruction Set Attribute Register 5* on page 3-152.

Traps and enables

EL1 read accesses to ID_ISAR0 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_ISAR0 is a 32-bit register.

The following figure shows the ID_ISAR0 bit assignments.

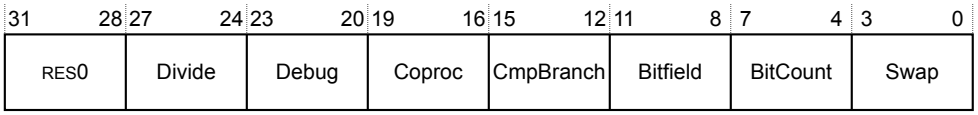

Figure 3-52 ID_ISAR0 bit assignments

The following table shows the ID_ISAR0 bit assignments.

Table 3-96 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 system register access instructions: 0x0 None implemented, except for separately attributed by the architecture including (coproc==0b1110), (coproc==0b1111) 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 3-96 ID_ISAR0 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:

```
MRC p15, 0, <Rt>, c0, c2, 0 ; Read ID_ISAR0 into Rt
```

3.3.61 Instruction Set Attribute Register 1

The ID_ISAR1 provides information about the instruction sets implemented by the processor.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

The ID_ISAR1 must be interpreted with ID_ISAR0, ID_ISAR2, ID_ISAR3, ID_ISAR4 and ID_ISAR5. See:

- *3.3.60 Instruction Set Attribute Register 0* on page 3-144.
- *3.3.62 Instruction Set Attribute Register 2* on page 3-147.
- *3.3.63 Instruction Set Attribute Register 3* on page 3-149.
- *3.3.64 Instruction Set Attribute Register 4* on page 3-151.
- *3.3.65 Instruction Set Attribute Register 5* on page 3-152.

Traps and enables

EL1 read accesses to ID_ISAR1 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_ISAR1 is a 32-bit register.

The following figure shows the ID_ISAR1 bit assignments.

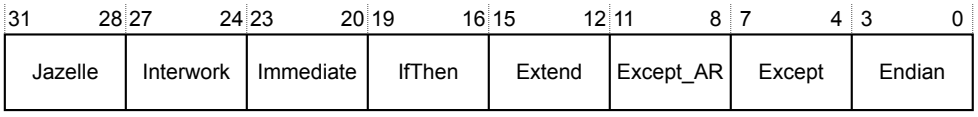

Figure 3-53 ID_ISAR1 bit assignments

The following table shows the ID_ISAR1 bit assignments.

Table 3-97 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 R profile exception-handling instructions: `0x1` The `SRS` and `RFE` instructions, and the R 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
```

3.3.62 Instruction Set Attribute Register 2

The ID_ISAR2 provides information about the instruction sets implemented by the processor.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

The ID_ISAR2 must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR3, ID_ISAR4 and ID_ISAR5. See:

- *3.3.60 Instruction Set Attribute Register 0* on page 3-144.
- *3.3.61 Instruction Set Attribute Register 1* on page 3-146.
- *3.3.63 Instruction Set Attribute Register 3* on page 3-149.
- *3.3.64 Instruction Set Attribute Register 4* on page 3-151.
- *3.3.65 Instruction Set Attribute Register 5* on page 3-152.

Traps and enables

EL1 read accesses to ID_ISAR2 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_ISAR2 is a 32-bit register.

The following figure shows the ID_ISAR2 bit assignments.

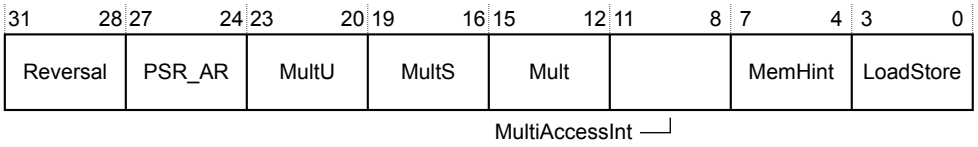

Figure 3-54 ID_ISAR2 bit assignments

The following table shows the ID_ISAR2 bit assignments.

Table 3-98 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 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.

Table 3-98 ID_ISAR2 bit assignments (continued)

Bits	Name	Function
[23:20]	MultU	Indicates the implemented advanced unsigned Multiply instructions: `0x2` The `UMULL` and `UMLAL` instructions. The `UMAAL` instruction.
[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: `0x1` LDM and STM instructions are restartable.
[7:4]	MemHint	Indicates the implemented memory hint instructions: `0x4` The `PLD` instruction. The `PLI` instruction. The `PLDW` instruction. **Note** The Cortex-R52 processor: • Treats the `PLI` instruction as a NOP. • Triggers a cache linefill when there is a `PLD` or `PLDW` instruction unless the line is already in the cache or the location is considered non-cacheable.
[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
```

3.3.63 Instruction Set Attribute Register 3

The ID_ISAR3 provides information about the instruction sets implemented by the processor.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

The ID_ISAR3 must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR2, ID_ISAR4, and ID_ISAR5. See:

- *3.3.60 Instruction Set Attribute Register 0* on page 3-144.
- *3.3.61 Instruction Set Attribute Register 1* on page 3-146.
- *3.3.62 Instruction Set Attribute Register 2* on page 3-147.
- *3.3.64 Instruction Set Attribute Register 4* on page 3-151.
- *3.3.65 Instruction Set Attribute Register 5* on page 3-152.

Traps and enables

EL1 read accesses to ID_ISAR3 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_ISAR3 is a 32-bit register.

The following figure shows the ID_ISAR3 bit assignments.

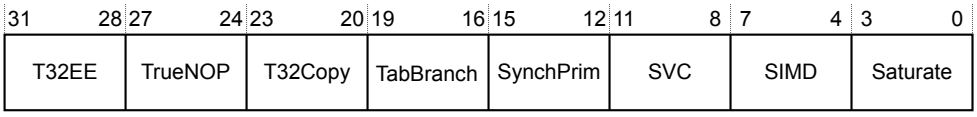

Figure 3-55 ID_ISAR3 bit assignments

The following table shows the ID_ISAR3 bit assignments.

Table 3-99 ID_ISAR3 bit assignments

Bits	Name	Function
[31:28]	T32EE	Indicates the implemented 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]	T32Copy	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 3-99 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
```

3.3.64 Instruction Set Attribute Register 4

The ID_ISAR4 provides information about the instruction sets implemented by the processor.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

The ID_ISAR4 must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR2, ID_ISAR3, and ID_ISAR5. See:

- *3.3.60 Instruction Set Attribute Register 0* on page 3-144.
- *3.3.61 Instruction Set Attribute Register 1* on page 3-146.
- *3.3.62 Instruction Set Attribute Register 2* on page 3-147.
- *3.3.63 Instruction Set Attribute Register 3* on page 3-149.
- *3.3.65 Instruction Set Attribute Register 5* on page 3-152.

Traps and enables

EL1 read accesses to ID_ISAR4 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_ISAR4 is a 32-bit register.

The following figure shows the ID_ISAR4 bit assignments.

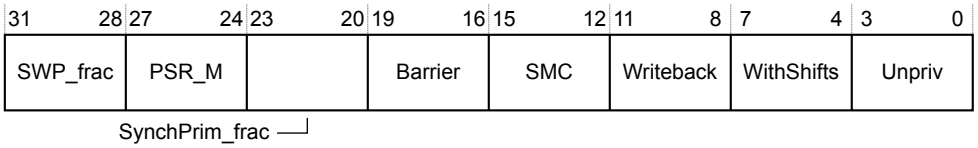

Figure 3-56 ID_ISAR4 bit assignments

The following table shows the ID_ISAR4 bit assignments.

Table 3-100 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: `0x0` The `SMC` instruction not supported.
[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
```

3.3.65 Instruction Set Attribute Register 5

The ID_ISAR5 provides information about the instruction sets that the processor implements.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

The ID_ISAR5 must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR2, ID_ISAR3, and ID_ISAR4. See:

- *3.3.60 Instruction Set Attribute Register 0* on page 3-144.
- *3.3.61 Instruction Set Attribute Register 1* on page 3-146.
- *3.3.62 Instruction Set Attribute Register 2* on page 3-147.
- *3.3.63 Instruction Set Attribute Register 3* on page 3-149.
- *3.3.64 Instruction Set Attribute Register 4* on page 3-151.

Traps and enables

EL1 read accesses to ID_ISAR5 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_ISAR5 is a 32-bit register.

The following figure shows the ID_ISAR5 bit assignments.

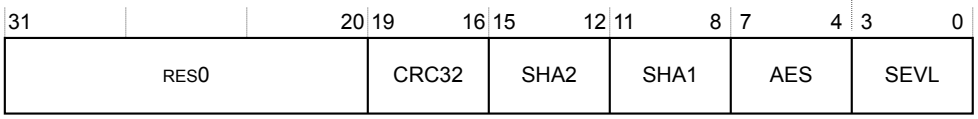

Figure 3-57 ID_ISAR5 bit assignments

The following table shows the ID_ISAR5 bit assignments.

Table 3-101 ID_ISAR5 bit assignments

Bits	Name	Function
[31:20]	-	Reserved, RES0.
[19:16]	CRC32	Indicates whether CRC32 instructions are implemented in AArch32 state: 0x1 CRC32 instructions are implemented.
[15:12]	SHA2	Indicates whether SHA2 instructions are implemented in AArch32 state: 0x0 Cryptography Extensions are not implemented.
[11:8]	SHA1	Indicates whether SHA1 instructions are implemented in AArch32 state: 0x0 Cryptography Extensions are not implemented.
[7:4]	AES	Indicates whether AES instructions are implemented in AArch32 state: 0x0 Cryptography Extensions are not implemented.
[3:0]	SEVL	Indicates whether the SEVL instruction is implemented: 0x1 SEVL implemented to send event local.

To access the ID_ISAR5:

```
MRC p15,0,<Rt>,c0,c2,5 ; Read ID_ISAR5 into Rt
```

3.3.66 Interrupt Monitoring Register

The IMP_INTMONR provides access to counters for detecting when the core has entered hypervisor mode, but the interrupt masks in the current PSR have not been cleared within a given time limit. IMP_INTMONR counts cycles in which interrupt masks are set for physical system errors, physical interrupts, and physical fast interrupts, and signals a timeout on **ERREVENTx[18]** if a limit is exceeded. IMP_INTMONR does not require there to be a pended interrupt being masked. IMP_INTMONR supports two modes of operation, watchdog mode, and maximum value monitor mode. For both modes of operation the counter increments when any of the masks are set so corresponding interrupts cannot be taken. The counter resets to zero when all enabled interrupts can be taken. In watchdog mode, an event is generated when the counter reaches IMP_INTMONR.MAXCYCLESBY16×16. In maximum value monitor mode, IMP_INTMONR.MAXCYCLESBY16 holds the maximum value the counter has reached, before resetting, divided by 16.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW/RO	RW

RW from EL1 when HACTLR.INTMONR is set, RO from EL1 when HACTLR.INTMONR is clear and inaccessible from EL0.

Traps and enables

EL1 write accesses to INTMONR are trapped to EL2 when HACTLR.INTMONR is 0.

Configurations

This register is available in all build configurations.

Attributes

IMP_INTMONR is a 32-bit register.

The following figure shows the IMP_INTMONR bit assignments.

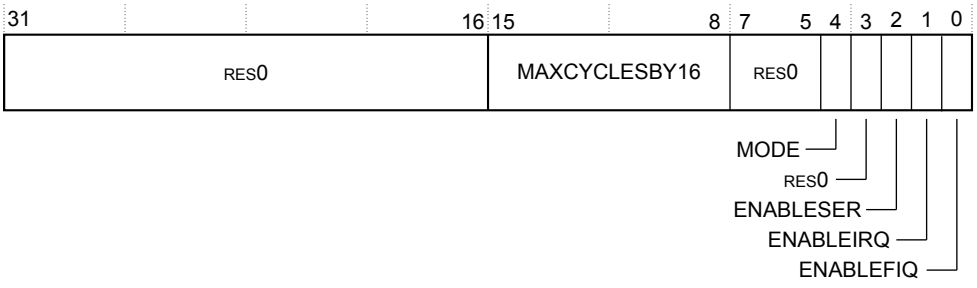

Figure 3-58 IMP_INTMONR bit assignments

The following table shows the IMP_INTMONR bit assignments.

Table 3-102 IMP_INTMONR bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15:8]	MAXCYCLESBY16	Maximum count divided by 16.
[7:5]	-	Reserved, RES0.
[4]	MODE	Sets the operation mode of the counter: 0 Watchdog mode 1 Maximum value monitor mode.
[3]	-	Reserved, RES0.
[2]	ENABLESER	Enable counting of cycles in which physical system errors are masked.
[1]	ENABLEIRQ	Enable counting of physical interrupts that cannot be taken.
[0]	ENABLEFIQ	Enable counting of fast interrupts that cannot be taken.

To access the IMP_INTMONR:

```
MRC p15, 1, <Rt>, c15, c3, 4 ; Read IMP_INTMONR into Rt
MCR p15, 1, <Rt>, c15, c3, 4 ; Write Rt to IMP_INTMONR
```

3.3.67 Interrupt Status Register

The ISR 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	EL2
-	RO	RO

Traps and enables

EL1 read accesses to ISR are trapped to EL2 when HSTR.T12 is set.

Configurations

This register is available in all build configurations.

Attributes

ISR is a 32-bit register.

The following figure shows the ISR bit assignments.

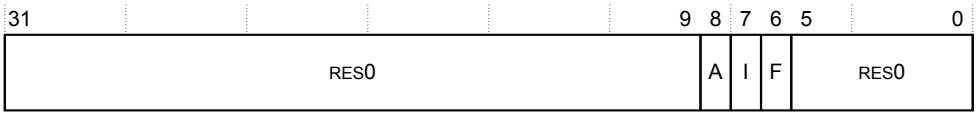

Figure 3-59 ISR bit assignments

The following table shows the ISR bit assignments.

Table 3-103 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, 0 ; Read ISR into Rt
```

3.3.68 Invalidate All Register

IMP_CDBGDCI is always RES0.

3.3.69 Main ID Register

The MIDR 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	EL2
-	RO	RO

Traps and enables

EL1 accesses to MIDR are trapped to EL2 when HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

MIDR is a 32-bit register.

The following figure shows the MIDR bit assignments.

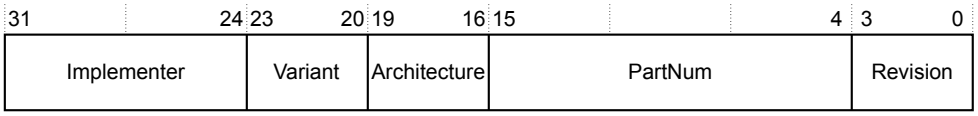

Figure 3-60 MIDR bit assignments

The following table shows the MIDR bit assignments.

Table 3-104 MIDR bit assignments

Bits	Name	Function
[31:24]	Implementer	Indicates the implementer code. This value is: `0x41` 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` r1p3.
[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: `0xD13` Cortex-R52 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: `0x3` r1p3.

To access the MIDR:

```
MRC p15, 0, <Rt>, c0, c0, 0 ; Read MIDR into Rt
```

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

3.3.70 Memory Attribute Indirection Registers 0 and 1

The MAIR0 and MAIR1 provide the memory attribute encodings corresponding to the possible AttrIndx values in a Protection Region Limit Address Register. The register gives the value for memory accesses from states other than Hyp mode.

Usage constraints

These registers are accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

EL1 write accesses to these registers are trapped to EL2 when HCR.TVM is set. EL1 read accesses are trapped to EL2 when HCR.TRVM is set. EL1 accesses are trapped to EL2 when HSTR.T10 is set.

Configurations

These registers are available in all build configurations.

Attributes

MAIR0 and MAIR1 are 32-bit registers.

The following figure shows the MAIR0 and MAIR1 bit assignments.

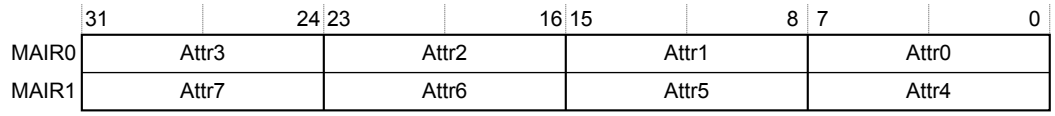

Figure 3-61 MAIR0 and MAIR1 bit assignments

The following table shows the MAIR0 and MAIR1 bit assignments.

Table 3-105 MAIR0 and MAIR1 bit assignments

Bits	Name	Function
[7:0]	Attr*m*[bf]	The memory attribute encoding for an AttrIndx[2:0] entry in a Protection Region Limit Address Register, 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 3-106 Attr<n>[7:4] bit assignments

Bits	Description
`0b0000`	Device memory. See The following table for the type of Device memory.
`0b00RW`, RW not `00`	Normal Memory, Outer Write-Through transient[bg].
`0b0100`	Normal Memory, Outer Non-Cacheable.
`0b01RW`, RW not `00`	Normal Memory, Outer Write-Back transient[bg].
`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].

Table 3-107 Attr<n>[3:0] bit assignments

Bits	Description when Attr<n>[7:4] is 0000	Description 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

bf Where m is 0-7.
bg The transient hint is ignored.

The following table shows the encoding of the R and W bits that are used, in some Attr<n> encodings in *Table 3-106 Attr<n>[7:4] bit assignments* on page 3-158 and *Table 3-107 Attr<n>[3:0] bit assignments* on page 3-158 to define the Read-Allocate and Write-Allocate policies respectively.

Table 3-108 Encoding of R and W bits in some Attr*m* fields

R or W	Description
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
```

3.3.71 Memory Model Feature Register 0

The ID_MMFR0 provides information about the memory model and memory management support.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Must be interpreted with ID_MMFR1, ID_MMFR2, and ID_MMFR3. See:

- *3.3.72 Memory Model Feature Register 1* on page 3-160.
- *3.3.73 Memory Model Feature Register 2* on page 3-161.
- *3.3.74 Memory Model Feature Register 3* on page 3-163.

Traps and enables

EL1 read accesses to ID_MMFR0 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_MMFR0 is a 32-bit register.

The following figure shows the ID_MMFR0 bit assignments.

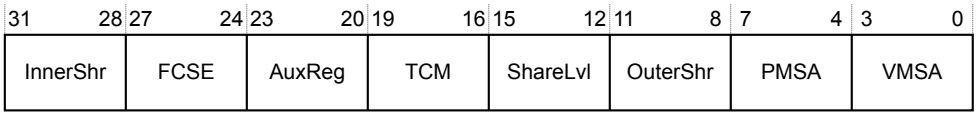

Figure 3-62 ID_MMFR0 bit assignments

The following table shows the ID_MMFR0 bit assignments.

Table 3-109 ID_MMFR0 bit assignments

Bits	Name	Function
[31:28]	InnerShr	Indicates the innermost shareability domain implemented: `0x0` Implemented as non-cacheable.
[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: `0x1` TCMs and associated DMAs 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: `0x0` Implemented as non-cacheable.
[7:4]	PMSA	Indicates support for a PMSA: `0x4` Support for Armv8-R base and limit PMSA.
[3:0]	VMSA	Indicates support for a *Virtual Memory System Architecture* (VMSA). `0x0` VMSA not supported.

To access the ID_MMFR0:

```
MRC p15,0,<Rt>,c0,c1,4 ; Read ID_MMFR0 into Rt
```

3.3.72 Memory Model Feature Register 1

The ID_MMFR1 provides information about the memory model and memory management support.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Must be interpreted with ID_MMFR0, ID_MMFR2, and ID_MMFR3. See:

- *3.3.71 Memory Model Feature Register 0* on page 3-159.
- *3.3.73 Memory Model Feature Register 2* on page 3-161.
- *3.3.74 Memory Model Feature Register 3* on page 3-163.

Traps and enables

EL1 read accesses to ID_MMFR1 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_MMFR1 is a 32-bit register.

The following figure shows the ID_MMFR1 bit assignments.

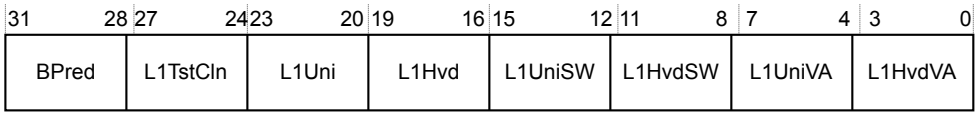

Figure 3-63 ID_MMFR1 bit assignments

The following table shows the ID_MMFR1 bit assignments. This table should be read in conjunction with the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Table 3-110 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.
[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
```

3.3.73 Memory Model Feature Register 2

The ID_MMFR2 provides information about the implemented memory model and memory management support.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Must be interpreted with ID_MMFR0, ID_MMFR1, and ID_MMFR3. See:

- *3.3.71 Memory Model Feature Register 0* on page 3-159.
- *3.3.72 Memory Model Feature Register 1* on page 3-160.
- *3.3.74 Memory Model Feature Register 3* on page 3-163.

Traps and enables

EL1 read accesses to ID_MMFR2 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_MMFR2 is a 32-bit register.

The following figure shows the ID_MMFR2 bit assignments.

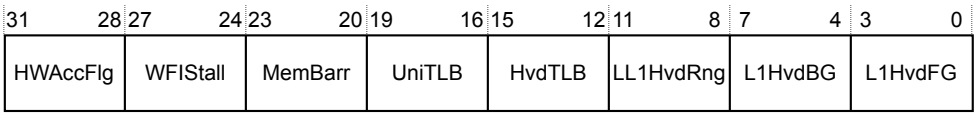

Figure 3-64 ID_MMFR2 bit assignments

The following table shows the ID_MMFR2 bit assignments.

Table 3-111 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* (CP15DSB). • *Instruction Synchronization Barrier* (CP15ISB). • *Data Memory Barrier* (CP15DMB).
[19:16]	UniTLB	Unified TLB. Indicates the supported TLB maintenance operations, for a unified TLB implementation. `0x0` Not supported.
[15:12]	HvdTLB	Harvard TLB. Indicates the supported TLB maintenance operations, for a Harvard TLB implementation: `0x0` Not supported.

Table 3-111 ID_MMFR2 bit assignments (continued)

Bits	Name	Function
[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
```

3.3.74 Memory Model Feature Register 3

The ID_MMFR3 provides information about the memory model and memory management support.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Must be interpreted with ID_MMFR0, ID_MMFR1, and ID_MMFR2. See:

- *3.3.71 Memory Model Feature Register 0* on page 3-159.
- *3.3.72 Memory Model Feature Register 1* on page 3-160.
- *3.3.73 Memory Model Feature Register 2* on page 3-161.

Traps and enables

EL1 read accesses to ID_MMFR3 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_MMFR3 is a 32-bit register.

The following figure shows the ID_MMFR3 bit assignments.

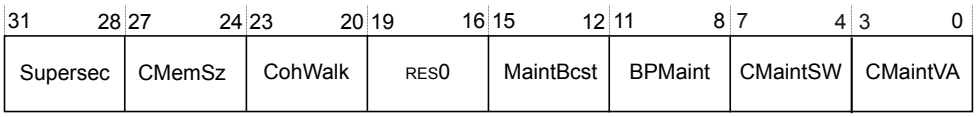

Figure 3-65 ID_MMFR3 bit assignments

The following table shows the ID_MMFR3 bit assignments. This table should be read in conjunction with the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Table 3-112 ID_MMFR3 bit assignments

Bits	Name	Function
[31:28]	Supersec	Supersections. Indicates support for supersections: `0xF` Supersections not supported.
[27:24]	CMemSz	Cached Memory Size. Indicates the size of physical memory supported by the processor caches: `0x0` 4GB, corresponding to a 32-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 and branch predictor operations are broadcast: `0x2` Cache 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.
[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.
[3:0]	CMaintVA	Cache maintenance by VA. Indicates the supported cache maintenance operations by VA. `0x1` Supported hierarchical cache maintenance operations by VA.

To access the ID_MMFR3:

```
MRC p15, 0, <Rt>, c0, c1, 7 ; Read ID_MMFR3 into Rt
```

3.3.75 Memory Model Feature Register 4

The ID_MMFR4 provides information about the implemented memory model and memory management support.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and enables

EL1 read accesses to ID_MMFR4 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_MMFR4 is a 32-bit register.

The following figure shows the ID_MMFR4 bit assignments.

Figure 3-66 ID_MMFR4 bit assignments

The following table shows the ID_MMFR4 bit assignments.

Table 3-113 ID_MMFR4 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RAZ.
[7:4]	AC2	Indicates the extension of the ACTLR and HACTLR registers using ACTLR2 and HACTLR2: `0b0001` ACTLR2 and HACTLR2 are implemented.
[3:0]	-	Reserved, RAZ.

To access the ID_MMFR4:

```
MRC p15, 0, <Rt>, c0, c2, 6 ; Read ID_MMFR4 into Rt
```

3.3.76 Memory Protection Control Register

The IMP_MEMPROTCTLR provides control to enable and disable ECC protection for the Flash interface, L1 cache memory, and TCM memory.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

EL1 accesses to the IMP_MEMPROTCTLR are trapped to EL2 when HCR.TIDCP is set.

Configurations

This register is available in all build configurations.

Attributes

IMP_MEMPROTCTLR is a 32-bit register.

The following figure shows the IMP_MEMPROTCTLR bit assignments.

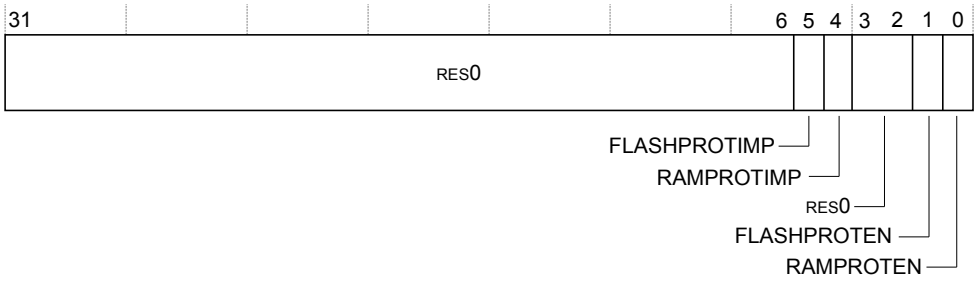

Figure 3-67 IMP_MEMPROTCTLR bit assignments

The following table shows the IMP_MEMPROTCTLR bit assignments.

Table 3-114 IMP_MEMPROTCTLR bit assignments

Bits	Name	Function
[31:6]	-	Reserved, RES0.
[5]	FLASHPROTIMP	Flash protection implemented. Indicates the value read from the **CFGFLASHPROTIMP** configuration signal. This bit is RO/WI.
[4]	RAMPROTIMP	RAM protection implemented. This bit is RO/WI.
[3:2]	-	Reserved, RES0.
[1]	FLASHPROTEN	Enable Flash interface protection. `0` Flash protection disabled. `1` Flash protection enabled. Input signal **CFGFLASHPROTEN** defines the reset value. **Note** Flash protection FLASHPROTEN is always disabled when FLASHPROTIMP is `0`, but is always enabled (though not necessarily showing the value of `1`) when signal integrity protection is implemented.
[0]	RAMPROTEN	Enable TCM and L1 cache RAM protection. `0` RAM protection disabled. `1` RAM protection enabled. Input signal **CFGRAMPROTEN** defines the reset value. **Note** The RAMPROTEN value is ignored (the design behaves as if it is 0) when RAMPROTIMP is 0.

To access the IMP_MEMPROTCTLR:

```
MRC p15, 1, <Rt>, c9, c1, 2 ; Read IMP_MEMPROTCTLR into Rt
MCR p15, 1, <Rt>, c9, c1, 2 ; Write Rt to IMP_MEMPROTCTLR
```

3.3.77 MPU Type Register

The MPUIR indicates the number of programmable memory regions implemented by the EL1-controlled MPU.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and enables

EL1 read accessed to MPUIR are trapped to EL2 when HCR.TID1 is set.

Configurations

This register is available in all build configurations.

Attributes

MPUIR is a 32-bit register.

The following figure shows the MPUIR bit assignments.

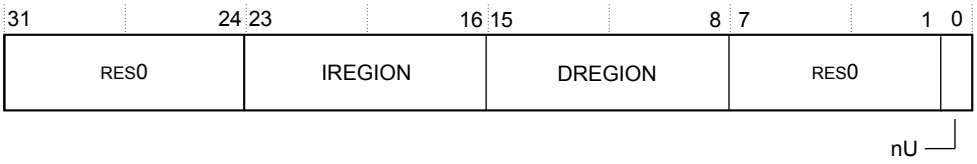

Figure 3-68 MPUIR bit assignments

The following table shows the MPUIR bit assignments.

Table 3-115 MPUIR bit assignments

Bits	Name	Function
[31:24]	-	This field is RES0.
[23:16]	IREGION	This field is RES0.
[15:8]	DREGION	Indicates the number of programmable memory regions implemented by the EL1-controlled MPU. This value can be 16, 20, or 24.
[7:1]	-	This field is RES0.
[0]	nU	Not Unified MPU. Indicates whether the MPU implements a unified memory map. `0` Unified memory map.

To access the MPUIR:

```
MRC p15, 0, <Rt>, c0, c0, 4 ; Read MPUIR into Rt
```

3.3.78 Multiprocessor Affinity Register

The MPIDR provides core identification mechanism for scheduling purposes in a cluster.

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 the external EDDEVAFF0 register. The register is available in all build configurations.

Attributes

MPIDR is a 32-bit register.

The following figure shows the MPIDR bit assignments.

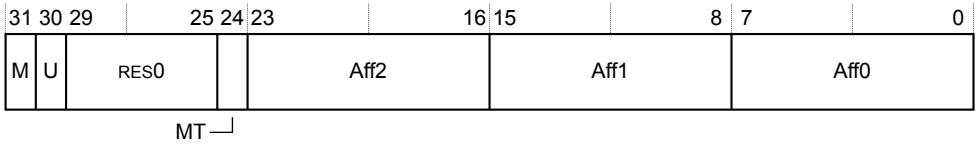

Figure 3-69 MPIDR bit assignments

The following table shows the MPIDR bit assignments.

Table 3-116 MPIDR bit assignments

Bits	Name	Function
[31]	M	Reserved, 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 PEs that are implemented using a multi-threading type approach. This value is: `0` Performance of PEs at the lowest affinity level is largely independent.
[23:16]	Aff2	Affinity level 2. The least significant affinity field, for this PE in the system. Indicates the value read from the **CFGMPIDRAFF2** configuration signal.
[15:8]	Aff1	Affinity level 1. The intermediate affinity level field, for this PE in the system. Indicates the value read from the **CFGMPIDRAFF1** configuration signal.
[7:0]	Aff0	Affinity level 0. The most significant affinity level field, for this PE in the system.

To access the MPIDR:

```
MRC p15, 0, <Rt>, c0, c0, 5 ; Read MPIDR into Rt
```

The EDDEVAFF0 can be accessed through the memory-mapped interface and the external debug interface, offset `0xFA8`.

3.3.79 Non-Secure Access Control Register

The Cortex-R52 processor does not implement TrustZone technology. Any read of the NSACR from EL2 or from EL1 returns a value of `0x00000C00`.

3.3.80 Peripheral Port Region Register

The IMP_PERIPHPREGIONR indicates the base address and size of the peripheral port region. Provides control to enable and disable the peripheral port.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW/RO	RW

Traps and enables

EL1 accesses to IMP_PERIPHPREGIONR are trapped to EL2 when HCR.TIDCP is set. EL1 writes to IMP_PERIPHPREGIONR are trapped to EL2 when HACTLR.PERIPHPREGIONR is 0.

Configurations

This register is available in all build configurations.

Attributes

IMP_PERIPHPREGIONR is a 32-bit register.

The following figure shows the IMP_PERIPHPREGIONR bit assignments.

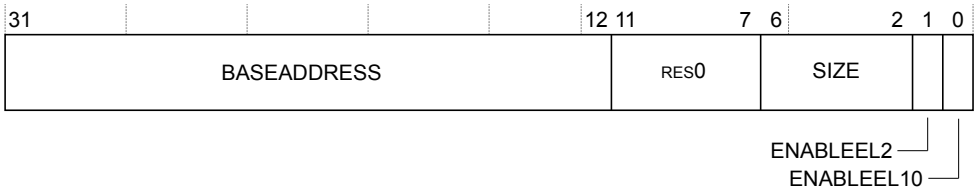

Figure 3-70 IMP_PERIPHPREGIONR bit assignments

The following table shows the IMP_PERIPHPREGIONR bit assignments.

Table 3-117 PERIPHPREGIONR bit assignments

Bits	Name	Function
[31:12]	BASEADDRESS	Peripheral port region base address. This value is set by the input signal **CFGLLPPBASEADDR[31:12]**.
[11:7]	-	Reserved, RES0.

Table 3-117 PERIPHPREGIONR bit assignments (continued)

Bits	Name	Function
[6:2]	SIZE	Indicates the peripheral port region size: `0b00000` No peripheral port region. `0b00011` 4KB. `0b00100` 8KB. `0b00101` 16KB. `0b00110` 32KB. `0b00111` 64KB. `0b01000` 128KB. `0b01001` 256KB. `0b01010` 512KB. `0b01011` 1MB. `0b01100` 2MB. `0b01101` 4MB. This value is set by the input signals **CFGLLPPSIZE[3:0]** and **CFGLLPPIMP**.
[1]	ENABLEEL2	Enable peripheral port at EL2. This bit resets to 0.
[0]	ENABLEEL10	Enable peripheral port at EL1 and EL0. This bit resets to 0.

To access the IMP_PERIPHPREGIONR:

```
MRC p15, 0, <Rt>, c15, c0, 0 ; Read IMP_PERIPHPREGIONR into Rt
MCR p15, 0, <Rt>, c15, c0, 0 ; Write Rt to IMP_PERIPHPREGIONR
```

3.3.81 Physical Address Register

The PAR receives the Physical Address from any address translation operation. In the Cortex-R52 processor translation refers to a lookup in the MPU.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

EL1 accesses to PAR are trapped to EL2 when HSTR.T7 is set.

Configurations

There are two formats for this register which depends on the value of bit[0], F bit, of this register.

This register available in all build configurations.

Attributes

Both formats of the PAR register are 64 bits.

The following table shows the PAR bit assignments common to all formats.

Table 3-118 PAR bit assignments

Bits	Name	Function
[0]	F	Indicates whether the conversion completed successfully: `0` Address translation completed successfully. `1` Address translation aborted.

To access the PAR when accessing as a 32-bit register:

```
MRC p15, 0, <Rt>, c7, c4, 0 ; Read PAR[31:0] into Rt
MCR p15, 0, <Rt>, c7, c4, 0 ; Write Rt to PAR[31:0]. PAR[63:32] are unchanged
```

To access the PAR when accessing as a 64-bit register:

```
MRRC p15, 0, <Rt>, <Rt2>, c7 ; Read 64-bit PAR into Rt (low word) and Rt2 (high word)
MCRR p15, 0, <Rt>, <Rt2>, c7 ; Write Rt (low word) and Rt2 (high word) to 64-bit PAR
```

Physical Address Register (F==0)

The following figure shows the PAR bit assignments for `PAR.F==0`.

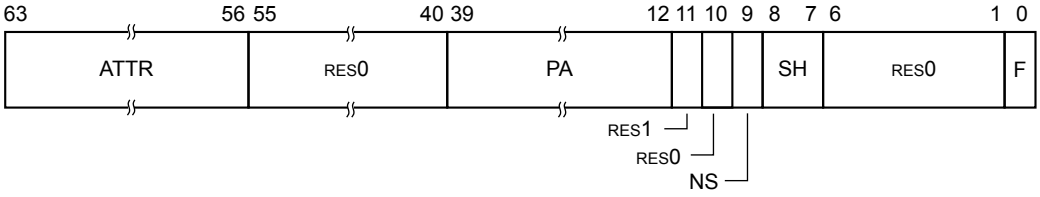

Figure 3-71 PAR bit assignments for PAR.F==0

The following table shows the PAR bit assignments for `PAR.F==0`.

Table 3-119 PAR bit assignments for PAR.F==0

Bits	Name	Function
[63:56]	ATTR	Memory attributes for the returned output address. This field uses the same encoding as the Attr<n> fields in MAIR0 and MAIR1.
[55:40]	-	Reserved, RES0.
[39:12]	PA	Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[31:12]. Bits[39:32] are RES0.
[11]	-	Reserved, RES1
[10]	-	Reserved, RES0.
[9]	NS	Non-secure: `1` No security extensions are implemented. This is the reset value.

Table 3-119 PAR bit assignments for PAR.F==0 (continued)

Bits	Name	Function
[8:7]	SH	Shareability attribute, for the returned output address. Permitted values are: `0b00` Non-shareable. `0b10` Outer Shareable. `0b11` Inner Shareable. The value `0b01` is reserved. **Note** This field returns the value `0b10` for: • Any type of Device memory. • Normal memory with both Inner Non-cacheable and Outer Non-cacheable attributes.
[6:1]	-	Reserved, RES0.

Physical Address Register (F==1)

The following figure shows the PAR bit assignments for `PAR.F==1`.

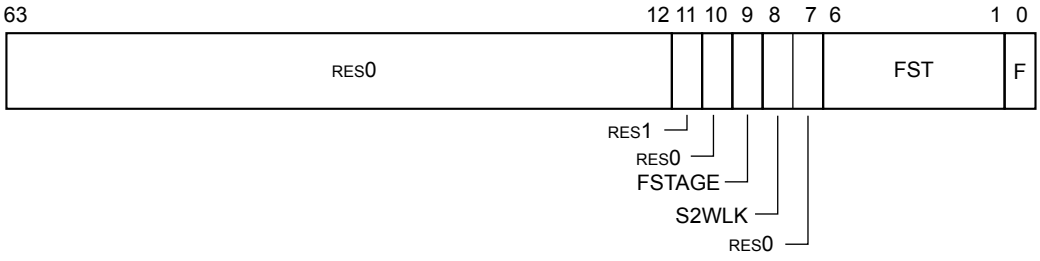

Figure 3-72 PAR bit assignments for PAR.F==1

The following table shows the PAR bit assignments for `PAR.F==1`.

Table 3-120 PAR bit assignments for PAR.F==1

Bits	Name	Function
[63:12]	-	Reserved, RES0
[11]	-	Reserved, RES1.
[10]	-	Reserved, RES0
[9]	FSTAGE	Indicates the translation stage at which the translation aborted: `0` Translation aborted because of a fault in the stage 1 translation. `1` Translation aborted because of a fault in the stage 2 translation.
[8]	S2WLK	In Cortex-R52, this field is always set to 0.
[7]	-	Reserved, RES0
[6:1]	FST	Fault status field. Values are as in the DFSR.STATUS and IFSR.STATUS fields.

3.3.82 Pin Options Register

The IMP_PINOPTR shows the tie-off values of the configuration signals used, to configure the processor, during integration of the processor with the rest of the SoC. This information is not otherwise visible to software.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

IMP_PINOPTR is a 32-bit register.

The following figure shows the IMP_PINOPTR bit assignments.

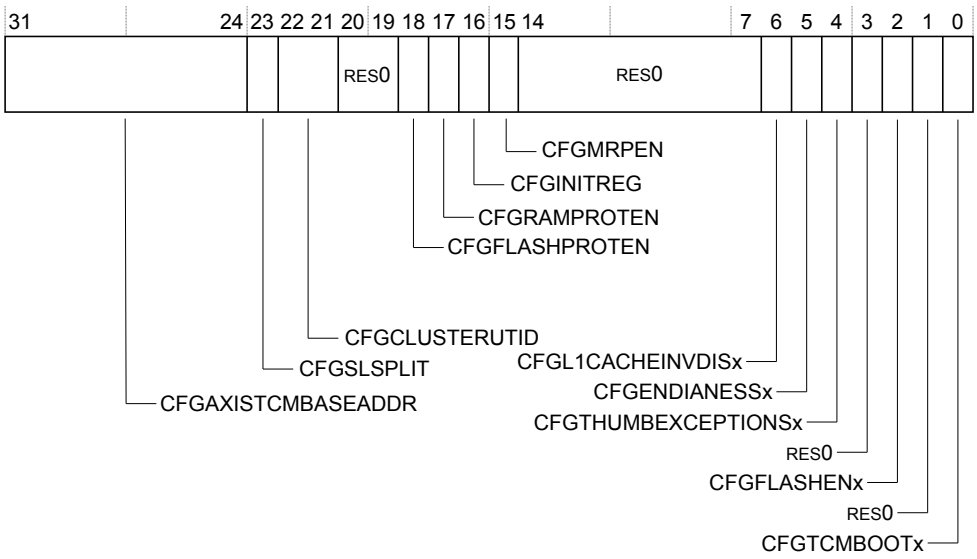

Figure 3-73 IMP_PINOPTR bit assignments

The following table shows the IMP_PINOPTR bit assignments.

Table 3-121 IMP_PINOPTR bit assignments

Bits	Name	Function
[31:24]	CFGAXISTCMBASEADDR	Value of the **CFGAXISTCMBASEADDR** signal.
[23]	CFGSLSPLIT	Value of the **CFGSLSPLIT** signal.
[22:21]	CFGCLUSTERUTID	Value of the **CFGCLUSTERUTID** signal.
[20:19]	-	Reserved, RES0.
[18]	CFGFLASHPROTEN	Value of the **CFGFLASHPROTEN** signal.

Table 3-121 IMP_PINOPTR bit assignments (continued)

Bits	Name	Function
[17]	CFGRAMPROTEN	Value of the **CFGRAMPROTEN** signal.
[16]	CFGINITREG	Value of the **CFGINITREG** signal.
[15]	CFGMRPEN	Value of the **CFGMRPEN** signal.
[14:7]	-	Reserved, RES0.
[6]	CFGL1CACHEINVDISx	Value of the **CFGL1CACHEINVDISx** signal.
[5]	CFGENDIANESSx	Value of the **CFGENDIANESSx** signal.
[4]	CFGTHUMBEXCEPTIONSx	Value of the **CFGTHUMBEXCEPTIONSx** signal.
[3]	-	Reserved, RES0.
[2]	CFGFLASHENx	Value of the **CFGFLASHENx** signal.
[1]	-	Reserved, RES0.
[0]	CFGTCMBOOTx	Value of the **CFGTCMBOOTx** signal.

To access the IMP_PINOPTR:

```
MRC p15, 0, <Rt>, c15, c2, 7 ; Read IMP_PINOPTR into Rt
```

3.3.83 Processor Feature Register 0

The ID_PFR0 provides top-level information about the instruction sets supported by the processor in AArch32.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

ID_PFR0 must be interpreted with ID_PFR1.

Traps and enables

EL1 read accesses to ID_PFR0 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_PFR0 is a 32-bit register.

The following figure shows the ID_PFR0 bit assignments.

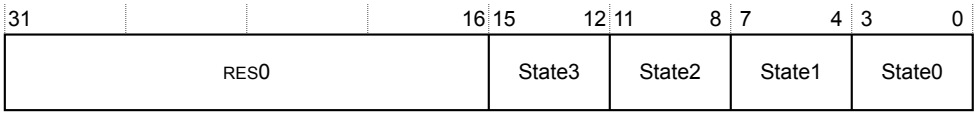

Figure 3-74 ID_PFR0 bit assignments

The following table shows the ID_PFR0 bit assignments.

Table 3-122 ID_PFR0 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[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
```

3.3.84 Processor Feature Register 1

The ID_PFR1 register provides information about the programmers model and architecture extensions supported by the processor.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Must be interpreted with ID_PFR0.

Traps and enables

EL1 read accesses to ID_PFR1 are trapped to EL2 when HCR.TID3 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

ID_PFR1 is a 32-bit register.

The following figure shows the ID_PFR1 bit assignments.

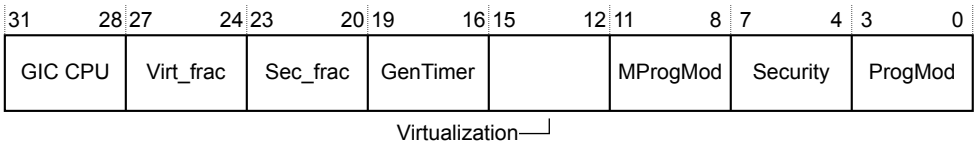

Figure 3-75 ID_PFR1 bit assignments

The following table shows the ID_PFR1 bit assignments.

Table 3-123 ID_PFR1 bit assignments

Bits	Name	Function
[31:28]	GIC CPU	GIC CPU support: `0x1` System register interface to the GIC CPU interface is supported.
[27:24]	Virt_frac	Virtualization fractional field: `0x0` No features from the Armv7 Virtualization Extensions are implemented.
[23:20]	Sec_frac	Security fractional field: `0x1` The VBAR register is implemented. The TTBCR register is not implemented in the Cortex-R52 processor.
[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: `0x0` Security not implemented.
[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
```

3.3.85 Protection Region Base Address Register

The Protection Region Base Address Register indicates the base address of the EL1- controlled MPU region, and provides two mechanisms for accessing it, direct (PRBARn) and indirect (PRBAR). For the indirect access also see PRSELR.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

The PRBAR is accessible from EL2, and from EL1 when VSCTLR.MSA is 0.

Configurations

This register is available in all build configurations.

Attributes

PRBAR and PRBARn are 32-bit registers.

The following figure shows the PRBAR bit assignments:

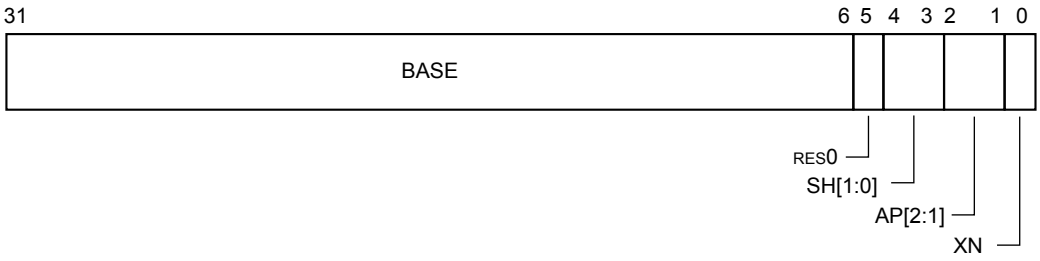

Figure 3-76 PRBAR bit assignments

The following table shows the PRBAR bit assignments:

Table 3-124 PRBAR bit assignments

Bits	Name	Function
[31:6]	BASE	Contains bits[31:6] of the lower inclusive limit of the selected EL1-controlled MPU memory region. This value is zero extended to provide the base address to be checked against.
[5]	-	Reserved, RES0.
[4:3]	SH[1:0]	Shareability field.
[2:1]	AP[2:1]	Access Permission bits.
[0]	XN	Execute-never.

The following table describes the SH[1:0] field encodings for Normal memory.

Table 3-125 SH[1:0] field encoding for Normal memory

SH[1:0]	Normal memory
0b00	Non-shareable
0b01	UNPREDICTABLE
0b10	Outer Shareable
0b11	Inner Shareable

The following table describes the data access permissions for EL1-controlled MPU.

Table 3-126 AP[2:1] Data access permissions for EL1-controlled MPU

AP[2:1]	Access from EL1	Access from EL0
0b00	Read/write	None
0b01	Read/write	Read/write
0b10	Read-only	None
0b11	Read-only	Read-only

This register resets to an UNKNOWN value.

To access the PRBAR as selected by PRSELR:

```
MRC p15, 0, <Rt>, c6, c3, 0 ; Read PRBAR into Rt
MCR p15, 0, <Rt>, c6, c3, 0 ; Write Rt to PRBAR
```

Direct access is provided to PRBAR0 to PRBAR15. To access PRBARn, where n is referenced as a binary number:

```
MRC p15, 0, <Rt>, c6, c8+n[3:1], 4*n[0] ; Read PRBARn into Rt
MCR p15, 0, <Rt>, c6, c8+n[3:1], 4*n[0] ; Write Rt into PRBARn
```

3.3.86 Protection Region Limit Address Register

The Protection Region Limit Address Register indicates the limit address of the EL1-controlled MPU region, and provides two mechanisms for accessing it, direct (PRLARn) and indirect (PRLAR). For the indirect access also see PRSELR.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

The PRLAR is accessible from EL2, and from EL1 when VSCTLR.MSA is 0.

Configurations

This register is available in all build configurations.

Attributes

PRLAR and PRLARn are 32-bit registers.

The following figure shows the PRLAR bit assignments:

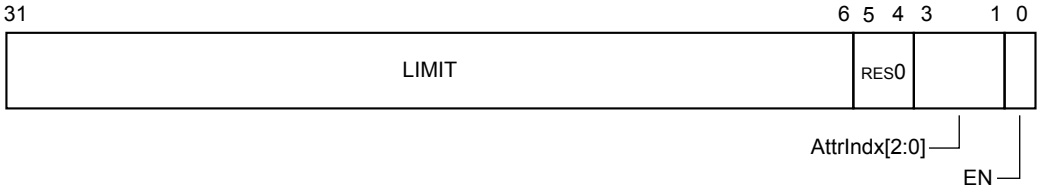

Figure 3-77 PRLAR bit assignments

The following table shows PRLAR bit assignments.

Table 3-127 PRLAR bit assignments

Bits	Name	Function
[31:6]	LIMIT	Contains bits[31:6] of the upper inclusive limit of the selected EL1 MPU memory region. This value is postfixed with `0x3F` to provide the limit address to be checked against. Resets to an UNKNOWN value.
[5:4]	-	Reserved, RES0.

Table 3-127 PRLAR bit assignments (continued)

Bits	Name	Function
[3:1]	AttrIndx[2:0]	Indexes a set of attributes in one of the MAIRx registers. Resets to an UNKNOWN value.
[0]	EN	Region enable. 0 Region disabled. 1 Region enabled. This field resets to zero.

To access the PRLAR:

```
MRC p15, 0, <Rt>, c6, c3, 1 ; Read PRLAR into Rt
MCR p15, 0, <Rt>, c6, c3, 1 ; Write Rt to PRLAR
```

Direct access is provided to PRLAR0 to PRLAR15. To access PRLARn, where n is referenced as a binary number:

```
MRC p15, 0, <Rt>, c6, c8+n[3:1], 4*n[0]+1 ; Read PRLARn into Rt
MCR p15, 0, <Rt>, c6, c8+n[3:1], 4*n[0]+1 ; Write Rt into PRLARn
```

3.3.87 Protection Region Selection Register

The PRSELR indicates, and selects the current EL1-controlled MPU region registers, PRBAR, and PRLAR.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

The PRSELR is accessible from EL2, and from EL1 when VSCTLR.MSA is 0.

Configurations

This register is available in all build configurations.

Attributes

PRSELR is a 32-bit register.

0 or 16 EL2-controlled MPU regions

The following figure shows the PRSELR bit assignments if 0 or 16 EL2-controlled MPU regions are implemented.

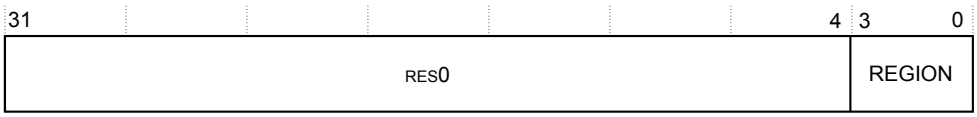

Figure 3-78 PRSELR bit assignments for 0 or 16 EL2-controlled MPU regions

The following table shows the PRSELR bit assignments if 0 or 16 EL2-controlled MPU regions are implemented.

Table 3-128 PRSELR bit assignments for 0 or 16 EL2-controlled MPU regions

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	REGION	The number of the current region visible in PRBAR and PRLAR. If a 0 region MPU is implemented, writing a value to this register has UNPREDICTABLE results. Writing a value to this register that is greater than or equal to 16 has UNPREDICTABLE results.

20 or 24 EL2-controlled MPU regions

The following figure shows the PRSELR bit assignments if 20 or 24 EL2-controlled MPU regions are implemented.

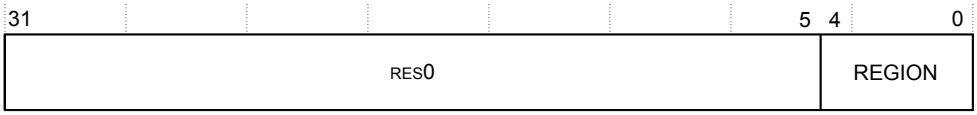

Figure 3-79 PRSELR bit assignments for 20 or 24 EL2-controlled MPU regions

The following table shows the PRSELR bit assignments if 20 or 24 EL2-controlled MPU regions are implemented.

Table 3-129 PRSELR bit assignments for 20 or 24 EL2-controlled MPU regions

Bits	Name	Function
[31:5]	-	Reserved, RES0.
[4:0]	REGION	The number of the current region visible in PRBAR and PRLAR. If a 20 region MPU is implemented, writing a value to this register greater than or equal to 20 has UNPREDICTABLE results. If a 24 region MPU is implemented, writing a value to this register greater than or equal to 24 has UNPREDICTABLE results.

To access the PRSELR:

```
MRC p15, 0, <Rt>, c6, c2, 1 ; Read PRSELR into Rt
MCR p15, 0, <Rt>, c6, c2, 1 ; Write Rt to PRSELR
```

3.3.88 Quality Of Service Register

The IMP_QOSR provides a programmable *Quality of Service* (QoS) identifier for the AXIM read and write channels.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW/RO	RW

Traps and enables

Register is accessible as RW from EL1 when HACTLR.QOSR is set, RO from EL1 when HACTLR.QOSR is clear.

Configurations

This register is available in all build configurations.

Attributes

IMP_QOSR is a 32-bit register.

The following figure shows the IMP_QOSR bit assignments.

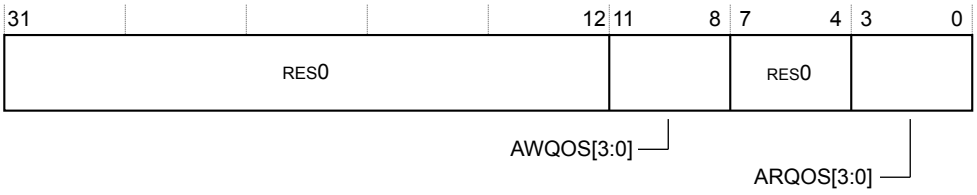

Figure 3-80 IMP_QOSR bit assignments

The following table shows the IMP_QOSR bit assignments.

Table 3-130 IMP_QOSR bit assignments

Bits	Name	Function
[31:12]	-	Reserved, RES0
[11:8]	AWQOS[3:0]	QoS identifier sent on the write address channel for each write transaction
[7:4]	-	Reserved, RES0
[3:0]	ARQOS[3:0]	QoS identifier sent on the read address channel for each read transaction

To access the IMP_QOSR:

```
MRC p15, 1, <Rt>, c15, c3, 1 ; Read IMP_QOSR into Rt
MCR p15, 1, <Rt>, c15, c3, 1 ; Write Rt to IMP_QOSR
```

3.3.89 Reset Vector Base Address Register

RVBAR contains the address that execution starts from after reset.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RO

Traps and enables

EL1 accesses to RVBAR are trapped to EL2 when HSTR.T12 is set.

Configurations

This register is available in all build configurations.

Attributes

RVBAR is a 32-bit register.

The following figure shows the RVBAR bit assignments.

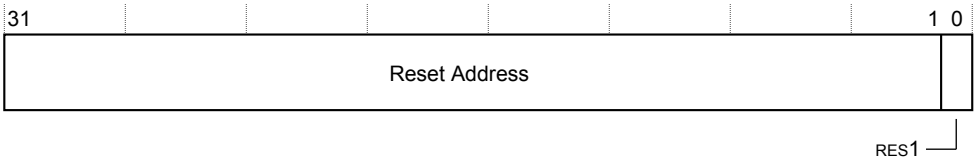

Figure 3-81 RVBAR bit assignments

The following table shows the RVBAR bit assignments.

Table 3-131 RVBAR bit assignments

Bits	Name	Function
[31:1]	Reset Address	The Reset Address[31:1]. **[31:5]** This is the value of **CFGVECTABLEx[31:5]**. **[4:1]** `0b0000`
[0]	-	Reserved, RES1.

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

To access the RVBAR:

```
MRC p15, 0, <Rt>, c12, c0, 1 ; Read RVBAR into Rt
```

3.3.90 Revision ID Register

The REVIDR 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	EL2
-	RO	RO

Traps and enables

EL1 read accesses to REVIDR are trapped to EL2 when HCR.TID1 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

REVIDR is a 32-bit register.

The following figure shows the REVIDR bit assignments.

Figure 3-82 REVIDR bit assignments

The following table shows the REVIDR bit assignments.

Table 3-132 REVIDR bit assignments

Bits	Name	Function
[31:12]	-	Reserved, RES0.
[11:0]	ID number	Implementation-specific revision information. The reset value is determined by the specific Cortex-R52 implementation. `0x000` Revision code is zero.

To access the REVIDR:

```
MRC p15, 0, <Rt>, c0, c0, 6 ; Read REVIDR into Rt
```

3.3.91 Slave Port Control Register

The IMP_SLAVEPCTLR provides control to set the bus privilege level required for the AXIS to access the TCM.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

EL1 accesses to the IMP_SLAVEPCTLR are trapped to EL2 when HCR.TIDCP is set.

Configurations

This register is available in all build configurations.

Attributes

IMP_SLAVEPCTLR is a 32-bit register.

The following figure shows the IMP_SLAVEPCTLR bit assignments.

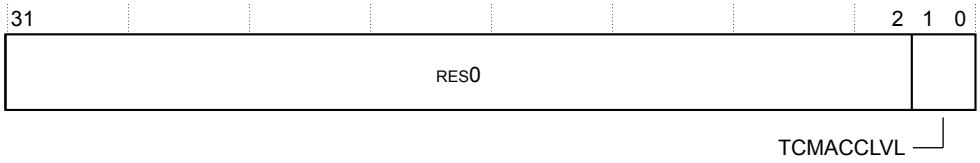

Figure 3-83 IMP_SLAVEPCTLR bit assignments

The following table shows the IMP_SLAVEPCTLR bit assignments.

Table 3-133 IMP_SLAVEPCTLR bit assignments

Bits	Name	Function
[31:2]	-	Reserved, RES0.
[1:0]	TCMACCLVL	Indicates the privilege level required for the AXIS to access the TCM. 0b00 No access. 0b01 Privileged access only. This is the reset value. 0b10 Reserved. 0b11 Privileged and unprivileged access.

To access the IMP_SLAVEPCTLR:

```
MRC p15, 0, <Rt>, c11, c0, 0 ; Read IMP_SLAVEPCTLR into Rt
MCR p15, 0, <Rt>, c11, c0, 0 ; Write Rt to IMP_SLAVEPCTLR
```

3.3.92 System Control Register

The SCTLR provides control of the core, including its memory system.

Usage constraints

The SCTLR is accessible as follows:

EL0	EL1	EL2
-	RW	RW

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.

Traps and enables

EL1 write accesses to SCTLR are trapped to EL2 if HCR.TVM is set. EL1 read accesses to SCTLR are trapped to EL2 if HCR.TRVM is set. EL1 accesses to this register are trapped to EL2 if HSTR.T1 is set.

Configurations

This register is available in all build configurations.

Attributes

SCTLR is a 32-bit register.

The following figure shows the SCTLR bit assignments.

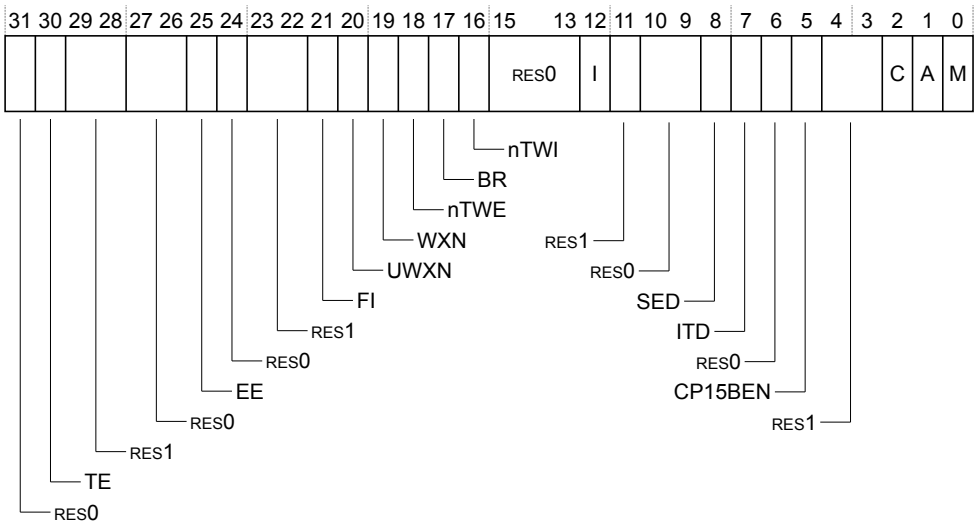

Figure 3-84 SCTLR bit assignments

The following table shows the SCTLR bit assignments.

Table 3-134 SCTLR bit assignments

Bits	Name	Function
[31]	-	Reserved, RES0.
[30]	TE	T32 Exception enable. This bit controls whether exceptions taken to EL1 are taken in A32 or T32 state: 0 Exceptions taken in A32 state. 1 Exceptions taken in T32 state. The input **CFGTHUMBEXCEPTIONSx** defines the reset value of the TE bit.
[29:28]	-	Reserved, RES1.
[27:26]	-	Reserved, RES0.
[25]	EE	Exception Endianness bit. The value of this bit defines the value of the CPSR.E bit on taking an exception to EL1. 0 Little endian. 1 Big endian. The input **CFGENDIANESSx** defines the reset value of the EE bit.
[24]	-	Reserved, RES0.
[23:22]	-	Reserved, RES1.
[21]	FI	Fast Interrupts. Read-only copy of the HSCTLR.FI bit.

Table 3-134 SCTLR bit assignments (continued)

Bits	Name	Function
[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 XN. This bit can be used to require all EL1-controlled MPU memory regions with write permissions to be treated as XN at EL0 and EL1. `0` EL1-controlled MPU regions with write permission are not forced to be XN, this is the reset value. `1` EL1-controlled MPU regions with write permissions are forced to be XN.
[18]	nTWE	Do not trap *Wait for Event* (`WFE`). `0` If a `WFE` instruction executed at EL0 would cause execution to be suspended, for example 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 `1`. For more information about the dependency of the WFE instruction on the state of the Event Register, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.
[17]	BR	Background Region for EL1. The reset value is 0.
[16]	nTWI	Do not trap WFI. `0` If a `WFI` instruction executed at EL0 would cause execution to be suspended, for example if there is no 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 `1`.
[15:13]	-	Reserved, RES0.
[12]	I	Instruction cache enable bit. This is a global enable bit for instruction caches: `0` All instruction access to Normal memory from EL1 and EL0 are Non-cacheable for the instruction cache. `1` All instruction access to Normal memory from EL1 and EL0 can be cached in the instruction. The reset value is 0.
[11]	-	Reserved, RES1.
[10:9]	-	Reserved, RES0.
[8]	SED	`SETEND` instruction disable. Disables SETEND instructions at EL0 and EL1. `0` The `SETEND` instruction execution is enabled at EL0 and EL1. `1` The `SETEND` instructions are UNDEFINED at EL0 and EL1. The reset value is 0.

Table 3-134 SCTLR bit assignments (continued)

Bits	Name	Function
[7]	ITD	IT Disable: 0 The IT instruction functionality is available at EL0 and EL1. 1 All encodings of the IT instruction with hw1[3:0]!=1000 are UNDEFINED at EL0 and EL1. All encodings of the subsequent instruction with the following values for hw1 are UNDEFINED at EL0 and EL1: 0x11xxxxxxxxxxxxxx All 32-bit instructions, B(2), B(1), Undefined, SVC, Load/Store multiple 0x1x11xxxxxxxxxxxx Miscellaneous 16-bit instructions 0x1x10xxxxxxxxxx ADD Rd, PC, #imm 0x01001xxxxxxxxxxx LDR Rd, [PC, #imm] 0x0101xxx1111xxx ADD(4), CMP(3), MOV, BX pc, BLX pc 0x010001xx1xxxx111 ADD(4), CMP(3), MOV The reset value is 0. Though the Cortex-R52 processor supports this functionality, it is deprecated in Armv8.
[6]	-	Reserved, RES0.
[5]	CP15BEN	CP15* Barrier enable. Enables use of the CP15DMB, CP15DSB, and CP15ISB barrier operations at EL0 and EL1: 0 CP15* barrier operations disabled at EL0 and EL1. Their encodings are UNDEFINED. 1 CP15* barrier operations enabled at EL0 and EL1. The reset value is 1.
[4:3]	-	Reserved, RES1.
[2]	C	Cache enable, for data caching. This is a global enable bit for data and unified caches: 0 All data access to Normal memory from EL1 and EL0 are Non-cacheable. 1 All data access to Normal memory from EL1 and EL0 can be cached at all levels of data and unified cache. The reset value is 0.
[1]	A	Alignment check enable. This is the enable bit for Alignment fault checking: 0 Alignment fault checking disabled at EL0 and EL1, this is the reset value. 1 Alignment fault checking enabled at EL0 and EL1.
[0]	M	EL1-controlled MPU enable. 0 EL1-controlled MPU disabled. 1 EL1-controlled MPU enabled. The reset value is 0.

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
```

3.3.93 TCM Error Record Register 0 and 1

The IMP_TCMERR0 and IMP_TCMERR1 indicate the TCM and the address index of a detected TCM error.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW/RO	RW

The IMP_TCMERR0 and IMP_TCMERR1 are accessible as RW from EL1 when HACTLR.ERR is set, RO from EL1 when HACTLR.ERR is clear.

Traps and enables

EL1 write accesses are trapped to EL2 when HACTLR.ERR is 0.

Configurations

This register is available in all build configurations.

Attributes

IMP_TCMERR0 and IMP_TCMERR1 are 32-bit registers.

The following figure shows the IMP_TCMERR0 and IMP_TCMERR1 bit assignments.

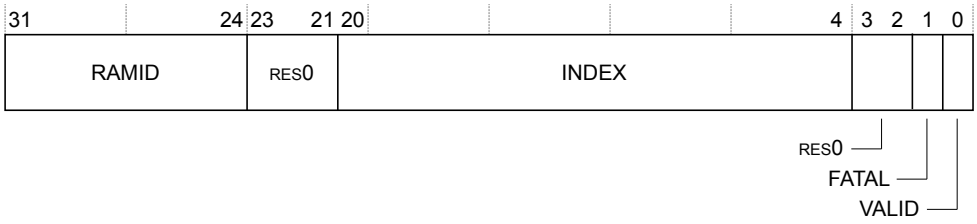

Figure 3-85 IMP_TCMERR0 and IMP_TCMERR1 bit assignments

The following table shows the IMP_TCMERR0 and IMP_TCMERR1 bit assignments.

Table 3-135 IMP_TCMERR0 and IMP_TCMERR1 bit assignments

Bits	Name	Function
[31:24]	RAMID	RAM bank identifier. `0x00` ATCM one bank only. `0x11` BTCM bank 0. `0x12` BTCM bank 1. `0x13` BTCM both banks. `0x21` CTCM bank 0. `0x22` CTCM bank 1. `0x23` CTCM both banks.
[23:21]	-	Reserved, RES0.
[20:4]	INDEX	Bits [19:3] of the access address.
[3:2]	-	Reserved, RES0.

Table 3-135 IMP_TCMERR0 and IMP_TCMERR1 bit assignments (continued)

Bits	Name	Function
[1]	FATAL	Recorded error is a fatal error.
[0]	VALID	Register contents are valid.

To access the IMP_TCMERR0:

```
MRC p15, 2, <Rt>, c15, c2, 0 ; Read IMP_TCMERR0 into Rt
MCR p15, 2, <Rt>, c15, c2, 0 ; Write Rt to IMP_TCMERR0
```

To access the IMP_TCMERR1:

```
MRC p15, 2, <Rt>, c15, c2, 1 ; Read IMP_TCMERR1 into Rt
MCR p15, 2, <Rt>, c15, c2, 1 ; Write Rt to IMP_TCMERR1
```

Note

Writes to these registers are unique. If bit[1] and bit[0] are set to 0, and if the register holds information about a correctable error, then the register is cleared. If bit[1] is set to 1 and bit[0] is set to 0, then the register is always cleared.

3.3.94 TCM Region Registers A, B, and C

Each of the IMP_ATCMREGIONR, IMP_BTCMREGIONR, and IMP_CTCMREGIONR registers indicates the TCM size, and controls its base address and enable for processor core accesses.

Usage constraints

These registers are accessible as follows:

EL0	EL1	EL2
-	RW	RW

Traps and enables

EL1 accesses to the IMP_ATCMREGIONR, IMP_BTCMREGIONR, and IMP_CTCMREGIONR are trapped to EL2 when HCR.TIDCP is set.

Configurations

These registers are available in all build configurations.

Attributes

IMP_ATCMREGIONR, IMP_BTCMREGIONR, and IMP_CTCMREGIONR are 32-bit registers.

The following figure shows the IMP_ATCMREGIONR, IMP_BTCMREGIONR, and IMP_CTCMREGIONR bit assignments.

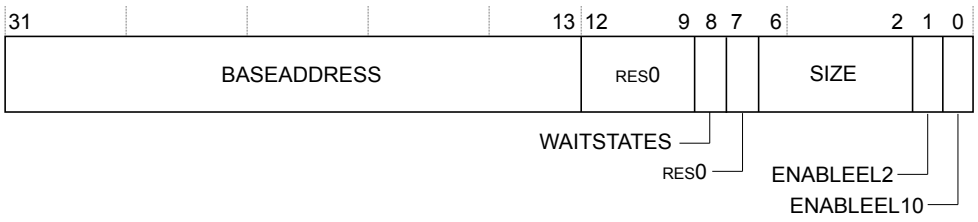

Figure 3-86 IMP_ATCMREGIONR, IMP_BTCMREGIONR, and IMP_CTCMREGIONR bit assignments

The following table shows the IMP_ATCMREGIONR, IMP_BTCMREGIONR, and IMP_CTCMREGIONR bit assignments.

Table 3-136 IMP_ATCMREGIONR, IMP_BTCMREGIONR, and IMP_CTCMREGIONR bit assignments

Bits	Name	Function
[31:13]	BASEADDRESS	Bits [31:13] of the TCM base address. These bits reset to 0.
[12:9]	-	Reserved, RES0.
[8]	WAITSTATES	Indicates the number of wait states for TCM accesses.
[7]	-	Reserved, RES0.
[6:2]	SIZE	Indicates the TCM size: 0b0000 No TCM present. 0b00100 8KB. 0b00101 16KB. 0b00110 32KB. 0b00111 64KB. 0b01000 128KB. 0b01001 256KB. 0b01010 512KB. 0b01011 1MB.
[1]	ENABLEEL2	Enable TCM at EL2.
[0]	ENABLEEL10	Enable TCM at EL1 and EL0.

At reset all bits are 0 apart from SIZE and WAITSTATES. If **CFGTCMBOOTx** is HIGH, then IMP_ATCMREGIONR.ENABLEEL2 and IMP_ATCMREGIONR.ENABLEEL10 reset at 1.

To access the IMP_ATCMREGIONR:

```
MRC p15, 0, <Rt>, c9, c1, 0 ; Read IMP_ATCMREGIONR into Rt
MCR p15, 0, <Rt>, c9, c1, 0 ; Write Rt to IMP_ATCMREGIONR
```

To access the IMP_BTCMREGIONR:

```
MRC p15, 0, <Rt>, c9, c1, 1 ; Read IMP_BTCMREGIONR into Rt
MCR p15, 0, <Rt>, c9, c1, 1 ; Write Rt to IMP_BTCMREGIONR
```

To access the IMP_CTCMREGIONR:

```
MRC p15, 0, <Rt>, c9, c1, 2 ; Read IMP_CTCMREGIONR into Rt
MCR p15, 0, <Rt>, c9, c1, 2 ; Write Rt to IMP_CTCMREGIONR
```

Note

The IMP_ATCMREGIONR, IMP_BTCMREGIONR, and IMP_CTCMREGIONR register base address and TCM enable settings have no impact on accesses that the AXIS TCM slave port makes.

3.3.95 TCM Syndrome Register 0 and 1

The IMP_TCMSYNDR0 and IMP_TCMSYNDR1 are specific to the TCMs and record the syndrome values in case of errors.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and enables

These registers are always RO.

Configurations

This register is available in all build configurations.

Attributes

IMP_TCMSYNDR0 and IMP_TCMSYNDR1 are 32-bit registers.

The following figure shows the IMP_TCMSYNDR0 and IMP_TCMSYNDR1 bit assignments.

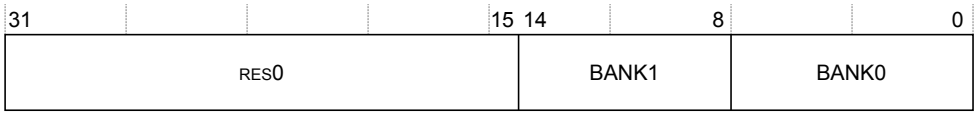

Figure 3-87 IMP_TCMSYNDR0 and IMP_TCMSYNDR1 bit assignments

The following table shows the IMP_TCMSYNDR0 and IMP_TCMSYNDR1 bit assignments.

Table 3-137 IMP_TCMSYNDR0 and IMP_TCMSYNDR1 bit assignments

Bits	Name	Function
[31:15]	-	Reserved, RES0.
[14:8]	BANK1	Syndrome for a bank 1 error.
[7:0]	BANK0	Syndrome for a bank 0 error.

To access the IMP_TCMSYNDR0:

```
MRC p15, 2, <Rt>, c15, c2, 2 ; Read IMP_TCMSYNDR0 into Rt
```

To access the IMP_TCMSYNDR1:

```
MRC p15, 2, <Rt>, c15, c2, 3 ; Read IMP_TCMSYNDR1 into Rt
```

The IMP_TCMSYNDRn registers are linked to the IMP_TCMERRn registers. When a new error is recorded in IMP_TCMERRn, the corresponding syndrome or syndromes are recorded simultaneously in IMP_TCMSYNDRn. The fields populated within IMP_TCMSYNDRn depend on the IMP_TCMERRn.RAMID. The following table shows the possible associations between the registers. When software clears an error in IMP_TCMERRn, IMP_TCMSYNDRn is also cleared.

Table 3-138 IMP_TCMSYNDRn fields

IMP_TCMERRn.RAMID	Errors	IMP_TCMSYNDRn fields written	Syndrome description
0x00	ATCM, bank 0	BANK0[7:0]	1 x 8-bit (64-bit ECC scheme)
0x11	BTCM, bank 0	BANK0[6:0]	1 x 7-bit (32-bit ECC scheme)
0x12	BTCM, bank 1	BANK1[6:0]	1 x 7-bit (32-bit ECC scheme)
0x13	BTCM, bank 0 and bank 1	BANK0[6:0], BANK1[6:0]	2 x 7-bit (32-bit ECC schemes)

Table 3-138 IMP_TCMSYNDRn fields (continued)

IMP_TCMERRn.RAMID	Errors	IMP_TCMSYNDRn fields written	Syndrome description
0x21	CTCM, bank 0	BANK0[6:0]	1 x 7-bit (32-bit ECC scheme)
0x22	CTCM, bank 1	BANK1[6:0]	1 x 7-bit (32-bit ECC scheme)
0x23	CTCM, bank 0 and bank 1	BANK0[6:0], BANK1[6:0]	2 x 7-bit (32-bit ECC schemes)

3.3.96 TCM Type Register

The TCMTR indicates which TCMs are implemented.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and enables

EL1 read accesses to TCMTR are trapped to EL2 when HCR.TID1 or HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

TCMTR is a 32-bit register.

The following figure shows the TCMTR bit assignments.

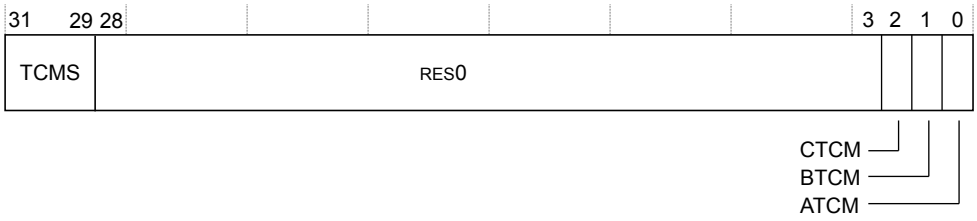

Figure 3-88 TCMTR bit assignments

The following table shows the TCMTR bit assignments.

Table 3-139 TCMTR bit assignments

Bits	Name	Function
[31:29]	TCMS	Indicates if TCMs are implemented: 0b000 No TCMs implemented. 0b100 One or more TCMs implemented.
[28:3]	-	Reserved, RES0.
[2]	CTCM	CTCM implemented and size is non-zero.
[1]	BTCM	BTCM implemented and size is non-zero.
[0]	ATCM	ATCM implemented and size is non-zero.

The reset value is the value of the bit fields, that depends on the number of TCMs implemented.

To access the TCMTR:

```
MRC p15, 0, <Rt>, c0, c0, 2 ; Read TCMTR into Rt
MCR p15, 0, <Rt>, c0, c0, 2 ; Write Rt to TCMTR
```

3.3.97 Test Register 0

The IMP_TESTR0 test register enables easier software testing by making some parts of the hardware directly visible to the software.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RO	RO

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

IMP_TESTR0 is a 32-bit register.

The following figure shows the IMP_TESTR0 bit assignments.

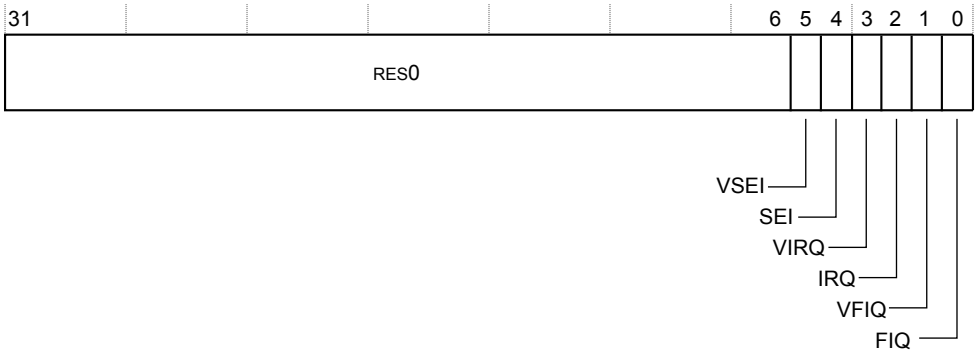

Figure 3-89 IMP_TESTR0 bit assignments

The following table shows the IMP_TESTR0 bit assignments.

Table 3-140 IMP_TESTR0 bit assignments

Bits	Name	Function
[31:6]	-	Reserved, RES0.
[5]	VSEI	Value of the Virtual System Error Interrupt signal to the PE.
[4]	SEI	Value of the System Error Interrupt signal to the PE.
[3]	VIRQ	Value of the Virtual IRQ Interrupt signal to the PE.
[2]	IRQ	Value of the IRQ Interrupt signal to the PE.

Table 3-140 IMP_TESTR0 bit assignments (continued)

Bits	Name	Function
[1]	VFIQ	Value of the Virtual FIQ Interrupt signal to the PE.
[0]	FIQ	Value of the FIQ Interrupt signal to the PE.

To access the IMP_TESTR0:

```
MRC p15, 4, <Rt>, c15, c0, 0 ; Read IMP_TESTR0 into Rt
```

3.3.98 Test Register 1

The IMP_TESTR1 register is used for Arm testing purposes only.

3.3.99 TLB Type Register

TLBTR is always RES0.

3.3.100 Vector Base Address Register

The VBAR holds the exception base address for exceptions that are not taken to Hyp mode.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	RW	RW

Software must program the register with the required initial value as part of the processor boot sequence.

Traps and enables

EL1 accesses to VBAR are trapped to EL2 when HSTR.T12 is set.

Configurations

This register is available in all build configurations.

Attributes

VBAR is a 32-bit register.

The following figure shows the VBAR bit assignments.

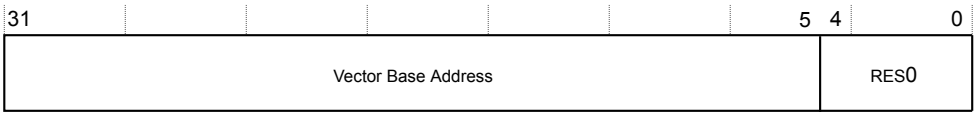

Figure 3-90 VBAR bit assignments

The following table shows the VBAR bit assignments.

Table 3-141 VBAR bit assignments

Bits	Name	Function
[31:5]	Vector Base Address	Vector Base Address. Bits[31:5] of the base address of the exception vectors for exceptions taken to this Exception level. Bits[4:0] of an exception vector are the exception offset.
[4:0]	-	Reserved, RES0.

The reset value for VBAR is `0x00000000`.

See the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R 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
```

3.3.101 Virtualization Multiprocessor ID Register

The VMPIDR provides the value of the Virtualization Multiprocessor ID. This is the value returned by EL1 reads of the MPIDR.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to VMPIDR are trapped to EL2 when HSTR.T0 is set.

Configurations

This register is available in all build configurations.

Attributes

VMPIDR is a 32-bit register.

VMPIDR resets to the value of MPIDR.

The following figure shows the VMPIDR bit assignments.

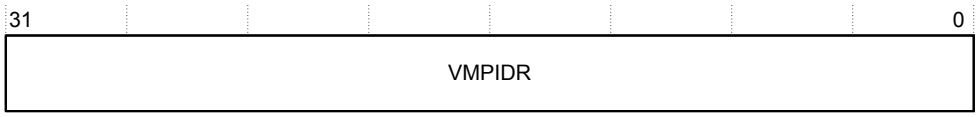

Figure 3-91 VMPIDR bit assignments

The following table shows the VMPIDR bit assignments.

Table 3-142 VMPIDR bit assignments

Bits	Name	Function
[31:0]	VMPIDR	MPIDR value returned by EL1 reads of the MPIDR. The MPIDR description defines the subdivision of this value. See *3.3.78 Multiprocessor Affinity Register* on page 3-167.

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
```

3.3.102 Virtualization Processor ID Register

The VPIDR holds the value of the Virtualization Processor ID. This is the value returned by EL1 reads of the MIDR.

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

EL1 accesses to VPIDR are trapped to EL2 when HSTR.T0 is set.

Configurations

The register is available in all build configurations.

Attributes

VPIDR is a 32-bit register.

VPIDR resets to the value of MIDR.

The following figure shows the VPIDR bit assignments.

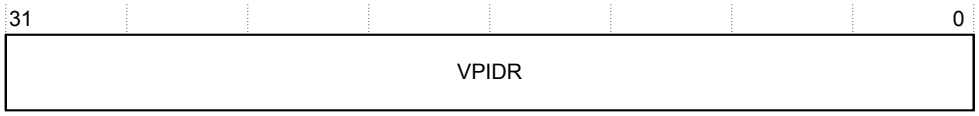

Figure 3-92 VPIDR bit assignments

The following table shows the VPIDR bit assignments.

Table 3-143 VPIDR bit assignments

Bits	Name	Function
[31:0]	VPIDR	MIDR value returned by EL1 reads of the MIDR. The MIDR description defines the subdivision of this value. See *3.3.69 Main ID Register* on page 3-156.

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
```

3.3.103 Virtualization System Control Register

The VSCTLR holds the *Virtual Machine ID* (VMID).

Usage constraints

This register is accessible as follows:

EL0	EL1	EL2
-	-	RW

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

VSCTLR is a 32-bit register.

The following figure shows the VSCTLR bit assignments.

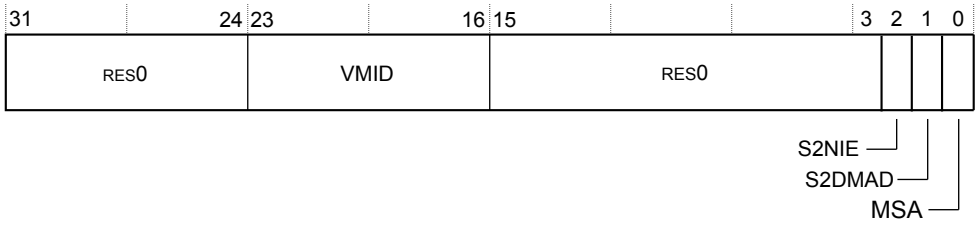

Figure 3-93 VSCTLR bit assignments

The following table shows the VSCTLR bit assignments.

Table 3-144 VSCTLR bit assignments

Bits	Name	Function
[31:24]	-	Reserved, RES0.
[23:16]	VMID	Virtual Machine ID. This resets to zero.
[15:3]	-	Reserved, RES0.
[2]	S2NIE	EL2-controlled Normal Interrupt Enable. This permits interrupts with base-restore of all accesses executing in EL0 or EL1 that are marked as Normal memory in the EL2-controlled MPU. This is regardless of the attributes returned by the EL1-controlled MPU. 0 Disabled. 1 Enabled. Accesses executing in EL0 or EL1 can be interrupted. The reset value is 0.
[1]	S2DMAD	EL2-controlled Device Multiple Access Disable. This causes any load or store instructions from EL0 and EL1 to memory regions marked as Device in the EL2-controlled MPU that spans an aligned 64-bit boundary to generate a permission fault. 0 Disabled. 1 Enabled. Multiple accesses generate Permission Faults. The reset value is 0.
[0]	MSA	This field is RES0. EL1 uses PMSA memory management.

To access the VSCTLR:

```
MRC p15, 4, <Rt>, c2, c0, 0 ; Read VSCTLR into Rt
MCR p15, 4, <Rt>, c2, c0, 0 ; Write Rt to VSCTLR
```

Chapter 4
Clocking and Resets

This chapter describes clocks and resets used within the processor.

It contains the following sections:

4.1 Clock and clock enables

The Cortex-R52 processor has a single clock which drives all its flops and RAMs. Various inputs, including the reset inputs, have synchronizing logic that allows them to be operated asynchronously to the processor clock. Most of the buses have enable inputs that allow them to operate at an integer division of the processor clock.

The single Cortex-R52 processor clock is distributed to all the cores and associated logic. Each core uses a gated clock that can be disabled in WFI low-power mode or WFE low power mode. For more information about these modes, see *5.3 Architectural clock gating* on page 5-207.

In systems with DCLS and Split/Lock, a separate clock input is provided for the redundant logic. This clock input must have the same frequency as the main clock input and be balanced against the main clock input. In DCLS and Split/Lock configurations, all inputs must be synchronous to the clock to prevent divergence caused by the synchronizers.

The following table shows the clock signals.

Table 4-1 Clock signals

Signal	Direction	Description
CLKIN	Input	Main clock.
CLKINDCLS	Input	Redundant clock for the redundant logic in lock-step configurations.

Each of the processor bus ports can operate at an integer division of the main processor clock. This is achieved using a clock enable input.

The following table shows the clock enable signals.

Table 4-2 Clock enable signals

Signal	Direction	Description
PCLKENDBG	Input	APB clock enable.
ACLKENMx	Input	AXIM interface clock enable.
ACLKENS	Input	AXIS interface clock enable.
ATCLKEND	Input	ATB clock enable for data trace.
ATCLKENI	Input	ATB clock enable for instruction trace and clock enable for **TSVALUEB[63:0]**.
ACLKENFx	Input	Flash interface clock enable.
CNTCLKEN	Input	Counter clock enable **CNTVALUEB**.
ACLKENPx	Input	LLPP clock enable.

4.2 Reset signals

The Cortex-R52 processor has reset inputs to enable the following operations:

- Powerup reset for the entire processor or unexpected safety error.
- Reset of an individual core after shutdown.
- Warm reset of an individual core or reset after an emulated shutdown.
- Debug reset for the whole processor.
- MBIST reset.

Reset of individual cores, and not the surrounding system, is possible only when the core in question is in a quiescent state. This is the case after a shutdown, whether emulated or not. If this function is required at another time, for example as a periodic preventive reset for safety purposes, then the software must first put the processor into a quiescent state which is similar to entering shutdown mode.

The following table shows the reset and reset control signals.

——— **Note** ———

If DCLS or Split/Lock is configured, all reset signals must be synchronous to the processor clock.

———————————

Table 4-3 Reset signals

Signal	Direction	Description
nCORERESETx	Input	Individual core warm reset. **0** Apply reset to core x excluding debug and trace logic. **1** Do not apply reset to the core.
nCPUPORESETx	Input	Individual core powerup cold reset. **0** Apply reset to core x including debug and trace logic. **1** Do not apply reset to the core.
nTOPRESET	Input	Top-level reset. Resets top-level functional logic. **0** Reset top-level functional logic. **1** Do not reset top-level functional logic.
nCORERESETDCLSx	Input	Individual redundant core warm reset. This input must be identical to **nCORERESET**. **0** Apply reset to the redundant copy of core x excluding debug logic. **1** Do not apply reset to the redundant core.
nCPUPORESETDCLSx	Input	Individual redundant core powerup cold reset. This input must be identical to **nCPUPORESET**. **0** Apply reset to the redundant copy of core x including debug logic. **1** Do not apply reset to the redundant core.
nTOPRESETDCLS	Input	Top-level redundant logic reset. Resets all the redundant logic in the processor. This input must be identical to **nTOPRESET**. **0** Apply reset to all the redundant logic. **1** Do not apply reset to all the redundant logic.

Table 4-3 Reset signals (continued)

Signal	Direction	Description
nPRESETDBG	Input	APB reset: **0** Apply reset to APB and top-level debug logic. **1** Do not apply reset to APB and top-level debug logic.
nMBISTRESET	Input	MBIST reset. **0** Apply reset to MBIST. **1** Do not apply reset to MBIST.

In normal operation, all reset signals are deasserted.

The following table shows the various reset scenarios and associated asserted reset signals. Reset flops use an asynchronous reset input.

Table 4-4 Reset scenarios and asserted reset signals

Signal and scenario	Powerup	Core powerup	Core warm	Debug	MBIST
Core functional: **nCORERESETx**	Yes	Yes	Yes	No	No
Core functional and debug: **nCPUPORESETx**	Yes	Yes	No	No	No
L2 system (AXIS and GIC) **nTOPRESET**	Yes	No	No	No	No
Debug: **nPRESETDBG**	Yes	No	No	Yes	No
MBIST: **nMBISTRESET**	Yes	No	No	No	Yes

On assertion, powerup reset propagates to the flops asynchronously.

All resets are synchronized for deassertion to meet timing requirements at the flops. All the reset signals are capable of being disabled when performing logic scanning, this includes the main input synchronizers and the synchronizers provided for each core.

In systems with DCLS, separate reset inputs are provided for the redundant logic and these are expected to be identical in behavior to the main reset inputs. The only exception to this condition is if a fault occurs, in which case they are not identical in behavior.

When the Cortex-R52 processor leaves reset, it automatically invalidates the instruction and data caches and prevents any lookups occurring until this process completes. While the automatic cache invalidate operation is in progress, the core does not use the cache. This can impact core performance for a short time after reset. The automatic cache invalidate also has the effect of initializing the ECC syndrome values of the RAMs.

Reset assertion sequence

All reset signals are asserted at the same time during powerup cold reset. If you perform reset outside of initial powerup then the order of signal assertion does not matter.

——— **Important** ———

You must ensure that the processor is quiescent before asserting resets.

———————————

Reset deassertion sequence

If you are deasserting the reset signals synchronously, then all the reset signals can be deasserted on the same clock cycle. If you are deasserting the reset signals asynchronously and the internal reset synchronizers are responsible for synchronizing the reset, then Arm recommends that you deassert the

nTOPRESET signal before deasserting the other reset signals. This ensures that the top-level logic is out of reset before the processor cores come out of reset.

4.3 Reset-related signals

Each core has two reset request outputs which it can assert to request a warm reset. Whether these signals are factored into the reset inputs is determined by the reset control logic external to the Cortex-R52 processor.

The following table shows the reset-related signals.

Table 4-5 Reset-related signals

Signal	Direction	Description
WARMRSTREQx	Output	Warm core reset request.
DBGRSTREQx	Output	Request for reset from external debug logic.
CPUHALTx	Input	Core waits out of reset before taking reset exception and fetching instructions.
CFGINITREG	Input	Program-visible registers initialized to fixed value out of reset.
CFGL1CACHEINVDISx	Input	Automatic post-reset L1 cache invalidate disable.

WARMRSTREQx is asserted when software writes 1 to HRMR.RR. **DBGRSTREQx** is asserted when the debugger writes 1 to the EDPRCR.CWRR.

Chapter 5
Power Management

This chapter describes the power management facilities provided by the Cortex-R52 processor.

It contains the following sections:

5.1 About power management

The processor provides features to reduce both dynamic and static power dissipation.

These features include:

- Local and regional clock gates with automatic controls built into the design for reducing dynamic power.
- Architectural clock gating for each core. This is used in WFI low-power state, and WFE low-power state to further reduce dynamic power when the core is idle.
- Control mechanisms and partitioning to support power gating of individual cores to reduce static power.

5.2 Local and regional clock gating

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

Explicit clock gates in the design enable the clock for a larger region of logic to be disabled when idle, which further reduces dynamic power consumption. These regional clock gates are automatically controlled by logic in the processor according to the current activity.

5.3 Architectural clock gating

The `WFI` and the `WFE` instructions are features of the Armv8-R architecture that put the core in a low-power state. These instructions typically disable the clocks in the core while keeping the core powered up.

This eliminates most of the dynamic power consumption in the core. This is a form of standby mode, which is called WFI low-power state or WFE low-power state depending on whether it was entered using `WFI` or `WFE` respectively.

WFI low-power state is part of a core powerdown sequence.

This section contains the following subsections:

5.3.1 WFI low-power state

A core enters into WFI low-power state by executing the `WFI` instruction.

When executing the `WFI` instruction, the core waits for all instructions in the core to retire before entering the WFI low-power 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 receive the required data or responses from the memory system:

- Load instructions.
- Cache maintenance operations.
- Store-Exclusive instructions.

In addition, the `WFI` instruction ensures that store instructions have retired from the pipeline and completed. The definition of completed is:

- Stores have written to the end point as required by the memory attribute.
- Response is known (either okay or error).

While the core is in WFI low-power state, the core automatically disables its clocks using clock gating when it is not handling an incoming transaction. The clocks in the core are temporarily enabled without causing the core to exit WFI low-power state, when any of the following are ongoing:

- An APB access to the debug or trace registers residing in the core power domain.
- An AXIS interface access to one of the TCMs in the core.

The core exits from WFI low-power state when it is reset or when a WFI wake-up event, for example an interrupt, occurs. See the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* for information about the various WFI wake-up events.

When it is in WFI low-power state, **STANDBYWFIx** for that core is asserted. This assertion only indicates that the core has stopped executing instructions. **STANDBYWFIx** continues to assert even if the clocks in the core are temporarily enabled because of an AXIS or APB access.

5.3.2 WFE low-power state

A core enters into WFE low-power state by executing the `WFE` instruction.

When executing the `WFE` instruction, the core waits for all instructions in the core to complete before entering WFE low-power state. The `WFE` instruction ensures that all explicit memory accesses that occurred before the `WFE` instruction in program order, have completed.

While the core is in WFE low-power state, the core automatically disables its clocks using clock gating when it is not handling an incoming transaction. The clocks in the core are temporarily enabled without causing the core to exit WFE low-power state, when any of the following events are ongoing:

- An APB access to the debug or trace registers residing in the core power domain.
- An AXIS interface access to one of the TCMs in the core.

The core exits from WFE low-power state when it is reset or when a WFE wake-up event occurs. An example of a WFE wake-up event is an event that is generated by a `SEV` instruction on another core. See the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* for information about the various WFE wake-up events.

A Cortex-R52 core also exits from WFE low-power state when one of the following occurs:

- The **EVENTI** input signal asserts.
- The **CLREXMONREQ** input signal asserts.

When it is in WFE low-power state, **STANDBYWFEx** for that core is asserted. This assertion only indicates that the core has stopped executing instructions. **STANDBYWFEx** remains asserted even if the clocks in the core are temporarily enabled because of an AXIS access or an APB access.

5.3.3 Event communication using WFE and SEV instructions

The **EVENTI** signal enables an external agent to participate in the WFE and SEV event communication.

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

The external agent can determine that at least one of the cores in the processor has executed an `SEV` instruction by checking the **EVENTO** signal. When any of the cores in the processor execute a `SEV` instruction, an event is signaled to all the cores in the processor, and the **EVENTO** signal is asserted. This signal is asserted HIGH for three **CLKIN** cycles when any of the cores executes an `SEV` instruction.

5.3.4 CLREXMON request and acknowledge signaling

The **CLREXMONREQ** input can be used by the system to signal to the Cortex-R52 processor that an external global exclusive monitor has been cleared. It also acts as WFE wake-up event to all the cores in the processor.

5.4 Power gating

The Cortex-R52 processor is designed to allow individual cores within the cluster to be powered down. Powering down removes most of the dynamic and static power consumption that is associated with the core.

This feature requires that the processor is implemented with the appropriate power domains and clamps, and integrated into an SoC with an appropriate power controller. Therefore, some of the details of the core powerdown are specific to the device you are using. For power domains that can be implemented in the Cortex-R52 processor, see *5.4.1 Power domains* on page 5-209.

Before a core can be powered down, the software running on it must put it into a suitable state. This might involve saving data that is stored in the registers or TCMs into system memory, which remains powered. The software must execute a `WFI` instruction to stop the pipeline and drain the bus masters. Finally the power controller must use the Cortex-R52 *Low-Power Interface* (LPI) to ensure that the core is quiescent and cannot be woken up, before removing the power.

The Cortex-R52 LPI provides hint signals to the power controller. These hint signals indicate when an incoming interrupt or debug request is received, the core should be powered up. After power is restored to the core, the core must be reset. The boot software of the core can either restart it from cold, or restore the state that was previously saved. The Cortex-R52 core cannot retain any state itself. If you require any state to be restored after powerdown, the software must do this.

This section contains the following subsections:

5.4.1 Power domains

The Cortex-R52 processor has two hierarchical power domains PDCPUx and PDTOP.

PDTOP is the top-level power domain and includes:

- GIC distributor, Debug APB slave, CTM, and CTIs.
- AXIS interface.
- MBIST logic.
- Timers.

PDCPU is the power domain for each core and includes:

- Core x.
- L1 instruction and cache RAMs and TCM RAMs.
- Core-level debug logic.
- ETM logic.

The following figure shows an example of the processor domains that are embedded in a SoC power domain.

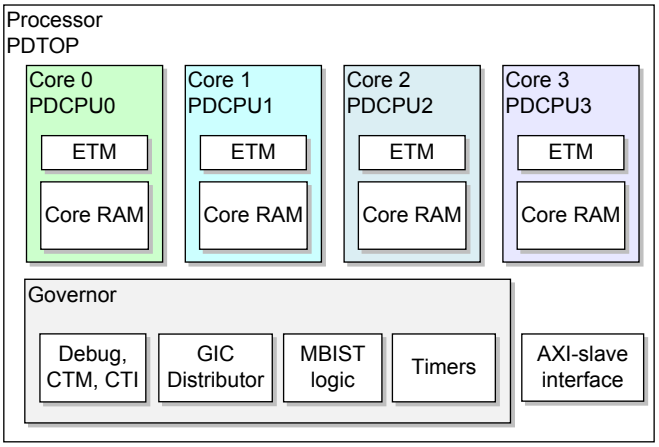

Figure 5-1 Processor power domains

5.4.2 Cortex-R52 LPI

Each Cortex-R52 core has its own P-channel LPI. Each LPI operates independently of each other. The LPI can be used by a power controller to request that the core enters a state in which it can be safely powered down. The LPI also provides hint information about when a power down request is likely to be successful and when a powerup might be required.

See *Low Power Interface Specification Arm® Q-Channel and P-Channel Interfaces*.

Each P-channel can request one of two states:

RUN (COREPSTATEx=0)

Normal operation of the core is available, including servicing of transactions that are received through the AXIS interface and APB transactions to core domain debug registers. The core can enter and exit WFI low-power state or WFE low-power state while it is in RUN state.

SHUTDOWN (COREPSTATEx=1)

The core is not executing instructions, or servicing interrupts or transactions, that are received through the AXIS interface and APB transactions to core domain debug registers. In this state, a transaction that is received by the AXIS interface is given an error response. A transaction that is received by the APB interface or a core domain debug register is given an error response unless emulated powerdown has been enabled.

Each P-channel provides activity hints on two bits of **COREPACTIVEx**:

Bit[1]

In RUN state, this bit is asserted if the core is either:

- Executing instructions.
- Has outstanding transactions.
- Is available to handle an interrupt.
- Is servicing a transaction that is received through the AXIS interface or an APB transaction to a core domain debug register.
- Is servicing an APB transaction to a core domain debug register unless emulated powerdown is enabled.

In SHUTDOWN state, this bit becomes asserted if there is an incoming interrupt which would normally be routed to the core, or if there is a debug powerup request.

Bit[0]

Asserted if the debug no powerdown (DBGPRCR.CORENPDRQ) or debug powerup request (EDPRCR.COREPURQ, TRCPDCR.PU) functions are set. This indicates to the power controller that any powerdown must be emulated.

Note

When the core is in SHUTDOWN state, an error response is given for either:

- A transaction that is received by the AXIS interface for the core.
- A transaction that is received by the debug-APB interface for a core domain debug register, except when emulated powerdown is enabled.

Such transactions are not blocked and do not cause any of the activity hints to become asserted.

5.4.3 Powerdown sequence

To powerdown a core, the software running on that core must use the following procedure:

Procedure

1. If necessary, save the state of the core to system memory that is not powered down.

   This must include any data that is stored in the TCMs which you do not want to lose.
2. Disable the interrupt enable bits in the ICC_IGRPEN0 and ICC_IGRPEN1 registers. Set the GIC distributor wake-up request for the core using the GICR_WAKER register. Read the GICR_WAKER to confirm that the ChildrenAsleep bit indicates that the interface is quiescent.
3. Execute an *Instruction Synchronization Barrier* (`ISB`) to ensure that all the changes outlined in steps 1 and 2 have completed.
4. Program the power control logic in the SoC, as appropriate, for core powerdown.
5. Execute a `WFI` instruction.

In response to the request made in step 4, the power controller:

1. Waits until **COREPACTIVEx[1]** goes LOW.
2. Makes a request on the LPI for SHUTDOWN state.
3. When the request is successful, power down the corresponding core power domain, including asserting the clamps.

Note

After **COREPACTIVEx[1]** goes LOW, the only stimulus that can prevent a SHUTDOWN state request being successful is an incoming *System Error Interrupt* (**SEI**) or *Virtual System Error Interrupt* (**VSEI**). After handling such an **SEI** or **VSEI**, software must either execute a WFI, so that the core can shut down, or restart the core as described in *5.4.4 Powerup sequence* on page 5-211. The restart of the core must include clearing the GIC distributor wake request. When the SHUTDOWN request has been accepted by the core, no stimulus can cause it to restart other than a reset.

5.4.4 Powerup sequence

If **COREPACTIVEx[1]** goes HIGH, or for any other SoC-specific reason, the power-controller must powerup the core using the following procedure:

Procedure

1. Assert **nCPUPORESETx** and **nCORERESETx** LOW.
2. Restore power to the corresponding core power domain, including releasing the clamps.
3. Make a request on the LPI for RUN state. The request is accepted after reset has been deasserted.
4. Deassert **nCPUPORESETx** and **nCORERESETx**.

The core takes a reset exception and starts execution of the reset handler. Software must initialize the core as normal, including initializing registers as required and clearing the GIC distributor wake request

using the GICR_WAKER register. If necessary, software must restore the state of the core which was saved before the powerdown.

5.4.5 Debug over powerdown

When a core is powered down normally, the state of the core domain debug registers is lost including any programmed breakpoints or watchpoints. When a core leaves powerdown, EDPRSR.SPD reads 1, indicating that the state of the core domain debug registers has been lost.

To enable debug of software associated with power control, the processor has features which enable emulated power down. In an emulated power down scenario, the software sequence for powering down and powering up the core is the same as described in *5.4.3 Powerdown sequence* on page 5-211 and *5.4.4 Powerup sequence* on page 5-211. However, the power controller behaves differently so that debug state is not lost.

The debugger can request that all powerdown operations are emulated by setting DBGPRCR.CORENPDRQ for the core or cores. This causes the **COREPACTIVEx[0]** output signal to be asserted, which the power controller must take as an indication to follow different powerdown and powerup sequences.

To perform an emulated powerdown the power controller must:

1. Wait until **COREPACTIVEx[1]** goes LOW.
2. Make a request on the LPI for SHUTDOWN state.

Note

The power controller must not assert the clamps or remove power from the core.

When performing a powerup after an emulated powerdown the power controller must:

1. Assert **nCORERESETx** LOW.
2. Make a request on the LPI for RUN state.
3. Deassert **nCORERESETx**.

When a core is in emulated powerdown, the effect of accesses to the debug registers is the same as in a normal run state. When a core leaves emulated powerdown, EDPRSR.SPD reads 0, indicating that the state of the core domain debug registers has not been lost.

5.4.6 Powerdown of the cluster

The Cortex-R52 processor does not include any features to assist with powerdown of the complete processor cluster.

It is the responsibility of the SoC to ensure that, in addition to powerdown of the individual cores, the interfaces to the AXIS, debug APB, and interrupts are idle if required by the system. Therefore, the software and hardware sequences for shutting down the cluster are specific to the SoC you are using.

Chapter 6
Initialization

This chapter describes considerations for initializing the Cortex-R52 processor.

It contains the following sections:

6.1 Initialization

The Cortex-R52 processor provides a mechanism for initializing all programmer-visible registers out of reset, including those which do not have an architecturally defined reset value.

Most of the architectural registers in the Cortex-R52 processor, such as r0-r14 and s0-s31, and d0-d31 when Advanced SIMD is included, have an UNKNOWN value after reset. Because of this, you must initialize these for all modes before they are used, using an immediate-MOV instruction, or a PC-relative load instruction.

The *Current Program Status Register* (CPSR) and some system register fields are given a known value on reset, see the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* for more information.

The Cortex-R52 processor provides a mechanism for initializing all programmer-visible registers to a fixed value out of reset. After reset, the registers have a fixed value. This mechanism is controlled by **CFGINITREG**.

In addition, before you run an application, you might want to:

- Program particular values into various registers, for example, Stack Pointers.
- Enable various features, for example, error correction.
- Program particular values into memory, for example, the TCMs.

This section contains the following subsections:

- *6.1.1 MPU* on page 6-214.
- *6.1.2 Floating-point Unit* on page 6-214.
- *6.1.3 Caches* on page 6-214.

6.1.1 MPU

Before you can use an MPU, you must program and enable at least one of the regions, and enable it in the SCTLR or HSCTLR.

Do not enable the MPU unless at least one MPU region is programmed and active. If the MPU is enabled, before using the TCM you must program MPU regions to cover the TCM regions to give access permissions to them.

6.1.2 Floating-point Unit

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

Enable the FPU as follows:

- Enable access to the FPU in the *3.3.1 Architectural Feature Access Control Register* on page 3-76.
- Enable the FPU by setting the EN bit in the FPEXC register.

6.1.3 Caches

If the Cortex-R52 processor has been built with instruction or data caches, they are automatically invalidated before they are used by the processor, unless **CFGL1CACHEINVDISx** is tied HIGH.

This operation can never report any ECC errors.

6.2 TCM

You can use TCMs either for instructions or data. You can also configure the Cortex-R52 processor to use the ATCM from reset.

This section contains the following subsections:

6.2.1 Preloading TCMs

You can write data to the TCMs using either store instructions or the AXIS interface.

Depending on the method you choose, you might require:

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

The methods to preload TCMs include:

Memory copy with running boot code
The boot code includes a memory copy routine that reads data from an external memory, and writes it into the appropriate TCM. You must enable the TCM to do this, and it might be necessary to give the TCM one base address while the copy is occurring, and a different base address when the application is being run.

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

Execute code in debug halt state
The debug host puts the Cortex-R52 processor into debug halt state and then feeds instructions into it through the *Instruction Transfer Register* (EDITR). The Cortex-R52 processor executes these instructions, as an alternative to using boot code in either of the two methods previously described.

DMA into TCM
The SoC includes a DMA device that reads data from a ROM, and writes it to the TCMs through the AXIS interfaces.

6.2.2 Preloading TCMs with ECC

The error code bits in the TCM RAM, if present, are not initialized by the Cortex-R52 processor. Before a RAM location is read with ECC enabled, the error code bits must be initialized.

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

You can use the **CFGTCMBOOTx** signal to enable the ATCM when leaving reset, and the **CFGRAMPROTEN** signal to enable TCM (and cache) ECC when leaving reset. When the core is running, the TCM ECC can be enabled and disabled using the *3.3.76 Memory Protection Control Register* on page 3-165.

You can initialize the TCM RAM with error checking turned on or off, according to the following rules:

- If the AXIS interface is used to initialize a TCM with ECC enabled, AXI slave transactions must start at 128-bit aligned addresses, writing continuous blocks of memory and all bytes in the block enabled.
- If initialization is done by running code on the Cortex-R52 processor, this is best done by a loop of stores that write to the whole of the TCM memory as follows:

— For BTCM or CTCM use *Store Word* (`STR`), *Store Two Words* (`STRD`), or *Store Multiple Words* (`STM`) instructions to 32-bit aligned addresses.
— For ATCM use `STRD`, or `STM` with a 64-bit aligned address.

Note

You can use the alignment-checking features of the Cortex-R52 processor to help you ensure that memory accesses are 32-bit aligned, but there is no checking for 64-bit alignment. If you are using `STRD` or `STM`, an alignment fault is generated if the address is not 32-bit aligned. For the same behavior with `STR` instructions, enable strict-alignment-checking by setting the A-bit in the *3.3.92 System Control Register* on page 3-184 or *3.3.53 Hyp System Control Register* on page 3-133.

6.2.3 Using TCMs from reset

The Cortex-R52 processor has a primary input for each core which when asserted prevents the core from starting to execute instructions out of reset. This enables the TCMs to be preloaded before the core boots. If an external debug request is made before this input is deasserted, then the core enters debug halt state before executing any instructions.

You can use the **CFGTCMBOOTx** signal to enable the ATCM from reset. This enables you to configure the processor to boot from TCM but, to do this, the TCM must first be preloaded with the boot code.

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

Note

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

6.3 Entering EL1

The Cortex-R52 processor always boots into Hyp mode (EL2).

If you do not need to use EL2 you can program the processor and switch to EL1 so that it can never return to EL2 except by means of a reset. This involves setting all exceptions to be taken at EL1 and disabling HVC and the EL2-controlled MPU.

To enter EL1:

- Program the HACTLR register because it defaults to only allowing EL2 accesses. HACTLR controls whether EL1 can access memory region registers and CPUACTLR.
- Program the SPSR before entering EL1.
- Other registers default to allowing accesses at EL1 from reset.
- Set VBAR to the correct location for the vector table.
- Disable the HVC instruction by setting HCR.HCD to 1.
- Set ELR to point to the entry point of the EL1 code and call ERET.

Chapter 7
Memory System

This chapter describes the memory system.

It contains the following sections:

7.1 About the memory system

The memory system controls access to internal RAM, caches, external memory, and the peripheral port.

The memory system consists of:

- Memory private to each core:
  - An optional L1 instruction cache.
  - An optional L1 data cache.
  - Optional TCMs.
- Interfaces private to each core:
  - An optional LLPP master interface that conforms to AXI4 (32-bit data bus).
  - An optional Flash interface that conforms to AXI4, but has some additional features (these features do not have to be used).
  - An AXIM interface that conforms to AXI4 (128-bit data bus).
- An AXIS interface that conforms to AXI4 providing external access to TCMs in all cores (128-bit data bus).

The following figure shows the memory system and external interfaces.

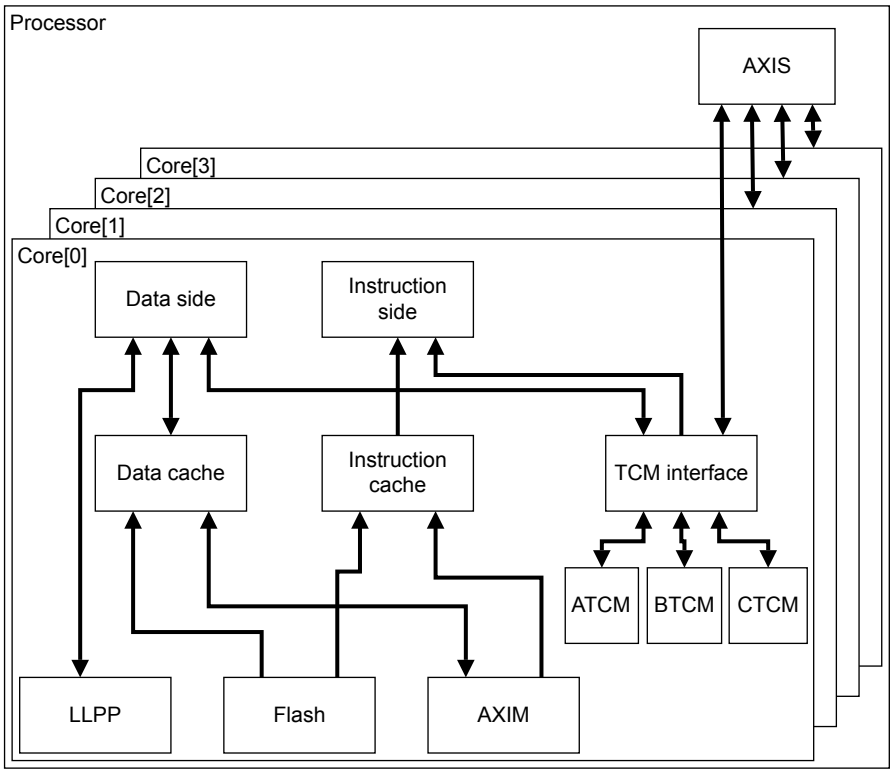

Figure 7-1 Memory system block diagram

The core has Harvard memory architecture, meaning that the core has independent paths to access instructions and data. The instruction side fetches instructions. The data side reads and writes data. The core can make concurrent accesses from the instruction side and data side.

The TCM provides fast access memory. The TCM is unified, meaning that it can hold both instructions and data. The TCM interface controls access to the TCM. The TCM interface includes a full crossbar switch that allows concurrent access from the three masters (instruction side, data side, and AXIS) to TCM (A, B, and C). If one or more masters attempt to access the same TCM, the TCM interface arbitrates between the requests on a fixed priority scheme with *Quality of Service* (QoS) mechanisms. TCM provides the most deterministic timing for memory accesses.

Where access timing determinism is less critical but fast access is still required, the instruction cache and data cache can be used. All stores that the data side generates that target the AXIM interface are written to external memory through the AXIM interface, and are either updated or allocated in the data cache. TCM and LLPP transactions are never allocated into the data cache. On cache misses, linefill requests are made to either the Flash or AXIM interfaces. The data cache always has Write-Through behavior.

The AXIM interface is the main interface to external memory or device systems. The Flash interface provides access to an external read-only memory controller, such as flash memory. The LLPP interface provides access to external peripherals or small specialized memory systems. The AXIS interface provides external access to the TCM. Accesses through the AXIS interface contend for TCM access cycles so they might reduce the determinism of TCM memory.

The memory system includes a local monitor for exclusive accesses. Exclusive load and store instructions can be used, for example, `LDREX`, `STREX`, to provide inter-process synchronization and semaphores. They can provide inter-process synchronization and semaphores using appropriate external memory monitoring logic. See the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

All instruction side and data side accesses are looked up in the memory map that is defined by the MPU. It returns access permissions before the instruction side or data side makes an access to external memory. If the access is permitted, the MPU provides the memory attributes for the transaction.

Note

Except for TCM overlap with flash, if present and enabled, it is illegal to perform an access which hits more than one region out of the ATCM, BTCM, and CTCM regions. Also, if present, it is illegal to perform an access which hits more than one region out of the LLPP, Flash, and internal peripheral regions.

Related references

7.2 TCM memory on page 7-221
7.5 AXIM interface on page 7-228
7.6 Low-latency peripheral port on page 7-233
7.7 Flash interface on page 7-241

7.2 TCM memory

TCMs are unified, provide fast access, and have the most deterministic memory access timing.

TCMs are private to the core. The size of each TCM is independently implemented as either 0KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB, 512KB, or 1MB. A size of 0KB indicates that the TCM is not implemented.

Each TCM can be implemented with a wait state meaning that all accesses to that TCM incur an extra cycle of latency. If some, but not all, TCMs have wait cycles implemented, the performance of the TCMs might differ from each other.

If the primary input, **CFGTCMBOOTx**, is asserted at reset, the ATCM is enabled, and the base address is `0x00000000`. Other TCMs are disabled and their base addresses are UNKNOWN.

The base address of each TCM and whether it is enabled can be programmed by writing to the respective TCM region register (IMP_ATCMREGIONR, IMP_BTCMREGIONR, IMP_CTCMREGIONR). Separate enables are provided for software running at EL2 and software running at EL1 or EL0. The TCM base address must be size-aligned. The size of each TCM is indicated in the corresponding TCM region register. If a TCM is disabled for the current Exception level, then accesses to its address region are performed through either the AXIM interface, the Flash interface, or the LLPP interface according to the configuration of the Flash interface.

Writes to TCM region registers, from EL1, can be trapped to the hypervisor, running at EL2, by setting the register bit HCR.TIDCP=1.

The TCMs can be accessed through the AXIS interface.

TCM attributes and permissions

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

7.3 Level-1 caches

Optional instruction and data caches are independently implemented in the core.

Instruction cache

The features of the instruction cache include:

- The instruction cache, when present, is implemented as either 4KB, 8KB, 16KB, or 32KB and is organized as four-way set associative with 64-byte lines.
- Instruction fetches from Flash or AXIM interfaces can be cached in the instruction cache, according to the cacheability attributes, the cache segregation policy, and how the settings to control the caches are programmed.
- When the instruction flow is sequential, the cache can automatically prefetch the next line from memory. The PFU requests non-sequential lines early using its branch prediction structures.
- Prefetching behavior is controlled by CPUACTLR.L1IPFCTL.

Data cache

The features of the data cache include:

- The data cache, when present, is implemented as either 4KB, 8KB, 16KB, or 32KB and is organized as four-way set associative with 64-byte lines.
- Data reads from Flash or AXIM interfaces can be cached in the data cache depending on the attributes set within the MPU, the cache segregation policy, and how the settings to control the caches are programmed.
- When two or three accesses in the pipeline miss in the data cache, the memory system can initiate linefills for them all. This so-called miss-under-miss behavior allows the cache to have up to three linefills outstanding at the same time.
- There are 4 data prefetchers for the AXIM interface that look for patterns in cacheable accesses and prefetches within 4KB regions. There are no data prefetchers for the Flash interface.

The Cortex-R52 processor is not coherent and the inner shareability domain consists of an individual Cortex-R52 core. The Cortex-R52 processor does not cache data that is marked as shareable, and all cache maintenance instructions are performed locally. This means that instruction cache maintenance operations are not broadcast to any other core. The outer shareability domain is external to the Cortex-R52 processor, and is therefore system-dependent.

In the Cortex-R52 processor, instruction side and data side accesses have an independent path to access main memory. As a result the point of coherency exists external to the processor, and is therefore system-dependent.

The inner cacheability domain consists of an individual Cortex-R52 core. The outer cacheability domain is external to the Cortex-R52 processor.

The write behavior for the data cache is always Write-Through caching. Locations that are marked as Write-Back Cacheable are treated as Write-Through. Caching of Write-Back and Write-Through regions is subject to any Read-Allocate or Write-Allocate hints being set. Locations that are marked as Inner or Outer Shareable are treated as Non-cacheable, regardless of their cacheability attributes. The data cache does not use the Transient allocation hint. Any regions that are marked as transient are treated as Non-transient.

This section contains the following subsections:

- *7.3.1 Cache segregation* on page 7-222.
- *7.3.2 Data cache invalidation* on page 7-223.
- *7.3.3 Write streaming mode* on page 7-224.

7.3.1 Cache segregation

L1 instruction and data caches segregate allocations from the Flash or AXIM interface into separate cache ways. IMP_CSCTLR.IFLW and IMP_CSCTLR.DFLW control the number of ways that are

allocated to each interface for the instruction cache and data cache respectively. Updates to the cache segregation controls are only permitted before the caches have ever been enabled, following a system reset, otherwise the update is ignored.

7.3.2 Data cache invalidation

The EDCCR enables overlay of the flash and AXIM memory with RAM and modification of the RAM content.

The following figure shows the state machine of data cache invalidation.

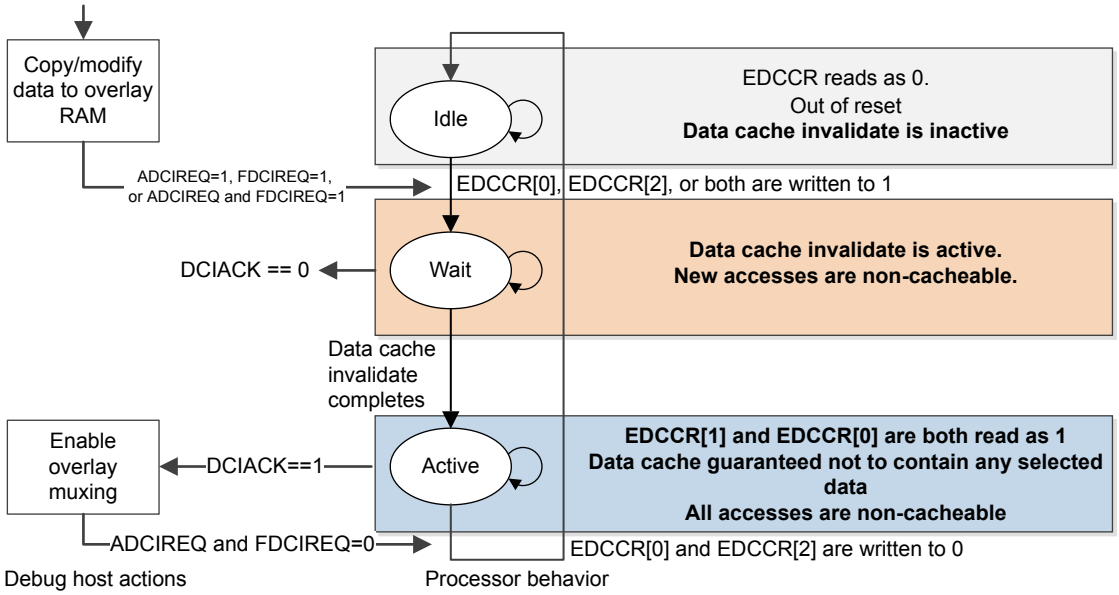

Figure 7-2 Data cache invalidation state machine

Idle

Out of reset, the state machine is in the Idle state. Reads from EDCCR is 0, indicating that data cache invalidation is inactive.

Wait

When EDCCR[0], EDCCR[2], or both are written to 1:

- A request is sent for the memory system to invalidate all data cache lines containing flash data (EDCCR[0] is written to 1), AXIM data (EDCCR[2] is written to 1), or both (EDCCR[0] and EDCCR[2] are written to 1). This is indicated as ADCIREQ=1, FDCIREQ=1, or ADCIREQ=1 and FDCIREQ=1.
- DCIACK=0.
- New flash, AXI, or both flash and AXI accesses are non-cacheable.
- The state machine is in the Wait state, where reads from EDCCR return 1, indicating that the cache does not allocate new cache lines and the invalidation of old cache lines is in progress.

Active

When the memory system completes the data cache invalidation request:

- DCIACK=1, indicating that cache invalidation is complete.
- The state machine is in the Active state.
- EDCCR[0] and EDCCR[1] both read as 1.
- The data cache does not contain the invalidated data type.
- All flash, AXI, or both flash and AXI accesses are non-cacheable.

Note

- When EDCCR[0] and EDCCR[2] are written to 0, the state machine returns to the Idle state and flash or AXIM addresses are cacheable.
- When the state machine is not in the Idle state, writes of 1 to EDCCR are ignored.
- When the state machine is not in Active state, writes of 0 to EDCCR are ignored.

For more information on EDCCR, see *11.6.1 External Debug Calibration Control Register* on page 11-367

7.3.3 Write streaming mode

A cache line is allocated to the L1 cache based on the allocation hints configured in the MPU programming.

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

To counter this, the core includes logic to detect when the core has written a full cache line before the linefill completes. If this situation is detected on a configurable number of consecutive linefills, then it switches into write streaming mode. This is sometimes referred to as read allocate mode.

When in write streaming mode, loads behave as normal, and can still cause linefills, and writes still lookup the cache, but if they miss, then they write out to the bus rather than starting a linefill.

CPUACTLR.WSTRNOL1ACTL configures the write streaming mode threshold. For more information on CPUACTLR, see *3.3.19 CPU Auxiliary Control Register* on page 3-90.

Note

More than the specified number of linefills might be observed on master interface, before the core detects that the specified number of full cache lines have been written and switches to write streaming mode. The core continues in write streaming mode until it detects either a cacheable write burst that is not a full cache line, or there is a load from the same line that is currently being written to the bus.

7.4 Direct access to internal memory

The Cortex-R52 processor provides a mechanism to read the internal memory used by the L1 cache 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.

During processor execution, 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 EL2. In all other modes, executing the CP15 instruction results in an Undefined Instruction exception.

Table 7-1 CP15 registers used to access internal memory

Function	Access	CP15 operation	Rd Data
Cache Debug Data Register 0	RO	`MRC p15, 3, <Rd>, c15, c0, 0`	Data
Cache Debug Data Register 1	RO	`MRC p15, 3, <Rd>, c15, c0, 1`	Data
Data Cache Tag Read Operation Register	WO	`MCR p15, 3, <Rd>, c15, c2, 0`	Set/Way
Instruction Cache Tag Read Operation Register	WO	`MCR p15, 3, <Rd>, c15, c2, 1`	Set/Way
Data Cache Data Read Operation Register	WO	`MCR p15, 3, <Rd>, c15, c4, 0`	Set/Way/Offset
Instruction Cache Data Read Operation Register	WO	`MCR p15, 3, <Rd>, c15, c4, 1`	Set/Way/Offset

7.4.1 Data cache tag and data encoding

The Cortex-R52 processor data cache consists of a 4-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 7-2 Data cache tag and data location encoding

Bit field of Rd	Description
[31:30]	Cache way
[29:13]	Unused
[12:6], [11:6], [10:6], or [9:6]	Set index
[5:3]	Data offset, Data Register only
[2:0]	Unused

Table 7-3 Index encoding depending on cache size

Cache size	Index
32KB	[12:6]
16KB	[11:6]
8KB	[10:6]
4KB	[9:6]

Data cache data reads return 64 bits of data with bits [31:0] placed in Cache Debug Data Register 0 and [63:32] placed in Cache Debug Data Register 1. Data cache tag reads return up to 23 bits of data placed in Cache Debug Data Register 0.

Table 7-4 Data cache tag data format

Register	Bit field	Description
Cache Debug Data Register 0	[22]	Valid
	[21:0]	Tag bits (4KB cache)
	[21:1]	Tag bits (8KB cache)
	[21:2]	Tag bits (16KB cache)
	[21:3]	Tag bits (32KB cache)

7.4.2 Instruction cache tag and data encoding

The Cortex-R52 processor consists of a 4-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 similar for both data and tag RAM access. Data RAM accesses include an additional field to locate the appropriate doubleword in the cache line.

Table 7-5 Instruction cache tag and data location encoding

Bit field of Rd	Description
[31:30]	Cache Way
[29:13]	Unused
[12:6], [11:6], [10:6], or [9:6]	Set index
[5:3]	Data offset, Data Register only
[2:0]	Unused

Table 7-6 Index encoding depending on cache size

Cache size	Index
32KB	[12:6]
16KB	[11:6]
8KB	[10:6]
4KB	[9:6]

Instruction cache data reads return 64 bits of data with bits [31:0] placed in Cache Debug Data Register 0 and bits [63:32] placed in Cache Debug Data Register 1. In T32 state, these two fields can represent any combination of 16-bit and partial or full 32-bit instructions. Instruction cache tag reads return up to 23 bits of data placed in Cache Debug Data Register 0.

Table 7-7 Instruction cache tag data format

Register	Bit field	Description
Cache Debug Data Register 0	[22]	Valid
	[21:0]	Tag bits (4KB cache)
	[21:1]	Tag bits (8KB cache)
	[21:2]	Tag bits (16KB cache)
	[21:3]	Tag bits (32KB cache)

7.5 AXIM interface

The AXIM interface is the main interface to external memory and peripherals. Each core has its own AXIM interface that is used only for accesses from that core.

The AXIM interface implements the AXI4 protocol, having 128-bit data width.

The AXIM interface receives requests from the instruction side and data side. You can configure whether data or instruction accesses have the highest priority in the case of contention, by setting CPUACTLR.AXIMARBCTL. The AXIM interface can have bus protection.

7.5.1 AXIM interface attributes

This section describes the AXIM interface attributes.

The following table shows the AXI4 master interface attributes for one core.

Table 7-8 AXI4 master interface attributes

Attribute	Value	Comments
Write issuing capability	3	Each core can issue a maximum of three write requests.
Read issuing capability	11	Each core can issue eight outstanding data side and three outstanding instruction side AXIM read requests.
Exclusive thread capability	1	Each core can have one Exclusive access sequence in progress.
Write ID capability	3	The maximum number of different **AXIDMx** values that a master interface can generate for all active write transactions at any one time. Device memory can have more than one outstanding transaction with the same AXI ID. For Normal memory, stores use the same ID only when there are multiple stores with the same address. This maintains ordering between the stores which would otherwise generate a *Write-after-Write* (WAW) hazard.
Write ID width	3	-
Read ID capability	11	The maximum number of different **ARIDMx** values that a master interface can generate for all active read transactions at any one time. The ID encodes the source of the memory transaction as Exclusive[bh], data side Device, instruction side Normal, or data side Normal. There can be one data side Device request. Transactions to Normal memory use a unique AXI ID for every outstanding transaction. The Cortex-R52 processor treats all Device regions of memory as *Execute-never* (XN), regardless of the programmed value of XN.
Read ID width	4	-

Related references

7.5.4 AXIM transaction IDs on page 7-230

7.5.2 AXIM interface transfers

The AXIM conforms to the AXI4 specification, but it does not generate all the AXI transaction types that the specification permits. This section describes the types of AXI transactions that the AXIM generates.

[bh] The attributes indicate non-Reordering Device requests (Device nGnRnE or nGnRE).

If you are designing an AXIS to work only with the Cortex-R52 AXIM, you can take advantage of these restrictions and the interface attributes to simplify the slave.

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.

For Write-Back or Write-Through (Inner cacheable) transfers, the supported transfers are:

- WRAP 4 128-bit for read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, and 64-bit for write transfers.
- INCR N (N:1-4) 128-bit for write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, and 64-bit for exclusive write transfers.

Note

The processor may treat a memory location as Non-cacheable even if the MPUs are programmed as Write-Back or Write-Through. This can happen when the caches are disabled. In such scenarios, see the supported transfer types for Non-cacheable transactions.

For Non-cacheable transactions:

- INCR N (N:1-4) 128-bit for data side read transfers.
- INCR N (N:1-4) 128-bit for write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, and 64-bit for write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit for data side read transfers.
- INCR N (N:1-3) 128-bit for instruction side read transfers.
- WRAP 4 128-bit for instruction side read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit for exclusive write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit for exclusive read transfers.

For Device transactions:

- INCR 1 8-bit, 16-bit, 32-bit, 64-bit for data side read transfers.
- INCR N (N:1-4) 128-bit for write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, and 64-bit for write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit for exclusive read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit for exclusive write transfers.

The following points apply to AXI transactions:

- WRAP bursts are only 128-bit and request a 4-beat burst.
- 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, some, or all byte strobes low can occur.

Note

These restrictions are not generic and you must not infer any other restrictions.

7.5.3 AXIM data prefetchers

The Cortex-R52 processor contains four independent data prefetchers, which look for patterns in the addresses of linefills.

The CPUACTLR register contains the DPFSTRCTL field which controls the number of enabled independent data prefetch streams between 1 and 4.

The data prefetcher uses patterns in the addresses of linefills to predict where future linefills are likely to be needed, and speculatively fetches them. A correct speculative fetch can significantly increase

performance, particularly if the memory system has a high latency. The CPUACTLR register contains a STRIDECTL field which controls the number of consecutive linefills for a particular pattern, between 2 or 3, that must be seen before a data prefetcher can be started.

It can be advantageous to limit the total number of linefills initiated by the data prefetchers so that some of the linefill resources are available to non-prefetch requests. The CPUACTLR register contains a L1DPFCTL field which allows the number of linefills initiated by all data prefetchers to be limited in the range 0 to 8, where 0 indicates that the data prefetchers disabled.

7.5.4 AXIM transaction IDs

The following table shows the identifier encoding for the AXIM read address and read response channels.

Table 7-9 AXIM read address and read response channel identifiers

ARIDMx Encoding	Transaction type	Description
0b0000	Exclusive accesses	Exclusive (LDREX). There can be one Exclusive sequence outstanding.
0b0010	Device accesses (data side)	Device-type data read. There can be one data side Device request using this ID.
0b01xx	Normal requests (instruction side)	Instruction fetch. There can be three instruction side Normal requests each using a unique ID.
0b1xxx	Normal requests (data side)	Normal-type data read. There can be eight data side Normal requests each using a unique ID.

The following table shows the identifier encoding for AXIM write address and write response channels.

Table 7-10 AXIM write address and write response channel identifiers

AWIDMx Encoding	Transaction type	Description
0b000	Exclusive accesses	Exclusive (STREX). There can be one Exclusive outstanding.
0b011	Device accesses	Device-type write. There can be three Device requests outstanding using the same ID.
0b1xx	Normal requests	Normal-type. There can be three Normal requests outstanding and each might have a unique ID.

The following tables show the encoding for the **ARCACHEMx** and **AWCACHEMx** fields. The Assigned attributes column indicates the memory type given by the MPU. A transaction uses a Default row, if it does not fall into one of the other categories. For example, if an access is exclusive without Device attributes, then the Default row is used. Otherwise, the appropriate Device attribute row is used.

Table 7-11 AXIM ARCACHEMx encodings

ARCACHEMx	Memory type	Assigned attributes
0b0000	Device Non-Bufferable	Device nGnRnE
0b0001	Device Bufferable	Device nGnRE, nGRE, or GRE
0b0011	Normal Non-cacheable Bufferable	Default
0b1110	Write-Through Read-Allocate	Normal outer Write-Through
0b1111	Write-Back Read-Allocate	Normal outer Write-Back

Table 7-12 AXIM AWCACHEMx encodings

AWCACHEMx	Memory type	Assigned attributes
0b0000	Device Non-Bufferable	Device nGnRnE
0b0001	Device Bufferable	Device nGnRE, nGRE, or GRE
0b0011	Normal Non-cacheable Bufferable	Default
0b1110	Write-Through Write-Allocate	Normal outer Write-Through
0b1111	Write-Back Write-Allocate	Normal outer Write-Back

7.5.5 AXI privilege information

The AXIM interface provides information about the privilege level of an access on the **ARPROTMx[2:0]** and **AWPROTMx[2:0]** signals.

Accesses might be cached or merged together and the resulting transaction can have both privileged and user data combined.

The following table shows the processor privilege level and associated **ARPROTMx[2:0]** and **AWPROTMx[2:0]** values. Cacheable read requests are always marked privileged. Non-cacheable read requests take on the privilege of the instruction that requested the read. Stores to Normal and Device GRE memories are always marked privileged, but Device nGRE and Exclusive stores indicate the privilege of the instruction that requested the store.

Table 7-13 Processor mode and ARPROTMx and AWPROTMx values

Processor mode	Type of access	Value of ARPROTMx or AWPROTMx
EL0, EL1, EL2	Cacheable, Non-shareable read access	Privileged access
EL0	Device or Normal Non-cacheable read access	Unprivileged access
EL1, EL2		Privileged access
EL0, EL1, EL2	Cacheable, Non-shareable write access	Privileged access
EL0	Device write	Unprivileged access
EL1, EL2		Privileged access
EL0	Normal Non-cacheable write, except for STREX, STREXB, STREXH, and STREXD to shareable memory	Privileged access
EL0	Normal Non-cacheable write for STREX, STREXB, STREXH, and STREXD to shareable memory	Unprivileged access
EL1, EL2	Normal Non-cacheable write	Privileged access

Note

All transactions are marked as Secure (**AxPROTMx[1]**=0).

7.5.6 AXIM QoS and user signals

This section describes the AXIM QoS and user signals.

Quality of Service

The AXIM outputs the QoS identifiers that are programmed in IMP_QOSR.AWQOS, for writes and IMP_QOSR.ARQOS, for reads. **ARQOSMx** and **AWQOSMx** are the AXIM QoS signals that read and write the quality of service identifier.

User signals

A valid VMID is only guaranteed for Device memory accesses. A valid VMID value is unavailable for Normal memory accesses. **ARVMIDMx** and **AWVMIDMx** are the AXIM VMID signals that read and write the VMID.

7.5.7 AXIM interface timeout

The AXIM includes a timeout mechanism to detect transactions that fail to complete within a programmable time limit.

A transaction fails to complete if an address or write data beat is not accepted, or a response is not received.

If timeout detection is enabled, the response to a timeout occurrence is:

- For each timeout occurrence, the originating core asserts output signal **ERREVENTx[13]**.
- If transaction abort is enabled, the transaction failing to complete is aborted and the memory system treats the AXIM as broken. All subsequent transactions targeting the AXIM are aborted.

The timeout duration is determined by setting the IMP_BUSTIMEOUTR.MAXCYCLESBY16AXIM field. To enable the timeout detection, set IMP_BUSTIMEOUTR.ENABLEAXIM. To enable the transaction abort, set IMP_BUSTIMEOUTR.ABORTAXIM.

7.6 Low-latency peripheral port

The LLPP is designed as a dedicated path to access private peripherals, but not restricted to them.

The LLPP implements the AXI4 protocol and has a 32-bit data width.

Each core has an interface to an LLPP. Accesses made through the LLPP are optimized for low-latency and are not cached. The LLPP is not optimal for Normal memory accesses. Accesses are always treated as non-Gathering, non-Reordering, with no Early Write Acknowledgement.

Because the LLPP uses the AXI4 protocol it is possible to connect all the peripheral ports from the processor to the same, shared memory system. However, this configuration is likely to incur a latency penalty. The LLPP can also be connected to memory instead of peripherals. This configuration is not optimal because the interface always applies Device-nGnRnE gathering, ordering, and write acknowledgment rules. An LLPP region that is marked as Normal memory allows unaligned and speculative memory accesses. Otherwise it follows Device-nGnRnE gathering, ordering, and write acknowledgement rules.

An example of connecting small amounts of shared memory to the LLPPs might be to store shared semaphores (with the appropriate exclusive monitor).

The LLPP address range and size is indicated in the IMP_PERIPHPREGIONR. It is set from the configuration signals **CFGLLPPBASEADDR** and **CFGLLPPSIZE** and is the same for all cores in a cluster. The LLPP region size is between 4KB and 4MB in powers of two. **CFGLLPPIMP** indicates if the LLPP interface is implemented. This signal must be tied low if the system does not use the LLPP, otherwise it must be tied high.

The LLPP is disabled at reset. IMP_PERIPHPREGIONR.ENABLEEL2 controls whether the LLPP is enabled for access at EL2 and IMP_PERIPHPREGIONR.ENABLEEL10 controls whether the LLPP is enabled for access at EL0 and EL1.

If the LLPP is implemented and enabled, data accesses to an address in the LLPP region are performed through the LLPP. If the LLPP is implemented and not enabled, data accesses to the LLPP region generate a synchronous external abort. If the LLPP is not implemented, then data accesses to the LLPP region are performed through other parts of the memory system.

If the LLPP is implemented, instruction accesses to an address in the LLPP region cause a synchronous external abort. If the LLPP is not implemented, instruction accesses are performed through other parts of the memory system.

The LLPP supports exclusive accesses, generating the appropriate exclusive transactions for all access sizes smaller than or equal to 32 bits. A load or store exclusive double operation that is performed through the LLPP generates a synchronous external abort.

There is no internal mechanism for one core to access peripherals attached to peripheral port on another core.

The AXIS interface does not have access to the LLPP.

The LLPP interface supports bus protection.

The following table shows LLPP attributes.

Table 7-14 LLPP attributes

Attribute	Value	Comments
Write issuing capability	8	Each core can issue a maximum of eight write requests.
Read issuing capability	2	Each core can issue two LLPP read requests.
Exclusive thread capability	1	Each core can have one Exclusive access sequence in progress.
Write ID capability	1	All writes use the same ID.

Table 7-14 LLPP attributes (continued)

Attribute	Value	Comments
Write ID width	1	-
Read ID capability	1	All reads use the same ID.
Read ID width	1	-

LLPP transfers

The LLPP only issues INCR bursts of length one, and transfers of up to 32 bits per transfer. The LLPP read-channel can post two 32-bit outstanding transactions when a single instruction requests 64 bits of data from a 64-bit aligned location. The LLPP does not post two read transactions on the bus for two separate instructions or for data within two separate 64-bit aligned locations. The LLPP write-channel generates up to eight 32-bit transactions on the bus from one or more instructions. All transactions use the same ID to ensure that responses are received in the same order the transactions were issued.

7.6.1 LLPP Memory attributes

The LLPP uses the **ARCACHEPx** and **AWCACHEPx** signals to indicate the memory attributes of transactions.

The following table shows the encoding that is used for the **ARCACHEPx** and **AWCACHEPx** signals of the LLPP master interface. Because the LLPP is optimized for peripherals, all accesses propagate Device Non-bufferable attributes, regardless of the attributes that are returned by the MPU. All other encodings are unused.

Table 7-15 ARCACHEPx and AWCACHEPx encodings

Encoding	Meaning
`0b0000`	Device Non-bufferable

7.6.2 LLPP AXI transfer restrictions

The LLPP conforms to the AXI4 specification, but it does not generate all the AXI transaction types that the specification permits.

This section describes the types of AXI transactions that the LLPP generates. If you are designing an AXIS to work only with the Cortex-R52 LLPP, you can take advantage of these restrictions and the interface attributes to simplify the slave.

This section also contains tables that show some examples of the types of AXI burst that the core generates. However, because a particular type of transaction is not shown here does not mean that the core does not generate such a transaction.

An AXI4 slave device that is connected to the LLPP must be capable of handling every kind of transaction that the AXI4 specification permits, except where there is an explicit statement in this chapter that such a transaction is not generated. You must not infer any additional restrictions from the example tables given.

Restrictions on AXI peripheral transfers describes restrictions on the type of transfers that the LLPP generates. If a core exists and is powered up, **BREADYPx** and **RREADYPx** are always asserted. You must not make any assumptions about the AXI handshaking signals, except that they conform to the AXI4 Protocol Specification.

Restrictions on AXI peripheral transfers

The LLPP applies the following restrictions to the AXI transactions it generates:

- A burst never transfers more than four bytes.
- The burst length is never more than one transfer.

- No transaction ever crosses a 4-byte boundary in memory.
- All transactions are incrementing (INCR) bursts.
- All transactions are secure data accesses.
- Transactions to Device memory are always to addresses that are aligned to the transfer size.
- Exclusive accesses are always to addresses that are aligned to the transfer size.

LLPP Device transactions

A load or store instruction to or from Device memory always generates AXI transactions of the same size as the instruction implies.

The following table shows the values of **ARADDRPx**, **ARBURSTPx**, **ARSIZEPx**, and **ARLENPx** for `LDRB` from bytes 0-3 in Device memory.

Table 7-16 LDRB transfers

Address[1:0]	ARADDRPx[7:0]	ARBURSTPx	ARSIZEPx	ARLENPx
`0x0` (byte 0)	`0x00`	INCR	8-bit	1 data transfer
`0x1` (byte 1)	`0x01`	INCR	8-bit	1 data transfer
`0x2` (byte 2)	`0x02`	INCR	8-bit	1 data transfer
`0x3` (byte 3)	`0x03`	INCR	8-bit	1 data transfer

The following table shows the values of **ARADDRPx**, **ARBURSTPx**, **ARSIZEPx**, and **ARLENPx** for `LDRH` from halfwords 0-1 in Device memory.

Table 7-17 LDRH transfers

Address[1:0]	ARADDRPx	ARBURSTPx	ARSIZEPx	ARLENPx
`0x0` (halfword 0)	`0x00`	INCR	16-bit	1 data transfer
`0x2` (halfword 1)	`0x02`	INCR	16-bit	1 data transfer

The following table shows the values of **ARADDRPx**, **ARBURSTPx**, **ARSIZEPx**, and **ARLENPx** for an `LDR` or an `LDM` that transfers one register, an `LDM1`, in Device memory.

Table 7-18 LDR or LDM transfers

Address[1:0]	ARADDRPx	AWBURSTPx	ARSIZEPx	ARLENPx
`0x0` (word 0)	`0x00`	INCR	32-bit	1 data transfer

The following table shows the values of **ARADDRPx**, **ARBURSTPx**, **ARSIZEPx**, and **ARLENPx** for an `LDM` that transfers five registers, an `LDM5`, in Device memory.

All accesses using `LDM`, `STM`, `LDRD`, or `STRD` instructions to Device memory occur as one or multiple 32-bit transactions.

Table 7-19 LDM transfers

Address[2:0]	ARADDRPx	ARBURSTPx	ARSIZEPx	ARLENPx
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer
	0x04	INCR	32-bit	1 data transfer
	0x08	INCR	32-bit	1 data transfer
	0x0C	INCR	32-bit	1 data transfer
	0x10	INCR	32-bit	1 data transfer
0x4 (word 1)	0x04	INCR	32-bit	1 data transfer
	0x08	INCR	32-bit	1 data transfer
	0x0C	INCR	32-bit	1 data transfer
	0x10	INCR	32-bit	1 data transfer
	0x14	INCR	32-bit	1 data transfer

The following table shows the values of **AWADDRPx**, **AWBURSTPx**, **AWSIZEPx**, **AWLENPx**, and **WSTRBPx** for an `STRB` from bytes 0-3 in Device memory.

Table 7-20 STRB transfers

Address[1:0]	AWADDRPx	AWBURSTPx	AWSIZEPx	AWLENPx	WSTRBPx
0x0 (byte 0)	0x00	INCR	8-bit	1 data transfer	0b0001
0x1 (byte 1)	0x01	INCR	8-bit	1 data transfer	0b0010
0x2 (byte 2)	0x02	INCR	8-bit	1 data transfer	0b0100
0x3 (byte 3)	0x03	INCR	8-bit	1 data transfer	0b1000

The following table shows the values of **AWADDRPx**, **AWBURSTPx**, **AWSIZEPx**, **AWLENPx**, and **WSTRBPx** for an `STRB` from halfwords 0-1 in Device memory.

Table 7-21 STRH transfers

Address[1:0]	AWADDRPx	AWBURSTPx	AWSIZEPx	AWLENPx	WSTRBPx
0x0 (halfword 0)	0x00	INCR	16-bit	1 data transfer	0b0011
0x2 (halfword 1)	0x02	INCR	16-bit	1 data transfer	0b1100

The following table shows the values of **AWADDRPx**, **AWBURSTPx**, **AWSIZEPx**, **AWLENPx**, and **WSTRBPx** for an `STR` or an `STM` that transfers one register, an `STM1`, to Device memory.

Table 7-22 STR or STM transfers

Address[1:0]	AWADDRPx	AWBURSTPx	AWSIZEPx	AWLENPx	WSTRBPx
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer	0b1111

The following table shows the values of **AWADDRPx**, **AWBURSTPx**, **AWSIZEPx**, **AWLENPx**, and **WSTRBPx** for an `STM` that writes five registers, an `STM5`, to Device memory.

Table 7-23 STM transfers

Address[2:0]	AWADDRPx	AWBURSTPx	AWSIZEPx	AWLENPx	WSTRBPx
`0x0` (word 0)	`0x00`	INCR	32-bit	1 data transfer	`0b1111`
	`0x04`	INCR	32-bit	1 data transfer	`0b1111`
	`0x08`	INCR	32-bit	1 data transfer	`0b1111`
	`0x0C`	INCR	32-bit	1 data transfer	`0b1111`
	`0x10`	INCR	32-bit	1 data transfer	`0b1111`
`0x4` (word 1)	`0x04`	INCR	32-bit	1 data transfer	`0b1111`
	`0x08`	INCR	32-bit	1 data transfer	`0b1111`
	`0x0C`	INCR	32-bit	1 data transfer	`0b1111`
	`0x10`	INCR	32-bit	1 data transfer	`0b1111`
	`0x14`	INCR	32-bit	1 data transfer	`0b1111`

Note

- A load of a halfword from Device memory addresses `0x1` or `0x3` generates an alignment fault.
- A load of a word from Device memory addresses `0x1`, `0x2`, or `0x3` generates an alignment fault.
- A load-multiple from Device memory addresses `0x1`, `0x2`, `0x3`, `0x5`, `0x6`, or `0x7` generates an alignment fault.
- A store of a halfword from Device memory addresses `0x1`, or `0x3` generates an alignment fault.
- A store of a word to Device memory addresses `0x1`, `0x2`, or `0x3` generates an alignment fault.
- A store-multiple to Device memory address `0x1`, `0x2`, `0x3`, `0x5`, `0x6`, or `0x7` generates an alignment fault.

LLPP Normal reads

Load instructions accessing Normal memory generate transactions on the LLPP that are not necessarily the same size as the instruction implies.

An LLPP Normal read might generate more than one bus transaction to obtain the required data.

The tables in this section give examples of the types of AXI transactions that might result from various load instructions, accessing various addresses in Normal memory. They are provided as examples only, and are not an exhaustive description of the AXI transactions.

The following table shows possible values of **ARADDRP**, **ARBURSTP**, **ARSIZEP**, and **ARLENP** for an `LDRH` from bytes 0-7 in Normal memory.

Table 7-24 LDRH transfers

Address[2:0]	ARADDRP	ARBURSTP	ARSIZEP	ARLENP
`0x0` (halfword 0)	`0x00`	INCR	16-bit	1 data transfer
`0x1`	`0x00`	INCR	32-bit	1 data transfer
`0x2` (halfword 1)	`0x02`	INCR	16-bit	1 data transfer

Table 7-24 LDRH transfers (continued)

Address[2:0]	ARADDRP	ARBURSTP	ARSIZEP	ARLENP
0x3	0x00	INCR	32-bit	1 data transfer
	0x04	INCR	32-bit	1 data transfer
0x4 (halfword 2)	0x04	INCR	16-bit	1 data transfer
0x5	0x04	INCR	32-bit	1 data transfer
0x6 (halfword 3)	0x06	INCR	16-bit	1 data transfer
0x7	0x04	INCR	32-bit	1 data transfer
	0x08	INCR	32-bit	1 data transfer

The following table shows possible values of **ARADDRP**, **ARBURSTP**, **ARSIZEP**, and **ARLENP** for an `LDR` to Normal memory.

Table 7-25 LDR transfers

Address[2:0]	ARADDRP	ARBURSTP	ARSIZEP	ARLENP
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer
0x1	0x00	INCR	32-bit	1 data transfer
	0x04	INCR	32-bit	1 data transfer
0x2	0x00	INCR	32-bit	1 data transfer
	0x04	INCR	32-bit	1 data transfer
0x3	0x00	INCR	32-bit	1 data transfer
	0x04	INCR	32-bit	1 data transfer
0x4 (word 1)	0x04	INCR	32-bit	1 data transfer
0x5	0x04	INCR	32-bit	1 data transfer
	0x08	INCR	32-bit	1 data transfer
0x6	0x04	INCR	32-bit	1 data transfer
	0x08	INCR	32-bit	1 data transfer
0x7	0x04	INCR	32-bit	1 data transfer
	0x08	INCR	32-bit	1 data transfer

The following table shows possible values of **ARADDRP**, **ARBURSTP**, **ARSIZEP**, and **ARLENP** for an `LDM` that transfers five registers, an `LDM5`, to Normal memory.

Table 7-26 LDM transfers

Address[2:0]	ARADDRP	ARBURSTP	ARSIZEP	ARLENP
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer
	0x04	INCR	32-bit	1 data transfer
	0x08	INCR	32-bit	1 data transfer
	0x0C	INCR	32-bit	1 data transfer
	0x10	INCR	32-bit	1 data transfer
0x4 (word 1)	0x04	INCR	32-bit	1 data transfer
	0x08	INCR	32-bit	1 data transfer
	0x0C	INCR	32-bit	1 data transfer
	0x10	INCR	32-bit	1 data transfer
	0x1C	INCR	32-bit	1 data transfer

LLPP Normal writes

Store instructions accessing Normal memory generate transactions on the LLPP that are not necessarily the same size as the instruction implies.

An LLPP Normal write might generate more than one bus transaction to write the required data.

The LLPP asserts byte-lane strobes, **WSTRBPx[3:0]**, to ensure that only the bytes that were written by the instruction are updated.

The tables in this section give examples of the types of AXI transaction that might result from various store instructions, accessing various addresses in Normal memory. They are provided as examples only, and are not an exhaustive description of the AXI transactions.

The following table shows the values of **AWADDRP**, **AWBURSTP**, **AWSIZEP**, and **AWLENP** for an `STRH` to Normal memory.

Table 7-27 STRH transfers

Address[2:0]	AWADDRP	AWBURSTP	AWSIZEP	AWLENP	WSTRBP
0x0 (halfword 0)	0x00	Incr	32-bit	1 data transfer	0b0011
0x1	0x00	Incr	32-bit	1 data transfer	0b0110
0x2 (halfword 1)	0x00	Incr	32-bit	1 data transfer	0b1100
0x3	0x00	Incr	32-bit	1 data transfer	0b1000
	0x04	Incr	32-bit	1 data transfer	0b0001
0x4 (halfword 2)	0x04	Incr	32-bit	1 data transfer	0b0011
0x5	0x04	Incr	32-bit	1 data transfer	0b0110
0x6 (halfword 3)	0x04	Incr	32-bit	1 data transfer	0b1100
0x7	0x04	Incr	32-bit	1 data transfer	0b1000
	0x08	Incr	32-bit	1 data transfer	0b0001

The following table shows the values of **AWADDRP**, **ARBURSTP**, **AWSIZEP**, and **AWLENP** for an `STR` to Normal memory.

Table 7-28 STR transfers

Address[2:0]	AWADDRP	AWBURSTP	AWSIZEP	AWLENP	WSTRBP
0x0 (word 0)	0x00	Incr	32-bit	1 data transfer	0b1111
0x1	0x00	Incr	32-bit	1 data transfer	0b1110
	0x04	Incr	32-bit	1 data transfer	0b0001
0x2	0x00	Incr	32-bit	1 data transfer	0b1100
	0x04	Incr	32-bit	1 data transfer	0b0011
0x3	0x00	Incr	32-bit	1 data transfer	0b1000
	0x04	Incr	32-bit	1 data transfer	0b0111
0x4 (word 1)	0x04	Incr	32-bit	1 data transfer	0b1111
0x5	0x04	Incr	32-bit	1 data transfer	0b1110
	0x08	Incr	32-bit	1 data transfer	0b0001
0x6	0x04	Incr	32-bit	1 data transfer	0b1100
	0x08	Incr	32-bit	1 data transfer	0b0011
0x7	0x04	Incr	32-bit	1 data transfer	0b1000
	0x08	Incr	32-bit	1 data transfer	0b0111

7.6.3 LLPP timeout

The LLPP includes a timeout mechanism to detect transactions that fail to complete (that is, an address or write data beat is not accepted, or a response is not received) within a programmable time limit.

If timeout detection is enabled, the response to a timeout occurrence is:

- For each timeout occurrence, the originating core asserts output signal **ERREVENTx[15]**.
- If timeout abort is enabled, the transaction failing to complete is aborted and the memory system treats the LLPP as broken. All subsequent transactions targeting the LLPP are aborted.

The timeout duration is determined by setting the IMP_BUSTIMEOUTR.MAXCYCLESBY16LLPP field and enabled by setting IMP_BUSTIMEOUTR.ENABLELLPP. Timeout abort is enabled by setting IMP_BUSTIMEOUTR.ABORTLLPP.

7.7 Flash interface

The Flash interface provides low-latency access to external read-only memory. Writes to the read-only Flash port abort.

The Flash interface conforms to the AXI4 specification and is a 128-bit read-only master interface. It receives requests from the instruction side and data side. You can configure whether data or instruction accesses have the highest priority in the case of contention, by setting CPUACTLR.FLASHARBCTL. Data and instructions read from the Flash interface can be cached in the L1 caches. Flash data allocations can be invalidated under control of an external debugger. There can be up to four outstanding Flash linefill requests in progress. Across Flash and AXIM ports, the Cortex-R52 processor can track a maximum of eight linefills.

CFGFLASHIMP indicates if the Flash interface is implemented. If the Flash interface is implemented and not enabled, accesses falling within the Flash region generate a synchronous external abort. If the Flash interface is not implemented, accesses falling within the Flash region are performed through other parts of the memory system. The base address is set from the **CFGFLASHBASEADDR[31:27]** configuration signals. The Flash interface base address is indicated in IMP_FLASHIFREGIONR. Each core has a separate IMP_FLASHIFREGIONR which also holds the Flash interface enable bit. Whether the Flash interface is enabled at reset is implemented individually for each core, and set by the **CFGFLASHENx** configuration signals.

The Flash region size is 128MB.

The following features are not permitted by AXI4 and the system controls whether they are used.

- The interface can accept received data in the same cycle as the read address is valid. Each core can issue four Flash read requests.
- The Flash interface includes a late error input signal, **LATEERRFx**. The late error signal allows the Flash controller to signal a correctable error and cause the core to discard the data and, if necessary, repeat the transaction.

The following table shows the different types of Flash read requests.

Table 7-29 Flash Read Requests

Access type	Cacheable	Bursts	Transaction size
Read	Non-cacheable	Single	128-bit
	Non-cacheable	Burst, *Increment* (incr)	128-bit
	Cacheable	Burst, *Wrapping* (wrap)	128-bit

All WRAP bursts fetch a complete cache line starting with the critical word first. A burst does not cross a cache line boundary.

The following table shows the Flash interface attributes.

Table 7-30 Flash interface attributes

Attribute	Value	Comments
Read issuing capability	4	Each core can issue four outstanding Flash read requests.
Read ID capability	2	The ID encodes the source of the memory transaction as instruction side or data side access.
Read ID width	1	-

The following table describes the list of IDs for the Flash interface.

Table 7-31 IDs for the Flash interface

ID	Description
0	For data side access
1	For instruction side access

Note

The AXIS interface of the processor must not appear in the memory map of the Flash interface.

7.7.1 AXI privilege information

The Flash interface provides information about the privilege level of an access on the **ARPROTFx[2:0]** signal.

Accesses might be cached or merged together and the resulting transaction can have both privileged and user data combined.

The following table shows the processor privilege level and associated **ARPROTFx[2:0]** value. Cacheable read requests are always marked privileged. Non-cacheable read requests take on the privilege of the instruction that requested the read.

Table 7-32 Processor mode and ARPROTFx values

Processor mode	Type of access	Value of ARPROTFx
EL0, EL1, EL2	Cacheable, Non-shareable read access	Privileged access
EL0	Normal Non-cacheable read access	Unprivileged access
EL1, EL2		Privileged access

Note

All transactions are marked as Secure (**ARPROTFx[1]**=0).

7.7.2 Flash interface timeout

A transaction fails to complete if an address is not accepted, or a response is not received.

If timeout detection is enabled, the response to a timeout occurrence is:

- For each timeout occurrence, the originating core asserts output signal **ERREVENTx[14]**.
- If transaction abort is enabled, the transaction failing to complete is aborted and the memory system treats the Flash interface as broken. All subsequent transactions targeting the Flash interface are aborted.

The timeout duration is determined by setting the IMP_BUSTIMEOUTR.MAXCYCLESBY16FLASH field and enabled by setting IMP_BUSTIMEOUTR.ENABLEFLASH. Transaction abort is enabled by setting IMP_BUSTIMEOUTR.ABORTFLASH.

7.8 AXIS interface

The AXIS interface is shared between all cores in the processor. It provides external access to TCM memory. The AXIS can have bus protection.

The AXIS address space is a 16MB region located at a base address set by configuration input signals **CFGAXISTCMBASEADDR[31:24]**, providing access to all TCM memory within the processor. The base address set by the configuration input signals **CFGAXISTCMBASEADDR[31:24]** must match the address the AXIS port in the system memory map. The 16MB region is divided into four 4MB blocks, one for each core. Each 4MB block is subdivided into four 1MB regions. TCMs (A, B, and C) within a core are mapped to the first three 1MB regions.

The memory map is the same regardless of the number of cores configured, and the sizes and number of TCMs present. Regions associated with cores that are not present generate a DECERR response. If a TCM is accessed outside its configured range, or an unoccupied 1MB region is accessed, a DECERR response is generated.

For each core, access control checks can be enabled for AXIS transactions by programming IMP_SLAVEPCTLR.TCMACCLVL. Access control allows either all transactions or only privileged transactions to access the TCM. If a transaction is not permitted, the AXIS generates an SLVERR response.

When an external agent or core attempts to write to the TCMs through the AXIS interface and the affected core is in the process of handling a `DMB`, the write to the AXIS interface is scheduled to be performed before outstanding writes from the core to the same TCM. If the two writes are to the same TCM line and some of the bytes overlap, the MRP output is adjusted to show the resulting memory contents and the AXIS write is signaled as having been reordered on **MRPATTRx**.

The following figure shows how each TCM is mapped into AXIS address space.

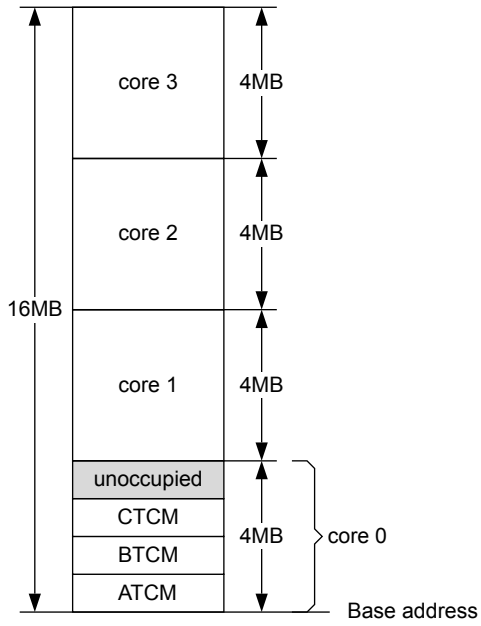

Figure 7-3 AXIS TCM mapping

7.8.1 Accessing TCM with ECC

When TCM implements ECC, the ECC code is generated within the core before writing to the TCM.

If the core detects an error on reads or writes the behavior is:

Writes

- If a write to TCM requires a read-modify-write, and a correctable error is detected when reading the TCM, the ECC is recalculated.
- If a write to TCM requires a read-modify-write, and a non-correctable error is detected when reading the TCM, a SLVERR response is returned.

Reads

- For a correctable error, the core corrects the data in the TCM and returns this corrected data with OKAY response.
- For a non-correctable error, a SLVERR response is returned.

7.8.2 AXIS characteristics

This section describes the capabilities and attributes of the AXIS interface.

The AXIS implements a subset of the AXI transactions, the Accelerator Coherency Port Interface Restrictions, that are described in *AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™ ACE and ACE-Lite™*

The **AWLENS[7:0]** and **ARLENS[7:0]** signals specify the number of data transfers which occur within each burst. In the following signal names, m stands for R or W. The AXIS interface supports two burst lengths:

- 16-bytes, **AmLENS[7:0]** = `0x00`, **AmADDRS[3:0]** = `0b0000`
- 64-bytes, **AmLENS[7:0]** = `0x03`, **AmADDRS[5:0]** = `0b000000`

For all other combinations of **AmLENS** and **AmADDRS**, AXIS generates an error SLVERR.

For AXIS write bursts:

- 16-byte write bursts are permitted to have any combination of valid bytes.
- 64-byte write bursts must have all bytes valid.

The AXIS interface does not support:

- Exclusive transactions, therefore, **AmLOCK** is not used.
- Memory type and cacheability, therefore, **AmCACHE** is not used.
- Security Extensions, therefore, **AmPROT[1]** is not used.
- Data and instruction transaction signaling, therefore, **AmPROT[2]** is not used.
- QoS is not supported, therefore, **AmQOS** is not used.
- Multiple address region signaling is not supported, therefore, **AmREGION** is not used.

The transaction ID width is configurable, with a minimum value of 1 bit.

The following table shows the AXIS interface attributes.

Table 7-33 AXIS interface attributes

Attribute	Value	Comments
Write interleave depth	1	All write data must be presented to the AXIS interface in order
Read data reorder depth	1	The AXIS interface returns all read data in order, even if the bursts have different IDs
Write acceptance capability	12	The maximum number of outstanding write transactions that a slave can accept
Read acceptance capability	12	The maximum number of outstanding read transactions that a slave can accept

7.9 Error detection and handling

This section describes the various error detection and handling features present in the memory system.

The Cortex-R52 processor memories can store ECC for detecting errors in data read from the RAM. Buses can optionally have redundant signals for detecting errors in the transmitted or received signals, or detecting errors in the function of the interconnect.

Error types

ECC codes are computed and checked for a group of data bits which is referred to as a chunk. A RAM location or bus signal may comprise one or more chunks. For example, the Flash read data bus is 128-bits wide but, if configured with 64-bit ECC chunks, separate ECC codes are used for the upper 64 bits of data and the lower 64 bits of data.

A single-bit error refers to a data chunk, including the corresponding ECC code, in which one bit only has the incorrect value. A double-bit error refers to a data chunk, including the corresponding ECC code in which two bits have the incorrect value.

The following figure shows the error detection and correction scheme.

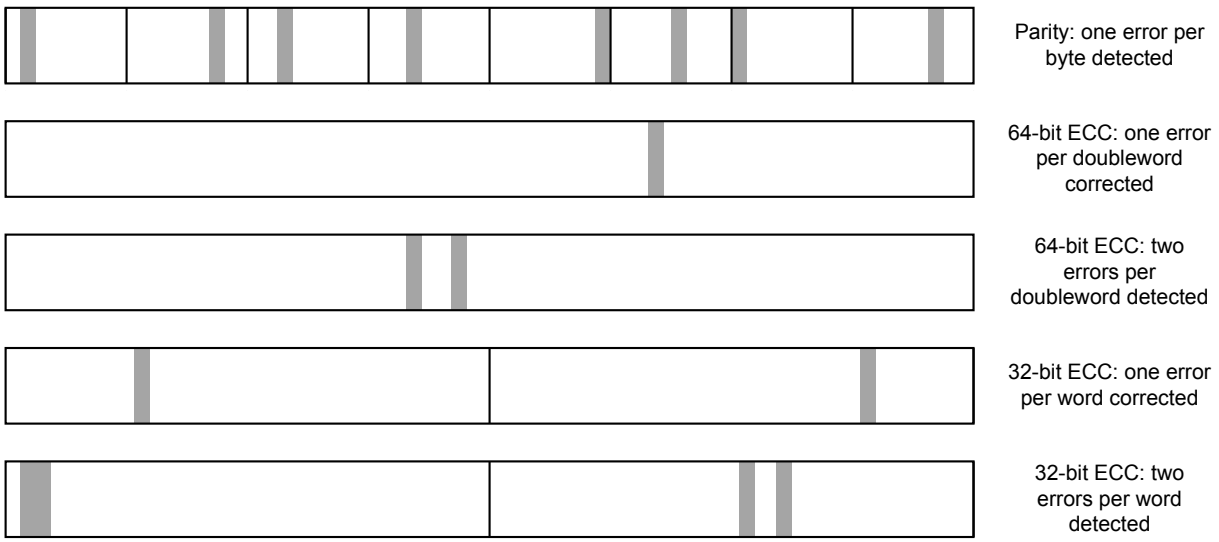

Figure 7-4 Error detection and correction scheme

An error in a RAM is described as a soft error when the RAM location contains an error, but that location can be over-written in the normal way. An error in a RAM is described as a hard error when the RAM has a fault, and that location cannot be over-written, meaning that data written to it are not read. In this document RAM errors are soft errors unless indicated otherwise.

Read-modify write

The smallest unit of data that the processor can write is a byte. However, ECC schemes can be computed on data chucks which are larger than this. To write any data to a RAM protected with ECC requires the error code for that data to be recomputed and rewritten. If the entire data chunk is not written, for example, a halfword, 16 bits, is written to address `0x4` of a RAM with a 32-bit ECC chunk, the error code must be computed partly from the data being written, and partly from data already stored in the RAM. In this example, the halfword in the RAM is at address `0x6`. To compute the error code for such a write, the processor must first read data from the RAM, then merge the data to be written with it to compute the error code, then write the data to the RAM along with the new error code. This process is referred to as read-modify-write.

This section contains the following subsections:

7.9.1 TCM error detection and correction

The TCMs have ECC logic, which can detect and correct single-bit errors and detect double-bit errors in data read from the RAM. The RAMs implemented for the TCMs must be wide enough to store the ECC codes for ECC to be used, otherwise ECC must not be enabled.

The ATCM uses a 64-bit ECC chunk, while the BTCM and CTCM use 32-bit ECC chunks. The core always automatically performs read-modify-write operations as appropriate for the ECC scheme.

At reset, **CFGRAMPROTEN** enables all RAM (TCM and cache) ECC protection. Software can enable and disable ECC protection by writing to register bit IMP_MEMPROTCTLR.RAMPROTEN.

When a correctable error is detected on a read, the correction is written back to the RAM and the relevant instruction is flushed and then replayed. Single-bit hard errors are handled by TCM ECC logic to prevent repetitively correcting the error.

The following table lists the error response when a double-bit error is detected in a TCM.

Table 7-34 Error response

Access type	Response
Core read - Data	Synchronous Data Abort
Core read - Instruction	Prefetch Abort
Core write	Asynchronous Data Abort
AXIS read	SLVERR response on bus

Because double-bit errors cannot be corrected, the TCM does not update the RAM location containing the double-bit error. As a result, subsequent accesses to the same RAM location might observe the same error.

7.9.2 Cache memory ECC

The caches have ECC logic which can detect and correct single-bit and double-bit errors in data or tags that are read from the RAM. The RAMs implemented for the caches must be wide enough to store the ECC codes for ECC to be used, otherwise ECC must not be enabled.

The instruction cache uses a 64-bit ECC chunk for data, while the data cache uses 32-bit ECC chunks. The tags in both caches use an ECC chunk which is the full size of the tag. The core always automatically performs read-modify-write operations as necessary for the data cache.

At reset, **CFGRAMPROTEN** enables all RAM (cache and TCM) ECC protection. Software can enable and disable ECC protection by writing to register bit IMP_MEMPROTCTLR.RAMPROTEN.

Errors that are detected in the instruction and data cache are always correctable by invalidating the cache, flushing the core pipeline, refetching and executing the failed instruction. This is feasible because the caches implement Write-Through caching and can never hold dirty data. This behavior allows the failing cache line to be read from main, or Flash memory.

Single-bit and double-bit hard errors are handled by cache logic to prevent repeatedly correcting the same error.

7.9.3 ECC error reporting

Each core includes two record registers each for the instruction cache, data cache, TCMs, and Flash interface.

When a correctable ECC error is detected the location of the error is recorded in an empty record register, which is then marked as valid. When a non-correctable ECC error is detected, the location of the error is recorded in an empty record register, or a record register containing a correctable error.

Record registers become empty only by a software write to the register.

Detected error events are signaled to the system using the global **ERREVENT** bus or the per-core **ERREVENTx** buses. If both record registers are full, the loss of correctable error information is indicated on **ERREVENTx[16]** signal, for correctable errors and **ERREVENTx[17]** signal for non-correctable errors.

Related references

3.3.57 Instruction Cache Error Record Registers 0 and 1 on page 3-141

3.3.21 Data Cache Error Record Registers 0 and 1 on page 3-95

3.3.93 TCM Error Record Register 0 and 1 on page 3-188

3.3.29 Flash Error Record Registers 0 and 1 on page 3-102

7.9.4 Bus protection

The AXIM interface, the LLPP, the Flash interface, and the AXIS interface, optionally, implement ECC protection for signal integrity.

ECC, when implemented, protects:

Payload data

The core checks and corrects read data and generates ECC codes for write data. The ECC chunk size, for data, depends on the interface:

- The AXIM ECC chunk size is 64 bits.
- The LLPP ECC chunk size is 32 bits.
- The Flash interface is configurable for 64-bit or 128-bit ECC chunk.
- The AXIS ECC chunk size is 64 bits.

Payload address, control, and response signals

The core checks response signals received from the bus and generates ECC codes for address and control signals sent to the bus. The ECC chunk size is 32 bits for addresses and no more than 32 bits for control and response payload signals.

Parity protection of handshake signals

The AXIM interface, the LLPP, the Flash interface, and the AXIS bus, optionally, implement parity protection of handshake signals for signal integrity. The core checks handshake signals that are received from the bus, and generates parity bits for handshake signals that are sent to the bus.

Interconnect protection

The AXIM interface, the LLPP, the Flash interface, and the AXIS bus, optionally, implement interconnect protection. The core checks read data and write response beats received, and generates additional information for address, control, and write data beats that are sent to the bus.

All single-bit errors in data read from the AXIM interface, the LLPP, the Flash interface, and the AXIS bus interface are correctable, that is, the function of the software is unchanged by such errors, although the timing of operation might be affected.

All double-bit errors in data read from each interface are uncorrectable. Single and double-bit errors in response signals received on the LLPP are uncorrectable. Uncorrectable errors either generate an external

abort or the LLPP ignores these transactions. All errors generate events as described in *12.6 Events* on page 12-419.

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

Related references

7.5 AXIM interface on page 7-228
A.8.1 AXIM interface signals on page Appx-A-560
7.8 AXIS interface on page 7-243
A.8.2 AXIS interface signals on page Appx-A-564

7.9.5 Flash data ECC

Data read from the Flash interface has ECC protection regardless of whether or not signal integrity protection is implemented. If signal integrity protection is implemented, the Flash data ECC forms part of the bus protection scheme. Errors are recorded in the Flash Error Record Registers. Read-modify-write is not required because the Flash interface is read-only.

Flash data protection is enabled at reset by **CFGFLASHPROTEN**. Software can enable and disable flash data protection by writing to the IMP_MEMPROTCTLR.FLASHPROTEN bit. If signal integrity protection is implemented, ECC protection on data is always performed, irrespective of the state of FLASHPROTEN.

The Flash interface can also receive correctable errors signaled by the external controller. These are known as late errors and cause the core to refetch and execute the instruction. Late errors never generate aborts. Late errors are recorded in the Flash Error Record Registers.

7.10 Exclusive accesses

Each core has a local exclusive monitor supporting Load-Exclusive and Store-Exclusive instructions.

The local monitor enables the TCMs or Non-shareable memory to be used for semaphores between processes running on the same core. The local monitor is not available for use with other cores or external masters accessing the TCMs through the AXIS. The local monitor exclusive granule size is 512-bits.

A Load-Exclusive instruction marks a 512-bit memory region as exclusive. A Store-Exclusive passes the local exclusive monitor check if it accesses the same 512-bit memory region and returns the exclusive monitor to the open access state. If a Store-Exclusive accesses an address outside the marked 512-bit region, then the Store-Exclusive instruction fails the exclusive monitor check and the monitor is returned to the open access state.

If a non-exclusive store is attempted when the monitor is in the exclusive access state, then the local exclusive monitor state is not affected, regardless of the address being accessed. If a Load-Exclusive is executed while the local exclusive monitor is already in the exclusive access state, the monitor remains in the exclusive access state, but the marked 512-bit address region is updated.

The local monitor also acts as a filter for Shareable exclusive accesses before they are sent to the global exclusive monitor. A Load-Exclusive to shareable memory sets the local monitor when it retires cleanly. A subsequent Store-Exclusive to shareable memory checks the local monitor before being propagated externally. If it fails, the store is not propagated externally and the Store-Exclusive immediately retires with an exclusive failure. This filtering behavior is possible because a failed local exclusive monitor check implies that the global exclusive monitor check would also fail.

The Flash interface is read-only, there exclusives are not supported. The LLPP interface is 32-bits wide, therefore, doubleword exclusives are not supported.

Exclusives to Outer Shareable memory on the AXIM or LLPP are performed as exclusive transactions on the bus and require an external global exclusive monitor to support exclusives. Device and Normal Non-cacheable memory is treated as Outer Shareable, and therefore also requires a global exclusive monitor on the AXIM and LLPP interfaces to support exclusives.

The AXIS interface does not support exclusives.

7.11 Bus timeouts

The memory system has three distinct bus timeouts. For more information, see:

- *7.6.3 LLPP timeout* on page 7-240.
- *7.7.2 Flash interface timeout* on page 7-242.
- *7.5.7 AXIM interface timeout* on page 7-232.

Chapter 8
Memory Protection Unit

This chapter describes the MPU.

It contains the following sections:

8.1 About the MPU

The Cortex-R52 processor has two programmable MPUs, controlled from EL1 and EL2. Each MPU allows the 4GB memory address range to be subdivided into regions.

Each memory region is defined by a base address, limit address, access permissions, and memory attributes.

For more information on the MPU, see the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

For data accesses, the MPU checks that the type of access (read or write) to a region is allowed for the current translation regime. For instruction accesses, the MPU checks access is allowed in the region, and that the translation regime allows execution. For both data and instruction accesses, if access is allowed, the MPU assigns the memory attributes defined for the region. If access is not allowed, a permission fault is taken. A translation fault is taken for the following reasons:

- If an access hits in more than one region in one of the MPUs.
- If an access does not hit in any MPU region and the background region cannot be used (based on the MPU configuration and current privilege level).

As a result of pipelined operation, the processor tries to predict program flow and future data accesses, and so it fetches data and instructions ahead of their use. These transactions are known as speculative transactions unless or until the pipeline completes execution of the corresponding instruction. This might result in the processor generating addresses either outside permitted regions, or not having privilege to attempt the access. In these cases, speculative accesses are prevented from generating bus transactions by the MPU but do not raise a translation or permission fault.

Each processor core has an EL1-controlled MPU with 16, 20, or 24 programmable regions, and an EL2-controlled MPU which optionally supports 0, 16, 20, or 24 programmable regions. When the EL2-controlled MPU and virtualization are enabled, all transactions using the EL0/EL1 translation regime perform a lookup in both MPUs. The resulting attributes are combined so that the least permissive attributes are taken. These two stages of protection allow the hypervisor to retain control over the EL0/EL1 translation regime and therefore enables support for virtualization. When software executes using the EL2 translation regime, only the EL2-controlled MPU is used.

8.2 MPU regions

A region is a contiguous range of addresses starting at a base address, extending up to and including a limit address.

The base address is configured by PRBAR (HPBAR for EL2-controlled MPU) and the limit address is configured by PRLAR (HPRLAR for EL2-controlled MPU). The base address is aligned on a 64-byte boundary and the limit address is aligned to the byte below a 64-byte boundary. Both base and limit addresses are inclusive, meaning that an address within a region is given by:

```
PRBAR.BASE:0b000000 <= address <= PRLAR.LIMIT:0b111111
```

Where : is a bit concatenation operator.

The minimum size for a region is 64 bytes.

PRBAR and PRLAR also hold the access permissions (PRBAR.AP), shareability (PRBAR.SH), the Execute-never bit (PRBAR.XN), and memory attribute index (PRLAR.AttrIndx).

Memory attributes are determined by indexing the *Memory Attribute Indirection Registers* (MAIRx) with PRLAR.AttrIndx.

A region is enabled or disabled by setting or clearing the region enable bit (PRLAR.EN). In the EL2-controlled MPU, regions can also be enabled or disabled by writing to the *Hypervisor MPU Region Enable Register* (HPRENR).

8.2.1 EL1-controlled MPU background region

The EL1-controlled MPU background region is used as a default memory map when the MPU is disabled (SCTLR.M=0).

When the MPU is enabled, the background region can be enabled by setting SCTLR.BR. In this case, accesses from the EL1 translation regime that do not hit any programmable regions use the background region. Accesses from the EL0 translation regime are faulted when the MPU is enabled.

EL1-controlled MPU background region, instruction access

The following table shows the EL1-controlled MPU background region for instruction access.

Table 8-1 EL1-controlled MPU background region - instruction access

Address range	Instruction caching (SCTLR.I)		Execute-never (XN)
	Enabled	Disabled	
0x00000000–0x7FFFFFFF	Normal, Non-shareable, Write-Through Cacheable	Normal, Shareable, Non-cacheable	Execution permitted
0x80000000–0xFFFFFFFF	-	-	Execute-never

EL1-controlled MPU background region, data access

The following table shows the EL1-controlled MPU background region for data access.

Table 8-2 EL1-controlled MPU background region - data access

Address range	Data caching (SCTLR.C)	
	Enabled	Disabled
0x00000000–0x3FFFFFFF	Normal, Non-shareable, Write-Back, Write-Allocate Cacheable	Normal, Shareable, Non-cacheable
0x40000000–0x5FFFFFFF	Normal, Non-shareable, Write-Through Cacheable	Normal, Shareable, Non-cacheable

Table 8-2 EL1-controlled MPU background region - data access (continued)

Address range	Data caching (SCTLR.C)	
	Enabled	**Disabled**
0x60000000–0x7FFFFFFF	Normal, Shareable, Non-cacheable	Normal, Shareable, Non-cacheable
0x80000000–0xBFFFFFFF	Device-nGnRE	Device-nGnRE
0xC0000000–0xFFFFFFFF	Device-nGnRnE	Device-nGnRnE

8.2.2 EL2-controlled MPU background region

The EL2-controlled background region is used as a default memory map for all accesses when the EL2-controlled MPU is disabled (HSCTLR.M=0).

It can also be used for EL2 accesses that do not hit any programmable regions, when the EL2-controlled MPU is enabled, by setting the background region enable (HSCTLR.BR=1). When the EL2-controlled MPU is enabled, accesses from the EL0/EL1 translation regime that do not hit in the EL2 programmable regions generate a translation fault. This is a result of a two-stage translation.

The following table shows the EL2-controlled MPU background region, for instruction access.

Table 8-3 EL2-controlled MPU background region - instruction access

Address range	Instruction caching (HSCTLR.I)		Execute-never (XN)
	Enabled	**Disabled**	
0x00000000–0x7FFFFFFF	Normal, Non-shareable, Write-Through Cacheable	Normal, Shareable, Non-cacheable	Execution permitted
0x80000000–0xFFFFFFFF	-	-	Execute-never

The following table shows the EL2-controlled MPU background region, for data access.

Table 8-4 EL2-controlled MPU background region - data access

Address range	Data caching (HSCTLR.C)	
	Enabled	**Disabled**
0x00000000–0x3FFFFFFF	Normal, Non-shareable, Write-Back, Write-Allocate Cacheable	Normal, Shareable, Non-cacheable
0x40000000–0x5FFFFFFF	Normal, Non-shareable, Write-Through Cacheable	Normal, Shareable, Non-cacheable
0x60000000–0x7FFFFFFF	Normal, Shareable, Non-cacheable	Normal, Shareable, Non-cacheable
0x80000000–0xBFFFFFFF	Device-nGnRE	Device-nGnRE
0xC0000000–0xFFFFFFFF	Device-nGnRnE	Device-nGnRnE

8.2.3 Default cacheability

When default cacheability is enabled (HCR.DC=1), transactions using the EL1-controlled MPU background region have Normal, Inner Write-Back, Outer Write-Back, Non-Shareable attributes applied with both Read-Allocate and Write-Allocate hints enabled. Instruction accesses that hit in the background region when HCR.DC=1 are always executable.

The default attributes are the most permissive, meaning that when combined with any attribute from the EL2-controlled MPU the resulting attribute is the same as the EL2-controlled MPU attribute. This allows the EL2-controlled MPU to effectively make the EL1-controlled MPU transparent to transactions from the EL1 translation regime that hit in the background region. When HCR.DC=1, all translations from the

EL0/EL1 translation regime perform a two-stage MPU lookup and the processor behaves as if HCR.VM is set.

8.3 Virtualization support

To support virtualization, two stages of MPU lookup are performed.

Virtualization allows processes running at EL1 and EL0 (typically one or more guest operating systems and their applications) to be managed by processes running at EL2 (typically a single hypervisor).

The EL1-controlled MPU checks transactions from processes running at EL0 or EL1 and is programmed by processes running at EL1 or EL2. The EL2-controlled MPU also checks transactions executed from the EL0/EL1 translation regime when virtualization is enabled, and is programmed by software at EL2. Transactions executed under the EL2 translation regime uses the EL2-controlled MPU only.

When virtualization is enabled (HCR.VM=1) and the EL2-controlled MPU is enabled (HSCTLR.M=1), transactions permitted by the EL1-controlled MPU are checked by the EL2-controlled MPU as part of a two stage lookup. If both MPUs permit the transaction, memory attributes from stage 1 are combined with attributes from the matching region in stage 2 and the stricter of the two sets of attributes are applied to the transaction.

8.3.1 Combining MPU memory attributes

When a two-stage lookup is performed, the memory type, cacheability, and shareability attributes from each MPU are combined.

Combining the memory type attribute

The following table shows how the memory type assignments are combined as part of a two-stage lookup.

Table 8-5 Combining the memory type assignments

Assignment in EL1-controlled MPU	Assignment in EL2-controlled MPU	Resultant type
Device-nGnRnE	Any	Device-nGnRnE
Device-nGnRE	Device-nGnRnE	Device-nGnRnE
	Not Device-nGnRnE	Device-nGnRE
Device-nGRE	Device-nGnRnE	Device-nGnRnE
	Device-nGnRE	Device-nGnRE
	Not (Device-nGnRnE or Device-nGnRE)	Device-nGRE
Device-GRE	Device-nGnRnE	Device-nGnRnE
	Device-nGnRE	Device-nGnRE
	Device-nGRE	Device-nGRE
	Device-GRE or Normal	Device-GRE
Normal	Any type of Device	Device type assigned at stage 2
	Normal	Normal

Combining the cacheability attribute

The following table shows how the cacheability assignments are combined as part of a two-stage lookup.

Table 8-6 Combining the cacheability assignments

Assignment in EL1-controlled MPU	Assignment in EL2-controlled MPU	Resultant cacheability
Non-cacheable	Any	Non-cacheable
Any	Non-cacheable	Non-cacheable
Write-Through Cacheable	Write-Through or Write-Back Cacheable	Write-Through Cacheable
Write-Through or Write-Back Cacheable	Write-Through Cacheable	Write-Through Cacheable
Write-Back Cacheable	Write-Back Cacheable	Write-Back Cacheable

Combining the shareability attribute

The following table shows how the shareability assignments are combined as part of a two-stage lookup.

Table 8-7 Combining the shareability assignments

Assignment in EL1-controlled MPU	Assignment in EL2-controlled MPU	Resultant cacheability
Outer Shareable	Any	Outer Shareable
Inner Shareable	Outer Shareable	Outer Shareable
	Inner Shareable	Inner Shareable
	Non-shareable	Inner Shareable
Non-shareable	Outer Shareable	Outer Shareable
	Inner Shareable	Inner Shareable
	Non-shareable	Non-shareable

8.4 MPU register access

The MPU base and limit registers can be accessed indirectly or directly.

Indirectly

A region is selected by writing to the PRSELR (HPRSELR for EL2 MPU). The selected region is programmed by writing to the PRBAR and PRLAR (HPRBAR and HPRLAR for EL2 MPU).

Directly

The base and limit registers, for region n, are directly accessed by encoding the region number into CRm and opcode2 of the following system register access instructions:

CRm = `0b1rrr`, where `rrr` = region_number[3:1].

To access base registers:

op2 = `0b000`, for even-numbered regions.

op2 = `0b100`, for odd-numbered regions.

To access limit registers:

op2 = `0b001`, for even-numbered regions.

op2 = `0b101`, for odd-numbered regions.

Writing base and limit registers:

PRBAR0-PRBAR15	MCR p15, 0, <Rt>, c6, CRm, op2
PRLAR0-PRLAR15	MCR p15, 0, <Rt>, c6, CRm, op2
HPRBAR0-HPRBAR15	MCR p15, 4, <Rt>, c6, CRm, op2
HPRLAR0-HPRLAR15	MCR p15, 4, <Rt>, c6, CRm, op2
PRBAR16-PRBAR24	MCR p15, 1, <Rt>, c6, CRm, op2
PRLAR16-PRLAR24	MCR p15, 1, <Rt>, c6, CRm, op2
HPRBAR16-HPRBAR24	MCR p15, 5, <Rt>, c6, CRm, op2
HPRLAR16-HPRLAR24	MCR p15, 5, <Rt>, c6, CRm, op2

Reading base and limit registers:

PRBAR0-PRBAR15	MRC p15, 0, <Rt>, c6, CRm, op2
PRLAR0-PRLAR15	MRC p15, 0, <Rt>, c6, CRm, op2
HPRBAR0-HPRBAR15	MRC p15, 4, <Rt>, c6, CRm, op2
HPRLAR0-HPRLAR15	MRC p15, 4, <Rt>, c6, CRm, op2
PRBAR16-PRBAR24	MRC p15, 1, <Rt>, c6, CRm, op2
PRLAR16-PRLAR24	MRC p15, 1, <Rt>, c6, CRm, op2
HPRBAR16-HPRBAR24	MRC p15, 5, <Rt>, c6, CRm, op2
HPRLAR16-HPRLAR24	MRC p15, 5, <Rt>, c6, CRm, op2

8.5 MPU Register summary

The following table is a summary of registers affecting the MPU. All registers are accessible through (coproc==0b1111).

Table 8-8 MPU Register summary (coproc==0b1111)

Name	CRn	Op1	CRm	Op2	Reset	Description
PRSELR	c6	0	c2	1	UNK	*3.3.87 Protection Region Selection Register* on page 3-179
PRBAR	c6	0	c3	0	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
PRLAR	c6	0	c3	1	UNK[bi]	*3.3.86 Protection Region Limit Address Register* on page 3-178
HPRBAR	c6	4	c3	0	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
HPRLAR	c6	4	c3	1	UNK[bi]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128
HPRSELR	c6	4	c2	1	UNK	*3.3.50 Hyp Protection Region Selection Register* on page 3-129
HPRENR	c6	4	c1	1	0x00000000	*3.3.46 Hyp MPU Region Enable Register* on page 3-124
HMPUIR	c0	4	c0	4	-[e]	*3.3.47 Hyp MPU Type Register* on page 3-125
MPUIR	c0	0	c0	4	-[b]	*3.3.77 MPU Type Register* on page 3-166

Table 8-9 MPU Register direct access summary (coproc==0b1111)

Name	CRn	Op1	CRm	Op2	Reset	Description
PRBAR0-14 (even)	c6	0	c8-15	0	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
PRBAR1-15 (odd)	c6	0	c8-15	4	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
PRBAR16-24 (even)	c6	1	c8-12	0	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
PRBAR17-23 (odd)	c6	1	c8-11	4	UNK	*3.3.85 Protection Region Base Address Register* on page 3-176
PRLAR0-14 (even)	c6	0	c8-15	0	UNK[bi]	*3.3.86 Protection Region Limit Address Register* on page 3-178
PRLAR1-15 (odd)	c6	0	c8-15	4	UNK[bi]	*3.3.86 Protection Region Limit Address Register* on page 3-178
PRLAR16-24 (even)	c6	1	c8-12	0	UNK[bi]	*3.3.86 Protection Region Limit Address Register* on page 3-178
PRBAR17-23 (odd)	c6	1	c8-11	4	UNK[bi]	*3.3.86 Protection Region Limit Address Register* on page 3-178
HPRBAR0-14 (even)	c6	4	c8-15	0	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
HPRBAR1-15 (odd)	c6	4	c8-15	4	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
HPRBAR16-24 (even)	c6	5	c8-12	0	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
HPRBAR17-23 (odd)	c6	5	c8-11	4	UNK	*3.3.48 Hyp Protection Region Base Address Register* on page 3-126
HPRLAR0-14 (even)	c6	4	c8-15	0	UNK[bi]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128
HPRLAR1-15 (odd)	c6	4	c8-15	4	UNK[bi]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128
HPRLAR16-24 (even)	c6	5	c8-12	0	UNK[bi]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128
HPRLAR17-23 (odd)	c6	5	c8-11	4	UNK[bi]	*3.3.49 Hyp Protection Region Limit Address Register* on page 3-128

[bi] The reset value for bit[0] is 0.

Chapter 9
Generic Interrupt Controller

This chapter describes the Cortex-R52 processor implementation of the *Generic Interrupt Controller* (GIC).

It contains the following sections:

9.1 About the GIC

The GIC is a resource for supporting and managing interrupts in a cluster system. It supports interrupt prioritization, interrupt routing to a core or export port, interrupt preemption, and interrupt virtualization.

The Cortex-R52 processor implements one internal GIC Distributor for the processor and one GIC CPU interface per core. The GIC Distributor contains one GIC Redistributor per core. The GIC Distributor and the GIC CPU interface are described in the *Arm® Generic Interrupt Controller Architecture Specification GIC architecture version 3.0 and version 4.0*.

This chapter describes only features that are specific to the Cortex-R52 processor implementation.

9.2 GIC functional description

The GIC Distributor receives wired interrupts from peripherals (SPIs) and from the cores (PPIs).

The GIC Distributor arbitrates the interrupts that are routed to each core to determine the *Highest Priority Pending Interrupt* (HPPI) which is then communicated to the GIC CPU interface. Activations, deactivations, and software generated interrupts from the core pass through the GIC CPU interface to update the state of the GIC Distributor. The GIC Distributor contains memory-mapped configuration and status registers. The GIC Distributor contains one Redistributor for each interrupt target. A Redistributor is mainly responsible for containing the registers for PPIs and SGIs.

The GIC CPU interface comprises a part that handles physical interrupts and a part that handles virtual interrupts. The GIC CPU interface contains configuration and status registers that are accessible as system registers. An interrupt is configured as either a Group 0 interrupt or a Group 1 interrupt. Group 0 interrupts are signaled with FIQ and Group 1 interrupts are signaled with IRQ.

The GIC CPU interface part that handles physical interrupts receives the HPPI from the GIC Distributor. It performs preemption calculation and signals FIQ or IRQ to the core. Activations, deactivations, and software generated interrupts from the core are passed back through the CPU interface to the Distributor.

The GIC CPU interface part that handles virtual interrupts can contain up to four virtual interrupts, the highest priority of which is used to generate a virtual FIQ or virtual IRQ to the core.

Note

When multiple SPIs with equal priority are sent to the processor, then the interrupt with the lowest ID is selected first.

This section contains the following subsections:

- *9.2.1 GIC Distributor memory map* on page 9-262.
- *9.2.2 Interrupt sources* on page 9-264.
- *9.2.3 Optional export interface* on page 9-264.

9.2.1 GIC Distributor memory map

The GIC Distributor registers are memory-mapped, with a physical base address specified by **CFGPERIPHBASE[31:21]**.

The **CFGPERIPHBASE** value is sampled during reset into CBAR for each core in the device.

The GIC Distributor registers are grouped into 64KB pages. There is one page for the GIC Distributor (GICD_*) registers and two pages per core for the GICR registers. All the GIC Redistributor (GICR_*) registers are accessible to all the Cortex-R52 cores.

Ensure that the memory region used for the GIC Distributor is configured as Device nGnRnE.

The following table shows the GIC Distributor memory map of a Cortex-R52 processor. It lists the address offsets for the 64KB pages relative to the **CFGPERIPHBASE** base address. If an interrupt export interface is implemented, it is the last target in the memory map after the core targets.

Table 9-1 Cortex-R52 processor GIC memory map

Offset range from CFGPERIPHBASE[31:21]	GIC block
0x000000-0x00FFFF	Distributor Registers (GICD_*).
0x010000-0x0FFFFF	Reserved.
0x100000-0x10FFFF	Redistributor Registers (GICR_*) for Control target 0.
0x110000-0x11FFFF	Redistributor Registers (GICR_*) for SGIs and PPIs target 0.
0x120000-0x12FFFF	Redistributor Registers (GICR_*) for Control target 1.

Table 9-1 Cortex-R52 processor GIC memory map (continued)

Offset range from CFGPERIPHBASE[31:21]	GIC block
0x130000-0x13FFFF	Redistributor Registers (GICR_*) for SGIs and PPIs target 1.
0x140000-0x14FFFF	Redistributor Registers (GICR_*) for Control target 2.
0x150000-0x15FFFF	Redistributor Registers (GICR_*) for SGIs and PPIs target 2.
0x160000-0x16FFFF	Redistributor Registers (GICR_*) for Control target 3.
0x170000-0x17FFFF	Redistributor Registers (GICR_*) for SGIs and PPIs target 3.
0x180000-0x18FFFF	Redistributor Registers (GICR_*) for Control target 4.
0x190000-0x19FFFF	Redistributor Registers (GICR_*) for SGIs target 4.

The following table shows the mapping of target IDs to cores and export ports depending on the Cortex-R52 processor.

In the following table, x indicates the number of cores.

Note

The export port is optional. If an export port is excluded in the configuration, the following table has the same entries but without the export port fields.

Table 9-2 Cortex-R52 processor target ID mapping

Target ID	x=1	x=2	x=3	x=4
0	Core 0	Core 0	Core 0	Core 0
1	Export port	Core 1	Core 1	Core 1
2	-	Export port	Core 2	Core 2
3	-	-	Export port	Core 3
4	-	-	-	Export port

Related references

3.3.17 Configuration Base Address Register on page 3-88

9.2.2 Interrupt sources

The GIC Distributor receives the following interrupt types:

Private Peripheral Interrupts (PPIs)

A wired interrupt generated by a peripheral that is specific to a single core. There are 16 PPIs, INTID16-INTID31, per core. The export interface does not support PPIs. Unallocated PPIs the core does not need are made available as extra interrupt inputs at the Cortex-R52 processor top level. Unallocated PPIs can be configured to be either rising-edge triggered or active-LOW level-sensitive. Assigned PPIs from the core peripherals have fixed configuration of level-sensitive.

The following table shows the PPI assignments.

Note

PPI INTIDs 16-21, 28, 29, and 31 are not allocated and are made available as top-level ports on Cortex-R52.

Table 9-3 PPI assignments

INTID	PPI
30	Non-secure physical timer interrupt
27	Virtual timer interrupt
26	Hypervisor timer interrupt
25	Virtual CPU Interface Maintenance interrupt
24	Cross Trigger Interface interrupt
23	Performance Monitor Counter Overflow interrupt
22	Debug Communications Channel interrupt

Shared Peripheral Interrupts (SPIs)

A wired interrupt generated by a peripheral that is routed to one specific core. Routing is configured by software programming the routing information for that SPI. Up to 960 SPIs are configurable. The SPIs are identified as INTID32-INTID991. SPIs are made available as Cortex-R52 processor top-level inputs. Each SPI can be configured to be either rising-edge triggered or active-HIGH level-sensitive. SPI[0] corresponds to INTID32 through to SPI[30] corresponds to INTID991.

For each of the cores in a processor, 32 unique SPIs are routed to that core using low latency hardware. The same SPIs route to all other cores using normal latency hardware. For core x, INTID(32x+32) to INTID(32x+63) are low latency, where x=0, 1, 2, or 3. For example, for core 1, INTID64-INTID95 are low latency.

Software Generated Interrupts (SGIs)

SGIs are generated by writing to an SGI generation system register in the GIC CPU interface. There are 16 SGIs, INTID0-INTID15, that can be generated for each core. An SGI has edge-triggered properties. The software triggering of the interrupt is equivalent to the edge transition of the interrupt signal on a peripheral input.

9.2.3 Optional export interface

The GIC Distributor provides an optional export interface to allow an external device to be a target to which interrupts can be routed in the same way as to a core. The following figure shows the HPPI from the GIC Distributor to the export interface and the interrupt ACTIVATIONs, DEACTIVATIONs, and SGIs from the export interface. The operation of the export interface is the same as for the GIC CPU interface with the exception that WAKE_INFO is only relevant to CPU cores.

The target number for the export interface is derived from the number of cores and the number of GIC external devices. To route an SPI to the export interface, you must configure the corresponding GICD_IROUTERn register with the target number of the export interface. These SPIs must have their priority, group, and edge or level trigger type configured using the GICD registers as per the standard SPI configuration.

When these SPIs are enabled, the corresponding interrupt group or groups are enabled and the ProcessorSleep bit is 0 for the export interface. The GIC Distributor presents the HPPI to the external device over the export interface. The external device normally responds by activating and then deactivating the SPI. For edge triggered SPIs, it is possible to simultaneously activate and deactivate an interrupt. For level sensitive SPIs, activation is followed by some action by the external device to cause deassertion of the corresponding interrupt signal. The external device then performs deactivation. External devices can also pass SGIs to a core by passing a message to the GIC Distributor. PPIs are not supported for external devices connected to the export interface.

The GIC Distributor does not provide a memory mapped bus interface for the export interface, that is, one of the cores has the responsibility for configuring the GIC Distributor on behalf of the external device.

The following figure is a block diagram of the GIC.

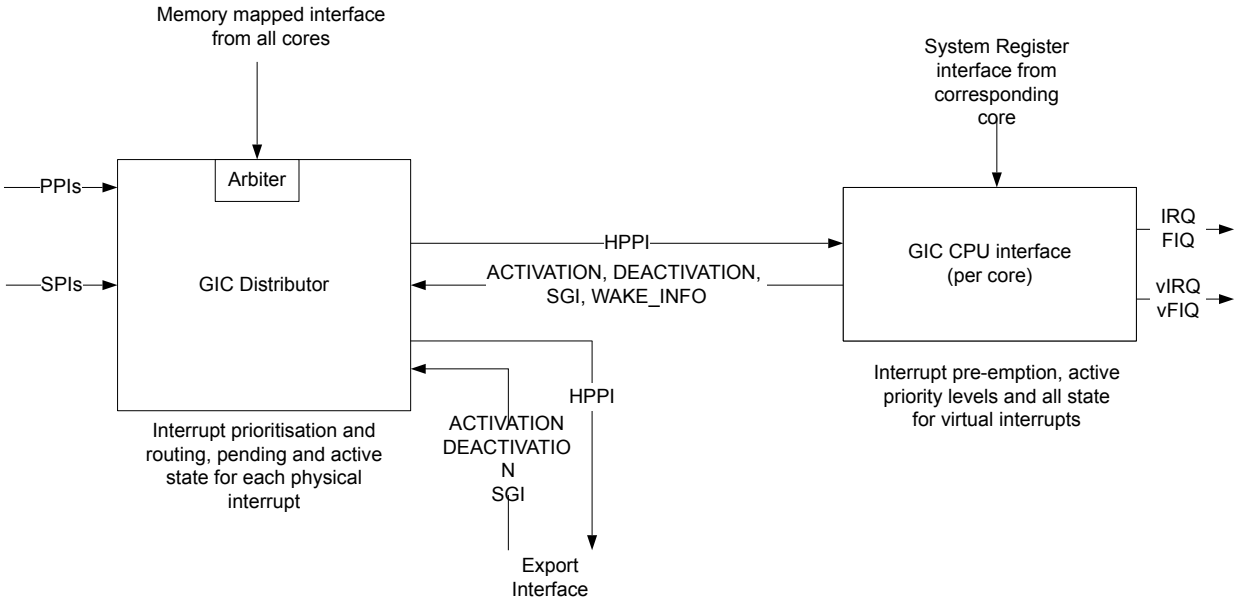

Figure 9-1 GIC block diagram

9.3 GIC programmers model

In this section, the Distributor Registers (GICD) and Redistributor Registers (GICR) describe the memory-mapped registers of the GIC Distributor. The GICR are part of the Cortex-R52 GIC Distributor. The redistributor registers are per core.

In this section, Hypervisor Control System Registers (ICH), CPU interface Registers (ICC), and Virtual CPU interface Registers (ICV) describe the system registers of the GIC CPU interface.

This section contains the following subsections:

9.3.1 Distributor Registers (GICD)

This section describes the implemented GICD registers in the Cortex-R52 processor. The following table summarizes the GICD registers.

Table 9-4 Distributor Registers (GICD) summary

Offset	Name	Type	Reset	Description
0x0000	GICD_CTLR	RW	0x00000050	*Distributor Control Register* on page 9-267.
0x0004	GICD_TYPER	RO	0x0248001E	*Interrupt Controller Type Register* on page 9-269.
0x0008	GICD_IIDR	RO	0x0101343B	*Distributor Implementer Identification Register* on page 9-270.
0x0084-0x00F8	GICD_IGROUPR1-30	RW	0x00000000[bk]	*Interrupt Group Registers 1-30* on page 9-271.
0x0104-0x0178	GICD_ISENABLER1-30	RW	0x00000000[bk]	*Interrupt Set-Enable Registers 1-30* on page 9-271.
0x0184-0x01F8	GICD_ICENABLER1-30	RW	0x00000000[bk]	*Interrupt Clear-Enable Registers 1-30* on page 9-272.
0x0204-0x0278	GICD_ISPENDR1-30	RW	0x00000000[bk]	*Interrupt Set-Pending Registers 1-30* on page 9-273.
0x0284-0x02F8	GICD_ICPENDR1-30	RW	0x00000000[bk]	*Interrupt Clear-Pending Registers 1-30* on page 9-274.
0x0304-0x0378	GICD_ISACTIVER1-30	RW	0x00000000[bk]	*Interrupt Set-Active Registers 1-30* on page 9-275.
0x0384-0x03F8	GICD_ICACTIVER1-30	RW	0x00000000[bk]	*Interrupt Clear-Active Registers 1-30* on page 9-276.
0x0420-0x07DF	GICD_IPRIORITYR8-247	RW	0x00000000[bk]	*Interrupt Priority Registers 8-247* on page 9-277.
0x0C08-0x0CF4	GICD_ICFGR2-61	RW	0x00000000[bk]	*Interrupt Configuration Registers 2-61* on page 9-278.
0x6100-0x7EF8	GICD_IROUTER32-991	RW	-[bj]	*Interrupt Routing Registers 32-991* on page 9-279.
0xFFE0	GICD_PIDR0	RO	0x00000092	*Identification Registers 0-7* on page 9-280.
0xFFE4	GICD_PIDR1	RO	0x000000B4	*Identification Registers 0-7* on page 9-280.
0xFFE8	GICD_PIDR2	RO	0x0000003B	*Identification Registers 0-7* on page 9-280.
0xFFEC	GICD_PIDR3	RO	0x00000000	*Identification Registers 0-7* on page 9-280.
0xFFD0	GICD_PIDR4	RO	0x00000044	*Identification Registers 0-7* on page 9-280.
0xFFD4	GICD_PIDR5	RO	0x00000000	*Identification Registers 0-7* on page 9-280.

[bj] Reset value depends on Cortex-R52 top-level affinity configuration. This value is replicated across the group of registers.

Table 9-4 Distributor Registers (GICD) summary (continued)

Offset	Name	Type	Reset	Description
0xFFD8	GICD_PIDR6	RO	0x00000000	*Identification Registers 0-7* on page 9-280.
0xFFDC	GICD_PIDR7	RO	0x00000000	*Identification Registers 0-7* on page 9-280.
0xFFF0	GICD_CIDR0	RO	0x0000000D	*Component Identification Registers 0-3* on page 9-281.
0xFFF4	GICD_CIDR1	RO	0x000000F0	*Component Identification Registers 0-3* on page 9-281.
0xFFF8	GICD_CIDR2	RO	0x00000005	*Component Identification Registers 0-3* on page 9-281.
0xFFFC	GICD_CIDR3	RO	0x000000B1	*Component Identification Registers 0-3* on page 9-281.

Note

The following correspondence applies:

- GICD_IGROUPR1[0] corresponds to INTID32 through to GICD_IGROUPR30[31] corresponds to INTID991.
- GICD_ISENABLER1[0] corresponds to INTID32 through to GICD_ISENABLER30[31] corresponds to INTID991.
- GICD_ICENABLER1[0] corresponds to INTID32 through to GICD_ICENABLER30[31] corresponds to INTID991.
- GICD_ISPENDR1[0] corresponds to INTID32 through to GICD_ISPENDR30[31] corresponds to INTID991.
- GICD_ICPENDR1[0] corresponds to INTID32 through to GICD_ICPENDR30[31] corresponds to INTID991.
- GICD_ISACTIVE1[0] corresponds to INTID32 through to GICD_ISACTIVE30[31] corresponds to INTID991.
- GICD_ICACTIVE1[0] corresponds to INTID32 through to GICD_ICACTIVE30[31] corresponds to INTID991.
- GICD_IPRIORITYR8[7:0] corresponds to INTID32 through to GICD_IPRIORITYR247[31:24] corresponds to INTID991.
- GICD_ICFGR2[1:0] corresponds to INTID32 through to GICD_ICFGR61[31:30] corresponds to INTID991.
- GICD_IROUTER32 corresponds to INTID32 through to GICD_IROUTER991 corresponds to INTID991.

Distributor Control Register

The GICD_CTLR register independently enables interrupt group 0 and group 1 from the GIC Distributor to the GIC CPU interfaces.

Usage constraints This register is read/write.

Traps and enables There are no traps and enables affecting this register.

Configurations This register is available in all configurations.

Attributes GICD_CTLR is a 32-bit register.

The following figure shows the GICD_CTLR bit assignments.

[bk] This reset value is replicated across the group of the registers.

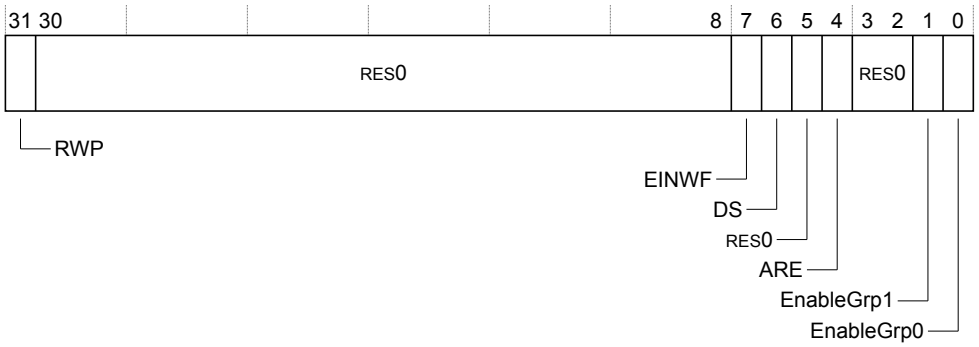

Figure 9-2 GICD_CTLR bit assignments

The following table shows the GICD_CTLR bit assignments.

Table 9-5 GICD_CTLR bit assignments

Bits	Name	Function
[31]	RWP	Register Write Pending. This read-only bit indicates whether a register write is in progress or not: `0` The effect of all previous writes to the following registers are visible to the GIC CPU interface. `1` The effect of all previous writes to the following registers are not guaranteed to be visible to the GIC CPU interface because the changes are still being propagated. Registers: • GICD_CTLR[1:0] – the clearing of Group Enables. • GICD_ICENABLERn. The clearing of enable state for SPIs. No other updates to register fields are tracked by this field.
[30:8]	-	Reserved, RES0.
[7]	EINWF	Enable 1 of N Wakeup functionality: **RAZ/WI** 1 of N interrupt distribution. Not supported.
[6]	DS	Disable Security: **RAO/WI** Security is not supported.
[5]	-	Reserved, RES0.
[4]	ARE	Affinity Routing Enable: **RAO/WI** Only supports GICv3 software access to the GIC Distributor.
[3:2]	-	Reserved, RES0.
[1]	EnableGrp1	Enable Group 1 interrupts from the GIC Distributor to the GIC CPU interfaces: `0` Group 1 interrupts are disabled. This is the reset value. `1` Group 1 interrupts are enabled.
[0]	EnableGrp0	Enable Group 0 interrupts from the Distributor to the GIC CPU interfaces: `0` Group 0 interrupts are disabled. This is the reset value. `1` Group 0 interrupts are enabled.

GICD_CTLR can be accessed through its memory-mapped interface:

Table 9-6 GICD_CTLR access information

Component	Offset	Reset
GIC Distributor	0x0000	0x00000050

Interrupt Controller Type Register

The GICD_TYPER provides information about the configuration of the GIC Distributor.

Usage constraints This register is read-only.

Traps and enables There are no traps and enables affecting this register.

Configurations This register is available in all configurations.

Attributes GICD_TYPER is a 32-bit register.

The following figure shows the GICD_TYPER bit assignments.

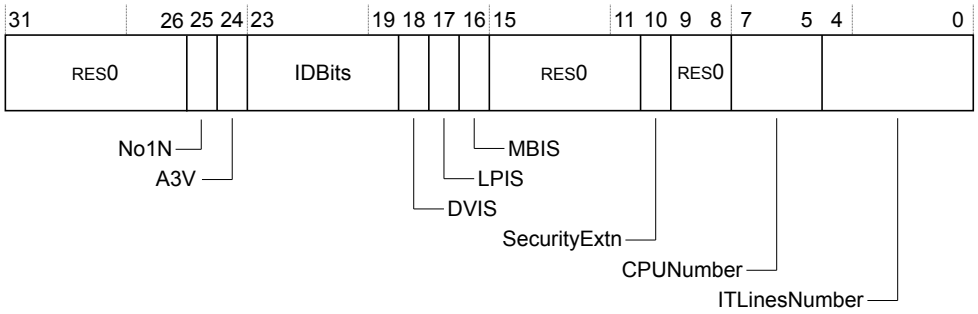

Figure 9-3 GICD_TYPER bit assignments

The following table shows the GICD_TYPER bit assignments.

Table 9-7 GICD_TYPER bit assignments

Bits	Name	Function
[31:26]	-	Reserved, RES0.
[25]	No1N	Indicates whether the GIC Distributor supports 1 of N SPIs: 1 1 of N SPI interrupts are not supported.
[24]	A3V	Indicates whether the GIC Distributor supports nonzero values of Affinity level 3: 0 The Distributor only supports zero values of Affinity level 3.
[23:19]	IDBits	Indicates the number of INTID bits that the GIC Distributor supports, minus one: 0b01001 INTID is 10 bits.
[18]	DVIS	Indicates whether the GIC Distributor supports Direct Virtual LPI injection: 0 Direct Virtual LPI injection is not supported.
[17]	LPIS	Indicates whether the GIC Distributor supports Locality-specific Peripheral Interrupts (LPIs): 0 LPIs are not supported.
[16]	MBIS	Indicates whether the GIC Distributor supports Message-Based Interrupts Supported: 0 Message-Based Interrupts Supported are not supported.
[15:11]	-	Reserved, RES0.

Table 9-7 GICD_TYPER bit assignments (continued)

Bits	Name	Function
[10]	SecurityExtn	Indicates whether the GIC Distributor supports two Security states: `0` Single Security state supported.
[9:8]	-	Reserved, RES0.
[7:5]	CPUNumber	Indicates the number of cores that can be used as interrupt targets when GICD_CTLR.ARE is 0: `0b000` Not supported.
[4:0]	ITLinesNumber	Indicates the number of SPI INTIDs that the GIC Distributor supports. The valid values for this field range from 1 to 30, depending on the number of SPIs configured for Cortex-R52: Valid interrupt INTID range is 0 to 32*(ITLinesNumber + 1) - 1.

GICD_TYPER can be accessed through its memory-mapped interface.

Table 9-8 GICD_TYPER access information

Component	Offset	Reset
GIC Distributor	`0x0004`	`0x0248001E`

Distributor Implementer Identification Register

The GICD_IIDR provides information about the implementer and revision of the Distributor. The revision and variant fields must be read with the MIDR.

Usage constraints This register is read-only.

Traps and enables There are no traps and enables affecting this register.

Configurations This register is available in all configurations.

Attributes GICD_IIDR is a 32-bit register.

The following figure shows the GICD_IIDR bit assignments.

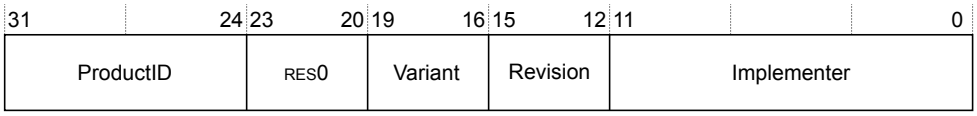

Figure 9-4 GICD_IIDR bit assignments

The following table shows the GICD_IIDR bit assignments.

Table 9-9 GICD_IIDR bit assignments

Bits	Name	Function
[31:24]	ProductID	`0x01`, Cortex-R52 GIC.
[23:20]	-	Reserved, RES0
[19:16]	Variant	`0x1`
[15:12]	Revision	`0x3`
[11:0]	Implementer	`0x43B`, Arm Limited.

GICD_IIDR can be accessed through its memory-mapped interface.

Table 9-10 GICD_IIDR access information

Component	Offset	Reset
GIC Distributor	0x0008	0x0101343B

Related references

3.3.69 Main ID Register on page 3-156

Interrupt Group Registers 1-30

The GICD_IGROUPR1-30 registers control whether the corresponding SPI is in Group 0 or Group 1. Each register contains the group bits for 32 SPIs.

Usage constraints These registers are read/write.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations.

The number of implemented GICD_IGROUP registers is determined by GICD_TYPER.ITLinesNumber.

GICD_IGROUPR registers that are not implemented are RAZ/WI.

Attributes GICD_IGROUPR1-30 are 32-bit registers.

The following figure shows the GICD_IGROUPR1-30 bit assignments.

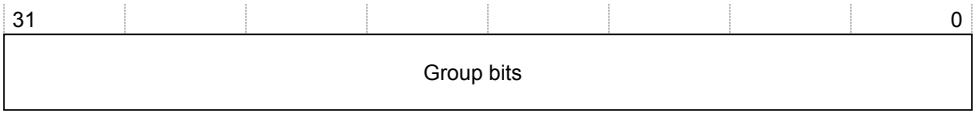

Figure 9-5 GICD_IGROUPR1-30 bit assignments

The following table shows the GICD_IGROUPR1-30 bit assignments.

Table 9-11 GICD_IGROUPR1-30 bit assignments

Bits	Name	Function
[31:0]	Group bits	Controls whether the corresponding SPI is in Group 0 or Group 1: 0 The corresponding SPI is in Group 0. This is the reset value. 1 The corresponding SPI is in Group 1.

GICD_IGROUPR1-30 can be accessed through its memory-mapped interface.

Table 9-12 GICD_IGROUPR1-30 access information

Component	Offset	Reset
GIC Distributor	0x0084-0x00F8	0x00000000

Interrupt Set-Enable Registers 1-30

The GICD_ISENABLER1-30 registers enable forwarding of the corresponding SPI from the Distributor to the CPU interfaces. Each register contains the set-enable bits for 32 SPIs.

Usage constraints These registers are read/write.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations.

The number of implemented GICD_ISENABLER registers is determined by GICD_TYPER.ITLinesNumber.

GICD_ISENABLER registers that are not implemented are RAZ/WI.

Attributes GICD_ISENABLER1-30 are 32-bit registers.

The following figure shows the GICD_ISENABLER1-30 bit assignments.

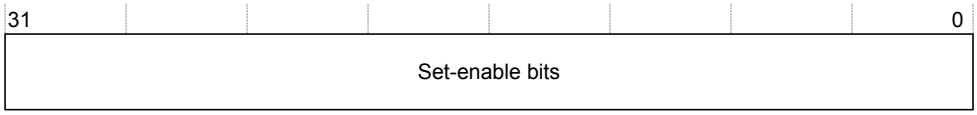

Figure 9-6 GICD_ISENABLER1-30 bit assignments

The following table shows the GICD_ISENABLER1-30 bit assignments.

Table 9-13 GICD_ISENABLER1-30 bit assignments

Bits	Name	Function
[31:0]	Set-enable bits	For each bit: **Reads** `0` Indicates forwarding of SPI is disabled. This is the reset value. `1` Indicates forwarding of SPI is enabled. **Writes** `0` No effect. `1` Enables forwarding of SPI.

GICD_ISENABLER1-30 can be accessed through its memory-mapped interface.

Table 9-14 GICD_ISENABLER1-30 access information

Component	Offset	Reset
GIC Distributor	`0x0104-0x0178`	`0x00000000`

Interrupt Clear-Enable Registers 1-30

The GICD_ICENABLER1-30 registers disable forwarding of the corresponding SPI to the CPU interfaces. Each register contains the clear-enable bits for 32 SPIs.

Usage constraints These registers are read/write.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations of the GIC Distributor.

The number of implemented GICD_ICENABLER registers is determined by GICD_TYPER.ITLinesNumber.

GICD_ICENABLER registers that are not implemented are RAZ/WI.

Attributes GICD_ICENABLER1-30 are 32-bit registers.

The following figure shows the GICD_ICENABLER1-30 bit assignments.

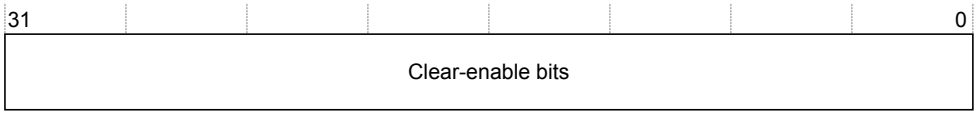

Figure 9-7 GICD_ICENABLER1-30 bit assignments

The following table shows the GICD_ICENABLER1-30 bit assignments.

Table 9-15 GICD_ICENABLER1-30 bit assignments

Bits	Name	Function
[31:0]	Clear-enable bits	For each bit: **Reads** `0` Indicates forwarding of SPI is disabled. This is the reset value. `1` Indicates forwarding of SPI is enabled. **Writes** `0` No effect. `1` Disables forwarding of SPI.

GICD_ICENABLER1-30 can be accessed through its memory-mapped interface.

Table 9-16 GICD_ICENABLER1-30 access information

Component	Offset	Reset
GIC Distributor	`0x0184-0x01F8`	`0x00000000`

Interrupt Set-Pending Registers 1-30

The GICD_ISPENDR1-30 registers set the pending bit for the corresponding SPI. Each register contains the set-pending bits for 32 SPIs.

Usage constraints These registers are read/write.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations of the GIC Distributor.

The number of implemented GICD_ISPENDR registers is determined by GICD_TYPER.ITLinesNumber.

GICD_ISPENDR registers that are not implemented are RAZ/WI.

Attributes GICD_ISPENDR1-30 are 32-bit registers.

The following figure shows the GICD_ISPENDR1-30 bit assignments.

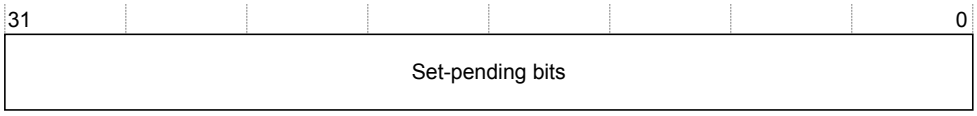

Figure 9-8 GICD_ISPENDR1-30 bit assignments

The following table shows the GICD_ISPENDR1-30 bit assignments.

Table 9-17 GICD_ISPENDR1-30 bit assignments

Bits	Name	Function
[31:0]	Set-pending bits	For each bit: **Reads** `0` Indicates SPI is not pending. `1` Indicates SPI is pending. **Writes** `0` No effect. `1` Sets the pending bit. The reset value is 0.

GICD_ISPENDR1-30 can be accessed through its memory-mapped interface.

Table 9-18 GICD_ISPENDR1-30 access information

Component	Offset	Reset
GIC Distributor	`0x0204-0x0278`	`0x00000000`

Interrupt Clear-Pending Registers 1-30

The GICD_ICPENDR1-30 registers clear the pending bit for the corresponding SPI. Each register contains the clear-pending bits for 32 SPIs.

Usage constraints These registers are read/write.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations of the GIC Distributor.

The number of implemented GICD_ICPENDR registers is determined by GICD_TYPER.ITLinesNumber.

GICD_ICPENDR registers that are not implemented are RES0.

Attributes GICD_ICPENDR1-30 are 32-bit registers.

The following figure shows the GICD_ICPENDR1-30 bit assignments.

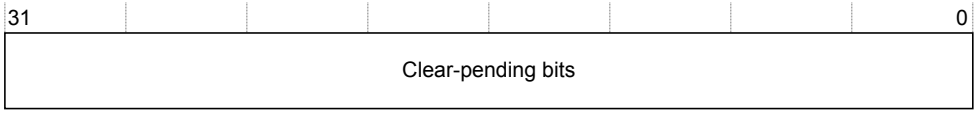

Figure 9-9 GICD_ICPENDR1-30 bit assignments

The following table shows the GICD_ICPENDR1-30 bit assignments.

Table 9-19 GICD_ICPENDR1-30 bit assignments

Bits	Name	Function
[31:0]	Clear-pending bits	For each bit: **Reads** `0` SPI is not pending. `1` SPI is pending. **Writes** `0` No effect. `1` Clears the pending bit.

GICD_ICPENDR1-30 can be accessed through its memory-mapped interface.

Table 9-20 GICD_ICPENDR1-30 access information

Component	Offset	Reset
GIC Distributor	`0x0284-0x02F8`	`0x00000000`

Interrupt Set-Active Registers 1-30

The GICD_ISACTIVER1-30 registers set the active bit for the corresponding SPI. Each register contains the set-active bits for 32 SPIs.

Usage constraints These registers are read/write.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations.

The number of implemented GICD_ISACTIVER registers is determined by GICD_TYPER.ITLinesNumber.

GICD_ISACTIVER registers that are not implemented are RAZ/WI.

Attributes GICD_ISACTIVER1-30 are 32-bit registers.

The following figure shows the GICD_ISACTIVER1-30 bit assignments.

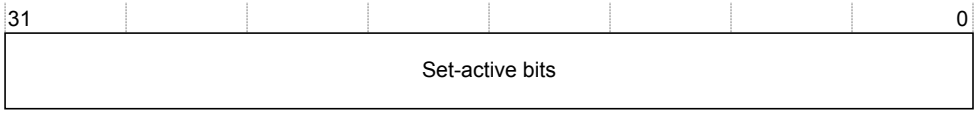

Figure 9-10 GICD_ISACTIVER1-30 bit assignments

The following table shows the GICD_ISACTIVER1-30 bit assignments.

Table 9-21 GICD_ISACTIVER1-30 bit assignments

Bits	Name	Function
[31:0]	Set-active bits	For each bit: **Reads** `0` SPI is not active. `1` SPI is active. **Writes** `0` No effect. `1` Sets the active bit. The reset value is 0.

GICD_ISACTIVER1-30 can be accessed through its memory-mapped interface.

Table 9-22 GICD_ISACTIVER1-30 access information

Component	Offset	Reset
GIC Distributor	`0x0304-0x0378`	`0x00000000`

Interrupt Clear-Active Registers 1-30

The GICD_ICACTIVER1-30 registers clear the active bit for the corresponding SPI. Each register contains the clear-active bits for 32 SPIs.

Usage constraints These registers are read/write.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations.

The number of implemented GICD_ICACTIVER registers is determined by GICD_TYPER.ITLinesNumber.

GICD_ICACTIVER registers that are not implemented are RAZ/WI.

Attributes GICD_ICACTIVER1-30 are 32-bit registers.

The following figure shows the GICD_ICACTIVER1-30 bit assignments.

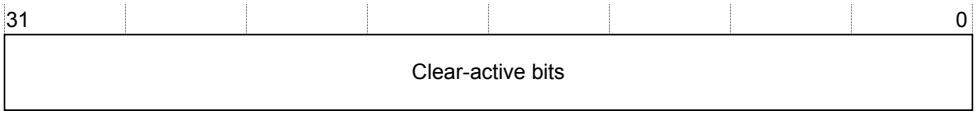

Figure 9-11 GICD_ICACTIVER1-30 bit assignments

The following table shows the GICD_ICACTIVER1-30 bit assignments.

Table 9-23 GICD_ICACTIVER1-30 bit assignments

Bits	Name	Function
[31:0]	Clear-active bits	For each bit: **Reads** `0` SPI is not active. `1` SPI is active. **Writes** `0` No effect. `1` Clears the active bit.

GICD_ICACTIVER1-30 can be accessed through its memory-mapped interface.

Table 9-24 GICD_ICACTIVER1-30 access information

Component	Offset	Reset
GIC Distributor	`0x0384-0x03F8`	`0x00000000`

Interrupt Priority Registers 8-247

The GICD_IPRIORITYR8-247 registers provide a 5-bit priority field for each SPI supported by the GIC. This field stores the priority of the corresponding SPI.

Usage constraints These registers are read/write.

These registers are byte-accessible.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations of the GIC.

The number of implemented GICD_IPRIORITYR8-247 registers is (8× (GICD_TYPER.ITLinesNumber)).

GICD_IPRIORITYR8-247 registers that are not implemented are RAZ/WI.

Attributes GICD_IPRIORITYR8-247 are 32-bit registers.

Each register holds four priority fields as shown in the following figure, where PRI_Nn is the priority corresponding to the SPI with ID INTIDn.

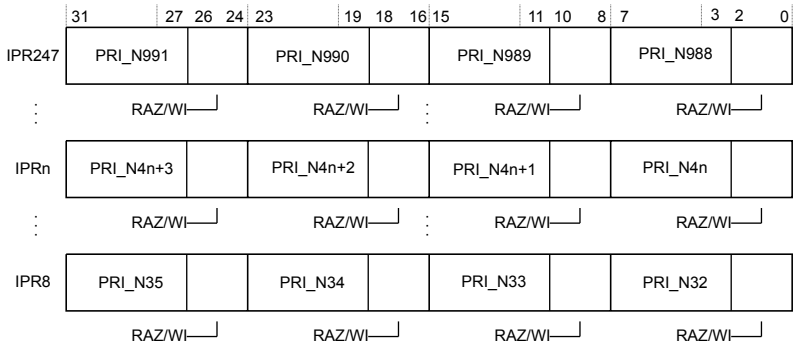

Figure 9-12 GICD_IPRIORITYR8-247 priority fields

The following table shows the GICD_IPRIORITYR8-247 bit assignments.

Table 9-25 GICD_IPRIORITYR8-247 bit assignments

Bits	Name[bl]	Function
[31:27]	Priority, byte offset 3	Each priority field holds a priority value, 0-31. The lower the value, the higher the priority of the corresponding interrupt.
[26:24]	-	Reserved, RAZ/WI.
[23:19]	Priority, byte offset 2	Each priority field holds a priority value, 0-31. The lower the value, the higher the priority of the corresponding interrupt.
[18:16]	-	Reserved, RAZ/WI.
[15:11]	Priority, byte offset 1	Each priority field holds a priority value, 0-31. The lower the value, the higher the priority of the corresponding interrupt.
[10:8]	-	Reserved, RAZ/WI.
[7:3]	Priority, byte offset 0	Each priority field holds a priority value, 0-31. The lower the value, the higher the priority of the corresponding interrupt.
[2:0]	-	Reserved, RAZ/WI.

For INTID*m*, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_IPRIORITYRn number, *n*, is given by *n* = *m* DIV 4, where *m*=32 to 991.
- The address offset of the required GICD_IPRIORITYRn is (`0x400` + (4×*n*)).
- The byte offset of the required Priority field in this register is *m* MOD 4, where:
  - Byte offset 0 refers to register bits[7:3].
  - Byte offset 1 refers to register bits[15:11].
  - Byte offset 2 refers to register bits[23:19].
  - Byte offset 3 refers to register bits[31:27].

GICD_IPRIORITYR8-247 can be accessed through its memory-mapped interface.

Table 9-26 GICD_IPRIORITYR8-247 access information

Component	Offset	Reset
GIC Distributor	`0x0420-0x07DF`	`0x00000000`

Interrupt Configuration Registers 2-61

The GICD_ICFGR2-61 registers provide a 2-bit Int_config field for each interrupt supported by the GIC. This field determines whether the corresponding interrupt is rising edge-triggered or active-HIGH level-sensitive.

Usage constraints These registers are read/write.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations.

The number of implemented GICD_ICFGR2-61 registers is GICD_TYPER.ITLinesNumber x 2.

GICD_ICFGR2-61 registers that are not implemented are RAZ/WI.

Attributes GICD_ICFGR2-61 are 32-bit registers.

The following figure shows the GICD_ICFGR2-61 bit assignments.

[bl] Each field holds the priority value for a single interrupt. This section describes how the INTID value determines the GICD_IPRIORITYRn register number and the byte offset of the priority field in that register.

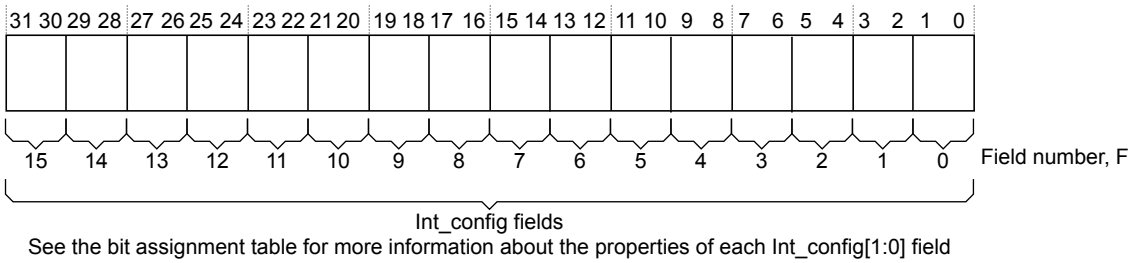

Figure 9-13 GICD_ICFGR2-61 bit assignments

The following table shows the GICD_ICFGR2-61 bit assignments.

Bits	Name	Function
[2*F*+1:2*F*]	Int_config, field *F*	For Int_config[1], the most significant bit, bit [2F+1], the encoding is: `0` Corresponding interrupt is active-HIGH level-sensitive. This is the reset value. `1` Corresponding interrupt is rising edge-triggered. Int_config[0], the least significant bit, bit [2F], is RES0.

For INTID*m*, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_ICFGRn number, *n*, is given by *n* = *m* DIV 16, where *m*=32 to 991.
- The offset of the required GICD_ICFGRn is (`0xC00` + (4×*n*)).
- The required Int_config field in this register, *F*, is given by *F* = *m* MOD 16, where field 0 refers to register bits[1:0], field 1 refers to bits[3:2], up to field 15 that refers to bits[31:30].

GICD_ICFGR2-61 can be accessed through its memory-mapped interface.

Table 9-27 GICD_ICFGR2-61 access information

Component	Offset	Reset
GIC Distributor	`0x0C08-0x0CF4`	`0x00000000`

Interrupt Routing Registers 32-991

The GICD_IROUTER32-991 registers provide routing information for the SPI with GICD_IROUTERn corresponding to INTID*n*.

Usage constraints These registers are read/write.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations of the GIC.

The number of implemented GICD_IROUTER32-991 registers is GICD_TYPER.ITLinesNumber x 32. One 64-bit register per SPI.

GICD_IROUTER32-991 registers that are not implemented are RAZ/WI.

Attributes GICD_IROUTER32-991 are 64-bit registers.

The following figure shows the GICD_IROUTER32-991 bit assignments.

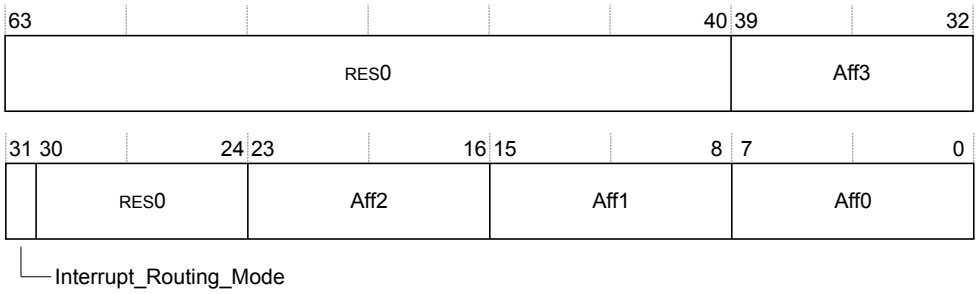

Figure 9-14 GICD_IROUTER32-991 bit assignments

The following table shows the GICD_IROUTER32-991 bit assignments.

Table 9-28 GICD_IROUTER32-991 bit assignments

Bits	Name	Function
[63:40]	-	Reserved, RES0.
[39:32]	Aff3	Affinity level 3. This field is not used: `0` This is the reset value.
[31]	Interrupt_Routing_Mode	Defines how SPIs are routed in an affinity hierarchy: **RAZ/WI** 1 of N distribution of SPIs is not supported.
[30:24]	-	Reserved, RES0.
[23:16]	Aff2	Affinity level 2. This field is read-only and reads returns the Cortex-R52 top-level configuration value **CFGMPIDRAFF2[7:0]**.
[15:8]	Aff1	Affinity level 1. This field is read-only and reads returns the Cortex-R52 top-level configuration value **CFGMPIDRAFF1[7:0]**.
[7:0]	Aff0	Affinity level 0. • The number of implemented LSBs is defined as Aff0[CEILING(LOG2(NUM_TARGET))-1:0] with remaining bits 0. • When NUM_TARGET is 1, all bits are 0. NUM_TARGET is the total number of targets including cores and export ports. • Supported values for Aff0 are defined by the GICR_TYPER. • Aff0 fields of all redistributors which are numbered contiguously from `0` to `NUM_TARGET-1`.

GICD_IROUTER32-991 can be accessed through its memory-mapped interface.

Table 9-29 GICD_IROUTER32-991 access information

Component	Offset	Reset
GIC Distributor	`0x6100-0x7EF8`	`0x0000000000000000`

Identification Registers 0-7

The GICD_PIDR0-7 registers provide the peripheral identification information.

Usage constraints These registers are read only.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations.

Attributes GICD_PIDR0-7 are 32-bit registers.

The following figure shows the GICD_PIDR0-7 bit assignments.

Figure 9-15 GICD_PIDR0-7 bit assignments

The following table shows the GICD_PIDR0-7 bit assignments.

Table 9-30 GICD_PIDR4-7 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	-	Component IDn

GICD_PIDR0-7 can be accessed through its memory-mapped interface.

Table 9-31 GICD_PIDR0-7 access information

Component	Register	Offset	Reset value
GIC Distributor	GICD_PIDR0	0xFFE0	0x00000092
	GICD_PIDR1	0xFFE4	0x000000B4
	GICD_PIDR2	0xFFE8	0x0000003B
	GICD_PIDR3	0xFFEC	0x00000000
	GICD_PIDR4	0xFFD0	0x00000044
	GICD_PIDR5	0xFFD4	0x00000000
	GICD_PIDR6	0xFFD8	0x00000000
	GICD_PIDR7	0xFFDC	0x00000000

Component Identification Registers 0-3

The GICD_CIDR0-3 registers provide the component identification information.

Usage constraints These registers are read-only.

Traps and enables There are no traps and enables affecting these registers.

Configurations These registers are available in all configurations.

Attributes GICD_CIDR0-3 are 32-bit registers.

The following figure shows the GICD_CIDR0-3 bit assignments.

Figure 9-16 GICD_CIDR0-3 bit assignments

The following table shows the GICD_CIDR0-3 bit assignments.

Table 9-32 GICD_CIDR0-3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0
[7:0]	-	Component IDn

GICD_CIDR0-3 can be accessed through its memory-mapped interface.

Table 9-33 GICD_CIDR0-3 reset values

Component	Register	Offset	Reset value
GIC Distributor	GICD_CIDR0	0xFFF0	0x0000000D
	GICD_CIDR1	0xFFF4	0x000000F0
	GICD_CIDR2	0xFFF8	0x00000005
	GICD_CIDR3	0xFFFC	0x000000B1

9.3.2 Redistributor Registers (GICR)

The Redistributor is a component that provides the interface from the Distributor to a core to forward all interrupts that are in the physical domain. The following two tables summarize the Redistributor registers in both the Control and SGI/PPI pages.

Table 9-34 Redistributor Registers (GICR) for Control summary

Offset	Name	Type	Reset	Description
0x0000	GICR_CTLR	RO	0x00000000	*Redistributor Control Register* on page 9-284
0x0004	GICR_IIDR	RO	0x0101343B	*Redistributor Implementer Identification Register* on page 9-285
0x0008-0x000C	GICR_TYPER	RO	0x0000000000000000 For core0 in a two core system. 0x0000000100000110 For core1 in a two core system.	*Redistributor Type Register* on page 9-286
0x0014	GICR_WAKER	RW	0x00000006	*Redistributor Wake Register* on page 9-287
0xFFE0	GICR_PIDR0	RO	0x00000093	*Redistributor Identification Registers 0-7* on page 9-288
0xFFE4	GICR_PIDR1	RO	0x000000B4	*Redistributor Identification Registers 0-7* on page 9-288
0xFFE8	GICR_PIDR2	RO	0x0000003B	*Redistributor Identification Registers 0-7* on page 9-288
0xFFEC	GICR_PIDR3	RO	0x00000000	*Redistributor Identification Registers 0-7* on page 9-288
0xFFD0	GICR_PIDR4	RO	0x00000044	*Redistributor Identification Registers 0-7* on page 9-288
0xFFD4	GICR_PIDR5	RO	0x00000000	*Redistributor Identification Registers 0-7* on page 9-288

Table 9-34 Redistributor Registers (GICR) for Control summary (continued)

Offset	Name	Type	Reset	Description
0xFFD8	GICR_PIDR6	RO	0x00000000	*Redistributor Identification Registers 0-7* on page 9-288
0xFFDC	GICR_PIDR7	RO	0x00000000	*Redistributor Identification Registers 0-7* on page 9-288
0xFFF0	GICR_CIDR0	RO	0x0000000D	*Redistributor Component Identification Registers 0-3* on page 9-289
0xFFF4	GICR_CIDR1	RO	0x000000F0	*Redistributor Component Identification Registers 0-3* on page 9-289
0xFFF8	GICR_CIDR2	RO	0x00000005	*Redistributor Component Identification Registers 0-3* on page 9-289
0xFFFC	GICR_CIDR3	RO	0x000000B1	*Redistributor Component Identification Registers 0-3* on page 9-289

Table 9-35 Redistributor Registers (GICR) for SGIs and PPIs summary

Offset	Name	Type	Reset	Description
0x0080	GICR_IGROUPR0	RW	0x00000000	*Interrupt Group Register 0* on page 9-290
0x0100	GICR_ISENABLER0	RW	0x00000000	*Interrupt Set-Enable Register 0* on page 9-290
0x0180	GICR_ICENABLER0	RW	0x00000000	*Interrupt Clear-Enable Register 0* on page 9-291
0x0200	GICR_ISPENDR0	RW	0x00000000	*Interrupt Set-Pending Register 0* on page 9-292
0x0280	GICR_ICPENDR0	RW	0x00000000	*Interrupt Clear-Pending Register 0* on page 9-293
0x0300	GICR_ISACTIVER0	RW	0x00000000	*Interrupt Set-Active Register 0* on page 9-294
0x0380	GICR_ICACTIVER0	RW	0x00000000	*Interrupt Clear-Active Register 0* on page 9-295
0x0400-0x041C	GICR_IPRIORITYR0-7	RW	0x00000000	*Interrupt Priority Registers 0-7* on page 9-296
0x0C00	GICR_ICFGR0	RO	0xAAAAAAAA	*Interrupt Configuration Register 0* on page 9-298
0x0C04	GICR_ICFGR1	RW	0x00000000	*Interrupt Configuration Register 1* on page 9-298

Note

The following correspondence applies:

- GICR_IGROUPR0[0] corresponds to INTID0 through to GICR_IGROUPR0[31] corresponds to INTID31.
- GICR_ISENABLER0[0] corresponds to INTID0 through to GICR_ISENABLER0[31] corresponds to INTID31.
- GICR_ICENABLER0[0] corresponds to INTID0 through to GICR_ICENABLER0[31] corresponds to INTID31.
- GICR_ISPENDR0[0] corresponds to INTID0 through to GICR_ISPENDR0[31] corresponds to INTID31.
- GICR_ICPENDR0[0] corresponds to INTID0 through to GICR_ICPENDR0[31] corresponds to INTID31.
- GICR_ISACTIVER0[0] corresponds to INTID0 through to GICR_ISACTIVER0[31] corresponds to INTID31.
- GICR_ICACTIVER0[0] corresponds to INTID0 through to GICR_ICACTIVER0[31] corresponds to INTID31.

- GICR_IPRIORITYR0[7:0] corresponds to INTID0 through to GICR_IPRIORITYR7[31:24] corresponds to INTID31.
- GICR_ICFGR0[1:0] corresponds to INTID0 through to GICR_ICFGR0[31:30] corresponds to INTID31.
- GICR_ICFGR1[1:0] corresponds to INTID0 through to GICR_ICFGR1[31:30] corresponds to INTID31.

PPIs correspond to INTID16 to INTID31. SGIs correspond to INTID0 to INTID15.

Redistributor Control Register

The GICR_CTLR register controls the operation of a Redistributor.

Usage constraints This register is read-only.

Traps and enables There are no traps and enables affecting this register.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_CTLR is a 32-bit register.

The following figure shows the GICR_CTLR bit assignments.

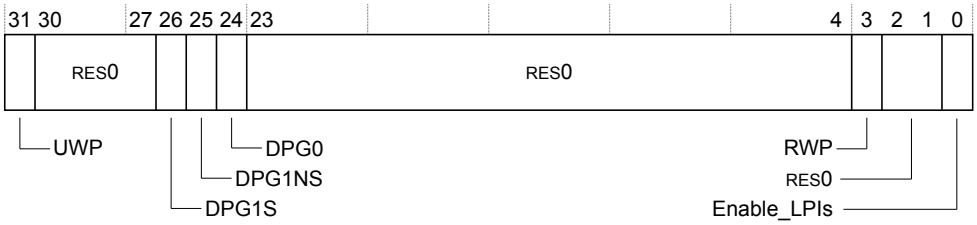

Figure 9-17 GICR_CTLR bit assignments

The following table shows the GICR_CTLR bit assignments.

Table 9-36 GICR_CTLR bit assignments

Bits	Name	Function
[31]	UWP	Read-only. Indicates whether upstream writes are pending: 0 No upstream writes pending.
[30:27]	-	Reserved, RES0.
[26]	DPG1S	Relates to 1 of N distribution which is not supported, RAZ/WI.
[25]	DPG1NS	Relates to 1 of N distribution which is not supported, RAZ/WI.
[24]	DPG0	Relates to 1 of N distribution which is not supported, RAZ/WI.
[23:4]	-	Reserved, RES0.
[3]	RWP	Register Write Pending. This read-only bit indicates whether a GICR_ICENABLERn write is in progress or not: 0 The effect of all previous writes to GICR_ICENABLERn are visible to the GIC CPU interface. 1 The effect of all previous writes to GICR_ICENABLERn are not guaranteed to be visible to the GIC CPU interface because the changes are still being propagated: • GICR_ICENABLERn. The clearing of enable state for PPIs and SGIs.

Table 9-36 GICR_CTLR bit assignments (continued)

Bits	Name	Function
[2:1]	-	Reserved, RES0.
[0]	Enable LPIs	This bit controls whether physical LPIs are enabled: RES0 LPIs are not supported.

GICR_CTLR can be accessed through its memory-mapped interface.

Table 9-37 GICR_CTLR access information

Component	Offset	Reset
GIC Redistributor	0x0000	0x00000000

Redistributor Implementer Identification Register

The GICR_IIDR provides information about the implementer and revision of the Redistributor.

Usage constraints This register is read-only.
This register returns the same value as GICD_IIDR.
Traps and enables There are no traps and enables affecting this register.
Configurations A copy of this register is provided for each Redistributor.
Attributes GICR_IIDR is a 32-bit register.

The following figure shows the GICR_IIDR bit assignments.

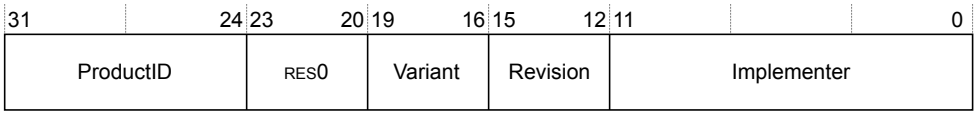

Figure 9-18 GICR_IIDR bit assignments

The following table shows the GICR_IIDR bit assignments.

Table 9-38 GICR_IIDR bit assignments

Bits	Name	Function
[31:24]	ProductID	0x01 Cortex-R52 GIC
[23:20]	-	Reserved, RES0
[19:16]	Variant	0x1
[15:12]	Revision	0x3
[11:0]	Implementer	0x43B Arm Limited

GICR_IIDR can be accessed through its memory-mapped interface.

Table 9-39 GICR_IIDR access information

Component	Offset	Reset
GIC Redistributor	0x0004	0x0101343B

Redistributor Type Register

The GICR_TYPER register provides information about the configuration of the Redistributor.

Usage constraints This register is read-only.

Traps and enables There are no traps and enables affecting this register.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_TYPER is a 64-bit register.

The following figure shows the GICR_TYPER bit assignments.

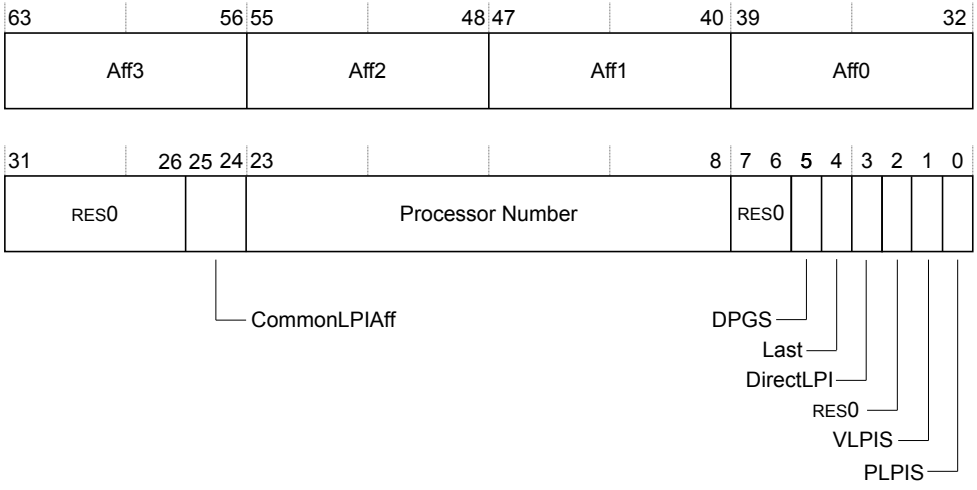

Figure 9-19 GICR_TYPER bit assignments

The following table shows the GICR_TYPER bit assignments.

Table 9-40 GICR_TYPER bit assignments

Bits	Name	Function
[63:56]	Aff3	Affinity level 3, not used. `0x00`
[55:48]	Aff2	Affinity level 2, not used. `0x00`
[47:40]	Aff1	Affinity level 1, not used. `0x00`
[39:32]	Aff0	Affinity level 0. The target ID of the Redistributor. Targets are contiguously numbered `0` to `NUM_TARGET-1`. `NUM_TARGET` is the total number of cores and export ports. For a two core system with an export port, this would be core0=`0`, core1=`1`, export=`2`
[31:26]	-	Reserved, RES0.
[25:24]	CommonLPIAff	Relates to LPIs which are not supported. `0x00`
[23:8]	Processor Number	A unique identifier for the core. The same as Aff0, unused MSBs zero padded.
[7:6]	-	Reserved, RES0.
[5]	DPGS	Sets support for GICR_CTLR.DPG* bits: `0` GICR_CTLR.DPG* bits are not supported.

Table 9-40 GICR_TYPER bit assignments (continued)

Bits	Name	Function
[4]	Last	Indicates whether this Redistributor is the last numbered Redistributor in a series of contiguous Redistributor pages: `0` This Redistributor is not the last Redistributor in a series of contiguous Redistributors. `1` This Redistributor is the last Redistributor in a series of contiguous Redistributors. In a processor configured with an interrupt export port, this bit is set for the Redistributor associated with the export port. Otherwise, in a system with n cores, this bit is set for the Redistributor associated with core n-1.
[3]	DirectLPI	Indicates whether this Redistributor supports direct injection of LPIs: `0` This Redistributor does not support direct injection of LPIs.
[2]	-	Reserved, RES0.
[1]	VLPIS	`0` Virtual LPIs not supported.
[0]	PLPIS	Indicates whether the GIC implementation supports physical LPIs: `0` Physical LPIs are not supported.

GICR_TYPER can be accessed through its memory-mapped interface.

Table 9-41 GICR_TYPER access information

Component	Offset	Reset	
GIC Redistributor	`0x0008-0x000C`	`0x0000000000000000`	For core0 in a two core system.
		`0x0000000100000110`	For core1 in a two core system.

Redistributor Wake Register

The GICR_WAKER register permits software to quiesce the interface between the GIC Distributor and GIC CPU interface before a target enters the processor sleep state. Exit from the processor sleep state is caused by a pending interrupt for that target.

Usage constraints This register is read/write.

Traps and enables There are no traps and enables affecting this register.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_WAKER is a 32-bit register.

The following figure shows the GICR_WAKER bit assignments.

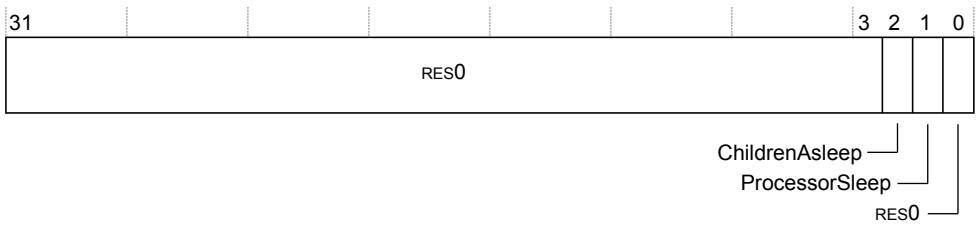

Figure 9-20 GICR_WAKER bit assignments

The following table shows the GICR_WAKER bit assignments.

Table 9-42 GICR_WAKER bit assignments

Bits	Name	Function
[31:3]	-	Reserved, RES0.
[2]	ChildrenAsleep	Indicates whether the connected target is quiescent: `0` All interfaces to the connected target are not quiescent. `1` All interfaces to the connected target are quiescent. This is the reset value.
[1]	ProcessorSleep	Instructs the Redistributor that the target is entering the processor sleep state. On writing `1`, the Redistributor quiesces the interface to the GIC CPU interface. When an interrupt for the target is pending and it is in processor sleep state, a request is made to power up the target: `0` Writing `0` indicates that the target is not in Processor Sleep state. This is done during initialization. `1` Writing `1` indicates that the target is entering Processor Sleep state. This is the reset value.
[0]	-	Reserved, RES0.

GICR_WAKER can be accessed through its memory-mapped interface.

Table 9-43 GICR_WAKER access information

Component	Offset	Reset
GIC Redistributor	`0x0014`	`0x00000006`

Redistributor Identification Registers 0-7

The GICR_PIDR0-7 registers provide the Peripheral identification information.

Usage constraints These registers are read-only.

Traps and enables There are no traps and enables affecting these registers.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_PIDR0-7 are 32-bit registers.

The following figure shows the GICR_PIDR0-7 bit assignments.

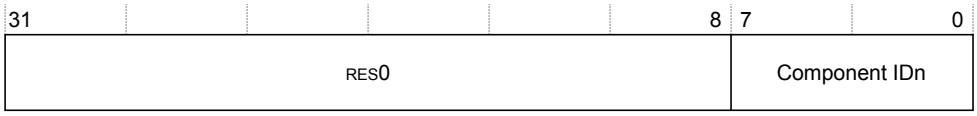

Figure 9-21 GICR_PIDR0-7 bit assignments

The following table shows the GICR_PIDR0-7 bit assignments.

Table 9-44 GICR_PIDR0-7 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	-	Component ID.

GICR_PIDR0-7 can be accessed through its memory-mapped interface.

Table 9-45 GICR_PIDR0-7 reset values

Component	Register	Offset	Reset value
GIC Redistributor	GICR_PIDR0	0xFFE0	0x00000093
	GICR_PIDR1	0xFFE4	0x000000B4
	GICR_PIDR2	0xFFE8	0x0000003B
	GICR_PIDR3	0xFFEC	0x00000000
	GICR_PIDR4	0xFFD0	0x00000044
	GICR_PIDR5	0xFFD4	0x00000000
	GICR_PIDR6	0xFFD8	0x00000000
	GICR_PIDR7	0xFFDC	0x00000000

Redistributor Component Identification Registers 0-3

The GICR_CIDR0-3 registers provide the component identification information.

Usage constraints These registers are read-only.

Traps and enables There are no traps and enables affecting these registers.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_CIDR0-3 are 32-bit registers.

The following figure shows the GICR_CIDR0-3 bit assignments.

Figure 9-22 GICR_CIDR0-3 bit assignments

The following table shows the GICR_CIDR0-3 bit assignments.

Table 9-46 GICR_CIDR0-3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0
[7:0]	-	Component IDn

GICR_CIDR0-3 can be accessed through its memory-mapped interface.

Table 9-47 GICR_CIDR0-3 reset values

Component	Register	Offset	Reset value
GIC Redistributor	GICR_CIDR0	0xFFF0	0x0000000D
	GICR_CIDR1	0xFFF4	0x000000F0
	GICR_CIDR2	0xFFF8	0x00000005
	GICR_CIDR3	0xFFFC	0x000000B1

Interrupt Group Register 0

The GICR_IGROUPR0 register controls whether the corresponding SGI or PPI is in Group 0 or Group 1.

Usage constraints This register is read/write.

Traps and enables There are no traps and enables affecting this register.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_IGROUPR0 is a 32-bit register.

The following figure shows the GICR_IGROUPR0 bit assignments.

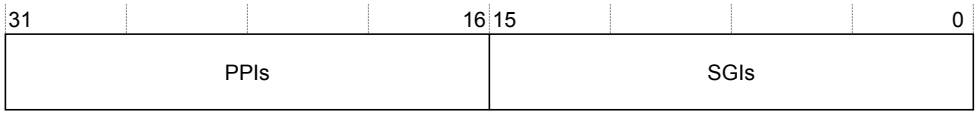

Figure 9-23 GICR_IGROUPR0 bit assignments

The following table shows the GICR_IGROUPR0 bit assignments.

Table 9-48 GICR_IGROUPR0 bit assignments

Bits	Name	Function
[31:16]	PPIs	Controls the group for the corresponding PPIs: `0` The PPI is in Group 0. This is the reset value. `1` The PPI is in Group 1.
[15:0]	SGIs	Controls the group for the corresponding SGIs: `0` The SGI is in Group 0. This is the reset value. `1` The SGI is in Group 1.

GICR_IGROUPR0 can be accessed through its memory-mapped interface.

Table 9-49 GICR_IGROUPR0 access information

Component	Offset	Reset
GIC Redistributor	`0x0080`	`0x00000000`

Interrupt Set-Enable Register 0

The GICR_ISENABLER0 register enables forwarding of the corresponding SGI or PPI from the Distributor to the CPU interfaces.

Usage constraints This register is read/write.

Traps and enables There are no traps and enables affecting this register.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_ISENABLER0 is a 32-bit register.

The following figure shows the GICR_ISENABLER0 bit assignments.

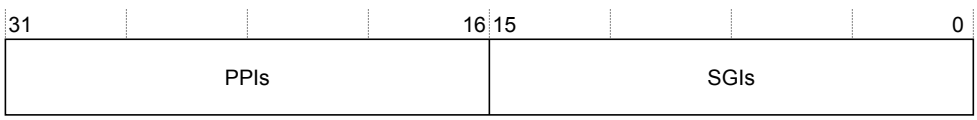

Figure 9-24 GICR_ISENABLER0 bit assignments

The following table shows the GICR_ISENABLER0 bit assignments.

Table 9-50 GICR_ISENABLER0 bit assignments

Bits	Name	Function
[31:16]	PPIs	Enables forwarding of the corresponding PPIs: **Reads** `0` Indicates forwarding of PPI is disabled. This is the reset value. `1` Indicates forwarding of PPI is enabled. **Writes** `0` No effect. `1` Enables forwarding of PPI.
[15:0]	SGIs	Enables forwarding of the corresponding SGIs: **Reads** `0` Indicates forwarding of SGI is disabled. This is the reset value. `1` Indicates forwarding of SGI is enabled. **Writes** `0` No effect. `1` Enables forwarding of SGI.

GICR_ISENABLER0 can be accessed through its memory-mapped interface.

Table 9-51 GICR_ISENABLER0 access information

Component	Offset	Reset
GIC Redistributor	`0x0100`	`0x00000000`

Interrupt Clear-Enable Register 0

The GICR_ICENABLER0 register disables forwarding of the corresponding SGI or PPI to the CPU interfaces.

Usage constraints This register is read/write.

Traps and enables There are no traps and enables affecting this register.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_ICENABLER0 is a 32-bit register.

The following figure shows the GICR_ICENABLER0 bit assignments.

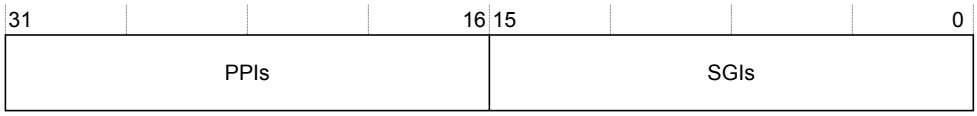

Figure 9-25 GICR_ICENABLER0 bit assignments

The following table shows the GICR_ICENABLER0 bit assignments.

Table 9-52 GICR_ICENABLER0 bit assignments

Bits	Name	Function
[31:16]	PPIs	Disables forwarding of the corresponding PPIs: **Reads** `0` Indicates forwarding of PPI is disabled. This is the reset value. `1` Indicates forwarding of PPI is enabled. **Writes** `0` No effect. `1` Disables forwarding of PPI.
[15:0]	SGIs	Disables forwarding of the corresponding SGIs: **Reads** `0` Indicates forwarding of SGI is disabled. This is the reset value. `1` Indicates forwarding of SGI is enabled. **Writes** `0` No effect. `1` Disables forwarding of SGI.

GICR_ICENABLER0 can be accessed through its memory-mapped interface.

Table 9-53 GICR_ICENABLER0 access information

Component	Offset	Reset
GIC Redistributor	`0x0180`	`0x00000000`

Interrupt Set-Pending Register 0

The GICR_ISPENDR0 register sets the pending bit for SGIs and PPIs.

Usage constraints This register is read/write.

Traps and enables There are no traps and enables affecting this register.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_ISPENDR0 is a 32-bit register.

The following figure shows the GICR_ISPENDR0 bit assignments.

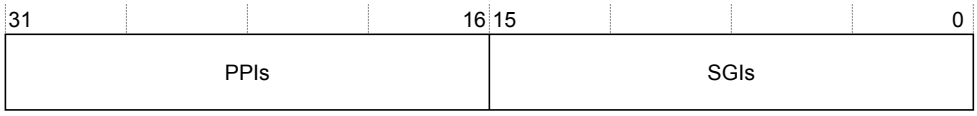

Figure 9-26 GICR_ISPENDR0 bit assignments

The following table shows the GICR_ISPENDR0 bit assignments.

Table 9-54 GICR_ISPENDR0 bit assignments

Bits	Name	Function
[31:16]	PPIs	Sets the PPI pending bits: **Reads** `0` Indicates that the PPI is not pending. This is the reset value. `1` Indicates that the PPI is pending. **Writes** `0` No effect. `1` Sets the PPI pending bit.
[15:0]	SGIs	Sets the SGI pending bits: **Reads** `0` Indicates that the SGI is not pending. This is the reset value. `1` Indicates that the SGI is pending. **Writes** `0` No effect. `1` Sets the SGI pending bit.

GICR_ISPENDR0 can be accessed through its memory-mapped interface.

Table 9-55 GICR_ISPENDR0 access information

Component	Offset	Reset
GIC Redistributor	`0x0200`	`0x00000000`

Interrupt Clear-Pending Register 0

The GICR_ICPENDR0 register clears the pending SGI or PPI bit.

Usage constraints This register is read/write.

Traps and enables There are no traps and enables affecting this register.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_ICPENDR0 is a 32-bit register.

The following figure shows the GICR_ICPENDR0 bit assignments.

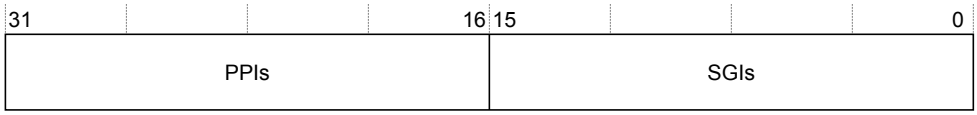

Figure 9-27 GICR_ICPENDR0 bit assignments

The following table shows the GICR_ICPENDR0 bit assignments.

Table 9-56 GICR_ICPENDR0 bit assignments

Bits	Name	Function
[31:16]	PPIs	Clears the pending bit for the corresponding PPI: **Reads** `0` Indicates that the PPI is pending. This is the reset value. `1` Indicates that the PPI is not pending. **Writes** `0` No effect. `1` Sets the PPI pending bit.
[15:0]	SGIs	Clears the pending bit for the corresponding SGI: **Reads** `0` Indicates that the SGI is pending. This is the reset value. `1` Indicates that the SGI is not pending. **Writes** `0` No effect. `1` Sets the SGI pending bit.

GICR_ICPENDR0 can be accessed through its memory-mapped interface.

Table 9-57 GICR_ICPENDR0 access information

Component	Offset	Reset
GIC Redistributor	`0x0280`	`0x00000000`

Interrupt Set-Active Register 0

The GICR_ISACTIVER0 register sets the active bit for SGIs and PPIs.

Usage constraints This register is read/write.

Traps and enables There are no traps and enables affecting this register.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_ISACTIVER0 is a 32-bit register.

The following figure shows the GICR_ISACTIVER0 bit assignments.

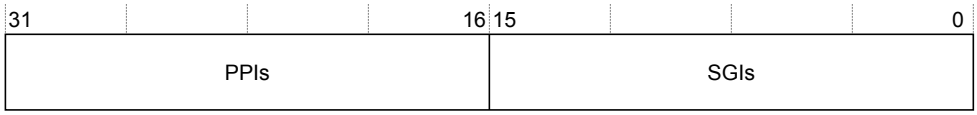

Figure 9-28 GICR_ISACTIVER0 bit assignments

The following table shows the GICR_ISACTIVER0 bit assignments.

Table 9-58 GICR_ISACTIVER0 bit assignments

Bits	Name	Function
[31:16]	PPIs	Sets the active bit for the corresponding PPI: **Reads** `0` Indicates that the PPI is not active. This is the reset value. `1` Indicates that the PPI is active. **Writes** `0` No effect. `1` Sets the PPI active bit.
[15:0]	SGIs	Sets the active bit for the corresponding SGI: **Reads** `0` Indicates that the SGI not active. This is the reset value. `1` Indicates that the SGI is active. **Writes** `0` No effect. `1` Sets the SGI active bit.

GICR_ISACTIVER0 can be accessed through its memory-mapped interface.

Table 9-59 GICR_ISACTIVER0 access information

Component	Offset	Reset
GIC Redistributor	`0x0300`	`0x00000000`

Interrupt Clear-Active Register 0

The GICR_ICACTIVER0 register clears the active bit for the corresponding SGI or PPI.

Usage constraints This register is read/write.

Traps and enables There are no traps and enables affecting this register.

Configurations A copy of this register is provided for each Redistributor.

Attributes GICR_ICACTIVER0 is a 32-bit register.

The following figure shows the GICR_ICACTIVER0 bit assignments.

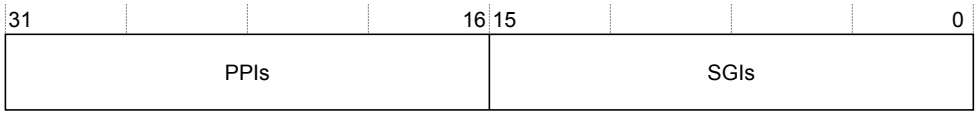

Figure 9-29 GICR_ICACTIVER0 bit assignments

The following table shows the GICR_ICACTIVER0 bit assignments.

Table 9-60 GICR_ICACTIVER0 bit assignments

Bits	Name	Function
[31:16]	PPIs	Clears the active bit for the corresponding PPI: **Reads** `0` Indicates that the PPI is not active. This is the reset value. `1` Indicates that the PPI is active. **Writes** `0` No effect. `1` Clears the PPI active bit.
[15:0]	SGIs	Clears the active bit for the corresponding SGI: **Reads** `0` Indicates that the SGI is not active. This is the reset value. `1` Indicates that the SGI is active. **Writes** `0` No effect. `1` Clears the SGI active bit.

GICR_ICACTIVER0 can be accessed through its memory-mapped interface.

Table 9-61 GICR_ICACTIVER0 access information

Component	Offset	Reset
GIC Redistributor	`0x0380`	`0x00000000`

Interrupt Priority Registers 0-7

The GICR_IPRIORITYR0-7 registers provide a 5-bit priority field for each SGI and PPI. This field stores the priority of the corresponding interrupt.

Usage constraints These registers are read/write.

These registers are byte-accessible.

Traps and enables There are no traps and enables affecting these registers.

Configurations A copy of these registers is provided for each Redistributor.

Attributes GICR_IPRIORITYR0-7 are 32-bit registers.

Each register holds four priority fields as shown in the following figure.

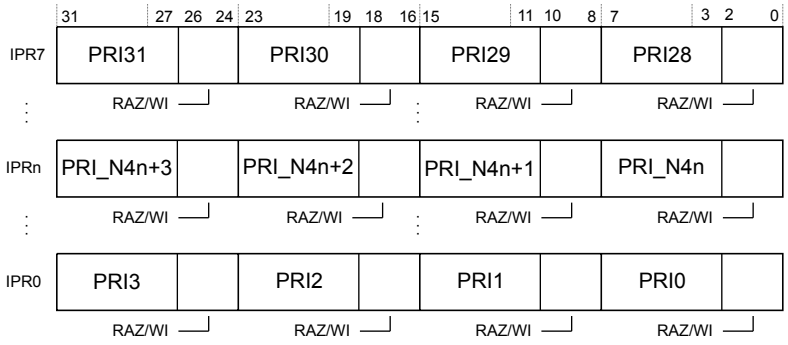

Figure 9-30 GICR_IPRIORITYR0-7 priority fields

Note

PRIn corresponds to INTIDn.

The following table shows the GICR_IPRIORITYR0-7 bit assignments.

Table 9-62 GICR_IPRIORITYR0-7 bit assignments

Bits	Name[bm]	Function
[31:27]	Priority, byte offset 3	Each priority field holds a priority value, 0-31. The lower the value, the higher the priority of the corresponding interrupt.
[26:24]	-	Reserved, RES0.
[23:19]	Priority, byte offset 2	Each priority field holds a priority value, 0-31. The lower the value, the higher the priority of the corresponding interrupt.
[18:16]	-	Reserved, RES0.
[15:11]	Priority, byte offset 1	Each priority field holds a priority value, 0-31. The lower the value, the higher the priority of the corresponding interrupt.
[10:8]	-	Reserved, RES0.
[7:3]	Priority, byte offset 0	Each priority field holds a priority value, 0-31. The lower the value, the higher the priority of the corresponding interrupt.
[2:0]	-	Reserved, RES0.

For INTID*m*, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICR_IPRIORITYRn number, *n*, is given by *n* = *m* DIV 4, where *m*=0 to 31.
- The offset of the required GICR_IPRIORITYRn is (`0x400` + (4×*n*)).
- The byte offset of the required Priority field in this register is *m* MOD 4, where:
  - Byte offset 0 refers to register bits[7:3].
  - Byte offset 1 refers to register bits[15:11].
  - Byte offset 2 refers to register bits[23:19].
  - Byte offset 3 refers to register bits[31:27].

GICR_IPRIORITYR0-7 can be accessed through its memory-mapped interface.

[bm] Each field holds the priority value for a single interrupt. This section describes how the INTID value determines the GICR_IPRIORITYR0-7 register number and the byte offset of the priority field in that register.

Table 9-63 GICR_IPRIORITYR0-7 access information

Component	Offset	Reset
GIC Redistributor	0x0400-0x041C	0x00000000

Interrupt Configuration Register 0

The GICR_ICFGR0 register provides a 2-bit Int_config field for each SGI supported by the GIC Distributor. All SGIs behave as edge-triggered interrupts and therefore this register is read only.

Note

Fields 0-15 correspond to SGIs 0-15.

Usage constraints	This register is read only.
Traps and enables	There are no traps and enables affecting this register.
Configurations	A copy of this register is provided for each Redistributor.
Attributes	GICR_ICFGR0 is a 32-bit register.

The following figure shows the GICR_ICFGR0 bit assignments.

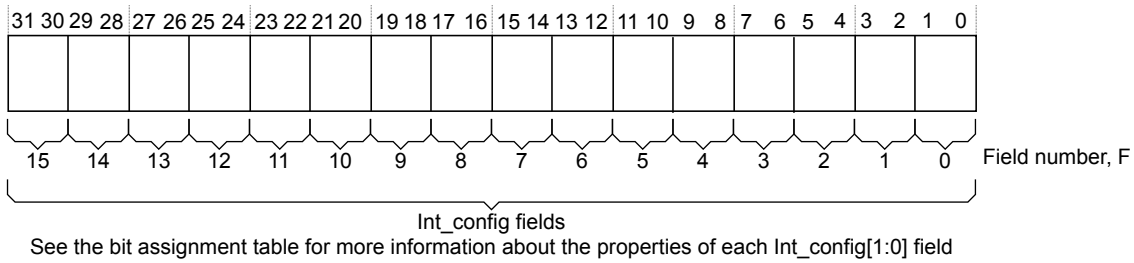

Figure 9-31 GICR_ICFGR0 bit assignments

The following table shows the GICR_ICFGR0 bit assignments.

Table 9-64 GICR_ICFGR0 bit assignments

Bits	Name	Function
[31:0]	Int_config	Read only bits. Reset: 0xAAAAAAAA

GICR_ICFGR0 can be accessed through its memory-mapped interface.

Table 9-65 GICR_ICFGR0 access information

Component	Offset	Reset
GIC Redistributor	0x0C00	0xAAAAAAAA

Interrupt Configuration Register 1

The GICR_ICFGR1 register provides a 2-bit Int_config field for each PPI supported by the GIC Distributor. Some PPIs are level-sensitive and their corresponding fields are read-only and they are read as 0b00.

Note

Fields 0-15 correspond to PPIs 16-31. Fields 0-5, 12, 13, and 15 are programmable, 6-11 and 14 are read-only because they are assigned to core peripherals that have a fixed level-sensitive configuration. For more information, see *9.2.2 Interrupt sources* on page 9-264.

Usage constraints	This register is read only.
Traps and enables	There are no traps and enables affecting this register.
Configurations	A copy of this register is provided for each Redistributor.
Attributes	GICR_ICFGR1 is a 32-bit register.

The following figure shows the GICR_ICFGR1 bit assignments.

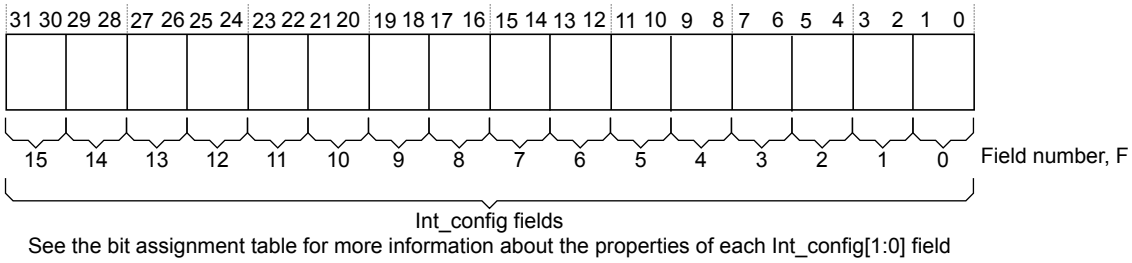

Figure 9-32 GICR_ICFGR1 bit assignments

The following table shows the GICR_ICFGR1 bit assignments.

Table 9-66 GICR_ICFGR1 bit assignments

Bits	Name	Function
[2*F*+1:2*F*]	Int_config, field *F*	For Int_config[1], the most significant bit, bit [2F+1], the encoding is: `0` Corresponding interrupt is Active-LOW level-sensitive. This is the reset value. `1` Corresponding interrupt is rising edge-triggered. Int_config[0], the least significant bit, bit [2F], is RES0. The reset value is `0b00`.

GICR_ICFGR1 can be accessed through its memory-mapped interface.

Table 9-67 GICR_ICFGR1 access information

Component	Offset	Reset
GIC Redistributor	`0x0C04`	`0x00000000`

9.3.3 Hypervisor Control System Registers

This section describes the implemented Hypervisor Control System registers within each core which are allocated to the (coproc==`0b1111`) space. The following table summarizes the Hypervisor Control System registers.

Table 9-68 Hypervisor Control System Registers summary

Name	CRn	Op1	CRm	Op2	Type	Reset	Description
ICH_HCR	c12	4	c11	0	RW	0x00000000	*Interrupt Controller Hyp Control Register* on page 9-300
ICH_VTR				1	RO	0x90180003	*Interrupt Controller Hyp VGIC Type Register* on page 9-303
ICH_VMCR				7	RW	0x004C0008	*Interrupt Controller Virtual Machine Control Register* on page 9-308
ICH_MISR				2	RO	0x00000000	*Interrupt Controller Hyp Maintenance Interrupt Status Register EL2* on page 9-304
ICH_EISR				3	RO	0x00000000	*Interrupt Controller End of Interrupt Status Register* on page 9-306
ICH_ELRSR				5	RO	0x0000000F	*Interrupt Controller Empty List Register Status Register* on page 9-307
ICH_AP0R0	c12	4	c8	0	RW	0x00000000	*Interrupt Controller Hyp Active Priorities Group 0 Register 0* on page 9-312
ICH_AP1R0	c12	4	c9	0	RW	0x00000000	*Interrupt Controller Hyp Active Priorities Group 1 Register 0* on page 9-312
ICH_LR0	c12	4	c12	0	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LR1				1	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LR2				2	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LR3				3	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-309
ICH_LRC0	c12	4	c14	0	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-310
ICH_LRC1				1	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-310
ICH_LRC2				2	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-310
ICH_LRC3				3	RW	0x00000000	*Interrupt Controller List Registers 0-3* on page 9-310

Note

Writing to ICH_AP0R0 and ICH_AP0R1 with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in UNPREDICTABLE behavior of the interrupt prioritization system.

Interrupt Controller Hyp Control Register

The ICH_HCR register controls the environment for guest operating systems.

Usage constraints This register is read/write, and is only accessible at EL2.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations This register is available in all build configurations.

Attributes ICH_HCR is a 32-bit register.

The following figure shows the ICH_HCR bit assignments.

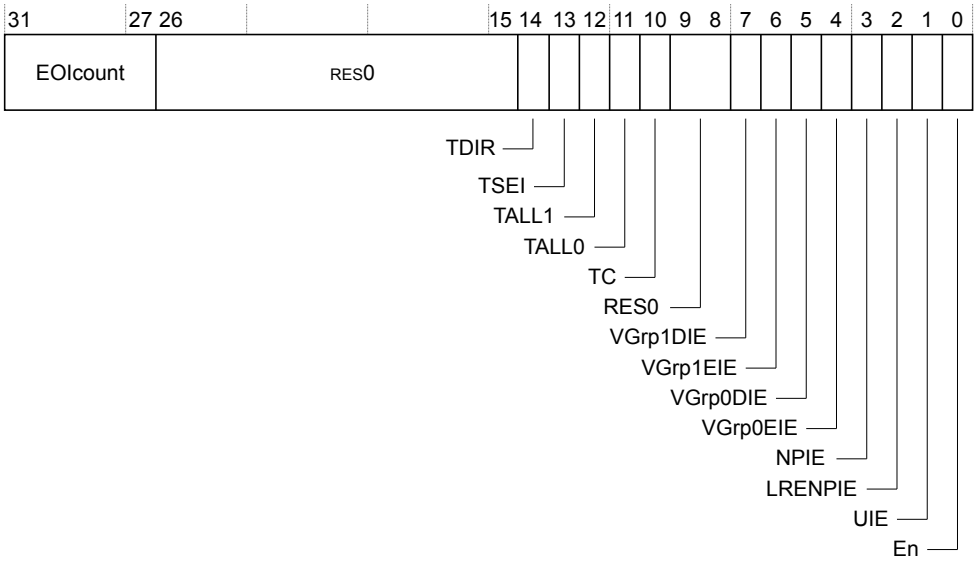

Figure 9-33 ICH_HCR bit assignments

The following table shows the ICH_HCR bit assignments.

Table 9-69 ICH_HCR bit assignments

Bits	Name	Function
[31:27]	EOIcount	This field increments when a successful write to a virtual EOIR or DIR register results in a virtual interrupt deactivation. Although not possible under correct operation, if an *End of Interrupt* (EOI) occurs when the value of this field is 31, this field wraps to 0. The maintenance interrupt, INTID25, is asserted whenever this field is non-zero and the LRENPIE bit is set to 1.
[26:15]	-	Reserved, RES0.
[14]	TDIR	Trap virtual EL1 writes to ICC_DIR: `0` Virtual EL1 writes of ICC_DIR are not trapped to EL2, unless trapped by other mechanisms. `1` Virtual EL1 writes of ICC_DIR are trapped to EL2.
[13]	TSEI	This field is RES0.
[12]	TALL1	Trap all virtual EL1 accesses to ICC_* System registers for Group 1 interrupts to EL2. `0` Virtual EL1 accesses to ICC_* registers for Group 1 interrupts proceed as normal. This is the reset value. `1` Any virtual EL1 accesses to ICC_* registers for Group 1 interrupts trap to EL2. This affects accesses to ICC_IAR1, ICC_EOIR1, ICC_HPPIR1, ICC_BPR1, ICC_AP1R0, and ICC_IGRPEN1.

Table 9-69 ICH_HCR bit assignments (continued)

Bits	Name	Function
[11]	TALL0	Trap all virtual EL1 accesses to ICC_* System registers for Group 0 interrupts to EL2. 0 Virtual EL1 accesses to ICC_* registers for Group 0 interrupts proceed as normal. This is the reset value. 1 Any virtual EL1 accesses to ICC_* registers for Group 0 interrupts trap to EL2. This affects accesses to ICC_IAR0, ICC_EOIR0, ICC_HPPIR0, ICC_BPR0, ICC_AP0R0, and ICC_IGRPEN0.
[10]	TC	Trap all virtual EL1 accesses to System registers that are common to Group 0 and Group 1 to EL2. 0 Virtual EL1 accesses to common registers proceed as normal. This is the reset value. 1 Any virtual EL1 accesses to common registers trap to EL2. This affects accesses to ICC_SGI0R, ICC_SGI1R, ICC_ASGI1R, ICC_CTLR, ICC_DIR, ICC_PMR, and ICC_RPR.
[9:8]	-	Reserved, RES0.
[7]	VGrp1DIE	VM Group 1 Disabled Interrupt Enable: 0 Maintenance interrupt disabled. This is the reset value. 1 Maintenance interrupt enabled when virtual ICC_IGRPEN1.Enable is 0.
[6]	VGrp1EIE	VM Group 1 Enabled Interrupt Enable: 0 Maintenance interrupt disabled. This is the reset value. 1 Maintenance interrupt enabled when virtual ICC_IGRPEN1.Enable is 1.
[5]	VGrp0DIE	VM Group 0 Disabled Interrupt Enable: 0 Maintenance interrupt disabled. This is the reset value. 1 Maintenance interrupt enabled when virtual ICC_IGRPEN0.Enable is 0.
[4]	VGrp0EIE	VM Group 0 Enabled Interrupt Enable: 0 Maintenance interrupt disabled. This is the reset value. 1 Maintenance interrupt enabled when virtual ICC_IGRPEN0.Enable is 1.
[3]	NPIE	No Pending Interrupt Enable. Enables the signaling of a maintenance interrupt while no pending interrupts are present in the List registers: 0 Maintenance interrupt disabled. This is the reset value. 1 Maintenance interrupt enabled while the List registers contain no interrupts in the pending state.
[2]	LRENPIE	List Register Entry Not Present Interrupt Enable. Enables the signaling of a maintenance interrupt while the virtual CPU interface does not have a corresponding valid List register entry for an EOI request: 0 Maintenance interrupt disabled. This is the reset value. 1 Maintenance interrupt is enabled while the EOIcount field is not 0.

Table 9-69 ICH_HCR bit assignments (continued)

Bits	Name	Function
[1]	UIE	Underflow Interrupt Enable. Enables the signaling of a maintenance interrupt when the List registers are empty, or hold only one valid entry: `0` Maintenance interrupt disabled. This is the reset value. `1` Maintenance interrupt is enabled if none, or only one, of the List register entries is marked as a valid interrupt.
[0]	En	Enable. Global enable bit for the virtual CPU interface: `0` Virtual CPU interface operation disabled. This is the reset value. `1` Virtual CPU interface operation enabled. When this field is set to 0: • The virtual CPU interface does not signal any maintenance interrupts. • The virtual CPU interface does not signal any virtual interrupts. • A virtual access to an interrupt acknowledge register returns a spurious interrupt ID.

To access ICH_HCR:

```
MRC p15,4,<Rt>,c12,c11,0 ; Read ICH_HCR into Rt
MCR p15,4,<Rt>,c12,c11,0 ; Write Rt to ICH_HCR
```

Interrupt Controller Hyp Control VGIC Type Register

The ICH_VTR register describes the number of implemented virtual priority bits and List registers.

Usage constraints This register is read-only, and is only available from EL2.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations This register is available in all build configurations.

Attributes ICH_VTR is a 32-bit register.

The following figure shows the ICH_VTR bit assignments.

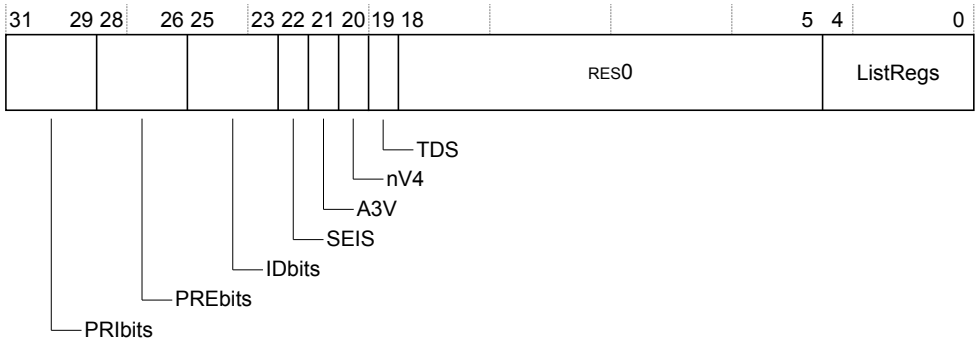

Figure 9-34 ICH_VTR bit assignments

The following table shows the ICH_VTR bit assignments.

Table 9-70 ICH_VTR bit assignments

Bits	Name	Function
[31:29]	PRIbits	The number of virtual priority bits implemented, minus one. `0x4`. 5 priority bits.
[28:26]	PREbits	The number of virtual preemption bits implemented, minus one. All bits can form the priority group field and be used for preemption. `0x4`. 5 preemption bits.
[25:23]	IDbits	The number of virtual interrupt identifier bits supported: `0b000` 16 bits.
[22]	SEIS	SEI Support. Indicates whether the virtual CPU interface supports generation of SEIs: `0` The virtual CPU interface logic does not support generation of SEIs.
[21]	A3V	Affinity 3 Valid: `0` The virtual CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
[20]	nV4	GICv4 direct injection of virtual interrupts: `1` The CPU interface logic does not support direct injection of virtual interrupts.
[19]	TDS	Separate trapping EL1 writes to ICC_DIR supported: `1` ICH_HCR.TDIR supported.
[18:5]	-	Reserved, RES0.
[4:0]	ListRegs	The number of implemented List registers, minus one: `0x03` Four List registers.

To access ICH_VTR:

```
MRC p15,4,<Rt>,c12,c11,1 ; Read ICH_VTR into Rt
MCR p15,4,<Rt>,c12,c11,1 ; Write Rt to ICH_VTR
```

Interrupt Controller Hyp Maintenance Interrupt Status Register EL2

The ICH_MISR register indicates which maintenance interrupts are asserted.

Usage constraints

This register is read only, and is only available at EL2.

Traps and enables

If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations

This register is available in all build configurations.

Attributes

ICH_MISR is a 32-bit register.

The following figure shows the ICH_MISR bit assignments.

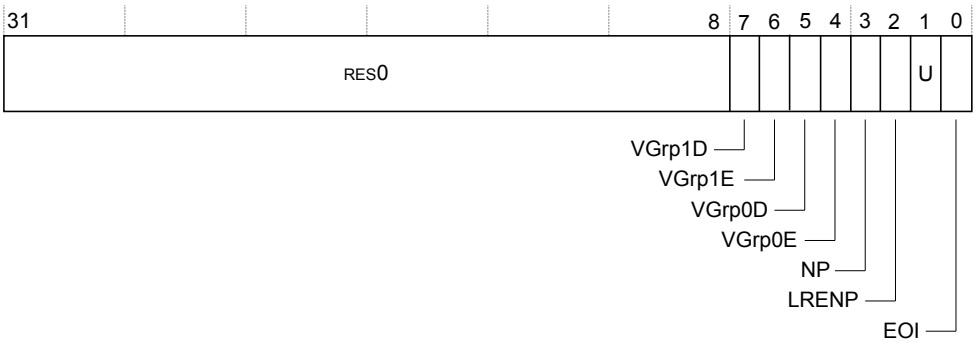

Figure 9-35 ICH_MISR bit assignments

The following table shows the ICH_MISR bit assignments.

Table 9-71 ICH_MISR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7]	VGrp1D	vPE Group 1 Disabled: 0 vPE Group 1 Disabled maintenance interrupt not asserted. 1 vPE Group 1 Disabled maintenance interrupt asserted. This maintenance interrupt is asserted when ICH_HCR.VENG1 is 1 and ICH_VMCR.VMGrp1En is 0. This bit resets to 0.
[6]	VGrp1E	vPE Group 1 Enabled: 0 vPE Group 1 Enabled maintenance interrupt not asserted. 1 vPE Group 1 Enabled maintenance interrupt asserted. This maintenance interrupt is asserted when ICH_HCR.VENG1 is 1 and ICH_VMCR.VMGrp1En is 1. This bit resets to 0.
[5]	VGrp0D	vPE Group 0 Disabled: 0 vPE Group 0 Disabled maintenance interrupt not asserted. 1 vPE Group 0 Disabled maintenance interrupt asserted. This maintenance interrupt is asserted when ICH_HCR.VENG0 is 1 and ICH_VMCR.VMGrp0En is 0. This bit resets to 0.
[4]	VGrp0E	vPE Group 0 Enabled: 0 vPE Group 0 Enabled maintenance interrupt not asserted. 1 vPE Group 0 Enabled maintenance interrupt asserted. This maintenance interrupt is asserted when ICH_HCR.VENG0 is 1 and ICH_VMCR.VMGrp0En is 1. This bit resets to 0.

Table 9-71 ICH_MISR bit assignments (continued)

Bits	Name	Function
[3]	NP	NoPending: `0` NoPending maintenance interrupt not asserted. `1` NoPending interrupt asserted. This maintenance interrupt is asserted when ICH_HCR.NPIE is 1 and no List register is in pending state. This bit resets to 0.
[2]	LRENP	List Register Entry Not Present: `0` List Register Entry Not Present maintenance interrupt not asserted. `1` List Register Entry Not Present maintenance interrupt asserted. This maintenance interrupt is asserted when ICH_HCR.LRENPIE is 1 and ICH_HCR.EOIcount is non-zero. This bit resets to 0.
[1]	U	Underflow: `0` Underflow maintenance interrupt not asserted. `1` Underflow maintenance interrupt asserted. This maintenance interrupt is asserted when ICH_HCR.UIE is 1 and zero or one of the List register entries are marked as a valid interrupt, that is, if the corresponding ICH_LR<n>.State bits do not equal `0x0`. This bit resets to 0.
[0]	EOI	End Of Interrupt. `0` End Of Interrupt maintenance interrupt not asserted. `1` End Of Interrupt maintenance interrupt asserted. This maintenance interrupt is asserted when at least one bit in ICH_EISR is 1. This bit resets to 0.

To access ICH_MISR:

```
MRC p15,4,<Rt>,c12,c11,2 ; Read ICH_MISR into Rt
MCR p15,4,<Rt>,c12,c11,2 ; Write Rt to ICH_MISR
```

Interrupt Controller End of Interrupt Status Register

The ICH_EISR register indicates which List registers have outstanding EOI maintenance interrupts.

Usage constraints

This register is read only, and is only available at EL2.

Traps and enables

If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations

This register is available in all build configurations.

Attributes

ICH_EISR is a 32-bit register.

The following figure shows the ICH_EISR bit assignments.

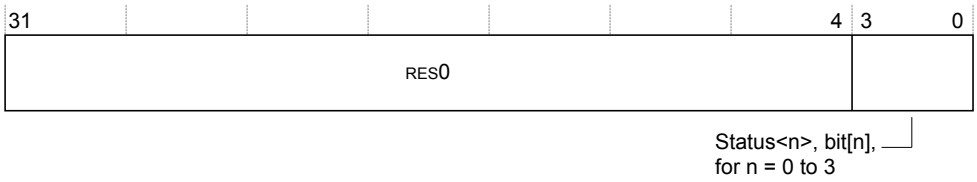

Figure 9-36 ICH_EISR bit assignments

The following table shows the ICH_EISR bit assignments.

Table 9-72 ICH_EISR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	Status<n>	Status<n>, bit[n], for n = 0 to 3. EOI maintenance interrupt status bit for List register <n>: `0` List register <n>, ICH_LR<n>, does not have an EOI maintenance interrupt. `1` List register <n>, ICH_LR<n>, has an EOI maintenance interrupt that has not been handled. For any ICH_LR<n>, the corresponding status bit is set to 1 if all of the following are true: • ICH_LRC<n>. State is `0b00`. • ICH_LRC<n>. HW is 0. • ICH_LRC<n>. EOI (bit [9]) is 1, indicating that when the interrupt corresponding to that List register is deactivated, a maintenance interrupt is asserted. These bits reset to 0.

To access ICH_EISR:

```
MRC p15,4,<Rt>,c12,c11,3 ; Read ICH_EISR into Rt
MCR p15,4,<Rt>,c12,c11,3 ; Write Rt to ICH_EISR
```

Interrupt Controller Empty List Register Status Register

The ICH_ELRSR register indicates which List registers contain valid interrupts.

Usage constraints This register is read only, and is only available at EL2.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations This register is available in all build configurations.

Attributes ICH_ELRSR is a 32-bit register.

The following figure shows the ICH_ELRSR bit assignments.

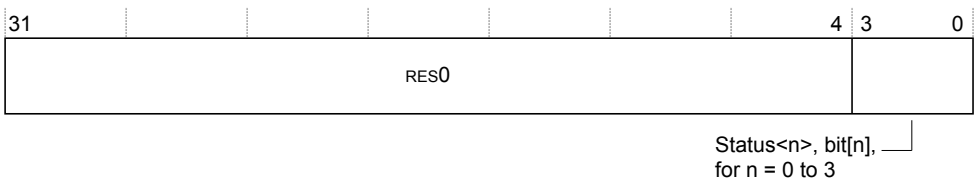

Figure 9-37 ICH_ELRSR bit assignments

The following table shows the ICH_ELRSR bit assignments.

Table 9-73 ICH_ELRSR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3:0]	Status<n>	Status<n>, bit[n], for n = 0 to 3. Status bit for List register <n>, ICH_LR<n>: `0` List register ICH_LR<n>,contains a valid interrupt. `1` List register ICH_LR<n>, does not contain a valid interrupt. The List register is empty and can be used without overwriting a valid interrupt or losing an EOI maintenance interrupt. For any List register<n>, the corresponding status bit is set to 1 if ICH_LRC<n>.State is `0b00` and either ICH_LRC<n>.HW is 1 or ICH_LRC<n>.EOI (bit [9]) is 0.

To access ICH_ELRSR:

```
MRC p15,4,<Rt>,c12,c11,5 ; Read ICH_ELRSR into Rt
MCR p15,4,<Rt>,c12,c11,5 ; Write Rt to ICH_ELRSR
```

Interrupt Controller Virtual Machine Control Register

The ICH_VMCR register enables the hypervisor to save and restore the virtual machine view of the GIC state.

Usage constraints This register is read/write, and is only available at EL2.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations This register is available in all build configurations.

Attributes ICH_VMCR is a 32-bit register.

The following figure shows the ICH_VMCR bit assignments.

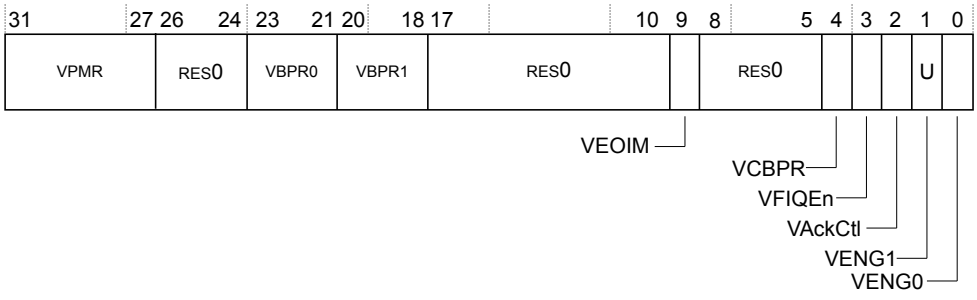

Figure 9-38 ICH_VMCR bit assignments

The following table shows the ICH_VMCR bit assignments.

Table 9-74 ICH_VMCR bit assignments

Bits	Name	Function
[31:27]	VPMR	Virtual Priority Mask. This field is an alias of ICV_PMR.Priority.
[26:24]	-	Reserved, RES0.
[23:21]	VBPR0	Virtual Binary Point Register, Group 0. This field is an alias of ICV_BPR0.BinaryPoint.
[20:18]	VBPR1	Virtual Binary Point Register, Group 1. This field is an alias of ICV_BPR1.BinaryPoint.
[17:10]	-	Reserved, RES0.
[9]	VEOIM	Virtual EOI mode. This bit is an alias of ICV_CTLR.EOImode.
[8:5]	-	Reserved, RES0.
[4]	VCBPR	Virtual Common Binary Point Register. This bit is an alias of ICV_CTLR.CBPR.
[3]	VFIQEn	This field is RES1.
[2]	VAckCtl	This field is RES0.
[1]	VENG1	Virtual Group 1 interrupt enable. This bit is an alias of ICV_IGRPEN1.Enable.
[0]	VENG0	Virtual Group 0 interrupt enable. This bit is an alias of ICV_IGRPEN0.Enable.

To access ICH_VMCR:

```
MRC p15,4,<Rt>,c12,c11,7 ; Read ICH_VMCR into Rt
MCR p15,4,<Rt>,c12,c11,7 ; Write Rt to ICH_VMCR
```

Interrupt Controller List Registers 0-3

The ICH_LR0-3 registers provide interrupt context information for the virtual CPU interface.

Usage constraints This register is read/write, and is only available at EL2.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations This register is available in all build configurations.

Attributes ICH_LR0-3 are 32-bit registers.

The following figure shows the ICH_LR0-3 bit assignments.

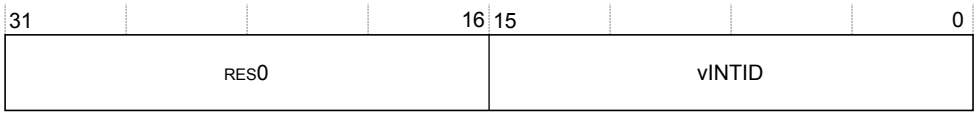

Figure 9-39 ICH_LR0-3 bit assignments

The following table shows the ICH_LR0-3 bit assignments.

Table 9-75 ICH_LR0-3 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0
[15:0]	vINTID	Virtual INTID of the interrupt.

To access ICH_LR0:

```
MRC p15,4,<Rt>,c12,c12,0 ; Read ICH_LR0 into Rt
MCR p15,4,<Rt>,c12,c12,0 ; Write Rt to ICH_LR0
```

To access ICH_LR1:

```
MRC p15,4,<Rt>,c12,c12,1 ; Read ICH_LR1 into Rt
MCR p15,4,<Rt>,c12,c12,1 ; Write Rt to ICH_LR1
```

To access ICH_LR2:

```
MRC p15,4,<Rt>,c12,c12,2 ; Read ICH_LR2 into Rt
MCR p15,4,<Rt>,c12,c12,2 ; Write Rt to ICH_LR2
```

To access ICH_LR3:

```
MRC p15,4,<Rt>,c12,c12,3 ; Read ICH_LR3 into Rt
MCR p15,4,<Rt>,c12,c12,3 ; Write Rt to ICH_LR3
```

Interrupt Controller List Registers 0-3

The ICH_LRC0-3 registers provide interrupt context information for the virtual CPU interface.

Usage constraints This register is read/write, and is only available at EL2.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations This register is available in all build configurations.

Attributes ICH_LRC0-3 are 32-bit registers.

If HW is 0, the following figure shows the ICH_LRC0-3 bit assignments.

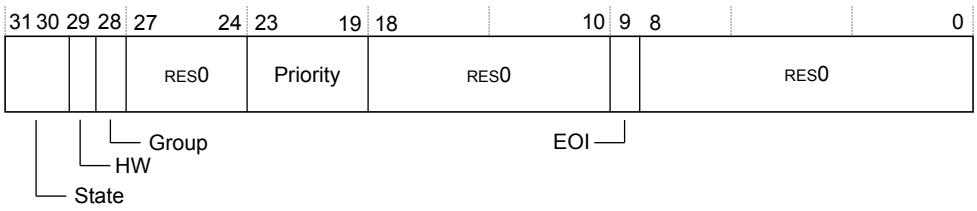

Figure 9-40 ICH_LRC0-3 bit assignments

If HW is 1, the following figure shows the ICH_LRC0-3 bit assignments.

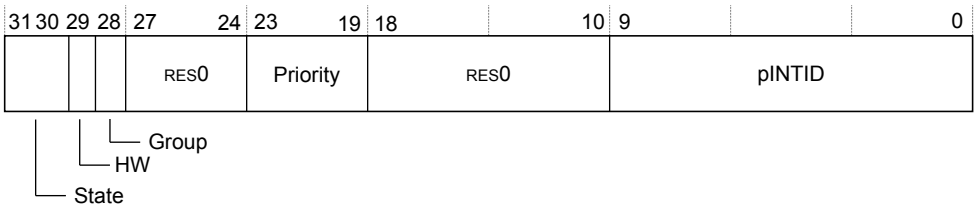

Figure 9-41 ICH_LRC0-3 bit assignments

The following table shows the ICH_LRC0-3 bit assignments.

Table 9-76 ICH_LRC0-3 bit assignments

Bits	Name	Function
[31:30]	State	The state of the interrupt: `0b00` Inactive `0b01` Pending `0b10` Active `0b11` Pending and active.
[29]	HW	Indicates whether this virtual interrupt maps directly to a hardware interrupt, meaning that it corresponds to a physical interrupt. Deactivation of the virtual interrupt also causes the deactivation of the physical interrupt with the INTID that the pINTID field indicates. `0` The interrupt is triggered entirely by software. `1` The interrupt maps directly to a hardware interrupt.
[28]	Group	Indicates the group for this virtual interrupt. `0` This is a Group 0 virtual interrupt. `1` This is a Group 1 virtual interrupt.
[27:24]	-	Reserved, RES0.
[23:19]	Priority	The priority of this interrupt.
[18:10]	-	Reserved, RES0.
[9:0]	If HW is `0`, then: **[9]** EOI **[8:0]** -	If HW is `0`, then: **[9]** End of Interrupt. If this bit is 1, then when the interrupt identified by the vINTID is deactivated, an EOI maintenance interrupt is asserted. **[8:0]** Reserved, RES0
	If HW is `1`, then: **[9:0]** pINTID	If HW is `1`, then: **[9:0]** Physical INTID for hardware interrupts.

To access ICH_LRC0:

```
MRC p15,4,<Rt>,c12,c14,0 ; Read ICH_LRC0 into Rt
MCR p15,4,<Rt>,c12,c14,0 ; Write Rt to ICH_LRC0
```

To access ICH_LRC1:

```
MRC p15,4,<Rt>,c12,c14,1 ; Read ICH_LRC1 into Rt
MCR p15,4,<Rt>,c12,c14,1 ; Write Rt to ICH_LRC1
```

To access ICH_LRC2:

```
MRC p15,4,<Rt>,c12,c14,2 ; Read ICH_LRC2 into Rt
MCR p15,4,<Rt>,c12,c14,2 ; Write Rt to ICH_LRC2
```

To access ICH_LRC3:

```
MRC p15,4,<Rt>,c12,c14,3 ; Read ICH_LRC3 into Rt
MCR p15,4,<Rt>,c12,c14,3 ; Write Rt to ICH_LRC3
```

Interrupt Controller Hyp Active Priorities Group 0 Register 0

The ICH_AP0R0 register provides information about Group 0 active priorities for EL2. This register is an alias of ICV_AP0R0.

Usage constraints This register is read/write, and is only available at EL2.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations This register is available in all build configurations.

Attributes ICH_AP0R0 is a 32-bit register.

The following figure shows the ICH_AP0R0 bit assignments.

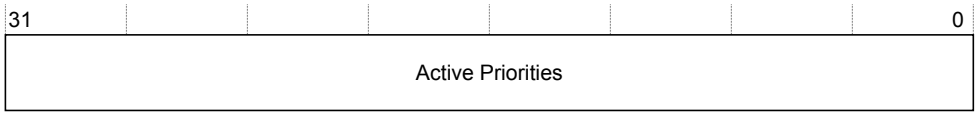

Figure 9-42 ICH_AP0R0 bit assignments

The following table shows the ICH_AP0R0 bit assignments.

Table 9-77 ICH_AP0R0 bit assignments

Bits	Name	Function
[n]	P<n>	P<n>, bit[n], for n = 0 to 31. Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: `0` There is no Group 0 interrupt active at priority n. `1` There is a Group 0 interrupt active at priority n.

To access ICH_AP0R0:

```
MRC p15,4,<Rt>,c12,c8,0 ; Read ICH_AP0R0 into Rt
MCR p15,4,<Rt>,c12,c8,0 ; Write Rt to ICH_AP0R0
```

Interrupt Controller Hyp Active Priorities Group 1 Register 0

The ICH_AP1R0 register provides information about Group 1 active priorities for EL2. This register is an alias of ICV_AP1R0.

Usage constraints This register is read/write, and is only available at EL2.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations This register is available in all build configurations.

Attributes ICH_AP1R0 is a 32-bit register.

The following figure shows the ICH_AP1R0 bit assignments.

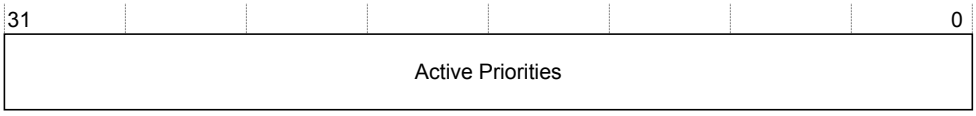

Figure 9-43 ICH_AP1R0 bit assignments

The following table shows the ICH_AP1R0 bit assignments.

Table 9-78 ICH_AP1R0 bit assignments

Bits	Name	Function
[n]	P<n>	P<n>, bit[n], for n = 0 to 31. Group 1 interrupt active priorities. Possible values of each bit are: `0` There is no Group 1 interrupt active at priority n. `1` There is a Group 1 interrupt active at priority n.

To access ICH_AP1R0:

```
MRC p15,4,<Rt>,c12,c9,0 ; Read ICH_AP1R0 into Rt
MCR p15,4,<Rt>,c12,c9,0 ; Write Rt to ICH_AP1R0
```

9.3.4 CPU Interface Registers

The following table summarizes the CPU interface registers within each core which are allocated to the (coproc==`0b1111`) space.

Table 9-79 CPU Interface Registers summary

Name	CRn	Op1	CRm	Op2	Type	Reset	Description
ICC_IAR0	c12	0	c8	0	RO	`0x000003FF`	*Interrupt Controller Interrupt Acknowledge Register 0* on page 9-314
ICC_IAR1		0	c12	0	RO	`0x000003FF`	*Interrupt Controller Interrupt Acknowledge Register 1* on page 9-315
ICC_EOIR0		0	c8	1	WO	UNK	*Interrupt Controller End Of Interrupt Register 0* on page 9-316
ICC_EOIR1		0	c12	1	WO	UNK	*Interrupt Controller End Of Interrupt Register 1* on page 9-316
ICC_HPPIR0		0	c8	2	RO	`0x000003FF`	*Interrupt Controller Highest Priority Pending Interrupt Register 0* on page 9-317
ICC_HPPIR1		0	c12	2	RO	`0x000003FF`	*Interrupt Controller Highest Priority Pending Interrupt Register 1* on page 9-318
ICC_BPR0		0	c8	3	RW	`0x00000002`	*Interrupt Controller Binary Point Register 0* on page 9-318
ICC_BPR1		0	c12	3	RW	`0x00000003`	*Interrupt Controller Binary Point Register 1* on page 9-319
ICC_DIR		0	c11	1	WO	UNK	*Interrupt Controller Deactivate Interrupt Register* on page 9-320
ICC_PMR	c4	0	c6	0	RW	`0x00000000`	*Interrupt Controller Interrupt Priority Mask Register* on page 9-321

Table 9-79 CPU Interface Registers summary (continued)

Name	CRn	Op1	CRm	Op2	Type	Reset	Description
ICC_RPR	c12	0	c11	3	RO	0x000000FF	*Interrupt Controller Running Priority Register* on page 9-322
ICC_CTLR		0	c12	4	RW	0x00000400	*Interrupt Controller Control Register (EL1)* on page 9-322
ICC_SRE		0	c12	5	RW	0x00000007	*Interrupt Controller System Register Enable register (EL1)* on page 9-323
ICC_HSRE		4	c9	5	RO	0x0000000F	*Interrupt Controller System Register Enable register (EL2)* on page 9-324
ICC_IGRPEN0		0	c12	6	RW	0x00000000	*Interrupt Controller Interrupt Group 0 Enable register* on page 9-325
ICC_IGRPEN1		0	c12	7	RW	0x00000000	*Interrupt Controller Interrupt Group 1 Enable register* on page 9-326
ICC_SGI0R	-	2	c12	-	WO	UNK	*Interrupt Controller Software Generated Interrupt Group 0 Register* on page 9-326
ICC_SGI1R	-	0	c12	-	WO	UNK	*Interrupt Controller Software Generated Interrupt Group 1 Register* on page 9-328
ICC_ASGI1R	-	1	c12	-	WO	UNK	*Interrupt Controller Alias Software Generated Interrupt Group 1 Register* on page 9-329
ICC_AP0R0	c12	0	c8	4	RW	0x00000000	*Interrupt Controller Active Priorities Group 0 Register* on page 9-330
ICC_AP1R0		0	c9	0	RW	0x00000000	*Interrupt Controller Active Priorities Group 1 Register* on page 9-331

Note

Writing to ICC_AP0R0 and ICC_AP0R1 with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in UNPREDICTABLE behavior of the interrupt prioritization system.

Interrupt Controller Interrupt Acknowledge Register 0

The ICC_IAR0 register contains the INTID of the signaled Group 0 interrupt. When the core reads this INTID, it acts as an acknowledge for the interrupt. In normal operation, the corresponding active priority bit of group 0 is set and the interrupt is activated.

Usage constraints This register is read-only.

Traps and enables If HSTR.T12 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then read accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_IAR0 is a 32-bit register.

The following figure shows the ICC_IAR0 bit assignments.

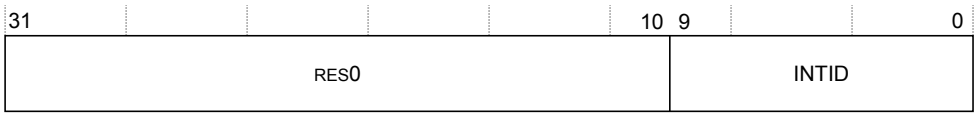

Figure 9-44 ICC_IAR0 bit assignments

The following table shows the ICC_IAR0 bit assignments.

Table 9-80 ICC_IAR0 bit assignments

Bits	Name	Function
[31:10]	-	Reserved, RES0.
[9:0]	INTID	The INTID of the signaled interrupt.

To access ICC_IAR0:

```
MRC p15,0,<Rt>,c12,c8,0 ; Read ICC_IAR0 into Rt
MCR p15,0,<Rt>,c12,c8,0 ; Write Rt to ICC_IAR0
```

Interrupt Controller Interrupt Acknowledge Register 1

The ICC_IAR1 register contains the INTID of the signaled Group 1 interrupt. When the core reads this INTID, it acts as an acknowledge for the interrupt. In normal operation, the corresponding active priority bit of group 1 is set and the interrupt is activated.

Usage constraints This register is read-only.

Traps and enables If HSTR.T12 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then read accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_IAR1 is a 32-bit register.

The following figure shows the ICC_IAR1 bit assignments.

Figure 9-45 ICC_IAR1 bit assignments

The following table shows the ICC_IAR1 bit assignments.

Table 9-81 ICC_IAR1 bit assignments

Bits	Name	Function
[31:10]	-	Reserved, RES0.
[9:0]	INTID	The INTID of the signaled interrupt.

To access ICC_IAR1:

```
MRC p15,0,<Rt>,c12,c12,0 ; Read ICC_IAR1 into Rt
MCR p15,0,<Rt>,c12,c12,0 ; Write Rt to ICC_IAR1
```

Interrupt Controller End Of Interrupt Register 0

A core can write to the ICC_EOIR0 register to inform the CPU interface that it has completed the processing of the specified Group 0 interrupt. In normal operation, the highest priority set group 0 priority bit is cleared and additionally the interrupt is deactivated if ICC_CTLR.EOImode == 0.

Usage constraints This register is write-only.

Traps and enables If HSTR.T12 is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then write accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_EOIR0 is a 32-bit register.

The following figure shows the ICC_EOIR0 bit assignments.

Figure 9-46 ICC_EOIR0 bit assignments

The following table shows the ICC_EOIR0 bit assignments.

Table 9-82 ICC_EOIR0 bit assignments

Bits	Name	Function
[31:10]	-	Reserved, RES0.
[9:0]	INTID	The INTID from the corresponding ICC_IAR0 access.

To access ICC_EOIR0:

```
MRC p15,0,<Rt>,c12,c8,1 ; Read ICC_EOIR0 into Rt
MCR p15,0,<Rt>,c12,c8,1 ; Write Rt to ICC_EOIR0
```

Interrupt Controller End Of Interrupt Register 1

A core can write to the ICC_EOIR1 register to inform the CPU interface that it has completed the processing of the specified Group 1 interrupt. In normal operation, the highest priority set group 1 priority bit is cleared and additionally the interrupt is deactivated if ICC_CTLR.EOIMode = 0.

Usage constraints This register is write-only.

Traps and enables If HSTR.T12 is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then write accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_EOIR1 is a 32-bit register.

The following figure shows the ICC_EOIR1 bit assignments.

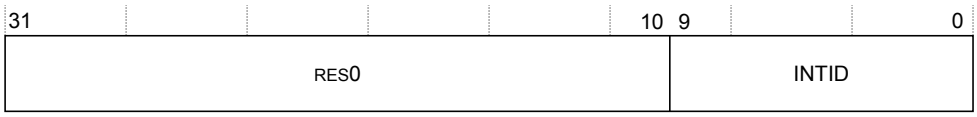

Figure 9-47 ICC_EOIR1 bit assignments

The following table shows the ICC_EOIR1 bit assignments.

Table 9-83 ICC_EOIR1 bit assignments

Bits	Name	Function
[31:10]	-	Reserved, RES0.
[9:0]	INTID	The INTID from the corresponding ICC_IAR1 access.

To access ICC_EOIR1:

```
MRC p15,0,<Rt>,c12,c12,1 ; Read ICC_EOIR1 into Rt
MCR p15,0,<Rt>,c12,c12,1 ; Write Rt to ICC_EOIR1
```

Interrupt Controller Highest Priority Pending Interrupt Register 0

The ICC_HPPIR0 register indicates the highest priority pending Group 0 interrupt on the CPU interface without changing the state of the GIC.

Usage constraints This register is read-only without changing the state of the GIC.

Traps and enables If HSTR.T12 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then read accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_HPPIR0 is a 32-bit register.

The following figure shows the ICC_HPPIR0 bit assignments.

Figure 9-48 ICC_HPPIR0 bit assignments

The following table shows the ICC_HPPIR0 bit assignments.

Table 9-84 ICC_HPPIR0 bit assignments

Bits	Name	Function
[31:10]	-	Reserved, RES0.
[9:0]	INTID	The INTID of the highest priority pending Group 0 interrupt on the CPU interface.

To access ICC_HPPIR0:

```
MRC p15,0,<Rt>,c12,c8,2 ; Read ICC_HPPIR0 into Rt
MCR p15,0,<Rt>,c12,c8,2 ; Write Rt to ICC_HPPIR0
```

Interrupt Controller Highest Priority Pending Interrupt Register 1

The ICC_HPPIR1 register indicates the highest priority pending Group 1 interrupt on the CPU interface without changing the state of the GIC.

Usage constraints This register is read-only.

Traps and enables If HSTR.T12 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then read accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_HPPIR1 is a 32-bit register.

The following figure shows the ICC_HPPIR1 bit assignments.

Figure 9-49 ICC_HPPIR1 bit assignments

The following table shows the ICC_HPPIR1 bit assignments.

Table 9-85 ICC_HPPIR1 bit assignments

Bits	Name	Function
[31:10]	-	Reserved, RES0.
[9:0]	INTID	The INTID of the highest priority pending Group 1 interrupt on the CPU interface.

To access ICC_HPPIR1:

```
MRC p15,0,<Rt>,c12,c12,2 ; Read ICC_HPPIR1 into Rt
MCR p15,0,<Rt>,c12,c12,2 ; Write Rt to ICC_HPPIR1
```

Interrupt Controller Binary Point Register 0

The ICC_BPR0 register defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 0 interrupt preemption.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_BPR0 is a 32-bit register.

The following figure shows the ICC_BPR0 bit assignments.

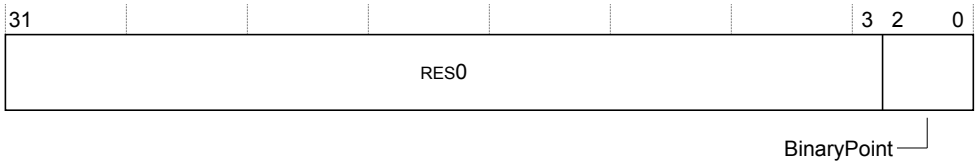

Figure 9-50 ICC_BPR0 bit assignments

The following table shows the ICC_BPR0 bit assignments.

Table 9-86 ICC_BPR0 bit assignments

Bits	Name	Function
[31:3]	-	Reserved, RES0.
[2:0]	BinaryPoint	This value controls how the 5-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. See *Table 9-87 ICC_BPR0 relationship between binary point value and group priority, subpriority fields* on page 9-319. The reset value is `0x2`.

Table 9-87 ICC_BPR0 relationship between binary point value and group priority, subpriority fields

BinaryPoint value	Group priority field	Subpriority field	Field with binary point
2	[7:3]	-	ggggg.
3	[7:4]	[3]	gggg.s
4	[7:5]	[4:3]	ggg.ss
5	[7:6]	[5:3]	gg.sss
6	[7]	[6:3]	g.ssss
7	No preemption	[7:3]	.sssss

To access ICC_BPR0:

```
MRC p15,0,<Rt>,c12,c8,3 ; Read ICC_BPR0 into Rt
MCR p15,0,<Rt>,c12,c8,3 ; Write Rt to ICC_BPR0
```

Interrupt Controller Binary Point Register 1

The ICC_BPR1 register defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_BPR1 is a 32-bit register.

The following figure shows the ICC_BPR1 bit assignments.

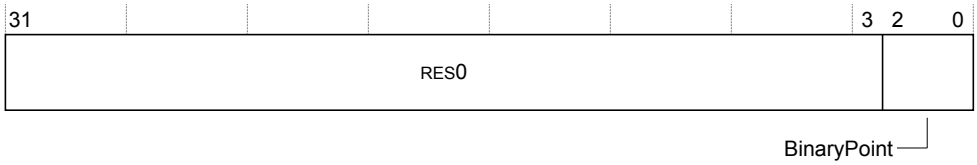

Figure 9-51 ICC_BPR1 bit assignments

The following table shows the ICC_BPR1 bit assignments.

Table 9-88 ICC_BPR1 bit assignments

Bits	Name	Function
[31:3]	-	Reserved, RES0.
[2:0]	BinaryPoint	This value controls how the 5-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. See *Table 9-89 ICC_BPR1 relationship between binary point value and group priority, subpriority fields* on page 9-320. The reset value is `0x3`.

Table 9-89 ICC_BPR1 relationship between binary point value and group priority, subpriority fields

BinaryPoint value	Group priority field	Subpriority field	Field with binary point
3	[7:3]	-	ggggg.
4	[7:4]	[3]	gggg.s
5	[7:5]	[4:3]	ggg.ss
6	[7:6]	[5:3]	gg.sss
7	[7]	[6:3]	g.ssss

To access ICC_BPR1:

```
MRC p15,0,<Rt>,c12,c12,3 ; Read ICC_BPR1 into Rt
MCR p15,0,<Rt>,c12,c12,3 ; Write Rt to ICC_BPR1
```

Interrupt Controller Deactivate Interrupt Register

When interrupt priority drop is separated from interrupt deactivation, ICC_CTLR.EOIMode=1, a write to the ICC_DIR register deactivates the specified interrupt.

Usage constraints This register is write-only.

Traps and enables If HSTR.T12 is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TC is set to 1, then write accesses to this register from EL1 are trapped to EL2.

If ICH_HCR.TDIR is set to 1, then write accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_DIR is a 32-bit register.

The following figure shows the ICC_DIR bit assignments.

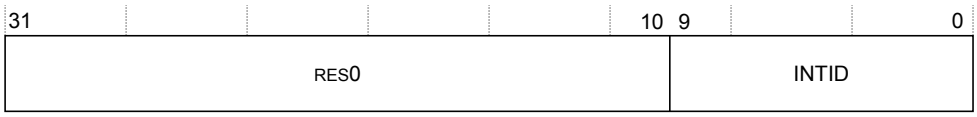

Figure 9-52 ICC_DIR bit assignments

The following table shows the ICC_DIR bit assignments.

Table 9-90 ICC_DIR bit assignments

Bits	Name	Function
[31:10]	-	Reserved, RES0.
[9:0]	INTID	The INTID of the interrupt to be deactivated.

To access ICC_DIR:

```
MRC p15,0,<Rt>,c12,c11,1 ; Read ICC_DIR into Rt
MCR p15,0,<Rt>,c12,c11,1 ; Write Rt to ICC_DIR
```

Interrupt Controller Interrupt Priority Mask Register

The ICC_PMR register provides an interrupt priority filter. Only interrupts with higher priority than the value in this register are signaled to the core.

Usage constraints This register is read/write.

Traps and enables If ICH_HCR.TC is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_PMR is a 32-bit register.

The following figure shows the ICC_PMR bit assignments.

Figure 9-53 ICC_PMR bit assignments

The following table shows the ICC_PMR bit assignments.

Table 9-91 ICC_PMR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:3]	Priority	The priority mask level for the CPU interface. If the priority of an interrupt is higher than the value indicated by this field, the interface signals the interrupt to the core. Lower values have higher priority.
[2:0]	-	RAZ/WI.

To access ICC_PMR:

```
MRC p15,0,<Rt>,c4,c6,0 ; Read ICC_PMR into Rt
MCR p15,0,<Rt>,c4,c6,0 ; Write Rt to ICC_PMR
```

Interrupt Controller Running Priority Register

The ICC_RPR register indicates the Running priority, the highest active priority across Groups 0 and 1, of the CPU interface.

Usage constraints This register is read only.

Traps and enables If HSTR.T12 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TC is set to 1, then read accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_RPR is a 32-bit register.

The following figure shows the ICC_RPR bit assignments.

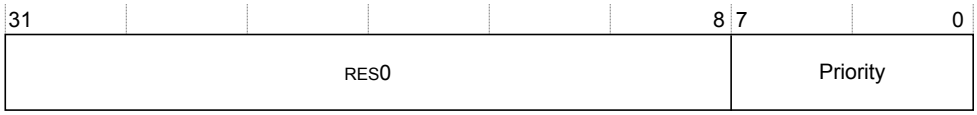

Figure 9-54 ICC_RPR bit assignments

The following table shows the ICC_RPR bit assignments.

Table 9-92 ICC_RPR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	Priority	The current running priority. Returns the value `0xFF` when ICC_AP0R0 and ICC_AP1R0 are both `0x0`. Otherwise returns the index in bits [7:3] of the lowest set bit from ICC_AP0R0 and ICC_AP1R0.

To access ICC_RPR:

```
MRC p15,0,<Rt>,c12,c11,3 ; Read ICC_RPR into Rt
MCR p15,0,<Rt>,c12,c11,3 ; Write Rt to ICC_RPR
```

Interrupt Controller Control Register (EL1)

The ICC_CTLR register controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TC is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_CTLR is a 32-bit register.

The following figure shows the ICC_CTLR bit assignments.

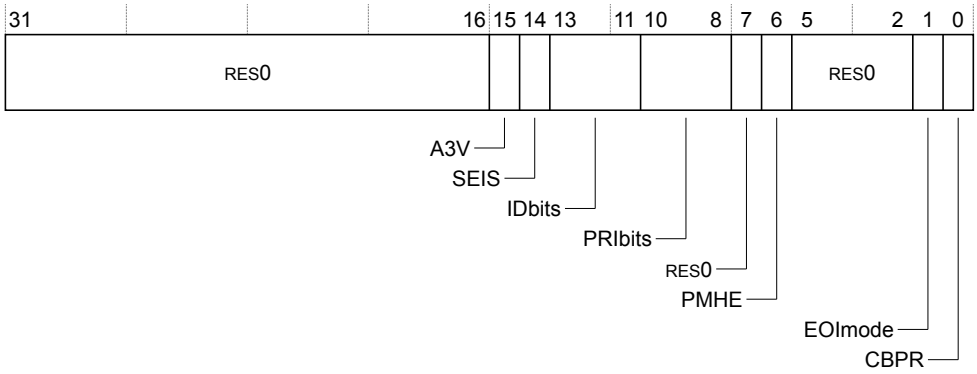

Figure 9-55 ICC_CTLR bit assignments

The following table shows the ICC_CTLR bit assignments.

Table 9-93 ICC_CTLR bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored: **0** The CPU interface logic only supports zero values of Affinity 3 in SGI generation system registers.
[14]	SEIS	SEI support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs: **0** The CPU interface logic does not support local generation of SEIs.
[13:11]	IDbits	Indicates the number of physical interrupt identifier bits supported: `0b000` 16 interrupt identifier bits are supported.
[10:8]	PRIbits	Indicates the number of priority bits implemented, minus one `0b100` Five priority bits are supported.
[7]	-	Reserved, RES0.
[6]	PMHE	This field is RES0.
[5:2]	-	Reserved, RES0.
[1]	EOImode	Controls whether a write to an ICC_EOIR0 and ICC_EOIR1 also deactivates the interrupt.
[0]	CBPR	Controls whether the ICC_BPR0 is used for interrupt preemption of both Group 0 and Group 1 interrupts.

To access ICC_CTLR:

```
MRC p15,0,<Rt>,c12,c12,4 ; Read ICC_CTLR into Rt
MCR p15,0,<Rt>,c12,c12,4 ; Write Rt to ICC_CTLR
```

Interrupt Controller System Register Enable Register (EL1)

The ICC_SRE register indicates that system registers are used to access the GIC CPU interface and that interrupt bypass is not supported.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations This register is available in all build configurations.

Attributes ICC_SRE is a 32-bit register.

The following figure shows the ICC_SRE bit assignments.

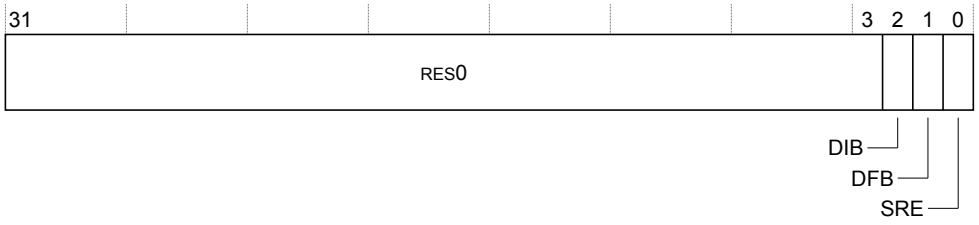

Figure 9-56 ICC_SRE bit assignments

The following table shows the ICC_SRE bit assignments.

Table 9-94 ICC_SRE bit assignments

Bits	Name	Function
[31:3]	-	Reserved, RES0.
[2]	DIB	Disable IRQ bypass: **RAO/WI** IRQ bypass is not supported.
[1]	DFB	Disable FIQ bypass: **RAO/WI** FIQ bypass is not supported.
[0]	SRE	System register enable: **RAO/WI** System registers are used to access the GIC CPU interface.

To access ICC_SRE:

```
MRC p15,0,<Rt>,c12,c12,5 ; Read ICC_SRE into Rt
MCR p15,0,<Rt>,c12,c12,5 ; Write Rt to ICC_SRE
```

Interrupt Controller System Register Enable Register (EL2)

The ICC_HSRE register controls whether the system register interface or the memory-mapped interface to the GIC CPU interface is used for EL2.

Usage constraints This register is read-only.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

Configurations This register is available in all build configurations.

Attributes ICC_HSRE is a 32-bit register.

The following figure shows the ICC_HSRE bit assignments.

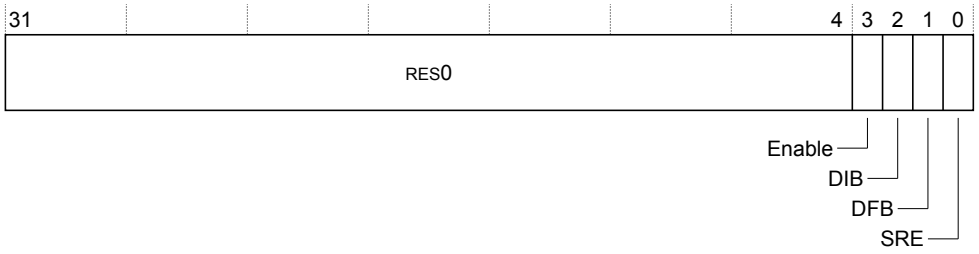

Figure 9-57 ICC_HSRE bit assignments

The following table shows the ICC_HSRE bit assignments.

Table 9-95 ICC_HSRE bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3]	Enable	Enables lower Exception level access to ICC_HSRE: **RAO/WI** Lower exception levels can read ICC_SRE.
[2]	DIB	Disable IRQ bypass: **RAO/WI** IRQ bypass is not supported.
[1]	DFB	Disable FIQ bypass: **RAO/WI** FIQ bypass is not supported.
[0]	SRE	System register enable: **RAO/WI** System registers are used to access the GIC CPU interface.

To access ICC_HSRE:

```
MRC p15,4,<Rt>,c12,c9,5 ; Read ICC_HSRE into Rt
MCR p15,4,<Rt>,c12,c9,5 ; Write Rt to ICC_HSRE
```

Interrupt Controller Interrupt Group 0 Enable Register

The ICC_IGRPEN0 register controls whether Group 0 interrupts are enabled.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, accesses to the register from EL1 are trapped to Hyp mode.
If ICH_HCR.TALL0 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_IGRPEN0 is a 32-bit register.

The following figure shows the ICC_IGRPEN0 bit assignments.

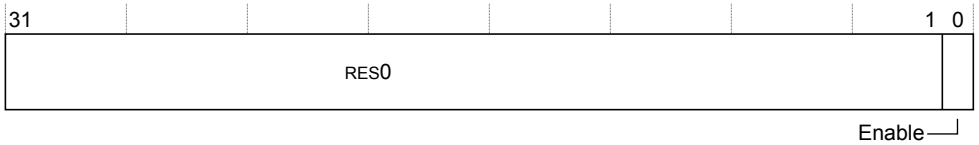

Figure 9-58 ICC_IGRPEN0 bit assignments

The following table shows the ICC_IGRPEN0 bit assignments.

Table 9-96 ICC_IGRPEN0 bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	Enable	Enables Group 0 interrupts: 0 Group 0 interrupts are disabled. This is the reset value. 1 Group 0 interrupts are enabled.

To access ICC_IGRPEN0:

```
MRC p15,0,<Rt>,c12,c12,6 ; Read ICC_IGRPEN0 into Rt
MCR p15,0,<Rt>,c12,c12,6 ; Write Rt to ICC_IGRPEN0
```

Interrupt Controller Interrupt Group 1 Enable Register

The ICC_IGRPEN1 register controls whether Group 1 interrupts are enabled.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_IGRPEN1 is a 32-bit register.

The following figure shows the ICC_IGRPEN1 bit assignments.

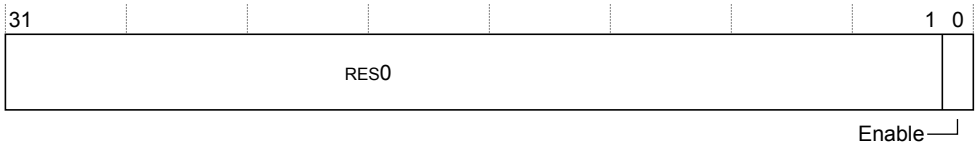

Figure 9-59 ICC_IGRPEN1 bit assignments

The following table shows the ICC_IGRPEN1 bit assignments.

Table 9-97 ICC_IGRPEN1 bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	Enable	Enables Group 1 interrupts: 0 Group 1 interrupts are disabled. This is the reset value. 1 Group 1 interrupts are enabled.

To access ICC_IGRPEN1:

```
MRC p15,0,<Rt>,c12,c12,7 ; Read ICC_IGRPEN1 into Rt
MCR p15,0,<Rt>,c12,c12,7 ; Write Rt to ICC_IGRPEN1
```

Interrupt Controller Software Generated Interrupt Group 0 Register

The ICC_SGI0R register is used to request Group 0 SGIs according to the GICR_IGROUPR configuration.

Usage constraints This register is write-only.

When IRM=0, unused affinity bits must be written as zeros or the Generate SGI request is not sent.

Traps and enables If HCR.FMO is set to 1, then write accesses to this register from EL1 are trapped to EL2.

If HCR.IMO is set to 1, then write accesses to this register from EL1 are trapped to EL2.

If HSTR.T12 is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TC is set to 1, then write accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_SGI0R is a 64-bit register.

The following figure shows the ICC_SGI0R bit assignments.

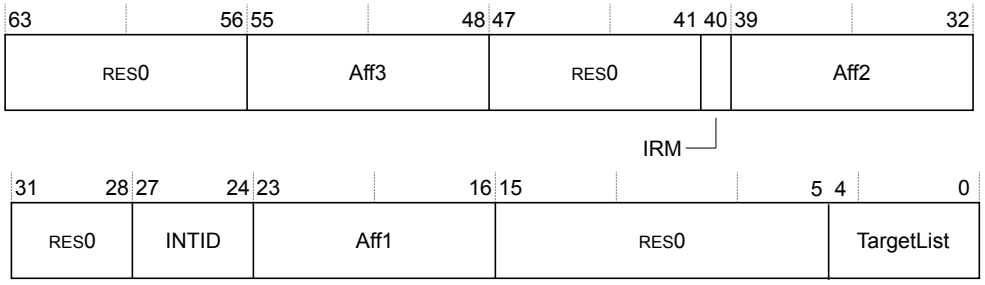

Figure 9-60 ICC_SGI0R bit assignments

The following table shows the ICC_SGI0R bit assignments.

Table 9-98 ICC_SGI0R bit assignments

Bits	Name	Function
[63:56]	-	Reserved, RES0.
[55:48]	Aff3	The Affinity level 3 value of the affinity path of the cluster for which SGI interrupts are generated. If the IRM bit is 1, this field is RES0.
[47:41]	-	Reserved, RES0.
[40]	IRM	Interrupt Routing Mode. Determines how the generated interrupts are distributed to cores: 0 Interrupts routed to the cores specified by Aff3.Aff2.Aff1.<target list>. 1 Interrupts routed to all cores in the system, excluding "self".
[39:32]	Aff2	The affinity 2 value of the affinity path of the cluster for which SGI interrupts are generated. If the IRM bit is 1, this field is RES0.
[31:28]	-	Reserved, RES0.
[27:24]	INTID	The interrupt ID of the SGI.
[23:16]	Aff1	The affinity 1 value of the affinity path of the cluster for which SGI interrupts are generated. If the IRM bit is 1, this field is RES0.

Table 9-98 ICC_SGI0R bit assignments (continued)

Bits	Name	Function
[15:5]	-	Reserved, RES0.
[4:0]	TargetList	Target List. The set of cores for which SGI interrupts are generated. Each bit corresponds to the core within a cluster with an Affinity 0 value equal to the bit number. If the IRM bit is 1, this field is RES0.

To access ICC_SGI0R:

```
MCRR p15,2,<Rt>,<Rt2>,c12 ; Write Rt to ICC_SGI0R[31:0] and Rt2 to ICC_SGI0R[63:32]
```

Interrupt Controller Software Generated Interrupt Group 1 Register

The ICC_SGI1R register is used to request Group 0 and 1 SGIs according to the configuration of GICR_IGROUPR configuration.

Use this register to generate an SGI regardless of GICR_IGROUPR0[INTID] for the targets.

Usage constraints This register is write-only. When IRM=0, unused affinity bits must be written as zeros or the Generate SGI request is not sent.

Traps and enables If HCR.FMO is set to 1, then write accesses to this register from EL1 are trapped to EL2.

If HCR.IMO is set to 1, then write accesses to this register from EL1 are trapped to EL2.

If HSTR.T12 is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TC is set to 1, then write accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_SGI1R is a 64-bit register.

The following figure shows the ICC_SGI1R bit assignments.

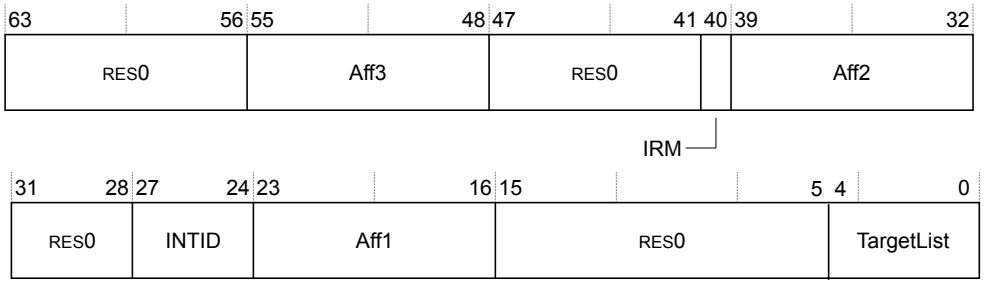

Figure 9-61 ICC_SGI1R bit assignments

The following table shows the ICC_SGI1R bit assignments.

Table 9-99 ICC_SGI1R bit assignments

Bits	Name	Function
[63:56]	-	Reserved, RES0.
[55:48]	Aff3	The affinity 3 value of the affinity path of the cluster for which SGI interrupts are generated. If the IRM bit is 1, this field is RES0.

Table 9-99 ICC_SGI1R bit assignments (continued)

Bits	Name	Function
[47:41]	-	Reserved, RES0.
[40]	IRM	Interrupt Routing Mode. Determines how the generated interrupts are distributed to cores: `0` Interrupts routed to the cores specified by Aff3.Aff2.Aff1.<target list>. `1` Interrupts routed to all cores in the system, excluding "self".
[39:32]	Aff2	The affinity 2 value of the affinity path of the cluster for which SGI interrupts are generated. If the IRM bit is 1, this field is RES0.
[31:28]	-	Reserved, RES0.
[27:24]	INTID	The interrupt ID of the SGI.
[23:16]	Aff1	The affinity 1 value of the affinity path of the cluster for which SGI interrupts are generated. If the IRM bit is 1, this field is RES0.
[15:5]	-	Reserved, RES0.
[4:0]	TargetList	Target List. The set of cores for which SGI interrupts are generated. Each bit corresponds to the core within a cluster with an Affinity 0 value equal to the bit number. If the IRM bit is 1, this field is RES0.

To access ICC_SGI1R:

```
MCRR p15,0,<Rt>,<Rt2>,c12 ; Write Rt to ICC_SGI1R[31:0] and Rt2 to ICC_SGI1R[63:32]
```

Interrupt Controller Alias Software Generated Interrupt Group 1 Register

The ICC_ASGI1R generates Group 0 SGIs, if GICR_IGROUPR0[INTID] is configured to 0 for that target. This register has identical behavior to ICC_SGI0R.

Usage constraints This register is write-only.

When IRM=0, unused affinity bits must be written as zeros or the Generate SGI request is not sent.

Traps and enables If HCR.FMO is set to 1, then write accesses to this register from EL1 are trapped to EL2.

If HCR.IMO is set to 1, then write accesses to this register from EL1 are trapped to EL2.

If HSTR.T12 is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TC is set to 1, then write accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_ASGI1R is a 64-bit register.

The following figure shows the ICC_ASGI1R bit assignments.

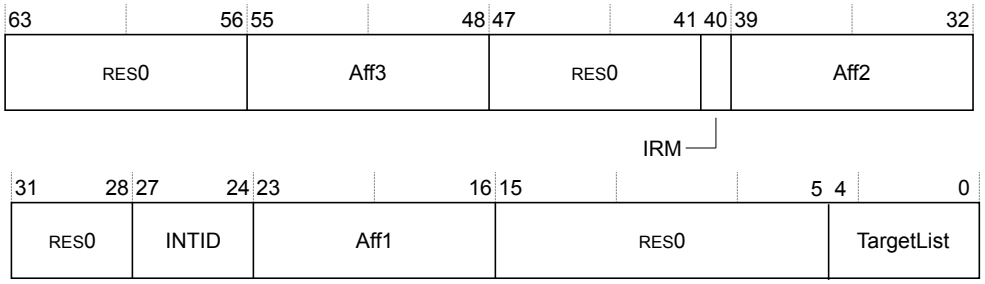

Figure 9-62 ICC_ASGI1R bit assignments

The following table shows the ICC_ASGI1R bit assignments.

Table 9-100 ICC_ASGI1R bit assignments

Bits	Name	Function
[63:56]	-	Reserved, RES0.
[55:48]	Aff3	The affinity 3 value of the affinity path of the cluster for which SGI interrupts are generated. If the IRM bit is 1, this field is RES0.
[47:41]	-	Reserved, RES0.
[40]	IRM	Interrupt Routing Mode. Determines how the generated interrupts are distributed to cores: `0` Interrupts routed to the cores specified by Aff3.Aff2.Aff1.<target list>. `1` Interrupts routed to all cores in the system, excluding "self".
[39:32]	Aff2	The affinity 2 value of the affinity path of the cluster for which SGI interrupts are generated. If the IRM bit is 1, this field is RES0.
[31:28]	-	Reserved, RES0.
[27:24]	INTID	The interrupt ID of the SGI.
[23:16]	Aff1	The affinity 1 value of the affinity path of the cluster for which SGI interrupts are generated. If the IRM bit is 1, this field is RES0.
[15:5]	-	Reserved, RES0.
[4:0]	TargetList	Target List. The set of cores for which SGI interrupts are generated. Each bit corresponds to the core within a cluster with an Affinity 0 value equal to the bit number. If the IRM bit is 1, this field is RES0.

To access ICC_ASGI1R:

```
MCRR p15,1,<Rt>,<Rt2>,c12 ; Write Rt to ICC_ASGI1R[31:0] and Rt2 to ICC_ASGI1R[63:32]
```

Interrupt Controller Active Priorities Group 0 Register

The ICC_AP0R0 register provides information about Group 0 active priorities.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_AP0R0 is a 32-bit register.

The following figure shows the ICC_AP0R0 bit assignments.

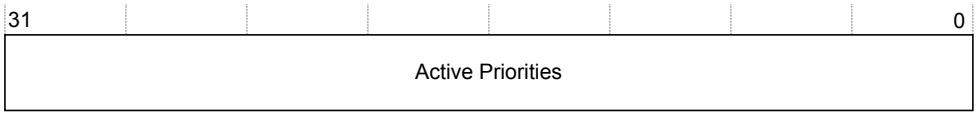

Figure 9-63 ICC_AP0R0 bit assignments

The following table shows the ICC_AP0R0 bit assignments.

Table 9-101 ICC_AP0R0 bit assignments

Bits	Name	Function
[n]	P<n>	P<n>, bit[n], for n = 0 to 31. Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: `0` There is no Group 0 interrupt active at the priority corresponding to that bit. `1` There is a Group 0 interrupt active at the priority corresponding to that bit.

To access ICC_AP0R0:

```
MRC p15,0,<Rt>,c12,c8,4 ; Read ICC_AP0R0 into Rt
MCR p15,0,<Rt>,c12,c8,4 ; Write Rt to ICC_AP0R0
```

Interrupt Controller Active Priorities Group 1 Register

The ICC_AP1R0 register provides information about Group 1 active priorities.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICC_AP1R0 is a 32-bit register.

The following figure shows the ICC_AP1R0 bit assignments.

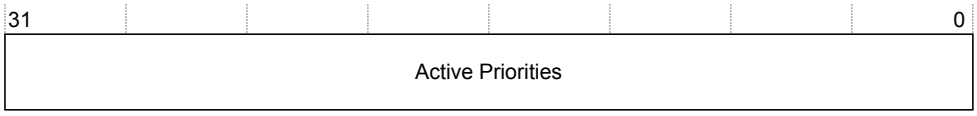

Figure 9-64 ICC_AP1R0 bit assignments

The following table shows the ICC_AP1R0 bit assignments.

Table 9-102 ICC_AP1R0 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[n]	P<n>	P<n>, bit[n], for n = 0 to 31. Group 1 interrupt active priorities. Possible values of each bit are: 0 There is no Group 1 interrupt active at the priority corresponding to that bit. 1 There is a Group 1 interrupt active at the priority corresponding to that bit.

To access ICC_AP1R0:

```
MRC p15,0,<Rt>,c12,c9,0 ; Read ICC_AP1R0 into Rt
MCR p15,0,<Rt>,c12,c9,0 ; Write Rt to ICC_AP1R0
```

9.3.5 Virtual CPU Interface Registers

The following table summarizes the virtual CPU interface System registers.

Accesses at EL1 to Group 0 registers are virtual when HCR.FMO == 1.

Virtual accesses to the following Group 0 ICC_* registers access the ICV_* equivalents:

- Accesses to ICC_AP0R0 access ICV_AP0R0.
- Accesses to ICC_BPR0 access ICV_BPR0.
- Accesses to ICC_EOIR0 access ICV_EOIR0.
- Accesses to ICC_HPPIR0 access ICV_HPPIR0.
- Accesses to ICC_IAR0 access ICV_IAR0.
- Accesses to ICC_IGRPEN0 access ICV_IGRPEN0.

Accesses at EL1 to Group 1 registers are virtual when HCR.IMO == 1.

Virtual accesses to the following Group 1 ICC_* registers access the ICV_* equivalents:

- Accesses to ICC_AP1R0 access ICV_AP1R0.
- Accesses to ICC_BPR1 access ICV_BPR1.
- Accesses to ICC_EOIR1 access ICV_EOIR1.
- Accesses to ICC_HPPIR1 access ICV_HPPIR1.
- Accesses to ICC_IAR1 access ICV_IAR1.
- Accesses to ICC_IGRPEN1 access ICV_IGRPEN1.

Accesses at EL1 to the common registers are virtual when either HCR.IMO == 1 or HCR.FMO == 1, or both.

Virtual accesses to the following Common ICC_* registers access the ICV_* equivalents:

- Accesses to ICC_RPR access ICV_RPR.
- Accesses to ICC_CTLR access ICV_CTLR.
- Accesses to ICC_DIR access ICV_DIR.
- Accesses to ICC_PMR access ICV_PMR.

A virtual write to ICC_SGI0R, ICC_SGI1R, or ICC_ASGI1R traps to EL2.

Software executing at EL2 can access some ICV_* register state using ICH_VMCR and ICH_VTR as follows:

- ICV_PMR.Priority aliases ICH_VMCR.VPMR.
- ICV_BPR0.BinaryPoint aliases ICH_VMCR.VBPR0.
- ICV_BPR1.BinaryPoint aliases ICH_VMCR.VBPR1.
- ICV_CTLR.EOImode aliases ICH_VMCR.VEOIM.
- ICV_CTLR.CBPR aliases ICH_VMCR.VCBPR.

- ICV_IGRPEN0 aliases ICH_VMCR.VENG0.
- ICV_IGRPEN1 aliases ICH_VMCR.VENG1.
- ICV_CTLR.PRIbits aliases ICH_VTR.PRIbits.

The following table shows the virtual CPU interface registers.

Table 9-103 Virtual CPU Interface Registers summary

Name	CRn	Op1	CRm	Op2	Type	Reset	Description
ICV_IAR0	c12	0	c8	0	RO	0x000003ff	*Interrupt Controller Virtual Interrupt Acknowledge Register 0* on page 9-333
ICV_IAR1		0	c12	0	RO	0x000003ff	*Interrupt Controller Virtual Interrupt Acknowledge Register 1* on page 9-334
ICV_EOIR0		0	c8	1	WO	UNK	*Interrupt Controller Virtual End Of Interrupt Register 0* on page 9-335
ICV_EOIR1		0	c12	1	WO	UNK	*Interrupt Controller Virtual End Of Interrupt Register 1* on page 9-335
ICV_HPPIR0		0	c8	2	RO	0x000003ff	*Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0* on page 9-336
ICC_HPPIR1		0	c12	2	RO	0x000003ff	*Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1* on page 9-337
ICV_BPR0		0	c8	3	RW	0x00000002	*Interrupt Controller Virtual Binary Point Register 0* on page 9-337
ICV_BPR1		0	c12	3	RW	0x00000003	*Interrupt Controller Virtual Binary Point Register 1* on page 9-338
ICV_DIR		0	c11	1	WO	UNK	*Interrupt Controller Deactivate Virtual Interrupt Register* on page 9-339
ICV_PMR	c4	0	c6	0	RW	0x00000000	*Interrupt Controller Virtual Interrupt Priority Mask Register* on page 9-340
ICV_RPR	c12	0	c11	3	RO	0x000000ff	*Interrupt Controller Virtual Running Priority Register* on page 9-341
ICV_CTLR		0	c12	4	RW	0x00000400	*Interrupt Controller Virtual Control Register* on page 9-341
ICV_IGRPEN0		0	c12	6	RW	0x00000000	*Interrupt Controller Virtual Interrupt Group 0 Enable Register* on page 9-342
ICV_IGRPEN1		0	c12	7	RW	0x00000000	*Interrupt Controller Virtual Interrupt Group 1 Enable Register* on page 9-343
ICV_AP0R0		0	c8	4	RW	0x00000000	*Interrupt Controller Virtual Active Priorities Group 0 Register* on page 9-344
ICV_AP1R0		0	c9	0	RW	0x00000000	*Interrupt Controller Virtual Active Priorities Group 1 Register* on page 9-344

Interrupt Controller Virtual Interrupt Acknowledge Register 0

Software reads the ICV_IAR0 register to obtain the INTID of the signaled virtual Group 0 interrupt. This read acts as an acknowledge for the interrupt.

Usage constraints This register is read-only, and is only available at EL1.

Traps and enables If HSTR.T12 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then read accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_IAR0 is a 32-bit register.

The following figure shows the ICV_IAR0 bit assignments.

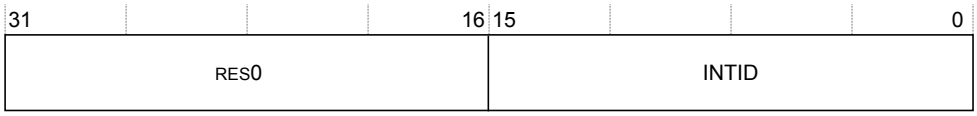

Figure 9-65 ICV_IAR0 bit assignments

The following table shows the ICV_IAR0 bit assignments.

Table 9-104 ICV_IAR0 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15:0]	INTID	The INTID of the signaled virtual interrupt.

To access ICV_IAR0:

```
MRC p15,0,<Rt>,c12,c8,0 ; Read ICV_IAR0 into Rt
MCR p15,0,<Rt>,c12,c8,0 ; Write Rt to ICV_IAR0
```

Interrupt Controller Virtual Interrupt Acknowledge Register 1

Software reads the ICV_IAR1 register to obtain the INTID of the signaled virtual Group 1 interrupt. This read acts as an acknowledge for the interrupt.

Usage constraints This register is read-only, and is only available at EL1.

Traps and enables If HSTR.T12 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then read accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_IAR1 is a 32-bit register.

The following figure shows the ICV_IAR1 bit assignments.

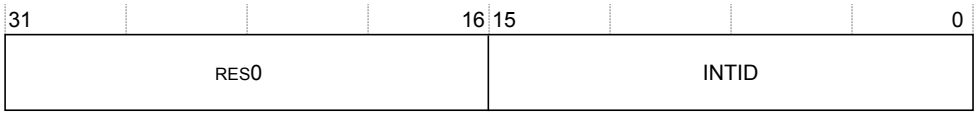

Figure 9-66 ICV_IAR1 bit assignments

The following table shows the ICV_IAR1 bit assignments.

Table 9-105 ICV_IAR1 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15:0]	INTID	The INTID of the signaled interrupt.

To access ICV_IAR1:

```
MRC p15,0,<Rt>,c12,c12,0 ; Read ICV_IAR1 into Rt
MCR p15,0,<Rt>,c12,c12,0 ; Write Rt to ICV_IAR1
```

Interrupt Controller Virtual End Of Interrupt Register 0

Software writes to the ICV_EOIR0 register to inform the CPU interface that it has completed the processing of the specified virtual Group 0 interrupt.

Usage constraints This register is write-only.

Traps and enables If HSTR.T12 is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then write accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_EOIR0 is a 32-bit register.

The following figure shows the ICV_EOIR0 bit assignments.

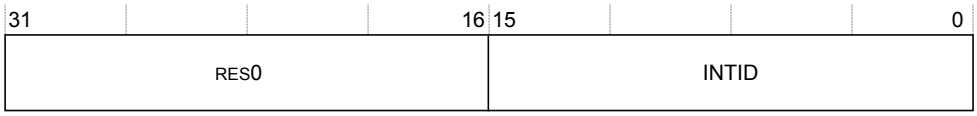

Figure 9-67 ICV_EOIR0 bit assignments

The following table shows the ICV_EOIR0 bit assignments.

Table 9-106 ICV_EOIR0 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15:0]	INTID	The INTID from the corresponding ICV_IAR0 access.

To access ICV_EOIR0:

```
MRC p15,0,<Rt>,c12,c8,1 ; Read ICV_EOIR0 into Rt
MCR p15,0,<Rt>,c12,c8,1 ; Write Rt to ICV_EOIR0
```

Interrupt Controller Virtual End Of Interrupt Register 1

Software writes to the ICV_EOIR1 register to inform the CPU interface that it has completed the processing of the specified virtual Group 1 interrupt.

Usage constraints This register is write-only.

Traps and enables If HSTR.T12 is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then write accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_EOIR1 is a 32-bit register.

The following figure shows the ICV_EOIR1 bit assignments.

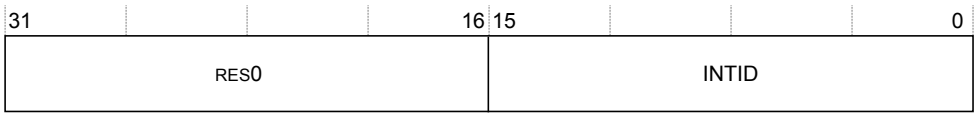

Figure 9-68 ICV_EOIR1 bit assignments

The following table shows the ICV_EOIR1 bit assignments.

Table 9-107 ICV_EOIR1 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15:0]	INTID	The INTID from the corresponding ICC_IAR1 access.

To access ICV_EOIR1:

```
MRC p15,0,<Rt>,c12,c12,1 ; Read ICV_EOIR1 into Rt
MCR p15,0,<Rt>,c12,c12,1 ; Write Rt to ICV_EOIR1
```

Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0

The ICV_HPPIR0 register indicates the highest priority pending virtual Group 0 interrupt on the virtual CPU interface.

Usage constraints This register is read-only, and is only available at EL1.

Traps and enables If HSTR.T12 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then read accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_HPPIR0 is a 32-bit register.

The following figure shows the ICV_HPPIR0 bit assignments.

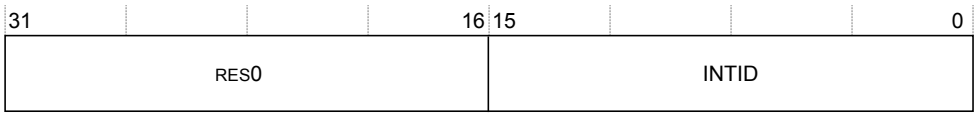

Figure 9-69 ICV_HPPIR0 bit assignments

The following table shows the ICV_HPPIR0 bit assignments.

Table 9-108 ICV_HPPIR0 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15:0]	INTID	The INTID of the highest priority pending virtual interrupt.

To access ICV_HPPIR0:

```
MRC p15,0,<Rt>,c12,c8,2 ; Read ICV_HPPIR0 into Rt
MCR p15,0,<Rt>,c12,c8,2 ; Write Rt to ICV_HPPIR0
```

Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1

The ICV_HPPIR1 register indicates the highest priority pending virtual Group 1 interrupt on the virtual CPU interface.

Usage constraints This register is read-only, and is only available at EL1.

Traps and enables If HSTR.T12 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then read accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_HPPIR1 is a 32-bit register.

The following figure shows the ICV_HPPIR1 bit assignments.

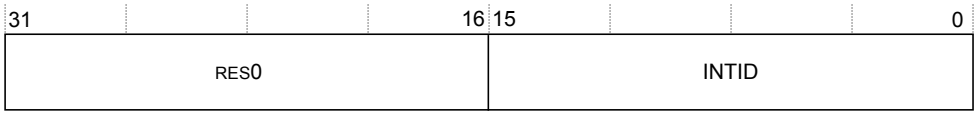

Figure 9-70 ICV_HPPIR1 bit assignments

The following table shows the ICV_HPPIR1 bit assignments.

Table 9-109 ICV_HPPIR1 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15:0]	INTID	The INTID of the highest priority pending virtual interrupt.

To access ICV_HPPIR1:

```
MRC p15,0,<Rt>,c12,c12,2 ; Read ICV_HPPIR1 into Rt
MCR p15,0,<Rt>,c12,c12,2 ; Write Rt to ICV_HPPIR1
```

Interrupt Controller Virtual Binary Point Register 0

The ICV_BPR0 register defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 0 interrupt preemption.

Usage constraints This register is read/write, and is only available at EL1.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_BPR0 is a 32-bit register.

The following figure shows the ICV_BPR0 bit assignments.

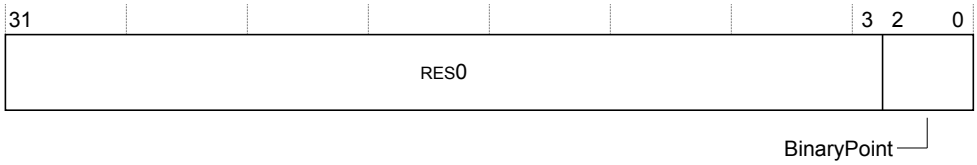

Figure 9-71 ICV_BPR0 bit assignments

The following table shows the ICV_BPR0 bit assignments.

Table 9-110 ICV_BPR0 bit assignments

Bits	Name	Function
[31:3]	-	Reserved, RES0.
[2:0]	BinaryPoint	This value controls how the 5-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. See *Table 9-111 ICV_BPR0 relationship between binary point value and group priority, subpriority fields* on page 9-338. The reset value is `0x2`.

Table 9-111 ICV_BPR0 relationship between binary point value and group priority, subpriority fields

Binary point value	Group priority field	Subpriority field	Field with binary point
2	[7:3]	-	ggggg.
3	[7:4]	[3]	gggg.s
4	[7:5]	[4:3]	ggg.ss
5	[7:6]	[5:3]	gg.sss
6	[7]	[6:3]	g.ssss
7	No preemption	[7:3]	.sssss

To access ICV_BPR0:

```
MRC p15,0,<Rt>,c12,c8,3 ; Read ICV_BPR0 into Rt
MCR p15,0,<Rt>,c12,c8,3 ; Write Rt to ICV_BPR0
```

Interrupt Controller Virtual Binary Point Register 1

The ICV_BPR1 register defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 1 interrupt preemption.

Usage constraints This register is read/write, and is only available at EL1.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_BPR1 is a 32-bit register.

The following figure shows the ICV_BPR1 bit assignments.

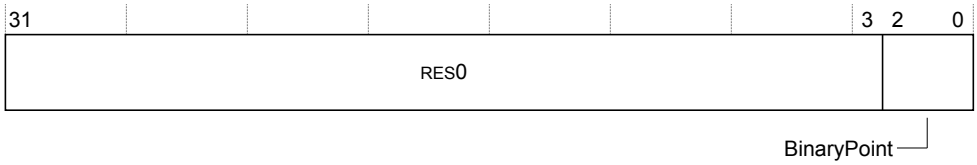

Figure 9-72 ICV_BPR1 bit assignments

The following table shows the ICV_BPR1 bit assignments.

Table 9-112 ICV_BPR1 bit assignments

Bits	Name	Function
[31:3]	-	Reserved, RES0.
[2:0]	BinaryPoint	This value controls how the interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. See *Table 9-113 ICV_BPR1 relationship between binary point value and group priority, subpriority fields* on page 9-339. The reset value is `0x3`.

Table 9-113 ICV_BPR1 relationship between binary point value and group priority, subpriority fields

Binary point value	Group priority field	Subpriority field	Field with binary point
3	[7:3]	-	ggggg.
4	[7:4]	[3]	gggg.s
5	[7:5]	[4:3]	ggg.ss
6	[7:6]	[5:3]	gg.sss
7	[7]	[6:3]	g.ssss

To access ICV_BPR1:

```
MRC p15,0,<Rt>,c12,c12,3 ; Read ICV_BPR1 into Rt
MCR p15,0,<Rt>,c12,c12,3 ; Write Rt to ICV_BPR1
```

Interrupt Controller Deactivate Virtual Interrupt Register

When interrupt priority drop is separated from interrupt deactivation, a write to the ICV_DIR register deactivates the specified virtual interrupt.

Usage constraints This register is write-only, and is only available at EL1.

Traps and enables If HSTR.T12 is set to 1, then write accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TC is set to 1, then write accesses to this register from EL1 are trapped to EL2.

If ICH_HCR.TDIR is set to 1, then write accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_DIR is a 32-bit register.

The following figure shows the ICV_DIR bit assignments.

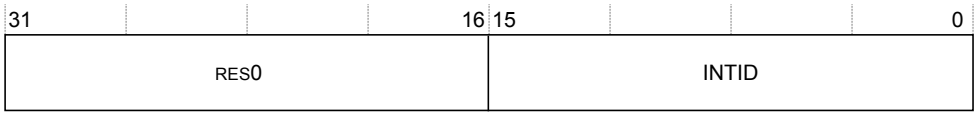

Figure 9-73 ICV_DIR bit assignments

The following table shows the ICV_DIR bit assignments.

Table 9-114 ICV_DIR bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15:0]	INTID	The INTID of the interrupt to be deactivated.

To access ICV_DIR:

```
MRC p15,0,<Rt>,c12,c11,1 ; Read ICV_DIR into Rt
MCR p15,0,<Rt>,c12,c11,1 ; Write Rt to ICV_DIR
```

Interrupt Controller Virtual Interrupt Priority Mask Register

The ICV_PMR register provides a virtual interrupt priority filter. Only virtual interrupts with higher priority than the value in this register are signaled to the core.

Usage constraints This register is read/write, and is only available at EL1.

Traps and enables If ICH_HCR.TC is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_PMR is a 32-bit register.

The following figure shows the ICV_PMR bit assignments.

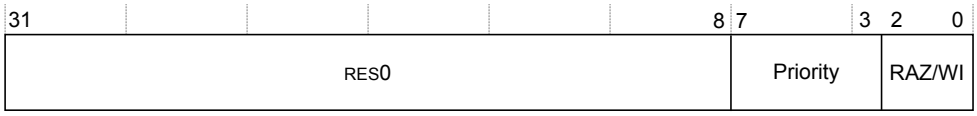

Figure 9-74 ICV_PMR bit assignments

The following table shows the ICV_PMR bit assignments.

Table 9-115 ICV_PMR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:3]	Priority	The priority mask level for the CPU interface. If the priority of an interrupt is higher than the value indicated by this field, the interface signals the interrupt to the core.
[2:0]	-	RAZ/WI.

To access ICV_PMR:

```
MRC p15,0,<Rt>,c4,c6,0 ; Read ICV_PMR into Rt
MCR p15,0,<Rt>,c4,c6,0 ; Write Rt to ICV_PMR
```

Interrupt Controller Virtual Running Priority Register

The ICV_RPR register indicates the running priority of the virtual CPU interface.

Usage constraints This register is read only, and is only available at EL1.

Traps and enables If HSTR.T12 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TC is set to 1, then read accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_RPR is a 32-bit register.

The following figure shows the ICV_RPR bit assignments.

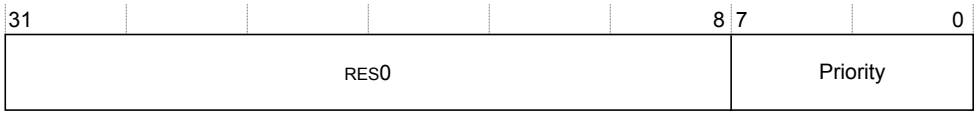

Figure 9-75 ICV_RPR bit assignments

The following table shows the ICV_RPR bit assignments.

Table 9-116 ICV_RPR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	Priority	The current running priority. Returns `0xFF` when ICV_AP0R0 and ICV_AP1R0 are both `0x0`. Otherwise returns the index in bits [7:3] of the lowest set bit from ICV_AP0R0 and ICV_AP1R0.

To access ICV_RPR:

```
MRC p15,0,<Rt>,c12,c11,3 ; Read ICV_RPR into Rt
MCR p15,0,<Rt>,c12,c11,3 ; Write Rt to ICV_RPR
```

Interrupt Controller Virtual Control Register

The ICV_CTLR register controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TC is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_CTLR is a 32-bit register.

The following figure shows the ICV_CTLR bit assignments.

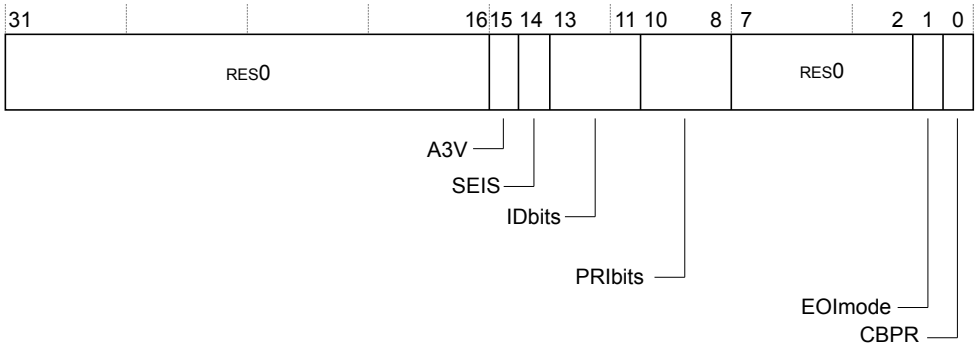

Figure 9-76 ICV_CTLR bit assignments

The following table shows the ICV_CTLR bit assignments.

Table 9-117 ICV_CTLR bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RES0.
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored: **0** The virtual CPU interface logic only supports zero values of Affinity 3 in SGI generation system registers.
[14]	SEIS	SEI support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs: **0** The virtual CPU interface logic does not support local generation of SEIs.
[13:11]	IDbits	Indicates the number of physical interrupt identifier bits supported. Read-only and writes are ignored: `0b000` 16 interrupt identifier bits are supported.
[10:8]	PRIbits	Indicates the number of priority bits implemented, minus one. Read-only and writes are ignored: `0b100` Five priority bits are supported.
[7:2]	-	Reserved, RES0.
[1]	EOImode	Controls whether a write to an End of Interrupt register also deactivates the interrupt.
[0]	CBPR	Controls whether ICV_BPR1 is used for interrupt preemption of both Group 0 and Group 1 interrupts.

To access ICV_CTLR:

```
MRC p15,0,<Rt>,c12,c12,4 ; Read ICV_CTLR into Rt
MCR p15,0,<Rt>,c12,c12,4 ; Write Rt to ICV_CTLR
```

Interrupt Controller Virtual Interrupt Group 0 Enable Register

The ICV_IGRPEN0 register controls if virtual Group 0 interrupts are enabled.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_IGRPEN0 is a 32-bit register.

The following figure shows the ICV_IGRPEN0 bit assignments.

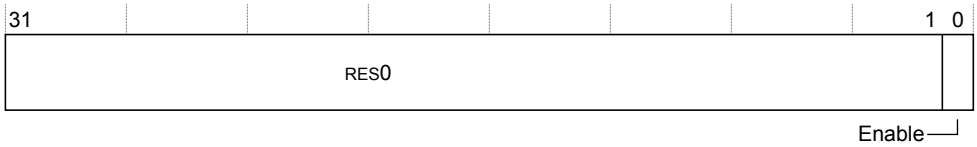

Figure 9-77 ICV_IGRPEN0 bit assignments

The following table shows the ICV_IGRPEN0 bit assignments.

Table 9-118 ICV_IGRPEN0 bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	Enable	Enables virtual Group 0 interrupts: `0` Group 0 interrupts are disabled. This is the reset value. `1` Group 0 interrupts are enabled.

To access ICV_IGRPEN0:

```
MRC p15,0,<Rt>,c12,c12,6 ; Read ICV_IGRPEN0 into Rt
MCR p15,0,<Rt>,c12,c12,6 ; Write Rt to ICV_IGRPEN0
```

Interrupt Controller Virtual Interrupt Group 1 Enable Register

The ICV_IGRPEN1 register controls if virtual Group 1 interrupts are enabled.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_IGRPEN1 is a 32-bit register.

The following figure shows the ICV_IGRPEN1 bit assignments.

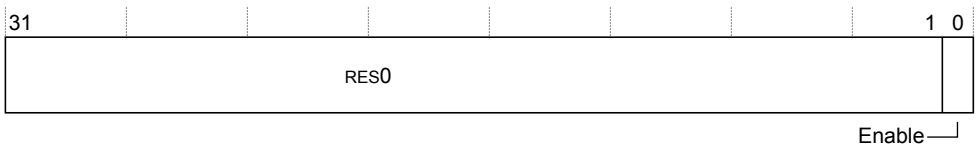

Figure 9-78 ICV_IGRPEN1 bit assignments

The following table shows the ICV_IGRPEN1 bit assignments.

Table 9-119 ICV_IGRPEN1 bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	Enable	Enables virtual Group 1 interrupts: 0 Group 1 interrupts are disabled. This is the reset value. 1 Group 1 interrupts are enabled.

To access ICV_IGRPEN1:

```
MRC p15,0,<Rt>,c12,c12,7 ; Read ICV_IGRPEN1 into Rt
MCR p15,0,<Rt>,c12,c12,7 ; Write Rt to ICV_IGRPEN1
```

Interrupt Controller Virtual Active Priorities Group 0 Register

The ICV_AP0R0 register provides information about virtual Group 0 active priorities. This register is an alias of ICH_AP0R0.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL0 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_AP0R0 is a 32-bit register.

The following figure shows the ICV_AP0R0 bit assignments.

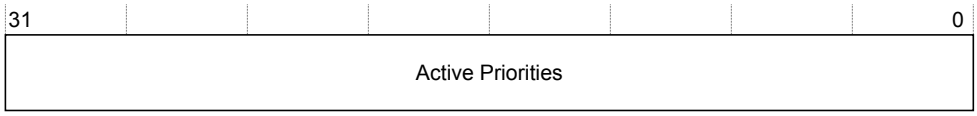

Figure 9-79 ICV_AP0R0 bit assignments

The following table shows the ICV_AP0R0 bit assignments.

Table 9-120 ICV_AP0R0 bit assignments

Bits	Name	Function
[n]	P<n>	P<n>, bit[n], for n = 0 to 31. Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: 0 There is no Group 0 interrupt active at the priority corresponding to that bit. 1 There is a Group 0 interrupt active at the priority corresponding to that bit.

To access ICV_AP0R0:

```
MRC p15,0,<Rt>,c12,c8,4 ; Read ICV_AP0R0 into Rt
MCR p15,0,<Rt>,c12,c8,4 ; Write Rt to ICV_AP0R0
```

Interrupt Controller Virtual Active Priorities Group 1 Register

The ICV_AP1R0 register provides information about virtual Group 1 active priorities. This register is an alias of ICH_AP1R0.

Usage constraints This register is read/write.

Traps and enables If HSTR.T12 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode.

If ICH_HCR.TALL1 is set to 1, then accesses to this register from EL1 are trapped to EL2.

Configurations This register is available in all build configurations.

Attributes ICV_AP1R0 is a 32-bit register.

The following figure shows the ICV_AP1R0 bit assignments.

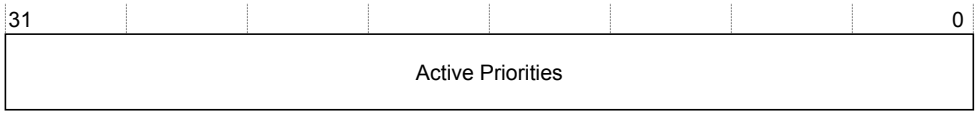

Figure 9-80 ICV_AP1R0 bit assignments

The following table shows the ICV_AP1R0 bit assignments.

Table 9-121 ICV_AP1R0 bit assignments

Bits	Name	Function
[n]	P<n>	P<n>, bit[n], for n = 0 to 31. Group 1 interrupt active priorities. Possible values of each bit are: `0` There is no Group 1 interrupt active at the priority corresponding to that bit. `1` There is a Group 1 interrupt active at the priority corresponding to that bit.

To access ICV_AP1R0:

```
MRC p15,0,<Rt>,c12,c9,0 ; Read ICV_AP1R0 into Rt
MCR p15,0,<Rt>,c12,c9,0 ; Write Rt to ICV_AP1R0
```

Chapter 10
Generic Timer

This chapter describes the Cortex-R52 processor implementation of the Arm Generic Timer.

It contains the following sections:

10.1 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 PPIs.
- Generation of event streams.

The Cortex-R52 processor Generic Timer is compliant with the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

This chapter describes only features that are specific to the Cortex-R52 processor implementation.

10.2 Generic Timer functional description

The Cortex-R52 processor provides a set of timer registers within each core of the cluster.

The timers are:

- An EL1 physical timer.
- An EL2 physical timer.
- A virtual timer.

The Cortex-R52 processor does not include the system counter. This resides in the SoC. The system counter value is distributed to the Cortex-R52 processor with a synchronous binary encoded 64-bit bus, **CNTVALUEB[63:0]**.

Because **CNTVALUEB** is generated from a system counter which typically operates at a slower frequency than the main processor **CLKIN**, the **CNTCLKEN** input is provided as a clock enable for the **CNTVALUEB** bus. This allows a multicycle path to be applied to the **CNTVALUEB[63:0]** bus. **CNTCLKEN** is registered inside the Cortex-R52 processor before being used as a clock enable for the **CNTVALUEB[63:0]** registers. The following figure shows the interface.

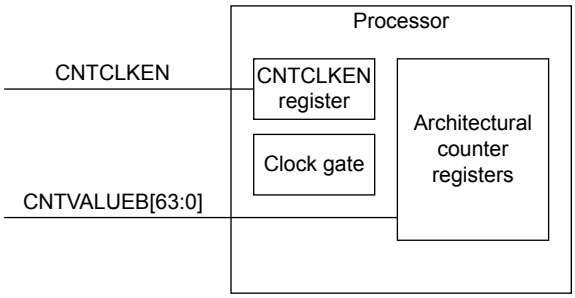

Figure 10-1 Architectural counter interface

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 **CLKIN** and must toggle at integer ratios of the processor **CLKIN**.

See *Chapter 4 Clocking and Resets* on page 4-198 for more information about **CNTCLKEN**.

Each timer provides an interrupt as a PPI to the core.

10.3 Generic Timer register summary

Within each core, a set of Generic Timer registers are allocated to the (coproc==`0b1111`) space.

This section contains the following subsection:

- *10.3.1 AArch32 Generic Timer register summary* on page 10-349.

10.3.1 AArch32 Generic Timer register summary

The following table shows the AArch32 Generic Timer registers.

The following registers are fully documented in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*. This implies that there are no Cortex-R52-specific details.

Table 10-1 AArch32 Generic Timer registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
CNTFRQ	c14	0	c0	0	UNK	32-bit	Counter-timer Frequency register
CNTPCT	-	0	c14	-	UNK	64-bit	Counter-timer Physical Count register
CNTKCTL	c14	0	c1	0	`0x00000000`	32-bit	Counter-timer Kernel Control register
CNTP_TVAL			c2	0	UNK	32-bit	Counter-timer Physical Timer TimerValue register
CNTP_CTL				1	-[bn]	32-bit	Counter-timer Physical Timer Control register
CNTV_TVAL			c3	0	UNK	32-bit	Counter-timer Virtual Timer TimerValue register
CNTV_CTL				1	-[bn]	32-bit	Counter-timer Virtual Timer Control register
CNTVCT	-	1	c14	-	UNK	64-bit	Counter-timer Virtual Count register
CNTP_CVAL	-	2			UNK	64-bit	Counter-timer Physical Timer CompareValue register
CNTV_CVAL	-	3			UNK	64-bit	Counter-timer Virtual Timer CompareValue register
CNTVOFF	-	4			UNK	64-bit	Counter-timer Virtual Offset register
CNTHCTL	c14	4	c1	0	_[bo]	32-bit	Counter-timer Hyp Control register
CNTHP_TVAL	-		c2	0	UNK	32-bit	Counter-timer Hyp Physical Timer TimerValue register
CNTHP_CTL	c14			1	-[bn]	32-bit	Counter-timer Hyp Physical Timer Control register
CNTHP_CVAL	-	6	c14	-	UNK	64-bit	Counter-timer Hyp Physical CompareValue register

[bn] The reset value for bit[0] is 0.
[bo] The reset value for bit[2] is 0 and for bits[1:0] is `0b11`.

Chapter 11
Debug

This chapter describes the Cortex-R52 processor debug registers and shows examples of how to use them.

It contains the following sections:

11.1 About Debug

The Cortex-R52 processor supports two methods of invasive debugging, external debug, and self-hosted debug. This section provides a brief introduction to these methods and outlines the main components that constitute a conventional invasive debug system.

This section contains the following subsections:

11.1.1 External debug

External debug is a conventional setup for debug, in which the debugger is external to the core that is being debugged. The debugger might be either hosted on another core within the Cortex-R52 processor, or running external to Cortex-R52. For example, the debugger might be hosted on a workstation connected using JTAG to a development system containing the Cortex-R52 processor.

External debug is useful for hardware bring-up of a Cortex-R52 processor-based system, that is, debugging during development when a system is first powered up and not all of the software functionality is available.

The following figure shows a typical external debug system.

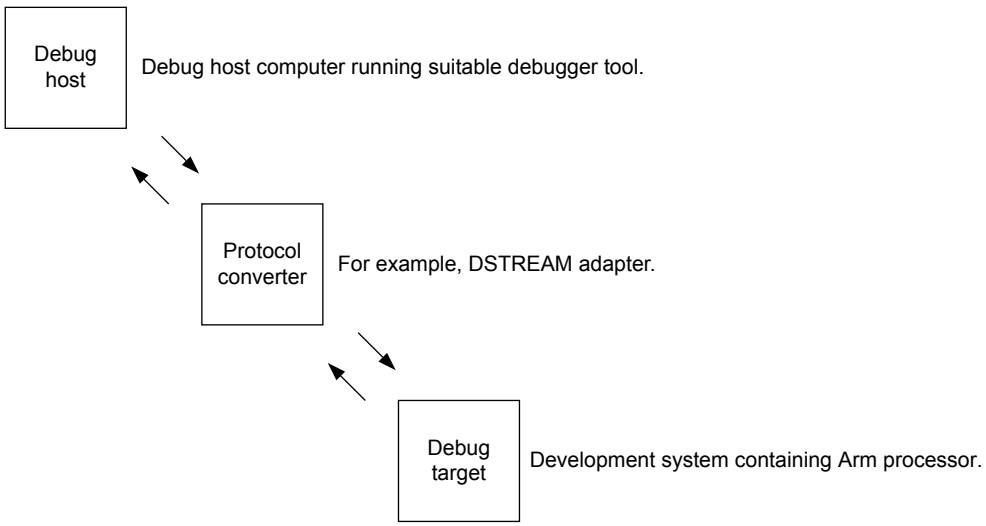

Figure 11-1 Typical external debug system

Note

In addition to the components shown in *Figure 11-1 Typical external debug system* on page 11-351, there is a debug unit that assists in debugging software on the processor.

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 a breakpoint at a certain location, or examining the contents of a memory address.

Protocol converter

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

Debug target

The debug target is the lowest level of the system. An example of a debug target is a development system with a test chip or a silicon part with a Cortex-R52 processor.

The debug target implements system support for the protocol converter to access the Cortex-R52 debug unit using the AMBA APB slave interface.

The Cortex-R52 debug unit

The processor debug unit assists in debugging software running on the processor.

The debug unit enables you to:

- Stop program execution.
- Examine and alter process and system control state.
- Examine and alter memory and input/output peripheral state.
- Restart the processor.

The Cortex-R52 processor presents a single debug-APB slave port and the debug registers for each core are accessible through that port. For the list of debug registers, see *11.4 System register descriptions* on page 11-359.

The ETM, CTI, PMU, and a ROM table to describe the registers are also available on this port. For a high-level memory map of this port, see *11.7 External debug interface* on page 11-384.

Related references

Chapter 12 Performance Monitor Unit on page 12-398
Chapter 13 Cross Trigger on page 13-429
Chapter 14 Embedded Trace Macrocell on page 14-447

11.1.2 Self-hosted debug

In self-hosted debug, the core being debugged within the Cortex-R52 processor hosts debug monitor software itself. Hardware watchpoints and breakpoints generate debug exceptions on debug events.

For more information, see *11.1.3 The debug model* on page 11-352 and *Watchpoint debug events* on page 11-353 for more information. These exceptions are handled by the debug monitor that typically resides alongside the operating system kernel or the hypervisor.

Self-hosted debug is useful in situations in which the Cortex-R52 processor has been deployed in a developed system, where there is no direct access to the Cortex-R52 processor debug hardware. Self-hosted debug supports:

- Task debugging.
- OS and kernel debugging.
- Hypervisor debugging[bp]

For details on self-hosted debug, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

11.1.3 The debug model

The debug logic of the processor is responsible for generating debug events. Debug events include events such as a breakpoint unit matching the address of an instruction against the address stored in its registers.

The Cortex-R52 processor responds to a debug event in one of the following ways:

[bp] Self-hosted debug support for Hypervisor code is limited to software breakpoint instructions.

- Enters a special Debug state.
- Takes a debug exception.
- Pends the debug event, for action later.
- Ignores the debug event.

Debug exceptions are the basis of self-hosted debug. Debug state is the basis of the external debug.

The conversion of debug events into exceptions or entry into Debug state depends on the configuration of the debug logic and the type of debug event. For example, in the case of *breakpoint instructions* (`BKPT`), the processor must assume the instruction has replaced an actual program instruction, and always takes an exception.

A debug event can be either:

- A software debug event.
- An external halting debug event.

When the Cortex-R52 processor is programmed for external debug, also known as halting debug, a software debug event causes entry into a special Debug state. External halting debug events always cause entry into the Cortex-R52 Debug state regardless of how the Cortex-R52 processor is programmed. In Debug state:

- The processor does not fetch instructions from memory, but from a special instruction transfer register.
- Data transfer registers are used to move register and memory contents between the debug host and the debug target.

The debug event that causes this entry into Debug state is known as a halting debug event.

For more information on the debug model, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Watchpoint debug events

In the Cortex-R52 processor, watchpoint debug events are always synchronous. Store exclusive instructions generate a watchpoint debug event even when the check for the control of the exclusive monitor fails.

Under normal operating conditions, the following do not generate watchpoint exceptions:

- Instruction cache maintenance instructions.
- Address translation instructions.
- Preload instructions.
- All data cache maintenance instructions, except `DCIMVAC`.

For watchpoint debug events, 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.

Debug OS Lock

Debug OS Lock is set by the powerup reset, **nCPUPORESETx**.

For normal behavior of debug events and debug register accesses, Debug OS Lock must be cleared. For more information, see the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

11.2 Debug register interfaces

This section describes the debug architecture and debug events of the processor.

The processor implements the Armv8-R Debug architecture and debug events as described in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

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.

This section contains the following subsections:

11.2.1 Processor interfaces

System register access allows the processor to directly access certain debug registers. The external debug interface allows both external and self-hosted debug agents to access debug registers.

Access to the debug registers is partitioned as follows:

Debug registers
This interface is System register based and memory-mapped. You can access the debug register map using the APB slave port.

Performance monitor
This interface is System register based and memory-mapped. You can access the performance monitor registers using the APB slave port.

Trace registers
This interface is memory-mapped. You can access the trace registers using the APB slave port.

Related references

11.3 System register summary on page 11-356
11.5 Memory-mapped register summary on page 11-363
Chapter 12 Performance Monitor Unit on page 12-398
Chapter 14 Embedded Trace Macrocell on page 14-447

11.2.2 Breakpoints and watchpoints

Each core in the processor supports eight hardware breakpoint register pairs, eight hardware watchpoint register pairs, and a standard *Debug Communication Channel* (DCC).

Two of the breakpoint register pairs can be configured to match against one of both of context ID and VMID. Each watchpoint can be linked to one of these two breakpoints to enable a memory request to be trapped in a given process context.

11.2.3 Effects of resets on debug registers

Each core has the following reset signals that affect the debug registers:

nCPUPORESETx
These primary, Cold reset signals initialize the core logic including debug, the performance monitor logic, and the ETM.

nCORERESETx
These primary, Warm reset signals initialize the core logic including some of the debug and performance monitor logic.

nPRESETDBG

This is an external debug reset signal that initializes the shared debug APB, CTI, and *Cross Trigger Matrix* (CTM) logic. This signal has no impact on the functionality for the rest of the processor.

11.2.4 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 that is specific to each register and the type of access to the register is not described in this document. For a detailed description of these features and their effects on the registers, see the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

The register descriptions provided in this section describe whether each register is read/write or read-only.

11.3 System register summary

The following table summarizes the debug control registers that are accessible from the internal (coproc==0b1110) interface.

These registers are accessed by the MCR, MRC instructions. See the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

The table also shows the offset address for the registers that are also accessible from the memory-mapped interface and the external debug interface. See *11.5 Memory-mapped register summary* on page 11-363 for a complete list of registers accessible from the memory-mapped and the external debug interface. With the exception of ID and configuration values in some read-only registers, all these registers are fully described in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*. This implies that there are no Cortex-R52-specific details.

Table 11-1 AArch32 debug register summary

Offset	CRn	op1	CRm	op2	Name	Type	Width	Description
-	c0	0	c0	0	DBGDIDR	RO	32-bit	*11.4.1 Debug ID Register* on page 11-359
-				2	DBGDTRRXext	RW	32-bit	Debug Data Transfer Register, Receive, External View
0x400				4	DBGBVR0	RW	32-bit	Debug Breakpoint Value Register 0
0x408				5	DBGBCR0	RW	32-bit	Debug Breakpoint Control Register 0
0x800				6	DBGWVR0	RW	32-bit	Debug Watchpoint Value Register 0
0x808				7	DBGWCR0	RW	32-bit	Debug Watchpoint Control Registers 0
-	c0	7	c0	0	JIDR	RO	32-bit	Legacy Jazelle ID Register
-	c0	0	c1	0	DBGDSCRint	RO	32-bit	Debug Status and Control Register, Internal View
0x410				4	DBGBVR1	RW	32-bit	Debug Breakpoint Value Register 1
0x418				5	DBGBCR1	RW	32-bit	Debug Breakpoint Control Register 1
0x810				6	DBGWVR1	RW	32-bit	Debug Watchpoint Value Register 1
0x818				7	DBGWCR1	RW	32-bit	Debug Watchpoint Control Registers 1
-	c0	0	c2	0	DBGDCCINT	RW	32-bit	Debug Comms Channel Interrupt Enable Register
-				2	DBGDSCRext	RW	32-bit	Debug Status and Control Register, External View
0x420				4	DBGBVR2	RW	32-bit	Debug Breakpoint Value Register 2
0x428				5	DBGBCR2	RW	32-bit	Debug Breakpoint Control Register 2
0x820				6	DBGWVR2	RW	32-bit	Debug Watchpoint Value Register 2
0x828				7	DBGWCR2	RW	32-bit	Debug Watchpoint Control Registers 2
-	c0	0	c3	2	DBGDTRTXext	RW	32-bit	Debug Data Transfer Register, Transmit, External View
0x430				4	DBGBVR3	RW	32-bit	Debug Breakpoint Value Register 3
0x438				5	DBGBCR3	RW	32-bit	Debug Breakpoint Control Register 3
0x830				6	DBGWVR3	RW	32-bit	Debug Watchpoint Value Register 3
0x828				7	DBGWCR3	RW	32-bit	Debug Watchpoint Control Registers 3

Table 11-1 AArch32 debug register summary (continued)

Offset	CRn	op1	CRm	op2	Name	Type	Width	Description
0x440	c0	0	c4	4	DBGBVR4	RW	32-bit	Debug Breakpoint Value Register 4
0x448				5	DBGBCR4	RW	32-bit	Debug Breakpoint Control Register 4
0x830				6	DBGWVR4	RW	32-bit	Debug Watchpoint Value Register 4
0x828				7	DBGWCR4	RW	32-bit	Debug Watchpoint Control Register 4
0x08C	c0	0	c5	0	DBGDTRTXint	WO	32-bit	Debug Data Transfer Register, Transmit, Internal View
0x080				0	DBGDTRRXint	RO	32-bit	Debug Data Transfer Register, Receive, Internal View
0x450				4	DBGBVR5	RW	32-bit	Debug Breakpoint Value Register 5
0x458				5	DBGBCR5	RW	32-bit	Debug Breakpoint Control Register 5
0x830				6	DBGWVR5	RW	32-bit	Debug Watchpoint Value Register 5
0x828				7	DBGWCR5	RW	32-bit	Debug Watchpoint Control Register 5
-	c0	0	c6	0	DBGWFAR	RW	32-bit	Debug Watchpoint Fault Address Register[bq].
0x098				2	DBGOSECCR	RW	32-bit	Debug OS Lock Exception Catch Control Register
0x450				4	DBGBVR6	RW	32-bit	Debug Breakpoint Value Register 6
0x458				5	DBGBCR6	RW	32-bit	Debug Breakpoint Control Register 6
0x830				6	DBGWVR6	RW	32-bit	Debug Watchpoint Value Register 6
0x828				7	DBGWCR6	RW	32-bit	Debug Watchpoint Control Register 6
-	c0	0	c7	0	DBGVCR	RW	32-bit	Debug Vector Catch Register
0x450				4	DBGBVR7	RW	32-bit	Debug Breakpoint Value Register 7
0x458				5	DBGBCR7	RW	32-bit	Debug Breakpoint Control Register 7
0x830				6	DBGWVR7	RW	32-bit	Debug Watchpoint Value Register 7
0x828				7	DBGWCR7	RW	32-bit	Debug Watchpoint Control Register 7
-	c1	0	c0	0	DBGDRAR	RO	32-bit	Debug ROM Address Register
0x300				4	DBGOSLAR	WO	32-bit	Debug OS Lock Access Register
	c1	7	c0	0	JOSCR	RW	32bit	Legacy Jazelle OS Control Register
-	c1	0	c1	4	DBGOSLSR	RO	32-bit	Debug OS Lock Status Register
-			c3	4	DBGOSDLR	RW	32-bit	Debug OS Double Lock Register
0x310			c4	4	DBGPRCR	RW	32-bit	Debug Power/Reset Control Register
0x444			c6	1	DBGBXVR6	RW	32-bit	Debug Breakpoint Extended Value Register 6
0x444			c7	1	DBGBXVR7	RW	32-bit	Debug Breakpoint Extended Value Register 7
-	c2	0	c0	0	DBGDSAR	RO	32-bit	Debug Self Address Register
-					JMCR	RW	32-bit	Legacy Jazelle Main Configuration Register.

[bq] Previously returned information about the address of the instruction that accessed a watchpoint address. This register is now deprecated and is RES0.

Table 11-1 AArch32 debug register summary (continued)

Offset	CRn	op1	CRm	op2	Name	Type	Width	Description
-	c7	0	c0	7	DBGDEVID2	RO	32-bit	Debug Device ID Register 2
-			c1		DBGDEVID1	RO	32-bit	*11.4.3 Debug Device ID Register 1* on page 11-361
-			c2		DBGDEVID	RO	32-bit	*11.4.2 Debug Device ID Register* on page 11-360
0xFA0			c8	6	DBGCLAIMSET	RW	32-bit	Debug Claim Tag Set Register
0xFA4			c9		DBGCLAIMCLR	RW	32-bit	Debug Claim Tag Clear Register
0xFB8			c14		DBGAUTHSTATUS	RO	32-bit	Debug Authentication Status Register

11.4 System register descriptions

This section describes some of the debug registers available through the internal (coproc==`0b1110`) interface, where these have Cortex-R52-specific values.

For a list of all the registers available through the internal interface, see *11.3 System register summary* on page 11-356.

This section contains the following subsections:

11.4.1 Debug ID Register

The DBGDIDR specifies the version of the Debug architecture that is implemented and some features of the debug implementation.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables If DBGDSCRext.UDCCdis is set to 1, then read accesses to this register from EL0 are trapped to UNDEFINED.

If HDCR.TDA is set to 1, then read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

Configurations Available in all processor build configurations.

Attributes DBGDIDR is a 32-bit register.

The following figure shows the DBGDIDR bit assignments.

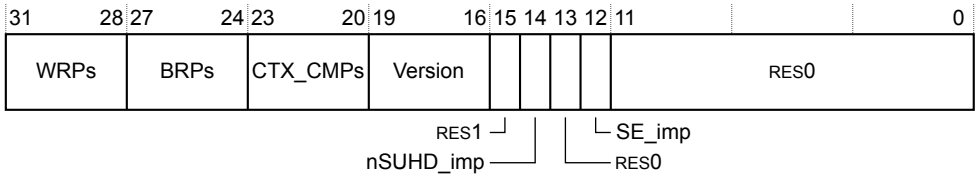

Figure 11-2 DBGDIDR bit assignments

The following table shows the DBGDIDR bit assignments.

Table 11-2 DBGDIDR bit assignments

Bits	Name	Function
[31:28]	WRPs	The number of *Watchpoint Register Pairs* (WRPs) implemented. The number of implemented WRPs is one more than the value of this field. The value is: `0x7` The processor implements 8 WRPs.
[27:24]	BRPs	The number of *Breakpoint Register Pairs* (BRPs) implemented. The number of implemented BRPs is one more than the value of this field. The value is: `0x7` The processor implements 8 BRPs.

Table 11-2 DBGDIDR bit assignments (continued)

Bits	Name	Function
[23:20]	CTX_CMPs	The number of BRPs that can be used for Context matching. This is one more than the value of this field. The value is: `0x1` The processor implements two Context matching breakpoints, breakpoints 6 and 7.
[19:16]	Version	The Debug architecture version. `0x6` The processor implements Armv8 Debug architecture.
[15]	-	Reserved, RES1.
[14]	nSUHD_imp	Secure User Halting Debug not implemented bit. The value is: `0` The processor does not implement Secure User Halting Debug.
[13]	-	Reserved, RES0.
[12]	SE_imp	Security Extensions implemented bit. The value is: `0` The processor does not implement Security Extensions.
[11:0]	-	Reserved, RES0.

To access the DBGDIDR:

```
MRC p14, 0, <Rt>, c0, c0, 0; Read DBGDIDR into Rt
```

11.4.2 Debug Device ID Register

The DBGDEVID specifies the version of the Debug architecture that is implemented and some features of the debug implementation.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables If HDCR.TDA is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

Configurations Available in all processor build configurations.

Attributes DBGDEVID is a 32-bit register.

The following figure shows the DBGDEVID bit assignments.

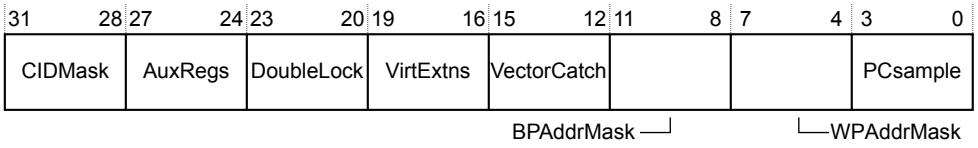

Figure 11-3 DBGDEVID bit assignments

The following table shows the DBGDEVID bit assignments.

Table 11-3 DBGDEVID bit assignments

Bits	Name	Function
[31:28]	CIDMask	Specifies 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	Specifies support for the Debug External Auxiliary Control Register. This value is: `0x0` The processor does not support Debug External Auxiliary Control Register.
[23:20]	DoubleLock	Specifies support for the Debug OS Double Lock Register. This value is: `0x1` The processor supports Debug OS Double Lock Register.
[19:16]	VirExtns	Specifies whether EL2 is implemented. This value is: `0x1` The processor implements EL2.
[15:12]	VectorCatch	Defines the form of the vector catch event implemented. This value is: `0x0` The processor implements address matching form of vector catch.
[11:8]	BPAddrMask	Indicates the level of support for the instruction address matching breakpoint masking capability. This value is: `0xF` Breakpoint address masking not implemented. DBGBCRn[28:24] are RES0.
[7:4]	WPAddrMask	Indicates the level of support for the data address matching watchpoint masking capability. This value is: `0x1` Watchpoint address mask implemented.
[3:0]	PCsample	Indicates the level of support for Program Counter sampling using debug registers 40 and 43. This value is: `0x3` EDPCSR, EDCIDSR, and EDVIDSR are implemented as debug registers 40, 41, and 42.

To access the DBGDEVID:

```
MRC p14, 0, <Rt>, c7, c2, 7; Read DBGDEVID into Rt
```

11.4.3 Debug Device ID Register 1

The DBGDEVID1 adds to the information given by the DBGDIDR by describing other features of the debug implementation.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

If HDCR.TDA is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

Configurations

Available in all processor build configurations.

Attributes

DBGDEVID1 is a 32-bit register.

The following figure shows the DBGDEVID1 bit assignments.

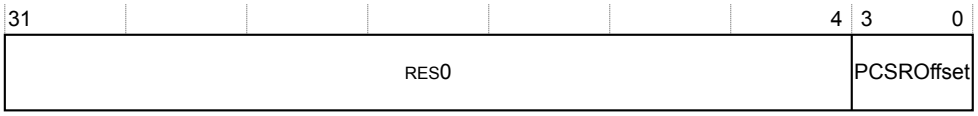

Figure 11-4 DBGDEVID1 bit assignments

The following table shows the DBGDEVID1 bit assignments.

Table 11-4 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 samples have no offset applied and do not sample the instruction set state.

To access the DBGDEVID1:

```
MRC p14, 0, <Rt>, c7, c1, 7 ; Read DBGDEVID1 into Rt
```

11.5 Memory-mapped register summary

The Cortex-R52 processor includes support for memory-mapped access to the external debug interface. This access capability is referred to as the memory-mapped external debug interface.

Note

The value of **PADDRDBG31** and the CoreSight Software lock restrict write accesses to the debug registers through the memory-mapped external debug interface. When **PADDRDBG31** is driven LOW, the CoreSight software lock must be used to unlock write permissions for the debug registers. When **PADDRDBG31** is driven HIGH, the software lock does not restrict the write access permissions of the debug registers. The software lock is controlled with the Lock Access register.

The following table shows the offset address for the registers that are accessible from the memory-mapped external debug interface. It also shows the CRn, op1, CRm, and op2 for the registers that are also accessible through the internal (coproc==`0b1110`) interface.

Table 11-5 Memory-mapped debug register summary

Offset	CRn	op1	CRm	op2	Name	Type	Width	Description
`0x020`	-	-	-	-	EDESR	RW	32-bit	External Debug Event Status Register
`0x024`					EDECR	RW	32-bit	External Debug Execution Control Register
`0x030`					EDWARlo	RO	32-bit	External Debug Watchpoint Address Register, low word
`0x034`					EDWARhi	RO	32-bit	External Debug Watchpoint Address Register, high word
`0x034`					DBGDTRRX_EL0	RO	32bit	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	RO	32-bit	Debug Data Transfer Register, Transmit
`0x090`					EDRCR	WO	32-bit	External Debug Reserve Control Register
`0x098`					EDECCR	RW	32-bit	External Debug Exception Catch Control Register
`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
`0x300`	c1	0	c0	4	DBGOSLAR	WO	32-bit	Debug OS Lock Access Register
`0x310`	-	-	-	-	EDPRCR	RW	32-bit	External Debug Power/Reset Control Register
`0x314`					EDPRSR	RO	32-bit	External Debug Processor Status Register

Table 11-5 Memory-mapped debug register summary (continued)

Offset	CRn	op1	CRm	op2	Name	Type	Width	Description
0x400	c0	0	c0	4	DBGBVR0	RW	32-bit	Debug Breakpoint Value Register 0
0x408				5	DBGBCR0	RW	32-bit	Debug Breakpoint Control Register 0
0x410			c1	4	DBGBVR1	RW	32-bit	Debug Breakpoint Value Register 1
0x418				5	DBGBCR1	RW	32-bit	Debug Breakpoint Control Register 1
0x420			c2	4	DBGBVR2	RW	32-bit	Debug Breakpoint Value Register 2
0x428				5	DBGBCR2	RW	32-bit	Debug Breakpoint Control Register 2
0x430			c3	4	DBGBVR3	RW	32-bit	Debug Breakpoint Value Register 3
0x438				5	DBGBCR3	RW	32-bit	Debug Breakpoint Control Register 3
0x440			c4	4	DBGBVR4	RW	32-bit	Debug Breakpoint Value Register 4
0x448				5	DBGBCR4	RW	32-bit	Debug Breakpoint Control Register 4
0x450			c5	4	DBGBVR5	RW	32-bit	Debug Breakpoint Value Register 5
0x458				5	DBGBCR5	RW	32-bit	Debug Breakpoint Control Register 5
0x460			c6	4	DBGBVR6	RW	32-bit	Debug Breakpoint Value Register 6
0x464				1	DBGBVR6	RW	32-bit	Debug Breakpoint Extended Value Register 6
0x468				5	DBGBCR6	RW	32-bit	Debug Breakpoint Control Register 6
0x470			c7	4	DBGBVR7	RW	32-bit	Debug Breakpoint Value Register 7
0x474				1	DBGBVR7	RW	32-bit	Debug Breakpoint Extended Value Register 7
0x478				5	DBGBCR7	RW	32-bit	Debug Breakpoint Control Register 7
0x800	c0	0	c0	6	DBGWVR0	RW	32-bit	Debug Watchpoint Value Register 0
0x808				7	DBGWCR0	RW	32-bit	Debug Watchpoint Control Register 0
0x810			c1	6	DBGWVR1	RW	32-bit	Debug Watchpoint Value Register 1
0x818				7	DBGWCR1	RW	32-bit	Debug Watchpoint Control Register 1
0x820			c2	6	DBGWVR2	RW	32-bit	Debug Watchpoint Value Register 2
0x828				7	DBGWCR2	RW	32-bit	Debug Watchpoint Control Register 2
0x830			c3	6	DBGWVR3	RW	32-bit	Debug Watchpoint Value Register 3
0x838				7	DBGWCR3	RW	32-bit	Debug Watchpoint Control Register 3
0x840			c4	6	DBGWVR4	RW	32-bit	Debug Watchpoint Value Register 4
0x848				7	DBGWCR4	RW	32-bit	Debug Watchpoint Control Register 4
0x850			c5	6	DBGWVR5	RW	32-bit	Debug Watchpoint Value Register 5
0x858				7	DBGWCR5	RW	32-bit	Debug Watchpoint Control Register 5
0x860			c6	6	DBGWVR6	RW	32-bit	Debug Watchpoint Value Register 6
0x868				7	DBGWCR6	RW	32-bit	Debug Watchpoint Control Register 6
0x870			c7	6	DBGWVR7	RW	32-bit	Debug Watchpoint Value Register 7
0x878				7	DBGWCR7	RW	32-bit	Debug Watchpoint Control Register 7

Table 11-5 Memory-mapped debug register summary (continued)

Offset	CRn	op1	CRm	op2	Name	Type	Width	Description
0xC00	-	-	-	-	EDCCR	RW	32-bit	*11.6.1 External Debug Calibration Control Register* on page 11-367
0xD00	-	-	-	-	MIDR	RO	32-bit	*3.3.69 Main ID Register* on page 3-156
0xD20					EDPFR[63:32]	RO	32-bit	*11.6.8 External Debug Processor Feature Register* on page 11-381
0xD24					EDPFR[31:0]	RO	32-bit	*11.6.8 External Debug Processor Feature Register* on page 11-381
0xD28					EDDFR[63:32]	RO	32-bit	*11.6.9 External Debug Feature Register* on page 11-382
0xD2C					EDDFR[31:0]	RO	32-bit	*11.6.9 External Debug Feature Register* on page 11-382
0xD60					EDAA32PFR[63:32]	RO	32-bit	*11.6.7 External Debug AArch32 Processor Feature Register* on page 11-379
0xD64					EDAA32PFR[31:0]	RO	32-bit	*11.6.7 External Debug AArch32 Processor Feature Register* on page 11-379
0xFA0	c7	0	c8	6	DBGCLAIMSET	RW	32-bit	Debug Claim Tag Set Register
0xFA4			c9		DBGCLAIMCLR	RW	32-bit	Debug Claim Tag Clear Register
0xFB8			c14		DBGAUTHSTATUS	RO	32-bit	Debug Authentication Status Register
0xFA8	-	-	-	-	EDDEVAFF0	RO	32-bit	*3.3.78 Multiprocessor Affinity Register* on page 3-167
0xFAC					EDDEVAFF1	RO	32-bit	External Debug Device Affinity Register 1. [br]
0xFB0					EDLAR	WO	32-bit	External Debug Lock Access Register
0xFB4					EDLSR	RO	32-bit	External Debug Lock Status Register
0xFBC					EDDEVARCH	RO	32-bit	External Debug Device Architecture Register
0xFC0					EDDEVID2	RO	32-bit	External Debug Device ID Register 2
0xFC4					EDDEVID1	RO	32-bit	*11.6.4 External Debug Device ID Register 1* on page 11-370
0xFC8					EDDEVID	RO	32-bit	*11.6.3 External Debug Device ID Register 0* on page 11-369
0xFCC					EDDEVTYPE	RO	32-bit	External Debug Device Type Register

[br] This register is RES0.

Table 11-5 Memory-mapped debug register summary (continued)

Offset	CRn	op1	CRm	op2	Name	Type	Width	Description
0xFD0	-	-	-	-	EDPIDR4	RO	32-bit	*External Debug Peripheral Identification Register 4* on page 11-375
0xFE0					EDPIDR0	RO	32-bit	*External Debug Peripheral Identification Register 0* on page 11-372
0xFE4					EDPIDR1	RO	32-bit	*External Debug Peripheral Identification Register 1* on page 11-372
0xFE8					EDPIDR2	RO	32-bit	*External Debug Peripheral Identification Register 2* on page 11-373
0xFEC					EDPIDR3	RO	32-bit	*External Debug Peripheral Identification Register 3* on page 11-374
0xFF0					EDCIDR0	RO	32-bit	*External Debug Component Identification Register 0* on page 11-376
0xFF4					EDCIDR1	RO	32-bit	*External Debug Component Identification Register 1* on page 11-377
0xFF8					EDCIDR2	RO	32-bit	*External Debug Component Identification Register 2* on page 11-378
0xFFC					EDCIDR3	RO	32-bit	*External Debug Component Identification Register 3* on page 11-379

11.6 Memory-mapped register descriptions

This section describes some of the Cortex-R52 processor memory-mapped debug registers, where these are Cortex-R52-specific or have Cortex-R52-specific values.

For those registers not described in this chapter, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

For a list of all the registers available through the external debug interface, see *11.5 Memory-mapped register summary* on page 11-363.

This section contains the following subsections:

11.6.1 External Debug Calibration Control Register

The EDCCR register is used to invalidate data, read from the Flash or AXIM interface, that has been cached in the Cortex-R52 data cache.

Usage constraints This register is accessible through the external debug interface, and is read/write. For a summary of the calibration conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355.

Traps and enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes EDCCR is a 32-bit register.

The following figure shows the EDCCR bit assignments.

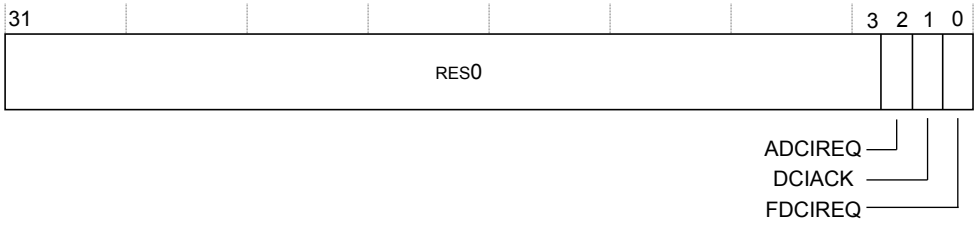

Figure 11-5 EDCCR bit assignments

The following table shows the EDCCR bit assignments.

Table 11-6 EDCCR bit assignments

Bits	Name	Function
[31:3]	-	Reserved, RES0.
[2]	ADCIREQ	For writes to this register: 0 Enables AXIM cache allocation 1 Invalidate cached data from the AXIM interface. Reads from this register is RES0.
[1]	DCIACK	Indicates when cache invalidation has completed. This bit is RO. 1 Cache invalidation is complete. 0 Cache is not yet invalidated. Writes are ignored.
[0]	FDCIREQ	For writes to this register: 0 Enables Flash cache allocation. 1 Invalidate cached data from the Flash interface. Reads from this register indicate the state: 0 Cache allocation for AXIM and Flash is enabled. 1 Data cache invalidation for AXIM, Flash, or both has been requested. New accesses are not allocated for the requested type.

EDCCR can be accessed through the external debug interface:

Table 11-7 EDCCR access information

Component	Offset
Debug	0xC00

For more information on data cache invalidation, see *7.3.2 Data cache invalidation* on page 7-223.

11.6.2 External Debug Reserve Control Register

The EDRCR is used to cancel bus requests and clear sticky bits in the EDSCR.

Usage constraints This register is accessible through the external debug interface, and is write-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes EDRCR is a 32-bit register.

The following figure shows the EDRCR bit assignments.

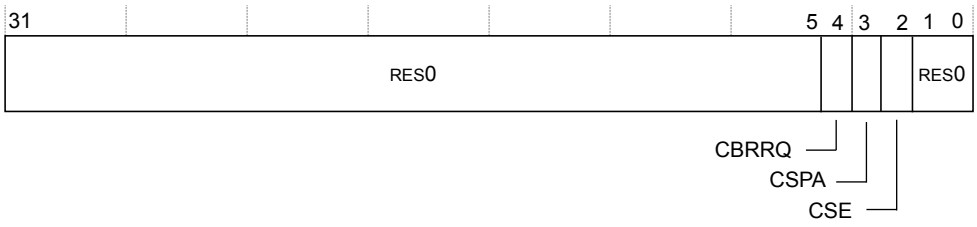

Figure 11-6 EDRCR bit assignments

The following table shows the EDRCR bit assignments.

Table 11-8 EDRCR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0.
[4]	CBRRQ	Allow imprecise entry to Debug state. The actions on writing to this bit are: 0 No action. 1 Allow imprecise entry to Debug state, for example by canceling pending bus accesses. Setting this bit to 1 allows a debugger to request imprecise entry to Debug state. An External Debug Request debug event must be pending before the debugger sets this bit to 1.
[3]	CSPA	Clear Sticky Pipeline Advance. This bit is used to clear the EDSCR.PipeAdv bit to 0. The possible values are: 0 No action. 1 Clear the EDSCR.PipeAdv bit to 0.
[2]	CSE	Clear Sticky Error. Used to clear the EDSCR cumulative error bits to 0. The possible values are: 0 No action. 1 Clear the EDSCR.{TXU, RXO, ERR} bits, and, if the processor is in Debug state, the EDSCR.ITO bit, to 0.
[1:0]	-	Reserved, RES0.

EDRCR can be accessed through the external debug interface:

Table 11-9 EDRCR access information

Component	Offset
Debug	0x090

11.6.3 External Debug Device ID Register 0

The EDDEVID provides extra information for external debuggers about features of the debug implementation.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

EDDEVID is a 32-bit register.

The following figure shows the EDDEVID bit assignments.

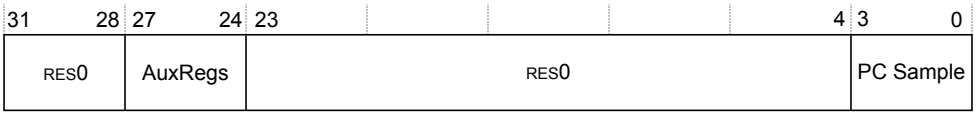

Figure 11-7 EDDEVID bit assignments

The following table shows the EDDEVID bit assignments.

Table 11-10 EDDEVID bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RES0.
[27:24]	AuxRegs	Indicates support for auxiliary registers. The possible values are: `0x1` External Debug Auxiliary Control Register, EDACR, is implemented.
[23:4]	-	Reserved, RES0.
[3:0]	PC Sample	Indicates the level of sample-based profiling support using external debug registers 40 through 43. The value is: `0x3` EDPCSR, EDCIDSR, and EDVIDSR are implemented.

EDDEVID can be accessed through the external debug interface:

Table 11-11 EDDEVID access information

Component	Offset
Debug	`0xFC8`

11.6.4 External Debug Device ID Register 1

The EDDEVID1 provides extra information for external debuggers about features of the debug implementation.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

The EDDEVID1 is in the Debug power domain, and is available in all processor build configurations.

Attributes

EDDEVID1 is a 32-bit register.

The following figure shows the EDDEVID1 bit assignments.

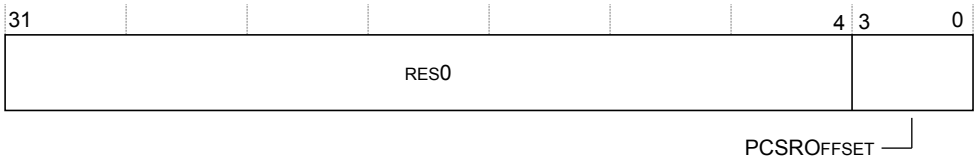

Figure 11-8 EDDEVID1 bit assignments

The following table shows the EDDEVID1 bit assignments.

Table 11-12 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. The value is: `0x2` EDPCSR samples have no offset applied and do not sample the instruction set state.

EDDEVID1 can be accessed through the external debug interface:

Table 11-13 EDDEVID1 access information

Component	Offset
Debug	`0xFC4`

11.6.5 External Debug Peripheral Identification Registers

The External Debug Peripheral Identification Registers provide information to identify an external debug component.

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*. For the Cortex-R52 processor, there is a set of five registers as listed, in address order, in the following table.

Table 11-14 Summary of the External Debug Peripheral Identification Registers

Register	Value	Offset
EDPIDR4	`0x04`	`0xFD0`
EDPIDR0	`0x13`	`0xFE0`
EDPIDR1	`0xBD`	`0xFE4`
EDPIDR2	`0x3B`	`0xFE8`
EDPIDR3	`0x00`	`0xFEC`

Only bits[7:0] of each External Debug Peripheral ID Register are used, with bits[31:8] *RAZ*.

The External Debug Peripheral ID registers are:

- *External Debug Peripheral Identification Register 0* on page 11-372.
- *External Debug Peripheral Identification Register 1* on page 11-372.
- *External Debug Peripheral Identification Register 2* on page 11-373.
- *External Debug Peripheral Identification Register 3* on page 11-374.
- *External Debug Peripheral Identification Register 4* on page 11-375.

External Debug Peripheral Identification Register 0

The EDPIDR0 provides information to identify an external debug component.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

EDPIDR0 is a 32-bit register.

The following figure shows the EDPIDR0 bit assignments.

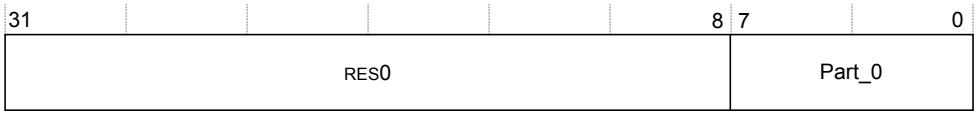

Figure 11-9 EDPIDR0 bit assignments

The following table shows the EDPIDR0 bit assignments.

Table 11-15 EDPIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	Part_0	`0x13` Least significant byte of the debug part number.

EDPIDR0 can be accessed through the external debug interface:

Table 11-16 EDPIDR0 access information

Component	Offset
Debug	`0xFE0`

External Debug Peripheral Identification Register 1

The EDPIDR1 provides information to identify an external debug component.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

EDPIDR1 is a 32-bit register.

The following figure shows the EDPIDR1 bit assignments.

Figure 11-10 EDPIDR1 bit assignments

The following table shows the EDPIDR1 bit assignments.

Table 11-17 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.

EDPIDR1 can be accessed through the external debug interface:

Table 11-18 EDPIDR1 access information

Component	Offset
Debug	`0xFE4`

External Debug Peripheral Identification Register 2

The EDPIDR2 provides information to identify an external debug component.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

EDPIDR2 is a 32-bit register.

The following figure shows the EDPIDR2 bit assignments.

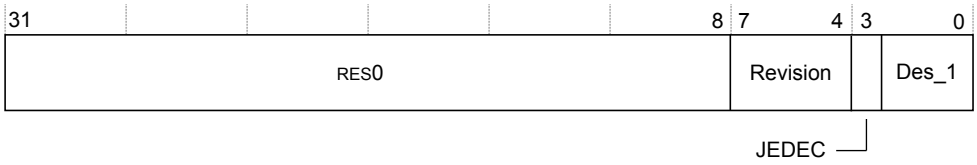

Figure 11-11 EDPIDR2 bit assignments

The following table shows the EDPIDR2 bit assignments.

Table 11-19 EDPIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Revision	`0x4`
[3]	JEDEC	`0b1` RAO. Indicates a JEP106 identity code is used.
[2:0]	DES_1	`0b011` Arm Limited. This is the most significant nibble of JEP106 ID code.

EDPIDR2 can be accessed through the external debug interface:

Table 11-20 EDPIDR2 access information

Component	Offset
Debug	`0xFE8`

External Debug Peripheral Identification Register 3

The EDPIDR3 provides information to identify an external debug component.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

EDPIDR3 is a 32-bit register.

The following figure shows the EDPIDR3 bit assignments.

Figure 11-12 EDPIDR3 bit assignments

The following table shows the EDPIDR3 bit assignments.

Table 11-21 EDPIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	REVAND	`0x0` Part minor revision.
[3:0]	CMOD	`0x0` Customer modified.

EDPIDR3 can be accessed through the external debug interface:

Table 11-22 EDPIDR3 access information

Component	Offset
Debug	`0xFEC`

External Debug Peripheral Identification Register 4

The EDPIDR4 provides information to identify an external debug component.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

EDPIDR4 is a 32-bit register.

The following figure shows the EDPIDR4 bit assignments.

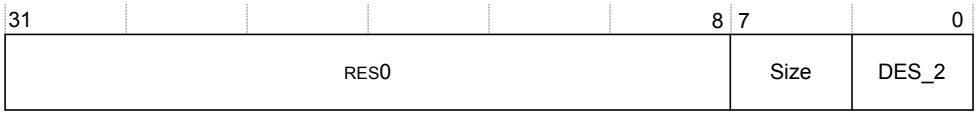

Figure 11-13 EDPIDR4 bit assignments

The following table shows the EDPIDR4 bit assignments.

Table 11-23 EDPIDR4 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Size	`0x0` 4KB. Size of the component.
[3:0]	DES_2	`0x4` Arm Limited. This is the least significant nibble JEP106 continuation code.

EDPIDR4 can be accessed through the external debug interface:

Table 11-24 EDPIDR4 access information

Component	Offset
Debug	`0xFD0`

11.6.6 External Debug Component Identification Registers

There are four read-only External Debug Component Identification Registers, Debug Component ID0 through Debug Component ID3. The following table shows these registers.

Table 11-25 Summary of the External Debug Component Identification Registers

Register	Value	Offset
Component ID0	`0x0D`	`0xFF0`
Component ID1	`0x90`	`0xFF4`
Component ID2	`0x05`	`0xFF8`
Component ID3	`0xB1`	`0xFFC`

The External Debug Component Identification Registers identify Debug as an Arm Debug Interface v5 component. The External Debug Component ID registers are:

- *External Debug Component Identification Register 0* on page 11-376.
- *External Debug Component Identification Register 1* on page 11-377.
- *External Debug Component Identification Register 2* on page 11-378.
- *External Debug Component Identification Register 3* on page 11-379.

External Debug Component Identification Register 0

The EDCIDR0 provides information to identify an external debug component.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

EDCIDR0 is a 32-bit register.

The following figure shows the EDCIDR0 bit assignments.

Figure 11-14 EDCIDR0 bit assignments

The following table shows the EDCIDR0 bit assignments.

Table 11-26 EDCIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_0	`0x0D` Preamble byte 0.

EDCIDR0 can be accessed through the external debug interface:

Table 11-27 EDCIDR0 access information

Component	Offset
Debug	`0xFF0`

External Debug Component Identification Register 1

The EDCIDR1 provides information to identify an external debug component.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

EDCIDR1 is a 32-bit register.

The following figure shows the EDCIDR1 bit assignments.

Figure 11-15 EDCIDR1 bit assignments

The following table shows the EDCIDR1 bit assignments.

Table 11-28 EDCIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	CLASS	`0x9` Debug component.
[3:0]	PRMBL_1	`0x0` Preamble.

EDCIDR1 can be accessed through the external debug interface:

Table 11-29 EDCIDR1 access information

Component	Offset
Debug	`0xFF4`

External Debug Component Identification Register 2

The EDCIDR2 provides information to identify an external debug component.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

EDCIDR2 is a 32-bit register.

The following figure shows the EDCIDR2 bit assignments.

Figure 11-16 EDCIDR2 bit assignments

The following table shows the EDCIDR2 bit assignments.

Table 11-30 EDCIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_2	`0x05` Preamble byte 2.

EDCIDR2 can be accessed through the external debug interface:

Table 11-31 EDCIDR2 access information

Component	Offset
Debug	0xFF8

External Debug Component Identification Register 3

The EDCIDR3 provides information to identify an external debug component.

Usage constraints

This register is accessible through the external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

EDCIDR3 is a 32-bit register.

The following figure shows the EDCIDR3 bit assignments.

Figure 11-17 EDCIDR3 bit assignments

The following table shows the EDCIDR3 bit assignments.

Table 11-32 EDCIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_3	0xB1 Preamble byte 3.

EDCIDR3 can be accessed through the external debug interface:

Table 11-33 EDCIDR3 access information

Component	Offset
Debug	0xFFC

11.6.7 External Debug AArch32 Processor Feature Register

EDAA32PFR provides additional information about implemented processor features in AArch32.

Usage constraints

This register is accessible through the external debug interface and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all build configurations.

Attributes

EDAA32PFR is a 64-bit register.

The following figure shows the EDAA32PFR bit assignments.

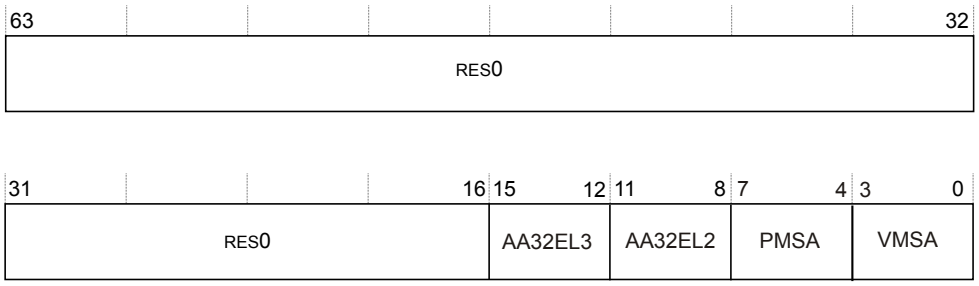

Figure 11-18 EDAA32PFR bit assignments

The following table shows the EDAA32PFR bit assignments.

Table 11-34 EDAA32PFR bit assignments

Bits	Name	Function
[63:16]	-	Reserved, RES0
[15:12]	AA32EL3	AArch32 EL3 Exception level handling. The value is: `0b0000` EL3 is not implemented.
[11:8]	AA32EL2	AArch32 EL2 Exception level handling. The value is: `0b0001` EL2 can be executed in Arch32 state only.
[7:4]	PMSA	Indicates support for PMSA. The value is: `0b0100` PMSAv8 supported, Armv8-R profile.
[3:0]	VMSA	Indicates support for an R-profile VMSA. The value is: `0b0000` VMSA not supported.

EDAA32PFR[31:0] can be accessed through the external debug interface:

Table 11-35 EDAA32PFR[31:0] access information

Component	Offset
Debug	`0xD64`

EDAA32PFR[63:32] can be accessed through the external debug interface:

Table 11-36 EDAA32PFR[63:32] access information

Component	Offset
Debug	`0xD60`

11.6.8 External Debug Processor Feature Register

EDPFR provides additional information about implemented processor features in AArch64. Though Armv8-R implemented by Cortex-R52 does not have AArch64 state, this register is still included.

Usage constraints

This register is accessible through the external debug interface and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all build configurations.

Attributes

EDPFR is a 64-bit register.

The following figure shows the EDPFR bit assignments.

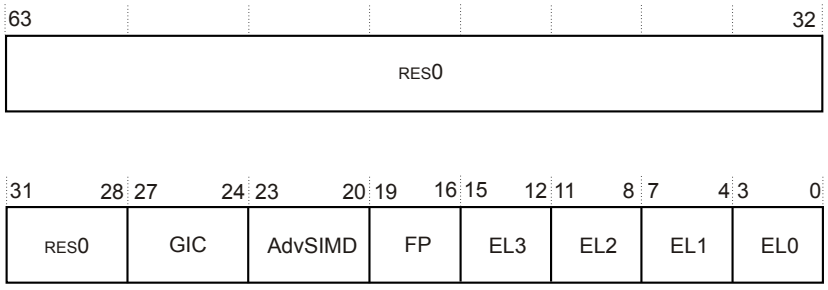

Figure 11-19 EDPFR bit assignments

The following table shows the EDPFR bit assignments.

Table 11-37 EDPFR bit assignments

Bits	Name	Function
[63:28]	-	Reserved, RES0.
[27:24]	GIC	GIC system register interface. The value is: `0b0001` System register interface to GICv3 supported.
[23:20]	AdvSIMD	Advanced SIMD. The value is: `0b0000` Advanced SIMD is implemented.
[19:16]	FP	Floating-point. The value is: `0b0000` Floating-point is implemented.
[15:12]	EL3	AArch64 ELn handling. The value is: `0b0000` ELn is not implemented or cannot use AArch64.
[11:8]	EL2	AArch64 ELn handling. The value is: `0b0000` ELn is not implemented or cannot use AArch64.

Table 11-37 EDPFR bit assignments (continued)

Bits	Name	Function
[7:4]	EL1	AArch64 ELn handling. The value is: `0b0000` ELn must use only AArch32.
[3:0]	EL0	AArch64 ELn handling. The value is: `0b0000` ELn must use only AArch32.

EDPFR[31:0] can be accessed through the external debug interface:

Table 11-38 EDPFR[31:0] access information

Component	Offset
Debug	`0xD20`

EDPFR[63:32] can be accessed through the external debug interface:

Table 11-39 EDPFR[63:32] access information

Component	Offset
Debug	`0xD24`

11.6.9 External Debug Feature Register

EDDFR provides top-level information about the debug system.

Usage constraints

This register is accessible through the external debug interface and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all build configurations.

Attributes

EDDFR is a 64-bit register.

The following figure shows the EDDFR bit assignments.

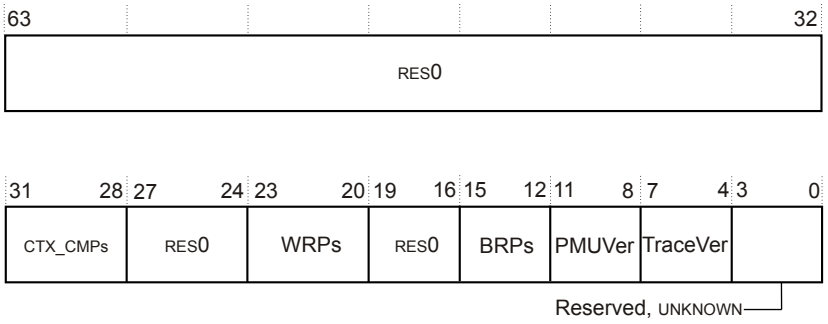

Figure 11-20 EDDFR bit assignments

The following table shows the EDDFR bit assignments.

Table 11-40 EDDFR bit assignments

Bits	Name	Function
[63:32]	-	Reserved, RES0.
[31:28]	CTX_CMPs	`0x1`. Two breakpoints are context-aware. These are the highest numbered breakpoints.
[27:24]	-	Reserved, RES0.
[23:20]	WRPs	`0x7`. Eight watchpoints implemented.
[19:16]	-	Reserved, RES0.
[15:12]	BRPs	`0x7`. Eight watchpoints implemented.
[11:8]	PMUVer	Performance Monitors extension version. Indicates whether System register interface to Performance Monitors extension is implemented. The value is: `0b0001` Performance Monitors extension System registers implemented, PMUv3.
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a trace macrocell is implemented. The value is: `0b0000` Trace macrocell System registers not implemented.
[3:0]	-	Reserved, UNKNOWN.

EDDFR[31:0] can be accessed through the external debug interface:

Table 11-41 EDDFR[31:0] access information

Component	Offset
Debug	`0xD2C`

EDDFR[63:32] can be accessed through the external debug interface:

Table 11-42 EDPFR[63:32] access information

Component	Offset
Debug	`0xD28`

11.7 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 protocol.

The following figure shows the debug interface implemented in the Cortex-R52 processor. For more information on these signals, see the *Arm® CoreSight™ Architecture Specification v2.0*.

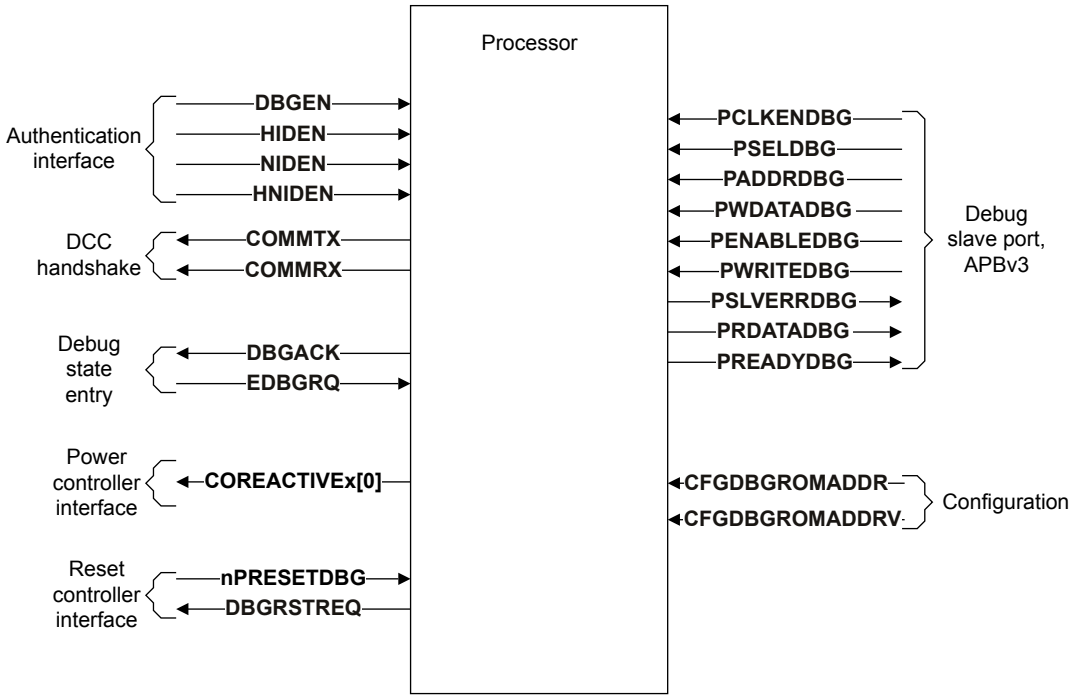

Figure 11-21 External debug interface, including APBv3 slave port

This section contains the following subsections:

- *11.7.1 Debug memory map* on page 11-384.
- *11.7.2 Debug power interface* on page 11-385.
- *11.7.3 Debug over warm reset* on page 11-385.
- *11.7.4 Changing the authentication signals* on page 11-386.

11.7.1 Debug memory map

The APB memory map supports up to four cores in a cluster. The following table shows the address mapping for the debug and trace components.

Table 11-43 Address mapping for debug and trace components

Address range	Component [bs]
0x000000 - 0x00FFFF	ROM table
0x010000 - 0x01FFFF	Core 0 Debug
0x020000 - 0x02FFFF	Core 0 CTI
0x030000 - 0x03FFFF	Core 0 PMU
0x040000 - 0x04FFFF	Core 0 Trace
0x050000 - 0x10FFFF	Reserved

[bs] Indicates the mapped component if present, otherwise reserved.

Table 11-43 Address mapping for debug and trace components (continued)

Address range	Component [bs]
0x110000 - 0x11FFFF	Core 1 Debug
0x120000 - 0x12FFFF	Core 1 CTI
0x130000 - 0x13FFFF	Core 1 PMU
0x140000 - 0x14FFFF	Core 1 Trace
0x150000 - 0x20FFFF	Reserved
0x210000 - 0x21FFFF	Core 2 Debug
0x220000 - 0x22FFFF	Core 2 CTI
0x230000 - 0x23FFFF	Core 2 PMU
0x240000 - 0x24FFFF	Core 2 Trace
0x250000 - 0x30FFFF	Reserved
0x310000 - 0x31FFFF	Core 3 Debug
0x320000 - 0x32FFFF	Core 3 CTI
0x330000 - 0x33FFFF	Core 3 PMU
0x340000 - 0x34FFFF	Core 3 Trace
0x350000 - 0x3FFFFF	Reserved

11.7.2 Debug power interface

The EDPRSR.PU bit reflects the core power state. The **COREACTIVEx[0]** output indicates to the power controller when emulated power down is required.

Related references

5.4.2 Cortex-R52 LPI on page 5-210

11.7.3 Debug over warm reset

When set HIGH, the **CFGL1CACHEINVDISx** input signal disables the automatic hardware controlled invalidation of the L1 data cache after the core is reset using **nCORERESETx** or **nCPUPORESETx**.

The **CFGL1CACHEINVDISx** must be used only to assist debug of an external watchdog triggered reset by allowing the contents of the L1 data cache before the reset to be observable after the reset. If reset is asserted, while L1 data cache fetch is being performed, the accuracy of those cache entries is not guaranteed.

You must not use the **CFGL1CACHEINVDISx** signal to disable automatic hardware-controlled invalidation of the L1 data cache in normal processor powerup sequences.

Each of the **CFGL1CACHEINVDISx** signals corresponds to one of the cores, core x, in the cluster. Each core samples the signal when its corresponding **nCORERESETx** or **nCPUPORESETx** is asserted.

If the functionality offered by the **CFGL1CACHEINVDISx** input signal is not required, the input must be tied to LOW.

Related concepts

6.1 Initialization on page 6-214

[bs] Indicates the mapped component if present, otherwise reserved.

11.7.4 Changing the authentication signals

The **NIDENx**, **DBGENx**, **HIDENx**, and **HNIDENx** input signals must be 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 value to a control register in a system peripheral.
2. If the prior step involves any memory operation, issue a `DSB` instruction.
3. Poll the DBGAUTHSTATUS register 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 or exception entry or 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 Instruction Transfer Register, EDITR, while in Debug state. The relevant combinations of the **DBGENx**, **NIDENx**, **HIDENx**, and **HNIDENx** values can be determined by polling DBGAUTHSTATUS.

11.8 ROM table

This section provides details on the ROM table of the processor.

The Cortex-R52 processor includes a ROM table that complies with the *Arm® CoreSight™ Architecture Specification v2.0*. This table contains a list of components such as core debug units, *Cross Trigger Interface*s (CTIs), core *Performance Monitoring Unit*s (PMUs), and *Embedded Trace Macrocell*s (ETMs). Debuggers can use the ROM table to determine which components are implemented inside the processor.

If a component is not included in your configuration of the processor, the corresponding debug APB ROM table entry is still present but the component is marked as not present.

The ROM table register interface to the ROM table entries is the APB slave port.

This section contains the following subsections:

11.8.1 ROM table register summary

The following table shows the address offsets of each of the entries from the physical base address of the ROM table, which is configured by tying off the **CFGDBGROMADDR** and **CFGDBGROMADDRV** pins when the processor is integrated and reflected in DBGDRAR.

Table 11-44 ROM table registers

Offset	Name	Type	Description
0x000	ROMENTRY0	RO	Core 0 Debug, see *ROM entry registers* on page 11-388
0x004	ROMENTRY1	RO	Core 0 CTI, see *ROM entry registers* on page 11-388
0x008	ROMENTRY2	RO	Core 0 PMU, see *ROM entry registers* on page 11-388
0x00C	ROMENTRY3	RO	Core 0 ETM, see *ROM entry registers* on page 11-388
0x010	ROMENTRY4	RO	Core 1 Debug, see *ROM entry registers* on page 11-388
0x014	ROMENTRY5	RO	Core 1 CTI, see *ROM entry registers* on page 11-388
0x018	ROMENTRY6	RO	Core 1 PMU, see *ROM entry registers* on page 11-388
0x01C	ROMENTRY7	RO	Core 1 ETM, see *ROM entry registers* on page 11-388
0x020	ROMENTRY8	RO	Core 2 Debug, see *ROM entry registers* on page 11-388
0x024	ROMENTRY9	RO	Core 2 CTI, see *ROM entry registers* on page 11-388
0x028	ROMENTRY10	RO	Core 2 PMU, see *ROM entry registers* on page 11-388
0x02C	ROMENTRY11	RO	Core 2 ETM, see *ROM entry registers* on page 11-388
0x030	ROMENTRY12	RO	Core 3 Debug, see *ROM entry registers* on page 11-388
0x034	ROMENTRY13	RO	Core 3 CTI, see *ROM entry registers* on page 11-388
0x038	ROMENTRY14	RO	Core 3 PMU, see *ROM entry registers* on page 11-388
0x03C	ROMENTRY15	RO	Core 3 ETM, see *ROM entry registers* on page 11-388
0x040-0xFCC	-	RO	Reserved, RES0
0xFD0	ROMPIDR4	RO	*ROM table Debug Peripheral Identification Register 4* on page 11-393

Table 11-44 ROM table registers (continued)

Offset	Name	Type	Description
0xFD4	ROMPIDR5	RO	*ROM table Debug Peripheral Identification Register 5-7* on page 11-394
0xFD8	ROMPIDR6	RO	*ROM table Debug Peripheral Identification Register 5-7* on page 11-394
0xFDC	ROMPIDR7	RO	*ROM table Debug Peripheral Identification Register 5-7* on page 11-394
0xFE0	ROMPIDR0	RO	*ROM table Debug Peripheral Identification Register 0* on page 11-390
0xFE4	ROMPIDR1	RO	*ROM table Debug Peripheral Identification Register 1* on page 11-391
0xFE8	ROMPIDR2	RO	*ROM table Debug Peripheral Identification Register 2* on page 11-392
0xFEC	ROMPIDR3	RO	*ROM table Debug Peripheral Identification Register 3* on page 11-392
0xFF0	ROMCIDR0	RO	*ROM table Debug Component Identification Register 0* on page 11-395
0xFF4	ROMCIDR1	RO	*ROM table Debug Component Identification Register 1* on page 11-395
0xFF8	ROMCIDR2	RO	*ROM table Debug Component Identification Register 2* on page 11-396
0xFFC	ROMCIDR3	RO	*ROM table Debug Component Identification Register 3* on page 11-397

11.8.2 ROM table register descriptions

This section describes the ROM table registers.

ROM entry registers

ROMENTRY0-15 indicates to the debugger whether the debug component is present in the debug logic of the processor. There are 16 ROMENTRY registers in the Cortex-R52 processor.

Usage constraints The ROMENTRY registers are all read-only.

Traps and enables There are no traps and enables affecting these registers.

Configurations Available in all processor build configurations.

Attributes ROMENTRY0-15 are 32-bit registers.

The following figure shows the ROMENTRY bit assignments.

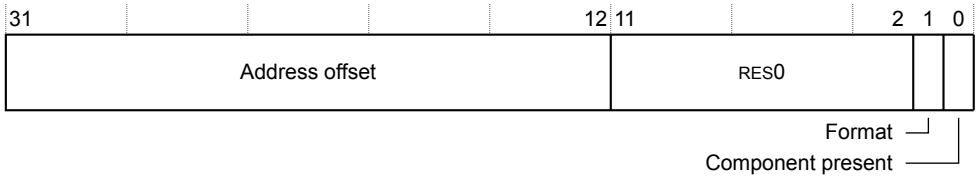

Figure 11-22 ROMENTRY bit assignments

The following table shows the ROMENTRY bit assignments.

Table 11-45 ROMENTRY bit assignments

Bits	Name	Function
[31:12]	Address offset	Address offset for the debug component.
[11:2]	-	Reserved, RES0.

Table 11-45 ROMENTRY bit assignments (continued)

Bits	Name	Function
[1]	Format	Format of the ROM table entry. The value for all ROMENTRY registers is 1, indicating 32 bit format.
[0]	Component present[bt]	Indicates whether the component is present: **0** Component is not present. **1** Component is present.

The address of a debug component is determined by shifting the address offset 12 places to the left and adding the result to the base address of the processor ROM table.

The following table shows the address offset values for all ROMENTRY values and the resulting ROMENTRY values. If a core is not implemented, the ROMENTRY registers for its Debug, CTI, PMU, and ETM components are `0x00000000`.

Table 11-46 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	`0x00040`	`0x00040003`
ROMENTRY4	Core 1 Debug	`0x00110`	`0x00110003` [bu]
ROMENTRY5	Core 1 CTI	`0x00120`	`0x00120003` [bu]
ROMENTRY6	Core 1 PMU	`0x00130`	`0x00130003` [bu]
ROMENTRY7	Core 1 ETM	`0x00140`	`0x00140003` [bu]
ROMENTRY8	Core 2 Debug	`0x00210`	`0x00210003` [bu]
ROMENTRY9	Core 2 CTI	`0x00220`	`0x00220003` [bu]
ROMENTRY10	Core 2 PMU	`0x00230`	`0x00230003` [bu]
ROMENTRY11	Core 2 ETM	`0x00240`	`0x00240003` [bu]
ROMENTRY12	Core 3 Debug	`0x00310`	`0x00310003` [bu]
ROMENTRY13	Core 3 CTI	`0x00320`	`0x00320003` [bu]
ROMENTRY14	Core 3 PMU	`0x00330`	`0x00330003` [bu]
ROMENTRY15	Core 3 ETM	`0x00340`	`0x00340003` [bu]

11.8.3 ROM table Debug Peripheral Identification Registers

The ROM table Debug Peripheral Identification Registers provide information to identify the ROM table.

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.

[bt] core 0 is always present. The component entries for core 1, 2, and 3 depend on your configuration.
[bu] If the component is present.

Table 11-47 Summary of the ROM table Debug Peripheral Identification Registers

Register	Offset	Value
ROMPIDR0	0xFE0	0xB8
ROMPIDR1	0xFE4	0xB4
ROMPIDR2	0xFE8	0x3B
ROMPIDR3	0xFEC	0x00
ROMPIDR4	0xFD0	0x04
ROMPIDR5	0xFD4	0x00
ROMPIDR6	0xFD8	0x00
ROMPIDR7	0xFDC	0x00

Only bits[7:0] of each ROM table Debug Peripheral ID Register are used, with bits[31:8] as RES0. 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 11-390.
- *ROM table Debug Peripheral Identification Register 1* on page 11-391.
- *ROM table Debug Peripheral Identification Register 2* on page 11-392.
- *ROM table Debug Peripheral Identification Register 3* on page 11-392.
- *ROM table Debug Peripheral Identification Register 4* on page 11-393.
- *ROM table Debug Peripheral Identification Register 5-7* on page 11-394.

ROM table Debug Peripheral Identification Register 0

The ROMPIDR0 provides information to identify the Cortex-R52 ROM table.

Usage constraints

This register is accessible through the memory mapped interface and external debug interface, and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

ROMPIDR0 is a 32-bit register.

The following figure shows the ROMPIDR0 bit assignments.

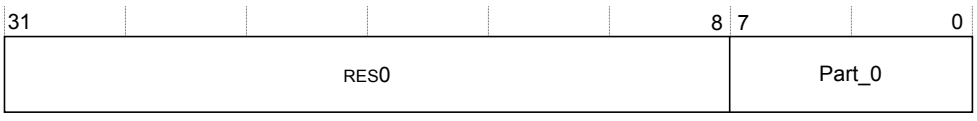

Figure 11-23 ROMPIDR0 bit assignments

The following table shows the ROMPIDR0 bit assignments.

Table 11-48 ROMPIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	Part_0	`0xB8` Least significant byte of the ROM table part number.

ROMPIDR0 can be accessed through the memory mapped interface and external debug interface:

Table 11-49 ROMPIDR0 access information

Component	Offset
Debug	`0xFE0`

ROM table Debug Peripheral Identification Register 1

The ROMPIDR1 provides information to identify the Cortex-R52 ROM table.

Usage constraints

This register is accessible through the memory mapped interface and external debug interface, and is read-only. For a summary of the conditions which affect whether access to this register is permitted see *11.2.4 External register access permissions* on page 11-355.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

ROMPIDR1 is a 32-bit register.

The following figure shows the ROMPIDR1 bit assignments.

Figure 11-24 ROMPIDR1 bit assignments

The following table shows the ROMPIDR1 bit assignments.

Table 11-50 ROMPIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	DES_0	`0xB` Least significant nibble of JEP106 ID code. For Arm Limited.
[3:0]	Part_1	`0x4` Most significant nibble of the ROM table part number.

ROMPIDR1 can be accessed through the memory mapped interface and external debug interface:

Table 11-51 ROMPIDR1 access information

Component	Offset
Debug	0xFE4

ROM table Debug Peripheral Identification Register 2

The ROMPIDR2 provides information to identify the Cortex-R52 ROM table.

Usage constraints

This register is accessible through the memory mapped interface and external debug interface, and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

ROMPIDR2 is a 32-bit register.

The following figure shows the ROMPIDR2 bit assignments.

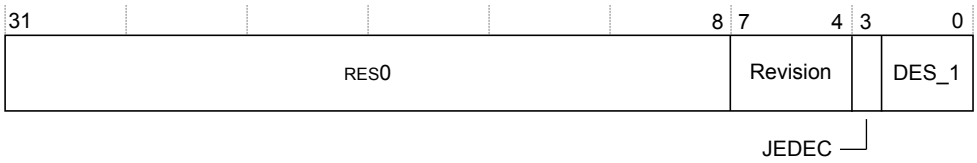

Figure 11-25 ROMPIDR2 bit assignments

The following table shows the ROMPIDR2 bit assignments.

Table 11-52 ROMPIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Revision	0x4
[3]	JEDEC	1 Indicates a JEP106 identity code is used.
[2:0]	DES_1	0b011 Designer, most significant bits of JEP106 ID code. For Arm Limited.

ROMPIDR2 can be accessed through the memory mapped interface and external debug interface:

Table 11-53 ROMPIDR2 access information

Component	Offset
Debug	0xFE8

ROM table Debug Peripheral Identification Register 3

The ROMPIDR3 provides information to identify the Cortex-R52 ROM table.

Usage constraints

This register is accessible through the memory mapped interface and external debug interface, and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

ROMPIDR3 is a 32-bit register.

The following figure shows the ROMPIDR3 bit assignments.

Figure 11-26 ROMPIDR3 bit assignments

The following table shows the ROMPIDR3 bit assignments.

Table 11-54 ROMPIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	REVAND	`0x0` Part minor revision.
[3:0]	CMOD	`0x0` Customer modified.

ROMPIDR3 can be accessed through the memory mapped interface and external debug interface:

Table 11-55 ROMPIDR3 access information

Component	Offset
Debug	`0xFEC`

ROM table Debug Peripheral Identification Register 4

The ROMPIDR4 provides information to identify the Cortex-R52 ROM table.

Usage constraints

This register is accessible through the memory mapped interface and external debug interface, and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

ROMPIDR4 is a 32-bit register.

The following figure shows the ROMPIDR4 bit assignments.

Figure 11-27 ROMPIDR4 bit assignments

The following table shows the ROMPIDR4 bit assignments.

Table 11-56 ROMPIDR4 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Size	`0x0` 4KB. Size of the component.
[3:0]	DES_2	`0x4` Designer, JEP106 continuation code, least significant nibble. For Arm Limited.

ROMPIDR4 can be accessed through the memory mapped interface and external debug interface:

Table 11-57 ROMPIDR4 access information

Component	Offset
Debug	`0xFD0`

ROM table Debug Peripheral Identification Register 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers. They are RES0.

11.8.4 ROM table Debug Component Identification Registers

There are four read-only Component Identification Registers, Component ID0 through Component ID3. The following table shows these registers.

Table 11-58 Summary of the ROM table Debug Component Identification registers

Register	Offset	Value
ROMCIDR0	`0xFF0`	`0x0D`
ROMCIDR1	`0xFF4`	`0x10`
ROMCIDR2	`0xFF8`	`0x05`
ROMCIDR3	`0xFFC`	`0xB1`

The ROM table Debug Component Identification Registers identify the ROM table as an Arm CoreSight Debug component. The ROM table Component ID registers are:

- *ROM table Debug Component Identification Register 0* on page 11-395.
- *ROM table Debug Component Identification Register 1* on page 11-395.
- *ROM table Debug Component Identification Register 2* on page 11-396.
- *ROM table Debug Component Identification Register 3* on page 11-397.

ROM table Debug Component Identification Register 0

The ROMCIDR0 provides information to identify an external debug component.

Usage constraints

This register is accessible through the memory mapped interface and external debug interface, and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

ROMCIDR0 is a 32-bit register.

The following figure shows the ROMCIDR0 bit assignments.

Figure 11-28 ROMCIDR0 bit assignments

The following table shows the ROMCIDR0 bit assignments.

Table 11-59 ROMCIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0
[7:0]	PRMBL_0	0x0D Preamble byte 0

ROMCIDR0 can be accessed through the memory mapped interface and external debug interface:

Table 11-60 ROMCIDR0 access information

Component	Offset
Debug	0xFF0

ROM table Debug Component Identification Register 1

The ROMCIDR1 provides information to identify an external debug component.

Usage constraints

This register is accessible through the memory mapped interface and external debug interface, and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

ROMCIDR1 is a 32-bit register.

The following figure shows the ROMCIDR1 bit assignments.

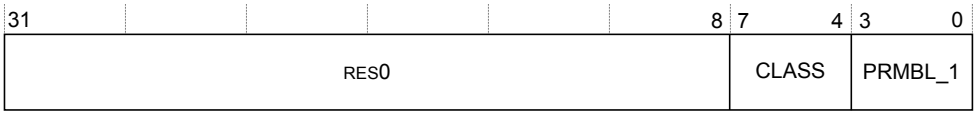

Figure 11-29 ROMCIDR1 bit assignments

The following table shows the ROMCIDR1 bit assignments.

Table 11-61 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.

ROMCIDR1 can be accessed through the memory mapped interface and external debug interface:

Table 11-62 ROMCIDR1 access information

Component	Offset
Debug	`0xFF4`

ROM table Debug Component Identification Register 2

The ROMCIDR2 provides information to identify an external debug component.

Usage constraints

This register is accessible through the memory mapped interface and external debug interface, and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

The ROMCIDR2 is in the Debug power domain, and is available in all processor build configurations.

Attributes

ROMCIDR2 is a 32-bit register.

The following figure shows the ROMCIDR2 bit assignments.

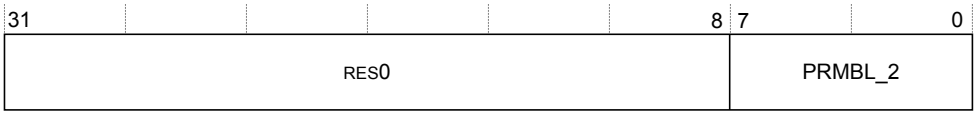

Figure 11-30 ROMCIDR2 bit assignments

The following table shows the ROMCIDR2 bit assignments.

Table 11-63 ROMCIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0
[7:0]	PRMBL_2	0x05 Preamble byte 2

ROMCIDR2 can be accessed through the memory mapped interface and external debug interface:

Table 11-64 ROMCIDR2 access information

Component	Offset
Debug	0xFF8

ROM table Debug Component Identification Register 3

The ROMCIDR3 provides information to identify an external debug component.

Usage constraints

This register is accessible through the memory mapped interface and external debug interface, and is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

ROMCIDR3 is a 32-bit register.

The following figure shows the ROMCIDR3 bit assignments.

Figure 11-31 ROMCIDR3 bit assignments

The following table shows the ROMCIDR3 bit assignments.

Table 11-65 ROMCIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0
[7:0]	PRMBL_3	0xB1 Preamble byte 3

ROMCIDR3 can be accessed through the memory mapped interface and external debug interface:

Table 11-66 ROMCIDR3 access information

Component	Offset
Debug	0xFFC

Chapter 12
Performance Monitor Unit

This section describes the *Performance Monitor Unit* (PMU) features and the registers that it uses.

It contains the following sections:

12.1 About the PMU

Each Cortex-R52 core includes performance monitors which 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 processor that you can use when debugging or profiling code.

The PMU provides four 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 short time.

The events are exported from the core for use by the ETM and counter overflows are exported for use by the GIC distributor unit. The events available include the architecturally required events (instructions, branch predictor accuracy, data cache hit rate), some of the common events (counting types of instruction, particularly branch and load/store), and events for counting different types (and totals) of ECC errors.

The following figure shows the major blocks inside the PMU.

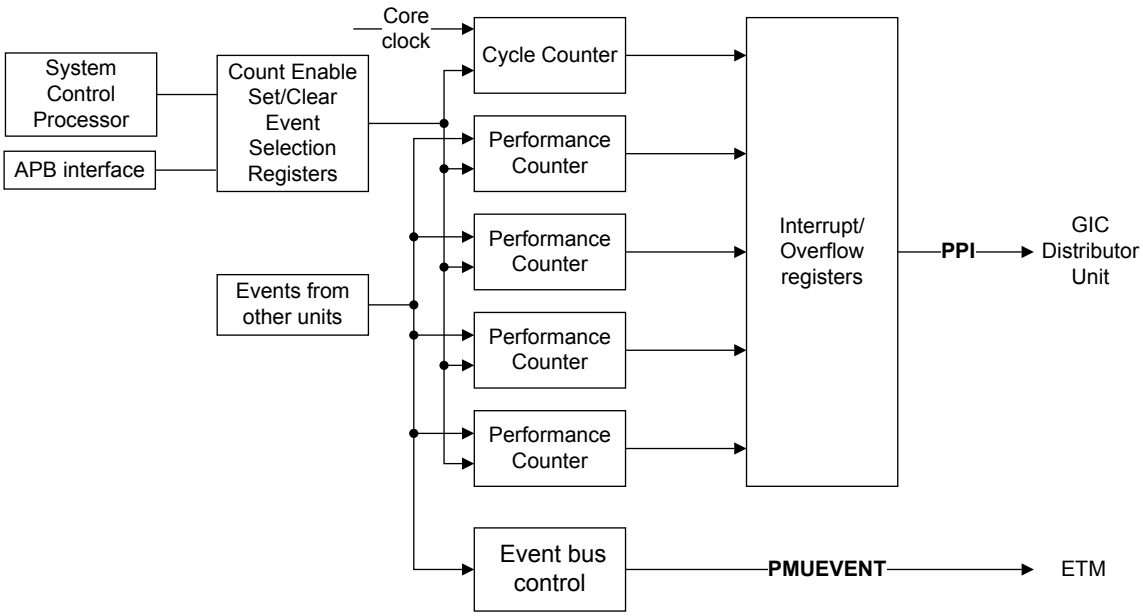

Figure 12-1 PMU block diagram

This section contains the following subsections:

- *12.1.1 Event interface* on page 12-399.
- *12.1.2 System register and APB interface* on page 12-399.
- *12.1.3 Counters* on page 12-400.
- *12.1.4 External register access permissions* on page 12-400.
- *12.1.5 Authentication signals and PMU behavior* on page 12-400.

12.1.1 Event interface

Events from all other units from across the design are provided to the PMU.

12.1.2 System register and APB interface

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

12.1.3 Counters

The PMU has 32-bit event counters which increment when they are enabled based on events and a 64-bit cycle counter.

12.1.4 External register access permissions

Whether or not access is permitted to a register depends on:

- If the core is powered up.
- The state of the PMU Software Lock.
- The state of the debug authentication inputs to the processor.

The behavior that is specific to each register and the type of access to the register is not described in this document. For a detailed description of these features and their effects on the registers, see the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

The register descriptions provided in this section describe whether each register is read/write or read-only.

The *Performance Monitors Cycle Count Register* (PMCCNTR) is always enabled regardless of whether non-invasive debug is enabled unless the DP bit of the PMCR register is set.

12.1.5 Authentication signals and PMU behavior

The PMU events are exported on the PMUEVENT bus when non-invasive debug is enabled, that is, when either **DBGENx** or **NIDENx** inputs are asserted. The *Performance Monitors Cycle Count Register* (PMCCNTR) is always enabled regardless of whether non-invasive debug is enabled. The only exception is if the disable cycle counter bit of the PMCR.DP is set.

The PMU counts events regardless of the non-invasive debug authentication. The export to the PMUEVENT bus is the only thing controlled by the debug authentication. Specifically, the effect of debug authentication on export is:

- If the core is in EL2, export is enabled if one of **DBGENx** or **NIDENx** is asserted, and one of **HIDENx** or **HNIDENx** is asserted.
- If the core is in EL1, export is enabled if one of **DBGENx** or **NIDENx** is asserted.
- If the core is in EL0, export is enabled if one of **DBGENx** or **NIDENx** is asserted, and HCR.TGE=0.
- If the core is in EL0, export is enabled if one of **DBGENx** or **NIDENx** is asserted, one of **HIDENx** or **HNIDENx** is asserted, and HCR.TGE=1.

The number of counters that are accessible from EL0 and EL1 is restricted by the hypervisor and this is controlled by HDCR.HPMN. Event counting is controlled by the HDCR.HPMD which prohibits events counting by the counters accessible to the Guest *Operating System* (OS) in Hyp mode. If HDCR.HPMD=0, event counting by Guest counters is allowed in Hyp mode. If HDCR.HPMD=1, event counting by Guest counters is prohibited in Hyp mode.

12.2 PMU register summary

The PMU counters and their associated control registers are accessible from the internal non-debug system register interface with `MCR` and `MRC` instructions.

The following table gives a summary of the Cortex-R52 PMU registers, with the exception of some read-only ID values and the effect of the number of counters and the event selection values, the PMU registers are fully described in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*. This implies that there is no Cortex-R52-specific behaviors.

See *12.4 Memory-mapped register summary* on page 12-408 for a complete list of registers that are accessible from the memory-mapped interface external debug interface.

Table 12-1 Performance monitor registers

Name	CRn	Op1	CRm	Op2	Type	Reset	Description
PMCR	c9	0	c12	0	RW	`0x41132000`	*12.3.1 Performance Monitors Control Register* on page 12-403
PMCNTENSET				1	RW	UNK	Performance Monitors Count Enable Set Register
PMCNTENCLR				2	RW	UNK	Performance Monitors Count Enable Clear Register
PMOVSR				3	RW	UNK	Performance Monitors Overflow Flag Status Register
PMSWINC				4	WO	UNK	Performance Monitors Software Increment Register
PMSELR				5	RW	UNK	Performance Monitors Event Counter Selection Register
PMCEID0				6	RO	`0x6E1FFFDB`	*12.3.2 Performance Monitors Common Event Identification Register 0* on page 12-405
PMCEID1				7	RO	`0x0000001E`	*12.3.3 Performance Monitors Common Event Identification Register 1* on page 12-406
PMCCNTR	c9	0	c13	0	RW	UNK	Performance Monitors Cycle Count Register
PMXEVTYPER				1	RW	`0x00000000`	Performance Monitors Selected Event Type Register
PMXEVCNTR				2	RW	UNK	Performance Monitors Event Count Registers
PMUSERENR	c9	0	c14	0	RW	`0x00000000`	Performance Monitors User Enable Register
PMINTENSET				1	RW	UNK	Performance Monitors Interrupt Enable Set Register
PMINTENCLR				2	RW	UNK	Performance Monitors Interrupt Enable Clear Register
PMOVSSET				3	RW	UNK	Performance Monitor Overflow Flag Status Set Register

Table 12-1 Performance monitor registers (continued)

Name	CRn	Op1	CRm	Op2	Type	Reset	Description
PMEVCNTR0	c14	0	c8	0	RW	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
PMEVTYPER0			c12	0		UNK	Performance Monitors Selected Event Type Register 0
PMEVTYPER1				1		UNK	Performance Monitors Selected Event Type Register 1
PMEVTYPER2				2		UNK	Performance Monitors Selected Event Type Register 2
PMEVTYPER3				3		UNK	Performance Monitors Selected Event Type Register 3
PMCCFILTR			c15	7		`0x00000000`	Performance Monitors Cycle Count Filter Register

12.3 PMU register descriptions

This section describes the processor PMU registers.

12.3.1 Performance Monitors Control Register

The PMCR provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Usage constraints

This register is read/write. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

If HDCR.TPM is set to 1, then accesses to this register from EL0 and EL1 are trapped to EL2.

If HDCR.TPMCR is set to 1, then accesses to this register from EL0 and EL1 are trapped to EL2.

If HSTR.T9 is set to 1, then accesses to this register from EL0 and EL1 are trapped to EL2.

If PMUSERENR.EN is set to 0, then accesses to this register from EL0 to Undefined mode.

Configurations

Available in all processor build configurations.

Attributes

PMCR is a 32-bit register.

The following figure shows the PMCR bit assignments.

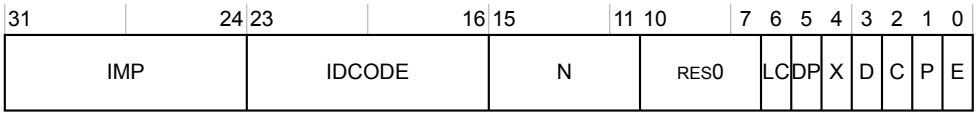

Figure 12-2 PMCR bit assignments

The following table shows the PMCR bit assignments.

Table 12-2 PMCR bit assignments

Bits	Name	Function
[31:24]	IMP	Implementer code: `0x41` Arm. This is a read-only field.
[23:16]	IDCODE	Identification code: `0x13` Cortex-R52. This is a read-only field.
[15:11]	N	Number of event counters. `0b00100` Four counters. This is a read-only field.

Table 12-2 PMCR bit assignments (continued)

Bits	Name	Function
[10:7]	-	Reserved, RES0.
[6]	LC	Long cycle count enable. Determines which PMCCNTR bit generates an overflow recorded in PMOVSR[31]. The possible values are: `0` Overflow on increment that changes PMCCNTR[31] from 1 to 0. The reset value is `0x00000000` `1` Overflow on increment that changes PMCCNTR[63] from 1 to 0. This bit is read/write.The reset value is UNKNOWN.
[5]	DP	Disable cycle counter, PMCCNTR when event counting is prohibited: `0` Cycle counter operates regardless of the non-invasive debug authentication settings. This is the reset value. `1` Cycle counter is disabled if non-invasive debug is not permitted and enabled. This bit is read/write.
[4]	X	Export enable. This bit permits events to be exported to the ETM and the **PMUEVENTx** external bus: `0` Export of events is disabled. This is the reset value. `1` Export of events is enabled. This bit is read/write and does not affect the generation of Performance Monitors interrupts on the **nPMUIRQ** pin.
[3]	D	Clock divider: `0` When enabled, PMCCNTR counts every clock cycle. This is the reset value. `1` When enabled, PMCCNTR counts every 64 clock cycles. This bit is read/write.
[2]	C	Clock counter reset. This bit is write-only. The effects of writing to this bit are: `0` No action. This is the reset value. `1` Reset PMCCNTR to 0. This bit is always RAZ.

Table 12-2 PMCR bit assignments (continued)

Bits	Name	Function
[1]	P	Event counter reset. This bit is write-only. The effects of writing to this bit are: `0` No action. This is the reset value. `1` Reset event counters, not including PMCCNTR, to zero. This bit is always RAZ. In EL0 and EL1, a write of 1 to this bit does not reset event counters that HDCR.HPMN reserves for EL2 use. In EL2 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, are disabled. This is the reset value. `1` All counters are enabled. This bit is read/write. In EL0 and EL1, this bit does not affect the operation of event counters that HDCR.HPMN reserves for EL2 use.

To access the PMCR:

```
MRC p15, 0, <Rt>, c9, c12, 0 ; Read PMCR into Rt
MCR p15, 0, <Rt>, c9, c12, 0 ; Write Rt to PMCR
```

The PMCR can be accessed through the memory-mapped external debug interface, offset `0xE04`.

12.3.2 Performance Monitors Common Event Identification Register 0

The PMCEID0 defines which common architectural and common microarchitectural feature events are implemented.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

If HDCR.TPM is set to 1, read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

If HSTR.T9 is set to 1, read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

If PMUSERENR.EN is set to 0, read accesses to this register from EL0 are trapped to Undefined mode.

Configurations

Available in all processor build configurations.

Attributes

PMCEID0 is a 32-bit register.

The following figure shows the PMCEID0 bit assignments.

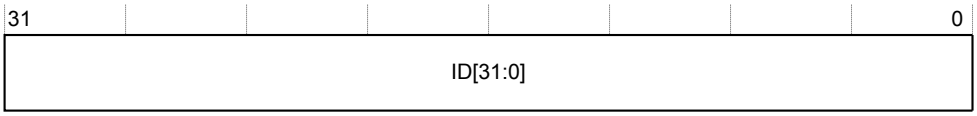

Figure 12-3 PMCEID0 bit assignments

The following table shows the PMCEID0 bit assignments with event implemented or not implemented when the associated bit is set to 1 or 0.

Table 12-3 PMCEID0 bit assignments

Bits	Name	Function
[31:0]	ID[31:0]	Common architectural and microarchitectural feature events that can be counted by the PMU event counters. For each bit described in *12.6 Events* on page 12-419, the event is implemented if the bit is set to 1, or not implemented if the bit is set to 0.

To access the PMCEID0:

```
MRC p15,0,<Rt>,c9,c12,6 ; Read PMCEID0 into Rt
```

The PMCEID0 can be accessed through the memory-mapped external debug interface, offset `0xE20`.

12.3.3 Performance Monitors Common Event Identification Register 1

The PMCEID1 defines which common architectural and common microarchitectural feature events are implemented.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

If HDCR.TPM is set to 1, read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

If HSTR.T9 is set to 1, then read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

If PMUSERENR.EN is set to 0, accesses to this register from EL0 are trapped to Undefined mode.

Configurations

Available in all processor build configurations.

Attributes

PMCEID1 is a 32-bit register.

The following figure shows the PMCEID1 bit assignments.

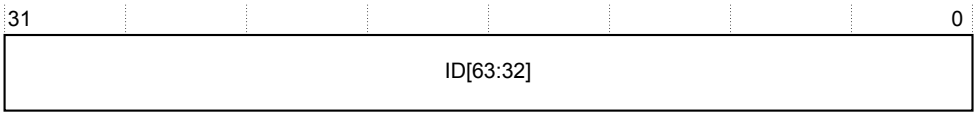

Figure 12-4 PMCEID1 bit assignments

The following table that shows the PMCEID1 bit assignments.

Table 12-4 PMCEID1 bit assignments

Bits	Name	Function
[31:0]	ID[63:32]	Common architectural and microarchitectural feature events that can be counted by the PMU event counters. For each bit described in *12.6 Events* on page 12-419, the event is implemented if the bit is set to 1, or not implemented if the bit is set to 0.

To access the PMCEID1:

```
MRC p15,0,<Rt>,c9,c12,7 ; Read PMCEID1 into Rt
```

The PMCEID1 can be accessed through the memory-mapped external debug interface, offset `0xE24`.

12.4 Memory-mapped register summary

The following table shows the PMU registers that are accessible through the memory-mapped external debug interface.

With the exception of ID values, the effects of the number of counters and the event encodings, these registers are fully described in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*. This implies that there is no Cortex-R52-specific behavior.

Note

The value of **PADDRDBG31** and the CoreSight Software lock restrict write accesses to the PMU registers through the memory-mapped external debug interface. When **PADDRDBG31** is driven LOW, the CoreSight software lock must be used to unlock write permissions for the PMU registers. When **PADDRDBG31** is driven HIGH, the software lock does not restrict the write access permissions of the PMU registers. The software lock is controlled with the Lock Access register.

Table 12-5 Memory-mapped PMU register summary

Offset	Name	Type	Description
0x000	PMEVCNTR0	RW	Performance Monitor Event Count Register 0
0x008	PMEVCNTR1	RW	Performance Monitor Event Count Register 1
0x010	PMEVCNTR2	RW	Performance Monitor Event Count Register 2
0x018	PMEVCNTR3	RW	Performance Monitor Event Count Register 3
0x0F8	PMCCNTR[31:0]	RW	Performance Monitor Cycle Count Register
0x0FC	PMCCNTR[63:32]	RW	
0x400	PMEVTYPER0	RW	Performance Monitor Event Type Register 0
0x404	PMEVTYPER1	RW	Performance Monitor Event Type Register 1
0x408	PMEVTYPER2	RW	Performance Monitor Event Type Register 2
0x40C	PMEVTYPER3	RW	Performance Monitor Event Type Register 3
0x47C	PMCCFILTR	RW	Performance Monitor Cycle Count Filter Register
0xC00	PMCNTENSET	RW	Performance Monitor Count Enable Set Register
0xC20	PMCNTENCLR	RW	Performance Monitor Count Enable Clear Register
0xC40	PMINTENSET	RW	Performance Monitors Interrupt Enable Set Register
0xC60	PMINTENCLR	RW	Performance Monitors Interrupt Enable Clear Register
0xC80	PMOVSCLR	RW	Performance Monitor Overflow Flag Status Register
0xCA0	PMSWINC	WO	Performance Monitor Software Increment Register
0xCC0	PMOVSSET	RW	Performance Monitor Overflow Flag Status Set Register
0xE00	PMCFGR	RO	*12.5.1 Performance Monitor Configuration Register* on page 12-410
0xE04	PMCR[bv]	RW	Performance Monitors Control Register

[bv] This register is distinct from the PMCR system register. It does not have the same value.

Table 12-5 Memory-mapped PMU register summary (continued)

Offset	Name	Type	Description
0xE20	PMCEID0	RO	*Performance Monitors Peripheral Identification Register 0* on page 12-411
0xE24	PMCEID1	RO	Performance Monitor Common Event Identification Register 1
0xFA8	PMDEVAFF0	RO	Performance Monitors Device Affinity Register 0, see *3.3.78 Multiprocessor Affinity Register* on page 3-167
0xFAC	PMDEVAFF1	RO	Performance Monitors Device Affinity Register 1. [bw]
0xFB0	PMLAR	WO	Performance Monitor Lock Access Register
0xFB4	PMLSR	RO	Performance Monitor Lock Status Register
0xFB8	PMAUTHSTATUS	RO	Performance Monitor Authentication Status Register
0xFBC	PMDEVARCH		Performance Monitor Device Architecture Register
0xFCC	PMDEVTYPE	RO	Performance Monitor Device Type Register
0xFD0	PMPIDR4	RO	*Performance Monitors Peripheral Identification Register 4* on page 12-414
0xFD4	PMPIDR5	RO	*Performance Monitors Peripheral Identification Register 5-7* on page 12-415
0xFD8	PMPIDR6	RO	
0xFDC	PMPIDR7	RO	
0xFE0	PMPIDR0	RO	*Performance Monitors Peripheral Identification Register 0* on page 12-411
0xFE4	PMPIDR1	RO	*Performance Monitors Peripheral Identification Register 1* on page 12-412
0xFE8	PMPIDR2	RO	*Performance Monitors Peripheral Identification Register 2* on page 12-413
0xFEC	PMPIDR3	RO	*Performance Monitors Peripheral Identification Register 3* on page 12-414
0xFF0	PMCIDR0	RO	*Performance Monitors Component Identification Register 0* on page 12-415
0xFF4	PMCIDR1	RO	*Performance Monitors Component Identification Register 1* on page 12-416
0xFF8	PMCIDR2	RO	*Performance Monitors Component Identification Register 2* on page 12-417
0xFFC	PMCIDR3	RO	*Performance Monitors Component Identification Register 3* on page 12-418

[bw] This register is RES0

12.5 Memory-mapped register descriptions

This section describes the Cortex-R52 processor PMU registers accessible through the memory-mapped debug interface.

This section contains the following subsections:

12.5.1 Performance Monitor Configuration Register

The PMCFGR contains PMU specific configuration data.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes PMCFGR is a 32-bit register.

The following figure shows the PMCFGR bit assignments.

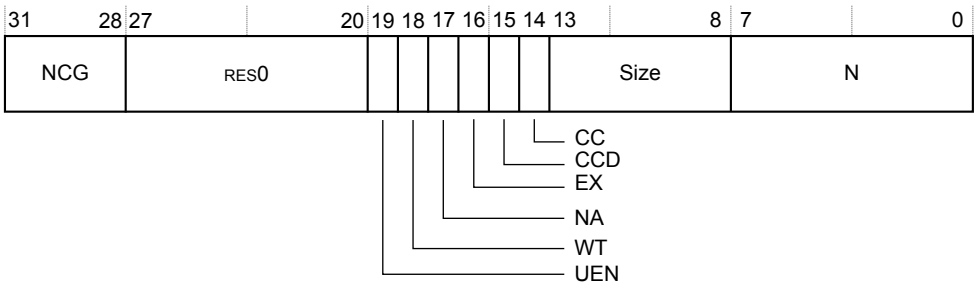

Figure 12-5 PMCFGR bit assignments

The following table shows the PMCFGR bit assignments.

Table 12-6 PMCFGR bit assignments

Bits	Name	Function
[31:28]	NCG	This feature is not supported, RAZ.
[27:20]	-	Reserved, RES0.
[19]	UEN	User-mode Enable Register supported. PMUSERENR_EL0 is not visible in the external debug interface, so this bit is RAZ.
[18]	WT	This feature is not supported, RAZ.
[17]	NA	This feature is not supported, RAZ.
[16]	EX	Export supported. The value is: 1 Export is supported.

Table 12-6 PMCFGR bit assignments (continued)

Bits	Name	Function
[15]	CCD	Cycle counter has prescale. The value is: `1` Cycle counter has prescale.
[14]	CC	Dedicated cycle counter supported. The value is: `1` Dedicated counter supported.
[13:8]	Size	Counter size. The value is: `0b111111` 64-bit counters.
[7:0]	N	Number of event counters. The value is: `0x04` Four counters.

PMCFGR can be accessed through the external debug interface:

Table 12-7 PMCFGR access information

Component	Offset
PMU	`0xEE0`

12.5.2 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.

These registers are required for CoreSight compliance. For more information, see *Arm® CoreSight™ Architecture Specification v2.0*.

There is a set of eight registers, listed in the following table.

Table 12-8 Summary of the Performance Monitors Peripheral Identification Registers

Register	Offset	Value
PMPIDR0	`0xFE0`	`0xB6`
PMPIDR1	`0xFE4`	`0xB9`
PMPIDR2	`0xFE8`	`0x3B`
PMPIDR3	`0xFEC`	`0x00`
PMPIDR4	`0xFD0`	`0x04`
PMPIDR5	`0xFD4`	`0x00`
PMPIDR6	`0xFD8`	`0x00`
PMPIDR7	`0xFDC`	`0x00`

Only bits[7:0] of each Performance Monitors Peripheral ID Register are used, with bits[31:8] as RES0. Together, the eight Performance Monitors Peripheral ID Registers define a single 64-bit Peripheral ID.

Performance Monitors Peripheral Identification Register 0

The PMPIDR0 provides information to identify the Performance Monitor component.

Usage constraints The register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions on page 11-355*. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes PMPIDR0 is a 32-bit register.

The following figure shows the PMPIDR0 bit assignments.

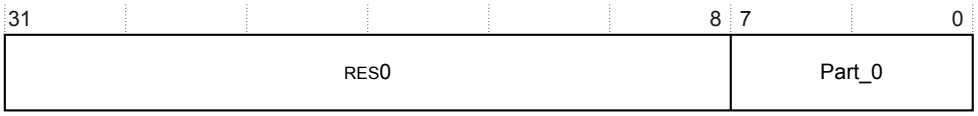

Figure 12-6 PMPIDR0 bit assignments

The following table shows the PMPIDR0 bit assignments.

Table 12-9 PMPIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	Part_0	`0xB6` Least significant byte of the performance monitor part number.

PMPIDR0 can be accessed through the external debug interface:

Table 12-10 PMPIDR0 access information

Component	Offset
PMU	`0xFE0`

Performance Monitors Peripheral Identification Register 1

The PMPIDR1 provides information to identify the Performance Monitor component.

Usage constraints The register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions on page 11-355*. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes PMPIDR1 is a 32-bit register.

The following figure shows the PMPIDR1 bit assignments.

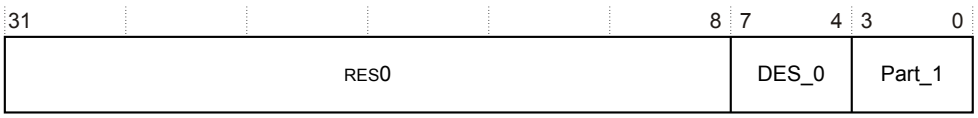

Figure 12-7 PMPIDR1 bit assignments

The following table shows the PMPIDR1 bit assignments.

Table 12-11 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.

PMPIDR1 can be accessed through the external debug interface:

Table 12-12 PMPIDR1 access information

Component	Offset
PMU	`0xFE4`

Performance Monitors Peripheral Identification Register 2

The PMPIDR2 provides information to identify the Performance Monitor component.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions on page 11-355*. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes PMPIDR2 is a 32-bit register.

The following figure shows the PMPIDR2 bit assignments.

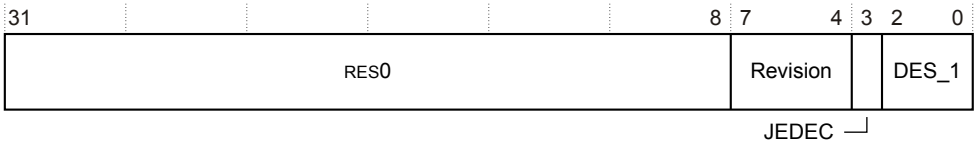

Figure 12-8 PMPIDR2 bit assignments

The following table shows the PMPIDR2 bit assignments.

Table 12-13 PMPIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Revision	`0x4`
[3]	JEDEC	`1` Indicates a JEP106 identity code is used.
[2:0]	DES_1	`0b011` Arm Limited. The most significant bits of JEP106 ID code.

PMPIDR2 can be accessed through the external debug interface:

Table 12-14 PMPIDR2 access information

Component	Offset
PMU	0xFE8

Performance Monitors Peripheral Identification Register 3

The PMPIDR3 provides information to identify the Performance Monitor component.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions on page 11-355*. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes PMPIDR3 is a 32-bit register.

The following figure shows the PMPIDR3 bit assignments.

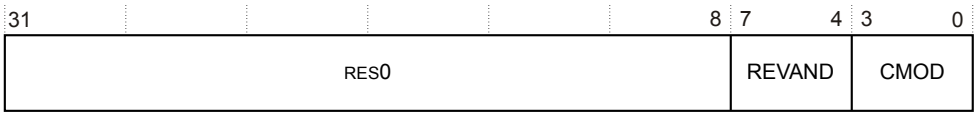

Figure 12-9 PMPIDR3 bit assignments

The following table shows the PMPIDR3 bit assignments.

Table 12-15 PMPIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	REVAND	0x0 Part minor revision.
[3:0]	CMOD	0x0 Customer modified.

PMPIDR3 can be accessed through the external debug interface:

Table 12-16 PMPIDR3 access information

Component	Offset
PMU	0xFEC

Performance Monitors Peripheral Identification Register 4

The PMPIDR4 provides information to identify the Performance Monitor component.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions on page 11-355*. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes PMPIDR4 is a 32-bit register.

The following figure shows the PMPIDR4 bit assignments.

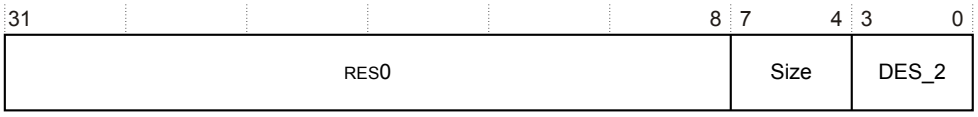

Figure 12-10 PMPIDR4 bit assignments

The following table shows the PMPIDR4 bit assignments.

Table 12-17 PMPIDR4 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Size	`0x0` Size of the component. The PMU occupies a 4KB space.
[3:0]	DES_2	`0x4` Arm Limited. This is the least significant nibble JEP106 continuation code.

PMPIDR4 can be accessed through the external debug interface:

Table 12-18 PMPIDR4 access information

Component	Offset
PMU	`0xFD0`

Performance Monitors Peripheral Identification Register 5-7

No information is held in PMPIDR5, PMPIDR6, and PMPIDR7. They are RES0.

12.5.3 Performance Monitors Component Identification Registers

There are four read-only Performance Monitors Component Identification Registers, Component ID0 through Component ID3.

These registers are required for CoreSight compliance. For more information, see *Arm® CoreSight™ Architecture Specification v2.0*.

The following table shows these registers.

Table 12-19 Summary of the Performance Monitors Component Identification Registers

Register	Offset	Value
PMCIDR0	`0xFF0`	`0x0D`
PMCIDR1	`0xFF4`	`0x90`
PMCIDR2	`0xFF8`	`0x05`
PMCIDR3	`0xFFC`	`0xB1`

The Performance Monitors Component Identification Registers identify the Performance Monitor as Arm PMUv3 architecture.

Performance Monitors Component Identification Register 0

The PMCIDR0 provides information to identify the Performance Monitor component.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions on page 11-355*. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and Enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes PMCIDR0 is a 32-bit register.

The following figure shows the PMCIDR0 bit assignments.

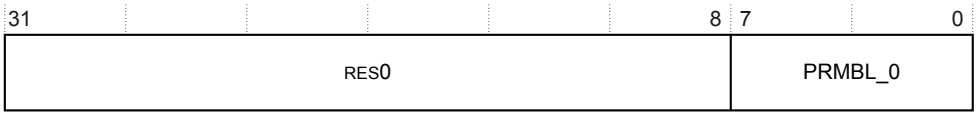

Figure 12-11 PMCIDR0 bit assignments

The following table shows the PMCIDR0 bit assignments.

Table 12-20 PMCIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_0	`0x0D` Preamble byte 0.

PMCIDR0 can be accessed through the external debug interface:

Table 12-21 PMCIDR0 access information

Component	Offset
PMU	`0xFF0`

Performance Monitors Component Identification Register 1

The PMCIDR1 provides information to identify the Performance Monitor component.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions on page 11-355*. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and Enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes PMCIDR1 is a 32-bit register.

The following figure shows the PMCIDR1 bit assignments.

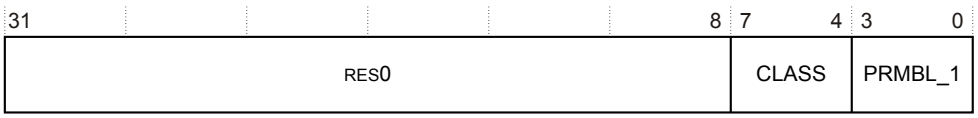

Figure 12-12 PMCIDR1 bit assignments

The following table shows the PMCIDR1 bit assignments.

Table 12-22 PMCIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	CLASS	`0x9` Debug component.
[3:0]	PRMBL_1	`0x0` Preamble.

PMCIDR1 can be accessed through the external debug interface:

Table 12-23 PMCIDR1 access information

Component	Offset
PMU	`0xFF4`

Performance Monitors Component Identification Register 2

The PMCIDR2 provides information to identify the Performance Monitor component.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions on page 11-355*. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and Enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes PMCIDR2 is a 32-bit register.

The following figure shows the PMCIDR2 bit assignments.

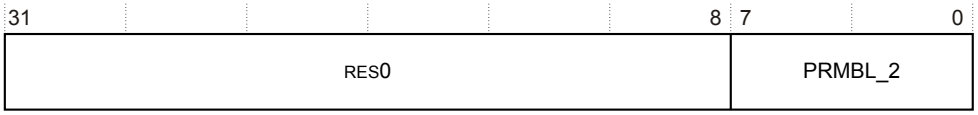

Figure 12-13 PMCIDR2 bit assignments

The following table shows the PMCIDR2 bit assignments.

Table 12-24 PMCIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_2	`0x05` Preamble byte 2.

PMCIDR2 can be accessed through the external debug interface:

Table 12-25 PMCIDR2 access information

Component	Offset
PMU	`0xFF8`

Performance Monitors Component Identification Register 3

The PMCIDR3 provides information to identify the Performance Monitor component.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions on page 11-355*. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and Enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes PMCIDR3 is a 32-bit register.

The following figure shows the PMCIDR3 bit assignments.

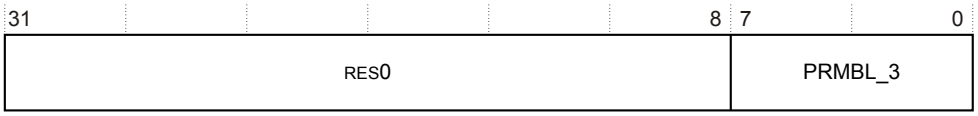

Figure 12-14 PMCIDR3 bit assignments

The following table shows the PMCIDR3 bit assignments.

Table 12-26 PMCIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_3	`0xB1` Preamble byte 3.

PMCIDR3 can be accessed through the external debug interface:

Table 12-27 PMCIDR3 access information

Component	Offset
PMU	`0xFFC`

12.6 Events

Each core also exports some of the events on an external event bus, **PMUEVENTx**, or an external error bus, **ERREVENTx**, according to whether they are profiling events or error detection events. Only the profiling events are connected to the ETM. The following table shows the bit position of each event on one of the event buses, as appropriate. In addition, the table documents the bit position of error detection events that are not associated with a specific core within the **ERREVENTx** bus.

Table 12-28 PMU events

ERREVENT	Event name
[0]	Correctable write data payload bus error occurred from AXIS.
[1]	Fatal write data payload bus error occurred from AXIS.
[2]	Fatal READY signal bus error occurred from AXIS.
[3]	Fatal non-protocol bus error (payload errors) occurred from AXIS.
[4]	Fatal protocol bus error (LEN, ID, LAST, READY, and interconnect protection errors) occurred from AXIS.

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.

Note

- `0x000` to `0x024` are architecturally defined events. See *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* for more information.
- `0x060` to `0x08F` are architecturally recommended. See *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* for more information.
- `0x0C0` to `0x0DF` are implementation defined events.
- **ERREVENT[25:0]** are the ECC error events.
- `0x100` to `0x10D` are attributable performance impact events.
- `0x200` to `0x207` are testability events. These events are for testing purposes only and Arm does not guarantee the accuracy of these events.

Table 12-29 PMU events

Event ID	ETM ID	Mnemonic	PMUEVENTx	ERREVENTx	Event Name
`0x000`	-	SW_INCR	-	-	Software increment. The register is incremented only on writes to the Software Increment Register (PMSWINC).
`0x001`	`0x004`	L1I_CACHE_REFILL	[0]	-	L1 instruction cache refill.
`0x003`	`0x005`	L1D_CACHE_REFILL	[1]	-	L1 data cache refill.
`0x004`	`0x006`	L1D_CACHE	[2]	-	L1 data cache access.
`0x006`	`0x007`	LD_RETIRED	[3]	-	Instruction architecturally executed, condition check pass - load.
`0x007`	`0x008`	ST_RETIRED	[4]	-	Instruction architecturally executed, condition check pass - store.

Table 12-29 PMU events (continued)

Event ID	ETM ID	Mnemonic	PMUEVENTx	ERREVENTx	Event Name
0x008	0x009	INST_RETIRED	[5]	-	Instruction architecturally executed.
-	0x00A	-	[6]	-	Two instructions architecturally executed in parallel. Asserted along with bit **PMUEVENTx[5]** when dual issuing.
0x009	0x00B	EXC_TAKEN	[7]	-	Exception taken.
0x00A	0x00C	EXC_RETURN	[8]	-	Instruction architecturally executed, condition code check pass, exception return.
0x00B	0x00D	CID_WRITE_RETIRED	[9]	-	Instruction architecturally executed, condition code check pass, write to CONTEXTIDR.
0x00C	0x00E	PC_WRITE_RETIRED	[10]	-	Instruction architecturally executed, condition check pass, software change of the PC.
0x00D	0x00F	BR_IMMED_RETIRED	[11]	-	Instruction architecturally executed, immediate branch.
0x00E	0x020	BR_RETURN_RETIRED	[28]	-	Instruction architecturally executed, condition code check pass, procedure return.
0x00F	0x010	UNALIGNED_LDST_RETIRED	[12]	-	Instruction architecturally executed, condition code check pass, unaligned load or store.
0x010	0x011	BR_MIS_PRED	[13]	-	Mispredicted or not predicted branch speculatively executed.
0x011	-	CPU_CYCLES	-	-	Cycle
0x012	0x012	BR_PRED	[14]	-	Predictable branch speculatively executed.
0x013	0x013	MEM_ACCESS	[15]	-	Data memory access.
0x014	0x014	L1I_CACHE	[16]	-	L1 instruction cache access.
0x019	-	BUS_ACCESS	-	-	AXIM, Flash, or LLPP bus access.
0x01A	-	MEMORY_ERROR	-	-	Local memory error (instruction cache, data cache, ATCM, BTCM, CTCM, or Flash).
0x01B	-	INST_SPEC	-	-	Operation speculatively executed.
0x01D	-	BUS_CYCLES	-	-	Bus cycle (AXIM).

Table 12-29 PMU events (continued)

Event ID	ETM ID	Mnemonic	PMUEVENTx	ERREVENTx	Event Name
0x01E	-	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.
0x021	0x01C	BR_RETIRED	[24]	-	Instruction architecturally executed, branch.
0x022	-	BR_MIS_PRED_RETIRED	-	-	Instruction architecturally executed, mispredicted branch.
0x023	-	STALL_FRONTEND	-	-	No operation issued due to the front end.
0x024	-	STALL_BACKEND	-	-	No operation issued due to the back end.
0x060	-	BUS_ACCESS_RD	-	-	Bus access - Read (AXIM, Flash, or LLPP).
0x061	-	BUS_ACCESS_WR	-	-	Bus access - Write (AXIM, Flash, or LLPP).
0x082	0x01B	EXC_SVC	[23]	-	Exception taken, supervisor call.
0x086	0x015	EXC_IRQ	[17]	-	Exception taken, IRQ.
0x087	0x016	EXC_FIQ	[18]	-	Exception taken, FIQ.
0x08A	0x017	EXC_HVC	[19]	-	Exception taken, Hypervisor Call.
0x08E	0x018	EXC_TRAP_IRQ	[20]	-	Exception taken, IRQ not taken locally.
0x08F	0x019	EXC_TRAP_FIQ	[21]	-	Exception taken, FIQ not taken locally.
0x0C0	0x023	KITE_AXI_READ	[31]	-	External memory request, AXIM read.
0x0C1	0x024	KITE_AXI_WRITE	[32]	-	External memory request, AXIM write.
0x0C2	0x025	KITE_FLASH_READ	[33]	-	External memory request, Flash (read-only).
0x0C3	0x026	KITE_LLPP_READ	[34]	-	External memory request, LLPP read.
0x0C4	0x027	KITE_LLPP_WRITE	[35]	-	External memory request, LLPP write.
0x0C5	0x028	KITE_NC_AXI_READ	-	-	Non-cacheable external memory request, AXIM read.
0x0C6	0x029	KITE_NC_AXI_WRITE	-	-	Non-cacheable external memory request, AXIM write.
0x0C7	0x02A	KITE_NC_FLASH_READ	-	-	Non-cacheable external memory request, Flash (read-only).

Table 12-29 PMU events (continued)

Event ID	ETM ID	Mnemonic	PMUEVENTx	ERREVENTx	Event Name
0x0C8	-	KITE_REFILL_PF_AXI	-	-	L1 data cache refill because of prefetch (AXIM only).
0x0C9	-	KITE_REFILL_LS_AXI	-	-	L1 data cache refill because of load or store, AXIM.
0x0CA	-	KITE_REFILL_LS_FLASH	-	-	L1 data cache refill because of load or store, Flash.[bx]
0x0CB	0x02B	KITE_DC_ACCESS_AXI	-	-	L1 data cache access in a way reserved for AXIM.
0x0CC	0x02C	KITE_DC_ACCESS_FLASH	-	-	L1 data cache access in a way reserved for Flash.[by]
0x0CD	0x02D	KITE_IC_ACCESS_AXI	-	-	L1 instruction cache access in a way reserved for AXIM.
0x0CE	0x02E	KITE_IC_ACCESS_FLASH	-	-	L1 instruction cache access in a way reserved for Flash.[bz]
0x0CF	-	KITE_NC_LS_HINTED_AXI	-	-	Non-cacheable external memory request because of load was hinted, AXIM.[ca]
0x0D0	-	KITE_NC_LS_HINTED_FLASH_READ	-	-	Non-cacheable external memory request because of load was hinted, Flash.[ca]
0x0D1	-	KITE_REFILL_IC_AXI	-	-	L1 Instruction cache refill, AXIM.
0x0D2	-	KITE_REFILL_IC_FLASH	-	-	L1 Instruction cache refill, Flash.[cb]
0x0D3	0x02F	KITE_NC_LS_AXI_READ	-	-	Non-cacheable external memory request because of load, AXIM.
0x0D4	0x030	KITE_NC_LS_FLASH_READ	-	-	Non-cacheable external memory request because of load, Flash.
0x0D5	0x01D	KITE_COND_BR_RETIRED	[25]	-	Conditional branch executed.
0x0D6	-	KITE_MIS_PRED_COND_BR	-	-	Conditional branch mispredicted.
0x0D7	0x01E	KITE_BTAC_BR_RETIRED	[26]	-	BTAC branch executed.
0x0D8	-	KITE_MIS_PRED_BTAC_BR	-	-	Conditional branch mispredicted.
0x0D9	0x022	KITE_VSCTLR_CHANGED	[30]	-	Instruction architecturally executed, MCR to VSCTLR.
0x0DA	0x021	KITE_DSB_ALL_RETIRED	[29]	-	Instruction architecturally executed, strong DSB, DFB.

bx Event 0x003 (LID_CACHE_REFILL) is the sum of events 0x0C9 and 0x0CA.
by Event 0x004 (LID_CACHE) is the sum of events 0x0CB and 0x0CC.
bz Event 0x014 (LII_CACHE) is the sum of events 0x0CD and 0x0CE.
ca A hinted *Load Store Unit* (LSU) request is an early request issued from the *Load Store 1* (LS1) pipeline stage to reduce memory latency. More specifically, hinted requests be can be made before the condition codes of the instruction have been determined.
cb Event 001 (LII_CACHE_REFILL) is the sum of events 0x0D1 and 0x0D2.

Table 12-29 PMU events (continued)

Event ID	ETM ID	Mnemonic	PMUEVENTx	ERREVENTx	Event Name
0x0DB	-	KITE_SIMULT_ACCESS_AXI	-	-	Simultaneous accesses from instruction side and data side to AXIM (causing contention).
0x0DC	-	KITE_SIMULT_ACCESS_FLASH	-	-	Simultaneous accesses from instruction side and data side to Flash (causing contention)
0x0DD	0x01A	KITE_EL2_ENTERED	[22]	-	Exception taken to EL2 (hyp mode entry), excluding reset.
0x0DE	0x01F	KITE_CRS_BR_RETIRED	[27]	-	Implementation defined event. CRS branch executed.
0x0DF	-	KITE_MIS_PRED_CRS_BR	-	-	CRS branch mispredicted.
0x0F0	-	KITE_COR_ERR_MEM	-	[0]	Correctable memory error occurred from any source (L1 instruction cache, L1 data cache, ATCM, BTCM, CTCM, or flash).
0x0F1	-	KITE_FAT_ERR_MEM	-	[1]	Fatal memory error occurred from any source (ATCM, BTCM, CTCM, or Flash).
0x0F2	-	KITE_BUS_COR_DATA	-	[2]	Correctable data payload bus error occurred from any source (AXIM or LLPP).
-	-	-	-	[3]	Fatal data payload bus error occurred from any source (AXIM or LLPP).
-	-	-	-	[4]	Fatal READY signal bus error occurred from any source (AXIM, Flash, or LLPP).
0x0F3	-	KITE_BUS_FAT_OTHER	-	[5]	Fatal non-protocol bus error (payload errors, except data) occurred from any source (AXIM, Flash, or LLPP).
0x0F4	-	KITE_BUS_PROTOCOL_ANY	-	[6]	Fatal protocol bus error (LEN, ID, LAST, READY, and interconnect protection errors) occurred from any source (AXIM, Flash, or LLPP).
-	-	-	-	[7]	Correctable L1 instruction cache (data or tag) memory error.
-	-	-	-	[8]	Correctable L1 data cache (data or tag) memory error.
-	-	-	-	[9]	Correctable ATCM, BTCM, or CTCM memory error.
-	-	-	-	[10]	Fatal ATCM, BTCM, or CTCM memory error.

Table 12-29 PMU events (continued)

Event ID	ETM ID	Mnemonic	PMUEVENTx	ERREVENTx	Event Name
-	-	-	-	[11]	Correctable flash memory error.
-	-	-	-	[12]	Fatal flash memory error.
-	-	-	-	[13]	Timeout for AXIM bus.
-	-	-	-	[14]	Timeout for Flash bus.
-	-	-	-	[15]	Timeout for LLPP bus.
-	-	-	-	[16]	Correctable memory occurred from any source (L1 instruction cache, L1 data cache, ATCM, BTCM, CTCM, or Flash) but was not recorded the ERRs because they were full or because it was overwritten by a fatal error.
-	-	-	-	[17]	Fatal memory error occurred from any source (L1 instruction cache, L1 data cache, ATCM, BTCM, CTCM, or Flash) but was not recorded in the ERRs because they were full.
-	-	-	-	[18]	Interrupts masked in Hyp mode for too long (the IMP_INTMONR watchdog has triggered).
-	-	-	-	[19]	A memory or bus fatal error was synchronously detected in Hyp mode.
-	-	-	-	[20]	An abort exception was taken because an EL2-controlled MPU fault generated an abort.
-	-	-	-	[21]	An abort exception was taken because an EL1-controlled MPU fault generated an abort.
-	-	-	-	[22]	An Undefined exception was taken due to any cause.
-	-	-	-	[23]	A memory access marked in EL1-controlled MPU as Device was flushed because it was marked as Normal in EL2-controlled MPU.
-	-	-	-	[24]	Processor livelock because of hard errors or exception at exception vector. This is a pulse bit. An exception of the same type and vector offset was consecutively taken 20 times.
-	-	-	-	[25]	Software running in EL2 unlocks IMP_TESTR1. IMP_TESTR1 is for testing purposes only.

Table 12-29 PMU events (continued)

Event ID	ETM ID	Mnemonic	PMUEVENTx	ERREVENTx	Event Name
0x100	-	KITE_IQ_EMPTY_NO_MISS	-	-	Counts every cycle that the DPU IQ is empty and that is not because of a recent instruction cache miss in any cache way.
0x101	-	KITE_IQ_EMPTY_AXI_MISS	-	-	Counts every cycle that the DPU IQ is empty and there is an instruction cache miss being processed for a cache way reserved for AXI Master.
0x102	-	KITE_IQ_EMPTY_FLASH_MISS	-	-	Counts every cycle that the DPU IQ is empty and there is an instruction cache miss being processed for a cache way reserved for Flash.
0x103	-	KITE_INTERLOCK_OTHER	-	-	Counts every cycle there is an interlock that is not because of an Advanced SIMD or floating-point instruction, and not because of a load/store instruction waiting for data to calculate the address in the AGU. Stall cycles because of a stall in Wr, typically awaiting load data, are excluded.
0x104	-	KITE_INTERLOCK_AGU	-	-	Counts every cycle there is an interlock that is because of a load/store instruction waiting for data to calculate the address in the AGU. Stall cycles because of a stall in Wr, typically awaiting load data, are excluded.
0x105	-	KITE_INTERLOCK_FPASIMD	-	-	Counts every cycle there is an interlock that is because of an Advanced SIMD or floating-point instruction. Stall cycles because of a stall in the Wr stage, typically waiting load data, are excluded.
0x106	-	KITE_LOAD_STALL_AXI	-	-	Counts every cycle there is a stall in the Wr stage because of a load miss from the AXIM.
0x107	-	KITE_LOAD_STALL_FLASH	-	-	Counts every cycle there is a stall in the Wr stage because of a load miss from Flash.
0x108	-	KITE_WR_STALL_STORE	-	-	Counts every cycle there is a stall in the Wr stage because of a store.
0x109	-	KITE_WR_STALL_AXI_STB_FULL	-	-	Store stalled because the AXIM part of the store buffer was full.

Table 12-29 PMU events (continued)

Event ID	ETM ID	Mnemonic	PMUEVENTx	ERREVENTx	Event Name
0x10A	-	KITE_WR_STALL_TCM_STB_FULL	-	-	Store stalled because the TCM part of the store buffer was full.
0x10B	-	KITE_WR_STALL_LLPP_STB_FULL	-	-	Store stalled because the LLPP part of the store buffer was full.
0x10C	-	KITE_BARRIER_STALL_BARRIER	-	-	Barrier stalled because store buffer was busy with another barrier.
0x10D	-	KITE_BARRIER_STORE_AXIWRITE	-	-	Barrier stalled because it was waiting for a write to complete on the AXIM bus.
0x200	-	KITE_IC_WT_HIT	-	-	L1 instruction cache way tracker hit.
0x201	-	KITE_DC_WT_HIT	-	-	L1 data cache way tracker hit.
0x202	-	KITE_I_UMPU_HIT	-	-	Instruction side micro MPU hit.
0x203	-	KITE_D_UMPU_HIT	-	-	Data side micro MPU hit. This can also be counted on a pipeline stall.
0x204	-	KITE_IC_CACHE_HIT	-	-	L1 instruction cache hit.
0x205	-	KITE_IC_LFB_HIT	-	-	L1 instruction cache linefill buffer hit.
0x206	-	KITE_IC_BIU_HIT	-	-	L1 instruction cache hit on BIU response.
0x207	-	KITE_IC_HINT_REQ	-	-	L1 instruction cache hint request sent.

Note

Error events are expected to be rare and combine errors from different sources. If multiple errors happen to occur at the same cycle, the respective PMU counter or **PMUEVENTx**/**ERREVENTx** bit indicates a single error.

12.7 Interrupts

The Cortex-R52 processor asserts the PMU interrupt signal when an interrupt is generated by the PMU. This signal is connected to the Cortex-R52 processor interrupt controller as one of the PPIs. It appears as interrupt ID 23.

This interrupt is also driven as a trigger input to the CTI.

Related references

Chapter 13 Cross Trigger on page 13-429

12.8 Exporting PMU events

This section describes exporting of PMU events.

This section contains the following subsections:

12.8.1 External hardware

In addition to the counters in the processor, there are some events that are exported on the **PMUEVENTx** bus and can be connected to external hardware. Export of events on the **PMUEVENTx** bus is enabled with the PMCR.X bit.

For information on these events, see *Table 12-29 PMU events* on page 12-419.

12.8.2 Debug trace hardware

The same events that are exported on **PMUEVENTx** are exported to the ETM trace unit to enable the events to be monitored for generating trace and other triggers.

For more information on these events, see *Table 12-29 PMU events* on page 12-419.

Related references

Chapter 13 Cross Trigger on page 13-429

Chapter 14 Embedded Trace Macrocell on page 14-447

Chapter 13
Cross Trigger

This chapter describes the cross trigger logic for the Cortex-R52 processor.

It contains the following sections:

13.1 About the cross trigger

In the Cortex-R52 processor, the debug logic in each core and each ETM have cross-trigger inputs and outputs which can be used to signal trigger events. Each core has an associated CoreSight *Cross-Trigger Interface* (CTI) which connects these signals. The CTI blocks are in turn connected by a *Cross-Trigger Matrix* (CTM) to a single external cross-trigger channel interface for the processor.

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.

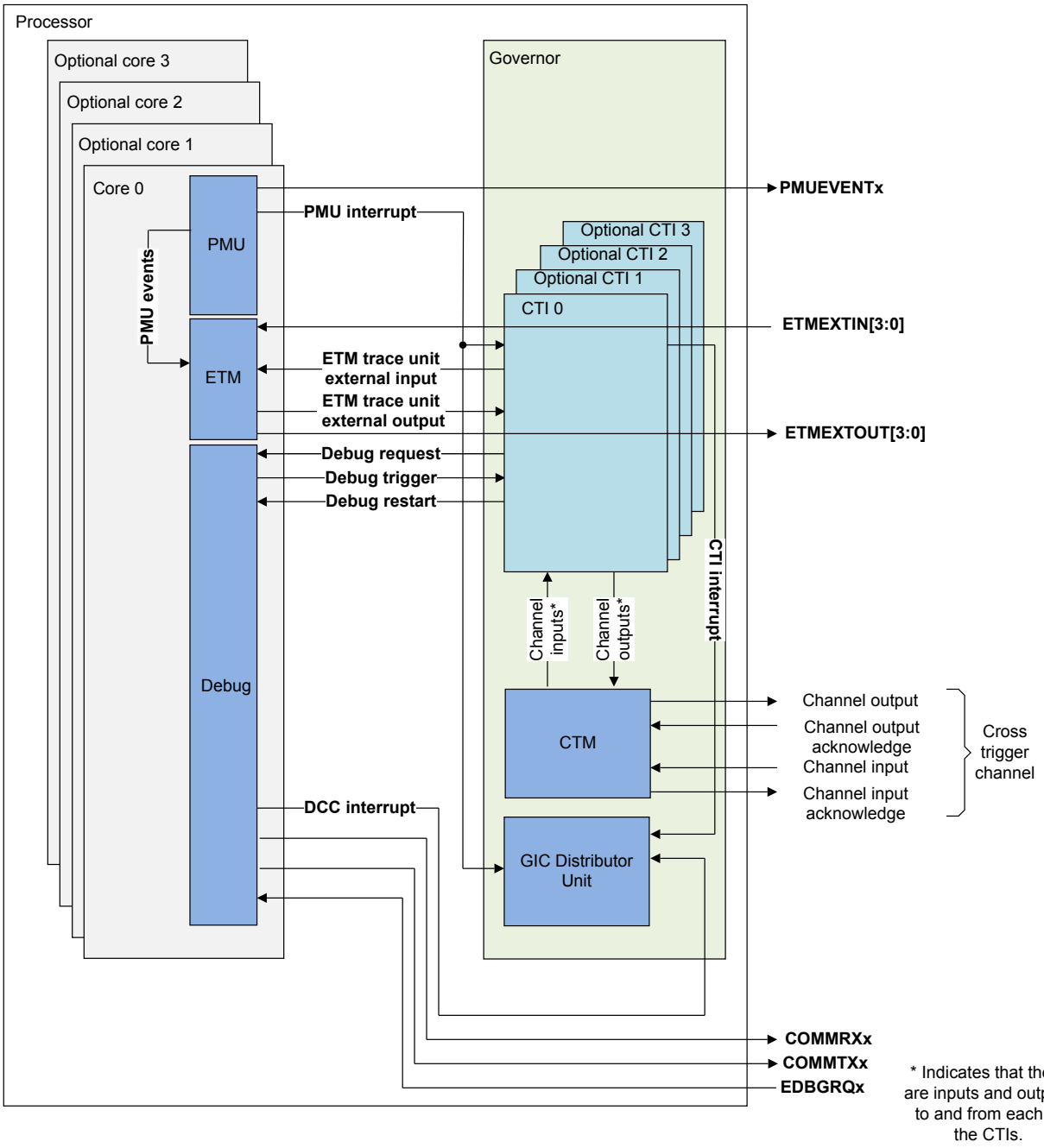

Figure 13-1 Debug system components

13.2 Trigger inputs and outputs

This section describes the trigger inputs and trigger outputs which are available to the CTI.

The following table shows the CTI inputs.

Table 13-1 Trigger inputs

CTI input	Description
0	Cross Halt trigger event.
1	PMU interrupt
2	-
3	-
4	ETM trace unit external output 0
5	ETM trace unit external output 1
6	ETM trace unit external output 2
7	ETM trace unit external output 3

The following table shows the CTI outputs.

Table 13-2 Trigger outputs

CTI output	Description
0	Requests the processor to enter debug state
1	Requests the processor to exit debug state
2	CTI interrupt
3	-
4	ETM trace unit external input 0
5	ETM trace unit external input 1
6	ETM trace unit external input 2
7	ETM trace unit external input 3

13.3 Cortex®-R52 CTM

The CoreSight CTI channel signals from all the cores are combined using a CTM block so that a single cross trigger channel interface is presented by the Cortex-R52 processor.

The following figure illustrates the Cortex-R52 processor CTM. In the 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. Only one of the bits of the channel is shown but each bit is similar to the figure. The figure does not illustrate the handshaking and synchronization logic associated with the channels.

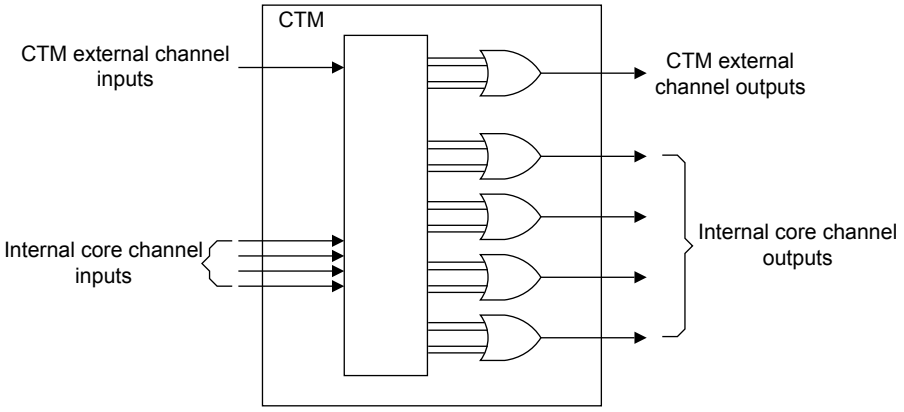

Figure 13-2 Cortex-R52 processor CTM

13.4 Cross trigger register summary

This section describes the registers for each CTI in the Cortex-R52 processor. These registers are accessed through the memory-mapped interface or the external debug interface.

For more information on the base address for each CTI in the Cortex-R52 processor, see *11.7.1 Debug memory map* on page 11-384.

The following table gives a summary of the Cortex-R52 cross trigger registers. With the exception of a number of read-only registers that contain ID information, the CTI registers are fully described in *Arm® CoreSight™ Architecture Specification v2.0*. This implies that there are no Cortex-R52-specific details.

Note

The value of **PADDRDBG31** and the CoreSight software lock restrict write accesses to the CTI registers through the memory-mapped external debug interface. When **PADDRDBG31** is driven LOW, the CoreSight software lock must be used to unlock write permissions for the cross trigger registers. When **PADDRDBG31** is driven HIGH, the software lock does not restrict the write access permissions of the cross trigger registers. The software lock is controlled with the Lock Access register (CTILAR).

Table 13-3 Cross trigger register summary

Offset	Name	Type	Description
0x000	CTICONTROL	RW	CTI Control Register
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	
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	
0x130	CTITRIGINSTATUS	RO	CTI Trigger In Status Register

Table 13-3 Cross trigger register summary (continued)

Offset	Name	Type	Description
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
0x144	ASICCTL	RW	CTI External Multiplexer Control Register
0xF00	CTIITCTRL	RW	*13.5.2 CTI Integration Mode Control Register* on page 13-437
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 13-436
0xFCC	CTIDEVTYPE	RO	CTI Device Type Register
0xFD0	CTIPIDR4	RO	*Peripheral Identification Register 4* on page 13-442
0xFD4	CTIPIDR5	RO	*Peripheral Identification Register 5-7* on page 13-442
0xFD8	CTIPIDR6	RO	
0xFDC	CTIPIDR7	RO	
0xFE0	CTIPIDR0	RO	*Peripheral Identification Register 0* on page 13-438
0xFE4	CTIPIDR1	RO	*Peripheral Identification Register 1* on page 13-439
0xFE8	CTIPIDR2	RO	*Peripheral Identification Register 2* on page 13-440
0xFEC	CTIPIDR3	RO	*Peripheral Identification Register 3* on page 13-441
0xFF0	CTICIDR0	RO	*Component Identification Register 0* on page 13-443
0xFF4	CTICIDR1	RO	*Component Identification Register 1* on page 13-444
0xFF8	CTICIDR2	RO	*Component Identification Register 2* on page 13-444
0xFFC	CTICIDR3	RO	*Component Identification Register 3* on page 13-445

13.4.1 External register access permissions

Whether access is permitted to a CTI register depends on:

- The state of the CTI software lock.
- The state of the debug authentication inputs to the processor.

13.5 Cross trigger register descriptions

This section contains the following subsections:

13.5.1 CTI Device Identification Register

The CTIDEVID describes the CTI component to the debugger.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

The CTIDEVID is a 32-bit register.

The following figure shows the CTIDEVID bit assignments.

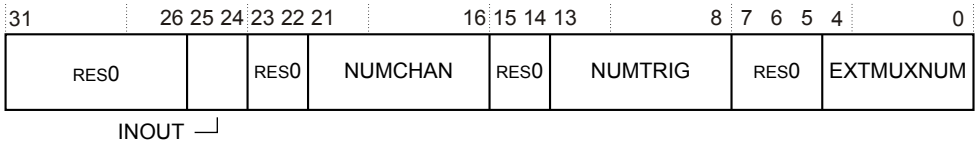

Figure 13-3 CTIDEVID bit assignments

The following table shows the CTIDEVID bit assignments.

Table 13-4 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 value is: `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.

Table 13-4 CTIDEVID bit assignments (continued)

Bits	Name	Function
[13:8]	NUMTRIG	Number of triggers implemented. This value is: `0b001000` Eight triggers implemented.
[7:5]	-	Reserved, RES0.
[4:0]	EXTMUXNUM	Indicates the number of multiplexers available on Trigger Inputs and Triggers Outputs. This value is: `0b00000` No external triggers implemented.

CTIDEVID can be accessed through the external debug interface:

Table 13-5 CTIDEVID access information

Component	Offset
CTI	`0xFC8`

13.5.2 CTI Integration Mode Control Register

The CTIITCTRL register shows that the Cortex-R52 processor does not implement an integration mode.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

The CTIITCTRL is a 32-bit register.

The following figure shows the CTIITCTRL bit assignments.

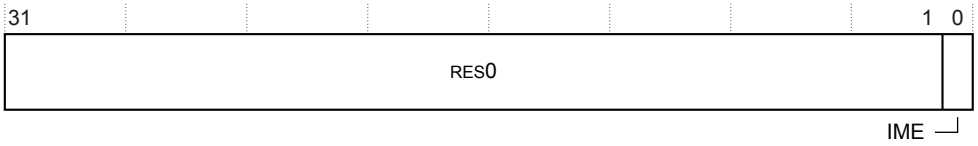

Figure 13-4 CTIITCTRL bit assignments

The following table shows the CTIITCTRL bit assignments.

Table 13-6 CTIITCTRL bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	IME	Integration mode enable. The value is: `0` Normal operation. **Note** Writes to this bit are ignored.

CTIITCTRL can be accessed through the external debug interface:

Table 13-7 CTIITCTRL access information

Component	Offset
CTI	`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. They are a set of eight registers, as listed in the following table.

Table 13-8 Summary of the Peripheral Identification Registers

Register	Value	Offset
Peripheral ID0	`0xB6`	`0xFE0`
Peripheral ID1	`0xB9`	`0xFE4`
Peripheral ID2	`0x0B`	`0xFE8`
Peripheral ID3	`0x00`	`0xFEC`
Peripheral ID4	`0x04`	`0xFD0`
Peripheral ID5	`0x00`	`0xFD4`
Peripheral ID6	`0x00`	`0xFD8`
Peripheral ID7	`0x00`	`0xFDC`

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 13-438.
- *Peripheral Identification Register 1* on page 13-439.
- *Peripheral Identification Register 2* on page 13-440.
- *Peripheral Identification Register 3* on page 13-441.
- *Peripheral Identification Register 4* on page 13-442.
- *Peripheral Identification Register 5-7* on page 13-442.

Peripheral Identification Register 0

The CTIPIDR0 provides information to identify a CTI component.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

The CTIPIDR0 is a 32-bit register.

The following figure shows the CTIPIDR0 bit assignments.

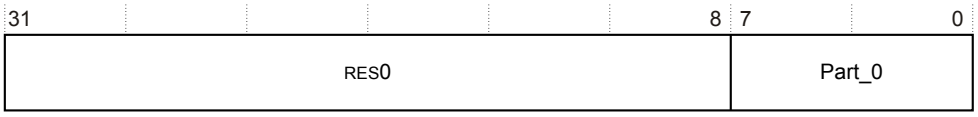

Figure 13-5 CTIPIDR0 bit assignments

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	`0xB6` Least significant byte of the cross trigger part number.

CTIPIDR0 can be accessed through the external debug interface:

Table 13-10 CTIPIDR0 access information

Component	Offset
CTI	`0xFE0`

Peripheral Identification Register 1

The CTIPIDR1 provides information to identify a CTI component.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

The CTIPIDR1 is a 32-bit register.

The following figure shows the CTIPIDR1 bit assignments.

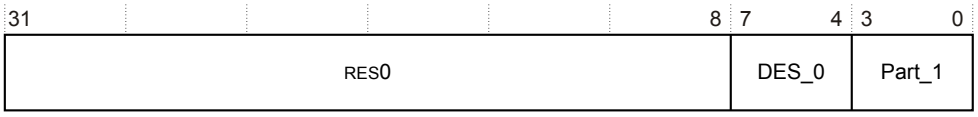

Figure 13-6 CTIPIDR1 bit assignments

The following table shows the CTIPIDR1 bit assignments.

Table 13-11 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.

CTIPIDR1 can be accessed through the external debug interface:

Table 13-12 CTIPIDR1 access information

Component	Offset
CTI	`0xFE4`

Peripheral Identification Register 2

The CTIPIDR2 provides information to identify a CTI component.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

The CTIPIDR2 is a 32-bit register.

The following figure shows the CTIPIDR2 bit assignments.

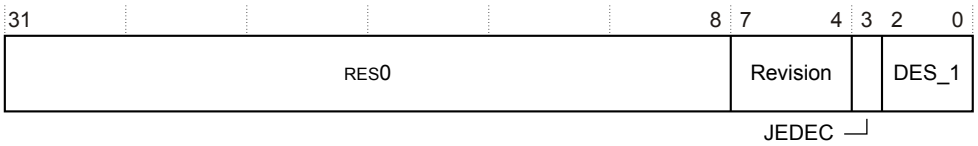

Figure 13-7 CTIPIDR2 bit assignments

The following table shows the CTIPIDR2 bit assignments.

Table 13-13 CTIPIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Revision	0x4
[3]	JEDEC	1 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:

Table 13-14 CTIPIDR2 access information

Component	Offset
CTI	0xFE8

Peripheral Identification Register 3

The CTIPIDR3 provides information to identify a CTI component.

Usage constraints This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables There are no traps and enables affecting this register.

Configurations Available in all processor build configurations.

Attributes The CTIPIDR3 is a 32-bit register.

The following figure shows the CTIPIDR3 bit assignments.

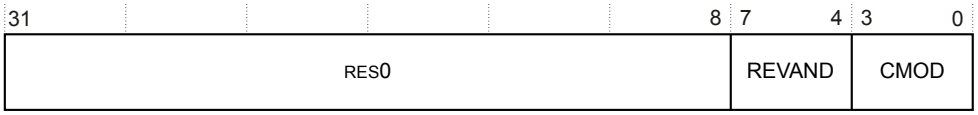

Figure 13-8 CTIPIDR3 bit assignments

The following table shows the CTIPIDR3 bit assignments.

Table 13-15 CTIPIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	REVAND	0x0 Part minor revision.
[3:0]	CMOD	0x0 Customer modified.

CTIPIDR3 can be accessed through the external debug interface:

Table 13-16 CTIPIDR3 access information

Component	Offset
CTI	0xFEC

Peripheral Identification Register 4

The CTIPIDR4 provides information to identify a CTI component.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

The CTIPIDR4 is a 32-bit register.

The following figure shows the CTIPIDR4 bit assignments.

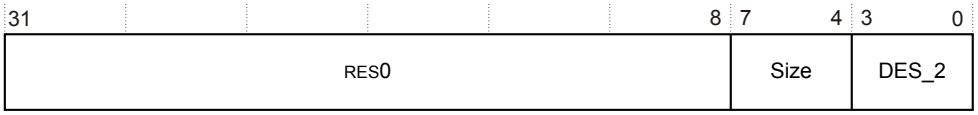

Figure 13-9 CTIPIDR4 bit assignments

The following table shows the CTIPIDR4 bit assignments.

Table 13-17 CTIPIDR4 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	Size	0x0 1KB. Size of the component.
[3:0]	DES_2	0x4 Arm Limited. This is the least significant nibble JEP106 continuation code.

CTIPIDR4 can be accessed through the external debug interface:

Table 13-18 CTIPIDR4 access information

Component	Offset
CTI	0xFD0

Peripheral Identification Register 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers. They 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 13-19 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 13-443.
- *Component Identification Register 1* on page 13-444.
- *Component Identification Register 2* on page 13-444.
- *Component Identification Register 3* on page 13-445.

Component Identification Register 0

The CTICIDR0 provides information to identify a CTI component.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

The CTICIDR0 is a 32-bit register.

The following figure shows the CTICIDR0 bit assignments.

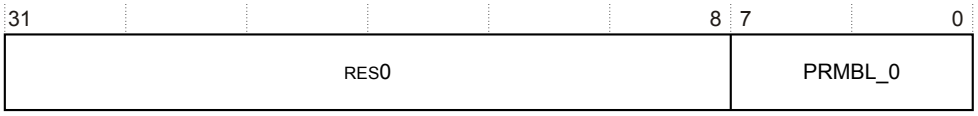

Figure 13-10 CTICIDR0 bit assignments

The following table shows the CTICIDR0 bit assignments.

Table 13-20 CTICIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_0	0x0D Preamble byte 0.

CTICIDR0 can be accessed through the external debug interface:

Table 13-21 CTICIDR0 access information

Component	Offset
CTI	0xFF0

Component Identification Register 1

The CTICIDR1 provides information to identify a CTI component.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

The CTICIDR1 is a 32-bit register.

The following figure shows the CTICIDR1 bit assignments.

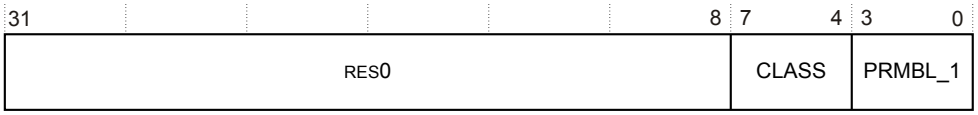

Figure 13-11 CTICIDR1 bit assignments

The following table shows the CTICIDR1 bit assignments.

Table 13-22 CTICIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	CLASS	0x9 Debug component.
[3:0]	PRMBL_1	0x0 Preamble byte 1.

CTICIDR1 can be accessed through the external debug interface:

Table 13-23 CTICIDR1 access information

Component	Offset
CTI	0xFF4

Component Identification Register 2

The CTICIDR2 provides information to identify a CTI component.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

The CTICIDR2 is a 32-bit register.

The following figure shows the CTICIDR2 bit assignments.

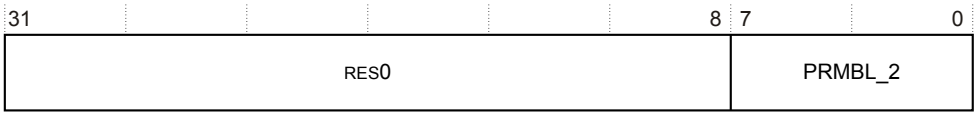

Figure 13-12 CTICIDR2 bit assignments

The following table shows the CTICIDR2 bit assignments.

Table 13-24 CTICIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_2	`0x05` Preamble byte 2.

CTICIDR2 can be accessed through the external debug interface:

Table 13-25 CTICIDR2 access information

Component	Offset
CTI	`0xFF8`

Component Identification Register 3

The CTICIDR3 provides information to identify a CTI component.

Usage constraints

This register is read-only. For a summary of the conditions which affect whether access to this register is permitted, see *11.2.4 External register access permissions* on page 11-355. For a detailed description, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all processor build configurations.

Attributes

The CTICIDR3 is a 32-bit register.

The following figure shows the CTICIDR3 bit assignments.

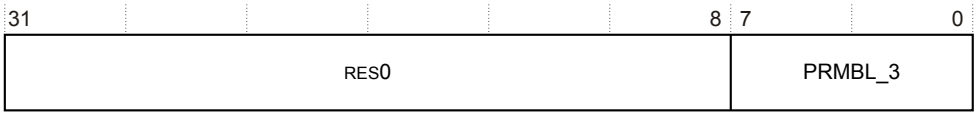

Figure 13-13 CTICIDR3 bit assignments

The following table shows the CTICIDR3 bit assignments.

Table 13-26 CTICIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_3	`0xB1` Preamble byte 3.

CTICIDR3 can be accessed through the external debug interface:

Table 13-27 CTICIDR3 access information

Component	Offset
CTI	`0xFFC`

Chapter 14
Embedded Trace Macrocell

This chapter describes the *Embedded Trace Macrocell* (ETM) for the Cortex-R52 processor.

It contains the following sections:

14.1 About the ETM

The Cortex-R52 ETM performs real-time instruction and data flow tracing based on the ETM architecture ETMv4.2. Cortex-R52 supports one ETM per core.

The ETM is a CoreSight component, and is an integral part of the Arm Real-time Debug solution, DS-5 Development Studio.

The following figure shows the main functional blocks of the ETM.

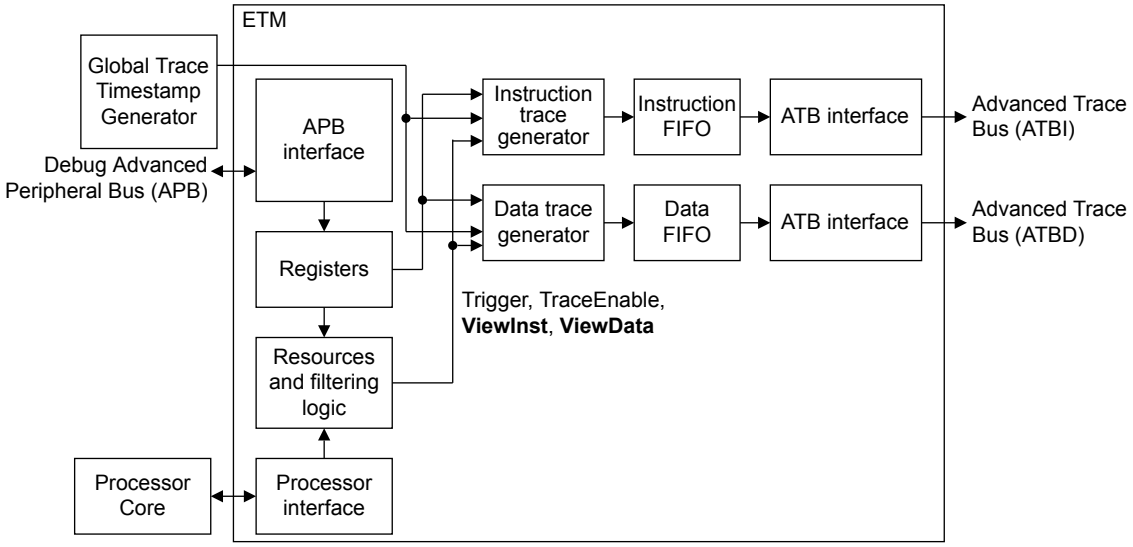

Figure 14-1 ETM block diagram

This section contains the following subsections:

- *14.1.1 Processor interface* on page 14-448.
- *14.1.2 Instruction trace generator* on page 14-448.
- *14.1.3 Data trace generator* on page 14-448.
- *14.1.4 FIFO* on page 14-449.
- *14.1.5 Resources and filtering logic* on page 14-449.
- *14.1.6 ATB interfaces* on page 14-449.
- *14.1.7 APB interface* on page 14-449.
- *14.1.8 Global timestamping* on page 14-449.

14.1.1 Processor interface

This block connects to the Cortex-R52 core. It tracks the execution information from the core, decodes the control signals, and passes on the information to the internal interfaces.

14.1.2 Instruction trace generator

This block generates the trace packets which are a compressed form of the instruction execution information provided by the Cortex-R52 core. The trace packets are then passed to the instruction FIFO.

14.1.3 Data trace generator

This block generates trace packets which are a compressed form of the data transfers (data address and data value) provided by the Cortex-R52 core. The trace packets are then passed to the data FIFO.

14.1.4 FIFO

This block buffers bursts of trace packets. Separate FIFOs are provided, one for the instruction trace stream, and one for the data trace stream.

14.1.5 Resources and filtering logic

These blocks contain resources which are programmed by trace software to trigger and filter the trace information. They start and stop trace generation, depending on the conditions that have been set.

14.1.6 ATB interfaces

There are two ATB interfaces:

Instruction ATB interface
This reads up to four bytes of packet information from the instruction FIFO and sends them over the instruction ATB interface.

Data ATB interface
This reads up to eight bytes of packet information from the data FIFO and sends them over the data ATB interface.

14.1.7 APB interface

This block implements the interface to the APB that provides access to the programmable registers.

14.1.8 Global timestamping

A global timestamp source can be connected to the ETMs through the **TSVALUEB**.

This provides a 64-bit timestamp that a debugger can use for coarse-grained profiling, and correlation of trace sources. Arm recommends that the timestamp counter is no slower than 10% of the processor clock frequency.

——— **Note** ———

Decompression of data trace relies on the presence of a global timestamp count.

———————————

14.2 ETM trace unit generation options and resources

The following table shows the resources of the ETM that are implemented.

Table 14-1 ETM trace unit resources implemented

Feature	Instance
Address comparators	4 pairs
Data value comparators	2
Data address comparators	2
Context ID comparators	1
Single-Shot comparator resource	2
Counters	2
Cycle count size	12 bits
Number of sequencer states	4
Processor comparator inputs	0
External inputs	53
External outputs	4
External input selectors	4
Resource selector pairs	8
Instruction trace port size	32-bit
Data trace port size	64-bit
Instruction FIFO[cc]	128-byte with 32-bit output
Data FIFO	256 byte with 64-bit output
Claim tag bits	4

The following table shows the optional features of the ETM architecture that the ETM implements.

Table 14-2 ETM trace unit generation options implemented

Feature	Implemented
Trace Start/Stop block	Yes
Trace all branches option	Yes
Trace of conditional instructions	Yes
Cycle counting in instruction trace	Yes
Data trace supported	Yes
Data address comparison	Yes
OS Lock mechanism	Yes
Secure non-invasive debug	Yes
Context ID tracing	Yes

[cc] Instruction trace can be configured to take priority over data trace. See bit[10] of the TRCSTALLCTLR.

Table 14-2 ETM trace unit generation options implemented (continued)

Feature	Implemented
Trace output	Yes
Timestamp size	64-bit
Memory mapped access to ETM registers	Yes
External debugger access to ETM registers	Yes
System instruction access to ETM registers	No
VMID comparator support	Yes
ATB trigger support	Yes

14.3 ETM Event connectivity

This section describes how the Cortex-R52 ETM inputs and outputs are connected to the CTI and PMU.

The following table shows the connection of the ETM external inputs that come from the CTI and PMU.

Table 14-3 ETM External Input connections

Bits	Description
ETM External Input 0	CTI Trigger Output 4
ETM External Input 1	CTI Trigger Output 5
ETM External Input 2	CTI Trigger Output 6
ETM External Input 3	CTI Trigger Output 7
ETM External Input [48:4]	Performance Monitor **PMUEVENTx[44:0]**
ETM External Input 49	**ETMEXTIN[0]**
ETM External Input 50	**ETMEXTIN[1]**
ETM External Input 51	**ETMEXTIN[2]**
ETM External Input 52	**ETMEXTIN[3]**

The ETM external output resources are connected to the CTI, as the following table shows.

Table 14-4 ETM External Output connections to CTI

ETM output	CTI input
ETM External Output 0	CTI Trigger Input 4 and **ETMEXTOUT[0]**
ETM External Output 1	CTI Trigger Input 5 and **ETMEXTOUT[1]**
ETM External Output 2	CTI Trigger Input 6 and **ETMEXTOUT[2]**
ETM External Output 3	CTI Trigger Input 7 and **ETMEXTOUT[3]**

14.4 Operation

This section describes the ETM-Cortex-R52 IMPLEMENTATION DEFINED features. These features are IMPLEMENTATION DEFINED.

For information on the operation, see the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

This section contains the following subsections:

14.4.1 Implementation defined registers

There are two groups of ETM-Cortex-R52 registers:

- Registers that are completely defined by the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.
- Registers that are partly IMPLEMENTATION DEFINED.

14.4.2 Precise TraceEnable events

The **ViewInst** and **ViewData** are imprecise under certain conditions, with some implementation-defined exceptions. The only condition which ensures that **ViewInst** and **ViewData** are precise is that the enabling event condition is TRUE.

For more information, see the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

14.4.3 Parallel instruction execution

The Cortex-R52 processor supports parallel instruction execution. This means the macrocell can trace two instructions per cycle and 64 bits of data transfer.

If **ViewInst** is active for a cycle, the ETM:

- Always traces the first of any paired instructions.
- Always traces the second instruction if **ViewInst** is active for that instruction.
- Can trace data for any of the instructions traced.

14.4.4 Comparator features

The ETM implements data address comparison. There are eight address comparators that can be configured for either instruction or data address comparison.

See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* for a description of data address comparison.

14.4.5 Trace features

The ETM implements all of the ETMv4 trace features.

This means it supports:
- Data value and data address tracing.
- Data suppression.
- Cycle-accurate tracing.
- Timestamping.

See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* for descriptions of these features.

14.4.6 Packet formats

This section describes the packet types that the ETM instruction trace interface supports.

The ETM instruction trace interface does not support the following packet formats:

- Speculation resolution:
  - Mispredict packet.
  - Cancel format 2 and 3 packets.
- Conditional tracing:
  - All instruction format packets.
  - Result format 1 packet.
- Q packets.

Except the following data trace packet types, the ETM data trace interface supports all data trace packet types.
- P1 Format 6 and 7 packets.
- P2 Format 2 and 3 packets.

See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* for the trace packet format descriptions.

14.4.7 Resource selection

The ETM uses event selectors to control resources.

The ETM controls the following resources:
- Trace events (triggers and markers in the trace stream).
- Timestamp event.
- ViewInst event.
- ViewData event.
- Counter control.
- Sequencer state transitions.

Each event selector is configured to be sensitive to a resource selector pair, and one resource selector pair can control more than one event selector.

The ETM provides one fixed resource selector pair, with static values of 0 and 1, and seven configurable selector pairs. A resource selector pair provides a bitfield OR selector for resources in two different groups, with each group and a configurable boolean combination provided.

The following table shows the resources that can be selected for the instruction and data trace.

Table 14-5 Instruction and data resource selection

Group	Select	Resource
0b0000	0-3	External input selector 0-3
0b0010	0-1	Counter at zero 0-1
	4-7	Sequencer states 0-3
0b0011	1	Single-Shot comparator 0-1

Table 14-5 Instruction and data resource selection (continued)

Group	Select	Resource
0b0100	0-7	Single address comparator 0-7
0b0101	0-3	Address range comparator 0-3
0b0110	0	Context ID Comparator 0
0b0111	0	VMID Comparator 0

For example, the following figure shows the steps necessary to use a single address comparator to generate a trigger event and an ATB trigger. This example uses the first single resource selector that can be user-configured.

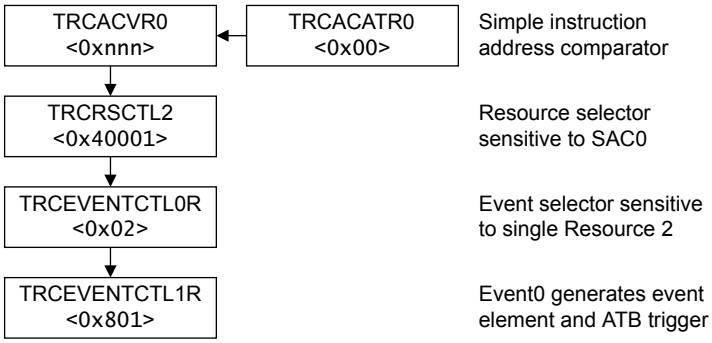

Figure 14-2 Trigger event resource selection

14.4.8 Trace flush behavior

Events that are observed by the ETM can be confirmed to have reached the trace bus output with the use of the ATB flush protocol. Both ATB ports must be flushed to determine when the trace infrastructure finishes capturing all the packets generated by the ETM.

The ETM internally flushes instruction and data trace together whenever either flush request is seen but does not guarantee that the trace data has drained from the ETM. When the processor enters a low power state, this also causes all trace to be output from the ETM.

14.4.9 Low power state behavior

When the processor enters a low power state there is a delay before the resources in the ETM become inactive.

This permits the last instruction executed to trigger a comparator, update the counter or sequencer, and the resultant event packet to be inserted in the specified trace stream. This event packet is presented on the trace bus before the ETM itself enters a low power state.

If an event packet is generated for a different reason, it is not guaranteed to be output before the ETM enters a low power state, but is traced when the processor leaves the low power state, if the ETM logic is not reset before this can occur.

The TRCEVENTCTL1R.LPOVERRIDE bit controls how a trace unit behaves in a low power state. If it is set to 1, trace unit low-power state behavior is overridden, that is, entry to a low-power state does not affect the trace unit resources or trace generation. In this case, the ETM resources remain active.

Related references

14.7.5 Event Control 1 Register on page 14-469

14.4.10 Cycle counter

The Cortex-R52 ETM uses a 12-bit cycle counter.

It does not count when non-invasive debug is disabled, or when the processor is in a low power state.

14.4.11 Non-architectural exceptions

Non-architectural behavior exceptions are indicated by the ETM.

The ETM indicates exceptions for the following non-architectural behaviors that use the following TYPE encodings:

`0b10001` ECC logic requires instruction trace to be replayed. This should not be relied on as providing trace of ECC behavior.

14.4.12 Trace synchronization

The ETM receives and combines all sources of trace synchronization requests to determine when synchronization is required. When synchronization is required, information is inserted in both trace streams as necessary (depending on whether data trace is active).

To decompress the trace streams, synchronization information in the data trace stream determines the alignment with synchronization information in the instruction stream. If the ETM is configured to trace only events in the data stream, you must configure the instruction trace stream to contain sufficient elements to permit the required data trace stream synchronization.

14.5 Modes of operation and execution

This section describes how to control the ETM programming and read and program the ETM registers.

This section contains the following subsections:

14.5.1 Use of the ETM main enable bit

When programming the ETM registers, you must enable all the changes at the same time.

For example, if the counter is reprogrammed, it might start to count based on incorrect events, before the trigger condition has been correctly set up.

To disable all trace operations during programming use:

- The ETM main enable in the TRCPRGCTLR, see *14.7.1 Programming Control Register* on page 14-464.
- The TRCSTATR to indicate the ETM status, see *14.7.2 Status Register* on page 14-464.

The following figure shows the procedure to use.

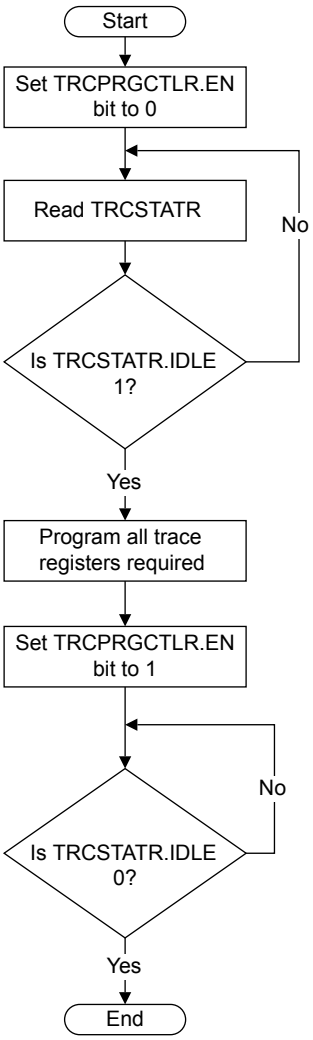

Figure 14-3 Programming ETM registers

The Cortex-R52 processor does not have to be in Debug state while you program the ETM registers.

14.5.2 Programming and reading ETM registers

To access the ETM registers, use the external APB interface. This provides a direct method of programming the ETM.

14.5.3 External register access permissions

Whether access is permitted to a register depends on:

- If the processor is powered up.
- The state of the OS Lock and Software Lock.
- The state of the debug authentication inputs to the processor.

The behavior that is specific to each register and the type of access to the register is not described in this document. For a detailed description of these features and their effects on the registers, see the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

The register descriptions provided in this section describe whether each register is read/write or read-only.

14.6 Register summary

This section summarizes the ETM registers.

For full descriptions of the ETM registers, see:

- *14.7 Register descriptions* on page 14-463, for the implementation defined registers.

Table 14-6 ETM register summary on page 14-459 lists the ETM registers in numerical order and describes each register.

The register table includes additional information about each register:

- The register access type. This is read-only, write-only, or read and write.
- The base offset address of the register. The base offset of a register is always four times its register number.
- Additional information about the implementation of the register, where appropriate.

Note

- Registers not listed here are not implemented. Reading a non-implemented register address returns 0. Writing to a non-implemented register address has no effect.
- In the following table:
  - The Reset value column shows the value of the register immediately after an ETM reset. For read-only registers, every read of the register returns this value.
  - Access type is described as follows:

    RW Read and write.

    RO Read only.

    WO Write only.
- The value of **PADDRDBG31** and the CoreSight Software lock restrict write accesses to the ETM registers through the memory-mapped external debug interface. When **PADDRDBG31** is driven LOW, the CoreSight software lock must be used to unlock write permissions for the ETM registers. When **PADDRDBG31** is driven HIGH, the software lock does not restrict the write access permissions of the ETM registers. The software lock is controlled with the Lock Access register.

All ETM registers are 32 bits wide.

Table 14-6 ETM register summary

Register number	Base offset	Name	Type	Reset	Description
1	0x004	TRCPRGCTLR	RW	0x00000000	*14.7.1 Programming Control Register* on page 14-464
3	0x00C	TRCSTATR	RO	UNK	*14.7.2 Status Register* on page 14-464
4	0x010	TRCCONFIGR	RW	UNK	*14.7.3 Trace Configuration Register* on page 14-465
8	0x020	TRCEVENTCTL0R	RW	UNK	*14.7.4 Event Control 0 Register* on page 14-468
9	0x024	TRCEVENTCTL1R	RW	UNK	*14.7.5 Event Control 1 Register* on page 14-469
11	0x02C	TRCSTALLCTLR	RW	UNK	*14.7.6 Stall Control Register* on page 14-470
12	0x030	TRCTSCTLR	RW	UNK	*14.7.7 Global Timestamp Control Register* on page 14-472
13	0x034	TRCSYNCPR	RW	UNK	*14.7.8 Synchronization Period Register* on page 14-473
14	0x038	TRCCCCTLR	RW	UNK	*14.7.9 Cycle Count Control Register* on page 14-474

Table 14-6 ETM register summary (continued)

Register number	Base offset	Name	Type	Reset	Description
15	0x03C	TRCBBCTLR	RW	UNK	*14.7.10 Branch Broadcast Control Register* on page 14-475
16	0x040	TRCTRACEIDR	RW	UNK	*14.7.11 Trace ID Register* on page 14-476
32	0x080	TRCVICTLR	RW	UNK	*14.7.12 ViewInst Main Control Register* on page 14-476
33	0x084	TRCVIIECTLR	RW	UNK	*14.7.13 ViewInst Include/Exclude Control Register* on page 14-478
34	0x088	TRCVISSCTLR	RW	UNK	*14.7.14 ViewInst Start/Stop Control Register* on page 14-479
40	0x0A0	TRCVDCTLR	RW	UNK	*14.7.15 ViewData Main Control Register* on page 14-480
41	0x0A4	TRCVDSACCTLR	RW	UNK	*14.7.16 ViewData Include/Exclude Single Address Comparator Register* on page 14-481
42	0x0A8	TRCVDARCCTLR	RW	UNK	*14.7.17 ViewData Include/Exclude Address Range Comparator Register* on page 14-482
64-66	0x100-0x108	TRCSEQEVR0-2	RW	UNK	*14.7.18 Sequencer State Transition Control Registers, n=0-2* on page 14-483
70	0x118	TRCSEQRSTEVR	RW	UNK	*14.7.19 Sequencer Reset Control Register* on page 14-484
71	0x11C	TRCSEQSTR	RW	UNK	*14.7.20 Sequencer State Register* on page 14-485
72	0x120	TRCEXTINSELR	RW	UNK	*14.7.21 External Input Select Register* on page 14-486
80-81	0x140-0x144	TRCCNTRLDVR0-1	RW	UNK	*14.7.22 Counter Reload Value Registers, n=0-1* on page 14-487
84-85	0x150-0x154	TRCCNTCTLR0-1	RW	UNK	*14.7.23 Counter Control Registers 0-1* on page 14-488
88-89	0x160-0x164	TRCCNTVR0-1	RW	UNK	*14.7.24 Counter Value Registers, n=0-1* on page 14-489
96	0x180	TRCIDR8	RO	0x00000001	*14.7.25 ID Registers, n=8-13* on page 14-490
97	0x184	TRCIDR9	RO	0x00000020	
98	0x188	TRCIDR10	RO	0x00000002	
99	0x18C	TRCIDR11	RO	0x00000000	
100	0x190	TRCIDR12	RO	0x00000001	
101	0x194	TRCIDR13	RO	0x00000000	
112	0x1C0	TRCIMSPEC0	RW	0x00000000	*14.7.26 Implementation Specific Register 0* on page 14-493
120	0x1E0	TRCIDR0	RO	0x08000EFF	*14.7.27 ID Register 0* on page 14-493
121	0x1E4	TRCIDR1	RO	0x4100F424	*14.7.28 ID Register 1* on page 14-495
122	0x1E8	TRCIDR2	RO	0x00420484	*14.7.29 ID Register 2* on page 14-496

Table 14-6 ETM register summary (continued)

Register number	Base offset	Name	Type	Reset	Description
123	0x1EC	TRCIDR3	RO	0x0D700004	*14.7.30 ID Register 3* on page 14-497
124	0x1F0	TRCIDR4	RO	0x11270124	*14.7.31 ID Register 4* on page 14-499
125	0x1F4	TRCIDR5	RO	0x28C7081A	*14.7.32 ID Register 5* on page 14-500
130-143	0x208-0x23C	TRCRSCTLR2-15	RW	UNK	*14.7.33 Resource Selection Registers, n=2-15* on page 14-501
160-161	0x280-0x284	TRCSSCCR0-1	RW	UNK	*14.7.34 Single-shot Comparator Control Registers, n=0-1* on page 14-502
168-169	0x2A0-0x2A4	TRCSSCSR0-1	RW	UNK	*14.7.35 Single-shot Comparator Status Registers n=0-1* on page 14-503
192	0x300	TRCOSLAR	WO	UNK	*14.7.36 OS Lock Access Register* on page 14-505
193	0x304	TRCOSLSR	RO	0x0000000A	*14.7.37 OS Lock Status Register* on page 14-505
196	0x310	TRCPDCR	RW	0x00000000	*14.7.38 Power Down Control Register* on page 14-506
197	0x314	TRCPDSR	RO	UNK	*14.7.39 Power Down Status Register* on page 14-507
256-270	0x400-0x438	TRCACVR0-7	RW	UNK	*14.7.40 Address Comparator Value Registers, n=0-7* on page 14-508
288-302	0x480-0x4B8	TRCACATR0-7	RW	UNK	*14.7.41 Address Comparator Access Type Registers, n=0-7* on page 14-509
320-324	0x500-0x510	TRCDVCVR0-1	RW	UNK	*14.7.42 Data Value Comparator Value Registers, n=0-1* on page 14-513
352-356	0x580-0x590	TRCDVCMR0-1	RW	UNK	*14.7.43 Data Value Comparator Mask Registers, n=0-1* on page 14-513
384	0x600	TRCCIDCVR0	RW	UNK	*14.7.44 Context ID Comparator Value Registers 0* on page 14-514
400	0x640	TRCVMIDCVR0	RW	UNK	*14.7.45 Virtual Context Identifier Comparator Value Register* on page 14-514
416	0x680	TRCCIDCCTLR0	RW	UNK	*14.7.46 Context ID Comparator Control Register 0* on page 14-515
960	0xF00	TRCITCTRL	RO	0x00000000	*14.7.47 Integration Mode Control Register* on page 14-516
1000	0xFA0	TRCCLAIMSET	RW	0x0000000F	*14.7.48 Claim Tag Set Register* on page 14-517
1001	0xFA4	TRCCLAIMCLR	RW	0x00000000	*14.7.49 Claim Tag Clear Register* on page 14-518
1002	0xFA8	TRCDEVAFF0	RO	-[cd]	*14.7.50 TRCDEVAFF0, Device Affinity Register 0* on page 14-519
1003	0xFAC	TRCDEVAFF1	RO	0x00000000	*14.7.51 TRCDEVAFF1, Device Affinity Register 1* on page 14-519
1004	0xFB0	TRCLAR	WO	UNK	*14.7.52 Software Lock Access Register* on page 14-519

[cd] The reset value depends on the cluster and processor ID. See *3.3.78 Multiprocessor Affinity Register* on page 3-167

Table 14-6 ETM register summary (continued)

Register number	Base offset	Name	Type	Reset	Description
1005	0xFB4	TRCLSR	RO	0x00000003	*14.7.53 Software Lock Status Register* on page 14-520
1006	0xFB8	TRCAUTHSTATUS	RO	UNK	*14.7.54 Authentication Status Register* on page 14-521
1007	0xFBC	TRCDEVARCH	RO	0x47724A13	*14.7.55 Device Architecture Register* on page 14-522
1010	0xFC8	TRCDEVID	RO	0x00000000	*14.7.56 Device ID Register* on page 14-523
1011	0xFCC	TRCDEVTYPE	RO	0x00000013	*14.7.57 Device Type Register* on page 14-523
1012	0xFD0	TRCPIDR4	RO	-[ce]	*14.7.58 Peripheral Identification Registers* on page 14-524
1013-1015	0xFD4-0xFDC	TRCPIDR[5,6,7]	RO		
1016	0xFE0	TRCPIDR0	RO		
1017	0xFE4	TRCPIDR1	RO		
1018	0xFE8	TRCPIDR2	RO		
1019	0xFEC	TRCPIDR3	RO		
1020-1023	0xFF0-0xFFC	TRCCIDR0-3	RO	-[cf]	*14.7.59 Component Identification Registers* on page 14-527

[ce] See *14.7.58 Peripheral Identification Registers* on page 14-524
[cf] *14.7.59 Component Identification Registers* on page 14-527

14.7 Register descriptions

This section describes the ETM-Cortex-R52 registers.

This section contains the following subsections:

14.7.1 Programming Control Register

The TRCPRGCTLR enables the ETM.

Usage constraints

This register is read/write. Might ignore writes when the trace unit is enabled or not idle.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCPRGCTLR is a 32-bit register.

The following figure shows the TRCPRGCTLR bit assignments.

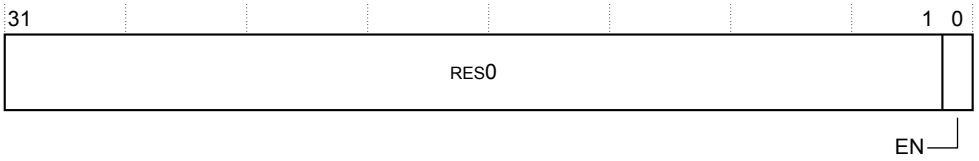

Figure 14-4 TRCPRGCTLR bit assignments

The following table shows the TRCPRGCTLR bit assignments.

Table 14-7 TRCPRGCTLR bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	EN	Trace program enable: `0` Enabled when necessary to process APB accesses, or drain any already generated trace. No trace is generated and all resources are inactive. This is the value after reset. `1` Clocks are enabled except for when the processor is in a low power state, or non-invasive debug is disabled, and all trace has been drained. The trace unit is enabled.

TRCPRGCTLR can be accessed through the memory mapped interface:

Table 14-8 TRCPRGCTLR access information

Component	Offset	Reset
ETM	`0x004`	`0x00000000`

14.7.2 Status Register

The TRCSTATR indicates the ETM status.

Usage constraints

This register is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCSTATR is a 32-bit register.

The following figure shows the TRCSTATR bit assignments.

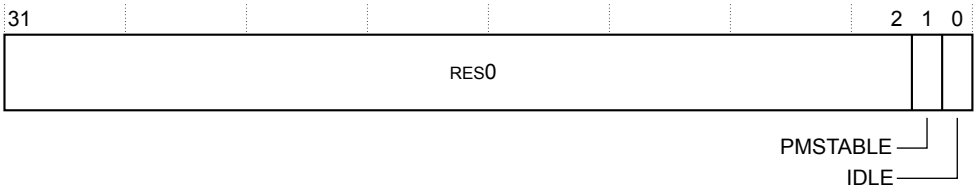

Figure 14-5 TRCSTATR bit assignments

The following table shows the TRCSTATR bit assignments.

Table 14-9 TRCSTATR bit assignments

Bits	Name	Function
[31:2]	-	Reserved, RES0.
[1]	PMSTABLE	Indicates whether the ETM registers are stable and can be read: `0` The programmers model is not stable. `1` The programmers model is stable.
[0]	IDLE	Indicates that the trace unit is inactive: `0` The ETM is not idle. `1` The ETM is idle.

TRCSTATR can be accessed through the memory mapped interface:

Table 14-10 TRCSTATR access information

Component	Offset	Reset
ETM	`0x00C`	UNK

14.7.3 Trace Configuration Register

The TRCCONFIGR sets the basic tracing options for the trace unit.

Usage constraints

This register is read/write and must always be programmed as part of the trace unit initialization.

Only accepts writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCCONFIGR is a 32-bit register.

The following figure shows the TRCCONFIGR bit assignments.

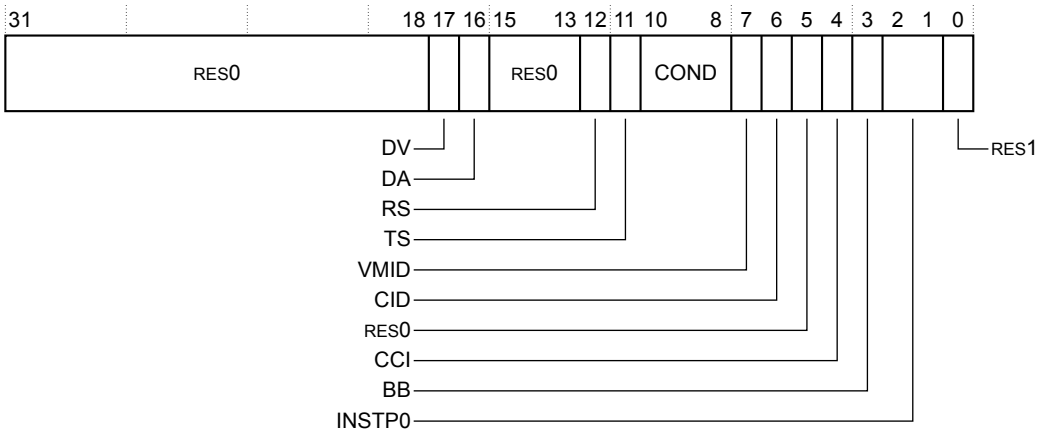

Figure 14-6 TRCCONFIGR bit assignments

The following table shows the TRCCONFIGR bit assignments.

Table 14-11 TRCCONFIGR bit assignments

Bits	Name	Function
[31:18]	-	Reserved, RES0.
[17]	DV	Data value tracing: 0 Data value tracing disabled. 1 Data value tracing enabled.
[16]	DA	Data address tracing: 0 Data address tracing disabled. 1 Data address tracing enabled.
[15:13]	-	Reserved, RES0.
[12]	RS	Return stack enable: 0 Return stack disabled. 1 Return stack enabled.
[11]	TS	Global timestamp tracing: 0 Global timestamp tracing disabled. 1 Global timestamp tracing enabled. For more global timestamping options, see *14.7.7 Global Timestamp Control Register* on page 14-472.

Table 14-11 TRCCONFIGR bit assignments (continued)

Bits	Name	Function
[10:8]	COND	Conditional instruction tracing. The supported values are: `0b000` Conditional instruction tracing disabled. `0b001` Conditional load instructions are traced. `0b010` Conditional store instructions are traced. `0b011` Conditional load and store instructions are traced. `0b111` All conditional instructions are traced. All other values are Reserved.
[7]	VMID	Virtual machine ID tracing: `0` VMID tracing is disabled. `1` VMID tracing is enabled.
[6]	CID	Context ID tracing: `0` Context ID tracing is disabled. `1` Context ID tracing is enabled.
[5]	-	Reserved, RES0.
[4]	CCI	Cycle counting in instruction trace: `0` Cycle counting in instruction trace disabled. `1` Cycle counting in instruction trace enabled. For more cycle counting options, see *14.7.9 Cycle Count Control Register* on page 14-474.
[3]	BB	Branch broadcast mode: `0` Branch broadcast mode disabled. `1` Branch broadcast mode enabled. For more branch broadcast mode options, see *14.7.10 Branch Broadcast Control Register* on page 14-475.
[2:1]	INSTP0	Determines the instructions that generate P0 elements: `0b00` Only branches are P0 elements. `0b01` Load instructions and branches are P0 elements. `0b10` Store instructions and branches are P0 elements. `0b11` Load and store instructions and branches are P0 elements.
[0]	-	Reserved, RES1.

TRCCONFIGR can be accessed through the memory mapped interface:

Table 14-12 TRCCONFIGR access information

Component	Offset	Reset
ETM	`0x010`	UNK

14.7.4 Event Control 0 Register

The TRCEVENTCTL0R controls the tracing of events in the trace streams. The events also drive the external outputs from the ETM.

Usage constraints

This register is read/write and must always be programmed as part of the trace unit initialization.

Only accepts writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCEVENTCTL0R is a 32-bit register.

The following figure shows the TRCEVENTCTL0R bit assignments.

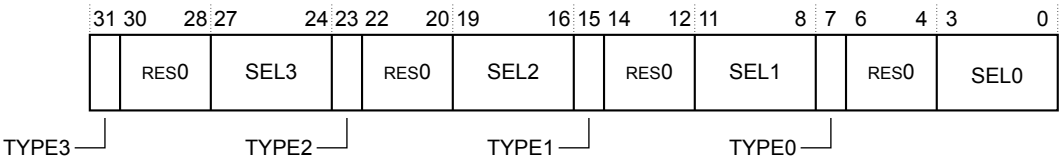

Figure 14-7 TRCEVENTCTL0R bit assignments

The following table shows the TRCEVENTCTL0R bit assignments.

Table 14-13 TRCEVENTCTL0R bit assignments

Bits	Name	Function
[31]	TYPE3	Selects the resource type for 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, selects a single selected resource from 0-15 defined by SEL3[3:0]. When TYPE3 is 1, selects a Boolean combined resource pair from 0-7 defined by SEL3[2:0].
[23]	TYPE2	Selects the resource type for 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, selects a single selected resource from 0-15 defined by SEL2[3:0]. When TYPE2 is 1, selects a Boolean combined resource pair from 0-7 defined by SEL2[2:0].

Table 14-13 TRCEVENTCTL0R bit assignments (continued)

Bits	Name	Function
[15]	TYPE1	Selects the resource type for 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, selects a single selected resource from 0-15 defined by SEL1[3:0]. When TYPE1 is 1, selects a Boolean combined resource pair from 0-7 defined by SEL1[2:0].
[7]	TYPE0	Selects the resource type for 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 SEL0[3:0]. When TYPE0 is 1, selects a Boolean combined resource pair from 0-7 defined by SEL0[2:0].

TRCEVENTCTL0R can be accessed through the memory mapped interface:

Table 14-14 TRCEVENTCTL0R access information

Component	Offset	Reset
ETM	0x020	UNK

14.7.5 Event Control 1 Register

The TRCEVENTCTL1R controls how the events selected by TRCEVENTCTL0R behave.

Usage constraints

This register is read/write and must always be programmed as part of the trace unit initialization.

Only accepts writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCEVENTCTL1R is a 32-bit register.

The following figure shows the TRCEVENTCTL1R bit assignments.

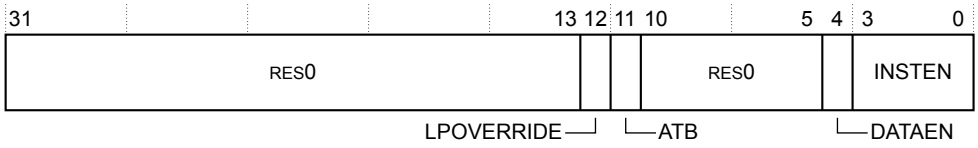

Figure 14-8 TRCEVENTCTL1R bit assignments

The following table shows the TRCEVENTCTL1R bit assignments.

Table 14-15 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.
[10:5]	-	Reserved, RES0.
[4]	DATAEN	Enables generation of an event element in the data trace stream when the selected event occurs: 0 Event does not cause an event element. 1 Event causes an event element.
[3:0]	INSTEN	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.

TRCEVENTCTL1R can be accessed through the memory mapped interface:

Table 14-16 TRCEVENTCTL1R access information

Component	Offset	Reset
ETM	0x024	UNK

14.7.6 Stall Control Register

The TRCSTALLCTLR enables the ETM to stall the Cortex-R52 processor if the ETM FIFO goes over the programmed level to minimize risk of overflow.

Usage constraints

Only accepts writes when the trace unit is disabled.

This register is read/write and must always be programmed as part of the trace unit initialization.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCSTALLCTLR is a 32-bit register.

The following figure shows the TRCSTALLCTLR bit assignments.

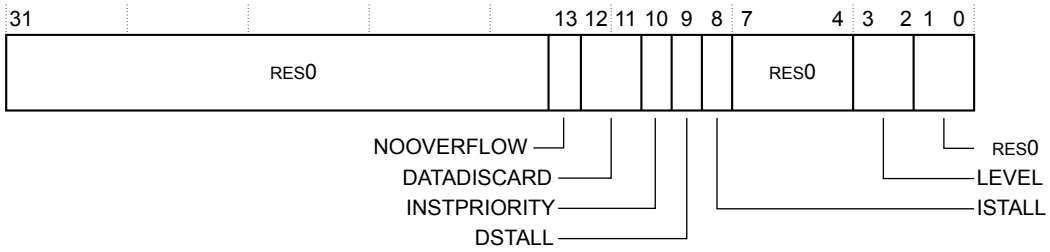

Figure 14-9 TRCSTALLCTLR bit assignments

The following table shows the TRCSTALLCTLR bit assignments.

Table 14-17 TRCSTALLCTLR bit assignments

Bits	Name	Function
[31:14]	-	Reserved, RES0.
[13]	NOOVERFLOW	Trace overflow prevention bit. This field is RES0.
[12:11]	DATADISCARD	Sets the priority of data trace components, enabling the ETM to discard some data if the data trace buffer space is less than LEVEL: `0b00` The trace unit must not discard any data trace elements. `0b01` The trace unit can discard P1 and P2 elements associated with data loads. `0b10` The trace unit can discard P1 and P2 elements associated with data stores. `0b11` The trace unit can discard P1 and P2 elements associated with both data loads and stores.
[10]	INSTPRIORITY	Prioritize instruction trace if instruction trace buffer space is less than LEVEL: `0` The trace unit must not prioritize instruction trace. `1` The trace unit can prioritize instruction trace.
[9]	DSTALL	Stall processor based on data trace buffer space: `0` The trace unit must not stall the processor. `1` The trace unit can stall the processor.
[8]	ISTALL	Stall processor based on instruction trace buffer space: `0` The trace unit must not stall the processor. `1` The trace unit can stall the processor.
[7:4]	-	Reserved, RES0.

Table 14-17 TRCSTALLCTLR bit assignments (continued)

Bits	Name	Function
[3:2]	LEVEL	Threshold at which stalling becomes active. This provides four levels. This level can be varied to optimize the level of invasion caused by stalling, balanced against the risk of a FIFO overflow: `0b00` Zero invasion. This setting has a greater risk of a FIFO overflow. `0b11` Maximum invasion occurs but there is less risk of a FIFO overflow.
[1:0]	-	Reserved, RES0.

TRCSTALLCTLR can be accessed through the memory mapped interface:

Table 14-18 TRCSTALLCTLR access information

Component	Offset	Reset
ETM	`0x02C`	UNK

14.7.7 Global Timestamp Control Register

The TRCTSCTLR controls the insertion of global timestamps into the trace streams. A timestamp is always inserted into the instruction trace stream, and also in the data trace stream if any data tracing is enabled.

Usage constraints

Only accepts writes when the trace unit is disabled.

This register is read/write and must always be programmed as part of the trace unit initialization.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCTSCTLR is a 32-bit register.

The following figure shows the TRCTSCTLR bit assignments.

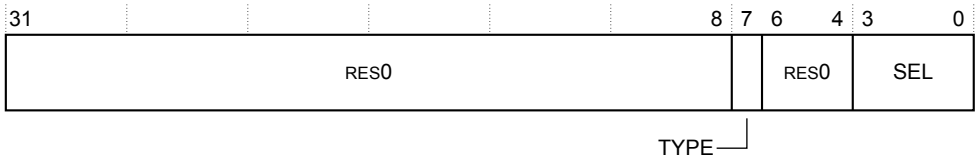

Figure 14-10 TRCTSCTLR bit assignments

The following table shows the TRCTSCTLR bit assignments.

Table 14-19 TRCTSCTLR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7]	TYPE	Selects the resource type: `0` Single selected resource. `1` Boolean combined resource pair.

Table 14-19 TRCTSCTLR bit assignments (continued)

Bits	Name	Function
[6:4]	-	Reserved, RES0.
[3:0]	SEL	Selects the resource number, based on the value of TYPE: When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0]. When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TRCTSCTLR can be accessed through the memory mapped interface:

Table 14-20 TRCTSCTLR access information

Component	Offset	Reset
ETM	0x030	UNK

14.7.8 Synchronization Period Register

The TRCSYNCPR specifies the period of trace synchronization of the trace streams. TRCSYNCPR defines a number of bytes of trace between requests for trace synchronization. This value is always a power of two.

Usage constraints

Only accepts writes when the trace unit is disabled.

This register is read/write and must always be programmed as part of the trace unit initialization.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCSYNCPR is a 32-bit register.

The following figure shows the TRCSYNCPR bit assignments.

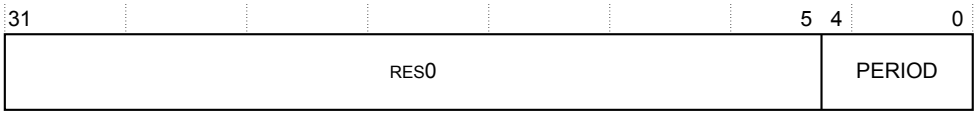

Figure 14-11 TRCSYNCPR bit assignments

The following table shows the TRCSYNCPR bit assignments.

Table 14-21 TRCSYNCPR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0.
[4:0]	PERIOD	Defines the number of bytes of trace between trace 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 trace synchronization requests, but does not disable other trace synchronization requests. • The minimum value that can be programmed, other than zero, is 8, providing a minimum trace synchronization period of 256 bytes. • The maximum value is 20, providing a maximum trace synchronization period of 2^{20} bytes.

TRCSYNCPR can be accessed through the memory mapped interface:

Table 14-22 TRCSYNCPR access information

Component	Offset	Reset
ETM	0x034	UNK

14.7.9 Cycle Count Control Register

The TRCCCCTLR sets the threshold value for instruction trace cycle counting. The threshold represents the minimum interval between cycle count trace packets.

Usage constraints

Only accepts writes when the trace unit is disabled.

This register is read/write and must always be programmed as part of the trace unit initialization.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCCCCTLR is a 32-bit register.

The following figure shows the TRCCCCTLR bit assignments.

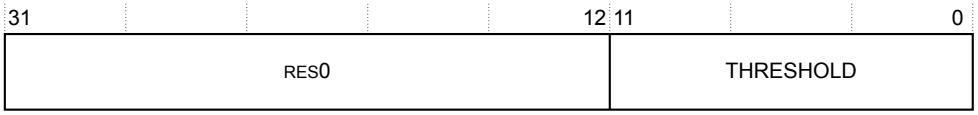

Figure 14-12 TRCCCCTLR bit assignments

The following table shows the TRCCCCTLR bit assignments.

Table 14-23 TRCCCCTLR bit assignments

Bits	Name	Function
[31:12]	-	Reserved, RES0.
[11:0]	THRESHOLD	Instruction trace cycle count threshold.

TRCCCCTLR can be accessed through the memory mapped interface:

Table 14-24 TRCCCCTLR access information

Component	Offset	Reset
ETM	0x038	UNK

14.7.10 Branch Broadcast Control Register

The TRCBBCTLR controls how branch broadcasting behaves, and enables branch broadcasting to be enabled for certain memory regions.

Usage constraints

This register is read/write. Only accepts writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCBBCTLR is a 32-bit register.

The following figure shows the TRCBBCTLR bit assignments.

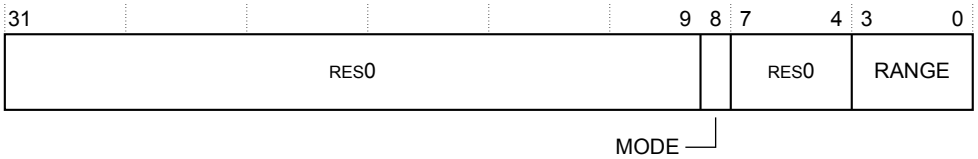

Figure 14-13 TRCBBCTLR bit assignments

The following table shows the TRCBBCTLR bit assignments.

Table 14-25 TRCBBCTLR bit assignments

Bits	Name	Function
[31:9]	-	Reserved, RES0.
[8]	MODE	Selects mode: 0 Exclude mode. The Address Range Comparators defined by the RANGE field indicate address ranges where branch broadcasting is not enabled. Selecting no ranges results in branch broadcasting being enabled over the whole memory map. 1 Include mode. The Address Range Comparators defined by the RANGE field indicate address ranges where branch broadcasting is enabled. Setting RANGE to all zeroes is UNPREDICTABLE when in Include mode.
[7:4]	-	Reserved, RES0.
[3:0]	RANGE	Selects Address Range Comparators to control where branch broadcasting is enabled. One bit is provided for each implemented Address Range Comparator.

TRCBBCTLR can be accessed through the memory mapped interface:

Table 14-26 TRCBBCTLR access information

Component	Offset	Reset
ETM	0x03C	UNK

14.7.11 Trace ID Register

The TRCTRACEIDR sets the trace ID on the trace bus. Controls two trace IDs, one for instruction trace and one for data trace.

Usage constraints

In a CoreSight system, writing of reserved trace ID values, `0x00` and `0x70-0x7F`, is UNPREDICTABLE. This register is read/write and must always be programmed as part of the trace unit initialization.

Only accepts writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCTRACEIDR is a 32-bit register.

The following figure shows the TRCTRACEIDR bit assignments.

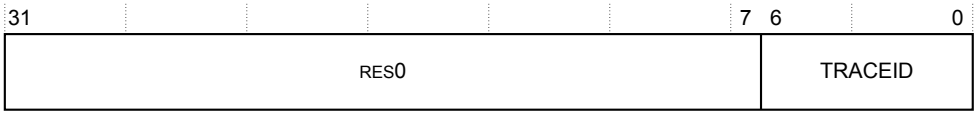

Figure 14-14 TRCTRACEIDR bit assignments

The following table shows the TRCTRACEIDR bit assignments.

Table 14-27 TRCTRACEIDR bit assignments

Bits	Name	Function
[31:7]	-	Reserved, RES0.
[6:0]	TRACEID	Trace ID value. When only instruction tracing is enabled, this provides the trace ID. When data tracing is enabled, this field must be written with bit[0] set to 0. The instruction and data trace streams use adjacent trace ID values: • The instruction trace stream uses the trace ID TRACEID[6:0]. • The data value trace stream uses the trace ID TRACEID{[6:1],1}.

TRCTRACEIDR can be accessed through the memory mapped interface:

Table 14-28 TRCTRACEIDR access information

Component	Offset	Reset
ETM	`0x040`	UNK

14.7.12 ViewInst Main Control Register

The TRCVICTLR controls instruction trace filtering.

Usage constraints

This registers is read/write. Only accepts writes when the trace unit is disabled.

Only returns stable data when TRCSTATR.PMSTABLE is 1.

Must be programmed to set the value of the SSSTATUS bit, that sets the state of the start-stop logic.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCVICTLR is a 32-bit register.

The following figure shows the TRCVICTLR bit assignments.

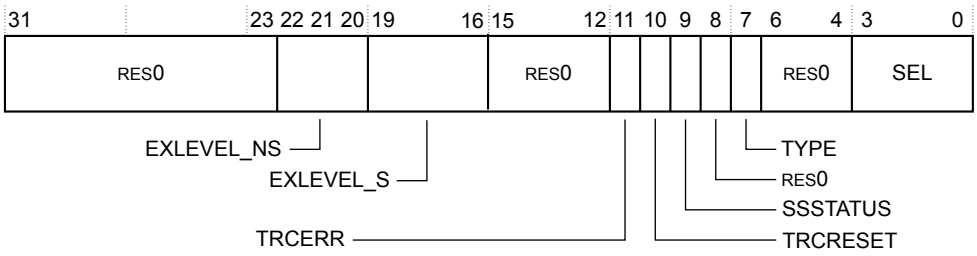

Figure 14-15 TRCVICTLR bit assignments

The following table shows the TRCVICTLR bit assignments.

Table 14-29 TRCVICTLR bit assignments

Bits	Value	Function
[31:23]	-	Reserved, RES0.
[22]	EXLEVEL_NS	Disables tracing for the specified exception level in EL2.
[21]	EXLEVEL_NS	Disables tracing for the specified exception level in EL1.
[20]	EXLEVEL_NS	Disables tracing for the specified exception level in EL0.
[19:16]	EXLEVEL_S	This field is RES0.
[15:12]	-	Reserved, RES0.
[11]	TRCERR	Selects whether a system error exception must always be traced: `0` System error exception is traced only if the instruction or exception immediately before the system error exception is traced. `1` System error exception is always traced regardless of the value of ViewInst.
[10]	TRCRESET	Selects whether a reset exception must always be traced: `0` Reset exception is traced only if the instruction or exception immediately before the reset exception is traced. `1` Reset exception is always traced regardless of the value of ViewInst.
[9]	SSSTATUS	Indicates the current status of the start/stop logic: `0` Start/stop logic is in the stopped state. `1` Start/stop logic is in the started state.
[8]	-	Reserved, RES0.

Table 14-29 TRCVICTLR bit assignments (continued)

Bits	Value	Function
[7]	TYPE	Selects the resource type: `0` Single selected resource. `1` Boolean combined resource pair.
[6:4]	-	Reserved, RES0.
[3:0]	SEL	Selects the resource number, based on the value of TYPE: When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0]. When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TRCVICTLR can be accessed through the memory mapped interface:

Table 14-30 TRCVICTLR access information

Component	Offset	Reset
ETM	`0x080`	UNK

14.7.13 ViewInst Include/Exclude Control Register

The TRCVIIECTLR defines the address range comparators that control the ViewInst Include/Exclude control.

Usage constraints

This register is read/write. Only accepts writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCVIIECTLR is a 32-bit register.

The following figure shows the TRCVIIECTLR bit assignments.

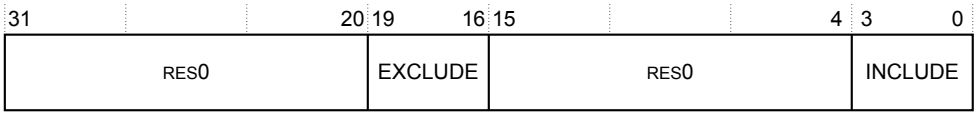

Figure 14-16 TRCVIIECTLR bit assignments

The following table shows the TRCVIIECTLR bit assignments.

Table 14-31 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 Address Range Comparator.

Table 14-31 TRCVIIECTLR bit assignments (continued)

Bits	Name	Function
[15:4]	-	Reserved, RES0.
[3:0]	INCLUDE	Defines the address range comparators for ViewInst include control. Selecting no include comparators indicates that all instructions must be included. The exclude control indicates which ranges must be excluded. One bit is provided for each Address Range Comparator.

TRCVIIECTLR can be accessed through the memory mapped interface:

Table 14-32 TRCVIIECTLR access information

Component	Offset	Reset
ETM	0x084	UNK

14.7.14 ViewInst Start/Stop Control Register

The TRCVISSCTLR defines the single address comparators that control the ViewInst Start/Stop logic.

Usage constraints

Only accepts writes when the trace unit is disabled.

This register is read/write and must always be programmed as part of the trace unit initialization.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCVISSCTLR is a 32-bit register.

The following figure shows the TRCVISSCTLR bit assignments.

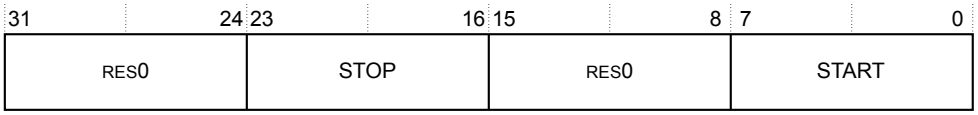

Figure 14-17 TRCVISSCTLR bit assignments

The following table shows the TRCVISSCTLR bit assignments.

Table 14-33 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 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 single address comparator.

TRCVISSCTLR can be accessed through the memory mapped interface:

Table 14-34 TRCVISSCTLR access information

Component	Offset	Reset
ETM	0x088	UNK

14.7.15 ViewData Main Control Register

The TRCVDCTLR controls data trace filtering.

Usage constraints

Only accepts writes when the trace unit is disabled.

This register is read/write and must be programmed when data tracing is enabled.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCVDCTLR is a 32-bit register.

The following figure shows the TRCVDCTLR bit assignments.

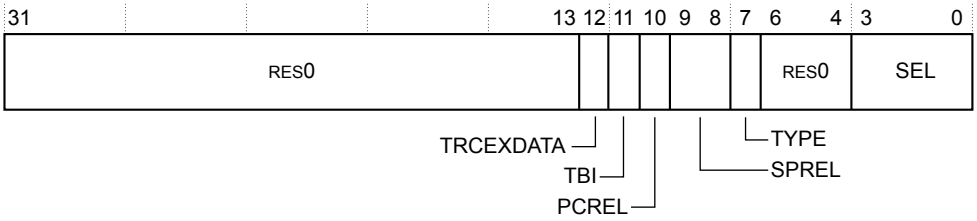

Figure 14-18 TRCVDCTLR bit assignments

The following table shows the TRCVDCTLR bit assignments.

Table 14-35 TRCVDCTLR bit assignments

Bits	Name	Function
[31:13]	-	Reserved, RES0.
[12]	TRCEXDATA	This field is RES0.
[11]	TBI	This field is RES0.
[10]	PCREL	Controls tracing of data for transfers that are relative to the PC: 0 Tracing of PC-relative transfers is unaffected. 1 Do not trace either the address or value portions of PC-relative transfers.
[9:8]	SPREL	Controls tracing of data for transfers that are relative to the *Stack Pointer* (SP): 0b00 Tracing of SP-relative transfers is unaffected. 0b01 Reserved. 0b10 Do not trace the address portion of SP-relative transfers. A P1 data address element is generated if data value tracing is enabled. 0b11 Do not trace either the address or value portions of SP-relative transfers.

Table 14-35 TRCVDCTLR bit assignments (continued)

Bits	Name	Function
[7]	TYPE	Selects the resource type: `0` Single selected resource. `1` Boolean combined resource pair.
[6:4]	-	Reserved, RES0.
[3:0]	SEL	Selects the resource number, based on the value of TYPE: When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0]. When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TRCVDCTLR can be accessed through the memory mapped interface:

Table 14-36 TRCVDCTLR access information

Component	Offset	Reset
ETM	`0x0A0`	UNK

14.7.16 ViewData Include/Exclude Single Address Comparator Register

The TRCVDSACCTLR defines the single address comparators that control the ViewData Include/ Exclude control.

Usage constraints

Only accepts writes when the trace unit is disabled. This register is read/write and must be programmed when data tracing is enabled and one or more address comparators are implemented.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCVDSACCTLR is a 32-bit register.

The following figure shows the TRCVDSACCTLR bit assignments.

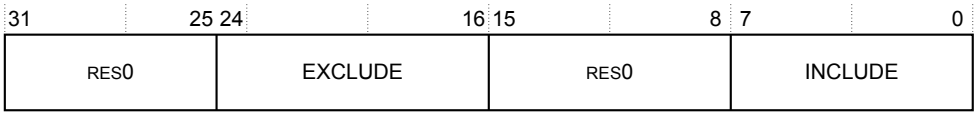

Figure 14-19 TRCVDSACCTLR bit assignments

The following table shows the TRCVDSACCTLR bit assignments.

Table 14-37 TRCVDSACCTLR bit assignments

Bits	Name	Function
[31:25]	-	Reserved, RES0.
[24:16]	EXCLUDE	Defines the single address comparators for ViewData exclude control. One bit is provided for each address comparator.

Table 14-37 TRCVDSACCTLR bit assignments (continued)

Bits	Name	Function
[15:8]	-	Reserved, RES0.
[7:0]	INCLUDE	Defines the single address comparators for ViewData include control. One bit is provided for each address comparator.

TRCVDSACCTLR can be accessed through the memory mapped interface:

Table 14-38 TRCVDSACCTLR access information

Component	Offset	Reset
ETM	0x0A4	UNK

14.7.17 ViewData Include/Exclude Address Range Comparator Register

The TRCVDARCCTLR defines the address range comparators that control the ViewData Include/Exclude control.

Usage constraints

Only accepts writes when the trace unit is disabled. This register is read/write and must be programmed when data tracing is enabled and one or more address comparators are implemented.

Traps and Enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCVDARCCTLR is a 32-bit register.

The following figure shows the TRCVDARCCTLR bit assignments.

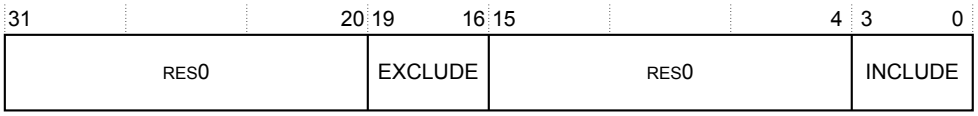

Figure 14-20 TRCVDARCCTLR bit assignments

The following table shows the TRCVDARCCTLR bit assignments.

Table 14-39 TRCVDARCCTLR bit assignments

Bits	Name	Function
[31:20]	-	Reserved, RES0
[19:16]	EXCLUDE	Defines the address range comparators for ViewData exclude control.One bit is provided for each address range comparator.
[15:4]	-	Reserved, RES0.
[3:0]	INCLUDE	Defines the address range comparators for ViewData include control. One bit is provided for each address range comparator.

TRCVDARCCTLR can be accessed through the memory mapped interface:

Table 14-40 TRCVDARCCTLR access information

Component	Offset	Reset
ETM	0x0A8	UNK

14.7.18 Sequencer State Transition Control Registers, n=0-2

The TRCSEQEVRn registers define the sequencer transitions that progress to the next state or backwards to the previous state. The ETM implements a sequencer state machine with four states.

Usage constraints

These registers is read/write. Only accept writes when the trace unit is disabled.

Traps and Enables

There are no traps and enables affecting these registers.

Configurations

Available in all configurations.

Attributes

TRCSEQEVRn are 32-bit registers.

The following figure shows the TRCSEQEVRn bit assignments.

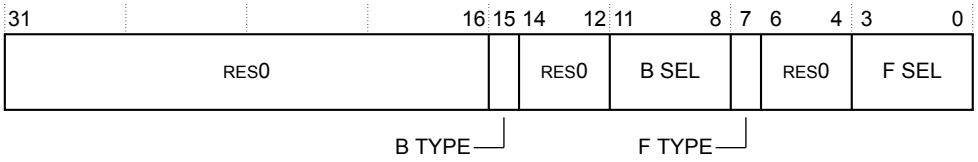

Figure 14-21 TRCSEQEVRn bit assignments

The following table shows the TRCSEQEVRn bit assignments.

Table 14-41 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.
[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.

Table 14-41 TRCSEQEVRn bit assignments (continued)

Bits	Name	Function
[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].

TRCSEQEVRn can be accessed through the memory mapped interface:

Table 14-42 TRCSEQEVRn access information

Component	Offset	Reset
ETM	0x100-0x108	UNK

14.7.19 Sequencer Reset Control Register

The TRCSEQRSTEVR resets the sequencer to state 0.

Usage constraints
This register is read/write. Only accepts writes when the trace unit is disabled.

Traps and Enables
There are no traps and enables affecting this register.

Configurations
Available in all configurations.

Attributes
TRCSEQRSTEVR is a 32-bit register.

The following figure shows the TRCSEQRSTEVR bit assignments.

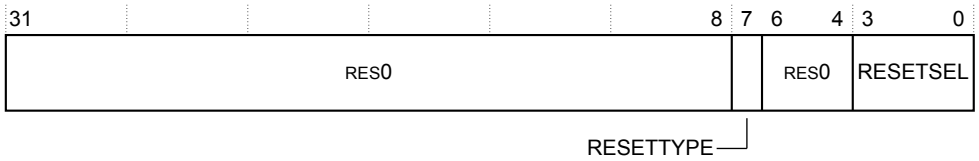

Figure 14-22 TRCSEQRSTEVR bit assignments

The following table shows the TRCSEQRSTEVR bit assignments.

Table 14-43 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 14-43 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, selects a single selected resource from 0-15 defined by bits[3:0]. When RESETTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TRCSEQRSTEVR can be accessed through the memory mapped interface:

Table 14-44 TRCSEQRSTEVR access information

Component	Offset	Reset
ETM	0x118	UNK

14.7.20 Sequencer State Register

The TRCSEQSTR holds the value of the current state of the sequencer.

Usage constraints

This register is read/write. Only accepts writes when the trace unit is disabled.

Must be programmed with an initial value when programming the sequencer.

Traps and Enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCSEQSTR is a 32-bit register.

The following figure shows the TRCSEQSTR bit assignments.

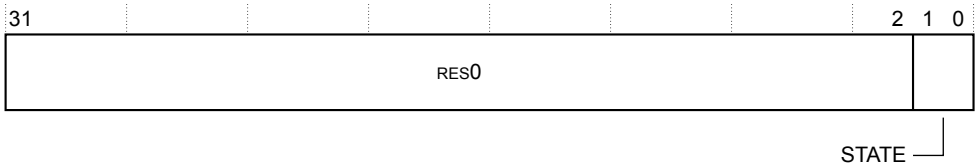

Figure 14-23 TRCSEQSTR bit assignments

The following table shows the TRCSEQSTR bit assignments.

Table 14-45 TRCSEQSTR bit assignments

Bits	Name	Function
[31:2]	-	Reserved, RES0.
[1:0]	STATE	Current sequencer state: 0b00 State 0. 0b01 State 1. 0b10 State 2. 0b11 State 3.

TRCSEQSTR can be accessed through the memory mapped interface:

Table 14-46 TRCSEQSTR access information

Component	Offset	Reset
ETM	0x11C	UNK

14.7.21 External Input Select Register

Use the TRCEXTINSELR register to set, or read, which external inputs are resources to the trace unit.

Usage constraints
This register is read/write. Ignores writes when the trace unit is enabled or not idle.

Traps and Enables
There are no traps and enables affecting this register.

Configurations
Available in all configurations.

Attributes
A 32-bit read/write register. This register is set to an UNKNOWN value on a trace unit reset.

The following figure shows the TRCEXTINSELR bit assignments.

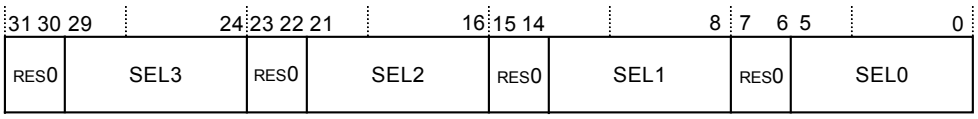

Figure 14-24 TRCEXTINSELR bit assignments

The following table shows the TRCEXTINSELR bit assignments.

Table 14-47 TRCEXTINSELR bit assignments

Bits	Name	Function
[31:30]	-	Reserved, RES0.
[29:24]	SEL3	Selects which external input is a resource for the trace unit.
[23:22]	-	Reserved, RES0.
[21:16]	SEL2	Selects which external input is a resource for the trace unit.
[15:14]	-	Reserved, RES0.

Table 14-47 TRCEXTINSELR bit assignments (continued)

Bits	Name	Function
[13:8]	SEL1	Selects which external input is a resource for the trace unit.
[7:6]	-	Reserved, RES0.
[5:0]	SEL0	Selects which external input is a resource for the trace unit.

TRCEXTINSELR can be accessed through the memory mapped interface:

Table 14-48 TRCEXTINSELR access information

Component	Offset	Reset
ETM	0x120	UNK

14.7.22 Counter Reload Value Registers, n=0-1

Each TRCCNTRLDVRn register defines the reload value for counter <n>.

Usage constraints

This register is read/write. Only accepts writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting these registers.

Configurations

Available in all configurations.

Attributes

TRCCNTRLDVRn are 32-bit registers.

The following figure shows the TRCCNTRLDVRn bit assignments.

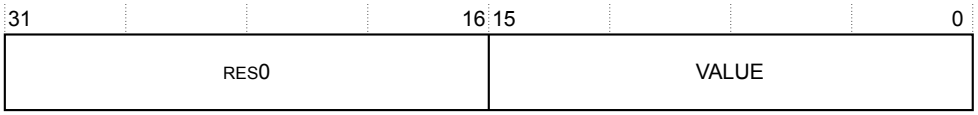

Figure 14-25 TRCCNTRLDVRn bit assignments

The following table shows the TRCCNTRLDVRn bit assignments.

Table 14-49 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.

TRCCNTRLDVRn can be accessed through the memory mapped interface:

Table 14-50 TRCCNTRLDVRn access information

Component	Offset	Reset
ETM	0x140-0x144	UNK

14.7.23 Counter Control Registers 0-1

Each TRCCNTCTLRn register controls the counter <n>.

Usage constraints

These registers are read/write. Only accepts writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting these registers.

Configurations

Available in all configurations.

Attributes

TRCCNTCTLRn are 32-bit registers.

The following figure shows the TRCCNTCTLRn bit assignments.

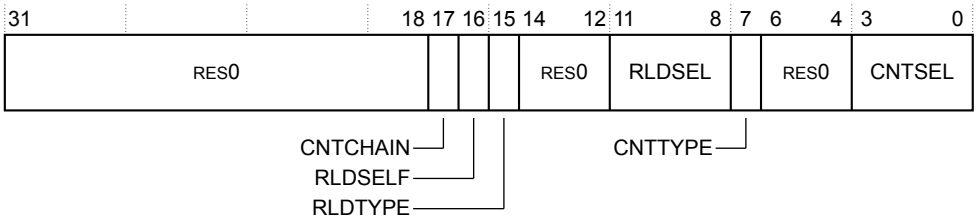

Figure 14-26 TRCCNTCTLRn bit assignments

The following table shows the TRCCNTCTLRn bit assignments.

Table 14-51 TRCCNTCTLRn bit assignments

Bits	Name	Function
[31:18]	-	Reserved, RES0.
[17]	CNTCHAIN[cg]	Defines whether the counter decrements when the counter reloads. This enables two counters to be used in combination to provide a larger counter: `0` The counter operates independently from the counter. The counter only decrements based on CNTTYPE and CNTSEL. `1` The counter decrements when the counter reloads. The counter also decrements when the resource selected by CNTTYPE and CNTSEL is active.
[16]	RLDSELF	Defines whether the counter reloads when it reaches zero: `0` The counter does not reload when it reaches zero. The counter only reloads based on RLDTYPE and RLDSEL. `1` The counter reloads when it reaches zero and the resource selected by CNTTYPE and CNTSEL is also active. The counter also reloads based on RLDTYPE and RLDSEL.
[15]	RLDTYPE	Selects the resource type for the reload: `0` Single selected resource. `1` Boolean combined resource pair.
[14:12]	-	Reserved, RES0.

[cg] Only present on TRCCNTCTLR1. RES0 on TRCCNTCTLR0.

Table 14-51 TRCCNTCTLRn bit assignments (continued)

Bits	Name	Function
[11:8]	RLDSEL	Selects the resource number, based on the value of RLDTYPE: When RLDTYPE is 0, selects a single selected resource from 0-15 defined by RLDSEL[3:0]. When RLDTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by RLDSEL[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, 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].

TRCCNTCTLRn can be accessed through the memory mapped interface:

Table 14-52 TRCCNTCTLRn access information

Component	Offset	Reset
ETM	`0x150-0x154`	UNK

14.7.24 Counter Value Registers, n=0-1

The TRCCNTVRn registers set or return the value of counter <n>.

Usage constraints

These registers is read/write. Only accept writes when the trace unit is disabled. The count value is only stable when TRCSTATR.PMSTABLE==1.

If software uses counter <n>, then it must write to these registers to set the initial counter value.

Traps and enables

There are no traps and enables affecting these registers.

Configurations

Available in all configurations.

Attributes

TRCCNTVRn are 32-bit registers.

The following figure shows the TRCCNTVRn bit assignments.

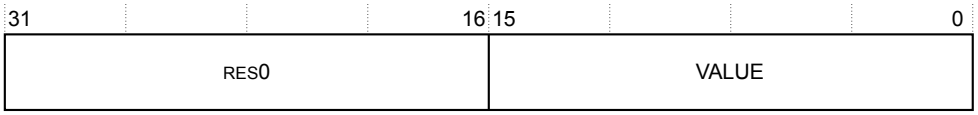

Figure 14-27 TRCCNTVRn bit assignments

The following table shows the TRCCNTVRn bit assignments.

Table 14-53 TRCCNTVRn bit assignments

Bits	Value	Function
[31:16]	-	Reserved, RES0.
[15:0]	VALUE	Contains the current counter value.

TRCCNTVRn can be accessed through the memory mapped interface:

Table 14-54 TRCCNTVRn access information

Component	Offset	Reset
ETM	0x160-0x164	UNK

14.7.25 ID Registers, n=8-13

The TRCIDRn registers provide information about the implemented trace streams that are required to analyze the trace.

Usage constraints
These registers are read-only.

Traps and enables
There are no traps and enables affecting these registers.

Configurations
These registers are available in all configurations.

Attributes
TRCIDRn are 32-bit registers.

The following figure shows the TRCIDR8 bit assignments.

Figure 14-28 TRCIDR8 bit assignments

The following table shows the TRCIDR8 bit assignments.

Table 14-55 TRCIDR8 bit assignments

Bits	Name	Function
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time. 0x00000001 Maximum trace speculation depth is one.

TRCIDR8 can be accessed through the memory mapped interface:

Table 14-56 TRCIDR8 access information

Component	Offset	Reset
ETM	0x180	0x00000001

The following figure shows the TRCIDR9 bit assignments.

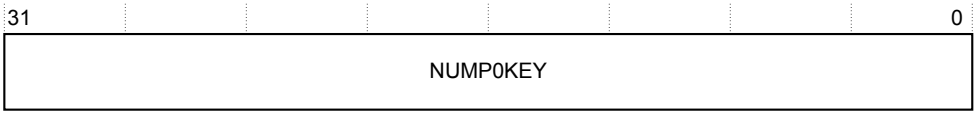

Figure 14-29 TRCIDR9 bit assignments

The following table shows the TRCIDR9 bit assignments.

Table 14-57 TRCIDR9 bit assignments

Bits	Name	Function
[31:0]	NUMP0KEY	Indicates the number of P0 right-hand keys that are used: `0x00000020` The ETM supports 32 P0 keys.

TRCIDR9 can be accessed through the memory mapped interface:

Table 14-58 TRCIDR9 access information

Component	Offset	Reset
ETM	`0x184`	`0x00000020`

The following figure shows the TRCIDR10 bit assignments.

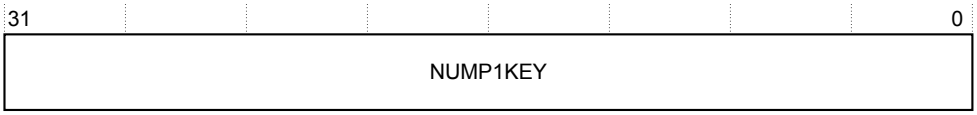

Figure 14-30 TRCIDR10 bit assignments

The following table shows the TRCIDR10 bit assignments.

Table 14-59 TRCIDR10 bit assignments

Bits	Name	Function
[31:0]	NUMP1KEY	Indicates the total number of P1 right-hand keys, including normal and special keys: `0x00000002` The ETM supports 2 P1 right-hand keys.

TRCIDR10 can be accessed through the memory mapped interface:

Table 14-60 TRCIDR10 access information

Component	Offset	Reset
ETM	`0x188`	`0x00000002`

The following figure shows the TRCIDR11 bit assignments.

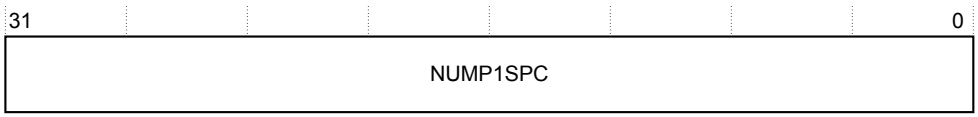

Figure 14-31 TRCIDR11 bit assignments

The following table shows the TRCIDR11 bit assignments.

Table 14-61 TRCIDR11 bit assignments

Bits	Name	Function
[31:0]	NUMP1SPC	Indicates the number of special P1 right-hand keys. `0x00000000` No special P1 right-hand keys used.

TRCIDR11 can be accessed through the memory mapped interface:

Table 14-62 TRCIDR11 access information

Component	Offset	Reset
ETM	`0x18C`	`0x00000000`

The following figure shows the TRCIDR12 bit assignments.

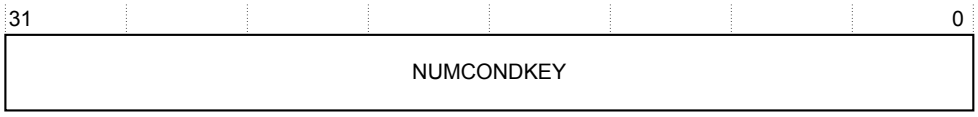

Figure 14-32 TRCIDR12 bit assignments

The following table shows the TRCIDR12 bit assignments.

Table 14-63 TRCIDR12 bit assignments

Bits	Name	Function
[31:0]	NUMCONDKEY	Indicates the total number of conditional instruction right-hand keys, including normal and special keys: `0x00000001` One conditional instruction right-hand key implemented.

TRCIDR12 can be accessed through the memory mapped interface:

Table 14-64 TRCIDR12 access information

Component	Offset	Reset
ETM	`0x190`	`0x00000001`

The following figure shows the TRCIDR13 bit assignments.

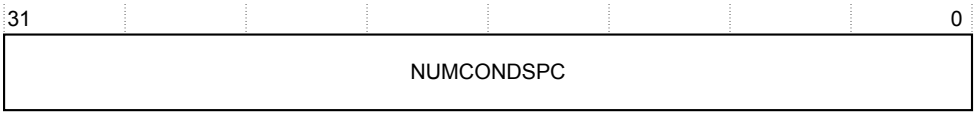

Figure 14-33 TRCIDR13 bit assignments

The following table shows the TRCIDR13 bit assignments.

Table 14-65 TRCIDR13 bit assignments

Bits	Name	Function
[31:0]	NUMCONDSPC	This indicates the number of special conditional instruction right-hand keys. `0x00000000` No special conditional instruction right-hand keys implemented.

TRCIDR13 can be accessed through the memory mapped interface:

Table 14-66 TRCIDR13 access information

Component	Offset	Reset
ETM	0x194	0x00000000

14.7.26 Implementation Specific Register 0

The TRCIMSPEC0 shows the presence of any IMPLEMENTATION-SPECIFIC features, and enables any features that are provided.

Usage constraints

This register is read/write.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCIMSPEC0 is a 32-bit register.

The following figure shows the TRCIMSPEC0 bit assignments.

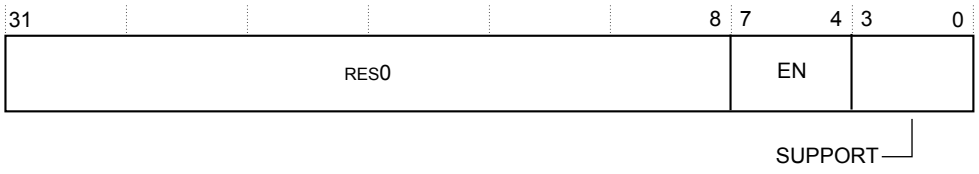

Figure 14-34 TRCIMSPEC0 bit assignments

The following table shows the TRCIMSPEC0 bit assignments.

Table 14-67 TRCIMSPEC0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	EN	SUPPORT set to 0, this field is RES0
[3:0]	SUPPORT	Set to 0. No IMPLEMENTATION-SPECIFIC extensions are supported.

TRCIMSPEC0 can be accessed through the memory mapped interface:

Table 14-68 TRCIMSPEC0 access information

Component	Offset	Reset
ETM	0x1C0	0x00000000

14.7.27 ID Register 0

The TRCIDR0 indicates the tracing capabilities of the ETM.

Usage constraints

This register is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCIDR0 is a 32-bit register.

The following figure shows the TRCIDR0 bit assignments.

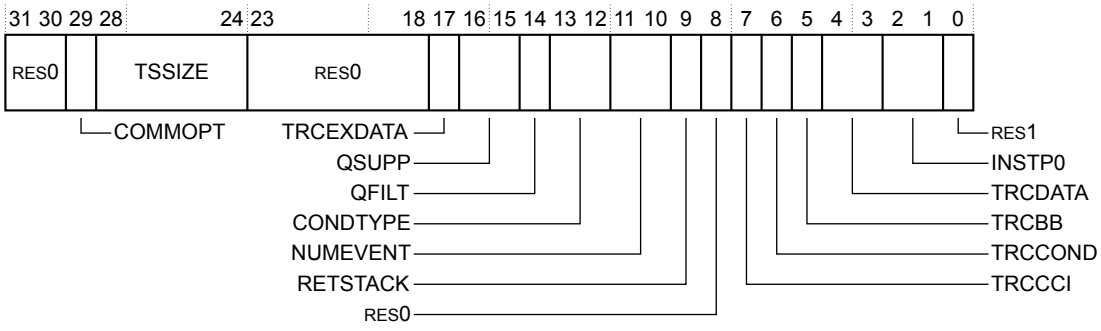

Figure 14-35 TRCIDR0 bit assignments

The following table shows the TRCIDR0 bit assignments.

Table 14-69 TRCIDR0 bit assignments

Bits	Name	Function
[31:30]	-	Reserved, RES0.
[29]	COMMOPT	Indicates the meaning of the commit field in some packets: `0` Commit mode 0.
[28:24]	TSSIZE	Global timestamp size: `0b01000` Maximum of 64-bit global timestamp implemented.
[23:18]	-	Reserved, RES0.
[17]	TRCEXDATA	Indicates support for the tracing of data transfers for exceptions and exception returns: `0` TRCVDCTLR.TRCEXDATA is not implemented.
[16:15]	QSUPP	Indicates Q element support: `0b00` Q elements not supported.
[14]	QFILT	This field is RES0.
[13:12]	CONDTYPE	Indicates how conditional results are traced: `0b00` The 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: `0b11` Four events supported.

Table 14-69 TRCIDR0 bit assignments (continued)

Bits	Name	Function
[9]	RETSTACK	Return stack support: `1` Return stack implemented.
[8]	-	Reserved, RES0.
[7]	TRCCCI	Support for cycle counting in the instruction trace: `1` Cycle counting in the instruction trace is implemented.
[6]	TRCCOND	Support for conditional instruction tracing: `1` Conditional instruction tracing is implemented.
[5]	TRCBB	Support for branch broadcast tracing: `1` Branch broadcast tracing is implemented.
[4:3]	TRCDATA	Support for tracing of data: `0b11` Tracing of data addresses and data values is supported.
[2:1]	INSTP0	Support for tracing of load and store instructions as P0 elements: `0b11` Tracing of load and store instructions as P0 elements is supported.
[0]	-	Reserved, RES1.

TRCIDR0 can be accessed through the memory mapped interface:

Table 14-70 TRCIDR0 access information

Component	Offset	Reset
ETM	`0x1E0`	`0x0C000EFF`

14.7.28 ID Register 1

The TRCIDR1 indicates the basic architecture of the ETM.

Usage constraints
This register is read-only.

Traps and enables
There are no traps and enables affecting this register.

Configurations
Available in all configurations.

Attributes
TRCIDR1 is a 32-bit register.

The following figure shows the TRCIDR1 bit assignments.

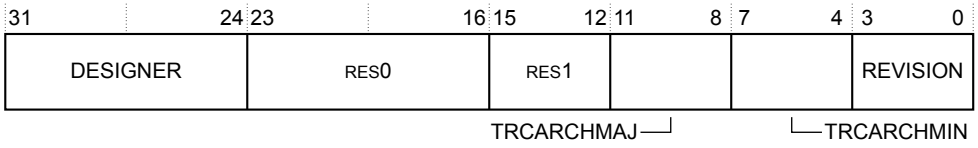

Figure 14-36 TRCIDR1 bit assignments

The following table shows the TRCIDR1 bit assignments.

Table 14-71 TRCIDR1 bit assignments

Bits	Name	Function
[31:24]	DESIGNER	Indicates the designer of the trace unit: `0x41` Arm.
[23:16]	-	Reserved, RES0.
[15:12]	-	Reserved, RES1.
[11:8]	TRCARCHMAJ	Major trace unit architecture version number: `0b0100` ETMv4.
[7:4]	TRCARCHMIN	Minor trace unit architecture version number: `0b0010` Minor revision 2.
[3:0]	REVISION	Implementation revision number: `0x4` Implementation revision 4.

TRCIDR1 can be accessed through the memory mapped interface:

Table 14-72 TRCIDR1 access information

Component	Offset	Reset
ETM	`0x1E4`	`0x4100F424`

14.7.29 ID Register 2

The TRCIDR2 indicates the maximum sizes of certain aspects of items in the trace.

Usage constraints
This register is read-only.

Traps and enables
There are no traps and enables affecting this register.

Configurations
Available in all configurations.

Attributes
TRCIDR2 is a 32-bit register.

The following figure shows the TRCIDR2 bit assignments.

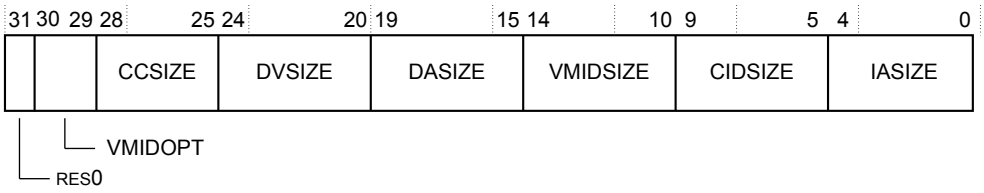

Figure 14-37 TRCIDR2 bit assignments

The following table shows the TRCIDR2 bit assignments.

Table 14-73 TRCIDR2 bit assignments

Bits	Name	Function
[31]	-	Reserved, RES0.
[30:29]	VMIDOPT	Indicates the options for observing the Virtual context identifier in the processor: `0b00` TRCCONFIGR.VMIDOPT is not implemented and this field is RES0.
[28:25]	CCSIZE	Indicates the size of the cycle counter in bits minus 12: `0b0000` Cycle count is 12 bits.
[24:20]	DVSIZE	Data value size in bytes: `0b00100` Maximum of 32-bit data value size is supported.
[19:15]	DASIZE	Data address size in bytes: `0b00100` Maximum of 32-bit address size is supported.
[14:10]	VMIDSIZE	Virtual Machine ID size: `0b00001` Virtual Machine ID size is 1 byte.
[9:5]	CIDSIZE	Context ID tracing: `0b00100` Context ID size is 4 bytes.
[4:0]	IASIZE	Instruction address size: `0b00100` Maximum of 32-bit address size.

TRCIDR2 can be accessed through the memory mapped interface:

Table 14-74 TRCIDR2 access information

Component	Offset	Reset
ETM	`0x1E8`	`0x00420484`

14.7.30 ID Register 3

The TRCIDR3 indicates certain aspects of the ETM configuration.

Usage constraints

This register is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCIDR3 is a 32-bit register.

The following figure shows the TRCIDR3 bit assignments.

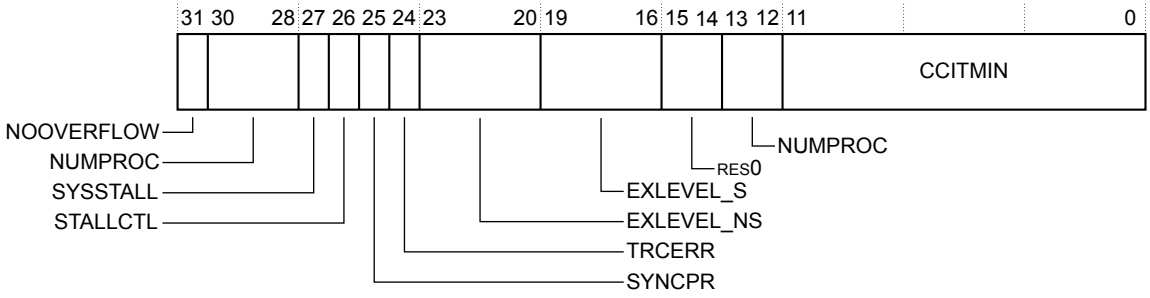

Figure 14-38 TRCIDR3 bit assignments

The following table shows the TRCIDR3 bit assignments.

Table 14-75 TRCIDR3 bit assignments

Bits	Name	Function
[31]	NOOVERFLOW	Indicates whether TRCSTALLCTLR.NOOVERFLOW is implemented: `0` NOOVERFLOW is not implemented.
[30:28]	NUMPROC	Indicates the number of processors available for tracing: `0b000` The trace unit can trace one processor.
[27]	SYSSTALL	System support for stall control of the processor: `1` System supports stall control of the processor.
[26]	STALLCTL	Stall control support: `1` TRCSTALLCTLR is implemented.
[25]	SYNCPR	Indicates trace synchronization period support: `0` TRCSYNCPR is read/write. Software can change the synchronization period.
[24]	TRCERR	Indicates whether TRCVICTLR.TRCERR is implemented: `1` TRCERR is implemented.
[23:20]	EXLEVEL_NS	Exception levels are implemented: `0b0111` EL0, EL1, and EL2 are implemented.
[19:16]	EXLEVEL_S	Related to whether instruction tracing is supported for corresponding Exception level. This field is `0`.
[15:14]	-	Reserved, RES0.

Table 14-75 TRCIDR3 bit assignments (continued)

Bits	Name	Function
[13:12]	NUMPROC	Indicates the number of PEs available for tracing. This field is `0b00000`.
[11:0]	CCITMIN	Instruction trace cycle counting minimum threshold: `0x4` Minimum threshold is 4 instruction trace cycle.

TRCIDR3 can be accessed through the memory mapped interface:

Table 14-76 TRCIDR3 access information

Component	Offset	Reset
ETM	`0x1EC`	`0x0D700004`

14.7.31 ID Register 4

The TRCIDR4 indicates the resources available in the ETM.

Usage constraints
This register is read-only.

Traps and enables
There are no traps and enables affecting this register.

Configurations
Available in all configurations.

Attributes
TRCIDR4 is a 32-bit register.

The following figure shows the TRCIDR4 bit assignments.

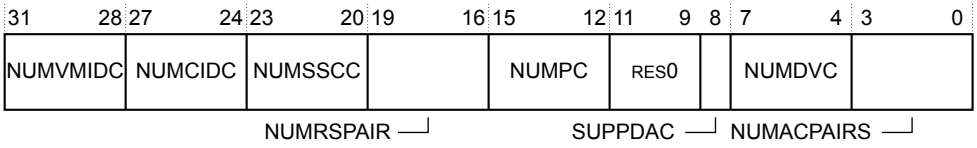

Figure 14-39 TRCIDR4 bit assignments

The following table shows the TRCIDR4 bit assignments.

Table 14-77 TRCIDR4 bit assignments

Bits	Name	Function
[31:28]	NUMVMIDC	Number of VMID comparators implemented: `0b0001` One VMID comparator is implemented.
[27:24]	NUMCIDC	Number of Context ID comparators implemented: `0b0001` One Context ID comparator is implemented.
[23:20]	NUMSSCC	Number of single-shot comparator controls implemented: `0b0010` Two single-shot comparator controls are implemented.

Table 14-77 TRCIDR4 bit assignments (continued)

Bits	Name	Function
[19:16]	NUMRSPAIR	Number of resource selection pairs implemented: `0b0111` Eight resource selection pairs are implemented.
[15:12]	NUMPC	Number of processor comparator inputs implemented: `0b0000` No processor comparator inputs.
[11:9]	-	Reserved, RES0.
[8]	SUPPDAC	Data address comparisons implemented: `1` Data address comparisons are supported.
[7:4]	NUMDVC	Number of data value comparators implemented: `0b0010` Two data value comparators are implemented.
[3:0]	NUMACPAIRS	Number of address comparator pairs implemented: `0b0100` Four address comparator pairs are implemented.

TRCIDR4 can be accessed through the memory mapped interface:

Table 14-78 TRCIDR4 access information

Component	Offset	Reset
ETM	`0x1F0`	`0x11270124`

14.7.32 ID Register 5

The TRCIDR5 indicates the resources available in the ETM.

Usage constraints
This register is read-only.

Traps and enables
There are no traps and enables affecting this register.

Configurations
Available in all configurations.

Attributes
TRCIDR5 is a 32-bit register.

The following figure shows the TRCIDR5 bit assignments.

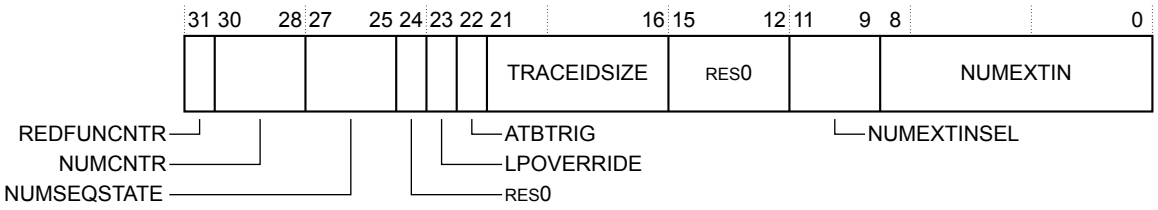

Figure 14-40 TRCIDR5 bit assignments

The following table shows the TRCIDR5 bit assignments.

Table 14-79 TRCIDR5 bit assignments

Bits	Name	Function
[31]	REDFUNCNTR	Reduced Function Counter implemented: `0` Reduced Function Counter not implemented.
[30:28]	NUMCNTR	Number of counters implemented: `0b010` Two counters implemented.
[27:25]	NUMSEQSTATE	Number of sequencer states implemented: `0b100` Four sequencer states implemented.
[24]	-	Reserved, RES0.
[23]	LPOVERRIDE	Low power state override support: `1` Low power state override support implemented.
[22]	ATBTRIG	ATB trigger support: `1` ATB trigger support implemented.
[21:16]	TRACEIDSIZE	Number of bits of trace ID: `0x07` Seven-bit trace ID implemented.
[15:12]	-	Reserved, RES0.
[11:9]	NUMEXTINSEL	Number of external input selectors implemented: `0b100` Four external input selectors are implemented.
[8:0]	NUMEXTIN	Number of external inputs implemented: `0x35` 35 external inputs implemented in instruction and data trace configuration.

TRCIDR5 can be accessed through the memory mapped interface:

Table 14-80 TRCIDR5 access information

Component	Offset	Reset
ETM	`0x1F4`	`0x28C70835`

14.7.33 Resource Selection Registers, n=2-15

The TRCRSCTLRn registers control the trace resources.

Usage constraints
These registers are read/write.

Traps and enables
There are no traps and enables affecting these registers.

Configurations
Available in all configurations.

Attributes
The TRCRSCTLRn are 32-bit registers.

The following figure shows the TRCRSCTLRn bit assignments.

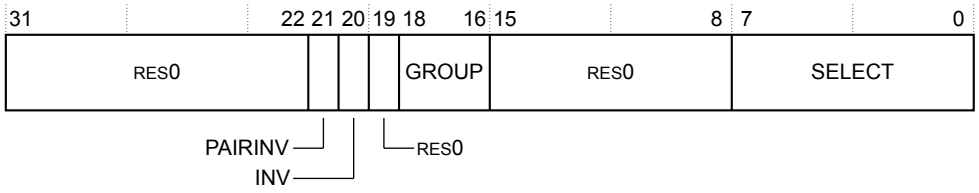

Figure 14-41 TRCRSCTLRn bit assignments

The following table shows the TRCRSCTLRn bit assignments.

Table 14-81 TRCRSCTLRn bit assignments

Bits	Name	Function
[31:22]	-	Reserved, RES0.
[21]	PAIRINV	Inverts the result of a combined pair of resources. This bit is only implemented on the lower register for a pair of resource selectors. This bit is RES0 for the upper register.
[20]	INV	Inverts the selected resources: `0` Resource is not inverted. `1` Resource is inverted.
[19]	-	Reserved, RES0.
[18:16]	GROUP	Selects a group of resources.
[15:8]	-	Reserved, RES0.
[7:0]	SELECT	Selects one or more resources from the wanted group. One bit is provided per resource from the group.

TRCRSCTLRn can be accessed through the memory mapped interface:

Table 14-82 TRCRSCTLRn access information

Component	Offset	Reset
ETM	`0x208-0x23C`	UNK

14.7.34 Single-shot Comparator Control Registers, n=0-1

The TRCSSCCRn registers control the single-shot comparator <n>.

Usage constraints
These registers are read/write.

Traps and enables
There are no traps and enables affecting these registers.

Configurations
Available in all configurations.

Attributes
The TRCSSCCRn registers are 32-bit.

The following figure shows the TRCSSCCRn bit assignments.

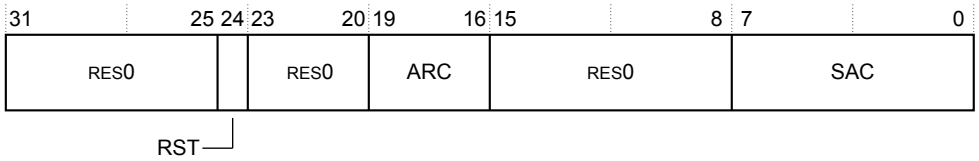

Figure 14-42 TRCSSCCRn bit assignments

The following table shows the TRCSSCCRn bit assignments.

Table 14-83 TRCSSCCRn bit assignments

Bits	Name	Function
[31:25]	-	Reserved, RES0.
[24]	RST	Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected: 1 Reset enabled. Multiple matches can occur.
[23:20]	-	Reserved, RES0.
[19:16]	ARC	Selects one or more address range comparators for single-shot control. One bit is provided for each 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 single address comparator.

TRCSSCCRn can be accessed through the memory mapped interface:

Table 14-84 TRCSSCCRn access information

Component	Offset	Reset
ETM	0x280-0x284	UNK

14.7.35 Single-shot Comparator Status Registers n=0-1

The TRCSSCSRn registers indicate the status of the single-shot comparator <n>. It is sensitive to instruction addresses.

Usage constraints

These registers are read/write.

Traps and enables

There are no traps and enables affecting these registers.

Configurations

Available in all configurations.

Attributes

The TRCSSCSRn registers are 32-bit.

The following figure shows the TRCSSCSRn bit assignments.

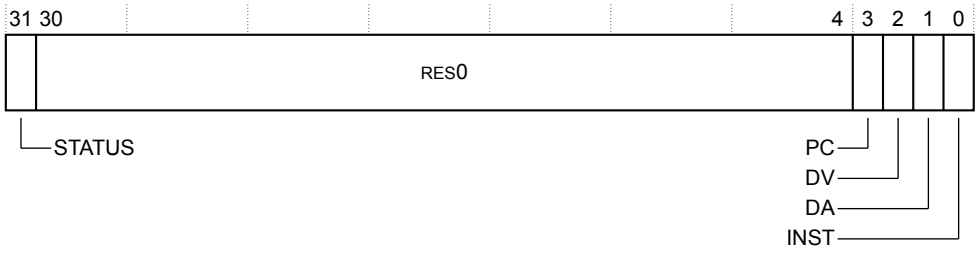

Figure 14-43 TRCSSCSRn bit assignments

The following table shows the TRCSSCSR0 bit assignments.

Table 14-85 TRCSSCSRn bit assignments

Bits	Name	Function
[31]	STATUS	Single-shot status. This indicates whether any of the selected comparators have matched: `0` Match has not occurred. `1` Match has occurred at least once. When programming the ETM, if TRCSSCCRn.RST is `0`, the STATUS bit must be explicitly written to `0` to enable this single-shot comparator control.
[30:4]	-	Reserved, RES0.
[3]	PC	Indicates that the Single-shot comparator is sensitive to processor comparator inputs. `0` Single-shot comparator inputs are not supported.
[2]	DV	Data value comparator support. For TRCSSCSR0, the value is: `0` Single-shot data value comparisons not supported. For TRCSSCSR1, the value is: `1` Single-shot data value comparisons supported.
[1]	DA	Data address comparator support. For TRCSSCSR0, the value is: `0` Single-shot data address comparisons not supported. For TRCSSCSR1, the value is: `1` Single-shot data address comparisons supported.
[0]	INST	Instruction address comparator support: `1` Single-shot instruction address comparisons supported.

TRCSSCSRn can be accessed through the memory mapped interface:

Table 14-86 TRCSSCSRn access information

Component	Offset	Reset
ETM	`0x2A0-0x2A4`	UNK

14.7.36 OS Lock Access Register

The TRCOSLAR controls whether the OS Lock is locked.

Usage constraints
This register is write-only.

Traps and enables
There are no traps and enables affecting this register.

Configurations
Always implemented because OS Lock is always implemented.

Attributes
TRCOSLAR is a 32-bit register.

The following figure shows the TRCOSLAR bit assignments.

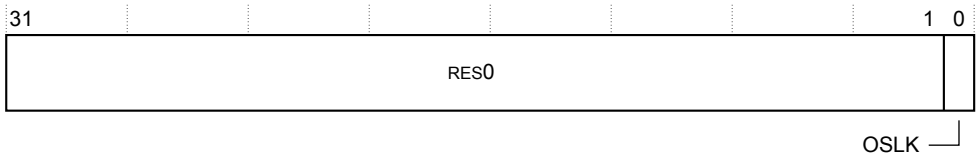

Figure 14-44 TRCOSLAR bit assignments

The following table shows the TRCOSLAR bit assignments.

Table 14-87 TRCOSLAR bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	OSLK	OS Lock control bit: `0` Unlocks the OS Lock. `1` Locks the OS Lock. This setting disables the trace unit.

TRCOSLAR can be accessed through the memory mapped interface:

Table 14-88 TRCOSLAR access information

Component	Offset	Reset
ETM	`0x300`	UNK

14.7.37 OS Lock Status Register

The TRCOSLSR returns the status of the OS Lock.

Usage constraints
This register is read-only.

Traps and enables
There are no traps and enables affecting this register.

Configurations
Available in all implementations.

Attributes
TRCOSLSR is a 32-bit register.

The following figure shows the TRCOSLSR bit assignments.

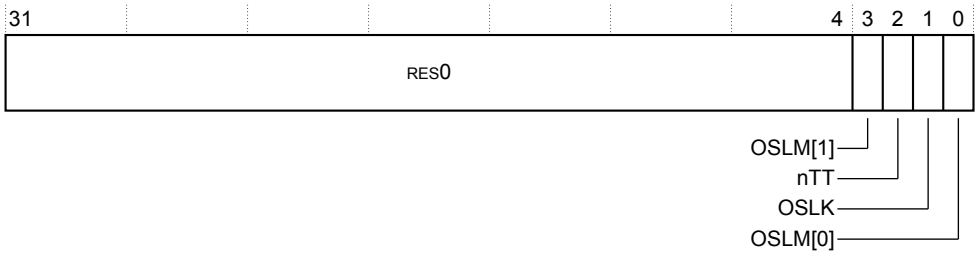

Figure 14-45 TRCOSLSR bit assignments

The following table shows the TRCOSLSR bit assignments.

Table 14-89 TRCOSLSR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3, 0]	OSLM[1]	Indicates whether the OS Lock model is implemented or not. `0b10` OS Lock is implemented.
[2]	nTT	This bit is RAZ, which indicates that software must perform a 32-bit write to update the TRCOSLAR.
[1]	OSLK	OS Lock status bit: `0` The OS Lock is unlocked. `1` The OS Lock is locked. The reset value is 1.
[0]	OSLM[0]	Indicates whether the OS Lock model is implemented or not. **`0b10`** OS Lock is implemented.

TRCOSLSR can be accessed through the memory mapped interface:

Table 14-90 TRCOSLSR access information

Component	Offset	Reset
ETM	`0x304`	`0x0000000A`

14.7.38 Power Down Control Register

TRCPDCR requests to the system power controller to keep the ETM powered up.

Usage constraints

This register is read/write. Accessible only from the memory-mapped interface or from an external agent such as a debugger.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCPDCR is a 32-bit register.

The following figure shows the TRCPDCR bit assignments.

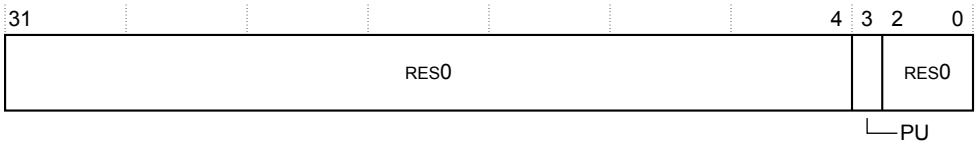

Figure 14-46 TRCPDCR bit assignments

The following table shows the TRCPDCR bit assignments.

Table 14-91 TRCPDCR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RES0.
[3]	PU	Powerup request, to request that power to the ETM and access to the trace registers is maintained: `0` Power not requested. `1` Power requested. This bit is reset to 0 on a trace unit reset.
[2:0]	-	Reserved, RES0.

TRCPDCR can be accessed through the memory mapped interface:

Table 14-92 TRCPDCR access information

Component	Offset	Reset
ETM	`0x310`	`0x00000000`

14.7.39 Power Down Status Register

The TRCPDSR indicates the power down status of the ETM.

Usage constraints

This register is read-only. Accessible only from the memory-mapped interface or from an external agent such as a debugger.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCPDSR is a 32-bit register.

The following figure shows the TRCPDSR bit assignments.

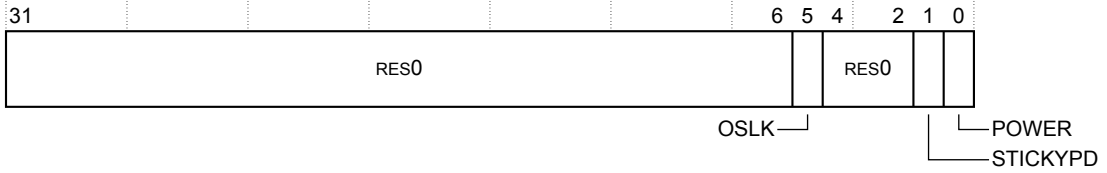

Figure 14-47 TRCPDSR bit assignments

The following table shows the TRCPDSR bit assignments.

Table 14-93 TRCPDSR bit assignments

Bits	Name	Function
[31:6]	-	Reserved, RES0.
[5]	OSLK	OS Lock status bit: 0 The OS Lock is unlocked. 1 The OS Lock is locked. This field is reset to 1 on a trace unit reset.
[4:2]	-	Reserved, RES0.
[1]	STICKYPD	Sticky power down state. 0 Trace register power has not been removed since the TRCPDSR was last read. 1 Trace register power has been removed since the TRCPDSR was last read. This bit is set to 1 when power to the ETM registers is removed, to indicate that programming state has been lost. It is cleared after a read of the TRCPDSR.
[0]	POWER	Indicates the ETM is powered up: 1 ETM is powered up. All registers are accessible. 0 The trace unit core power domain is not powered. The trace registers are not accessible and they all return an error response.

TRCPDSR can be accessed through the memory mapped interface:

Table 14-94 TRCPDSR access information

Component	Offset	Reset
ETM	0x314	UNK

14.7.40 Address Comparator Value Registers, n=0-7

Each TRCACVRn register indicates the address for the data address comparator <n>.

Usage constraints

These registers are read-write. Only accept writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCACVRn are 32-bit registers.

The following figure shows the TRCACVRn bit assignments.

Figure 14-48 TRCACVRn bit assignments

The following table shows the TRCACVRn bit assignments.

Table 14-95 TRCACVRn bit assignments

Bits	Name	Function
[31:0]	ADDRESS	The address value to compare against.

TRCACVRn can be accessed through the memory mapped interface:

Table 14-96 TRCACVRn access information

Component	Offset	Reset
ETM	0x400-0x438	UNK

14.7.41 Address Comparator Access Type Registers, n=0-7

Each TRCACATRn register controls the access for the data address comparator <n>.

Usage constraints

These registers are read/write. Only accept writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all build configurations.

Attributes

TRCACATRn are 32-bit registers.

The following figure shows the TRCACATR0 bit assignments.

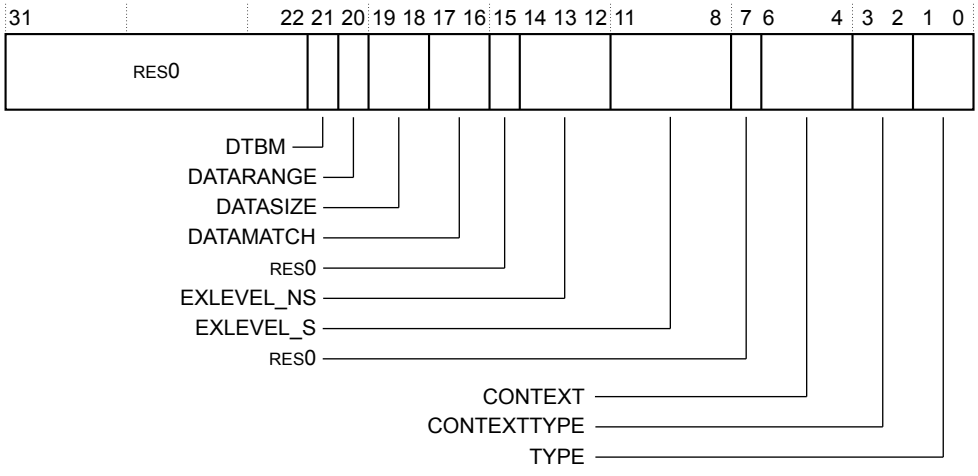

Figure 14-49 TRCACATR0 bit assignments

The following table shows the TRCACATR0 bit assignments.

Table 14-97 TRCACATR0 bit assignments

Bits	Name	Function
[31:22]	-	Reserved, RES0.
[21]	DTBM	This field is RES0.
[20]	DATARANGE	Data value comparison range control, to select whether the data value comparison is made against the single address comparator or the address range comparator: `0` Only the single address comparator matches. `1` Only the address range comparator matches.
[19:18]	DATASIZE	Data value comparison size control: `0b00` Byte size. `0b01` Halfword size. `0b10` Word size. `0b11` Doubleword size.
[17:16]	DATAMATCH	Data value comparison control: `0b01` Comparator matches only if the data value comparison matches. `0b11` Comparator matches only if the data value comparison does not match.
[15]	-	Reserved, RES0.
[14]	EXLEVEL_NS	Indicates if the comparator matches in EL2: `0` The comparator can match in this exception level. `1` The comparator must not match in this exception level.
[13]	EXLEVEL_NS	Indicates if the comparator matches EL1.
[12]	EXLEVEL_NS	Indicates if the comparator matches EL0.
[11:8]	EXLEVEL_S	This field is RES0.
[7]	-	Reserved, RES0.
[6:4]	CONTEXT	Context ID comparator or Virtual context identifier comparator: `0b00` Comparator 0.

Table 14-97 TRCACATR0 bit assignments (continued)

Bits	Name	Function
[3:2]	CONTEXTTYPE	The type of comparison: `0b00` The trace unit does not perform a Context ID comparison. `0b01` The trace unit performs a Context ID comparison, and signals a match if both the Context ID comparators match and the address comparator match. `0b10` The trace unit performs a Virtual context identifier comparison, and signals a match if both the Virtual context identifier comparator and the address comparator match. `0b11` The trace unit performs a Context ID comparison and a Virtual context identifier comparison, and signals a match if the Context ID comparator, the Virtual context identifier comparator, and the address comparator match.
[1:0]	TYPE	The type of comparison: `0b00` Instruction address. `0b01` Data load address. `0b10` Data store address. `0b11` Data load or store address.

Note

TRCACATR2, 4 and 6 are functionally identical to TRCACATR0.

The following figure shows the TRCACATR1 bit assignments.

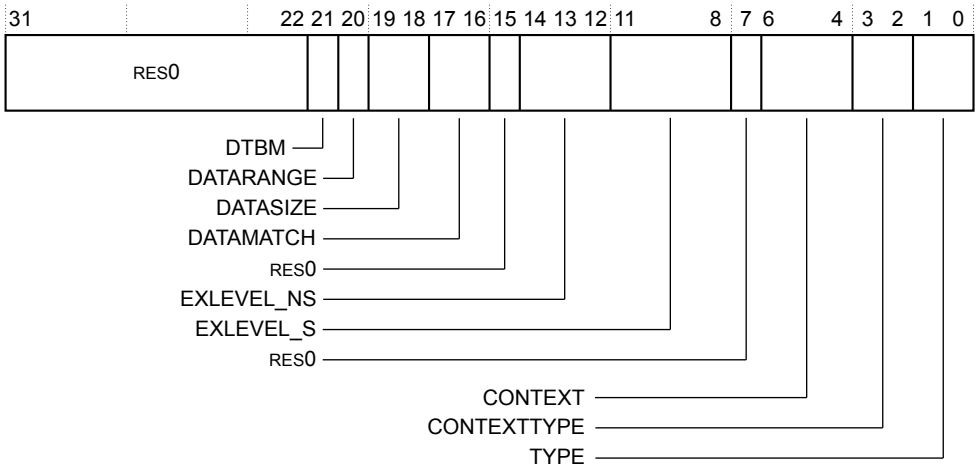

Figure 14-50 TRCACATR1 bit assignments

The following figure shows the TRCACATR1 bit assignments.

Table 14-98 TRCACATR1 bit assignments

Bits	Name	Function
[31:22]	-	Reserved, RES0.
[21]	DTBM	This field is RES0.

Table 14-98 TRCACATR1 bit assignments (continued)

Bits	Name	Function
[20]	DATARANGE	This field is RES0.
[19:18]	DATASIZE	This field is RES0.
[17:16]	DATAMATCH	This field is RES0.
[15]	-	Reserved, RES0.
[14]	EXLEVEL_NS	Indicates if the comparator matches in EL2: `0` The comparator can match in this exception level. `1` The comparator must not match in this exception level.
[13]	EXLEVEL_NS	Indicates if the comparator matches EL1.
[12]	EXLEVEL_NS	Indicates if the comparator matches EL0.
[11:8]	EXLEVEL_S	This field is RES0.
[7]	-	Reserved, RES0.
[6:4]	CONTEXT	Context ID comparator or Virtual context identifier comparator: `0b00` Comparator 0.
[3:2]	CONTEXTTYPE	The type of comparison: `0b00` The trace unit does not perform a Context ID comparison. `0b01` The trace unit performs a Context ID comparison, and signals a match if both the Context ID comparators match and the address comparator match. `0b10` The trace unit performs a Virtual context identifier comparison, and signals a match if both the Virtual context identifier comparator and the address comparator match. `0b11` The trace unit performs a Context ID comparison and a Virtual context identifier comparison, and signals a match if the Context ID comparator, the Virtual context identifier comparator, and the address comparator match.
[1:0]	TYPE	The type of comparison: `0b00` Instruction address. `0b01` Data load address. `0b10` Data store address. `0b11` Data load or store address.

Note

TRCACATR3, 5 and 7 are functionally identical to TRCACATR1.

TRCACATRn can be accessed through the memory mapped interface:

Table 14-99 TRCACATRn access information

Component	Offset	Reset
ETM	`0x480-0x4B8`	UNK

14.7.42 Data Value Comparator Value Registers, n=0-1

Each TRCDVCVRn register indicates the value for the data value comparator <n>.

Usage constraints

This register is read/write. Only accepts writes when the trace unit is disabled.

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

TRCDVCVRn registers are 32 bits.

The following figure shows the TRCDVCVRn bit assignments.

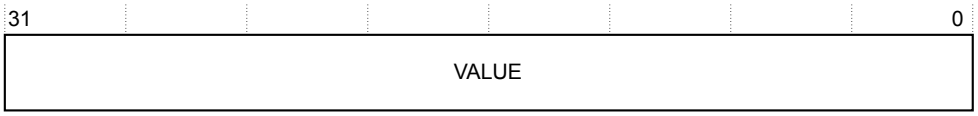

Figure 14-51 TRCDVCVRn bit assignments

The following table shows the TRCDVCVRn bit assignments.

Table 14-100 TRCDVCVRn bit assignments

Bits	Name	Function
[31:0]	VALUE	The data value to compare against.

TRCDVCVRn can be accessed through the memory mapped interface:

Table 14-101 TRCDVCVRn access information

Component	Offset	Reset
ETM	0x500-0x510	UNK

14.7.43 Data Value Comparator Mask Registers, n=0-1

Each TRCDVCMRn register controls the mask value for the data value comparator <n>.

Usage constraints

These registers are read/write.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all build configurations.

Attributes

TRCDVCMRn are 32-bit registers.

The following figure shows the TRCDVCMRn bit assignments.

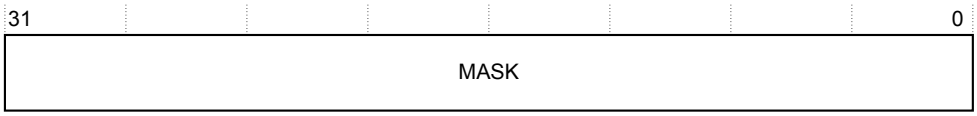

Figure 14-52 TRCDVCMRn bit assignments

The following table shows the TRCDVCMRn bit assignments.

Table 14-102 TRCDVCMRn bit assignments

Bits	Name	Function
[31:0]	MASK	The mask value to apply to the data value comparison.

TRCDVCMRn can be accessed through the memory mapped interface:

Table 14-103 TRCDVCMRn access information

Component	Offset	Reset
ETM	0x580-0x590	UNK

14.7.44 Context ID Comparator Value Registers 0

The TRCCIDCVR0 register contains a Context ID value.

Usage constraints
This register is read/write and ignores writes when the trace unit is enabled or not idle.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
TRCCIDCVR0 is a 32-bit register.

The following figure shows the TRCCIDCVR0 bit assignments.

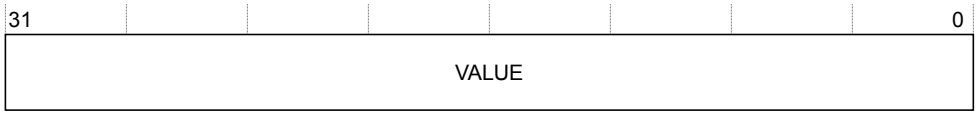

Figure 14-53 TRCCIDCVR0 bit assignments

The following table shows the TRCCIDCVR0 bit assignments.

Table 14-104 TRCCIDCVR0 bit assignments

Bits	Name	Function
[31:0]	VALUE	Context ID Value.

TRCCIDCVR0 can be accessed through the memory mapped interface:

Table 14-105 TRCCIDCVR0 access information

Component	Offset	Reset
ETM	0x600	UNK

14.7.45 Virtual Context Identifier Comparator Value Register

The TRCVMIDCVR0 register contains a virtual context identifier value.

Usage constraints
This register is read/write and ignores writes when the trace unit is enabled or not idle.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
TRCVMIDCVR0 is a 32-bit register.

The following figure shows the TRCVMIDCVR0 bit assignments.

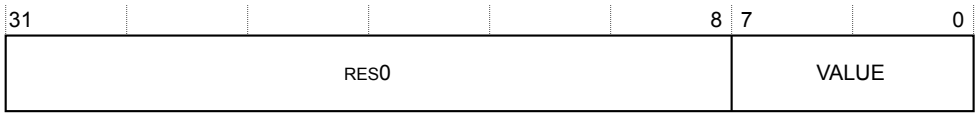

Figure 14-54 TRCVMIDCVR0 bit assignments

The following table shows the TRCVMIDCVR0 bit assignments.

Table 14-106 TRCVMIDCVR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	VALUE	VMID value.

TRCVMIDCVR0 can be accessed through the memory mapped interface:

Table 14-107 TRCVMIDCVR0 access information

Component	Offset	Reset
ETM	0x640	UNK

14.7.46 Context ID Comparator Control Register 0

The TRCCIDCCTLR0 contains Context ID mask values for the TRCCIDCVRn registers, where n=0-3.

Usage constraints
This register is read/write and ignores writes when the trace unit is enabled or not idle.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
TRCCIDCCTLR0 is a 32-bit register.

The following figure shows the TRCCIDCCTLR0 bit assignments.

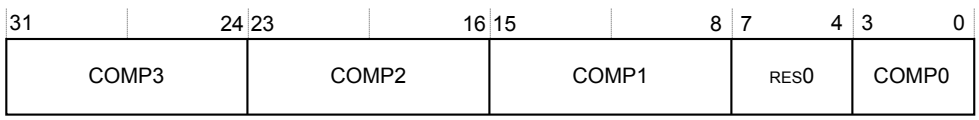

Figure 14-55 TRCCIDCCTLR0 bit assignments

The following table shows the TRCCIDCCTLR0 bit assignments.

Table 14-108 TRCCIDCCTLR0 bit assignments

Bits	Name	Function
[31:24]	COMP3	Controls the mask value that the trace unit applies to TRCCIDCVRn. This field is RES0.
[23:16]	COMP2	Controls the mask value that the trace unit applies to TRCCIDCVRn. This field is RES0.
[15:8]	COMP1	Controls the mask value that the trace unit applies to TRCCIDCVRn. This field is RES0.
[7:4]	-	Reserved, RES0.
[3:0]	COMP0	Controls the mask value that the trace unit applies to TRCCIDCVR0: `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.

TRCCIDCCTLR0 can be accessed through the memory mapped interface:

Table 14-109 TRCCIDCCTLR0 access information

Component	Offset	Reset
ETM	`0x680`	UNK

14.7.47 Integration Mode Control Register

The TRCITCTRL register enables topology detection or integration testing, by putting the ETM into integration mode.

Usage constraints

This register is read/write. 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.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCITCTRL is a 32-bit register.

The following figure shows the TRCITCTRL bit assignments.

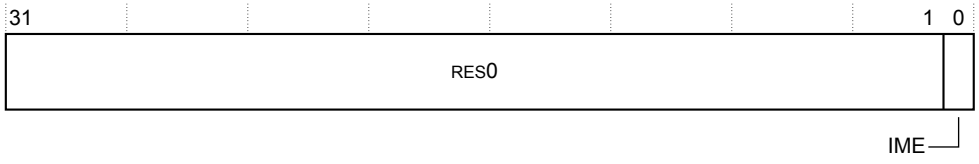

Figure 14-56 TRCITCTRL bit assignments

The following table shows the TRCITCTRL bit assignments.

Table 14-110 TRCITCTRL bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	IME	Integration mode enable: 0 ETM is not in integration mode.

TRCITCTRL can be accessed through the memory mapped interface:

Table 14-111 TRCITCTRL access information

Component	Offset	Reset
ETM	0xF00	0x00000000

Note

The Cortex-R52 processor does not include ATB integration registers.

14.7.48 Claim Tag Set Register

The TRCCLAIMSET register sets bits in the claim tag and determines the number of claim tag bits implemented.

Usage constraints
This register is read/write.

Traps and enables
There are no traps and enables affecting this register.

Configurations
Available in all configurations.

Attributes
TRCCLAIMSET is a 32-bit register.

The following figure shows the TRCCLAIMSET bit assignments.

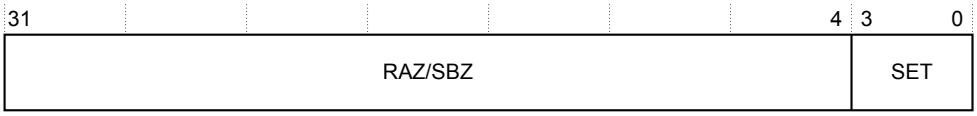

Figure 14-57 TRCCLAIMSET bit assignments

The following table shows the TRCCLAIMSET bit assignments.

Table 14-112 TRCCLAIMSET bit assignments

Bits	Name	Function
[31:4]	-	RAZ/SBZ.
[3:0]	SET	On reads, for each bit: 0 Claim tag bit is not implemented. 1 Claim tag bit is implemented. On writes, for each bit: 0 Has no effect. 1 Sets the relevant bit of the claim tag.

TRCCLAIMSET can be accessed through the memory mapped interface:

Table 14-113 TRCCLAIMSET access information

Component	Offset	Reset
ETM	0xFA0	0x0000000F

14.7.49 Claim Tag Clear Register

The TRCCLAIMCLR clears bits in the claim tag and determines the current value of the claim tag.

Usage constraints
This register is read/write.

Traps and enables
There are no traps and enables affecting this register.

Configurations
Available in all configurations.

Attributes
TRCCLAIMCLR is a 32-bit register.

The following figure shows the TRCCLAIMCLR bit assignments.

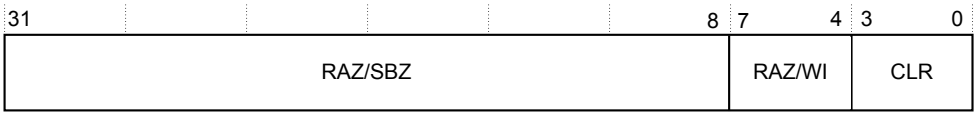

Figure 14-58 TRCCLAIMCLR bit assignments

The following table shows the TRCCLAIMCLR bit assignments.

Table 14-114 TRCCLAIMCLR bit assignments

Bits	Name	Function
[31:8]	-	RAZ/SBZ.
[7:4]	-	RAZ/WI.
[3:0]	CLR	On reads, for each bit: `0` Claim tag bit is not set. `1` Claim tag bit is set. On writes, for each bit: `0` Has no effect. `1` Clears the relevant bit of the claim tag.

TRCCLAIMCLR can be accessed through the memory mapped interface:

Table 14-115 TRCCLAIMCLR access information

Component	Offset	Reset
ETM	`0xFA4`	`0x00000000`

14.7.50 TRCDEVAFF0, Device Affinity Register 0

The TRCDEVAFF0 register provides a read-only copy of MPIDR accessible from the memory mapped interface and the external debug interface.

For more information on MPIDR, see *3.3.78 Multiprocessor Affinity Register* on page 3-167.

14.7.51 TRCDEVAFF1, Device Affinity Register 1

The TRCDEVAFF1 register has value `0x00000000`.

TRCDEVAFF1 is a read-only 32-bit register that can be accessed through the memory mapped interface or external debug interface.

14.7.52 Software Lock Access Register

The TRCLAR controls access to registers when **PADDRDBG31** is LOW. When the software lock is set, write accesses when **PADDRDBG31** is LOW to all ETM registers are ignored except for write accesses to the TRCLAR. When the software lock is set, read accesses of TRCPDSR with **PADDRDBG31** is LOW do not change the TRCPDSR.STICKYPD bit. Read accesses of all other registers are not affected. Accesses with **PADDRDBG31** HIGH are not affected by the software lock.

Usage constraints

This register is write-only and is only present for accesses with **PADDRDBG31** LOW, and is RES0 for accesses with **PADDRDBG31** HIGH.

Traps and enables

There are no traps and enables affecting this register.

Configurations

Available in all configurations.

Attributes

TRCLAR is a 32-bit register.

The following figure shows the TRCLAR bit assignments.

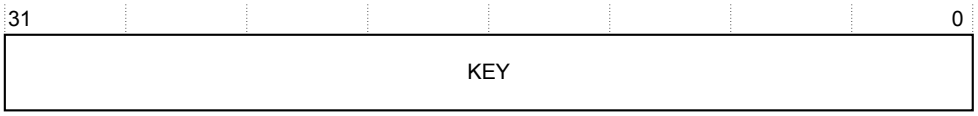

Figure 14-59 TRCLAR bit assignments

The following table shows the TRCLAR bit assignments.

Table 14-116 TRCLAR bit assignments

Bits	Name	Function
[31:0]	KEY	Software lock key value: `0xC5ACCE55` Clear the software lock. All other write values set the software lock.

TRCLAR can be accessed through the memory mapped interface:

Table 14-117 TRCLAR access information

Component	Offset	Reset
ETM	`0xFB0`	UNK

14.7.53 Software Lock Status Register

The TRCLSR indicates whether the software lock is implemented, and indicates the current status of the software lock.

Usage constraints
This register is read-only.

Configurations
Available in all configurations.

Attributes
TRCLSR is a 32-bit register.

The following figure shows the TRCLSR bit assignments.

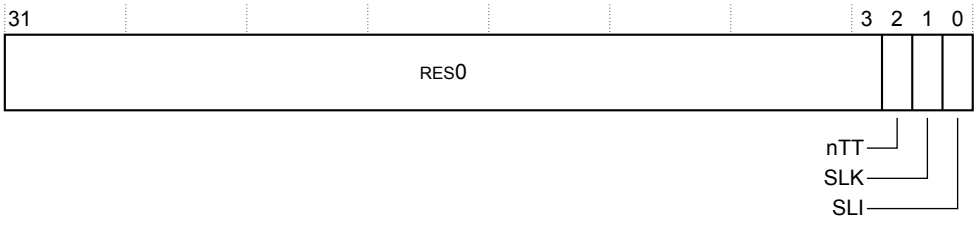

Figure 14-60 TRCLSR bit assignments

The following table shows the TRCLSR bit assignments.

Table 14-118 TRCLSR bit assignments

Bits	Name	Function
[31:3]	-	Reserved, RES0.
[2]	nTT	Indicates size of TRCLAR: 0 TRCLAR is always 32 bits.
[1]	SLK	Software lock status: 0 Software lock is clear. This value is returned for all accesses with **PADDRDBG31** HIGH. 1 Software lock is set.
[0]	SLI	Indicates whether the software lock is implemented on this interface: 0 Software lock is not implemented. This is returned for all accesses with **PADDRDBG31** HIGH. 1 Software lock is implemented. This is returned for all accesses with **PADDRDBG31** LOW.

TRCLSR can be accessed through the memory mapped interface and external debug interface:

Table 14-119 TRCLSR access information

Component	Offset	Reset
ETM	0xFB4	0x00000003

14.7.54 Authentication Status Register

The TRCAUTHSTATUS register indicates the current level of tracing permitted by the system.

Usage constraints
This register is read-only.

Traps and enables
There are no traps and enables affecting this register.

Configurations
Available in all configurations.

Attributes
TRCAUTHSTATUS is a 32-bit register.

The following figure shows the TRCAUTHSTATUS bit assignments.

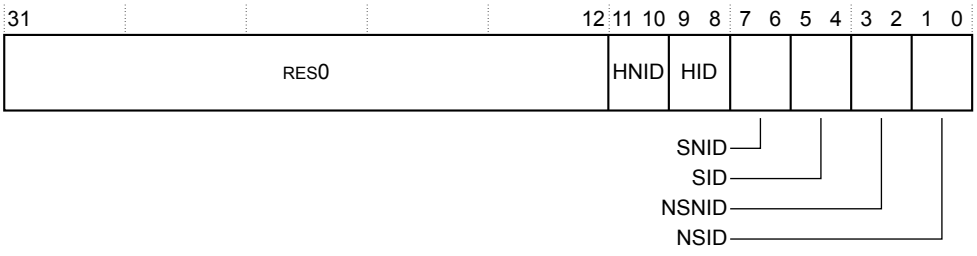

Figure 14-61 TRCAUTHSTATUS bit assignments

The following table shows the TRCAUTHSTATUS bit assignments.

Table 14-120 TRCAUTHSTATUS bit assignments

Bits	Name	Function
[31:12]	-	Reserved, RES0.
[11:10]	HNID	Non-Invasive Debug at EL2: 0b10 Implemented but disabled. 0b11 Implemented and enabled.
[9:8]	HID	Invasive Debug at EL2: 0b00 Not implemented.
[7:6]	SNID	Secure Non-Invasive Debug: 0b00 Not implemented.
[5:4]	SID	Secure Invasive Debug: 0b00 Not implemented.
[3:2]	NSNID	Non-Secure Non-Invasive Debug: 0b10 Implemented but disabled. 0b11 Implemented and enabled.
[1:0]	NSID	0b00 Not implemented.

TRCAUTHSTATUS can be accessed through the memory mapped interface:

Table 14-121 TRCAUTHSTATUS access information

Component	Offset	Reset
ETM	0xFB8	UNK

14.7.55 Device Architecture Register

The TRCDEVARCH register identifies the ETM as an ETMv4.2 component.

Usage constraints

This register is read-only.

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

TRCDEVARCH is a 32-bit register.

The following figure shows the TRCDEVARCH bit assignments.

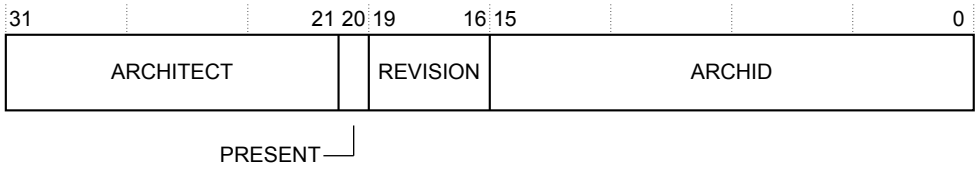

Figure 14-62 TRCDEVARCH bit assignments

The following table shows the TRCDEVARCH bit assignments.

Table 14-122 TRCDEVARCH bit assignments

Bits	Name	Function
[31:21]	ARCHITECT	Defines the architect of the component: `0x23B` Arm
[20]	PRESENT	Indicates the presence of this register: `1` Register is present.
[19:16]	REVISION	Architecture revision: `0b0010` Architecture revision 2.
[15:0]	ARCHID	Architecture ID: `0x4A13` ETMv4 component.

TRCDEVARCH can be accessed through the memory mapped interface:

Table 14-123 TRCDEVARCH access information

Component	Offset	Reset
ETM	`0xFBC`	`0x47724A13`

14.7.56 Device ID Register

The TRCDEVID register is RES0.

14.7.57 Device Type Register

The TRCDEVTYPE indicates the type of the component.

Usage constraints
This register is read-only.

Traps and enables
There are no traps and enables affecting this register.

Configurations
Available in all configurations.

Attributes
TRCDEVTYPE is a 32-bit register.

The following figure shows the TRCDEVTYPE bit assignments.

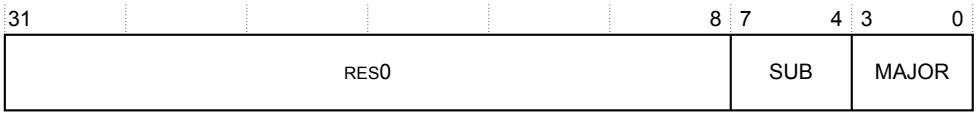

Figure 14-63 TRCDEVTYPE bit assignments

The following table shows the TRCDEVTYPE bit assignments.

Table 14-124 TRCDEVTYPE bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	SUB	The subtype of the component: `0b0001` Processor trace.
[3:0]	MAJOR	The main type of the component: `0b0011` Trace source.

TRCDEVTYPE can be accessed through the memory mapped interface and external debug interface:

Table 14-125 TRCDEVTYPE access information

Component	Offset	Reset
ETM	`0xFCC`	`0x00000013`

14.7.58 Peripheral Identification Registers

The TRCPIDR0-7 registers provide the standard Peripheral ID required by all CoreSight components.

See the *Arm® CoreSight™ Architecture Specification v2.0* for more information.

Usage constraints

These registers are read-only. Only bits[7:0] of each register are used. This means that TRCPIDR0-7 define a single 64-bit *Peripheral ID*, as the following figure shows.

Traps and enables

There are no traps and enables affecting these registers.

Configurations

Available in all configurations.

Attributes

The TRCPIDR0-7 registers are 32 bits.

The following figure shows the mapping between TRCPIDR0-7 and the single 64-bit *Peripheral ID* value.

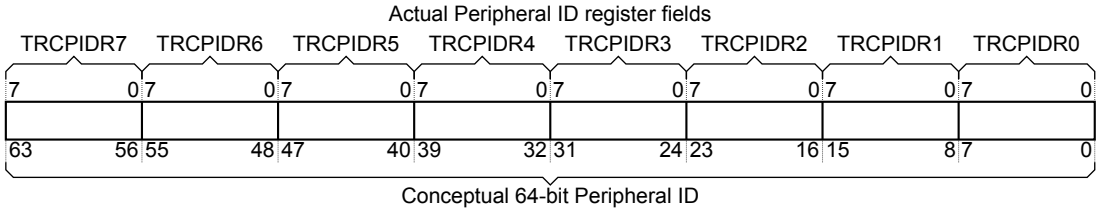

Figure 14-64 Mapping between TRCPIDR0-7 and the Peripheral ID value

The following figure shows the Peripheral ID bit assignments in the single conceptual Peripheral ID register.

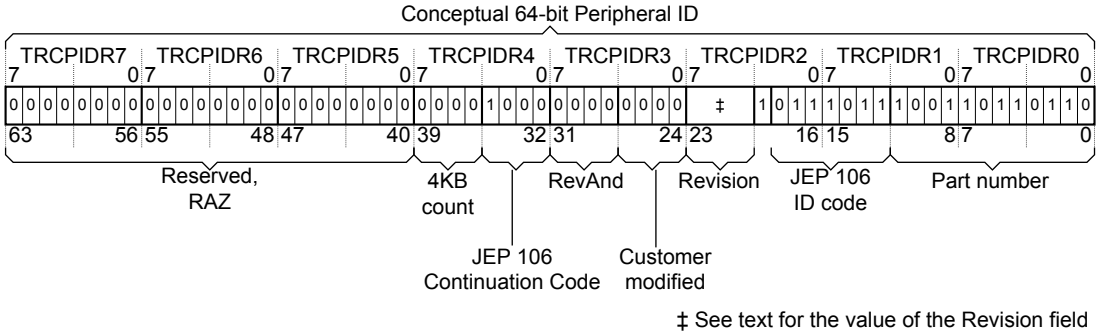

Figure 14-65 Peripheral ID fields

The following table shows the values of the fields when reading this set of registers. The *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* gives more information about many of these fields.

Note

The following table shows registers in order of register name, from most significant (TRCPIDR7) to least significant (TRCPIDR0). This does not match the order of the register offsets.

Table 14-126 TCRPIDR0-7 bit assignments

Register number	Register	Register offset	Bits	Value	Function
1015	TRCPIDR7	0xFDC	[31:8]	-	Reserved, RES0.
			[7:0]	0x00	Reserved, RES0.
1014	TRCPIDR6	0xFD8	[31:8]	-	Reserved, RES0.
			[7:0]	0x00	Reserved, RES0.
1013	TRCPIDR5	0xFD4	[31:8]	-	Reserved, RES0.
			[7:0]	0x00	Reserved, RES0.
1012	TRCPIDR4	0xFD0	[31:8]	-	Reserved, RES0.
			[7:4]	0x0	4KB
			[3:0]	0x4	JEP 106 continuation code.
1019	TRCPIDR3	0xFEC	[31:8]	-	Reserved, RES0.
			[7:4]	0x0	RevAnd. Part minor revision.
			[3:0]	0x0	Customer Modified. 0x0 indicates from Arm.
1018	TRCPIDR2	0xFE8	[31:8]	-	Reserved, RES0.
			[7:4]	-	Revision Number of ETM. This value is 0x4.
			[3]	1	Always 1. Indicates that a JEDEC assigned value is used.
			[2:0]	0x3	JEP 106 identity code [6:4].

Table 14-126 TCRPIDR0-7 bit assignments (continued)

Register number	Register	Register offset	Bits	Value	Function
1017	TRCPIDR1	0xFE4	[31:8]	-	Reserved, RES0.
			[7:4]	0xB	JEP 106 identity code [3:0].
			[3:0]	0x9	Part Number[11:8].
1016	TRCPIDR0	0xFE0	[31:8]	-	Reserved, RES0.
			[7:0]	0xB6	Part Number [7:0].

14.7.59 Component Identification Registers

The TRCCIDR0-3 registers identify the ETM as a CoreSight component.

For more information, see the *Arm® CoreSight™ Architecture Specification v2.0*.

Usage constraints

Only bits[7:0] of each register are used. This means that TRCCIDR0-3 define a single 32-bit Component ID.

Traps and enables

There are no traps and enables affecting these registers.

Configurations

Available in all configurations.

Attributes

The TRCCIDR0-3 registers are 32 bits.

The following figure shows the mapping between TRCCIDR0-3 and the single 32-bit *Component ID* value.

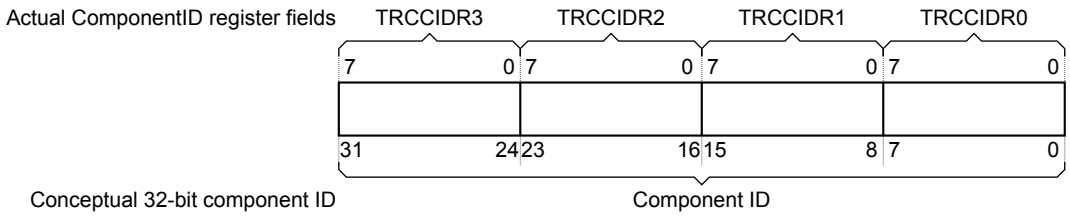

Figure 14-66 Mapping between TRCCIDR0-3 and the Component ID value

The following table shows the Component ID bit assignments in the single conceptual Component ID register.

Note

The following table lists registers in name order, from most significant (TRCCIDR3) to least significant (TRCCIDR0). This does not match the order of the register offsets.

Table 14-127 TRCCIDR0-3 bit assignments

Register	Register number	Register offset	Bits	Value	Function
TRCCIDR3	0x3FF	0xFFC	[31:8]	-	Reserved, RES0.
			[7:0]	0xB1	Component identifier, bits[31:24].
TRCCIDR2	0x3FE	0xFF8	[31:8]	-	Reserved, RES0.
			[7:0]	0x05	Component identifier, bits[23:16].
TRCCIDR1	0x3FD	0xFF4	[31:8]	-	Reserved, RES0.
			[7:4]	0x9	Debug component with CoreSight compatible registers (component identifier, bits[15:12]).
			[3:0]	0x0	Component identifier, bits[11:8].
TRCCIDR0	0x3FC	0xFF0	[31:8]	-	Reserved, RES0.
			[7:0]	0x0D	Component identifier, bits[7:0].

Chapter 15
Advanced SIMD and floating-point support

This chapter describes the Advanced SIMD and floating-point features and registers the processor uses.

It contains the following sections:

15.1 About the Advanced SIMD and floating-point support

The Cortex-R52 processor supports single-precision floating-point instructions and can optionally support the double-precision floating-point and Advanced SIMD instructions.

15.2 Floating-point support

The Cortex-R52 floating-point implementation:

- Does not support trapping of floating-point exceptions.
- Does not support the deprecated floating-point short vector feature.
- Implements all operations in hardware, with support for all combinations of:
  - Rounding modes.
  - Flush-to-zero mode, which can be enabled or not enabled.
  - Default *Not a Number* (NaN) mode, which can be enabled or not enabled.

15.3 AArch32 single-precision floating point instructions

A single-precision only configuration has fewer registers and instructions than a full implementation.

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

15.4 Accessing the feature identification registers

Software can identify the Advanced SIMD and floating-point features using the feature identification registers.

You can access the feature identification registers 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
```

15.5 Register summary

The following table gives a summary of the Cortex-R52 processor Advanced SIMD and floating-point system registers.

Table 15-1 Advanced SIMD and floating-point system registers

Name	Type	Reset	Description
FPSID	RO	`0x41034023`	See *15.6.1 Floating-point System ID Register* on page 15-534.
FPSCR	RW	`0x00000000`	See *15.6.2 Floating-point Status and Control Register* on page 15-535.
MVFR0	RO	`0x10110222` Full Advanced SIMD config `0x10110021` SP-only config	See *15.6.3 Media and Floating-point Feature Register 0* on page 15-537.
MVFR1	RO	`0x12111111` Full Advanced SIMD config `0x11000011` SP-only config	See *15.6.4 Media and Floating-point Feature Register 1* on page 15-539.
MVFR2	RW	`0x00000043` Full Advanced SIMD config `0x00000040` SP-only config	See *15.6.5 Media and Floating-point Feature Register 2* on page 15-540.
FPEXC	RW	`0x00000700`	See *15.6.6 Floating-Point Exception Control Register* on page 15-541.

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 Supplement Armv8, for Armv8-R architecture profile* for information about permitted accesses to the Advanced SIMD and floating-point system registers.

15.6 Register descriptions

This section describes the Advanced SIMD and floating-point system registers in the Cortex-R52 processor.

This section contains the following subsections:

15.6.1 Floating-point System ID Register

The FPSID provides top-level information about the floating-point implementation.

Usage constraints The accessibility to the FPSID by Exception level is:

EL0	EL1	EL2
-	Config	Config

This register largely duplicates information held in the MIDR. Arm deprecates use of it.

Traps and enables If CPACR.cp10 is `0b00`, then accesses to this register from EL1 are UNDEFINED.

If HCPTR.TCP10 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode and accesses to this register from EL2 are UNDEFINED.

If HCR.TID0 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

Configurations Available in all configurations.

Attributes The FPSID is a 32-bit register.

The following figure shows the FPSID bit assignments.

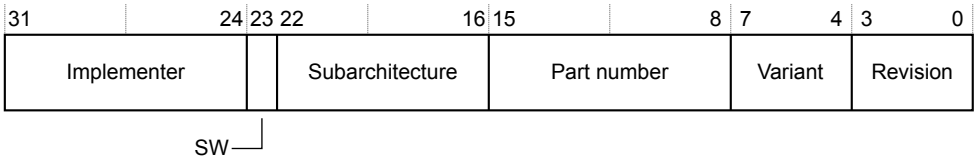

Figure 15-1 FPSID bit assignments

The following table shows the FPSID bit assignments.

Table 15-2 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.

Table 15-2 FPSID bit assignments (continued)

Bits	Name	Function
[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-R52 processor.
[7:4]	Variant	Indicates the variant number: `0x2` Cortex-R52 processor.
[3:0]	Revision	Indicates the revision number for the floating-point implementation: `0x4` r1p3.

To access the FPSID register:

```
VMRS <Rt>, FPSID ; Read FPSID into Rt
```

15.6.2 Floating-point Status and Control Register

The FPSCR provides floating-point system status information and control.

Usage constraints The accessibility to the FPSCR by Exception level is:

EL0	EL1	EL2
Config	Config	Config

Traps and enables If CPACR.cp10 is `0b00`,then accesses to this register from EL0 and EL1 are UNDEFINED.

If CPACR.cp10 is `0b01`,then accesses to this register from EL0 UNDEFINED.

If HCPTR.TCP10 is set to 1, then accesses to this register from EL0 and EL1 are trapped to Hyp mode and accesses to this register from EL2 are UNDEFINED

Configurations Available in all configurations.

Attributes The FPSCR is a 32-bit register.

The following figure shows the FPSCR bit assignments.

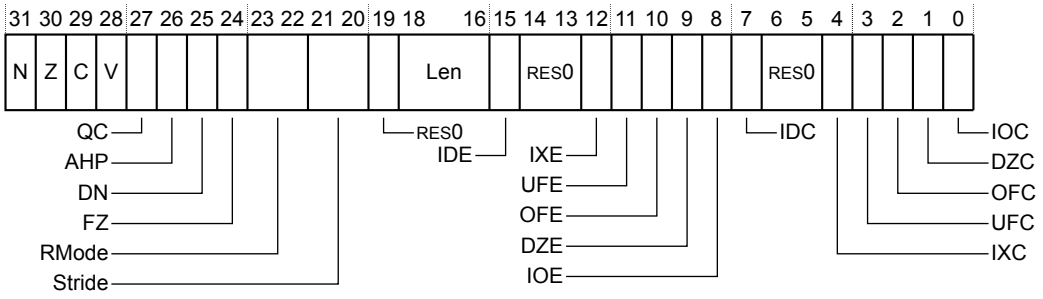

Figure 15-2 FPSCR bit assignments

The following table shows the FPSCR bit assignments.

Table 15-3 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. 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. Advanced SIMD arithmetic always uses the Flush-to-zero setting, regardless of the value of the FZ bit.
[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. Advanced SIMD arithmetic always uses the Round to Nearest setting, regardless of the value of the RMode bits.
[21:20]	Stride	This field is RES0.

Table 15-3 FPSCR bit assignments (continued)

Bits	Field	Function
[19]	-	Reserved, RES0.
[18:16]	Len	This field is RAZ.
[15]	IDE	This field is RES0.
[14:13]	-	Reserved, RES0.
[12]	IXE	This field is RES0.
[11]	UFE	This field is RES0.
[10]	OFE	This field is RES0.
[9]	DZE	This field is RES0.
[8]	IOE	This field is 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
```

15.6.3 Media and Floating-point Feature Register 0

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	EL1	EL2
-	Config	Config

Traps and enables If CPACR.cp10 is `0b00`, then read accesses to this register from EL1 are UNDEFINED.

If HCPTR.TCP10 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode and read accesses to this register from EL2 are UNDEFINED

If HCR.TID3 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

Configurations Available in all configurations.

Attributes The MVFR0 is a 32-bit register.

The following figure shows the MVFR0 bit assignments.

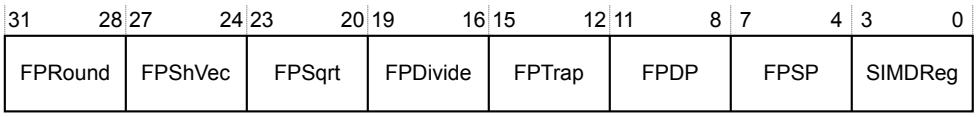

Figure 15-3 MVFR0 bit assignments

The following table shows the MVFR0 bit assignments.

Table 15-4 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: `0x0` Not supported in hardware when the processor is configured without optional double-precision floating-point and NEON technology support. `0x2` Supported, VFPv3 or greater, when the processor is configured with optional double-precision floating-point and NEON technology support. See the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* for more information.

Table 15-4 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 Supplement Armv8, for Armv8-R architecture profile* for more information.
[3:0]	SIMDReg	Indicates support for the Advanced SIMD register bank: `0x1` Supported, 16 x 64-bit registers, when the processor is configured without optional double-precision floating-point and NEON technology support. `0x2` Supported, 32 x 64-bit registers, when the processor is configured with optional double-precision floating-point and NEON technology support. See the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* for more information.

To access the MVFR0:

```
VMRS <Rt>, MVFR0 ; Read MVFR0 into Rt
```

15.6.4 Media and Floating-point Feature Register 1

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	EL1	EL2
-	Config	Config

Traps and enables If CPACR.cp10 is `0b00`, then read accesses to this register from EL1 are UNDEFINED.

If HCPTR.TCP10 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode and read accesses to this register from EL2 are UNDEFINED.

If HCR.TID3 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

Configurations Available in all configurations.

Attributes The MVFR1 is a 32-bit register.

The following figure shows the MVFR1 bit assignments.

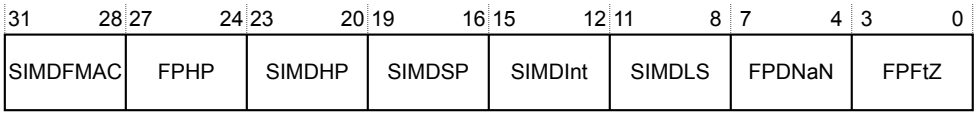

Figure 15-4 MVFR1 bit assignments

The following table shows the MVFR1 bit assignments.

Table 15-5 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: `0x1` Instructions to convert between half-precision and single-precision implemented in hardware when the processor is configured without optional double-precision floating-point and NEON technology support. `0x2` Supported, instructions to convert between half-precision and either single-precision or double-precision implemented when the processor is configured with optional double-precision floating-point and NEON technology support.
[23:20]	SIMDHP	Indicates whether Advanced SIMD and floating-point supports half-precision floating-point conversion operations: `0x0` Not implemented. `0x1` Implemented. This value is permitted only if the SIMDSP field is `0x1`.
[19:16]	SIMDSP	Indicates whether Advanced SIMD and floating-point supports single-precision floating-point operations: `0x0` Not implemented `0x1` Implemented. This value is permitted only if the SIMDINt field is `0x1`.
[15:12]	SIMDInt	Indicates whether Advanced SIMD and floating-point supports integer operations: `0x0` Not implemented. `0x1` Implemented.
[11:8]	SIMDLS	Indicates whether Advanced SIMD and floating-point supports load/store instructions: `0x0` Implemented. `0x1` Not 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
```

15.6.5 Media and Floating-point Feature Register 2

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:

Table 15-6

ELO	EL1	EL2
-	Config	Config

Traps and enables If CPACR.cp10 is `0b00`, then read accesses to this register from EL1 are UNDEFINED.

If HCPTR.TCP10 is set to 1, then read accesses to this register from EL1 are trapped in Hyp mode and read accesses to this register from EL2 are UNDEFINED.

If HCR.TID3 is set to 1, then read accesses to this register from EL1 are trapped to Hyp mode.

Configurations Available in all configurations.

Attributes The MVFR2 is a 32-bit register.

The following figure shows the MVFR2 bit assignments.

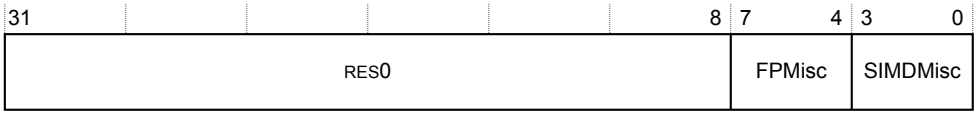

Figure 15-5 MVFR2 bit assignments

The following table shows the MVFR2 bit assignments.

Table 15-7 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. `0x0` Not implemented. `0x3` Implemented.

To access the MVFR2:

```
VMRS <Rt>, MVFR2 ; Read MVFR2 into Rt
```

15.6.6 Floating-Point Exception Control Register

The FPEXC 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	EL1	EL2
-	Config	Config

Traps and enables If CPACR.cp10 is `0b00`, then accesses to this register from EL1 are UNDEFINED.

If HCPTR.TCP10 is set to 1, then accesses to this register from EL1 are trapped to Hyp mode and accesses to this register from EL2 are UNDEFINED.

Configurations Available in all configurations.

Attributes FPEXC is a 32-bit register.

The following figure shows the FPEXC bit assignments.

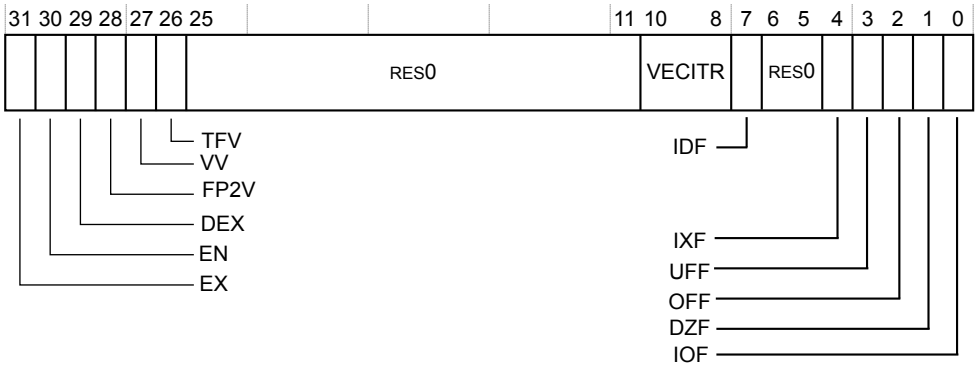

Figure 15-6 FPEXC bit assignments

The following table shows the FPEXC bit assignments.

Table 15-8 FPEXC bit assignments

Bits	Name	Function
[31]	EX	This field 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.
[29]	DEX	This field is RES0.
[28]	FP2V	This field is RES0.
[27]	VV	This field is RES0.
[26]	TFV	This field is RAZ/WI.
[25:11]	-	Reserved, RES0.
[10:8]	VECITR	This field is RES1.
[7]	IDF	This field is RAZ/WI.
[6:5]	-	Reserved, RES0.

Table 15-8 FPEXC bit assignments (continued)

Bits	Name	Function
[4]	IXF	This field is RAZ/WI.
[3]	UFF	This field is RAZ/WI.
[2]	OFF	This field is RAZ/WI.
[1]	DZF	This field is RAZ/WI.
[0]	IOF	This field is RAZ/WI.

To access the FPEXC register:

```
VMRS <Rt>, FPEXC ; Read FPEXC into Rt
VMRS FPEXC, <Rt> ; Write Rt to FPEXC
```

Appendix A
Signal Descriptions

This appendix describes the Cortex-R52 processor signals.

Note

Some of the signals appear twice in this appendix. For example, the AXIM interface clock enable signal is listed in *A.1 Clock and clock enable signals* on page Appx-A-545 and *AXIM clock enable signal* on page Appx-A-560.

It contains the following sections:

A.1 Clock and clock enable signals

This section shows the Cortex-R52 processor clock and clock enable signals.

Note

You can stop the processor clock indefinitely without loss of state.

The following table shows the clock signals.

Table A-1 Clock signals

Signal	Direction	Description
CLKIN	Input	Main clock.
CLKINDCLS	Input	Redundant clock for the redundant logic in lock-step configurations.

The following table shows the clock enable signals.

Table A-2 Clock enable signals

Signal	Direction	Description
PCLKENDBG	Input	APB clock enable.
ACLKENMx	Input	AXIM interface clock enable.
ACLKENS	Input	AXIS interface clock enable.
ATCLKEND	Input	ATB clock enable for data trace.
ATCLKENI	Input	ATB clock enable for instruction trace and clock enable for **TSVALUEB[63:0]**.
ACLKENFx	Input	Flash interface clock enable.
CNTCLKEN	Input	Counter clock enable **CNTVALUEB**.
ACLKENPx	Input	LLPP clock enable.

A.2 Resets

The following table shows the reset signals and reset control signals.

——— **Note** ———

If DCLS is configured, all reset signals must be synchronous.

———————————

Table A-3 Reset signals

Signal	Direction	Description
nCORERESETx	Input	Individual core warm reset. **0** Apply reset to the core x excluding debug and trace logic. **1** Do not apply reset to the core.
nCPUPORESETx	Input	Individual core powerup cold reset. **0** Apply reset to the core x including debug and trace logic. **1** Do not apply reset to the core.
nTOPRESET	Input	Top-level reset. Resets all the top-level functional logic. **0** Reset all the top-level functional logic. **1** Do not reset all the top-level functional logic.
nCORERESETDCLSx	Input	Individual redundant core warm reset. Must be driven by the same source as the equivalent **nCORERESETx** input. **0** Apply reset to the redundant core x excluding debug and trace logic. **1** Do not apply reset to the redundant core.
nCPUPORESETDCLSx	Input	Individual redundant core powerup cold reset. Must be driven by the same source as the equivalent **nCPUPORESETx** input. **0** Apply reset to the redundant core x including debug and trace logic. **1** Do not apply reset to the redundant core.
nTOPRESETDCLS	Input	Top-level redundant logic reset. Resets all the top-level functional redundant logic in the processor, this input must be identical to **nTOPRESET**. **0** Reset all the top-level functional redundant logic. **1** Do not reset all the top-level functional redundant logic.
nPRESETDBG	Input	APB reset: **0** Reset APB and top-level debug logic. **1** Do not reset APB and top-level debug logic.
nMBISTRESET	Input	MBIST reset. **0** Reset MBIST. **1** Do not reset MBIST.

A.3 Reset-related signals

Each core has two reset request outputs which it can assert to request a warm reset. Whether these signals are factored into the reset inputs is determined by the reset control logic external to the Cortex-R52 processor.

The following table shows the reset-related signals.

Table A-4 Reset-related signals

Signal	Direction	Description
WARMRSTREQx	Output	Warm core reset request.
DBGRSTREQx	Output	Request for reset from external debug logic.
CPUHALTx	Input	Core waits out of reset before taking reset exception and fetching instructions.
CFGINITREG	Input	Program-visible registers initialized to fixed value out of reset.
CFGL1CACHEINVDISx	Input	Automatic post-reset L1 cache invalidate disable.

WARMRSTREQx is asserted when software writes 1 to HRMR.RR. **DBGRSTREQx** is asserted when the debugger writes 1 to the EDPRCR.CWRR.

A.4 Configuration inputs

The following table shows the configuration inputs.

Table A-5 Configuration inputs

Signal	Direction	Description
CFGAXISTCMBASEADDR[31:24]	Input	Base address of the TCMs on the AXIS interface.
CFGCLUSTERUTID[1:0]	Input	Cluster unique transaction identifier for the purpose of interconnect protection.
CFGDBGROMADDRV	Input	Top-level debug ROM table address enable.
CFGDBGROMADDR[31:12]	Input	Top-level debug ROM table base address.
CFGENDIANESSx	Input	Data endianness (little-endian or byte-invariant big-endian) and value of HSCTLR.EE out of reset.
CFGFLASHBASEADDR[31:27]	Input	Base address of the Flash interface.
CFGFLASHENx	Input	Flash interface enabled or disabled out of reset.
CFGFLASHPROTEN	Input	Flash memory protection enable out of reset.
CFGFLASHPROTIMP	Input	Flash memory protection support.
CFGFLASHIMP	Input	Flash region implemented in memory map.
CFGLLPPIMP	Input	LLPP region implemented in memory map.
CFGLLPPBASEADDR[31:12]	Input	Base address of the LLPP.
CFGLLPPSIZE[3:0]	Input	Region size of the LLPP.
CFGMRPEN	Input	Memory reconstruction port enable.
CFGMPIDRAFF1[7:0]	Input	Cluster ID at affinity level 1.
CFGMPIDRAFF2[7:0]	Input	Cluster ID at affinity level 2.
CFGPERIPHBASE[31:21]	Input	Base address of the GIC Distributor Unit (GDU) registers
CFGRAMPROTEN	Input	RAM (caches and TCMs) memory protection enable out of reset.
CFGTCMBOOTx	Input	ATCM enabled and at address `0x0` out of reset.
CFGTHUMBEXCEPTIONSx	Input	Instruction set state (A32 or T32) and value of HSCTLR.TE out of reset.
CFGVECTABLEx[31:5]	Input	Vector table base address out of reset.
CFGSLSPLIT	Input	Split/Lock implementation. **0** Lock mode. **1** Split mode.

A.5 Memory correcting error reporting signals

The processor has primary and secondary memory error reporting interfaces to identify the source of the highest and next highest priority correctable memory error. A system safety controller can use these interfaces to build a log of the locations of correctable errors detected in the memories.

The following table shows the error reporting signals. These signals are valid for a single cycle whenever the associated valid bit is asserted.

Table A-6 Memory correcting error reporting signals

Signal	Direction	Description
PRIMEMERRIDXx[24:0]	Output	Primary error reporting interface index.
PRIMEMERRMEMx[14:0]	Output	Primary error reporting interface memory identifier.
PRIMEMERRVx	Output	Primary error reporting interface valid.
SECMEMERRIDXx[24:0]	Output	Secondary error reporting interface index.
SECMEMERRMEMx[14:0]	Output	Secondary error reporting interface memory identifier.
SECMEMERRVx	Output	Secondary error reporting interface valid.

Note

PRIMEMERRIDXx[24:0], **PRIMEMERRMEMx[14:0]**, **SECMEMERRIDXx[24:0]**, and **SECMEMERRMEMx[14:0]** are valid when **PRIMEMERRVx** and **SECMEMERRVx** are valid.

The interfaces indicate the RAM that is the source of the error on the **PRIMEMERRMEM** and **SECMEMERRMEM** signals. The following tables show the bits and their encoding, which varies according to the block indicated in bits [2:0].

Table A-7 ATCM

Bits	Function
[14:3]	These bits are unused for ATCM because it is a single bank.
[2:0]	Block in which the error was found. The value is `0b000`.

Table A-8 BTCM

Bits	Function
[14:3]	Bits indicating the RAM bank or banks within the BTCM in which correctable error or errors were detected. Multiple bits can be set. Bits [14:5] are unused. The possible values are: **[4]** Bank 1. **[3]** Bank 0.
[2:0]	Block in which the error was found. The value is `0b001`.

Table A-9 CTCM

Bits	Function
[14:3]	Bits indicating the RAM bank or banks within the CTCM in which correctable error or errors were detected. Multiple bits can be set. Bits [14:5] are unused. The possible values are: **[4]** Bank 1. **[3]** Bank 0.
[2:0]	Block in which the error was found. The value is `0b010`.

Table A-10 Flash

Bits	Function
[14:3]	These bits are unused for flash because it is a single bank.
[2:0]	Block in which the error was found. The value is `0b011`.

Table A-11 L1 instruction cache

Bits	Function
[14:3]	Bits indicating the RAM bank or banks within the instruction cache in which correctable error or errors were detected. Multiple bits can be set. Bits [14: 11] are unused. The possible values are: **[10]** Tag RAM, bank 3. **[9]** Tag RAM, bank 2. **[8]** Tag RAM, bank 1. **[7]** Tag RAM, bank 0. **[6]** Data RAM, bank 3. **[5]** Data RAM, bank 2. **[4]** Data RAM, bank 1. **[3]** Data RAM, bank 0.
[2:0]	Block in which the error was found. The value is `0b100`.

Table A-12 L1 data cache

Bits	Function
[14:3]	Bits indicating the RAM bank or banks within the data cache in which correctable error or errors were detected. Multiple bits can be set. The possible values are: **[14]** Tag RAM, bank 3. **[13]** Tag RAM, bank 2. **[12]** Tag RAM, bank 1. **[11]** Tag RAM, bank 0. **[10]** Data RAM, bank 7. **[9]** Data RAM, bank 6. **[8]** Data RAM, bank 5. **[7]** Data RAM, bank 4. **[6]** Data RAM, bank 3. **[5]** Data RAM, bank 2. **[4]** Data RAM, bank 1. **[3]** Data RAM, bank 0.
[2:0]	Block in which the error was found. The value is `0b101`.

PRIMEMERRINDEX and **SECMEMERRINDEX** show the row in the RAM that has the error.

For TCM errors

8 *Most Significant Bits* (MSBs) unused.

17 *Least Significant Bits* (LSBs) is the INDEX value that corresponds to the access address [19:3] masked with the TCM size. The following table shows the bits corresponding to the TCM size.

Table A-13 TCM errors

Bits	Size
[19:3]	1MB
[18:3]	512KB
[17:3]	256KB
[16:3]	128KB
[15:3]	64KB
[14:3]	32KB
[13:3]	16KB
[12:3]	8KB

For flash errors

All bits are used. The INDEX value that corresponds to the access address is [25:1]. In a normal scenario, bits [25:4] would be adequate as the flash region is 128MB and the accesses are 64-bit aligned. However, in Cortex-R52, the *Flash Error Record Registers* (FLASHERR) log accesses aligned to halfwords. This is done to capture the exact instruction that caused an error for instruction-side errors which are corrected inline.

For instruction and data cache errors

18 MSBs unused.

7 LSBs is the INDEX value that corresponds to the instruction cache index or data cache index and corresponds to bits [12:6] masked with the cache size. The following table shows the bits corresponding to the cache size.

Table A-14 Instruction and data cache errors

Bits	Size
[12:6]	32KB
[11:6]	16KB
[10:6]	8KB
[9:6]	4KB

A.6 Event output signals

These buses collate the error signals within each core. The system safety controller can use these signals to identify more closely the nature of any error detected in the system. This helps the system safety controller to determine its response to the error.

The following table shows the event output signals. An event occurrence is indicated for every clock cycle that the event signal is asserted HIGH.

Table A-15 Event output signals

Signal	Direction	Description
ERREVENT[4:0]	Output	Global error events. The bit assignment is: **[0]** Correctable data payload bus error occurred from AXIS. **[1]** Fatal data payload bus error occurred from AXIS. **[2]** Fatal READY signal bus error occurred from AXIS. **[3]** Fatal non-protocol bus errors (payload errors) occurred from AXIS. **[4]** Fatal protocol bus error (LEN, ID, LAST, READY, and interconnect protection errors) occurred from AXIS.

Table A-15 Event output signals (continued)

Signal	Direction	Description
ERREVENTx[25:0]	Output	Per-core error events. The bit assignment is: **[0]** Correctable memory error detected from L1 instruction cache, L1 data cache, ATCM, BTCM, CTCM, or flash. The PMU mnemonic is KITE_CORE_ERR_MEM. **[1]** Fatal memory error occurred from ATCM, BTCM, CTCM, or flash. The PMU mnemonic is KITE_FAT_ERR_MEM. **[2]** Correctable data payload bus error occurred from AXIM or LLPP. The PMU mnemonic is KITE_BUS_COR_DATA. **[3]** Fatal data payload bus error occurred from AXIM or LLPP. **[4]** Fatal READY signal bus error occurred from AXIM, Flash, or LLPP. **[5]** Fatal non-protocol bus error occurred from AXIM, Flash, or LLPP. These are payload errors. The PMU mnemonic is KITE_BUS_FAT_ANY. **[6]** Fatal protocol bus error occurred from AXIM, Flash, or LLPP. These are LEN, ID, LAST, VALID and READY, and interconnect protection errors. The mnemonic is KITE_BUS_PROTOCOL_ANY. **[7]** Correctable L1 instruction cache data or L1 instruction cache tag memory error. **[8]** Correctable L1 data cache data or L1 data cache tag memory error. **[9]** Correctable ATCM, BTCM, or CTCM memory error. **[10]** Fatal ATCM, BTCM, or CTCM memory error. **[11]** Correctable flash memory error. **[12]** Fatal flash memory error. **[13]** Timeout for main AXIM bus. **[14]** Timeout for Flash bus. **[15]** Timeout for LLPP bus. **[16]** Correctable memory error occurred L1 instruction cache, L1 data cache, ATCM, BTCM, CTCM, or flash, but was not recorded in the *Error Record Registers* (ERRs), because they were full or because it was overwritten by a fatal error. **[17]** Fatal memory error occurred L1 instruction cache, L1 data cache, ATCM, BTCM, CTCM, or flash, but was not recorded in the ERRs, because they were full. **[18]** Interrupts masked in Hyp mode for too long. The IMP_INTMONR watchdog is triggered. **[19]** A memory or bus fatal error was synchronously detected in Hyp mode. **[20]** An abort exception was taken because an EL2-controlled MPU fault generated an abort. **[21]** An Abort exception was taken because an EL1-controlled MPU fault generated an abort. **[22]** An Undefined exception was taken due to any cause. **[23]** A memory access marked in EL1-controlled MPU as Device was flushed because it was marked as Normal in EL2-controlled MPU. **[24]** Processor livelock because of hard errors or exception at exception vector. An exception of the same type and vector offset was consecutively taken 20 times. **[25]** Software running in EL2 unlocks IMP_TESTR1. IMP_TESTR1 is for testing purposes only.

Table A-15 Event output signals (continued)

Signal	Direction	Description
PMUEVENTx[22:0]	Output	Performance Monitoring Unit events. **[0]** L1 instruction cache refill. The PMU mnemonic is L1I_CACHE_REFILL. **[1]** L1 data cache refill. The PMU mnemonic is L1D_CACHE_REFILL. **[2]** L1 data cache access. The PMU mnemonic is L1I_CACHE. **[3]** Instruction architecturally executed, condition check pass-load. The PMU mnemonic is LD_RETIRED. **[4]** Instruction architecturally executed, condition check pass-store. The PMU mnemonic is SD_RETIRED. **[5]** Instruction architecturally executed. The PMU mnemonic is INST_RETIRED. **[6]** Two instructions architecturally executed in parallel. Asserted with **PMUEVENTx[5]** when dual-issuing. **[7]** Exception taken. The PMU mnemonic is EXC_TAKEN. **[8]** Instruction architecturally executed, condition code check pass, exception return. The PMU mnemonic is EXC_RETURN. **[9]** Instruction architecturally executed, condition code check pass, write to CONTEXTIDR. The PMU mnemonic is CID_WRITE_RETIRED. **[10]** Instruction architecturally executed, condition code check pass, software change of the PC. The PMU mnemonic is PC_WRITE_RETIRED. **[11]** Instruction architecturally executed, immediate branch. The PMU mnemonic is BR_IMMED_RETIRED. **[12]** Instruction architecturally executed, condition code check pass, unaligned load or store. The PMU mnemonic is UNALIGNED_LDST_RETIRED. **[13]** Mispredicted or unpredicted branch speculatively executed. The PMU mnemonic is BS_MIS_PRED. **[14]** Predictable branch speculatively executed. The PMU mnemonic is BR_PRED. **[15]** Data memory access. The PMU mnemonic is MEM_ACCESS. **[16]** L1 instruction cache access. The PMU mnemonic is L1I_CACHE. **[17]** Exception taken, IRQ. The PMU mnemonic is EXC_IRQ. **[18]** Exception taken, FIQ. The PMU mnemonic is EXC_FIQ. **[19]** Exception taken, Hypervisor call. The PMU mnemonic is EXC_HVC. **[20]** Exception taken, IRQ not taken locally. The PMU mnemonic is EXC_TRAP_IRQ. **[21]** Exception taken, FIQ not taken locally. The PMU mnemonic is EXC_TRAP_FIQ. **[22]** Exception taken to EL2 (Hypervisor mode entry), excluding reset. The PMU mnemonic is KITE_EL2_ENTERED.

Table A-15 Event output signals (continued)

Signal	Direction	Description
PMUEVENTx[35:23]	Output	**[23]** Exception taken, supervisor call. The PMU mnemonic is EXC_SVC. **[24]** Instruction architecturally executed, branch.The PMU mnemonic is BR_RETIRED. **[25]** Exception taken to EL2 (Hypervisor mode entry), excluding reset. The PMU mnemonic is KITE_EL2_ENTERED. **[26]** Exception taken to EL2 (Hypervisor mode entry), excluding reset. The PMU mnemonic is KITE_EL2_ENTERED. **[27]** Exception taken to EL2 (Hypervisor mode entry), excluding reset. The PMU mnemonic is KITE_EL2_ENTERED. **[28]** Instruction architecturally executed, condition code check pass, procedure return.The PMU mnemonic is BR_RETURN_RETIRED. **[29]** Exception taken to EL2 (Hypervisor mode entry), excluding reset. The PMU mnemonic is KITE_EL2_ENTERED. **[30]** Exception taken to EL2 (Hypervisor mode entry), excluding reset. The PMU mnemonic is KITE_EL2_ENTERED. **[31]** External memory request, AXIM read. The PMU mnemonic is KITE_AXI_READ. **[32]** External memory request, AXIM write.The PMU mnemonic is KITE_AXI_WRITE. **[33]** External memory request, Flash (read-only).The PMU mnemonic is KITE_FLASH_READ. **[34]** External memory request, LLPP read. The PMU mnemonic is KITE_LLPP_READ. **[35]** External memory request, LLPP write. The PMU mnemonic is KITE_LLPP_WRITE.

A.7 MRP signals

Each core of the Cortex-R52 processor provides an MRP for reporting write accesses so that an image of memory can be reconstructed.

The MRP is always present and has a control to enable or disable it. When the MRP is enabled, the store unit only accepts one store transaction per cycle to serialize stores into the store unit. The store unit outputs all write transactions to the MRP regardless of whether the source of the transaction is the core or an AXI master attached to the slave interface of the processor.

Note

The MRP is driven off **CLKIN** without support for a slower bus.

The following table shows the MRP signals.

Table A-16 MRP signals

Signal	Direction	Description
MRPBACKPRESSx	Input	Apply back-pressure to MRP generation. This signal should be asserted when the MRP trace generation logic is soon unable to accept more data. Typically this is when a FIFO holding the MRP trace information is almost full. The MRP can send the master can send up to three more transactions after the signal goes high and the back-pressure must be applied until it has taken effect.
MRPATTRx[4:0]	Output	Reflects the attribute encoding for inner-cacheability and inner-shareability output by the processor.
MRPCSx[4:0]	Output	Chip select signal which is one-hot and reflects the memory region targeted by the write.
MRPSOURCEx	Output	Indicates the source of the write (AXIS or core).
MRPADDRx[31:0]	Output	Address of the write access.
MRPDATAx[63:0]	Output	Data of the write access.
MRPSTRBx[7:0]	Output	Byte strobes of the write access.
MRPVALIDx	Output	Indicates the processor is committing a new write. There is no corresponding handshake input signal.
CFGMRPEN	Input	MRP enable. This signal can only change under reset. If the signal is HIGH, MRP is enabled.

The following tables show the attribute encoding for **MRPATTRx[4:0]**.

Table A-17 Attribute encoding for MRPATTRx[4:2]

MRPATTRx[4:2]	Attribute
0b000	Device-nGnRnE
0b001	Device-nGnRE
0b010	Device-nGRE
0b011	Device-GRE
0b100	Normal Inner NC or AXIS write
0b101	Normal Inner WB
0b110	Normal Inner WT
0b111	AXIS write, which was reordered

Table A-18 Attribute encoding for MRPATTRx[1:0]

MRPATTRx[1:0]	Attribute
0b00	Non-shareable or AXIS write
0b01	UNUSED
0b10	Outer-shareable
0b11	Inner-shareable

The following table shows the attribute encoding for **MRPCSx[4:0]**.

Table A-19 Attribute encoding for MRPCSx[4:0]

MRPCSx[4:0]	Attribute
0b00001	AXI
0b00010	ATCM
0b00100	BTCM
0b01000	CTCM
0b10000	LLPP

The following table shows the source encoding for **MRPSOURCEx**.

Table A-20 Attribute encoding for MRPSOURCEx

MRPSOURCEx	Attribute
0	Core
1	AXIS

A.8 Bus interface signals

This section describes the bus interface signals.

This section contains the following subsections:

A.8.1 AXIM interface signals

This section describes the AXIM interface signals.

AXIM clock enable signal

The following table shows the clock enable signal for the AXIM interface.

Table A-21 AXIM clock enable signal

Signal	Direction	Description
ACLKENMx	Input	Clock enable for the AXIM interface.

AXIM read address channel signals

The following table shows the read address channel signals for the AXIM interface.

Table A-22 AXIM read address channel signals

Signal	Direction	Description
ARREADYMx	Input	Read address ready.
ARADDRMx[31:0]	Output	Read address.
ARBURSTMx[1:0]	Output	Read burst type.
ARCACHEMx[3:0]	Output	Read cache type.
ARIDMx[3:0]	Output	Read request ID.
ARLENMx[7:0]	Output	Read burst length. **ARLENMx[7:2]** is always `0b000000`.
ARLOCKMx	Output	Read lock type.
ARPROTMx[2:0]	Output	Read protection type.
ARVALIDMx	Output	Read address valid.
ARSIZEMx[2:0]	Output	Read burst size.
ARQOSMx[3:0]	Output	Read quality of service read identifier.
ARVMIDMx[7:0]	Output	*Virtual Machine IDentifier* (VMID) where uniquely identifiable.

AXIM read address signal integrity protection signals

The following table shows the read address signal integrity protection signals for the AXIM interface.

Table A-23 AXIM read address signal integrity protection signals

Signal	Direction	Description
ARREADYCODEMx	Input	Protection code for **ARREADYMx**.
ARCTL0CODEMx[4:0]	Output	Protection code for read address channel control signals group 0.
ARCTL1CODEMx[4:0]	Output	Protection code for read address channel control signals group 1.
ARADDRCODEMx[6:0]	Output	Protection code for **ARADDRMx[31:0]**.
ARIDCODEMx[4:0]	Output	Protection code for **ARIDMx[3:0]**.
ARVALIDCODEMx	Output	Protection code for **ARVALIDMx**.
ARVMIDCODEMx[4:0]	Output	Protection code for **ARVMIDMx[4:0]**.

AXIM read address interconnect protection signal

The following table shows the read address interconnect protection signal for the AXIM interface.

Table A-24 AXIM read address interconnect protection signal

Signal	Direction	Description
ARUTIDMx[9:0]	Output	Read address unique transaction identifier for interconnect protection.

Read data channel signals

The following table shows the read data channel signals for the AXIM interface.

Table A-25 AXIM read data channel signals

Signal	Direction	Description
RDATAMx[127:0]	Input	Read data.
RIDMx[3:0]	Input	Read data ID.
RLASTMx	Input	Read data last transfer indication.
RRESPMx[1:0]	Input	Read data response.
RVALIDMx	Input	Read data valid.
RREADYMx	Output	Read data ready.

AXIM read data signal integrity protection signals

The following table shows the read data signal integrity protection signals for the AXIM interface.

Table A-26 AXIM read data signal integrity protection signals

Signal	Direction	Description
RCTLCODEMx[2:0]	Input	Protection code for read data channel control signals.
RDATACODEMx[15:0]	Input	Protection code for **RDATAMx[127:0]**.
RIDCODEMx[4:0]	Input	Protection code for **RIDMx[4:0]**.
RVALIDCODEMx	Input	Protection code for **RVALIDMx**.
RREADYCODEMx	Output	Protection code for **RREADYMx**.

AXIM read data interconnect protection signals

The following table shows the read data interconnect protection signals for the AXIM interface.

Table A-27 AXIM read data interconnect protection signals

Signal	Direction	Description
REOBIMx	Input	Read data even/odd beat indicator for interconnect protection.
RUTIDMx[9:0]	Input	Read data unique transaction identifier for interconnect protection.

AXIM write address channel signals

The following table shows the write address channel signals for the AXIM interface.

Table A-28 AXIM write address channel signals

Signal	Direction	Description
AWREADYMx	Input	Write address ready.
AWADDRMx[31:0]	Output	Write address.
AWBURSTMx[1:0]	Output	Write burst type.
AWCACHEMx[3:0]	Output	Write cache type.
AWIDMx[2:0]	Output	Write request ID.
AWLENMx[7:0]	Output	Write burst length. **AWLENM[7:2]** is always `0b000000`.
AWLOCKMx	Output	Write lock type.
AWPROTMx[2:0]	Output	Write protection type.
AWQOSMx[3:0]	Output	Quality of service.
AWSIZEMx[2:0]	Output	Write burst size.
AWVALIDMx	Output	Write address valid.
AWVMIDMx[7:0]	Output	Write virtual machine identifier.

AXIM write address signal integrity

The following table shows the write address signal integrity signals for the AXIM interface.

Table A-29 AXIM write address signal integrity protection signals

Signal	Direction	Description
AWREADYCODEMx	Input	Protection code for **AWREADYMx**.
AWADDRCODEMx[6:0]	Output	Protection code for **AWADDRMx[31:0]**.
AWCTL0CODEMx[4:0]	Output	Protection code for write address channel control signals group 0.
AWCTL1CODEMx[4:0]	Output	Protection code for write address channel control signals group 1.
AWIDCODEMx[4:0]	Output	Protection code for **AWIDMx[2:0]**.
AWVALIDCODEMx	Output	Protection code for **AWVALIDMx**.
AWVMIDCODEMx[4:0]	Output	Protection code for **AWVMIDMx[7:0]**.

AXIM write address interconnect protection signal

The following table shows the write address interconnect protection signal for the AXIM interface.

Table A-30 AXIM write address interconnect protection signal

Signal	Direction	Description
AWUTIDMx[9:0]	Output	Write address unique transaction identifier for interconnect protection.

AXIM write data channel signals

The following table shows the write data channel signals for the AXIM interface.

Table A-31 AXIM write data channel signals

Signal	Direction	Description
WREADYMx	Input	Write data ready.
WDATAMx[127:0]	Output	Write data.
WSTRBMx[15:0]	Output	Write byte-lane strobes.
WLASTMx	Output	Indicates the last transfer in a write burst.
WVALIDMx	Output	Write data valid.

AXIM write data signal integrity protection signals

The following table shows the write data signal integrity protection signals for the AXIM interface.

Table A-32 AXIM write data signal integrity protection signals

Signal	Direction	Description
WREADYCODEMx	Input	Protection code for **WREADYMx**.
WCTLCODEMx[4:0]	Output	Protection code for write data channel control signals.
WDATACODEMx[15:0]	Output	Protection code for **WDATAMx[127:0]**.
WVALIDCODEMx	Output	Protection code for **WVALIDMx**.

AXIM write data interconnect protection signals

The following table shows the write data interconnect protection signals for the AXIM interface.

Table A-33 AXIM write data interconnect protection signals

Signal	Direction	Description
WEOBIMx	Output	Write data even/odd beat indicator for interconnect protection.
WUTIDMx[9:0]	Output	Write data unique transaction identifier for interconnect protection.

AXIM write response channel signals

The following table shows the write response channel signals for the AXIM interface.

Table A-34 AXIM write response channel signals

Signal	Direction	Description
BIDMx[2:0]	Input	Write response ID.
BRESPMx[1:0]	Input	Write response.

Table A-34 AXIM write response channel signals (continued)

Signal	Direction	Description
BVALIDMx	Input	Write response valid.
BREADYMx	Output	Write response ready.

AXIM write response signal integrity protection signals

The following table shows the write response signal integrity protection signals for the AXIM interface.

Table A-35 AXIM write response signal integrity protection signals

Signal	Direction	Description
BCTLCODEMx[2:0]	Input	Protection code for **BRESPMx[1:0]**.
BIDCODEMx[4:0]	Input	Protection code for **BIDMx[2:0]**.
BVALIDCODEMx	Input	Protection code for **BVALIDMx**.
BREADYCODEMx	Output	Protection code for **BREADYMx**.

AXIM write response interconnect protection signal

The following table shows the write response interconnect protection signal for the AXIM interface.

Table A-36 AXIM write response interconnect protection signal

Signal	Direction	Description
BUTIDMx[9:0]	Input	Write response unique transaction identifier for interconnect protection.

A.8.2 AXIS interface signals

This section describes the AXIS interface signals.

AXIS clock enable signal

The following table shows the clock enable signal for the AXIS interface.

Table A-37 AXIS clock enable signal

Signal	Direction	Description
ACLKENS	Input	Clock enable for the AXIS interface. This signal gets registered once in the processor before being used as the clock enable for the interface.

AXIS read address channel signals

The following table shows the read address channel signals for the AXIS interface.

Table A-38 AXIS read address channel signals

Signal	Direction	Description
ARADDRS[31:0]	Input	Read address.
ARIDS[AXIS_ID_WIDTH-1:0]	Input	Read address ID.
ARLENS[7:0]	Input	Instruction fetch burst length.

Table A-38 AXIS read address channel signals (continued)

Signal	Direction	Description
ARPROTS[2:0]	Input	Protection information, privileged/normal access.
ARVALIDS	Input	Read address valid.
ARREADYS	Output	Read address ready.

AXIS read address signal integrity protection signals

The following table shows the read address signal integrity protection signals for the AXIS interface.

Table A-39 AXIS read address signal integrity protection signals

Signal	Direction	Description
ARADDRCODES[6:0]	Input	Protection code for **ARADDRS[31:0]**.
ARCTL0CODES[4:0]	Input	Protection code for read address channel control signals group 0.
ARCTL1CODES[4:0]	Input	Protection code for read address channel control signals group 1.
ARIDCODES[4:0]	Input	Protection code for **ARIDS[AXIS_ID_WIDTH-1:0]**.
ARVALIDCODES	Input	Protection code for **ARVALIDS**.
ARREADYCODES	Output	Protection code for **ARREADYS**.

AXIS read address interconnect protection signal

The following table shows the read address interconnect protection signal for the AXIS interface.

Table A-40 AXIS read address interconnect protection signal

Signal	Direction	Description
ARUTIDS[9:0]	Input	Read address unique transaction identifier for interconnect protection.

AXIS read data channel signals

The following table shows the read data channel signals for the AXIS interface.

Table A-41 AXIS read data channel signals

Signal	Direction	Description
RREADYS	Input	Read data ready.
RDATAS[127:0]	Output	Read data.
RIDS[AXIS_ID_WIDTH-1:0]	Output	Read data ID.
RLASTS	Output	Indicates the last transfer in a read burst.
RRESPS[1:0]	Output	Read response.
RVALIDS	Output	Read data valid.

AXIS read data signal integrity protection signals

The following table shows the read data signal integrity protection signals for the AXIS interface.

Table A-42 AXIS read data signal integrity protection signals

Signal	Direction	Description
RREADYCODES	Input	Protection code for **RREADYS**.
RCTLCODES[2:0]	Output	Protection code for read data channel control signals.
RDATACODES[15:0]	Output	Protection code for **RDATAS[127:0]**.
RIDCODES[4:0]	Output	Protection code for **RIDS[AXIS_ID_WIDTH-1:0]**.
RVALIDCODES	Output	Protection code for **RVALIDS**.

AXIS read data interconnect protection signals

The following table shows the read data interconnect protection signals for the AXIS interface.

Table A-43 AXIS read data interconnect protection signals

Signal	Direction	Description
REOBIS	Output	Read data even/odd beat indicator.
RUTIDS[9:0]	Output	Read data unique transaction identifier for interconnect protection.

AXIS write address channel signals

The following table shows the write address channel signals for the AXIS interface.

Table A-44 AXIS write address channel signals

Signal	Direction	Description
AWADDRS[31:0]	Input	Write transfer start address.
AWIDS[AXIS_ID_WIDTH-1:0]	Input	Write address ID.
AWLENS[7:0]	Input	Write transfer burst length.
AWPROTS[2:0]	Input	Protection information, privileged/normal access.
AWVALIDS	Input	Write address valid.
AWREADYS	Output	Write address ready.

AXIS write address signal integrity protection signals

The following table shows the write address signal integrity protection signals for the AXIS interface.

Table A-45 AXIS write address signal integrity protection signals

Signal	Direction	Description
AWADDRCODES[6:0]	Input	Protection code for **AWADDRS[31:0]**.
AWCTL0CODES[4:0]	Input	Protection code for write address channel control signals group 0.
AWCTL1CODES[4:0]	Input	Protection code for write address channel control signals group 1.
AWIDCODES[4:0]	Input	Protection code for **AWIDS[AXIS_ID_WIDTH-1:0]**.
AWVALIDCODES	Input	Protection code for **AWVALIDS**.
AWREADYCODES	Output	Protection code for **AWREADYS**.

AXIS write address interconnect protection signals

The following table shows the write address interconnect protection signal for the AXIS interface.

Table A-46 AXIS write address interconnect protection signals

Signal	Direction	Description
AWUTIDS[9:0]	Input	Write address unique transaction identifier for interconnect protection.

AXIS write data channel signals

The following table shows the write data channel signals for the AXIS interface.

Table A-47 AXIS write data channel signals

Signal	Direction	Description
WDATAS[127:0]	Input	Write data.
WLASTS	Input	Indicates the last data transfer of a burst.
WSTRBS[15:0]	Input	Write byte-lane strobes.
WVALIDS	Input	Write data valid.
WREADYS	Output	Write data ready.

AXIS write data signal integrity protection signals

The following table shows the write data signal integrity protection signals for the AXIS interface.

Table A-48 AXIS write data signal integrity protection signals

Signal	Direction	Description
WCTLCODES[4:0]	Input	Protection code for write data channel control signals.
WDATACODES[15:0]	Input	Protection code for **WDATAS[127:0]**.
WVALIDCODES	Input	Protection code for **WVALIDS**.
WREADYCODES	Output	Protection code for **WREADYS**.

AXIS write data interconnect protection signals

The following table shows the write data interconnect protection signals for the AXIS interface.

Table A-49 AXIS write data interconnect protection signals

Signal	Direction	Description
WEOBIS	Input	Write data even/odd beat indicator.
WUTIDS[9:0]	Input	Write data unique transaction identifier for interconnect protection.

AXIS write response channel signals

The following table shows the write response channel signals for the AXIS interface.

Table A-50 AXIS write response channel signals

Signal	Direction	Description
BREADYS	Input	Write response ready.
BIDS[AXIS_ID_WIDTH-1:0]	Output	Write response ID.
BRESPS[1:0]	Output	Write response.
BVALIDS	Output	Write response valid.

AXIS write response signal integrity protection signals

The following table shows the write response signal integrity protection signals for the AXIS interface.

Table A-51 AXIS write response signal integrity protection signals

Signal	Direction	Description
BREADYCODES	Input	Protection code for **BREADYS**.
BCTLCODES[2:0]	Output	Protection code for **BRESPS[1:0]**.
BIDCODES[4:0]	Output	Protection code for **BIDS[AXIS_ID_WIDTH-1:0]**.
BVALIDCODES	Output	Protection code for **BVALIDS**.

AXIS write response interconnect protection signal

The following table shows the write response interconnect protection signal for the AXIS interface.

Table A-52 AXIS write response interconnect protection signals

Signal	Direction	Description
BUTIDS[9:0]	Output	Write response unique transaction identifier for interconnect protection.

A.8.3 LLPP interface signals

This section describes the LLPP interface signals.

LLPP clock enable signal

The following table shows the clock enable signal for the LLPP interface.

Table A-53 LLPP clock enable signal

Signal	Direction	Description
ACLKENPx	Input	Clock enable for the LLPP. This signal gets registered once in the processor before being used as the clock enable for the interface.

LLPP read address channel signals

The following table shows the read address channel signals for the LLPP interface.

Table A-54 LLPP read address channel signals

Signal	Direction	Description
ARREADYPx	Input	Read address ready.
ARADDRPx[31:0]	Output	Read address.

Table A-54 LLPP read address channel signals (continued)

Signal	Direction	Description
ARBURSTPx[1:0]	Output	Read burst type.
ARCACHEPx[3:0]	Output	Read cache type.
ARIDPx	Output	Read address ID.
ARLENPx[7:0]	Output	Read burst length.
ARLOCKPx	Output	Read lock type.
ARSIZEPx[2:0]	Output	Read burst size.
ARVALIDPx	Output	Read address valid.
ARPROTPx[2:0]	Output	Read protection type.
ARQOSPx[3:0]	Output	Read quality of service read identifier.
ARVMIDPx[7:0]	Output	VMID where uniquely identifiable. For example, for Device memory transfers, the VMID is indicated. When a transfer does not have a uniquely identifiable VMID, the signal is driven as `0x00`.

LLPP read address signal integrity protection signals

The following table shows the read address signal integrity protection signals for the LLPP interface.

Table A-55 LLPP read address signal integrity protection signals

Signal	Direction	Description
ARREADYCODEPx	Input	Protection code for **ARREADYPx**
ARADDRCODEPx[6:0]	Output	Protection code for **ARADDRPx[31:0]**
ARCTL0CODEPx[4:0]	Output	Protection code for read address channel control signals group 0.
ARCTL1CODEPx[4:0]	Output	Protection code for read address channel control signals group 1.
ARIDCODEPx[4:0]	Output	Protection code for **ARIDPx**.
ARVALIDCODEPx	Output	Protection code for **ARVALIDPx**.
ARVMIDCODEPx[4:0]	Output	Protection code for **ARVMIDPx[7:0]**.

LLPP read address interconnect protection signals

The following table shows read address interconnect protection signals for the LLPP interface.

Table A-56 LLPP read address interconnect protection signals

Signal	Direction	Description
ARUTIDPx[9:0]	Output	Read address unique transaction identifier for interconnect protection.

LLPP read data channel signals

The following table shows the read data channel signals for the LLPP interface.

Table A-57 LLPP read address channel signals

Signal	Direction	Description
RREADYPx	Output	Read data ready.
RLASTPx	Input	Read data last transfer indication.
RRESPPx[1:0]	Input	Read response.
RVALIDPx	Input	Read data valid.
RDATAPx[31:0]	Input	Read data.
RIDPx	Input	Read data ID.

LLPP read data signal integrity protection signals

The following table shows the read data signal integrity protection channel signals for the LLPP interface.

Table A-58 LLPP read data signal integrity protection signals

Signal	Direction	Description
RVALIDCODEPx	Input	Protection code for **RVALIDPx**.
RDATACODEPx[6:0]	Input	Protection code for **RDATAPx[31:0]**.
RIDCODEPx[4:0]	Input	Protection code for **RIDPx**.
RCTLCODEPx[2:0]	Input	Protection code for read data channel control signals.
RREADYCODEPx	Output	Protection code for **RREADYPx**.

LLPP read data interconnect protection signals

The following table shows the read data interconnect protection signals for the LLPP interface.

Table A-59 LLPP read data interconnect protection signals

Signal	Direction	Description
RUTIDPx[9:0]	Input	Read data unique transaction identifier for interconnect protection.
REOBIPx	Input	Read data even/odd beat indicator for interconnect protection.

LLPP write address channel signals

The following table shows the write address channel signals for the LLPP interface.

Table A-60 LLPP write address channel signals

Signal	Direction	Description
AWBURSTPx[1:0]	Output	Write burst type.
AWCACHEPx[3:0]	Output	Write cache type.
AWIDPx	Output	Write address ID.
AWLENPx[7:0]	Output	Write burst length.
AWLOCKPx	Output	Write lock type.
AWPROTPx[2:0]	Output	Write protection type.

Table A-60 LLPP write address channel signals (continued)

Signal	Direction	Description
AWVALIDPx	Output	Write address valid.
AWSIZEPx[2:0]	Output	Write burst size.
AWREADYPx	Input	Write address ready.
AWADDRPx[31:0]	Output	Write address.
AWQOSPx[3:0]	Output	Quality of service.
AWVMIDPx[7:0]	Output	VMID where uniquely identifiable. For example, for Device memory transfers, the VMID is indicated. When a transfer does not have a uniquely identifiable VMID, the signal is driven as `0x00`.

LLPP write address signal integrity protection signals

The following table shows the write address signal integrity protection signals for the LLPP interface.

Table A-61 LLPP write address signal integrity protection signals

Signal	Direction	Description
AWCTL0CODEPx[4:0]	Output	Protection code for write address channel control signals group 0.
AWCTL1CODEPx[4:0]	Output	Protection code for write address channel control signals group 1.
AWIDCODEPx[4:0]	Output	Protection code for **AWIDPx**.
AWVALIDCODEPx	Output	Protection code for **AWVALIDPx**
AWADDRCODEPx[6:0]	Input	Protection code for **AWADDRPx[31:0]**.
AWREADYCODEPx	Input	Protection code for **AWREADYPx**.
AWVMIDCODEPx	Output	Protection code for **AWVMIDPx[7:0]**.

LLPP write address interconnect protection signal

The following table shows the write address interconnect protection signal for the LLPP interface.

Table A-62 LLPP write address interconnect protection signal

Signal	Direction	Description
AWUTIDPx[9:0]	Output	Write address unique transaction identifier for interconnect protection.

LLPP write data channels signals

The following table shows the write data channel signals for the LLPP interface.

Table A-63 LLPP write data channel signals

Signal	Direction	Description
WREADYPx	Input	Write data ready.
WVALIDPx	Output	Write data valid.
WDATAPx[31:0]	Output	Write data.

Table A-63 LLPP write data channel signals (continued)

Signal	Direction	Description
WLASTPx	Output	Write data last transfer indication.
WSTRBPx[3:0]	Output	Write byte-lane strobes.

LLPP write data signal integrity protection signals

The following table shows the write data signal integrity protection signals for the LLPP interface.

Table A-64 LLPP write data signal integrity protection signals

Signal	Direction	Description
WREADYCODEPx	Input	Protection code for **WREADYPx**.
WVALIDCODEPx	Output	Protection code for **WVALIDPx**.
WDATACODEPx[6:0]	Output	Protection code for **WDATAPx[31:0]**.
WCTLCODEPx[4:0]	Output	Protection code for write data channel control signals.

LLPP write data interconnect protection signals

The following table shows the write data interconnect protection signals for the LLPP interface.

Table A-65 LLPP write data interconnect protection signals

Signal	Direction	Description
WEOBIPx	Output	Write data even/odd beat indicator for interconnect protection.
WUTIDPx[9:0]	Output	Write data unique transaction identifier for interconnect protection.

LLPP write response channel signals

The following table shows the write response channel signals for the LLPP interface.

Table A-66 LLPP write response channel signals

Signal	Direction	Description
BIDPx	Input	Write response ID.
BREADYPx	Output	Write response ready.
BRESPPx[1:0]	Input	Write response.
BVALIDPx	Input	Write response valid.

LLPP write response signal integrity protection signals

The following table shows the write response signal integrity protection signals for the LLPP interface.

Table A-67 LLPP write response signal integrity protection signals

Signal	Direction	Description
BIDCODEPx[4:0]	Input	Protection code for **BIDPx**.
BREADYCODEPx	Output	Protection code for **BREADYPx**.

Table A-67 LLPP write response signal integrity protection signals (continued)

Signal	Direction	Description
BVALIDCODEPx	Input	Protection code for **BVALIDPx**.
BCTLCODEPx[2:0]	Input	Protection code for write response channel control signals.

LLPP write response interconnect protection signal

The following table shows the write response interconnect protection signal for the LLPP interface.

Table A-68 LLPP write response interconnect protection signal

Signal	Direction	Description
BUTIDPx[9:0]	Input	Write response unique transaction identifier for interconnect protection.

A.8.4 Flash interface signals

This section describes the Flash interface signals.

Flash clock enable signal

The following table shows the clock enable signal for the Flash interface.

Table A-69 Flash clock enable signal

Signal	Direction	Description
ACLKENFx	Input	Clock enable for the Flash interface. This signal gets registered once in the processor before being used as the clock enable for the interface.

Flash read address channel signals

The following table shows the read address channel signals for the Flash interface.

Table A-70 Flash read address channel signals

Signal	Direction	Description
ARREADYFx	Input	Read address ready.
ARLENFx[7:0]	Output	Read burst length.
ARIDFx	Output	Read address ID.
ARADDRFx[31:0]	Output	Read address.
ARPROTFx[2:0]	Output	Read protection type.
ARVALIDFx	Output	Read address valid.
ARBURSTFx[1:0]	Output	Read burst type.

Flash read address signal integrity protection signals

The following table shows the read address signal integrity protection signals for the Flash interface.

Table A-71 Flash read address signal integrity protection signals

Signal	Direction	Description
ARREADYCODEFx	Input	Protection code for **ARREADYFx**.
ARIDCODEFx[4:0]	Output	Protection code for **ARIDFx**.
ARADDRCODEFx[6:0]	Output	Protection code for **ARADDRFx[31:0]**.
ARCTL0CODEFx[4:0]	Output	Protection code for read address channel control signals group 0.
ARCTL1CODEFx[4:0]	Output	Protection code for read address channel control signals group 1.
ARVALIDCODEFx	Output	Protection code for **ARVALIDFx**.

Flash read address interconnect protection signals

The following table shows the read address interconnect protection signals for the Flash interface.

Table A-72 Flash read address interconnect protection signals

Signal	Direction	Description
ARUTIDFx[7:0]	Output	Read address unique transaction identifier for interconnect protection.

Flash read data channel signals

The following table shows the read data channel signals for the Flash interface.

Table A-73 Flash read data channel signals

Signal	Direction	Description
LATEERRFx	Input	Late data error. **LATEERRFx** is valid exactly one **CLKIN** cycle after a valid/ready handshake completes on the R channel because **RVALIDFx** and **RREADYFx** are both HIGH on a gated (registered **ACLKENFx** clock enable) clock positive edge.
RIDFx	Input	Read data ID.
RDATAFx[127:0]	Input	Read data.
RRESPFx[1:0]	Input	Read data response.
RVALIDFx	Input	Read data valid.
RLASTFx	Input	Indicates the last transfer in a read burst.
RREADYFx	Output	Read data ready.

Flash read data signal integrity protection signals

The following table shows the read data signal integrity protection signals for the Flash interface.

Table A-74 Flash read data signal integrity protection signals

Signal	Direction	Description
LATEERRCODEFx	Input	Protection code for **LATEERRFx**.
RCTLCODEFx[2:0]	Input	Protection code for read data channel control signals.

Table A-74 Flash read data signal integrity protection signals (continued)

Signal	Direction	Description
RDATACODEFx[8:0] **Note** The width of **RDATACODEFx** is configurable based on the value of the global configuration parameter FLASH_DATA_ECC_SCHEME. The possible values are: 1 **RDATACODEFx[8:0]** 2 **RDATACODEFx[15:0]**	Input	Protection code for **RDATAFx[127:0]**.
RIDCODEFx[4:0]	Input	Protection code for **RIDFx**.
RVALIDCODEFx	Input	Protection code for **RVALIDFx**.
RREADYCODEFx	Output	Protection code for **RREADYFx**.

Flash read data interconnect protection signals

The following table shows the read data interconnect protection signals for the Flash interface.

Table A-75 Flash read data interconnect protection signals

Signal	Direction	Description
REOBIFx	Input	Read data even/odd beat indicator for interconnect protection.
RUTIDFx[7:0]	Input	Read data unique transaction identifier for interconnect protection.

A.9 Debug and trace interface signals

This section describes the debug and trace interface signals.

This section contains the following subsections:

A.9.1 Debug interface signals

This section describes the debug interface signals.

The debug signals are described in:

Debug APB interface signals

The following table shows the debug APB interface signals.

Table A-76 Debug APB interface signals

Signal	Direction	Description
PCLKENDBG	Input	APB clock enable.
PADDRDBG31	Input	APB address bus bit[31]: **0** Not an external debugger access. **1** External debugger access.
PADDRDBG[21:2]	Input	APB address bus bits[21:2].
PENABLEDBG	Input	Indicates the second and subsequent cycles of an APB transfer.
PSELDBG	Input	Debug registers select: **0** Debug registers not selected. **1** Debug registers selected.
PWDATADBG[31:0]	Input	APB write data bus.
PWRITEDBG	Input	APB read or write signal: **0** Reads from APB. **1** Writes to APB.
nPRESETDBG	Input	APB reset: **0** Reset APB and top-level debug logic. **1** Do not reset APB and top-level debug logic.
PRDATADBG[31:0]	Output	APB read data bus.

Table A-76 Debug APB interface signals (continued)

Signal	Direction	Description
PREADYDBG	Output	APB slave ready. The processor can assert **PREADYDBG** to extend a transfer by inserting wait states.
PSLVERRDBG	Output	APB slave transfer error: **0** No transfer error. **1** Transfer error.

Authentication interface signals

The following table shows the authentication interface signals.

Table A-77 Authentication interface signals

Signal	Direction	Description
DBGENx	Input	Invasive debug enable.
NIDENx	Input	Non-invasive debug enable.
HIDENx	Input	Authentication hypervisor invasive debug enable.
HNIDENx	Input	Authentication hypervisor non-invasive debug enable.

Miscellaneous debug signals

The following table shows the miscellaneous debug signals.

Table A-78 Miscellaneous debug signals

Signal	Direction	Description
EDBGRQx	Input	External debug request.
COMMRXx	Output	*Debug Communications Channel* (DCC) data transfer register receive is full.
COMMTXx	Output	DCC data transfer register transmit is empty.
DBGACKx	Output	Debug request acknowledge.
DBGRSTREQx	Output	Request for reset from external debug logic.

A.9.2 ETM interface signals

This section describes the ETM interface signals.

The interface signals are described in:

- *ATB data trace interface signals* on page Appx-A-578.
- *ATB instruction trace interface signals* on page Appx-A-577.
- *Miscellaneous ETM interface signals* on page Appx-A-578.

ATB instruction trace interface signals

The following table shows the ATB instruction side interface signals.

Table A-79 ATB instruction trace interface signals

Signal	Direction	Description
ATCLKENI	Input	ATB clock enable for instruction trace and **TSVALUEB**.
AFVALIDIx	Input	FIFO flush request.
ATREADYIx	Input	ATB device ready.
SYNCREQIx	Input	Synchronization request. The input must be driven HIGH for one ATB bus cycle.
AFREADYIx	Output	FIFO flush finished.
ATBYTESIx[1:0]	Output	Data size.
ATDATAIx[31:0]	Output	Data.
ATIDIx[6:0]	Output	Trace source ID.
ATVALIDIx	Output	Data valid.

ATB data trace interface signals

The following table shows the ATB data side interface signals.

Table A-80 ATB data trace interface signals

Signal	Direction	Description
ATCLKEND	Input	ATB clock enable for data trace.
AFVALIDDx	Input	FIFO flush request.
ATREADYDx	Input	ATB device ready.
SYNCREQDx	Input	Synchronization request. The input must be driven HIGH for one ATB bus cycle.
AFREADYDx	Output	FIFO flush finished.
ATBYTESDx[2:0]	Output	Data size.
ATDATADx[63:0]	Output	Data.
ATIDDx[6:0]	Output	Trace source ID.
ATVALIDDx	Output	Data valid.

Miscellaneous ETM interface signals

The following table shows the miscellaneous ETM interface signals.

Table A-81 Miscellaneous ETM interface signals

Signal	Direction	Description
TSVALUEB[63:0]	Input	Time stamp value in binary encoding.
ETMEXTIN[3:0]	Input	Used to trigger events in the trace unit.
ETMEXTOUT[3:0]	Output	Used to indicate events to a trace analyzer.

Note

In order to use data trace, an active timestamp generator is required.

A.9.3 Cross trigger interface signals

The following table shows the cross trigger interface signals.

Note

You must balance the CTI and CTM interface signals with respect to **CLKIN**.

Table A-82 CTI and CTM interface signals

Signal	Direction	Description
CTMCHIN[3:0]	Input	CTM input channel interface.
CTMCHINACK[3:0]	Output	CTM input channel interface acknowledge.
CTMCHOUT[3:0]	Output	CTM output channel.
CTMCHOUTACK[3:0]	Input	CTM output channel acknowledge.
CTMCIHSBYPASS[3:0]	Input	CTM channel interfaces handshake bypass for each channel.
CTMCISBYPASS	Input	CTM channel interfaces synchronization bypass. Same for all channels.

A.10 Generic timer signals

The following table shows the generic timer signals.

Table A-83 Generic timer signals

Signal	Direction	Description
CNTCLKEN	Input	System counter clock enable.
CNTVALUEB[63:0]	Input	Global system counter value in binary format.

A.11 Power management signals

The following table shows the power management signals.

Table A-84 Power management signals

Signal	Direction	Description
CLREXMONREQ	Input	Clearing of the external global exclusive monitor request. When asserted, this signal acts as a WFE wake-up event to all cores.
CLREXMONACK	Output	Clearing of the external global exclusive monitor acknowledge.
EVENTI	Input	Event input for processor wake-up from WFE low-power state.
EVENTO	Output	Event output. Active when a `SEV` instruction is executed.
STANDBYWFIx	Output	Core is in WFI low-power state.
STANDBYWFEx	Output	Core is in WFE low-power state.

Complex power scenarios with multiple transitions are managed by the P-channel interface. See, *Low Power Interface Specification ARM® Q-Channel and P-Channel Interfaces* for more information on the P-channel interface.

The following table shows the P-channel signals.

Table A-85 P-channel signals

Signal	Direction	Description
COREPSTATEx	Input	P-channel state.
nCOREPREQx	Input	P-channel request.
COREPACTIVEx[1:0]	Output	P-channel active.
COREPDENYx	Output	P-channel deny request.
COREPACCEPTx	Output	P-channel accept request.

COREPSTATEx encodes the power state that the external power controller is requesting that the core enters when it asserts **COREPREQx**. There are two power states for each core, and the following table shows the encoding.

Table A-86 Power states for PE

Bit	Function
`0`	RUN
`1`	SHUTDOWN

When Cortex-R52 is initially switched on or fully resets, it is in RUN state. Arm recommends that **nCOREPREQx** is asserted while the core is in reset with **COREPSTATEx**set to RUN and after reset the power controller must wait until this transition is completed before making a request for SHUTDOWN state. If this method is not used, then the power controller must wait at least eight cycles. [ch] **CLKIN** cycles after reset deassertion before making a request for SHUTDOWN state. When a core enters SHUTDOWN, it sets the debug interface and AXIS interface for the core to respond to any

[ch] Four cycles while reset is held internally after the external deassertion, two additional cycles while the cores are held in reset, and finally, two cycles for any initialization registers to settle.

requests with an error and then asserts **COREPACCEPTx**. When **COREPACCEPTx** is deasserted, the core is in SHUTDOWN and the external power control can remove power from that part of the design.

COREPACTIVEx[1:0] indicates if the core has any activity within it. The following table shows the encoding.

Table A-87 Signals in COREPACTIVEx[1:0]

Bit	Function
[0]	Indicates if the debug systems require power. This bit is equivalent to the **DBGNOPWRDN** and **DBGPWRUPRQ** output on previous Arm processors.
[1]	In RUN state, indicates if there are any instructions being processed in the core or if there is any AXIS access through the core in progress. In SHUTDOWN state, indicates if there is a pending interrupt or debug powerup request.

COREPACTIVEx[1:0] must be used by the power controller as an indication that the core has entered WFI low-power state or WFE low-power state with the GIC ProcessorSleep for the core set, whether there is no AXIS or debug request in progress. When **COREPACTIVEx[1:0]** is LOW, a P-channel transition to SHUTDOWN state is normally accepted.

COREPACTIVEx[1:0] must be used by the power controller as an indication that any powerdown must be an emulated powerdown.

A.12 DFT and on-line MBIST signals

This section describes the DFT and MBIST signals.

This section contains the following subsections:

A.12.1 DFT signals

The following table shows the DFT interface signals.

Table A-88 DFT signals

Signal	Direction	Description
DFTCGEN	Input	Clock gate override control signal. When this signal is HIGH, the clock enables of the architectural clock gates that control the internal clocks are all forced HIGH so that all internal clocks always run.
DFTRAMHOLD	Input	Disables the chip selects of all the RAMs instantiated within the processor during scan shift.
DFTRSTDISABLE[1:0]	Input	Disables internal synchronized reset during scan shift.
DFTMCPHOLD	Input	Disable multicycle paths on RAM interfaces.

A.12.2 On-line MBIST signals

On-line MBIST signals function in the same way as production MBIST signals.

The following table shows the on-line MBIST interface signals.

Table A-89 On-line MBIST interface signals

Signal	Direction	Description
MBISTADDREXT[16:0]	Input	Address for array under test.
MBISTARRAYEXT[4:0]	Input	Select array under test.
MBISTCFGEXT	Input	Sets ALL mode.
MBISTINDATAEXT[77:0]	Input	Write data for array under test.
MBISTREADENEXT	Input	Read enable.
MBISTWRITEENEXT	Input	Write enable.
nMBISTRESET	Input	MBIST reset. **0** Reset MBIST. **1** Do not reset MBIST.
MBISTACKEXT	Output	MBIST acknowledge.
MBISTREQEXT	Input	MBIST mode request when using the external MBIST interface.
MBISTREQINT	Input	MBIST mode request when using the internal MBIST interface for production MBIST. Unused for on-line MBIST operation.
MBISTERREXT	Output	MBIST access has failed. For example, on-line MBIST access to a core that has powered down. This signal is only used for on-line MBIST and not for production MBIST.
MBISTOUTDATAEXT[77:0]	Output	Read data for array under test.

A.13 GIC Distributor external messaging port signals

The interrupt external messaging port allows an external component, for example a DMA engine, to receive interrupts which are signaled to the Cortex-R52 processor under the control of the Cortex-R52 GDU. The GIC Distributor external messaging port signals the HPPI to the external component and receives messages for interrupt-related activities such as, acknowledging interrupts and ending interrupts.

The following table shows the GIC Distributor external messaging port signals.

Table A-90 GIC Distributor external messaging port signals

Signal	Direction	Description
GDDATAIN[13:0]	Input	Data from external device to GIC Distributor for activate, deactivate, and generate SGI requests.
GDACTIVATEREQ	Input	Activate request from external device.
GDDEACTIVATEREQ	Input	Deactivate request from external device.
GDGENSGIREQ	Input	Generate SGI request from external device.
GDDATAOUT[15:0]	Output	Data from GIC Distributor to external device for HPPI update requests.
GDQUIESCEACK	Input	Quiesce acknowledge from external device for sleep entry.
GDACTIVATEACK	Output	Activate acknowledge from GIC Distributor.
GDDEACTIVATEACK	Output	Deactivate acknowledge from GIC Distributor.
GDGENSGIACK	Output	Generate SGI acknowledge from GIC Distributor.
GDHPPIREQ	Output	HPPI update request from GIC Distributor.
GDQUIESCEREQ	Output	Quiesce request from GIC Distributor for sleep entry.
GDASLEEP	Output	Indicates the interface between GIC Distributor and external device is quiesced and the external device can be powered down. This is the GICR_WAKER.ChildrenAsleep register interface bit.
GDWAKEREQUEST	Output	Indicates there is a pending interrupt for the external device and that it must be powered up.

A.14 Interrupt input signals

The following table shows the interrupt input signals.

Table A-91 Interrupt input signals

Signal	Direction	Description
SPI[NUM_SPIS-1:0]	Input	Shared peripheral interrupts into the GIC Distributor. These interrupts are active-HIGH when configured as level-sensitive, or rising-edge triggered when configured as edge-triggered in the GIC Distributor.
SEIx	Input	Physical system error interrupt into the core. Active HIGH, edge-sensitive.
VSEIx	Input	Virtual system error interrupt into the core. Active HIGH, edge-sensitive.
EXTPPIx[8:0]	Input	External private peripheral interrupts into the GIC Distributor. These interrupts are active-LOW when configured as level-sensitive, or they are rising-edge triggered when configured as edge-triggered in the GIC DIstributor. These are spare PPIs that are made available as extra interrupt inputs where the core does not require them for internal purposes.

A.15 DCLS signals

The following table shows the DCLS signals.

Table A-92 DCLS signals

Signal	Direction	Description
CLKINDCLS	Input	Redundant clock for the redundant logic in lock-step configurations.
nCORERESETDCLSx	Input	Individual redundant core warm reset. **0** Apply reset to the redundant core excluding debug and trace logic. **1** Do not apply reset to the redundant core.
nCPUPORESETDCLSx	Input	Individual redundant core powerup cold reset. **0** Apply reset to the redundant core including debug and trace logic. **1** Do not apply reset to the redundant core.
nTOPRESETDCLSx	Input	Top-level redundant logic reset. Resets all the top-level functional redundant logic in the processor. **0** Reset all the top-level functional redundant logic. **1** Do not reset all the top-level functional redundant logic.
DCLSCOMPIN[99:0]	Input	DCLS comparators control.
DCLSCOMPOUT[99:0]	Output	DCLS comparators status.

A.16 Split/Lock signal

The following table shows the Split/Lock signal.

Table A-93 Split/Lock signal

Signal	Direction	Description
CFGSLSPLIT	Input	Indicates whether Split mode or Lock mode is selected. **0** Logical Lock mode selected. **1** Logical Split mode selected.

Note

If Lock mode is selected, all the DCLS signals must be driven in addition to **CFGSLSPLIT**. If Split mode is selected, only **CLKINDCLS** must be driven in addition to **CFGSLSPLIT**. For more information on DCLS signals, see *A.15 DCLS signals* on page Appx-A-586.

Appendix B
Cycle Timings and Interlock Behavior

This appendix describes the cycle timing and interlock behavior of instructions on the Cortex-R52 processor.

It contains the following sections:

B.1 About cycle timings and interlock behavior

Complex instruction dependencies and memory system interactions make it impossible to exhaustively describe the exact cycle timing behavior for all instructions in all circumstances and all contexts. The timings described in this chapter are accurate in most cases.

If precise timings are required, you must use a cycle-accurate model of the Cortex-R52 processor. Unless stated otherwise, cycle counts described in this chapter are best-case numbers. They assume:

- No outstanding data dependencies between the current instruction and a previous instruction.
- The instruction does not encounter any resource conflicts.
- All data accesses hit in the data cache, do not cross protection region boundaries, and are aligned to the access size.
- All instruction accesses hit in the instruction cache.

This section contains the following subsections:

B.1.1 Pipeline information

All cores in the Cortex-R52 processor have the same pipeline design, which is two-way superscalar for high instruction throughput and good power efficiency. The following figure is a simplified view of the overall pipeline structure.

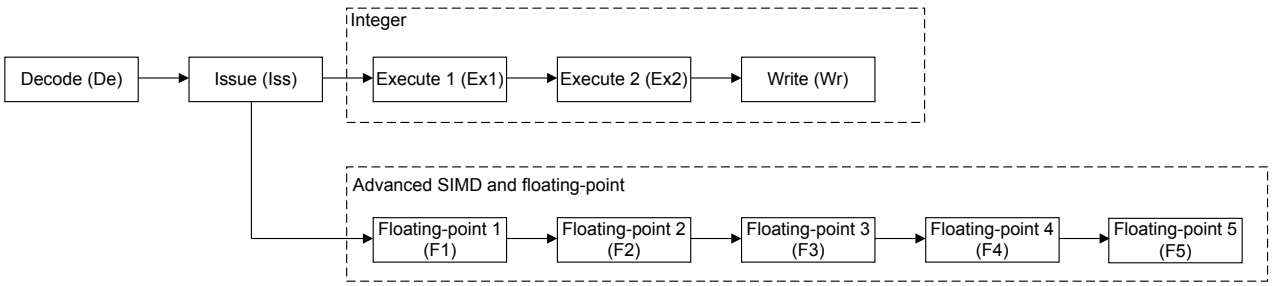

Figure B-1 Cortex-R52 pipeline

B.1.2 Instruction execution overview

The instruction execution pipeline for integer instructions has four stages which are, Iss, Ex1, Ex2, and Wr. The instruction execution pipeline for floating-point and Advanced SIMD instructions has six stages which are, Iss, F1, F2, F3, F4, and F5.

Extensive forwarding among all execution stages enables many dependent instruction sequences to run without pipeline stalls. General forwarding occurs from the Ex1, Ex2, Wr, F3, F4, and F5 pipeline stages. In addition, the multiplier contains an internal multiply accumulate forwarding path. The address generation unit also contains an internal forwarding path for instructions that perform base register update.

Instructions can forward out of a pipeline stage if the result is available and a forwarding path exists. For more information on when results are ready to be forwarded for each instruction, see *Table B-2 Base instructions cycle timings* on page Appx-B-593 and *Table B-3 Floating point and Advanced SIMD instructions cycle timings* on page Appx-B-602.

The tables do not specify by which pipeline stage an instruction needs its input to be ready in order not to interlock, but the following rules generally apply:

- If they do not require the shifter, integer ALU operations require their inputs to be ready by the end of the Ex1 stage. If they require the shifter, integer ALU operations require their inputs to be ready by the end of the Iss stage.
- Integer multiplication and division operations require their inputs by the end of the Iss stage.
- Both integer and floating-point and Advanced SIMD load/store operations require inputs which are used to compute the address by the beginning of the Iss stage, and inputs which are used for storing data to memory by the end of the Ex2 stage.
- Floating-point and Advanced SIMD arithmetic instructions require their inputs by the end of the F1 stage.

For example, the following sequence takes two cycles because when the `ADD/SUB` pair is in Ex2, the `ADDS/BEQ` pair is in Ex1, and the updated value for R1 can be forwarded from Ex2 to Ex1.

```
ADD R1, R2, R3  ; Produces R1 result in Ex2
SUB R4, R5, R6  ; Dual-issued with the ADD, no hazards
ADDS R5, R1, R7 ; Read-After-Write hazard on R1
BEQ loop        ; Dual-issued with the ADDS
```

The following sequence takes four cycles because, when the *Store Register* (`STR`) is in Iss and needs R1 early in Iss to produce the address, the *Load Register* (`LDR`) is in Ex1 and cannot produce a result yet. Therefore, the `STR` stalls for two cycles for the `LDR` to reach the Wr stage and to forward its result to the `STR` that is still in Iss:

```
LDR R1, [R2]  ; Produces R1 result early in Wr
STR R3, [R1]  ; Read-After-Write hazard on R1, interlocks for two cycles
```

The *Program Counter* (PC) is the only register that result latency does not affect. An instruction that alters the PC never causes a pipeline stall because of interlocking with a subsequent instruction that reads the PC. Such instructions do, however, cause the program to branch which can add latency if it was not correctly predicted.

The base register update for load or store instructions occurs in the ALU pipeline. To prevent an interlock for back-to-back load/store instructions reusing the same base register, there is a local forwarding path to recycle the updated base register around the address generator. This only applies when:

- The load/store instruction with base update uses pre-indexed addressing.
- It is a single load/store instruction that is not a load/store double instruction or load/store multiple instruction.

For example, with R2 aligned the following instruction sequence takes three cycles to execute:

```
LDR R5, [R2, #4]!
LDR R6, [R2, #0x10]!  ; Single-issued (structural hazard), but R2 is forwarded
LDR R7, [R2, #0x20]! ; Single-issued (structural hazard), but R2 is forwarded
```

B.1.3 Conditional instructions

If an instruction fails its condition code, in most cases, it takes the same number of cycles to execute as when it passes its condition code.

The exceptions to this condition are:

- Instructions that alter the PC, such as branches.
- Divide instructions which require only one execute cycle when they fail their condition code.
- Floating-point and Advanced SIMD multiply-accumulate instructions which require only one execute cycle when they fail their condition code.

Certain performance optimizations are also not used for conditional instructions. For example, forwarding from the Ex1 stage is not used for conditional instructions.

However, other performance optimizations are possible for conditional instructions. For example, the processor detects when a pair of dependent instructions have opposite condition codes and avoid

unnecessary interlocking between them. Similarly, the processor evaluates the condition code for an instruction early when possible and avoids generating interlocks if it fails.

B.1.4 Flag-setting instructions

If instructions are flag-setting, most of them take the same number of cycles to execute as if they were not flag-setting.

Subsequent instructions that depend on the NZCV flags are executed without any penalty. The exceptions to these rules are:

- Flag-setting multiply instructions.
- Instructions that explicitly consume the C flag for arithmetic operations or RRX shifts.

As an example, the following code sequence takes a single cycle to execute:

```
SUBS r1, #1
BEQ label  ; Dual-issued with the above. Z flag is forwarded appropriately.
```

The following code sequence takes two cycles to execute:

```
SUBS r1, #1
ADC r2, #2  ; Read-After-Write Hazard on the C flag, not dual-issued
```

Similarly, the following code sequence also takes two cycles to execute:

```
MULS r1, #2 ; Produces flags later than usual
BEQ label   ; Not dual-issued because of the flag-setting multiply
```

B.2 Instructions cycle timings

This section describes the cycle timing behavior for all the instructions supported by the Cortex-R52 processor in the A32 and T32 instruction sets.

This section contains the following subsections:

B.2.1 Definition of terms

The following table shows the cycle timing terms used in this section and their description.

Table B-1 Cycle timing terms

Term	Description
Interlock	There is a data dependency between two instructions in the pipeline, resulting in the Iss stage being stalled until the processor resolves the dependency.
Memory Cycles	This is the number of cycles during which an instruction sends a memory access to the cache or TCM. Barriers are not counted as inducing Memory Cycles.
Cycles	This is the minimum number of cycles required to issue an instruction. Issue cycles that produce memory accesses to the cache are included, so Cycles is always greater than or equal to Memory Cycles.
Instruction and Variant	The instruction mnemonic, according to the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*. For example, for the "`ADC`, `ADCS` (immediate)" instruction, the Instruction column is "`ADC`, `ADCS`" and the Variant column is "immediate".
Main Result	The pipeline stage in which the instruction result or results to be written to the register file by this instruction are produced, and can be potentially forwarded to a dependent instruction. This can be more than one value. For example, `UMLAL R0, R1, R2, R3` produces two results (R0 and R1) in Wr. For load/store operations, the Main Result is the value loaded from the memory, or the exclusive status information for store exclusive. Updates to the base register are described separately. For example, `LDR R0, [R1, #4]!` produces only one Main Result (R0) in Wr and the R1 update is described separately. If an instruction can produce its results in various stages, then a range of stages is given hyphenated. For example, `ADD, ADDS (register)` can produce its result either in Ex1 or in Ex2, because it is skewable. A notation of `(+N)` indicates an out-of-order delayed write. Most floating-point and Advanced SIMD instructions only produce results for the floating-point and Advanced SIMD register file. Some instructions, for example, `VMOV`, might also produce results for the integer register file, which is captured in a separate Main Result column.

Table B-1 Cycle timing terms (continued)

Term	Description
Base Register Update	Applicable only for load/store instructions. Indicates the stage in which a load/store operation which updates its base register has produced the new value that can be potentially forwarded to a dependent instruction. For example, `LDR R0, [R1, #4]!` updates its base register (R1) either in Ex1 or in Ex2.
Branch Prediction	Applicable only for instructions that are branches, for example, `BEQ` label, or might behave like branches if they write the PC, for example, `ADD PC, R1, R2`. It can take three values: **Predicted** The processor predicts the instruction. No performance penalty occurs on a correct prediction. On a misprediction, the pipeline is flushed. The penalty is equivalent to increasing the Cycles number by either seven for direct branches or eight for indirect branches. **Not Predicted** The processor is always making a not-take prediction. If the instruction is unconditional or if it passes its condition code, the pipeline is flushed, add eight to the Cycles. If the instruction fails its condition code, no penalty occurs. **Stop** The processor does not predict the instruction but pre-decodes it. Instructions are always unconditional, therefore, power is saved because no fetches are done and there can be second-order effect performance gains because of less traffic in the memory system. The performance penalty is similar to the Not Predicted case, therefore, add eight to the Cycles.

Related references

B.3.1 Skewing on page Appx-B-609
B.3.4 Division and square root on page Appx-B-610

B.2.2 Base instructions cycle timings

This section describes the cycle timing behavior of the base instructions in the A32 and T32 instruction sets, excluding floating-point and Advanced SIMD instructions.

Table B-2 Base instructions cycle timings

Instruction	Variant	Cycles	Memory cycles	Main result	Base register update	Branch prediction
ADC, ADCS	Immediate	1	0	Ex2	-	Not predicted
ADC, ADCS	Register	1	0	Ex2	-	Not predicted
ADC, ADCS	Register-shifted register	1	0	Ex2	-	-
ADD, ADDS	Immediate	1	0	Ex1-Ex2	-	Stop
ADD, ADDS	Register	1	0	Ex1-Ex2	-	Stop
ADD, ADDS	Register-shifted register	1	0	Ex2	-	-
ADR		1	0	Ex1	-	Stop
AND, ANDS	Immediate	1	0	Ex1-Ex2	-	Not predicted
AND, ANDS	Register	1	0	Ex1-Ex2	-	Not predicted
AND, ANDS	Register-shifted register	1	0	Ex2	-	-
ASR, ASRS	Immediate	1	0	Ex1	-	Not predicted
ASR, ASRS	Register	1	0	Ex1	-	-

Table B-2 Base instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	Main result	Base register update	Branch prediction
B		1	0	-	-	Predicted
BFC		1	0	Ex2	-	-
BFI		1	0	Ex2	-	-
BIC, BICS	Immediate	1	0	Ex1-Ex2	-	Not predicted
BIC, BICS	Register	1	0	Ex1-Ex2	-	Not predicted
BIC, BICS	Register-shifted register	1	0	Ex2	-	-
BKPT		1	0	-	-	-
BL	Immediate	1	0	Ex2	-	Predicted
BLX	Register	1	0	Ex2	-	Predicted
BX		1	0	-	-	Predicted
BXJ		1	0	-	-	Not predicted
CBZ, CBNZ		1	0	-	-	Predicted
CLREX		1	0	-	-	-
CLZ		1	0	Ex2	-	-
CMN	Immediate	1	0	-	-	-
CMN	Register	1	0	-	-	-
CMN	Register-shifted register	1	0	-	-	-
CMP	Immediate	1	0	-	-	-
CMP	Register	1	0	-	-	-
CMP	Register-shifted register	1	0	-	-	-
CPS, CPSIE, CPSID		1	0	-	-	-
CRC32		1	0	Ex2	-	-
CRC32C		1	0	Ex2	-	-
DBG		1	0	-	-	-
DMB		1	0	-	-	-
DSB		1	0	-	-	-
EOR, EORS	Immediate	1	0	Ex1-Ex2	-	Not predicted
EOR, EORS	Register	1	0	Ex1-Ex2	-	Not predicted
EOR, EORS	Register-shifted register	1	0	Ex2	-	-
ERET		1	0	-	-	Stop
HLT		1	0	-	-	-

Table B-2 Base instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	Main result	Base register update	Branch prediction
HVC		1	0	-	-	-
ISB		1	0	-	-	Stop
IT		1	0	-	-	-
LDA		1	1	Wr	-	-
LDAB		1	1	Wr	-	-
LDAEX		1	1	Wr	-	-
LDAEXB		1	1	Wr	-	-
LDAEXD		1	1	Wr	-	-
LDAEXH		1	1	Wr	-	-
LDAH		1	1	Wr	-	-
LDC	Immediate	1	1	-	Ex2	-
LDM	Exception return	1-8	1-8	Wr	Ex2	Stop
LDM, LDMIA, LDMFD		1-8	1-8	Wr	Ex2	Predicted
LDM	User registers	1-8	1-8	Wr	-	-
LDMDA, LDMFA		1-8	1-8	Wr	Ex2	Predicted
LDMDB, LDMEA		1-8	1-8	Wr	Ex2	Predicted
LDMIB, LDMED		1-8	1-8	Wr	Ex2	Predicted
LDR	Immediate	1	1	Wr	Ex1-Ex2	Predicted
LDR	Literal	1	1	Wr	-	Predicted
LDR	Register	1 or 3	1	Wr	Ex2	Predicted[ci]
LDRB	Immediate	1	1	Wr	Ex1-Ex2	-
LDRB	Literal	1	1	Wr	-	-
LDRB	Register	1 or 3	1	Wr	Ex2	-
LDRBT		1	1	Wr	Ex1-Ex2	-
LDRD	Immediate	1	1	Wr	Ex2	-
LDRD	Literal	1	1	Wr	-	-
LDRD	Register	1 or 3	1	Wr	Ex2	-
LDREX		1	1	Wr	-	-
LDREXB		1	1	Wr	-	-
LDREXD		1	1	Wr	-	-
LDREXH		1	1	Wr	-	-
LDRH	Immediate	1	1	Wr	Ex1-Ex2	-
LDRH	Literal	1	1	Wr	-	-

[ci] The instruction is predicted only for the A1 encoding when Offset addressing mode is used.

Table B-2 Base instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	Main result	Base register update	Branch prediction
LDRH	Register	1 or 3	1	Wr	Ex1-Ex2	-
LDRHT		1	1	Wr	Ex1-Ex2	-
LDRSB	Immediate	1	1	Wr	Ex1-Ex2	1
LDRSB	Literal	1	1	Wr	-	-
LDRSB	Register	1 or 3	1	Wr	Ex1-Ex2	-
LDRSBT		1	1	Wr	Ex1-Ex2	-
LDRSH	Immediate	1	1	Wr	Ex1-Ex2	-
LDRSH	Literal	1	1	Wr	-	-
LDRSH	Register	1 or 3	1	Wr	Ex1-Ex2	-
LDRSHT		1	1	Wr	Ex1-Ex2	-
LDRT		1	1	Wr	Ex1-Ex2	-
LSL, LSLS	Immediate	1	0	Ex1	-	Not predicted
LSL, LSLS	Register	1	0	Ex1	-	-
LSR, LSRS	Immediate	1	0	Ex1	-	Not predicted
LSR, LSRS	Register	1	0	Ex1	-	-
MCR		1-6	0	-	-	-
MCRR		1	0	-	-	-
MLA, MLAS		1	0	Wr	-	-
MLS		1	0	Wr	-	-
MOV, MOVS	Immediate	1	0	Ex1	-	Stop
MOV, MOVS	Register	1	0	Ex1-Ex2	-	Stop
MOVT		1	0	Ex2	-	-
MRC		1-2	0	Wr	-	-
MRRC		1	0	Wr	-	-
MRS		1	0	Wr	-	-
MRS	Banked register	1	0	Wr	-	-
MSR	Banked register	1	0	Wr	-	-
MSR	Immediate	1 or 5	0	-	-	-
MSR	Register	1 or 5	0	-	-	-
MUL, MULS		1	0	Wr	-	-
MVN, MVNS	Immediate	1	0	Ex1	-	Not predicted
MVN, MVNS	Register	1	0	Ex1-Ex2	-	Not predicted
MVN, MVNS	Register-shifted register	1	0	Ex2	-	-

Table B-2 Base instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	Main result	Base register update	Branch prediction
NOP		1	0	-	-	-
ORN, ORNS	Immediate	1	0	Ex1-Ex2	-	-
ORN, ORNS	Register	1	0	Ex1 - Ex2	-	-
ORR, ORRS	Immediate	1	0	Ex1-Ex2	-	Not predicted
ORR, ORRS	Register	1	0	Ex1-Ex2	-	Not predicted
ORR, ORRS	Register-shifted register	1	0	Ex2	-	-
PKHBT, PKHTB		1	0	Ex2	-	-
PLD, PLDB	Immediate	1	1	-	-	-
PLD	Literal	1	1	-	-	-
PLD, PLDW	Register	1 or 3	1	-	-	-
POP		1-6	1-6	Wr	Ex2	Predicted
PUSH		1-6	1-6	-	Ex2	-
QADD		1	0	Wr	-	-
QADD16		1	0	Wr	-	-
QADD8		1	0	Wr	-	-
QASX		1	0	Wr	-	-
QDADD		1	0	Wr	-	-
QDSUB		1	0	Wr	-	-
QSAX		1	0	Wr	-	-
QSUB		1	0	Wr	-	-
QSUB16		1	0	Wr	-	-
QSUB8		1	0	Wr	-	-
RBIT		1	0	Ex2	-	-
REV		1	0	Ex2	-	-
REV16		1	0	Ex2	-	-
REVSH		1	0	Ex2	-	-
RFE, RFEDA, RFEDB, RFEIA,RFEIB		1	1	Ex2	-	Stop
ROR, RORS	Immediate	1	0	Ex1	-	Not predicted
ROR, RORS	Register	1	0	Ex1	-	-
RRX, RRXS		1	0	Ex2	-	Not predicted
RSB, RSBS	Immediate	1	0	Ex1-Ex2	-	Stop
RSB, RSBS	Register	1	0	Ex1-Ex2	-	Stop

Table B-2 Base instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	Main result	Base register update	Branch prediction
RSB, RSBS	Register-shifted register	1	0	Ex2	-	-
RSC, RSCS	Immediate	1	0	Ex2	-	Not predicted
RSC, RSCS	Register	1	0	Ex2	-	Not predicted
RSC, RSCS	Register-shifted register	1	0	Ex2	-	-
SADD16		1	0	Ex1-Ex2	-	-
SADD8		1	0	Ex1-Ex2	-	-
SASX		1	0	Ex1-Ex2	-	-
SBC, SBCS	Immediate	1	0	Ex2	-	Not predicted
SBC, SBCS	Register	1	0	Ex2	-	Not predicted
SBC, SBCS	Register-shifted register	1	0	Ex2	-	-
SBFX		1	0	Ex2	-	-
SDIV		1-10[cj]	0	Wr	-	-
SEL		1	0	Ex2	-	-
SETEND		1	0	-	-	-
SEV		1	0	-	-	-
SEVL		1	0	-	-	-
SHADD16		1	0	Ex1-Ex2	-	-
SHADD8		1	0	Ex1-Ex2	-	-
SHASX		1	0	Ex1-Ex2	-	-
SHSAX		1	0	Ex1-Ex2	-	-
SHSUB16		1	0	Ex1-Ex2	-	-
SHSUB8		1	0	Ex1-Ex2	-	-
SMLABB, SMLABT, SMLATB, SMLATT		1	0	Wr	-	-
SMLAD, SMLADX		1	0	Wr	-	-
SMLAL, SMLALS		1	0	Wr	-	-
SMLALBB SMLALBT SMLALTB,SMLALTT		1	0	Wr	-	-
SMLALD, SMLALDX		1	0	Wr	-	-
SMLAWB, SMLAWT		1	0	Wr	-	-
SMLSD, SMLSDX		1	0	Wr	-	-

[cj] See *B.3.4 Division and square root* on page Appx-B-610

Table B-2 Base instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	Main result	Base register update	Branch prediction
SMLSLD, SMLSLDX		1	0	Wr	-	-
SMMLA, SMMLAR		1	0	Wr	-	-
SMMLS, SMMLSR		1	0	Wr	-	-
SMMUL, SMMULR		1	0	Wr	-	-
SMUAD, SMUADX		1	0	Wr	-	-
SMULBB, SMULBT, SMULTB, SMULTT		1	0	Wr	-	-
SMULL, SMULLS		1	0	Wr	-	-
SMULWB, SMULWT		1	0	Wr	-	-
SMUSD, SMUSDX		1	0	Wr	-	-
SRS, SRSDA, SRSDB, SRSIA, SRSIB		1	1	Ex2	-	-
SSAT		1	0	Ex2	-	-
SSAT16		1	0	Ex2	-	-
SSAX		1	0	Ex1-Ex2	-	-
SSUB16		1	0	Ex1-Ex2	-	-
SSUB8		1	0	Ex1-Ex2	-	-
STC		1	1	Ex2	-	-
STL		2	1	-	-	-
STLB		2	1	-	-	-
STLEX		2	1	Wr	-	-
STLEXB		2	1	Wr	-	-
STLEXD		2	1	Wr	-	-
STLEXH		2	1	Wr	-	-
STLH		2	1	-	-	-
STM, STMIA, STMEA		1-8	1-8	-	Ex2	-
STM	User registers	1-8	1-8	-	-	-
STMDA, STMED		1-8	1-8	-	Ex2	-
STMDB, STMFD		1-8	1-8	-	Ex2	-
STMIB, STMFA		1-8	1-8	-	Ex2	-
STR	Immediate	1	1	-	Ex1-Ex2	-
STR	Register	1 or 3	1	-	Ex2	-
STRB	Immediate	1	1	-	Ex1-Ex2	-
STRB	Register	1 or 3	1	-	Ex2	-

Table B-2 Base instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	Main result	Base register update	Branch prediction
STRBT		1	1	-	Ex1-Ex2	-
STRD	Immediate	1	1	-	Ex2	-
STRD	Register	1 or 3	1	-	Ex2	-
STREX		1	1	Wr	-	-
STREXB		1	1	Wr	-	-
STREXD		1	1	Wr	-	-
STREXH		1	1	Wr	-	-
STRH	Immediate	1	1	-	Ex1-Ex2	-
STRH	Register	1-3	1	-	Ex1-Ex2	-
STRHT		1	1	-	Ex1-Ex2	-
STRT		1	1	-	Ex1-Ex2	-
SUB, SUBS	Immediate	1	0	Ex1-Ex2	-	Stop
SUB, SUBS	Register	1	0	Ex1-Ex2	-	Stop
SUB, SUBS	Register-shifted register	1	0	Ex2	-	-
SVC		1	0	-	-	-
SXTAB		1	0	Ex2	-	-
SXTAB16		1	0	Ex2	-	-
SXTAH		1	0	Ex2	-	-
SXTB		1	0	Ex2	-	-
SXTB16		1	0	Ex2	-	-
SXTH		1	0	Ex2	-	-
TBB, TBH		1	1	-	-	Stop
TEQ	Immediate	1	0	-	-	-
TEQ	Register	1	0	-	-	-
TEQ	Register-shifted register	1	0	-	-	-
TST	Immediate	1	0	-	-	-
TST	Register	1	0	-	-	-
TST	Register-shift register	1	0	-	-	-
UADD16		1	0	Ex1-Ex2	-	-
UADD8		1	0	Ex1-Ex2	-	-
UASX		1	0	Ex1-Ex2	-	-
UBFX		1	0	Ex2	-	-

Table B-2 Base instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	Main result	Base register update	Branch prediction
UDIV		1-9[cj]	0	Wr	-	-
UHADD16		1	0	Ex1-Ex2	-	-
UHADD8		1	0	Ex1-Ex2	-	-
UHASX		1	0	Ex1-Ex2	-	-
UHSAX		1	0	Ex1-Ex2	-	-
UHSUB16		1	0	Ex1-Ex2	-	-
UHSUB8		1	0	Ex1-Ex2	-	-
UMAAL		2	0	Wr	-	-
UMLAL, UMLALS		1	0	Wr	-	-
UMULL, UMULLS		1	0	Wr	-	-
UQADD16		1	0	Wr	-	-
UQADD8		1	0	Wr	-	-
UQASX		1	0	Wr	-	-
UQSAX		1	0	Wr	-	-
UQSUB16		1	0	Wr	-	-
UQSUB8		1	0	Wr	-	-
USAD8		1	0	Wr	-	-
USADA8		1	0	Wr	-	-
USAT		1	0	Ex2	-	-
USAT16		1	0	Ex2	-	-
USAX		1	0	Ex1-Ex2	-	-
USUB16		1	0	Ex1-Ex2	-	-
USUB8		1	0	Ex1-Ex2	-	-
UXTAB		1	0	Ex2	-	-
UXTAB16		1	0	Ex2	-	-
UXTAH		1	0	Ex2	-	-
UXTB		1	0	Ex2	-	-
UXTB16		1	0	Ex2	-	-
UXTH		1	0	Ex2	-	-
WFE		5	0	-	-	-
WFI		5	0	-	-	-
YIELD		1	0	-	-	-

B.2.3 Floating-point and Advanced SIMD instructions cycle timings

This section describes the cycle timing behavior of the floating-point and Advanced SIMD instructions in the A32 and T32 instruction sets.

Table B-3 Floating point and Advanced SIMD instructions cycle timings

Instruction	Variant	Cycles	Memory cycles	FP/Advanced SIMD main result	Integer main result	Base register update
VABA		2	-	F4	-	-
VABAL		2	-	F4	-	-
VABD	Floating-point	1	-	F5	-	-
VABD	Integer	1	-	F4	-	-
VABDL	Integer	1	-	F4	-	-
VABS		1	-	F4-F5	-	-
VACGE		1	-	F3	-	-
VACGT		1	-	F3	-	-
VADD	Integer	1	-	F4	-	-
VADD	Floating-point	1	-	F5	-	-
VADDHN		1	-	F5	-	-
VADDL		1	-	F4	-	-
VADDW		1	-	F4	-	-
VAND	Register	1	-	F3	-	-
VBIC	Immediate	1	-	F3	-	-
VBIC	Register	1	-	F3	-	-
VBIF		1	-	F3	-	-
VBIT		1	-	F3	-	-
VBSL		1	-	F3	-	-
VCEQ	Immediate #0	1	-	F3	-	-
VCEQ	Register	1	-	F3	-	-
VCGE	Immediate #0	1	-	F3	-	-
VCGE	Register	1	-	F3	-	-
VCGT	Immediate #0	1	-	F3	-	-
VCGT	Register	1	-	F3	-	-
VCLE	Immediate #0	1	-	F3	-	-
VCLS		1	-	F4	-	-
VCLT	Immediate #0	1	-	F3	-	-
VCLZ		1	-	F4	-	-
VCMP		1	-	-	-	-
VCMPE		1	-	-	-	-

Table B-3 Floating point and Advanced SIMD instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	FP/Advanced SIMD main result	Integer main result	Base register update
VCNT		1	-	F3	-	-
VCVT	Between double-precision and single-precision	1	-	F5	-	-
VCVT	Between floating-point and fixed-point, Advanced SIMD	1	-	F5	-	-
VCVT	Between floating-point and fixed-point, floating-point	1	-	F5	-	-
VCVT	Between floating-point and integer, Advanced SIMD	1	-	F5	-	-
VCVT	Between half-precision and single-precision, Advanced SIMD	1	-	F5	-	-
VCVT	Floating-point to integer, floating-point	1	-	F5	-	-
VCVT	Integer to floating-point, floating-point	1	-	F5	-	-
VCVTA	Advanced SIMD	1	-	F5	-	-
VCVTA	Floating-point	1	-	F5	-	-
VCVTB		1	-	F5	-	-
VCVTM	Advanced SIMD	1	-	F5	-	-
VCVTM	Floating-point	1	-	F5	-	-
VCVTN	Advanced SIMD	1	-	F5	-	-
VCVTN	Floating-point	1	-	F5	-	-
VCVTP	Advanced SIMD	1	-	F5	-	-
VCVTP	Floating-point	1	-	F5	-	-
VCVTR		1	-	F5	-	-
VCVTT		1	-	F5	-	-
VDIV		1	-	F5(+9 or +18)	-	-
VDUP	General purpose register	1	-	F3	-	-
VDUP	Scalar	1	-	F3	-	-
VEOR		1	-	F3	-	-
VEXT	Byte elements	1	-	F3	-	-
VFMA		1	-	F5(+4)	-	-

Table B-3 Floating point and Advanced SIMD instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	FP/Advanced SIMD main result	Integer main result	Base register update
VFMS		1	-	F5(+4)	-	-
VFNMA		1	-	F5(+4)	-	-
VFNMS		1	-	F5(+4)	-	-
VHADD		1	-	F4	-	-
VHSUB		1	-	F4	-	-
VLD1	Multiple single elements	1-4	1-4	F4	-	Ex1-Ex2
VLD1	Single element to all lanes	1	1	F4	-	Ex1-Ex2
VLD1	Single element to one lane	1	1	F4	-	Ex1-Ex2
VLD2	Multiple 2-element structures	2-4	2-4	F4	-	Ex2
VLD2	Single 2-element structure to all lanes	1	1	F4	-	Ex1-Ex2
VLD2	Single 2-element structure to one lane	1	1	F4	-	Ex1-Ex2
VLD3	Multiple 3-element structures	3-4	3	F4	-	Ex2
VLD3	Single 3-element structure to all lanes	2	2	F4	-	Ex2
VLD3	Single 3-element structure to one lane	2	2	F4	-	Ex2
VLD4	Multiple 4-element structures	4-5	4	F4	-	Ex2
VLD4	Single 4-element structure to all lanes	2	2	F4	-	Ex2
VLD4	Single 4-element structure to one lane	2	2	F4	-	Ex2
VLDM, VLDMDB, VLDMIA		1-16	1-16	F4	-	Ex2
VLDR		1	1	F4	-	Ex2
VMAX	Floating-point	1	-	F5	-	-
VMAX	Integer	1	-	F3	-	-
VMAXNM		1	-	F5	-	-
VMIN	Floating-point	1	-	F5	-	-
VMIN	Integer	1	-	F3	-	-
VMINNM		1	-	F5	-	1

Table B-3 Floating point and Advanced SIMD instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	FP/Advanced SIMD main result	Integer main result	Base register update
VMLA	By scalar	1	-	F5(+0 or +4)	-	-
VMLA	Floating-point	1	-	F5(+4)	-	-
VMLA	Integer	1	-	F5	-	-
VMLAL	By scalar	1	-	F5	-	-
VMLAL	Integer	1	-	F5	-	-
VMLS	By scalar	1	-	F5(+0 or +4)	-	-
VMLS	Floating-point	1	-	F5(+4)	-	-
VMLS	Integer	1	-	F5	-	-
VMLSL	By scalar	1	-	F5	-	-
VMLSL	Integer	1	-	F5	-	-
VMOV	Between general purpose register and single-precision register	1	-	F3	Wr	-
VMOV	Between two general purpose registers and doubleword floating-point register	1	-	F3	Wr	-
VMOV	Between two general purpose registers and two single-precision registers	1	-	F3	Wr	-
VMOV	General purpose register to scalar	1	-	F3	-	-
VMOV	Immediate	1	-	F3	-	-
VMOV	Register	1	-	F3	-	-
VMOV	Scalar to general-purpose register	1	-	F3	Wr	-
VMOVL		1	-	F3	-	-
VMOVN		1	-	F3	-	-
VMRS		1	-	-	Wr	-
VMSR		7	-	-	-	-
VMUL	By scalar	1	-	F5	-	-
VMUL	Floating-point	1	-	F5	-	-
VMUL	Integer and polynomial	1	-	F5	-	-
VMULL	By scalar	1	-	F5	-	-
VMULL	Integer and polynomial	1	-	F4-F5	-	-
VMVN	Immediate	1	-	F3	-	-

Table B-3 Floating point and Advanced SIMD instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	FP/Advanced SIMD main result	Integer main result	Base register update
VMVN	Register	1	-	F3	-	-
VNEG		1	-	F4-F5	-	-
VNMLA		1	-	F5(+4)	-	-
VNMLS		1	-	F5(+4)	-	-
VNMUL		1	-	F5	-	-
VORN	Register	1	-	F3	-	-
VORR	Immediate	1	-	F3	-	-
VORR	Register	1	-	F3	-	-
VPADAL		2	-	F4	-	-
VPADD	Floating-point	1	-	F5	-	-
VPADD	Integer	1	-	F4	-	-
VPADDL		1	-	F4	-	-
VPMAX	Floating-point	1	-	F5	-	-
VPMAX	Integer	1	-	F3	-	-
VPMIN	Floating-point	1	-	F5	-	-
VPMIN	Integer	1	-	F3	-	-
VPOP		1-16	1-16	F4	-	Ex2
VPUSH		1-16	1-16		-	Ex2
VQABS		1	-	F5	-	-
VQADD		1	-	F5	-	-
VQDMLAL		1	-	F5	-	-
VQDMLSL		1	-	F5	-	-
VQDMULH		1	-	F5	-	-
VQDMULL		1	-	F5	-	-
VQMOVN, VQMOVUN		1	-	F5	-	-
VQNEG		1	-	F5	-	-
VQRDMULH		1	-	F5	-	-
VQRSHL		1	-	F5	-	-
VQRSHRN, VQRSHRUN		1	-	F5	-	-
VQSHL, VQSHLU	Immediate	1	-	F5	-	-
VQSHL	Register	1	-	F5	-	-
VQSHRN, VQSHRUN		1	-	F5	-	-
VQSUB		1	-	F5	-	-

Table B-3 Floating point and Advanced SIMD instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	FP/Advanced SIMD main result	Integer main result	Base register update
VRADDHN		2	-	F5	-	-
VRECPE		1	-	F5	-	-
VRECPS		1	-	F5(+4)	-	-
VREV16		1	-	F3	-	-
VREV32		1	-	F3	-	-
VREV64		1	-	F3	-	-
VRHADD		1	-	F4	-	-
VRINTA	Advanced SIMD	1	-	F5	-	-
VRINTA	Floating-point	1	-	F5	-	-
VRINTM	Advanced SIMD	1	-	F5	-	-
VRINTM	Floating-point	1	-	F5	-	-
VRINTN	Advanced SIMD	1	-	F5	-	-
VRINTN	Floating-point	1	-	F5	-	-
VRINTP	Advanced SIMD	1	-	F5	-	-
VRINTP	Floating-point	1	-	F5	-	-
VRINTR		1	-	F5	-	-
VRINTX	Advanced SIMD	1	-	F5	-	-
VRINTX	Floating-point	1	-	F5	-	-
VRINTZ	Advanced SIMD	1	-	F5	-	-
VRINTZ	Floating-point	1	-	F5	-	-
VRSHL		1	-	F4	-	-
VRSHR		1	-	F4	-	-
VRSHRN		1	-	F4	-	-
VRSQRTE		1	-	F5	-	-
VRSQRTS		1	-	F5(+4)	-	-
VRSRA		2	-	F4	-	-
VRSUBHN		2	-	F5	-	-
VSELEQ, VSELGE, VSELGT, VSELVS		1	-	F3	-	-
VSHL	Immediate	1	-	F3	-	-
VSHL	Register	1	-	F3	-	-
VSHLL		1	-	F3	-	-
VSHR		1	-	F3	-	-
VSHRN		1	-	F3	-	-

Table B-3 Floating point and Advanced SIMD instructions cycle timings (continued)

Instruction	Variant	Cycles	Memory cycles	FP/Advanced SIMD main result	Integer main result	Base register update
VSLI		1	-	F3	-	-
VSQRT		1	-	F5 (+8 or +18)	-	-
VSRA		1	-	F4	-	-
VSRI		1	-	F3	-	-
VST1	Multiple single elements	1-4	1-4	-	-	Ex2
VST1	Single element from one lane	1	1	-	-	Ex2
VST2	Multiple 2-element structures	2-4	2-4	-	-	Ex1-Ex2
VST2	Single 2-element structure from one lane	1	1	-	-	Ex2
VST3	Multiple 3-element structures	4	3	-	-	Ex1-Ex2
VST3	Single 3-element structure from one lane	2	2	-	-	Ex1-Ex2
VST4	Multiple 4-element structures	5	4	-	-	Ex1-Ex2
VST4	Single 4-element structure from one lane	2	2	-	-	Ex1-Ex2
VSTM, VSTMDB, VSTMIA		1-16	1-16	-	-	Ex2
VSTR		1	1	-	-	-
VSUB	Floating-point	1	-	F5	-	-
VSUB	Integer	1	-	F4	-	-
VSUBHN		1	-	F5	-	-
VSUBL		1	-	F4	-	-
VSUBW		1	-	F4	-	-
VSWP		1-2	-	F3	-	-
VTBL, VTBX		1-3	-	F3	-	-
VTRN		2	-	F3	-	-
VTST		1	-	F5	-	-
VUZP		2	-	F3	-	-
VZIP		1-2	-	F3	-	-

B.3 Pipeline behavior

This section provides the detailed aspects of pipeline behavior.

This section contains the following subsections:

B.3.1 Skewing

Normally, within each of the two integer addition or subtraction pipelines, an instruction accesses the ALU in the Ex2 stage, which means the operands are required at the end of Ex1 and the result is generated at the end of Ex2.

The Cortex-R52 processor features secondary *skewed* ALUs in the Ex1 stage of the pipeline. Most of the simple logical or arithmetic instructions that do not require shifting or saturation resources can access the skewed ALU in the Ex1 stage. By doing so, the instruction result is available one stage earlier for forwarding to dependent instructions. Also, the instruction inputs must be ready one stage earlier. The Cortex-R52 processor dynamically decides whether an instruction must be skewed or not to minimize the pipeline interlocks.

B.3.2 Dual-issuing

To increase instruction throughput, the Cortex-R52 processor can issue pairs of instructions simultaneously. This is called dual-issuing.

Most instructions capable of being dual-issued are examined for hazards in the *Decode* (De) stage of the pipeline. Both instructions must reach the De stage at the same time. This is less likely if there are many branches. If the Cortex-R52 processor determines that the pair must be dual-issued, it remains a pair until both instructions are retired. If one instruction of the pair is interlocked, both are interlocked.

The following broad classes of structural hazards limit the amount of dual-issuing:

- Only one load/store instruction can issue per cycle.
- Only one branch instruction or instruction that writes the PC can issue per cycle.
- Only one integer multiplication or division instruction can issue per cycle.
- Only one floating-point or Advanced SIMD division or square root instruction can issue per cycle.
- Advanced SIMD instructions that operate on 128-bit vectors cannot dual-issue with other floating-point or Advanced SIMD instructions.
- Multicycle instructions have limited dual-issuing capabilities.
- Load-acquire, store-release, and load/store-exclusive instructions have limited dual-issuing capabilities.
- Loads or stores that subtract a register from the base register or that perform complex shifts (shifts other than `LSL` of 0-3) have limited dual-issuing capabilities.
- Most instructions that access system registers cannot dual-issue. [ck]
- Instructions which read or write an unusually high number of registers have limited dual-issuing capabilities.

The following broad classes of dynamic hazards limit the amount of dual-issuing:

- Unless a suitable forwarding path exists, Read-After-Write hazards between the two instructions are not dual-issued.
- If dual-issuing a pair of instructions causes more interlocks that single-issuing, then the Cortex-R52 processor chooses to single-issue instead.

[ck] An exception to that rule is direct access to compatible MPU region address register pairs (`PRBARn` with `PRLARn`, or `HPRBAR` with `HPRLAR`) can dual-issue.

B.3.3 Load/store instructions

Typically, instructions which perform a single load or store have Cycles count of 1, for example, `LDR` and `STR`.

In the A32 instruction set, some of the instructions support addressing modes with either negative register offsets or shifts other than `LSL` #0, #1, #2, or #3. The Cycles count for these instructions is three because the full ALU has to compute the address to be loaded or stored.

Instructions which perform multiple loads or stores, for example, `LDM` and `STM`, have Cycles and Memory Cycles counts. The number depends on the number and width of registers that are accessed.

- For integer load/store multiples (`LDM`, `STM`, `POP`, `PUSH`) and for floating-point load/store multiples (`VLDM`, `VSTM`, `VPOP`, `VPUSH`) when accessing single-precision registers, the Cycles count and number of Memory Cycles is given as:

  Cycles = Memory Cycles = number of registers/2
- For floating-point load/store multiples (`VLDM`, `VSTM`, `VPOP`, `VPUSH`), when accessing double-precision registers, the Cycles count and number of Memory Cycles is given as:

  Cycles = Memory Cycles = number of registers
- For Advanced SIMD load/store multiples (`VLD1`, `VLD2`, `VLD3`, `VLD4`, `VST1`, `VST2`, `VST3`, `VST4`) the number of cycles depends on multiple factors. For information on the ranges for Cycles and Memory Cycles per instruction, see *Table B-3 Floating point and Advanced SIMD instructions cycle timings* on page Appx-B-602.

Note

If the Cycles count and Memory Cycles is a fractional result, it is rounded up.

Accesses to addresses that cross a 64-bit aligned boundary make additional memory accesses. For integer and floating-point load/stores or load/store-multiples, the Cycles and Memory Cycles is increased by one. For Advanced SIMD load/store multiples, the Memory Cycles are doubled and the Cycles are increased accordingly.

B.3.4 Division and square root

Integer divisions (`UDIV` and `SDIV` instructions) operate in-order. The Cycles required for the instruction to complete is variable and data-dependent.

The Cycles for a `UDIV` instruction that performs division of value A by B is given as:

$$1 + max\left(\left(\frac{clz(B) - clz(A) + 1}{4}\right), 0\right)$$

The Cycles for a `SDIV` instruction that performs division of value A by B is given as:

$$2 + max\left(\left(\frac{clz(B) - clz(A) + 1}{4}\right), 0\right)$$

Floating-point divisions and square roots (`VDIV` and `VSQRT` instructions) operate out-of-order. These instructions always complete in a single cycle but there is a delay in the division or square root result being written to the register file. Subsequent instructions are allowed to issue, execute, and retire, provided they do not depend on the result of the `VDIV`, or `VSQRT`, and they are not `VDIV` or `VSQRT` themselves. If a dependency is detected, the pipeline interlocks and waits for the availability of the result before continuing.

The delayed result for a `VDIV` instruction happens nine cycles later than normal for a single-precision operation or 18 cycles later than normal for a double-precision operation. These latencies are data-

independent. In *Table B-3 Floating point and Advanced SIMD instructions cycle timings on page Appx-B-602*, they are denoted as (+N) notation.

The delayed result write for a `VSQRT` instruction happens eight cycles later than normal for a single-precision operation or 18 cycles later than normal for a double-precision operation. These latencies are data-independent.

Note

- Any divide or square root instruction that fails its condition code has a Cycle count of 1. There is no delayed register file write for floating-point instructions which fail their condition code.
- The *clz(x)* function counts the number of leading zeros in the 32-bit value x. If x is negative, it is negated before this count occurs.
- The value of the *(clz(B) - clz(A) + 1)/4* component of these equations must be rounded down.

B.3.5 Floating-point and Advanced SIMD Multiply-Accumulate instructions

Floating-point multiply-accumulate operations first use the floating-point multiplication pipeline and then use the floating-point addition pipeline.

The second part is done in an out-of-order fashion. When the multiplication part of the instruction reaches the F5 stage of the pipeline, the partial result is forwarded back to the F1 stage of the pipeline that continues with the addition part of the instruction. If a newer floating-point addition instruction is in F1 at the time, the second part of the multiply-accumulate takes precedence over it. The new instruction interlocks for one cycle to make space for the older multiply-accumulate to complete. Other types of instruction, including newer floating-point multiply accumulate instructions can issue in parallel with the second part of a floating-point multiply-accumulate.

The result of such instructions (`VFMA`, `VFMS`, `VFNMA`, `VFNMS`, `VMLA`, `VMLS`, `VNMLA`, `VNMLS`) is written in a delayed fashion four cycles later than normal.

For example, the following code takes eight cycles to execute:

```
VFMA.F32 D16, D0, D0
VFMA.F32 D17, D1, D0  ; Dual-issued with the above
VFMA.F32 D18, D2, D0
VFMA.F32 D19, D3, D0  ; Dual-issued with the above
VFMA.F32 D20, D4, D0VFMA.F32 D21, D5, D0  ; Dual-issued with the above
VFMA.F32 D22, D6, D0
VFMA.F32 D23, D7, D0  ; Dual-issued with the above
VFMA.F32 D24, D8, D0VFMA.F32 D25, D9, D0  ; Dual-issued with the above
VFMA.F32 D26, D10, D0VFMA.F32 D27, D11, D0 ; Dual-issued with the above
VFMA.F32 D28, D12, D0VFMA.F32 D29, D13, D0 ; Dual-issued with the above
VFMA.F32 D30, D14, D0VFMA.F32 D31, D15, D0 ; Dual-issued with the above
```

The following code takes 12 cycles to execute because the four pairs of `VFMA` instructions use the floating-point addition pipelines and prevent the `VADD` instructions from issuing:

```
VFMA.F32 D16, D0, D0
VFMA.F32 D17, D1, D0  ; Dual-issued with the above
VFMA.F32 D18, D2, D0
VFMA.F32 D19, D3, D0  ; Dual-issued with the above
VFMA.F32 D20, D4, D0VFMA.F32 D21, D5, D0  ; Dual-issued with the above
VFMA.F32 D22, D6, D0
VFMA.F32 D23, D7, D0  ; Dual-issued with the above
VADD.F32 D24, D8, D0  ; Interlocks for four cycles because of VFMAs
VADD.F32 D25, D9, D0  ; Dual-issued with the above
VADD.F32 D26, D10, D0VADD.F32 D27, D11, D0 ; Dual-issued with the above
VADD.F32 D28, D12, D0VADD.F32 D29, D13, D0 ; Dual-issued with the above
VADD.F32 D30, D14, D0VADD.F32 D31, D15, D0 ; Dual-issued with the above
```

B.3.6 Instructions with exceptional behavior

Certain instructions might take an exception when they reach the *Write* (Wr) stage of the pipeline.

For example, `SVC` or `HVC` might take an exception to EL1 or EL2. Instructions might cause a Prefetch Abort or might be trapped by an EL1 or EL2 control register. For such cases, the pipeline is flushed and the Cycles count must be increased by eight.

Similarly, though they do not take any exceptions, there are certain instructions that might flush the pipeline. For example, the `ISB` instruction always flushes the pipe. `CPS` and `MSR` instructions might flush the pipe when they update CPSR.M. An exception to this rule is when CPS instructions are used to change modes within EL1. For example, when in FIQ mode, the pipeline is not flushed.

Certain `MCR` and `MCRR` accesses (IMP_ATCMREGIONR, IMP_BTCMREGIONR, IMP_CTCMREGIONR, IMP_CSCTLR, HSCTLR, IMP_QOSR, SCTLR, VSCTLR, and CPUACTLR) also cause the pipeline to flush. For simplicity, the cycle timing tables do not assume flushes. The Cycles count must be increased by eight for such cases.

Some instructions that write system registers are blocking. Examples of such instructions include some of the MCR, MCRR, MSR, VMSR instructions. This blocking refers to no subsequent instruction starting execution before the system register write is completed and being visible for all subsequent instructions. Therefore, such instructions might take more Cycles to complete.

All transfers to and from the floating-point and Advanced SIMD system registers (`VMRS`, `VMSR`) are also serializing. This indicates that if there are any outstanding out-of-order completion instructions, the system register transfer instruction stalls in the Iss stage until these instructions are complete.

Appendix C
Processor UNPREDICTABLE Behaviors

This appendix describes specific Cortex-R52 processor UNPREDICTABLE behaviors of particular interest. These UNPREDICTABLE behaviors differ from the Arm standard behavior.

For each scenario, the Arm standard specification is listed under *Specification* and the Cortex-R52 implementation is listed under *Implementation*. For detailed background information on UNPREDICTABLE behaviors, see *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*.

It contains the following sections:

C.1 Use of R15 by Instruction

The section describes the specification and implementation of R15 by instruction.

Specification

All uses of R15 as a named register specifier for a source register that are described as UNPREDICTABLE in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* pseudo-code, or in other places in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*, read 0 unless otherwise stated in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*, or as described in the following paragraph.

If the use of R15 as a base register for a load or store is UNPREDICTABLE, the value used by the load or store using R15 as a base register is the PC with its usual offset and, in the case of T32 instructions, with the forced word alignment. In this case, if the instruction specifies write-back, then the load or store is performed without write-back.

All uses of R15 as a named register specifier for a destination register that are described as UNPREDICTABLE in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* pseudo-code, or in other places in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*, ignore the write. For instructions which have two destination registers, for example, LDRD, MRCC, and many of the multipliers, if either Rt or Rt2 is R15, or if either RdLo or RdHi is R15, then the other destination register is the pair is UNKNOWN.

For instructions which affects any of the flags (CPSR.NZCV/Q/GE) if the register specifier is not R15, if the instruction is UNPREDICTABLE when the register specifier is R15, then the flags that the instruction affects become UNKNOWN.

In addition, `MRC` instructions for CP14 and CP15 that use R15 as the target register descriptor and therefore target APSR_nzcv, the APSR_nzvc bits become UNKNOWN where these are described as being UNPREDICTABLE.

Implementation

The Cortex-R52 processor deviates from this behavior in the following ways:

- If using R15 as a source register is UNPREDICTABLE, then the processor treats the instruction as UNDEFINED. This behavior is also aligned in the following specific UNPREDICTABLE cases for instructions.
  - — *C.4.1 CLZ* on page Appx-C-618.
  - — *C.4.2 PUSH* on page Appx-C-618.
  - — *C.4.3 RBIT* on page Appx-C-618.
  - — *C.4.4 REV, REV16, REVSH* on page Appx-C-618.
  - — *C.4.5 STC* on page Appx-C-619.
  - — *C.4.6 STM/STMIA/STMEA* on page Appx-C-619.
  - — *C.4.7 STMDA/STMED and STMIB/STMFA* on page Appx-C-619.
  - — *C.4.8 STMDB/STMFD* on page Appx-C-619.
  - — *C.4.9 STR (Immediate, Thumb), STR (Immediate, Arm), STR (register), STRB (immediate, Thumb), STRB (immediate, Arm), STRB (register), STRBT, STRH (immediate, Thumb), STRH (immediate, Arm), STRH (register), STRHT, and STRT* on page Appx-C-620.
  - — *C.4.10 STRD (immediate) and STRD (register)* on page Appx-C-620.
  - — *C.4.11 STREX, STREXB, STREXD, STREXH, STLEX, STLEXB, STLEXD, and STLEXH* on page Appx-C-621.

  The exceptions to this behavior are for the other specific UNPREDICTABLE cases that are described in the *Armv8 AArch32 UNPREDICTABLE behaviours* that not mentioned in the previous list.
- If using R15 as a base register is UNPREDICTABLE, the processor treats the instruction as UNDEFINED.
- If using R15 as a destination register UNPREDICTABLE, the processor treats the instruction as UNDEFINED.

C.2 UNPREDICTABLE instructions within an IT block

This section describes the specification and implementation of UNPREDICTABLE instructions within an IT block.

Instructions affected

- `CRC32` encodings A1, T1.
- `CRC32C` encodings A1, T1.

Specification

Several instructions are described in the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile* as UNPREDICTABLE either:

- Anywhere within an IT block.
- As any instruction in an IT block other than the last instruction within an IT block.

In Arm implementations of the Armv8 architecture, unless otherwise stated, such instructions take an UNDEFINED exception.

Implementation

The Cortex-R52 processor deviates from this behavior and the affected instructions are executed conditionally.

C.3 Instruction fetches from Device memory

This section describes the specification and implementation of instruction fetches from Device memory.

Specification

Instruction fetches from Device memory are UNPREDICTABLE.

If a region of memory has the Device attribute and is not marked as XN, then an implementation might perform speculative instruction accesses to this memory location at times when the MMU is enabled.

If branches cause the PC to point at an area of memory with the Device attribute for instruction fetches which is not marked as XN for the current exception level, an implementation treats the instruction fetch as if it is to a memory location with the Normal Non-cacheable attribute.

Implementation

The Cortex-R52 processor deviates from this behavior in the following ways:

- If the speculative instruction access happens on an AXI memory region, the processor takes a permission fault.
- If the speculative instruction access happens on a Flash or TCM memory region, the processor treats the instruction fetch as if it is to a memory location with the Normal Non-cacheable attribute.
- If the speculative instruction access happens on an LLPP memory region, the processor takes a synchronous external abort.

C.4 Specific UNPREDICTABLE cases for instructions

This section describes specific UNPREDICTABLE cases for instructions.

C.4.1 CLZ

The section describes the specification and implementation of `CLZ`.

Specification

- For T1 encoding, the two source register specifiers must be consistent, therefore, hw1[3:0] must equal hw2[3:0].
- If these two register specifiers are not consistent, hw2[3:0] specifies a register that is used as a source register.

Implementation

The Cortex-R52 processor deviates from this behavior and treats the instruction as UNDEFINED.

C.4.2 PUSH

The section describes the specification and implementation of `PUSH`.

Specification

- If the number of registers stored is zero, then the instruction is treated as NOP.
- For T2 encoding:
  - If the number of registers stored is one, then the instruction stores a single register using the specified addressing modes.
  - If hw2 bit[13] is set, then the instruction performs all of the stores using the specified addressing mode including R13 as one of the target registers.
  - If hw2 bit [15] is set, then the instruction performs all the stores using the specified addressing mode but the value corresponding to R15 is 0.

Implementation

The Cortex-R52 processor deviates from this behavior and for T2 encoding, if hw2 bit[15] is set, then the processor performs all the stores using the specified addressing mode and the value corresponding to R15 is the PC with the standard offset that applies for the current instruction set.

C.4.3 RBIT

The section describes the specification and implementation of `RBIT`.

Specification

- For encoding T1, hw1[3:0] must equal hw2[3:0] because the two target register specifiers must be consistent.
- If these two register specifiers are not consistent, hw2[3:0] specifies a register that is used as a source register.

Implementation

The Cortex-R52 processor deviates from this behavior and treats the instruction as UNDEFINED.

C.4.4 REV, REV16, REVSH

The section describes the specification and implementation of `REV`, `REV16`, and `REVSH`.

Specification

- For encoding T1, hw1[3:0] must equal hw2[3:0] because the two target register specifiers must be consistent.
- If these two register specifiers are not consistent, hw2[3:0] specifies a register that is used as a source register.

Implementation

The Cortex-R52 processor deviates from this behavior and treats the instructions as UNDEFINED.

C.4.5 STC

The section describes the specification and implementation of `STC`.

Specification

If the base register for the `STC` instruction set is R15, then if W is 1 or if not executing in the Arm instruction set, then the store instruction is performed without a Writeback.

Implementation

The Cortex-R52 processor deviates from this behavior and treats the instructions as UNDEFINED.

C.4.6 STM/STMIA/STMEA

The section describes the specification and implementation of `STM`/`STMIA`/`STMEA`.

Specification

- If the number of registers stored is zero, then the instruction is treated as a NOP.
- For encoding T2:
  - If the instruction performs a write-back and the base register is in the list, then the instruction performs all of the stores using the specified addressing mode with the value stored for the base register being the value before the instruction was executed.
  - If the number of registers stored is one, the instruction stores a single register using the specified addressing modes.
  - If hw2 bit[13] is set, then the instruction performs all of the stores using the specified addressing mode including R13 as one of the target registers.
  - If hw2 bit[15] is set then, the instruction performs all of the stores using the specified addressing mode, but the value corresponding to R15 is 0.
- If the instruction uses R15 as a base register and specifies write-back, then the store instruction is performed without a write-back.

Implementation

The Cortex-R52 processor deviates from this behavior in the following ways:

- For the T2 encoding, if hw2 bit[15] is set, then the processor performs all of the stores using the specified addressing mode and the value corresponding to R15 is the PC with the standard offset that applies for the current instruction set.
- If the instruction uses R15 as a base register and specifies write-back, then the processor treats the instruction as UNDEFINED.

C.4.7 STMDA/STMED and STMIB/STMFA

The section describes the specification and implementation of `STMDA`/`STMED` and `STMIB`/`STMFA`.

Specification

- If the number of registers stored is zero, then the instruction is treated as a NOP.
- If the instruction uses R15 as a base register and specifies write-back, then the store instruction is performed without a write-back.

Implementation

The Cortex-R52 processor deviates from this behavior and if the instruction uses R15 as a base register and specifies write-back, then the processor treats the instructions as UNDEFINED.

C.4.8 STMDB/STMFD

The section describes the specification and implementation of `STMDB`/`STMFD`.

Specification

- If the number of registers stored is zero, then the instruction is treated as a NOP.
- For encoding T1:
  - The instruction performs a write-back and the base register is in the list, then the instruction performs all of the stores using the specified addressing mode with the value stored for the base register being the value before the instruction is executed.
  - If the number of registers stored is one, then the instruction stores a single register using the specified addressing modes.
  - If hw2 bit[13] is set, then the instruction performs all of the stores using the specifies addressing mode including R13 as one of the target registers.
  - If hw2 bit[15] is set, then the instruction performs all of the stores using the specified addressing mode, but the value corresponding to R15 is 0.
- If the instruction uses R15 as a base register and specifies write-back, then the store instruction is performed without a write-back.

Implementation

The Cortex-R52 processor deviates from this behavior in the following ways:

- For the T1 encoding, if hw2 bit[15] is set, then the processor performs all the stores using the specified addressing mode and the value corresponding to R15 is the PC with the standard offset that applies to the current instruction set.
- If the instruction uses R15 as a base register and specifies write-back, then the processor treats the instructions as UNDEFINED.

C.4.9 STR (Immediate, Thumb), STR (Immediate, Arm), STR (register), STRB (immediate, Thumb), STRB (immediate, Arm), STRB (register), STRBT, STRH (immediate, Thumb), STRH (immediate, Arm), STRH (register), STRHT, and STRT

This section describes the specification and implementation of `STR` (Immediate, Thumb), `STR` (Immediate, Arm), `STR` (register), `STRB` (immediate, Thumb), `STRB` (immediate, Arm), `STRB` (register), `STRBT`, `STRH` (Immediate, Thumb), `STRH` (Immediate, Arm), `STRH` (register), `STRHT`, and `STRT`.

Specification

- If R15 is specified as the transfer register and the instruction describes the behavior as UNPREDICTABLE, then the instruction performs the store using the specified addressing mode, but the value corresponding to R15 is 0.
- If the instruction performs a write-back and the register is stored in the base register, then the instruction performs the store using the specified addressing mode storing the value of the register before the write-back.
- If the instruction uses R15 as a base register and specifies write-back and the instruction is defined as UNPREDICTABLE, then the store instruction is performed without a write-back.

Implementation

The Cortex-R52 processor deviates from this behavior in the following ways:

- If R15 is specified as the transfer register and the instruction describes the behavior as UNPREDICTABLE, then the processor treats the instruction as UNDEFINED.
- If the instruction uses R15 as a base register and specifies write-back and the instruction is defined as UNPREDICTABLE, then the processor treats the instruction as UNDEFINED.

C.4.10 STRD (immediate) and STRD (register)

This section describes the specification and implementation of `STRD` (immediate) and `STRD` (register).

Specification

- For the A1 encoding of STRD (immediate) and STRD (register), if P is 0 and W is 1, then the instruction behaves as if the values of P and W are both 0.
- If R15 is specified as one of the transfer registers, then the instruction performs the store using the specified addressing mode but the value corresponding to R15 is 0.
- If the instruction performs a write-back and one of the registers stored is the base register, then the instruction performs the store of the registers specified using the specified addressing mode storing the value of the register before the write-back.
- If the instruction uses R15 as a base register and specifies write-back, then the store instruction is performed without a write-back.
- For all Arm STRD encodings, if Rt<0> is 1, then the behavior is the same as if Rt<0> is 0.

Implementation

The Cortex-R52 processor deviates from this behavior in the following ways:

- If R15 is specified as one of the transfer registers, then the processor treats the instruction as UNDEFINED.
- If the instruction uses R15 as a base register and specifies write-back, then the processor treats the instruction as UNDEFINED.

C.4.11 STREX, STREXB, STREXD, STREXH, STLEX, STLEXB, STLEXD, and STLEXH

This section describes the specification and implementation of STREX, STREXB, STREXD, STREXH, STLEX, STLEXB, STLEXD, and STLEXH.

Specification

- If the destination register for the exclusive result (Rd) is the same register as the transfer register, then the store instruction is performed using the value of the register transfer before the instruction was executed.
- If the destination register for the exclusive result (Rd) is the same register as the base address register, then the store instruction is performed using the value of the base address register before the instruction was executed.
- If the Store Exclusive is Strongly-ordered or Device memory type, then the instruction functions in the same way as it would occur if it was to Normal memory.
- For the Arm STREXD and STLEXD encoding, if Rt<0> is 1, then the behavior is the same as if Rt<0> is 0.
- For the Arm STREXD and STLEXD encoding, if Rt<0> is 1110, then the behavior of the instruction is specified that Rt is 1110 is not UNPREDICTABLE, but having t2 as 15 is UNPREDICTABLE.

Implementation

The Cortex-R52 processor deviates from this behavior and for the Arm STREXD and STLEXD encoding, if Rt is 1110, then the processor treats the instruction as UNDEFINED.

C.5 Load/Store accesses crossing MPU regions

This section describes load or store accesses that cross MPU regions

This section contains the following subsections:

C.5.1 Crossing an MPU region with different memory types or shareability attributes

This section describes the specification for crossing an MPU region with different memory types or shareability attributes.

Specification

In the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*, having memory accesses from one load or store instruction cross an MPU region with different memory types or shareability, is UNPREDICTABLE.

Implementation

In this situation, the implementation uses the memory type and shareability attributes associated with its own address for each memory access from this instruction.

C.5.2 Crossing a 4KB boundary with Device (or Strongly-Ordered) accesses

This section describes the specification and implementation of crossing a 4KB boundary with Device (or Strongly-Ordered) accesses.

Specification

In the *Arm® Architecture Reference Manual Supplement Armv8, for Armv8-R architecture profile*, having memory accesses from one load or store instruction to Device, or Strongly-ordered, memory cross a 4 KB boundary, is UNPREDICTABLE.

Implementation (for both page boundary specifications)

In this situation, both Loads and Stores behave as separate memory accesses with their own address on either side of the boundary. Therefore, any Device access is subject to alignment checks on the address associated with that access.

C.6 Armv8 Debug UNPREDICTABLE behaviors

This section describes the behavior that the Cortex-R52 processor implements when:

- A topic has multiple options.
- The processor behavior diverges from either or both of the *Options* and *Preference* behavior.

Additionally, for any particular topics for which the Cortex-R52 processor behavior diverges from either or both of the *Options* and *Preferences*, an explanation of the actual behavior is given. If the Cortex-R52 processor follows the recommended behavior, then this appendix does not comment on that topic or behavior.

For more information on the Armv8 Debug UNPREDICTABLE behaviors, see *Armv8 AArch32 UNPREDICTABLE behaviours*.

Table C-1 Armv8 Debug UNPREDICTABLE behaviors

Scenario	Behavior
A32 `BKPT` instruction with condition code not AL	The processor implements the following preferred option: • Executed unconditionally.
Context matching type with DBGBCRn.BAS!=1111	The Cortex-R52 processor implements the following option: • If DBGBCRn.BT is set to Context matching type, DBGBCRn.BAS is RES1 and ignored.
VMID (only) matching breakpoint with DBGBVRn!=`0x00000000`	The Cortex-R52 processor implements the following option: • DBGBVRn is RES0 and ignored.
Unlinked breakpoint or watchpoint type with LBN!=`0b0000`	The Cortex-R52 processor implements the following option: • The LBN field reads UNKNOWN and the value is ignored.
Reserved DBGBCRn.BAS values	The Cortex-R52 processor implements the following option: • DBGBCRn.BAS[3] and DBGBCRn.BAS[1] are read-only copies of DBGBCRn.BAS[2] and DBGBCRn.BAS[0].
Address match breakpoint match only on second halfword of an instruction	The Cortex-R52 processor implements the following option: • Does not match.
Address matching breakpoint on A32 instruction with DBGBCRn.BAS=`0b1100`	The Cortex-R52 processor implements the following option: • Does not match.
Address match breakpoint match on T32 instruction at DBGBVRn+2 with DBGBCRn.BAS=`0b1111`	The Cortex-R52 processor implements the following option: • Does not match.
Address mismatch breakpoint match only on second halfword of an instruction	The Cortex-R52 processor implements the following option: • Does not match.
Address mismatch breakpoint match on T32 instruction at DBGBVRn +2 with DBGBCRn.BAS=`0b1111`	The Cortex-R52 processor implements the following option: • Does match.
Other mismatch breakpoint matches any address in current mode and state	The Cortex-R52 processor implements the following option: • Immediate breakpoint debug event.
Mismatch breakpoint on branch to self	The Cortex-R52 processor implements the following option: • Instruction is stepped an UNKNOWN number of times, while it continues to branch to itself.

Table C-1 Armv8 Debug UNPREDICTABLE behaviors (continued)

Scenario	Behavior
Address matching breakpoint with DBGBVRn[1:0]!=`0b00`	The Cortex-R52 processor implements the following option: • If DBGBCRn.BT is set to an Address matching type, DBGBVRn[1:0] are RES0 and ignored.
Link to non-existent breakpoint or breakpoint that is not context-aware	The Cortex-R52 processor implements the following option: • No breakpoint or watchpoint debug event is generated and the LBN field of the linker reads UNKNOWN.
Link to breakpoint not configured for Linked context matching	The Cortex-R52 processor implements the following option: • If the linkee is implemented and context-aware but is either not programmed for linked context matching or not enabled, no debug event is generated for the linker.
Link to breakpoint not configured to match everywhere	The Cortex-R52 processor implements the following option: • The LBN, SSC, HMC, BAS, and PMC fields of the linkee are ignored.
DBGWCRn[2]=1 and DBGWCRn.BAS=`0b0000`	The Cortex-R52 processor implements the following option: • If DBGWCRn[2] is set to 1, then DBGWCRn.BAS[7:4] is RES0 and ignored.
DBGWCRn.MASK!=`0b00000` and DBGWCRn.BAS!=`0b11111111`	The Cortex-R52 processor implements the following option: • DBGWCRn.BAS is ignored and treated as `0b11111111`.
DBGWCRn.MASK!=`0b00000` and masked DBGWVRn bits nonzero	The Cortex-R52 processor implements the following option: • If masked bits of DBGWVRn are not zero, no watchpoint debug event is generated.
Address-matching Vector catch on 32-bit T32 instruction at (vector-2)	The Cortex-R52 processor implements the following option: • Does not match.
Address-matching Vector catch on 32-bit T32 instruction at (vector+2)	The Cortex-R52 processor implements the following option: • Does match.
Address-matching Vector catch and breakpoint on same instruction	The Cortex-R52 processor implements the following option: • Report breakpoint.
Address match breakpoint with DBGBCRn.BAS=`0b0000`	The Cortex-R52 processor implements the following option: • As if disabled.
DBGWCRn.BAS specifies a non-contiguous set of bytes within a doubleword	The Cortex-R52 processor implements the following option: • A watchpoint debug event is generated for each matching byte.
A32 `HLT` instruction with condition code not AL	The Cortex-R52 processor implements the following option: • Executed unconditionally.
Execute instruction at a given Exception level when the corresponding EDECCR bit is 1 and Halting is allowed	The Cortex-R52 processor behaves as follows: • Generates debug event and Halt no later than the instruction following the next *Context Synchronization Operation* (CSO).
Routing general exceptions to Hyp mode	The Cortex-R52 processor behaves as follows: • If HDCR.TDE is set to 1, then the HDCR.{TDA, TDRA, TDOSA} bits are treated as 1 regardless of their actual state other then for the purpose of reading the bits.
Unlinked Context matching and Address mismatch breakpoints taken to Abort mode	The Cortex-R52 processor implements the following option: • The breakpoint debug event is ignored.

Table C-1 Armv8 Debug UNPREDICTABLE behaviors (continued)

Scenario	Behavior
Vector catch on Data or Prefetch abort, and taken to Abort mode	The Cortex-R52 processor implements the following option: • A Prefetch Abort debug exception is generated. If Vector catch is enabled on the Prefetch Abort vector, this generates a Vector catch debug event. **Note** The debug event is subject to the same CONSTRAINED UNPREDICTABLE behavior, therefore the breakpoint debug event is repeatedly generated an UNKNOWN number of times.
H > N or H = 0 at EL1 and EL0, including value read from PMCR.N	The Cortex-R52 processor implements the following option: • A simple implementation where all of HPMN[4:0] are implemented, and in EL1 and EL0: — If H > N then M = N. — If H = 0 then M = 0.
H > N or H = 0: value read back in HDCR.HPMN	The Cortex-R52 processor implements: • A simple implementation where all of HPMN[4:0] are implemented and for reads of HDCR.HPMN, return H.
P ≥ M and P ≠ 31: reads and writes of PMXEVTYPER and PMXEVCNTR	The Cortex-R52 processor implements: • RAZ/WI.
P ≥ M and P ≠ 31: value read in PMSELR.SEL	The Cortex-R52 processor implements: • A simple implementation where all of SEL[4:0] are implemented, and if P ≥ M and P ≠ 31 then the register is RES0.
P = 31: reads and writes of PMXEVCNTR	The Cortex-R52 processor implements: • RAZ/WI.
n ≥ M: Direct access to PMEVCNTRn and PMEVTYPERn	The Cortex-R52 processor implements: • If n ≥ N, then the instruction is UNALLOCATED. • Otherwise if n ≥ M, then the register is RES0.
Exiting Debug state while instruction issued through EDITR is in flight	The Cortex-R52 processor implements the following option: • The instruction completes in Debug state before executing the restart.
Using memory-access mode with a non-word-aligned address	The Cortex-R52 processor implements the following option: • For each memory access the processor makes, the processor makes an unaligned access to X0/R0. If alignment checking is enabled for the access, this generates an alignment fault.
Restart request and Halt request asserted at same time	The Cortex-R52 processor implements the following option: • The restart is ignored, then the processor enters Debug state.
Access to memory-mapped registers mapped to Normal memory	The Cortex-R52 processor implements the following option: • The access generates an external abort or not.
Not word-sized accesses	The Cortex-R52 processor implements the following option: • The access generates an external abort or not.
UNPREDICTABLE CP14 register accesses	The Cortex-R52 processor implements the following option: • Access permissions are as for the equivalent system registers in AArch64 state.

Table C-1 Armv8 Debug UNPREDICTABLE behaviors (continued)

Scenario	Behavior
External debug write to register that is being reset.	The Cortex-R52 processor implements the following option: • Takes reset value.
Accessing reserved debug registers	The Cortex-R52 processor deviates from preferred behavior because the hardware cost to decode some of these addresses in Debug power domain is significantly high. The actual behavior is: **Actual behavior:** 1. For reserved debug in the range `0x000-0xCFC` and Performance Monitors registers in the address range `0x000-0xF00`, the response is either CONSTRAINED UNPREDICTABLE Error or RES0 when any of the following errors occurs: **Off** The core power domain is either completely off or in a low-power state where the core power domain registers cannot be accessed. **DLK** `DoubleLockStatus()` is TRUE, OS double-lock is locked (EDPRSR.DLK is 1). **OSLK** OS lock is locked (OSLSR.OSLK is 1). 2. For reserved debug registers in the address ranges `0x400-0x4FC` and `0x800-0x8FC`, the response is either CONSTRAINED UNPREDICTABLE error or RES0 when the conditions in *1* do not apply and the following error occurs: **EDAD** `AllowExternalDebugAccess()` is FALSE. External debug access is disabled. 3. For reserved Performance Monitor registers in the address ranges `0x000-0x0FC` and `0x400-0x47C`, the response is CONSTRAINED UNPREDICTABLE Error or RES0 when the conditions in *1* and *2* do not apply, and the following error occurs: **EPMAD** `AllowExternalPMUAccess()` is FALSE. External Performance Monitors access is disabled.
Reserved DBGBCRn,BT values	The Cortex-R52 processor implements the following option: • If the breakpoint is not context-aware, BT[3] and BT[1] are RES0. The values `0b011x` and `0b11xx` are reserved but they must behave as if the breakpoint is disabled.
Reserved combinations of DBGBCRn.{SSC,HMC,PMC} or DBGWCRn.{SSC,HMC,PAC}	The Cortex-R52 processor behaves as if it is programmed with a combination that is not reserved.
Reserved DBGWCRn.LSC values	The Cortex-R52 processor behaves as if watchpoint is disabled.

Table C-1 Armv8 Debug UNPREDICTABLE behaviors (continued)

Scenario	Behavior
Reserved DBGWCRn.MASK values	The Cortex-R52 processor behaves as if no mask is set.
Clearing the clear-after-read EDPRSR bits when core power domain is on, and `DoubleLockStatus()` is TRUE	The Cortex-R52 processor implements the following option: • Bits are not cleared to zero.

C.7 Other UNPREDICTABLE behaviors

This section describes other UNPREDICTABLE behaviors.

Table C-2 Other UNPREDICTABLE behaviors

Scenario	Description
CSSELR indicates a cache that is not implemented.	If CSSELR indicates a cache that is not implemented, then on a read of the CCSIDR the behavior is CONSTRAINED UNPREDICTABLE, and can be one of the following: • The CCSIDR read returns an UNKNOWN value (preferred).
HDCR.HPMN is set to 0, or to a value larger than PMCR.N.	If HDCR.HPMN is set to 0, then there is no access to any counters. If HDCR.HPMN is set to a value higher than the number of available counters (in Cortex-R52, this value is four), then the number of counters accessible is four. 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 that is one of: • The value that was written to HDCR.HPMN.
Write access to IMPLEMENTATION DEFINED read-only registers.	This results in an UNDEFINED exception.

Appendix D
PMC-R52

This appendix describes the *Programmable MBIST Controller* (PMC) for the Cortex-R52 processor.

It contains the following sections:

D.1 Introduction

The Cortex-R52 processor has a top-level MBIST slave interface that allows read and write access at full speed to all SRAMs in each core.

The primary purpose of MBIST is for production testing, but the Cortex-R52 processor also supports SRAM and ECC logic testing in the field during normal execution. This is known as on-line MBIST. On-line memory testing is used in functional safety applications and it is carried out during either of the following:

- On a periodic basis.
- At powerup.
- When an ECC error is detected.

Existing production test MBIST controllers do not support the requirements of on-line MBIST, therefore, it is necessary to use PMC-R52 which was specifically developed for this purpose.

PMC-R52 uses a programmable microcode-based architecture to allow a high degree of flexibility to accommodate different use models and memory test algorithms. It has a standard Arm MBIST master interface and an AHB-Lite slave interface. The primary purpose is to allow testing of the:

- SRAMs.
- ECC logic.
- Any other associated logic.

The Cortex-R52 *Software Built-In Self-Test* (SBIST) library uses PMC-R52 to test programs to:

- Periodically test the memory ECC and associated logic.
- Test SRAMs in the background while the processor is executing application code. For more information, see *D.3.3 On-line MBIST, on-line memory* on page Appx-D-632.
- Periodically test SRAMs using production test algorithms. For more information, see *D.3.2 On-line MBIST, off-line memory* on page Appx-D-632.

PMC-R52 is programmed through the AHB-Lite slave interface. Before testing a memory array, PMC-R52 must be programmed with information for the array and the list of memory transactions to be carried out. For more information on the SBIST library, see *Arm® Cortex®-R52 Processor SBIST User Guide*.

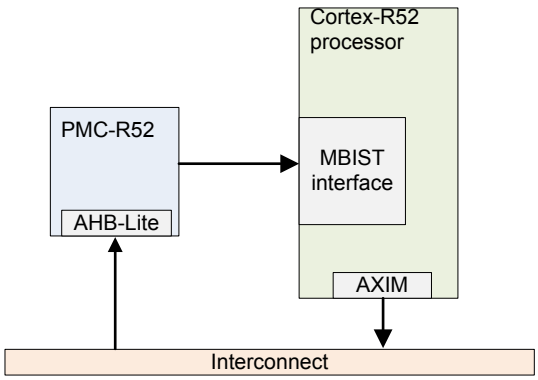

Figure D-1 Example PMC-R52 system design

D.2 PMC-R52 features and advantages

The main features of PMC-R52 are that it:

- Allows memory to be tested at-speed using back-to-back transactions at the processor clock frequency.
- Can use the short-burst transparent SRAM test algorithm, non-destructive.
- Can use the production test March SRAM test algorithm, destructive.
- Can interrupt the processor when a test has completed or if a memory fault is detected.
- Can use software polling to check if a test has completed or if a memory fault is detected.
- Has individual test bursts that can be triggered by hardware or software.
- Is microcode-based, making it highly flexible within the area constraints, so that different test algorithms can be implemented.
- Allows the processor to perform arbitrary read and write accesses to SRAM data and ECC values.

The advantages of the PMC-R52 are:

- Detection of latent faults in SRAMs, including timing faults.
- Detection of latent faults in memory ECC logic, including timing faults.
- ECC error analysis. This allows testing an SRAM entry when ECC error detected to determine if a hard or soft error has occurred. If it is a soft error, then there is no need to use the memory error cache register in the core to replace entry. If it a hard error, then the fault must be corrected. To do this, use error cache register to disable entry or SRAM correction features.
- SRAM retention testing.
- Memory error injection for error monitoring software and system hardware verification.
- Memory scrubbing to correct soft ECC errors in SRAM. This prevents error accumulation and ensures that correctable single-bit errors do not degenerate into multi-bit errors, which cannot be corrected.
- Memory dumping and monitoring by an external debugger. This is useful for software and hardware debug, silicon bring-up, diagnosing SRAM power issues, and core lockups.

D.3 MBIST usage models

The Cortex-R52 processor supports the following MBIST usage models:

- Production MBIST.
- On-line MBIST, off-line memory.
- On-line MBIST, on-line memory.

——— **Note** ———

PMC-R52 supports the on-line MBIST, off-line memory and on-line MBIST, on-line memory usage models. However, it does not support the production MBIST usage model. In this manual, on-line MBIST, on-line memory usage model is primarily used.

———————————

D.3.1 Production MBIST

The production MBIST usage model is when the SRAM testing is carried out primarily during manufacturing of the device, but might also be carried out during powerup, or periodically in the field.

Production MBIST mode is entered by resetting the processor and asserting the MBIST request signal at the same time. When reset is deasserted, the processor is immediately halted and is held in this state for the duration of the testing. This allows full access to all entries of all SRAMs. SRAMs are tested by different test algorithms that target different types of faults. The processor must be reset after the testing has been completed. The testing destroys the memory contents and the state of the processor. If the memory contents or processor state must be preserved after testing, then software must be used to save and restore the current state of the processor and memories.

D.3.2 On-line MBIST, off-line memory

In the on-line MBIST, off-line memory usage model, the memory under test is taken off-line, therefore, preventing software from accessing it. This allows access to all locations during testing. The following steps are carried out for each test:

1. Software disables the memory under test so the memory can be used for testing, but the processor can still execute from other memories with a possible degradation in performance.
2. If the memory contents must be preserved, then software should save the contents of the memory under test in a memory that is not being tested.
3. Software instructs the MBIST controller to test the memory.
4. The MBIST controller tests the memory.
5. When the test is complete, software checks the MBIST controller for errors.
6. Depending on your requirements and type of memory being tested, the software must do either of the following:
   - Restore the contents of the memory.
   - If a cache is being tested, then it must be invalidated.
7. Software enables the memory for functional use again.

This usage model allows standard production MBIST March algorithms to be used, which requires access to all entries within the memory under test. Memory contents are destroyed during testing and software might need to save and restore these memory contents. The processor cannot access a memory under test because it has been disabled. Therefore, an alternate memory is accessed instead. For example, if software performs an access in the TCM range, it accesses the main memory through the cache. The memory must be taken offline for some time to allow the test algorithm to be carried out. Software must be carefully written to ensure that it does not rely on data that is normally stored in the memory under test.

D.3.3 On-line MBIST, on-line memory

In the on-line MBIST, on-line memory usage model, periodic autonomous or software initiated short burst testing is performed.

For more information, see *D.4 Short-burst software transparent algorithm* on page Appx-D-634. In this case, a series of short sequences of transactions or bursts that are applied separately tests the memory. Typically, each burst tests two locations. Each burst lasts less than 20 clock cycles and targets different locations, therefore, allowing all locations in a memory to be tested. During each burst, the MBIST controller saves and restores the memory locations under test. Software is only required to initialize the MBIST controller and deal with any errors detected. Once the MBIST controller is initialized, it performs each short burst and steps through all locations in a RAM without processor intervention. The MBIST controller and on-line MBIST logic in the processor carries out the following steps automatically for each burst:

1. The MBIST controller requests access to the target memory.
2. When the processor is ready, it acknowledges the request and automatically locks the memory for normal accesses. This guarantees full speed MBIST access and guarantees that no changes are made to the targeted memory by software during test.
3. The MBIST controller performs a short burst of transactions.
4. The MBIST controller releases the request.
5. The lock on the targeted memory is then automatically removed.
6. Only one memory is locked for processor access during a burst, and therefore, the processor can access other memories. If the processor tries to access a locked memory, it stalls until the burst is complete. This usage model only has a small impact on performance if the processor tries to access a memory under test because bursts are short and occur infrequently.

D.4 Short-burst software transparent algorithm

On-line MBIST testing uses the same data paths as production MBIST testing. Therefore, RAM testing can be of high quality with the on-line MBIST, on-line memory usage model. During manufacture, devices already pass production MBIST testing. Therefore, the most common memory faults that occur in an operating system are different to those that occur in manufacturing. Also, production MBIST tests can run at powerup and powerdown. This provides the opportunity to use new test algorithms that target the memory faults that are most likely to occur in the field and that have a minimal impact on the performance of the processor.

This section contains the following subsections:

D.4.1 SRAM faults

The short-burst software transparent MBIST algorithm covers delay and stuck-at faults because of transistor aging and electromigration in the following SRAM circuits:

- Individual bit cells.
- Word lines.
- Timing circuits.
- Sense amplifiers.
- Data multiplexors.

D.4.2 Example short-burst software transparent on-line MBIST test algorithm

The following table demonstrates an example short-burst software transparent on-line MBIST test algorithm.

The example test algorithm detects the fault types that are listed in *D.4.1 SRAM faults* on page Appx-D-634. This algorithm is repeated for all values of `n`, where `n` is an SRAM entry under test and it is in the range, 0 to SRAM size-1. The algorithm uses an address offset `m`, where `m` is the constant RAM column mux factor for the array under test (4, 8, 16, or 32).

Table D-1 Example short-burst software transparent on-line MBIST test algorithm

Algorithm step	Operation	Description
1	Read location `n` and store in register X.	Location `n` is saved.
2	Read location `n+m` and store in register Y.	Location `n+m` saved and the next word line is activated.
3	Write X to location `n+m`	-
4	Write ~X to location `n`.	Switch between word lines and all bit lines.
5	Read location `n` and check that it is equal to ~X.	Verification that no bit is stuck in location `n`.
6	Read location `n+m` and check that it is equal to X.	Switch between word lines and all bit lines.
7	Write X to location `n`.	Location `n` is restored.
8	Write Y to location `n+m`.	Location `n+m` is restored.
9	Read location `n` and check that is it equal to X.	Checks that the X register is restored correctly.
10	Read location `n+m` and check that is it equal to Y.	Checks that the Y register is restored correctly.

For more information on X and Y registers, see *D.8.11 Data registers, X0-X7 and Y0-Y7* on page Appx-D-666.

D.4.3 Microcode

This section shows the microcode for an example short-burst software transparent on-line MBIST algorithm.

The following table shows the PMC-R52 microcode that implements the test algorithm in *D.4.2 Example short-burst software transparent on-line MBIST test algorithm* on page Appx-D-634. The microcode assumes that PMC-R52 is configured in the x-fast address update mode, which changes the row address (word line) every time it is executed. For more information on x-fast address update mode, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650(ADDRCD bit). For more information on the microcode format, see *D.8.15 Program registers, P0-P31* on page Appx-D-670.

Table D-2 Microcode for example short-burst software transparent on-line MBIST test algorithm

Register	Instruction						Output address	Address update	Data polarity	Data register	Transaction	Operation
P0	0	0	0	0	01	0010	Address.	Hold address	-	X	Read	Save read data.
P1	1	0	0	1	01	0010	Next address	Hold address	-	Y	Read	Save read data.
P2	1	0	0	0	10	0011	Next address	Hold address	No inversion	X	Write	Wait for register update.
P3	0	0	1	0	10	0000	Address	Hold address	Inversion	X	Write	No operation.
P4	0	0	1	0	01	0000	Address	Hold address	Inversion	X	Read	No operation, check data.
P5	1	0	0	0	01	0000	Next address	Hold address	No inversion	X	Read	No operation, check data.
P6	0	0	0	0	10	0000	Address	Hold address	No inversion	X	Write	No operation.
P7	1	0	0	1	10	0000	Next address	Hold address	No inversion	Y	Write	No operation.
P8	0	0	0	0	01	0000	Address	Hold address	No inversion	X	Read	No operation, check data.
P9	1	1	0	1	01	0001	Next address	Update address	No inversion	Y	Read	Last operation, check data.

Note

For more information on how the output address is calculated, see *Address output value* on page Appx-D-665.

D.5 Production test March algorithm

March MBIST algorithms are normally used in Production testing, but they might also be used for in-field SRAM testing at powerup or periodically during operation. PMC-R52 can execute these test algorithms with some additional software assistance.

March MBIST algorithms destroy the memory contents, therefore, they can only be run using the on-line MBIST, off-line memory use model. For more information on the on-line MBIST, off-line memory use model, see *D.3.2 On-line MBIST, off-line memory* on page Appx-D-632.

This section describes how to program PMC-R52 to perform an example March MBIST algorithm called March C-. This information can be used as a foundation to implement other Production test MBIST algorithms.

This section contains the following subsections:

D.5.1 March algorithms

A March MBIST algorithm tests an SRAM by filling all its entry test patterns. The algorithm also carries out several passes through an SRAM checking the patterns and writing new patterns. The SRAM read and write operations performed on each pass is called a March element and each element is repeated for each entry in an SRAM. The direction the address is incremented or decremented for each entry is indicated in the March notation.

March notation

March algorithms are described using the notation that is shown in the following table.

Table D-3 March notation

Notation	Description
()	March element. This contains read and write operations to be carried out on each SRAM entry.
r0	Read SRAM and check that the value is equal to pattern 0.
r1	Read SRAM and check that the value is equal to pattern 1.
w0	Write pattern 0 to SRAM.
w1	Write pattern 1 to SRAM.
⇑	Increment address after the operations in a March element are carried out.
⇓	Decrement address after the operations in a March element are carried out.

Pattern 0 and 1 can be any value, but they must be the inverse of each other.

The operations in a March element are performed on the current SRAM address, then the address is incremented or decremented as specified and the operations are repeated for the new address. This is repeated for all entries in an SRAM. Subsequently, the next March element is executed for all SRAM entries. When all elements have been executed the test ends.

D.5.2 March C- algorithm

The March C- MBIST algorithm has a high coverage of SRAM faults, including address decoder and coupling faults and it only requires ten accesses for each SRAM entry.

The following figure shows the March C- algorithm:

{⇑ (w0); ⇑ (r0, w1); ⇑ (r1, w0); ⇓ (r0, w1); ⇓ (r0, w1); ⇓ (r0)}

D.5.3 PMC-R52 programming for March C- algorithm

At the beginning of the test, PMC-R52 must be programmed with the attributes for the memory array under test. These attributes include, the:

- Array encoding.
- Mux factor.
- Data mask.
- Pipeline depth.
- Cycles per operation.

This attribute information does not need to be updated during the test. However, the microcode and address settings must be programmed with the information for the March element before each SRAM pass. Arm recommends that the X register is filled across its full width with only zeros and this is used as test pattern 0. For more information on the X register, see *D.8.11 Data registers, X0-X7 and Y0-Y7* on page Appx-D-666.

PMC-R52 must also be programmed to continuously loop the microcode program until the end address is reached. PMC-R52 can be programmed to interrupt the processor when a pass is complete or if a fault is detected. Alternatively, software can poll PMC-R52 to see when a pass is complete or if a fault is detected. The programming indicated in *Initial programming* on page Appx-D-637 demonstrates software that polls PMC-R52 to see when a pass is complete or if a fault is detected.

After PMC-R52 is programmed for a pass, it must be enabled to execute the microcode program by setting the CTRL.PEEN bit to `0b1`. For more information on CTRL, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.

The PMC-R52 programming for each pass is described in:

- *Pass 0* on page Appx-D-638.
- *Pass 1* on page Appx-D-639.
- *Pass 2* on page Appx-D-640.
- *Pass 3* on page Appx-D-641.
- *Pass 4* on page Appx-D-642.
- *Pass 5* on page Appx-D-642.

Note

The number of elements in the SRAM under test is N.

Initial programming

At the beginning of the test, the registers must be programmed as described in the following table.

For more information on the programmer's model, see *D.8 Programmers model* on page Appx-D-648.

Table D-4 Initial programming

Register	Programming
CTRL	`00000210` • X-fast, test end, and fault interrupts disabled. • Stop on failure. • **TC** input ignored (loop until stop address). For more information on CTRL, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650
MCR	Attributes for SRAM under test. For more information on MCR, see *D.8.4 Memory Control Register, MCR* on page Appx-D-657.
BER	`0xFFFFFFFF` For more information on BER, see *D.8.5 Byte Enable Register, BER* on page Appx-D-659.
PCR	`0x00000000` For more information on PCR, see *D.8.6 Program Counter Register, PCR* on page Appx-D-660.
X0-X7	All zero. For more information on X registers, see *D.8.11 Data registers, X0-X7 and Y0-Y7* on page Appx-D-666.
Y0-Y7	All zero. For more information on Y register, see *D.8.11 Data registers, X0-X7 and Y0-Y7* on page Appx-D-666.
P0-P31	All zero. For more information on P registers, see *D.8.15 Program registers, P0-P31* on page Appx-D-670.
DM0-DM7	Data mask for SRAM under test. For more information on DM registers, see *D.8.14 Data Mask Registers, DM0-DM7* on page Appx-D-669.
AOR	As required. For more information on AOR, see *D.8.13 Auxiliary Output Register, AOR* on page Appx-D-668.

Pass 0

The additional programming needed for Pass 0 is described in the following tables.

The number of elements in the SRAM under test is N.

The March element is:

⇑ (w0)

The pass-specific programming for Pass 0 is described in the following table.

Table D-5 Pass-specific programming

Register	Programming
STOPADDR	N-1 For more information on STOPADDR, see *D.8.8 Stop Address Register, STOPADDR* on page Appx-D-662.
CADDR	`0x00000000` For more information on CADDR, see *D.8.9 Column Address Register, CADDR* on page Appx-D-663.
RADDR	`0x00000000` For more information on RADDR, see *D.8.10 Row Address Register, RADDR* on page Appx-D-664.
CTRL	**`0x00000610`** Increment address. For more information on CTRL register, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
PCR	`0x00000000` For more information on PCR, see *D.8.6 Program Counter Register, PCR* on page Appx-D-660.

The microcode programming for Pass 0 is described in the following table.

Table D-6 Microcode programming

Register	Instruction						Output address	Address update	Data polarity	Data register	Transaction	Operation
P0	0	1	0	0	10	0001	Address	Update address	No inversion	X	Write	Last operation.

Pass 1

The additional programming needed for Pass 1 is described in the following tables.

The number of elements in the SRAM under test is N.

The March element is:

⇑ (r0, w1)

The pass-specific programming for Pass 1 is described in the following table.

Table D-7 Pass-specific programming

Register	Programming
STOPADDR	N-1 For more information on STOPADDR, see *D.8.8 Stop Address Register, STOPADDR* on page Appx-D-662.
CADDR	`0x00000000` For more information on CADDR, see *D.8.9 Column Address Register, CADDR* on page Appx-D-663.
RADDR	`0x00000000` For more information on RADDR, see *D.8.10 Row Address Register, RADDR* on page Appx-D-664.

Table D-7 Pass-specific programming (continued)

Register	Programming
CTRL	**0x00000610** Increment address. For more information on CTRL register, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
PCR	0x00000000 For more information on PCR, see *D.8.6 Program Counter Register, PCR* on page Appx-D-660.

The microcode programming for Pass 1 is described in the following table.

Table D-8 Microcode programming

Register	Instruction						Output address	Address update	Data polarity	Data register	Transaction	Operation
P0	0	0	0	0	01	0000	Address	Hold address	No inversion	X	Read	None, check data.
P1	0	1	1	0	10	0001	Address	Update address	Inversion	X	Write	Last operation.

Pass 2

The additional programming needed for Pass 2 is described in the following tables.

The number of elements in the SRAM under test is N.

The March element is:

⇑ (r1, w0)

The pass-specific programming for Pass 2 is described in the following table.

Table D-9 Pass-specific programming

Register	Programming
STOPADDR	N-1 For more information on STOPADDR, see *D.8.8 Stop Address Register, STOPADDR* on page Appx-D-662.
CADDR	0x00000000 For more information on CADDR, see *D.8.9 Column Address Register, CADDR* on page Appx-D-663.
RADDR	0x00000000 For more information on RADDR, see *D.8.10 Row Address Register, RADDR* on page Appx-D-664.
CTRL	**0x00000610** Increment address. For more information on CTRL register, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
PCR	0x00000000 For more information on PCR, see *D.8.6 Program Counter Register, PCR* on page Appx-D-660.

The microcode programming for Pass 2 is described in the following table.

Table D-10 Microcode programming

Register	Instruction						Output address	Address update	Data polarity	Data register	Transaction	Operation
P0	0	0	1	0	01	0000	Address	Hold address	Inversion	X	Read	None, check data.
P1	0	1	0	0	10	0001	Address	Update address	No inversion	X	Write	Last operation.

Pass 3

The additional programming needed for Pass 3 is described in the following tables.

The number of elements in the SRAM under test is N.

The March element is:

⇓ (r0, w1)

The pass-specific programming for Pass 3 is described in the following table.

Table D-11 Pass-specific programming

Register	Programming
STOPADDR	`0x00000000` For more information on STOPADDR, see *D.8.8 Stop Address Register, STOPADDR* on page Appx-D-662.
CADDR	N-1 **Note** Only *Least Significant Bits* (LSBs) are used ([MCR.CCW+1:0]). For more information on CADDR, see *D.8.9 Column Address Register, CADDR* on page Appx-D-663.
RADDR	N-1 >> MCR.CCW+1 For more information on RADDR, see *D.8.10 Row Address Register, RADDR* on page Appx-D-664.
CTRL	**`00000210`** Decrement address. For more information on CTRL register, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
PCR	`0x00000000` For more information on PCR, see *D.8.6 Program Counter Register, PCR* on page Appx-D-660.

The microcode programming for Pass 3 is described in the following table.

Table D-12 Microcode programming

Register	Instruction						Output address	Address update	Data polarity	Data register	Transaction	Operation
P0	0	0	0	0	01	0000	Address	Hold address	No inversion	X	Read	None, check data.
P1	0	1	1	0	10	0001	Address	Update address	Inversion	X	Write	Last operation.

Pass 4

The additional programming needed for Pass 4 is described in the following tables.

The number of elements in the SRAM under test is N.

The March element is:

⇓ (r1, w0)

The pass-specific programming for Pass 4 is described in the following table.

Table D-13 Pass-specific programming

Register	Programming
STOPADDR	`0x00000000` For more information on STOPADDR, see *D.8.8 Stop Address Register, STOPADDR* on page Appx-D-662.
CADDR	N-1 **Note** Only *Least Significant Bits* (LSBs) are used ([MCR.CCW+1:0]). For more information on CADDR, see *D.8.9 Column Address Register, CADDR* on page Appx-D-663.
RADDR	N-1 >> MCR.CCW+1 For more information on RADDR, see *D.8.10 Row Address Register, RADDR* on page Appx-D-664.
CTRL	**`00000210`** Decrement address. For more information on CTRL register, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
PCR	`0x00000000` For more information on PCR, see *D.8.6 Program Counter Register, PCR* on page Appx-D-660.

The microcode programming for Pass 4 is described in the following table.

Table D-14 Microcode programming

Register	Instruction						Output address	Address update	Data polarity	Data register	Transaction	Operation
P0	0	0	1	0	01	0000	Address	Hold address	Inversion	X	Read	None, check data.
P1	0	1	0	0	10	0001	Address	Update address	No inversion	X	Write	Last operation.

Pass 5

The additional programming needed for Pass 5 is described in the following tables.

The number of elements in the SRAM under test is N.

The March element is:

⇓ (r0)

The pass-specific programming for Pass 5 is described in the following table.

Table D-15 Pass-specific programming

Register	Programming
STOPADDR	`0x00000000` For more information on STOPADDR, see *D.8.8 Stop Address Register, STOPADDR* on page Appx-D-662.
CADDR	N-1 **Note** Only *Least Significant Bits* (LSBs) are used ([MCR.CCW+1:0]). For more information on CADDR, see *D.8.9 Column Address Register, CADDR* on page Appx-D-663.
RADDR	N-1 >> MCR.CCW+1 For more information on RADDR, see *D.8.10 Row Address Register, RADDR* on page Appx-D-664.
CTRL	**`00000210`** Decrement address. For more information on CTRL register, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
PCR	`0x00000000` For more information on PCR, see *D.8.6 Program Counter Register, PCR* on page Appx-D-660.

The microcode programming for Pass 5 is described in the following table.

Table D-16 Microcode programming

Register	Instruction						Output address	Address update	Data polarity	Data register	Transaction	Operation
P0	0	1	0	0	01	0001	Address	Update address	No inversion	X	Read	Last operation, check data.

D.6 Example test code

Example test code is provided which implements the following algorithms:

- *D.4 Short-burst software transparent algorithm* on page Appx-D-634.
- *D.5 Production test March algorithm* on page Appx-D-636.

The example test code is implemented in assembly code and can be executed under the Cortex-R52 SBIST library environment. For more information on the SBIST library, see *Arm® Cortex®-R52 Processor SBIST User Guide*.The example test code contain the following source files:

Table D-17 Example test code

Source file name	Description
`kite_sbist_ram.s`	Test wrapper which calls the test functions in sequence.
`kite_sbist_ram_save_port.s`	Architectural state save function that is called before executing the test.
`kite_sbist_ram_restore_port.s`	Architectural state restore function that is called after executing the test.
`kite_sbist_ram_functions_common_port.s`	Functions that the test requires.
`kite_sbist_ram_p001_n001_port.s`	Test part 1 (March algorithm on L1 data cache data RAM).
`kite_sbist_ram_p002_n001_port.s`	Test part 2 (Short-burst software transparent algorithm on ATCM).

The two test parts, `kite_sbist_ram_p001_n001_port.s` and `kite_sbist_ram_p002_n001_port.s`, implement:

- The production test March algorithm for testing the L1 data cache data RAM configured with a 32KB RAM.
- The short-burst software transparent algorithm for testing ATCM configured with a 32KB RAM.

To execute, the example test requires the Cortex-R52 processor to be configured with at least 16 EL2-controlled MPU regions.

D.7 Functional description

PMC-R52 contains the following main blocks:

AHB slave interface
The AHB slave interface provides processor read and write access to the register store.

Register store
The register store contains the following registers:

- Memory configuration.
- Control.
- Address.
- Data.
- Program.
- ID.

Execution unit
The execution unit uses the information stored in the registers to generate transactions on the MBIST interface and checks read data returned from the selected memory array.

MBIST master interface
The MBIST master interface contains the MBIST entry and exit state machine and other interface logic.

PMC-R52 occupies 4KB in the processor memory map. For more information on the programmer's model, see *D.8 Programmers model* on page Appx-D-648.

As shown in *Figure D-2 PMC-R52 block diagram* on page Appx-D-645, the processor can access the registers to configure and initialize PMC-R52 but it cannot access the execution unit. Therefore, the processor is decoupled from the MBIST interface and has to set up memory transactions in the program registers in the register store and wait for the execution unit to carry them out. PMC-R52 can be configured to interrupt the processor or set a flag when it has carried out the transactions configured by the processor or when an error is detected.

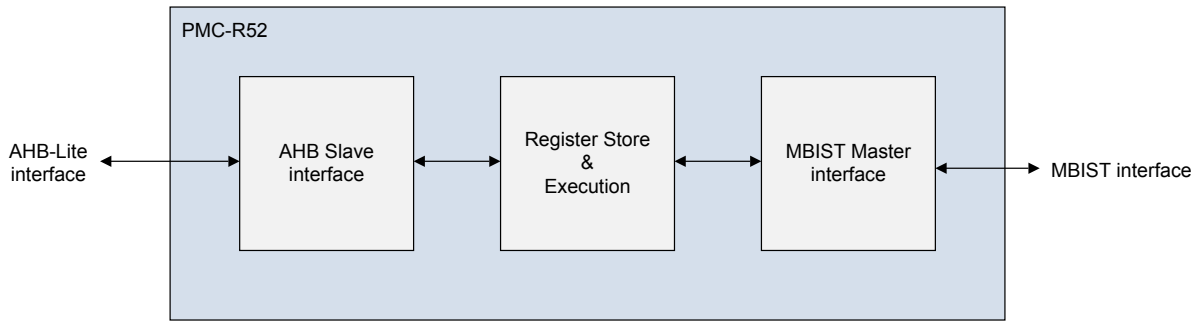

Figure D-2 PMC-R52 block diagram

This section contains the following subsection:

- *D.7.1 Internal RTL constants* on page Appx-D-645.

D.7.1 Internal RTL constants

The following table shows the internal RTL constant values that PMC-R52 uses.

The values described in *Table D-18 Internal RTL constants* on page Appx-D-646 are visible in the programmer's model.

Table D-18 Internal RTL constants

Name	Value	Description
MAWIDTH[5:0]	17	MBIST address width. • For more information, see **MBISTADDR** signal in the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*. • For more information, see RADDR in *D.8.10 Row Address Register, RADDR* on page Appx-D-664. • For more information, see STOPADDR in *D.8.8 Stop Address Register, STOPADDR* on page Appx-D-662.
MDWIDTH[8:0]	78	MBIST data width. • For more information, see **MBISTINDATA** and **MBISTOUTDATA** signals in the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*. • For more information, see X and Y registers in *D.8.11 Data registers, X0-X7 and Y0-Y7* on page Appx-D-666. • For more information, see DM registers in *D.8.14 Data Mask Registers, DM0-DM7* on page Appx-D-669.
MARWIDTH[3:0]	5	MBIST array width. • For more information, see **MBISTARRAY** signal in the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*. • For more information, see MCR.ARRAY field in *D.8.4 Memory Control Register, MCR* on page Appx-D-657.
MBWIDTH[4:0]	10	MBIST byte enable width. • For more information, see **MBISTBE** signal in the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*. • For more information, see BER.BE field in *D.8.5 Byte Enable Register, BER* on page Appx-D-659.

Table D-18 Internal RTL constants (continued)

Name	Value	Description
MCWIDTH[4:0]	2	MBIST configuration width. • For more information, see **MBISTCFG** signal in the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*. • For more information, see CTRL.CFG field in *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
PROGSIZE[5:0]	16	Program size. This is the number of program registers. • For more information, see *D.8.15 Program registers, P0-P31* on page Appx-D-670. • For more information, see *D.8.6 Program Counter Register, PCR* on page Appx-D-660.
PDWIDTH[2:0]	4	Pipeline depth field width, MCR.PD. For more information, see *D.8.4 Memory Control Register, MCR* on page Appx-D-657.
RCOWIDTH[2:0]	2	RAM cycles of operation field width, MCR.RCO. For more information, see *D.8.4 Memory Control Register, MCR* on page Appx-D-657.
AIWIDTH[5:0]	2	AIR register and **AUXIN** signal width. For more information, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*.
AOWIDTH[5:0]	2	AOR register and **AUXOUT** signal width. For more information, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*.

Note

- These values are fixed and you must not change them.
- The PMC-R52 RTL must not be modified.

D.8 Programmers model

This section describes the PMC-R52 programmers model.

This section contains the following subsections:

D.8.1 Memory map

The following figure shows the PMC-R52 memory map. The memory map occupies 4KB. It contains CoreSight and control registers.

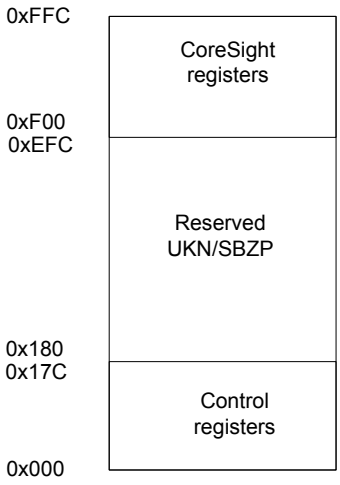

Figure D-3 PMC-R52 memory map

Register access

- The CoreSight registers are *read-only* (RO), except CLAIMSET, CLAIMCLR, and LAR.
- The control registers are *read/write* (RW).

Additionally, the following information applies to register access:

- Do not attempt access reserved or unused address locations. Attempting to access these locations can result in UNPREDICTABLE behavior.
- Unless explicitly stated:
  - Do not modify UNDEFINED register bits.
  - Ignore UNDEFINED register bits on reads.

  - The reset signal resets register bits are reset to a logic 0.
  - The software can write implemented, unreserved bits and fields to any value.
- The access types are used in this section are:
  - RW.
  - RO.
  - Write-only (WO).
  - UNKNOWN for reads (UNK).
  - Should be zero or preserve for writes (SBZP).
  - Read as zero (RAZ).
  - Read as one (RAO).
  - Write ignored (WI).

Reserved register bits are implemented as RAZ/WI and software must use them as UNK/SBZP. This approach minimizes the effect on software if new register bits are added in future PMC-R52 revisions.

The following rules apply to register access:

- Only 32-bit accesses are supported. The effect of accesses smaller than this size are that an ERROR response is returned.
- Only privileged accesses are supported. To hide potentially sensitive data from user mode code, the effect of non-privileged accesses is as follows:
  - Read accesses to CoreSight registers that return the register value as Normal and an OKAY response is returned.
  - Read accesses to control registers return a data value of 0 and an ERROR response.
  - Write accesses to the CoreSight registers are ignored and an OKAY response is returned.
  - Write accesses to control registers are ignored and an ERROR response is returned.
- Behavior is UNPREDICTABLE if control registers are written to when CTRL.PEEN is 1, except for writes to the CTRL register. Therefore, the control registers must not be written to when CTRL.PEEN is 1. The only exception is to clear the PEEN bit, which stops execution, or to clear the PES bit, which resumes execution.
- It is not expected to be useful to read any control registers when execution is enabled, other than the CTRL register to poll PMC-R52 to check if a test has successfully completed or failed.

D.8.2 Register summary

The following table lists the PMC-R52 registers with their offset in the 4KB memory region.

Table D-19 PMC-R52 register summary

Offset	Type	Name	Description
0x000	RW	CTRL	Main control register, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
0x004	RW	MCR	Memory control register, see *D.8.4 Memory Control Register, MCR* on page Appx-D-657.
0x008	RW	BER	Byte enable register, see *D.8.5 Byte Enable Register, BER* on page Appx-D-659.
0x00C	RW	PCR	Program control register, see *D.8.6 Program Counter Register, PCR* on page Appx-D-660.
0x010	RO	RPR	Read pipeline register, see *D.8.7 Read Pipeline Register, RPR* on page Appx-D-661.
0x014	RW	STOPADDR	Stop address register, see *D.8.8 Stop Address Register, STOPADDR* on page Appx-D-662.
0x018	RW	CADDR	Column address register, see *D.8.9 Column Address Register, CADDR* on page Appx-D-663.
0x01C	RW	RADDR	Row address register, see *D.8.10 Row Address Register, RADDR* on page Appx-D-664
0x020-0x03C	RW	X0-X7	Data register X, see *D.8.11 Data registers, X0-X7 and Y0-Y7* on page Appx-D-666

Table D-19 PMC-R52 register summary (continued)

Offset	Type	Name	Description
0x040-0x05C	RW	Y0-X7	Data register Y, see *D.8.11 Data registers, X0-X7 and Y0-Y7* on page Appx-D-666
0x060	RW	AIR	Auxiliary input register, see *D.8.12 Auxiliary Input Register, AIR* on page Appx-D-667.
0x064	RW	AOR	Auxiliary output register, see *D.8.13 Auxiliary Output Register, AOR* on page Appx-D-668.
0x068-0x07C	UNK/SBZP	-	Reserved.
0x080-0x09C	RW	DM0-DM7	Data mask register, see *D.8.14 Data Mask Registers, DM0-DM7* on page Appx-D-669.
0x0A0-0x0FC	UNK/SBZP	-	Reserved.
0x100-0x17C	RW	P0-P31	Program registers, see *D.8.15 Program registers, P0-P31* on page Appx-D-670.
0x180-0xEFC	UNK/SBZP	-	Reserved.
0xF00-0xFFC	-	-	CoreSight registers, see *D.8.16 CoreSight registers* on page Appx-D-672.

D.8.3 Main Control Register, CTRL

The CTRL register contains the main test control and status bits.

Usage constraints

This register is RW.

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

CTRL is a 32-bit register.

The following figure shows the CTRL bit assignments.

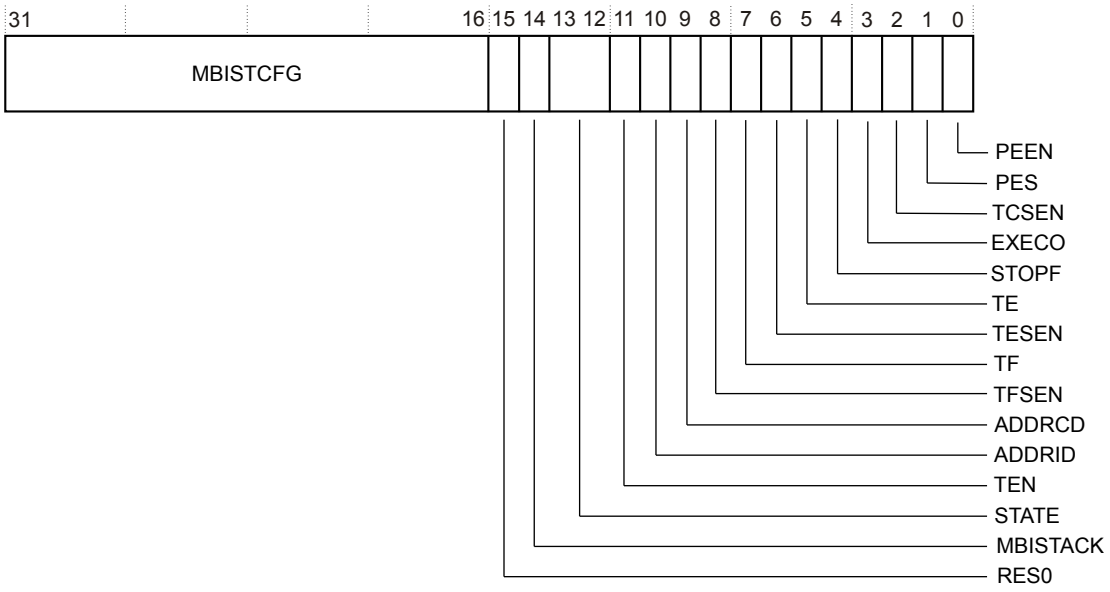

Figure D-4 CTRL bit assignments

The following table describes the CTRL bit assignments.

Table D-20 CTRL bit assignments

Bits	Name	Function
[31:16]	MBISTCFG	This is the **MBISTCFG** signal value. **Note** • This value is only used in production test. This field must always be written with 0 because PMC-R52 does not support the production test use model. • Unused field bits are reserved and must be treated as UNK/SBZP. • `MCWIDTH` sets the MBISTCFG field width. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646. The reset value is `0x0000`. This field is RW.
[15]	-	Reserved, RES0. The reset value is UNKNOWN. This field is UNK/SBZP.
[14]	MBISTACK	This is the **MBISTACK** signal value. There is no reset value. This field is RO. For more information on **MBISTACK**, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*.
[13:12]	STATE	This is the status of the PMC-R52 state machine. The status encodings are as follows: `0b00` Initial. `0b01` Run. `0b10` Suspended. `0b11` Reserved. The reset value is `0b0`. This field is RO. For more information, see *State machine* on page Appx-D-654.
[11]	TEN	Test enable. This bit is the value of the **TEN** signal at reset. When the TEN bit is 0, PMC-R52 is disabled and cannot be programmed. This field is RO. For more information on **TEN**, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*.
[10]	ADDRID	Address increment or decrement control. This bit affects the next RADDR and CADDR address register values. The options are: `0b0` Address registers are decremented. `0b1` Address registers are incremented. The address registers are updated with the next value when an instruction is executed with a UA bit value of 1. For more information on the UA bit, see *D.8.15 Program registers, P0-P31* on page Appx-D-670. The reset value is `0x0`. This field is RW.

Table D-20 CTRL bit assignments (continued)

Bits	Name	Function
[9]	ADDRCD	Address change direction. This bit controls the direction that address changes are made with respect to a memory array. It affects the next RADDR and CADDR address register values as follows: `0b0` y-fast. CADDR is changed first. All CADDR values are accessed before RADDR is changed. `0b1` x-fast. RADDR is changed first. All RADDR values are accessed before CADDR is changed. The address registers are updated with the next value when an instruction is executed with a UA bit value of 1. For more information on the UA bit, see *D.8.15 Program registers, P0-P31* on page Appx-D-670. The reset value is `0x0`. This field is RW.
[8]	TFSEN	Test failed signal enable. Controls the behavior of the **TF** signal. The options are: `0b0` **TF** signal is held LOW. `0b1` **TF** signal is equal to the value of the TF bit. The reset value is `0x0`. This field is RW. For more information on **TF**, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*.
[7]	TF	Test failed status bit. The possible values are: `0b0` Test has not failed. `0b1` Test has failed. TF is automatically set to 1, if a data comparison in the test program fails. TF can only be cleared to 0 by software. The reset value is `0x0`. This field is RW.
[6]	TESEN	Test ended signal enable. Controls the behavior of the TE signal. The options are: `0b0` **TE** signal is held LOW. `0b1` **TE** signal is equal to the value of the TE bit. The reset value is `0x0`. This field is RW. For more information on **TE**, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*.
[5]	TE	Test ended status bit. The possible values are: `0b0` Test has not completed. `0b1` Test has completed. TE is automatically set to 1 when the execution stop event occurs. For more information on the stop event, see *Stop event* on page Appx-D-656. TE can only be cleared to 0 by software. The reset value is `0x0`. This field is RW.

Table D-20 CTRL bit assignments (continued)

Bits	Name	Function
[4]	STOPF	Stop on failure. This bit causes execution to stop when a failure is detected. The possible values are: `0b0` Stop on failure mode is disabled. `0b1` Stop on failure is enabled. When STOPF is 1 and a data check in the test program fails, the PEEN bit is automatically cleared to 0 and the TF bit is set to 1, halting execution at the current instruction. The TE bit is not changed in this case. As data checks occur concurrently with program execution, the value of the PC will depend on the instructions that follow the failing read and the value of the MCR.PD value. For more information on MCR, see *D.8.4 Memory Control Register, MCR* on page Appx-D-657. The RPR register can be used to determine which read transaction failed. For more information on RPR, see *D.8.7 Read Pipeline Register, RPR* on page Appx-D-661. The reset value is `0x0`. This field is RW.
[3]	EXECO	Execute once. The options are: `0b0` Execute once mode is disabled. `0b1` Execute once mode is enabled. If EXECO is 1 and an instruction is executed with a LASTOP OP field value, the PEEN bit is automatically cleared to 0 and the TE bit is set to 1. Therefore, this causes the instructions to be executed only once. The reset value is `0x0`. This field is RW.
[2]	TCSEN	Test continue signal enable. The options are: `0b0` **TC** signal is ignored. `0b1` Enable **TC** signal. When TCSEN is 0, test suspend is disabled and the test loops back to instruction 0 when an instruction is executed with a LASTOP OP field value, until a stop event occurs. For more information, see *Resume event* on page Appx-D-656. The reset value is `0x0`. This field is RW. For more information on **TC**, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*.

Table D-20 CTRL bit assignments (continued)

Bits	Name	Function
[1]	PES	Program execution suspended. This bit operates with the TCSEN bit. The options are: `0b0` Program execution can take place. `0b1` Suspended, program execution does not take place. PES is automatically set to 1 when an instruction is executed with an OP field value of LASTOP, the TCSEN bit is 1, and the EXECO bit is 0. PES is automatically cleared to 0 when the STATE field value is Suspended, the **TC** signal is HIGH and the TCSEN bit is 1. It is also possible for software to trigger execution when the STATE field value is Suspended, by clearing the PES bit to 0. The TCSEN bit must still be set to 1, but the **TC** signal must be held LOW. In this case, PEEN is also 1,. The reset value is `0x0`. This field is RW. **Note** • Software must not write a 1 to PES, if it currently 0. • Software must write 0 to PES if it writes 0 to PEEN.
[0]	PEEN	Program execution enable. . This is the main execution enable bit. The options are: `0b0` Program execution is disabled. `0b1` Program execution is enabled. Program execution takes place if PEEN is 1 and PES is 0. PEEN is automatically cleared to 0 at the end of a test. For more information, see *Resume event* on page Appx-D-656. When PEEN is 0 the **MBISTREQ** signal is driven LOW. When software sets the PEEN bit to 1 the internal state is cleared, including the Read Pipeline Register, RPR. For more information, see *D.8.7 Read Pipeline Register, RPR* on page Appx-D-661. The reset value is `0x0`. This field is RW. **Note** PEEN must be 0 before software changes any registers values, except the CTRL register, when cleaning the PEEN bit to 0 or the PES bit to 0.

Note

All fields are software modifiable, except TEN, MBISTACK, and STATE. Additionally, automatic hardware mechanisms can update some fields.

State machine

PMC-R52 has a state machine that controls its execution state and current state. The current state can be read from the CTRL.STATE field.

There are several events that cause the state machine to transition between states. This is described in:

- *Start event* on page Appx-D-655.
- *Suspend event* on page Appx-D-656.
- *Resume event* on page Appx-D-656.

- *Stop event* on page Appx-D-656.
- *Abort event* on page Appx-D-656.

The following table shows the three main states.

Table D-21 Execution states

State	Description
Initial	Program is not executing and **MBISTREQ** is LOW.
Run	In MBIST entry or exit sequence or program is executing.
Suspended	Program execution is suspended and **MBISTREQ** is LOW.

The following figure shows the state transitions for the state machine.

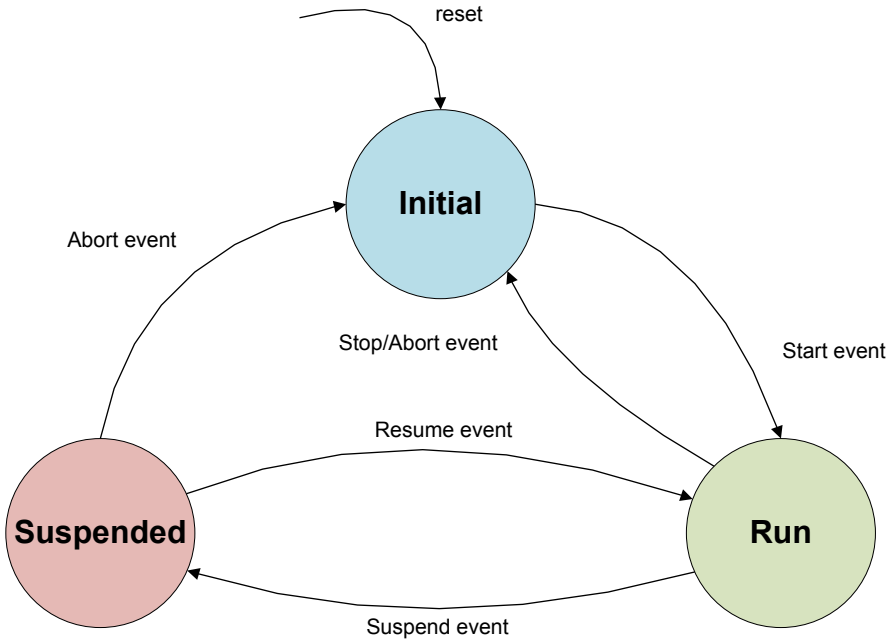

Figure D-5 State machine diagram

Start event

When a start event occurs:

- The internal state that is not software writeable is initialized.
- The MBIST interface entry sequence is carried out the program execution starts.

This event is used in the Initial state and is caused by the following condition:

- CTRL.PEEN is 0 and software writes 1 to CTRL.PEEN and 0 to CTRL.PES.

Note

- If this event occurs after a test has completed successfully and software has not re-initialized the PCR, CADDR or RADDR registers, then a stop condition might still be active and the state machine briefly enters the Run state and then returns to the Initial state. MBISTREQ is not asserted.

Suspend event

When a suspend event occurs, the MBIST interface exit sequence is carried out.

This event can only occur when the state machine is in the Run state and is caused by the following condition:

- CTRL.PEEN is 1, CTRL.TCSEN is 1, and CTRL.EXECO is 0 and an instruction is executed with LASTOP OP field value and a stop event does not occur at the same time. When this condition occurs, CTRL.PES is automatically set to 1.

Resume event

When the resume event occurs, the MBIST interface entry sequence is carried out. This event can only occur when the state machine is in the Suspended state and is caused by either of the following conditions:

- CTRL.PEEN, CTRL.PES, and CTRL.TCSEN bits are 1, and the **TC** signal is HIGH. When this occurs, CTRL.PES is automatically set to 0.
- CTRL.PEEN, CTRL.PES, and CTRL.TCSEN bits are 1, and the software writes 1 to CTRL.PEEN, 0 to CTRL.PES, and 1 to CTRL.TCSEN.

Stop event

The stop event occurs when a test ends normally. It causes the MBIST interface exit sequence to be carried out, and CTRL.PEEN is set to 0 and CTRL.TE is set to 1. This event can only occur in the Run state and it is caused by either of the following conditions:

- CTRL.PEEN is 1 and an instruction is executed with a LASTOP OP field value and the current address is equal to the STOPADDR register.
- CTRL.PEEN and CTRL.EXECO bits are 1 and an instruction is executed with a LASTOP OP field value.

Note

After this event, the PCR points to the LASTOP instruction and the CADDR and RADDR registers are not updated, even if the UA bit of the instruction is 1.

Abort event

The abort event can occur in either the Run or Suspended states and is used to prematurely abort a test and return to the Initial state. It sets CTRL.TE is 1 and if it caused by a read data check failure, then CTRL.TF is also set to1. This event is caused by either of the following conditions:

- CTRL.PEEN and CTRL.STOPF is 1 and a read data check fails.
- CTRL.PEEN is 1 and software writes 0 to CTRL.PEEN and CTRL.PES.

Note

- Execution is stopped immediately when this event occurs and the PCR, CADDR, RADDR, and RPR registers are not updated. This allows software to determine the failing RAM address where the read data check failure occurred which caused the event, see RPR.R field in *D.8.7 Read Pipeline Register, RPR* on page Appx-D-661.
- Arm recommends that the software does not cause the abort event to occur when the CTRL.MBISTACK is 0 and CTRL.STATE is `0b01` because it might violate the MBIST interface protocol if it occurs during MBIST entry.
- Before software causes an abort from the Suspended state it must follow the steps:
  1. Disable the source of the **TC** signal using the CTRL.TCSEN bit.
  2. Check that PMC-R52 is not in the Run state.
  3. Clear the CTRL.PEEN and CTRL.PES bits.

Program execution modes

The program execution mode is configured by the following bits in the CTRL register.

- PEEN.
- PES.
- EXECO.
- TCSEN.

The following table describes these execution modes.

Table D-22 Program execution modes

PEEN	PES	EXECO	TCSEN	Description
0	-	-	-	Program is not executed.
1	0	0	1	Program is executed until a suspend, stop, or abort event occurs. Used to suspend execution on each iteration of the program loop, waiting for the **TC** input pulse or for software to write 0 to PES.
1	0	1	-	Program is executed until a stop or abort event occurs. Used to execute instructions once.
1	0	0	0	Program is executed until a stop or abort event occurs. Used to loop through all memory locations without suspending.
1	1	-	-	Program execution is suspended waiting for a resume or abort event to occur.

D.8.4 Memory Control Register, MCR

The MCR configures the attributes for the memory array under test.

Usage constraints
This register is RW.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
MCR is a 32-bit register.

The following figure shows the MCR bit assignments.

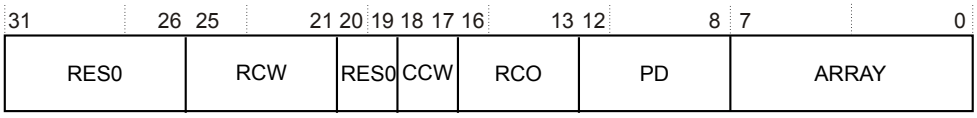

Figure D-6 MCR bit assignments

The following table describes the MCR bit assignments.

Table D-23 MCR bit assignments

Bits	Name	Function
[31:26]	-	Reserved, RES0. The reset value is UNKNOWN. This field is UNK/SBZP.
[25:21]	RCW	Row counter width. This must be set to the width of the row section of the address bus of the RAM under test-2. Therefore, 0 corresponds to two row address bits. The maximum value that this field can be programmed to is MAWIDTH-CCW-4 and the behavior is unpredictable if a value greater than this is used. The reset value is UNKNOWN. This field is RW.
[20:19]	-	Reserved, RES0. The reset value is UNKNOWN. This field is UNK/SBZP.
[18:17]	CCW	Column counter width. This must be set to the width of the column section of the address bus of the RAM under test-2. The values are as follows: `0b00` 2 bits, 4 RAM columns. `0b01` 3 bits, 8 RAM columns. `0b10` 4 bits, 16 RAM columns. `0b11` 5 bits, 32 RAM columns. The reset value is UNKNOWN. This field is RW. **Note** The encoding of this field might change in subsequent revisions of PMC-R52.
[16:13]	RCO	RAM cycles per operation. This must be set to the number of cycles that the RAM under test requires for each access-1. Therefore, a value of 0 corresponds to one cycle per access. The reset value is UNKNOWN. This field is RW. **Note** • Unused field bits are reserved and must be treated as UNK/SBZP. • RCO width is set by the `RCOWIDTH` parameter. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646.

Table D-23 MCR bit assignments (continued)

Bits	Name	Function
[12:8]	PD	Pipeline depth. This is the number of pipeline stages in the processor path from the **MBISTADDR** to the **MBISTOUTDATA** signal for the memory under test. For more information, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*. For example, if there are three pipeline stages, then this field must be set to 3. The reset value is UNKNOWN. This field is RW. **Note** • Unused field bits are reserved and must be treated as UNK/SBZP. • PD width is set by the `PDWIDTH` parameter. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646.
[7:0]	ARRAY	**MBISTARRAY** signal value. For more information, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*. The reset value is UNKNOWN. This field is RW. **Note** • Unused field bits are reserved and must be treated as UNK/SBZP. • ARRAY width is set by the `MARWIDTH` parameter. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646.

Note

- All fields are software modifiable.
- Software must initialize this register before CTRL.PEEN bit is set to 1. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.

D.8.5 Byte Enable Register, BER

The BER sets the value of the **MBISTBE** signal.

Usage constraints
This register is RW.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
BER is a 32-bit register.

The following figure shows the BER bit assignments.

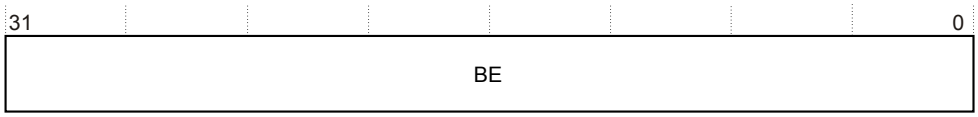

Figure D-7 BER bit assignments

The following table describes the BER bit assignments.

Table D-24 BER bit assignments

Bits	Name	Function
[31:0]	BE	**MBISTBE** signal value. For more information, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*. The reset value is UNKNOWN. This field is RW. **Note** • Unused field bits are reserved and must be treated as UNK/SBZP. • BE width is set by the `MBWIDTH` parameter. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646.

Note

This register is only software modifiable. Software must initialize this register before the CTRL.PEEN bit is set to 1. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.

D.8.6 Program Counter Register, PCR

The PCR contains the Program Counter field.

Usage constraints

This register is RW.

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

PCR is a 32-bit register.

The following figure shows the PCR bit assignments.

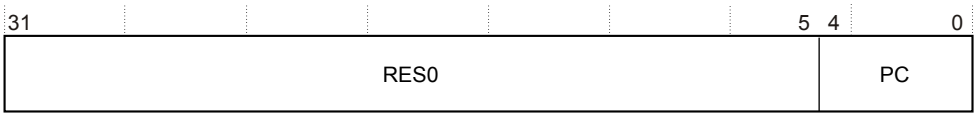

Figure D-8 PCR bit assignments

The following table describes the PCR bit assignments.

Table D-25 PCR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0. The reset value is UNKNOWN. This field is UNK/SBZP.
[4:0]	PC	Program Counter. This field points to the next program register that is executed. PC is automatically incremented when each instruction is executed. PC is automatically set to `0x00` when an instruction is executed with a LASTOP OP field value and the test is not ending. For more information, see*D.8.15 Program registers, P0-P31* on page Appx-D-670. The reset value is UNKNOWN. This field is RW. **Note** • Unused field bits are reserved and must be treated as UNK/SBZP. • PC width is set by the $\log_2$(`PROGSIZE`) parameter. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646.

Note

This register is software modifiable and is automatically updated. Software must initialize this register to 0 before the CTRL.PEEN bit is set to 1. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.

D.8.7 Read Pipeline Register, RPR

If a memory fault is detected and the CTRL.STOPF bit is 1, the RPR allows software to determine which read instruction the fault relates to and the memory address containing the fault.

For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.

Usage constraints

This register is RO.

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

RPR is a 32-bit register.

The following figure shows the RPR bit assignments.

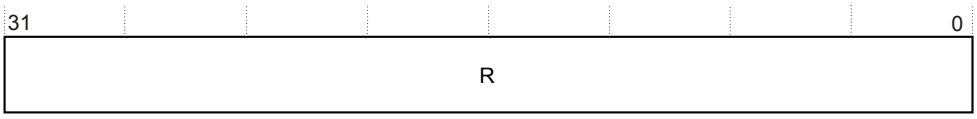

Figure D-9 RPR bit assignments

The following table describes the RPR bit assignments.

Table D-26 RPR bit assignments

Bits	Name	Function
[31:0]	R	This is the current value of the RPR in the execution unit. Only R[MCR.PD:0] bits are valid and the unused *Most Significant Bits* (MSBs) are set to 0. When a read is executed: • R[0] is set to 1. • R is shifted left for every clock cycle. When a data comparison fails, the corresponding read has reached R[MCR.PD]. To determine which read instruction failed: • Count the number of bits set in the R field (n). • And count back n read instructions from the instruction pointed to by PCR.PC. Using the example in *Table D-2 Microcode for example short-burst software transparent on-line MBIST test algorithm* on page Appx-D-635: • If the test stopped and PCR.PC=8 (row 9) and R has 2 bits set, then the failing read instruction was instruction 4 (row 5). • The failing address can be determined from the following values of the instructions executed since the failing read: — CADDR.CA. — RADDR.RA. — CTRL.ADDRID. — CTRL.ADDRCR. — Px.UA, where x ranges between 0 to 31. For more information on how MBISTOUTADDR is calculated, see *Address output value* on page Appx-D-665. The reset value is UNKNOWN. This field is RO. **Note** • Unused field bits are reserved and must be treated as UNK/WI. • R width is $2^{PDWIDTH}$. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646.

D.8.8 Stop Address Register, STOPADDR

The STOPADDR register contains the stop address that is used for comparison against the current address to determine when all memory locations have been tested, which then ends the test.

Usage constraints
This register is RW.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
STOPADDR is a 32-bit register.

The following figure shows the STOPADDR bit assignments.

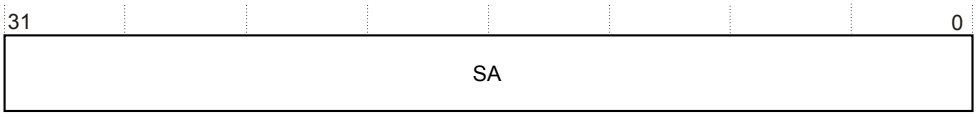

Figure D-10 STOPADDR bit assignments

The following table describes the STOPADDR bit assignments.

Table D-27 STOPADDR bit assignments

Bits	Name	Function
[31:0]	SA	Stop address. The reset value is UNKNOWN. This field is RW. **Note** • Unused field bits are reserved and must be treated as UNK/SBZP. • The width of field SA is set by `MAWIDTH`. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646. Depending on this parameter, the MSBs are unused.

Note

This register is only software modifiable. Software must initialize this register before the CTRL.PEEN bit is set to 1. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.

D.8.9 Column Address Register, CADDR

The CADDR contains the column address field.

Usage constraints
This register is RW.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
CADDR is a 32-bit register.

The following figure shows the CADDR bit assignments.

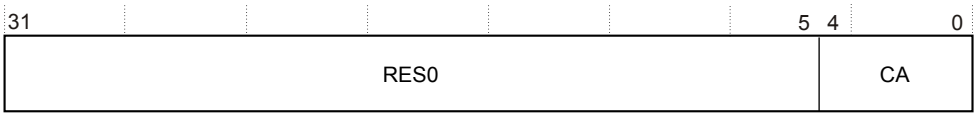

Figure D-11 CADDR bit assignments

The following table describes the CADDR bit assignments.

Table D-28 CADDR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, RES0. The reset value is UNKNOWN. This field is UNK/SBZP.
[4:0]	CA	Column address. This field contains the current value of the column bits of the memory address value. It is updated when an instruction is executed with the UA bit set 1, depending on the values of the CTRL.ADDRID and CTRL.ADDRCD bits. Only CA[MCR.CCW+1:0] bits are used and the remaining MSBs are cleared to 0 when an instruction is executed with the UA bit set to 1. For information on: • The UA bit, see *D.8.15 Program registers, P0-P31* on page Appx-D-670. • CTRL, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650. • Output address calculation, see *Address output value* on page Appx-D-665. The reset value is UNKNOWN. This field is RW. **Note** When software writes to this register, only bits CA[MCR.CCW+1:0] are updated and the remaining MSBs are cleared to 0.

Note

This register is software modifiable and is automatically updated. Software must initialize this register before the CTRL.PEEN bit is set to 1. It is recommended that the MCR.CCW field is set correctly for the memory under test before writing to this register. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.

D.8.10 Row Address Register, RADDR

The RADDR contains the row address register field.

Usage constraints
This register is RW.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
RADDR is a 32-bit register.

The following figure shows the RADDR bit assignments.

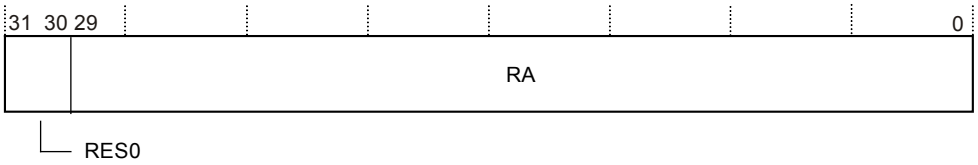

Figure D-12 RADDR bit assignments

The following table describes the RADDR bit assignments.

Table D-29 RADDR bit assignments

Bits	Name	Function
[31:30]	-	Reserved, RES0. The reset value is UNKNOWN. This field is UNK/SBZP.
[29:0]	RA	Row address. This field contains the current value of the row bits of the memory address value. It is updated when an instruction is executed with the UA bit set 1, depending on the values of the CTRL.ADDRID and CTRL.ADDRCD bits. Only RA[MCR.RCW+1:0] bits are used and the remaining MSBs are cleared to 0 when an instruction is executed with the UA bit set to 1. The reset value is UNKNOWN. This field is RW. **Note** • When software writes to this register, only bits RA[MCR.RCW+1:0] are updated and the remaining MSBs are cleared to 0. • Unused field bits are reserved and must be treated UNK/SBZP. • The width of the RA field is set by the `MAWIDTH` parameter-2. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646.

Note

This register is software modifiable and is automatically updated. Software must initialize this register before the CTRL.PEEN bit is set to 1. It is recommended that the MCR.RCW field is set correctly for the memory under test before writing to this register. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.

Address output value

The **MBISTADDR** output value is either the current or the next address depending on the value of the AO bit in the current instruction.

For more information, see *D.8.15 Program registers, P0-P31* on page Appx-D-670.

The RADDR and CADDR registers are combined to form the address. The next value of these registers is determined by the CTRL.ADDRID and CTRL.ADDRCD bits. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.

The following formula, using Verilog syntax, shows how the RADDR and CADDR registers are combined to generate the address. The current address uses the current values of the RADDR and CADDR registers and the next address uses the next values of these registers.

```
address={{MAWIDTH-aw{1'b0}}, RADDR[MCR.RCW+1:0], CADDR[MCR.CCW+1:0]}
```

Where, aw=MCR.RCW+MCR.CCW+4

To calculate the current address, software must concatenate the RADDR and CADDR registers together in the same way.

D.8.11 Data registers, X0-X7 and Y0-Y7

X0-X7 and Y0-Y7 registers are two independent data registers with the same register descriptions.

Usage constraints
These registers are RW.

Traps and enables
There are no traps and enables affecting these registers.

Configurations
These registers are available in all build configurations.

Attributes
These are 32-bit registers.

The following figure shows the X0-X7 and Y0-Y7 bit assignments.

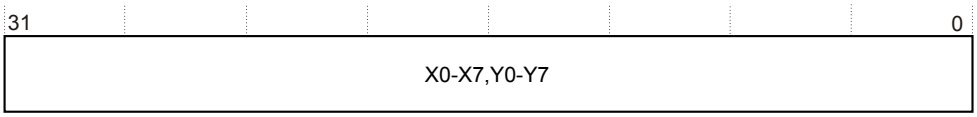

Figure D-13 X0-X7 and Y0-Y7 bit assignments

From a software perspective, they are split into 32-bit wide registers X0-X7 and Y0-Y7 as shown in the following table.

The following table describes the X0-X7 and Y0-Y7 bit assignments.

Table D-30 X0-X7 and Y0-Y7 bit assignments

Bits	Name	Function
[31:0]	X0,Y0	Bits [31:0]. The reset value is UNKNOWN. This field is RW.
[31:0]	X1,Y1	Bits [63:32]. The reset value is UNKNOWN. This field is RW.
[31:0]	X2,Y2	Bits [95:64]. The reset value is UNKNOWN. This field is RW.
[31:0]	X3,Y3	Bits [127:96]. The reset value is UNKNOWN. This field is RW.
[31:0]	X4,Y4	Bits [159:128]. The reset value is UNKNOWN. This field is RW.
[31:0]	X5,Y5	Bits [191:160]. The reset value is UNKNOWN. This field is RW.

Table D-30 X0-X7 and Y0-Y7 bit assignments (continued)

Bits	Name	Function
[31:0]	X6,Y6	Bits [223:192]. The reset value is UNKNOWN. This field is RW.
[31:0]	X7,Y7	Bits [255:224]. The reset value is UNKNOWN. This field is RW.

Note

- These registers are software modifiable and are automatically updated.
- Software must initialize them before the CTRL.PEEN bit is set to 1. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
- Unused register bits are reserved and must be treated as UNK/SBZP.
- The total width of the X and Y registers is set by the `MDWIDTH` parameter. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646. The number of 32-bit locations occupied by each register is int(`MDWIDTH`/32)+1.

D.8.12 Auxiliary Input Register, AIR

The AIR stores the value of the **AUXIN** signal when **MBISTACK** is asserted. The **AUXIN** signal is a general purpose signal that can be used, to capture signals during MBIST operation. AIR[0] contains the value of **MBISTERREXT** output signal from the Cortex-R52 processor. If a bit is set in a clock cycle, these bits remain set until software clears them.

For more information on **AUXIN**and **MBISTACK**, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*.

Usage constraints
This register is RW.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
AIR is a 32-bit register.

The following figure shows the AIR bit assignments.

Figure D-14 AIR bit assignments

The following table describes the AIR bit assignments.

Table D-31 AIR bit assignments

Bits	Name	Function
[31:0]	AIR	**AUXIN** signal value. The reset value is UNKNOWN. This field is RW. **Note** • Unused field bits are reserved and must be treated as UNK/SBZP. • AIR width is set by the `AIWIDTH` parameter. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646.

D.8.13 Auxiliary Output Register, AOR

The AOR stores the value of the **AUXOUT** signal. The **AUXOUT** signal is a general purpose signal that can be used, for example, to control external logic associated with on-line MBIST testing, such as, enabling TC signal pulse generation or the frequency of these pulses.

For more information on **AUXOUT** signal, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*.

Usage constraints
This register is RW.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
AOR is a 32-bit register.

The following figure shows the AOR bit assignments.

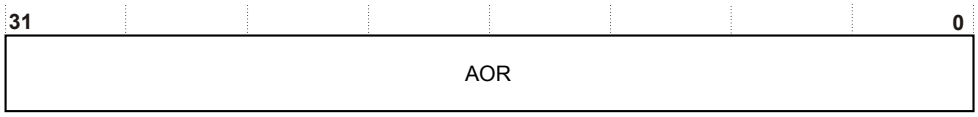

Figure D-15 AOR bit assignments

The following table describes the AOR bit assignments.

Table D-32 AOR bit assignments

Bits	Name	Function
[31:0]	AOR	**AUXOUT** signal value. The reset value is UNKNOWN. This field is RW. **Note** • Unused field bits are reserved and must be treated as UNK/SBZP. • AOR width is set by the `AOWIDTH` parameter. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646.

D.8.14 Data Mask Registers, DM0-DM7

The DM0-DM7 registers hold values that are used to mask all the RAM data values. These registers mask the **MBISTINDATA**, **MBISTOUTDATA**, and the data values before they are compared. When a DM bit is 0, the corresponding masked bit is 0 and when a DM bit is 1, the corresponding data bit is not masked.

For information on **MBISTACK**, see the PMC-R52 appendix in the *Arm® Cortex®-R52 Processor Integration Manual*.

Usage constraints

These registers are RW.

Traps and enables

There are no traps and enables affecting these registers.

Configurations

These registers are available in all build configurations.

Attributes

These are 32-bit registers.

The following figure shows the DM0-DM7 bit assignments.

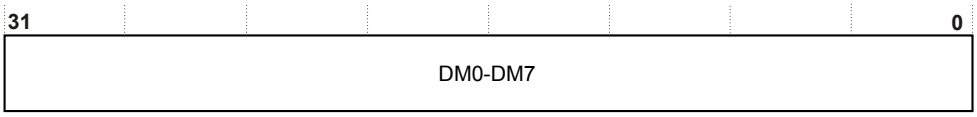

Figure D-16 DM0-DM7 bit assignments

From a software perspective, they are split into 32-bit wide registers DM0-DM7 as shown in the following table.

The following table describes the DM0-DM7 bit assignments.

Table D-33 DM0-DM7 bit assignments

Bits	Name	Function
[31:0]	DM0	Bits [31:0]. The reset value is UNKNOWN. This field is RW.
[31:0]	DM1	Bits [63:32]. The reset value is UNKNOWN. This field is RW.
[31:0]	DM2	Bits [95:64]. The reset value is UNKNOWN. This field is RW.
[31:0]	DM3	Bits [127:96]. The reset value is UNKNOWN. This field is RW.
[31:0]	DM4	Bits [159:128]. The reset value is UNKNOWN. This field is RW.
[31:0]	DM5	Bits [191:160]. The reset value is UNKNOWN. This field is RW.

Table D-33 DM0-DM7 bit assignments (continued)

Bits	Name	Function
[31:0]	DM6	Bits [223:192]. The reset value is UNKNOWN. This field is RW.
[31:0]	DM7	Bits [255:224]. The reset value is UNKNOWN. This field is RW.

Note

- These registers are only software modifiable.
- Software must initialize them before the CTRL.PEEN bit is set to 1. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
- Unused register bits are reserved and must be treated as UNK/SBZP.
- The total width of these registers is set by the `MDWIDTH` parameter. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646. The number of 32-bit locations occupied by each register is int(`MDWIDTH`/32)+1.

D.8.15 Program registers, P0-P31

The P0-P31 registers hold microcode instructions.

These instructions constitute the algorithm that is executed to generated the required MBIST transactions that test the memory array that the MCR.ARRAY field selects. For more information, see *D.8.4 Memory Control Register, MCR* on page Appx-D-657.

Usage constraints

These registers are RW.

Traps and enables

There are no traps and enables affecting these registers.

Configurations

These registers are available in all build configurations.

Attributes

These are 32-bit registers.

The following figure shows the P0-P31 bit assignments.

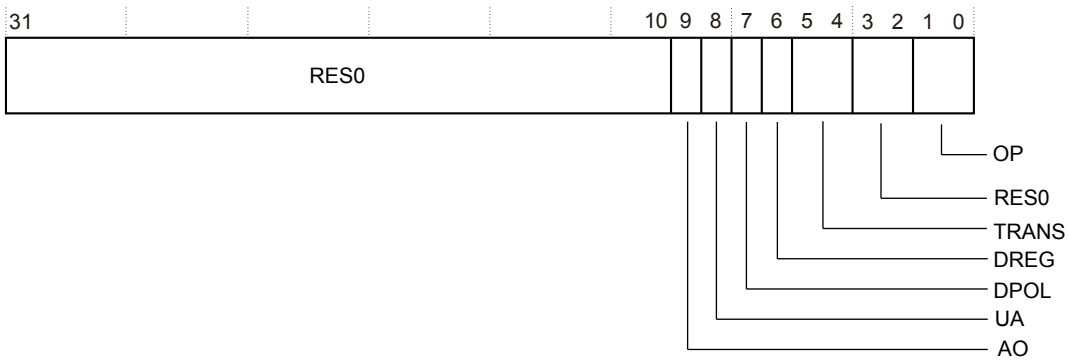

Figure D-17 P0-P31 bit assignments

From a software perspective, they are split into 32-bit wide registers P0-P31 as shown in the following table.

The following table describes the P0-P31 bit assignments.

Table D-34 P0-P31 bit assignments

Bits	Name	Function
[31:10]	-	Reserved, RES0. The reset value is UNKNOWN. This field is RW.
[9]	AO	Address output. This bit determines which address value is output on the **MBISTADDR** signal. The encoding of this bit is: `0b0` Current address. `0b1` Next address. The address is determined by the RADDR and CADDR registers. For more information, see *Address output value* on page Appx-D-665. The reset value is UNKNOWN. This field is RW.
[8]	UA	Update address. This bit causes CADDR and RADDR to be updated with the next value according to the values of CTRL.ADDRID and CTRL.ADDRCD bits. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650. The encoding of this bit is: `0b0` Hold address. Do not update the address registers. `0b1` Update address. Load the address registers with the next address value. The reset value is UNKNOWN. This field is RW.
[7]	DPOL	Data polarity. This bit controls whether the data register value is inverted or no. For READ transactions, it is used to determine if the value of the specified DREG is checked against the returned data. For WRITE transactions, it is used to determine if the value of the specified DREG is written to the memory array. The encoding of this bit is: `0b0` Use non-inverted DREG register value. `0b1` Use inverted DREG register value. The reset value is UNKNOWN. This field is RW.
[6]	DREG	Data register that the instruction uses. The options are: `0b0` X. `0b1` Y. For more information on data registers, see *D.8.11 Data registers, X0-X7 and Y0-Y7* on page Appx-D-666. The reset value is UNKNOWN. This field is RW.
[5:4]	TRANS	MBIST transaction generated. The encoding of this field is: `0b00` NONE. No transaction generated. `0b01` READ. `0b10` WRITE. `0b11` Reserved. This encoding must not be used. The reset value is UNKNOWN. This field is RW.

Table D-34 P0-P31 bit assignments (continued)

Bits	Name	Function
[3:2]	-	Reserved for future expansion of the OP field. The reset value is UNKNOWN. This field is UNK/SBZP.
[1:0]	OP	Operation field. This field specifies the operation that is performed by an instruction in addition to the MBIST transaction. The encoding and function of each code is: `0b00` NONE. No operation. For TRANS READ instructions, the returned data is checked so that it is equal to the contents of the data register selected by DREG. If they are not equal, then the CTRL.TF bit is set to 1. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650. `0b01` LASTOP. This causes the PCR.PC field to be set to `0x00` and execution either stops or suspends depending on the CTRL register settings. For more information, see *Suspend event* on page Appx-D-656 and *Resume event* on page Appx-D-656. Reads are checked in the same way as the NONE operation. `0b10` SAVERD. Save read data to data register specified by the DREG field. The read data returned is not checked. This operation must only be used with reads. `0b11` WAITRU. Before executing this instruction wait for the data register specified by the DREG field to be updated by a previous read (SAVERD) in the pipeline. **Note** • The OP WAITRU waits for all previous OP SAVERD reads in the pipeline that update DREG to complete. It is not useful to have more than one OP SAVERD read in the pipeline with the same DREG value. Therefore, it would a programming error if this is the case. • If there are no previous OP SAVERD reads in the pipeline that update DREG, then OP WAITRU is ignored, but no operation is performed on the MBIST interface for one cycle. • OP WAITRU is not required for writes that source data from a data register that was updated by an instruction that used OP SAVERD when the pipeline depth is less than the number of instructions between the read and the write multiplied by the cycles per operation of the MBIST transactions. The OP WAITRU should be avoided in this case because it adds a redundant cycle to the program execution. The reset value is UNKNOWN. This field is RW.

Note

- Software must initialize these registers before the CTRL.PEEN bit is set to 1. When CTRL.PEEN is set to 1, there must be at least one instruction with a LASTOP OP field value. For more information, see *D.8.3 Main Control Register, CTRL* on page Appx-D-650.
- Unused register bits are reserved and must be treated as UNK/SBZP.
- `PROGSIZE` sets the number of registers that are implemented. For more information, see *Table D-18 Internal RTL constants* on page Appx-D-646.

D.8.16 CoreSight registers

This section describes the CoreSight registers.

For a full description of the CoreSight registers, see *Arm® CoreSight™ Architecture Specification v3.0*.

Register summary

The following table lists the CoreSight registers.

Table D-35 CoreSight register summary

Offset	Type	Reset value	Name	Description
0xF00	RO	0x00000000	ITCTRL	Integration Mode Control register, see *Integration Mode Control Register, ITCTRL* on page Appx-D-674.
0xF04-0xF9C	UNK/SBZP	UNKNOWN	-	Reserved.
0xFA0	RW	0x0000000F	CLAIMSET	Claim Tag Set Register, see *Claim Tag Set Register, CLAIMSET* on page Appx-D-674.
0xFA4	RW	0x00000000	CLAIMCLR	Claim Tag Clear Register, see *Claim Tag Clear Register, CLAIMCLR* on page Appx-D-675.
0xFA8	RO	-[cl]	DEVAFF0	Device Affinity Register 0, see *Device Affinity 0 Register, DEVAFF0* on page Appx-D-676.
0xFAC	RO	0x00000000	DEVAFF1	Device Affinity Register 1, see *Device Affinity 1 Register, DEVAFF1* on page Appx-D-677.
0xFB0	WO	UNKNOWN	LAR	Lock Access Register, see *Lock Access Register, LAR* on page Appx-D-677.
0xFB4	RO	0x00000000	LSR	Lock Status Register, see *Lock Status Register, LSR* on page Appx-D-677.
0xFB8	RO	0x00000000	AUTHSTATUS	Authentication Status Register, see *Authentication Status Register, AUTHSTATUS* on page Appx-D-677.
0xFBC	RO	0x47700A55	DEVARCH	Device Architecture Register, see *Device Architecture Register, DEVARCH* on page Appx-D-677.
0xFC0	RO	0x00000000	DEVID2	Device Configuration Register 2, see *Device ID Registers, DEVID, DEVID1, and DEVID2* on page Appx-D-678.
0xFC4	RO	0x00000000	DEVID1	Device Configuration Register 1, see *Device ID Registers, DEVID, DEVID1, and DEVID2* on page Appx-D-678.
0xFC8	RO	0x00000000	DEVID	Device Configuration Register, see *Device ID Registers, DEVID, DEVID1, and DEVID2* on page Appx-D-678.
0xFCC	RO	0x00000055	DEVTYPE	Device Type Identifier Register, see *Device Type Identifier Register, DEVTYPE* on page Appx-D-678.
0xFD0	RO	0x00000004	PIDR4	Peripheral Identification Registers, see *Peripheral Identification Registers* on page Appx-D-679.
0xFD4	RO	0x00000000	PIDR5	
0xFD8	RO	0x00000000	PIDR6	
0xFDC	RO	0x00000000	PIDR7	
0xFE0	RO	0x000000B6	PIDR0	
0xFE4	RO	0x000000B9	PIDR1	
0xFE8	RO	0x0000004B	PIDR2	
0xFEC	RO	-[cm]	PIDR3	

[cl] See *Device Affinity 0 Register, DEVAFF0* on page Appx-D-676.
[cm] *Peripheral Identification Register 3, PIDR3* on page Appx-D-681.

Table D-35 CoreSight register summary (continued)

Offset	Type	Reset value	Name	Description
0xFF0	RO	0x0000000D	CIDR0	Component Identification Registers, see *Component Identification Registers* on page Appx-D-684.
0xFF4	RO	0x00000090	CIDR1	
0xFF8	RO	0x00000005	CIDR2	
0xFFC	RO	0x000000B1	CIDR3	

Integration Mode Control Register, ITCTRL

A CoreSight component can use the ITCTRL register to dynamically switch between functional mode and integration mode.

Usage constraints

This register is RO.

Traps and enables

There are no traps and enables affecting this register.

Configurations

This register is available in all build configurations.

Attributes

ITCTRL is a 32-bit register.

The following figure shows the ITCTRL bit assignments.

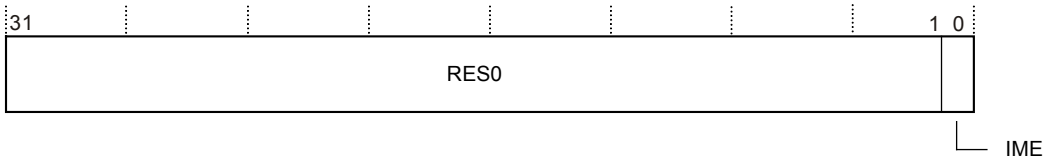

Figure D-18 ITCTRL bit assignments

The following table describes the ITCTRL bit assignments.

Table D-36 ITCTRL bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RES0.
[0]	IME	Integration mode enable. **0** PMC-R52 is not in integration mode.

Claim Tag Set Register, CLAIMSET

The CLAIMSET register sets bits to indicate that a debug agent is using a functionality that CoreSight components make available.

Usage constraints

This register is RW.

Traps and enables

There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
CLAIMSET is a 32-bit register.

The following figure shows the CLAIMSET bit assignments.

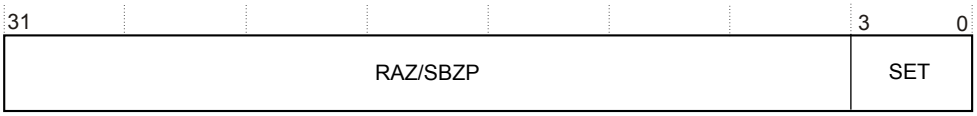

Figure D-19 CLAIMSET bit assignments

The following table describes the CLAIMSET bit assignments.

Table D-37 CLAIMSET bit assignments

Bits	Name	Function
[31:4]	-	RAZ/SBZP.
[3:0]	SET	On reads, for each bit: **0** Claim tag bit is not implemented. **1** Claim tag bit is implemented. On writes, for each bit: **0** No effect. **1** Relevant bit of the claim tag is set.

Claim Tag Clear Register, CLAIMCLR

The CLAIMCLR register clears bits to indicate that a debug agent is no longer using a functionality that CoreSight components make available.

Usage constraints
This register is RW.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
CLAIMCLR is a 32-bit register.

The following figure shows the CLAIMCLR bit assignments.

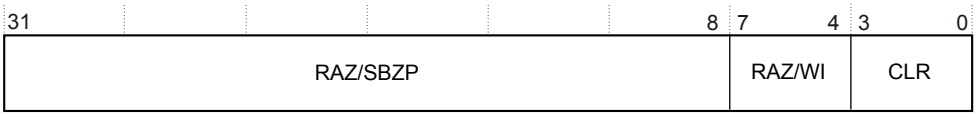

Figure D-20 CLAIMCLR bit assignments

The following table describes the CLAIMCLR bit assignments.

Table D-38 CLAIMCLR bit assignments

Bits	Name	Function
[31:8]	-	RAZ/SBZP.
[7:4]	-	RAZ/WI.
[3:0]	CLR	On reads, for each bit: **0** Claim tag bit is not set. **1** Claim tag bit is set. On writes, for each bit **0** No effect. **1** Relevant bit of the claim tag is cleared.

Device Affinity 0 Register, DEVAFF0

The DEVAFF0 register enables a debugger to determine whether two components have an affinity with each other.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
DEVAFF0 is a 32-bit register.

The following figure shows the DEVAFF0 bit assignments.

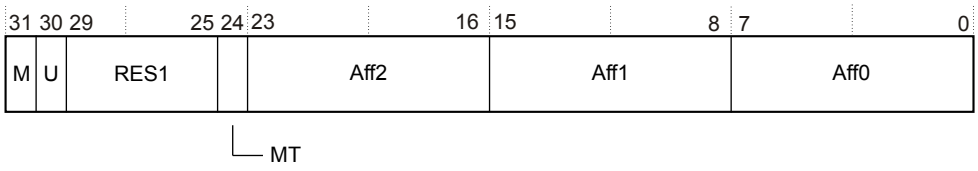

Figure D-21 DEVAFF0 bit assignments

The following table describes the DEVAFF0 bit assignments.

Table D-39 DEVAFF0 bit assignments

Bits	Name	Function
[31]	M	Reserved, RES1.
[30]	U	Indicates a single core system as being distinct from core 0 in a processor. This value is: **0** PMC-R52 is a part of the processor.
[29:25]	-	Reserved, RES0.
[24]	MT	Indicates whether the lowest level of affinity consists of logical cores that are implemented using a multi-threading approach. This value is: **0** Performance of PEs at the lowest affinity level is largely independent.
[23:16]	Aff2	Affinity level 2. The least significant affinity level field. Indicates the value read from the **CFGMPIDRAFF2** signal.
[15:8]	Aff1	Affinity level 1. The intermediate affinity level field. Indicates the value read from the **CFGMPIDRAFF1** signal.
[7:0]	Aff0	Affinity level 0. The most significant affinity level field. `0x80` Affinity level 0 is not applicable. PMC-R52 is a processor-level component and so this field is not used.

Device Affinity 1 Register, DEVAFF1

The DEVAFF1 register has the value `0x00000000`.

Lock Access Register, LAR

The LAR is not implemented.

Lock Status Register, LSR

The LSR is not implemented.

Authentication Status Register, AUTHSTATUS

The AUTHSTATUS register read value is `0x00000000`, indicating that no authentication is implemented.

Device Architecture Register, DEVARCH

The DEVARCH identifies the architect and architecture of a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
DEVARCH is a 32-bit register.

The following figure shows the DEVARCH bit assignments.

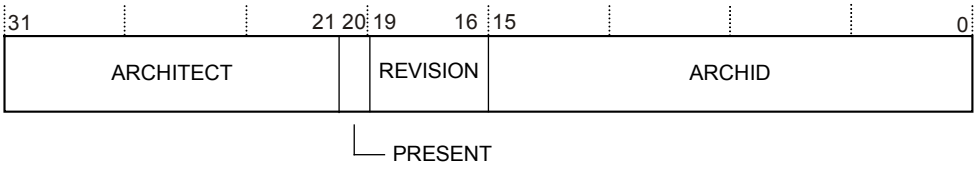

Figure D-22 DEVARCH bit assignments

The following table describes the DEVARCH bit assignments.

Table D-40 DEVARCH bit assignments

Bits	Name	Function
[31:21]	ARCHITECT	Defines the architect of the component. `0x23B` Arm
[20]	PRESENT	Indicates the presence of this register. 1 Register is present.
[19:16]	REVISION	Architecture revision: `0b0000` PMC architecture revision 0
[15:0]	ARCHID	Architecture ID: `0x0A55` PMC component.

Device ID Registers, DEVID, DEVID1, and DEVID2

The DEVID, DEVID1, and DEVID2 registers are not implemented and all have `0x000000000` read values.

Device Type Identifier Register, DEVTYPE

If the part number field is not recognized, a debugger can report the information that the DEVTYPE register provides about the component instead.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
DEVTYPE is a 32-bit register.

The following figure shows the DEVTYPE bit assignments.

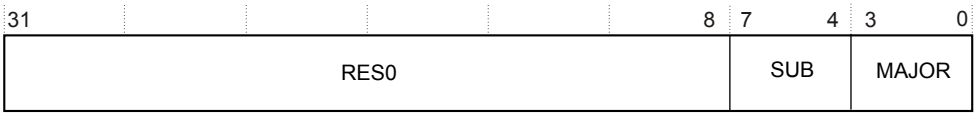

Figure D-23 DEVTYPE bit assignments

The following table describes the DEVTYPE bit assignments.

Table D-41 DEVTYPE bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	SUB	The subtype of the component: `0x5` Memory. Tightly-couple device, such as *Built-In Self-Test* (BIST).
[3:0]	MAJOR	The main type of the component: `0x5` Debug logic.

Peripheral Identification Registers

There are eight read-only Peripheral Identification Registers, PIDR7-PIDR0.

Peripheral Identification Register 7, PIDR7

PIDR7 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
PIDR7 is a 32-bit register.

The following figure shows the PIDR7 bit assignments.

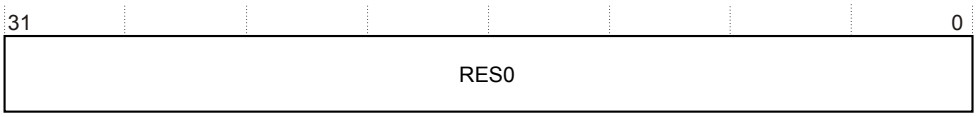

Figure D-24 PIDR7 bit assignments

The following table describes the PIDR7 bit assignments.

Table D-42 PIDR7 bit assignments

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

Peripheral Identification Register 6, PIDR6

PIDR6 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
PIDR6 is a 32-bit register.

The following figure shows the PIDR6 bit assignments.

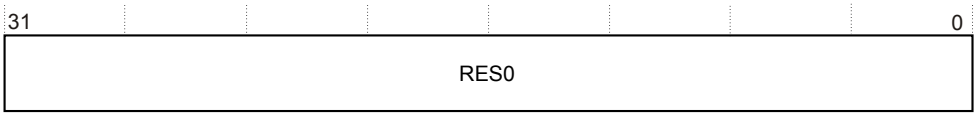

Figure D-25 PIDR6 bit assignments

The following table describes the PIDR6 bit assignments.

Table D-43 PIDR6 bit assignments

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

Peripheral Identification Register 5, PIDR5

PIDR5 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
PIDR5 is a 32-bit register.

The following figure shows the PIDR5 bit assignments.

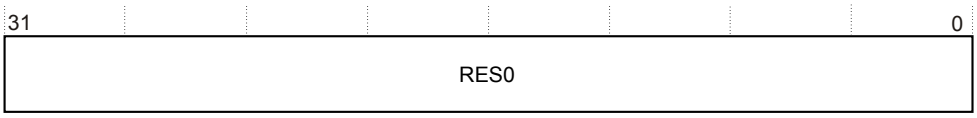

Figure D-26 PIDR5 bit assignments

The following table describes the PIDR5 bit assignments.

Table D-44 PIDR5 bit assignments

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

Peripheral Identification Register 4, PIDR4

PIDR4 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
PIDR4 is a 32-bit register.

The following figure shows the PIDR4 bit assignments.

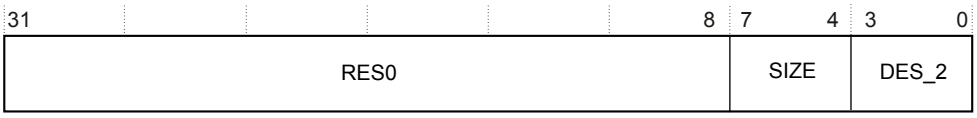

Figure D-27 PIDR4 bit assignments

The following table describes the PIDR4 bit assignments.

Table D-45 PIDR4 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	SIZE	The SIZE field indicates the memory size that this component uses. `0x0` 4KB block.
[3:0]	DES_2	JEP106 continuation code: `0x4` Arm JEP106 continuation code.

Peripheral Identification Register 3, PIDR3

PIDR3 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
PIDR3 is a 32-bit register.

The following figure shows the PIDR3 bit assignments.

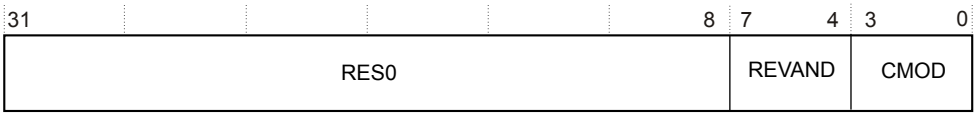

Figure D-28 PIDR3 bit assignments

The following table describes the PIDR3 bit assignments.

Table D-46 PIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	REVAND	Part minor revision. **ECOREVNUM** input signal sampled at reset.
[3:0]	CMOD	Customer modified. `0x0` Indicates from Arm.

Peripheral Identification Register 2, PIDR2

PIDR2 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
PIDR2 is a 32-bit register.

The following figure shows the PIDR2 bit assignments.

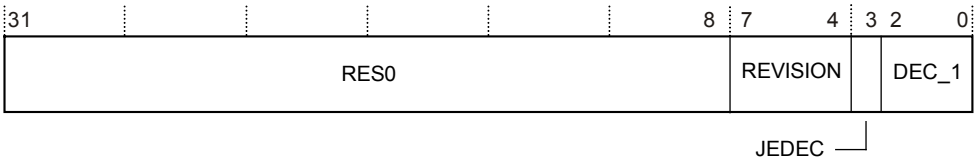

Figure D-29 PIDR2 bit assignments

The following table describes the PIDR2 bit assignments.

Table D-47 PIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	REVISION	Revision number of PMC-R52. `0x4` r1p3.

Table D-47 PIDR2 bit assignments (continued)

Bits	Name	Function
[3]	JEDEC	`0x1` JEDEC-assigned value is used.
[2:0]	DES_1	JEP106 identification bits [6:4]: `0x3` Arm ID.

Peripheral Identification Register 1, PIDR1

PIDR1 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
PIDR1 is a 32-bit register.

The following figure shows the PIDR1 bit assignments.

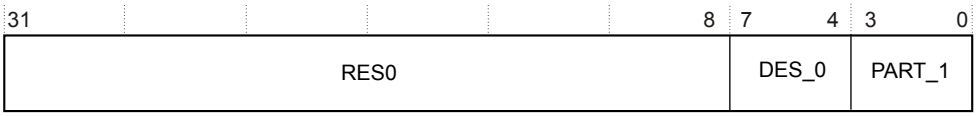

Figure D-30 PIDR1 bit assignments

The following table describes the PIDR1 bit assignments.

Table D-48 PIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	DES_0	JEP106 identification code [3:0]: `0xB` Arm ID.
[3:0]	PART_1	Part number bits [11:8]: `0x9` Indicates Cortex-R52 processor.

Peripheral Identification Register 0, PIDR0

PIDR0 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
PIDR0 is a 32-bit register.

The following figure shows the PIDR0 bit assignments.

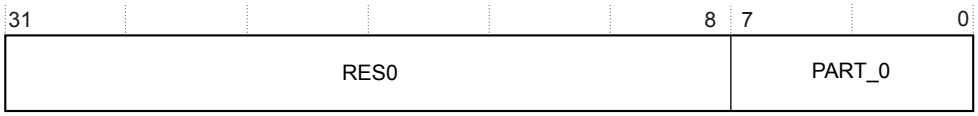

Figure D-31 PIDR0 bit assignments

The following table describes the PIDR0 bit assignments.

Table D-49 PIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PART_0	Part number bits [7:0]: 0xB6 Indicates Cortex-R52 processor.

Component Identification Registers

There are four read-only Component Identification Registers, CIDR3-CIDR3.

Component Identification Register 3, CIDR3

CIDR3 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
CIDR3 is a 32-bit register.

The following figure shows the CIDR3 bit assignments.

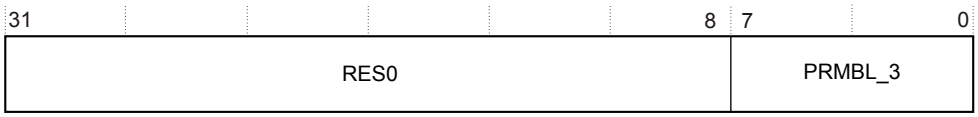

Figure D-32 CIDR3 bit assignments

The following table describes the CIDR3 bit assignments.

Table D-50 CIDR3 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_3	Preamble, segment 3, `0xB1`.

Component Identification Register 2, CIDR2

CIDR2 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
CIDR2 is a 32-bit register.

The following figure shows the CIDR2 bit assignments.

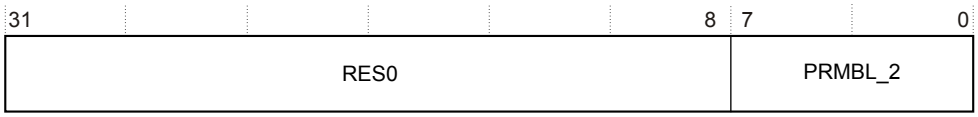

Figure D-33 CIDR2 bit assignments

The following table describes the CIDR2 bit assignments.

Table D-51 CIDR2 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_2	Preamble, segment 2, `0x5`.

Component Identification Register 1, CIDR1

CIDR1 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
CIDR1 is a 32-bit register.

The following figure shows the CIDR1 bit assignments.

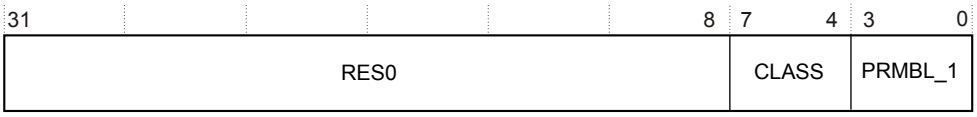

Figure D-34 CIDR1 bit assignments

The following table describes the CIDR1 bit assignments.

Table D-52 CIDR1 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:4]	CLASS	The component class: `0x9` CoreSight component.
[3:0]	PRMBL_1	Preamble, segment 1, `0x0`.

Component Identification Register 0, CIDR0

CIDR0 provides information that can be used to identify a CoreSight component.

Usage constraints
This register is RO.

Traps and enables
There are no traps and enables affecting this register.

Configurations
This register is available in all build configurations.

Attributes
CIDR0 is a 32-bit register.

The following figure shows the CIDR0 bit assignments.

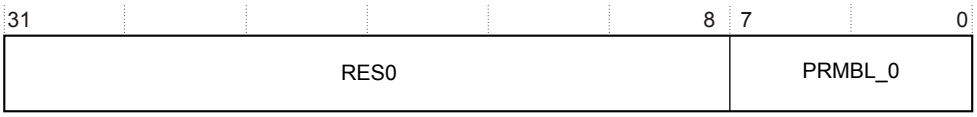

Figure D-35 CIDR0 bit assignments

The following table describes the CIDR0 bit assignments.

Table D-53 CIDR0 bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RES0.
[7:0]	PRMBL_0	Preamble, segment 0, `0x0D`.

Appendix E
Revisions

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

It contains the following section:

- *E.1 Revisions* on page Appx-E-688.

E.1 Revisions

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

Table E-1 Issue 0000-00

Change	Location	Affects
First release for r0p0	-	-

Table E-2 Differences between Issue 0000-00 and Issue 0100-00

Change	Location	Affects
First release for r1p0	Document history table.	r1p0
IMPLEMENTATION DEFINED register naming convention changed	• *3.3.10 Branch Predictor Control Register* on page 3-80. • *3.3.11 Build Options Register* on page 3-81. • *3.3.12 Bus Timeout Register* on page 3-82. • *3.3.14 Cache Segregation Control Register* on page 3-85. • *3.3.17 Configuration Base Address Register* on page 3-88. • *3.3.21 Data Cache Error Record Registers 0 and 1* on page 3-95. • *3.3.29 Flash Error Record Registers 0 and 1* on page 3-102. • *3.3.30 Flash Interface Region Register* on page 3-104. • *3.3.57 Instruction Cache Error Record Registers 0 and 1* on page 3-141. • *3.3.66 Interrupt Monitoring Register* on page 3-154. • *3.3.76 Memory Protection Control Register* on page 3-165. • *3.3.80 Peripheral Port Region Register* on page 3-169. • *3.3.82 Pin Options Register* on page 3-173. • *3.3.88 Quality Of Service Register* on page 3-180. • *3.3.91 Slave Port Control Register* on page 3-183. • *3.3.93 TCM Error Record Register 0 and 1* on page 3-188. • *3.3.94 TCM Region Registers A, B, and C* on page 3-189. • *3.3.97 Test Register 0* on page 3-193.	r1p0
New IMPLEMENTATION DEFINED registers added	• IMP_TESTR1, *3.2.17 c15 registers* on page 3-53 and *3.2.29 AArch32 Implementation defined registers* on page 3-67. • *3.3.68 Invalidate All Register* on page 3-156. • IMP_CDBGDCD, *7.4 Direct access to internal memory* on page 7-225. • IMP_CDBGICD, *7.4 Direct access to internal memory* on page 7-225. • IMP_CDBGDCT, *7.4 Direct access to internal memory* on page 7-225. • IMP_CDBGICT, *7.4 Direct access to internal memory* on page 7-225. • IMP_CDBGDR0, *7.4 Direct access to internal memory* on page 7-225. • IMP_CDBGDR1, *7.4 Direct access to internal memory* on page 7-225.	r1p0

Table E-2 Differences between Issue 0000-00 and Issue 0100-00 (continued)

Change	Location	Affects
ID register fields updated	• *3.3.69 Main ID Register* on page 3-156. • *15.6.1 Floating-point System ID Register* on page 15-534. • *External Debug Peripheral Identification Register 2* on page 11-373. • *ROM table Debug Peripheral Identification Register 2* on page 11-392. • *Performance Monitors Peripheral Identification Register 2* on page 12-413. • *Peripheral Identification Register 2* on page 13-440. • *14.7.58 Peripheral Identification Registers* on page 14-524. • *14.7.29 ID Register 2* on page 14-496. • *Distributor Implementer Identification Register* on page 9-270. • *Redistributor Implementer Identification Register* on page 9-285.	r1p0
CPUACTLR bit[47] added to enable fixed latency for integer divide instructions.	*3.3.19 CPU Auxiliary Control Register* on page 3-90.	r1p0
Syndrome reporting functionality for TCMs added	*3.3.95 TCM Syndrome Register 0 and 1* on page 3-190	r1p0
VMID availability changed	*7.5.6 AXIM QoS and user signals* on page 7-231	r1p0
MPU regions modified	*8.2 MPU regions* on page 8-253	r1p0
Hardware cache invalidate for AXIM added	• *7.3.2 Data cache invalidation* on page 7-223. • *11.6.1 External Debug Calibration Control Register* on page 11-367.	r1p0
IMP_BUILDOPTR.LOCK_STEP width modified to [31:30]	*3.3.11 Build Options Register* on page 3-81	r1p0
HACTLR bit [15] is HACTLR.TESTR1 and HACTLR bit [1] is HACTLR.CDBGDCI	*3.3.33 Hyp Auxiliary Control Register* on page 3-107	
IMP_PINOPTR bit[23] functionality modified	*3.3.82 Pin Options Register* on page 3-173	r1p0
IMP_FLASHFREGIONR.BASEADDRESS width modified	• *3.3.30 Flash Interface Region Register* on page 3-104. • *A.4 Configuration inputs* on page Appx-A-548.	r1p0
Split/Lock functionality added.	*Split/Lock* on page 1-17.	r1p0
CFGFLASHBASEADDR signal width changed to [31:27]	*A.4 Configuration inputs* on page Appx-A-548	r1p0
PMU events modified.	*12.6 Events* on page 12-419 *A.6 Event output signals* on page Appx-A-553	r1p0
Miscellaneous ETM signals added, **ETMEXTIN[3:0]** and **ETMEXTOUT[3:0]**	*Miscellaneous ETM interface signals* on page Appx-A-578	r1p0
DFT signal width modified, **DFTRSTDISABLE[1:0]**	*A.12.1 DFT signals* on page Appx-A-583	r1p0
Split/Lock signal added, **CFGSLSPLIT**	*A.16 Split/Lock signal* on page Appx-A-587	r1p0
Cortex-R52 processor UNPREDICTABLE behavior appendix modified	*Appendix C Processor UNPREDICTABLE Behaviors* on page Appx-C-613	r1p0

Table E-3 Differences between Issue 0100-00 and Issue 0101-00

Change	Location	Affects
First release for r1p1	Document history table.	First documentation release r1p1
IMPLEMENTATION DEFINED IMP_DBGDR2 register removed.	*7.4 Direct access to internal memory* on page 7-225	First documentation release for r1p1
MIDR.Variant and MIDR.Revision bits	*3.3.69 Main ID Register* on page 3-156	First documentation release for r1p1
FPSID.Variant and FPSID.Revision bits	*15.6.1 Floating-point System ID Register* on page 15-534	First documentation release for r1p1
ROMPIDR2.Revision bit	*ROM table Debug Peripheral Identification Register 2* on page 11-392	First documentation release for r1p1
TRCPIDR2.Revision bit	*14.7.58 Peripheral Identification Registers* on page 14-524	First documentation release for r1p1
EDPIDR2.Revision bit	*External Debug Peripheral Identification Register 2* on page 11-373	First documentation release for r1p1
PMPIDR2.Revision bit	*Performance Monitors Peripheral Identification Register 2* on page 12-413	First documentation release for r1p1
CTIPIDR2.Revision bit	*Peripheral Identification Register 2* on page 13-440	First documentation release for r1p1
GICD_IIDR.Variant and GICD_IIDR.Revision bits	*Distributor Implementer Identification Register* on page 9-270	First documentation release for r1p1
GICR_IIDR.Variant and GICR_IIDR.Revision bits	*Redistributor Implementer Identification Register* on page 9-285	First documentation release for r1p1

Table E-4 Differences between Issue 0101-00 and Issue 0101-01

Change	Location	Affects
Second release for r1p1	Document history table.	Second documentation release for r1p1
Note added to bit description for bits [7:4] for ID_ISAR2 register.	*3.3.62 Instruction Set Attribute Register 2* on page 3-147	Second documentation release for r1p1
IMP_ATCMREGIONR, IMP_BTCMREGIONR, and IMP_CTCMREGIONR register introduction changed to provide more clarity, and a note has been added to this section.	*3.3.94 TCM Region Registers A, B, and C* on page 3-189	Second documentation release for r1p1
PRBAR and PRLAR register diagrams corrected	*3.3.85 Protection Region Base Address Register* on page 3-176 and *3.3.86 Protection Region Limit Address Register* on page 3-178.	
ROMPID2 value in Summary of the ROM table Debug Peripheral Identification Registers table updated	*Table 11-47 Summary of the ROM table Debug Peripheral Identification Registers* on page 11-390	Second documentation release for r1p1
Integration Mode Control Register has a note added to indicate that Cortex-R52 processor does not include ATB integration registers	*14.7.47 Integration Mode Control Register* on page 14-516	Second documentation release for r1p1
TRCEXTINSELR.SEL fields width updated	*14.7.21 External Input Select Register* on page 14-486	Second documentation release for r1p1

Table E-4 Differences between Issue 0101-00 and Issue 0101-01 (continued)

Change	Location	Affects
TRCIDR5.NUMEXTIN value updated	*14.7.32 ID Register 5* on page 14-500	Second documentation release for r1p1
Note added to TRCITCTRL description	*14.7.47 Integration Mode Control Register* on page 14-516	Second documentation release for r1p1
Write streaming mode section added	*7.3.3 Write streaming mode* on page 7-224	Second documentation release for r1p1
Minor modification to the write ID capability and read ID capability comments in *Table 7-8 AXI4 master interface attributes* on page 7-228	*7.5.1 AXIM interface attributes* on page 7-228	Second documentation release for r1p1
AXIM interface transfers information on Non-cacheable transaction details updated	*7.5.2 AXIM interface transfers* on page 7-228	Second documentation release for r1p1
Minor modification to description of Normal requests in *Table 7-10 AXIM write address and write response channel identifiers* on page 7-230	*7.5.4 AXIM transaction IDs* on page 7-230	Second documentation release for r1p1
GICR_ISPENDR0 register bits [31:16] function description corrected.	*Interrupt Set-Pending Register 0* on page 9-292	Second documentation release for r1p1
Description of behavior when there are write accesses to IMPLEMENTATION DEFINED read-only registers added	*C.7 Other UNPREDICTABLE behaviors* on page Appx-C-628	Second documentation release for r1p1

Table E-5 Differences between Issue 0101-01 and Issue 0102-00

Change	Location	Affects
First release for r1p2	Document history table.	First documentation release for r1p2
Variant and Revision bits changed for MIDR to reflect r1p2 release	*3.3.69 Main ID Register* on page 3-156	First documentation release for r1p2
AXIS interface information slightly modified to describe functionality when an external agent or core attempts to write the TCMs through the AXIS interface	*7.8 AXIS interface* on page 7-243	First documentation release for r1p2
Revision bits changed for GICD_IIDR to reflect r1p2 release	*Distributor Implementer Identification Register* on page 9-270	First documentation release for r1p2
Revision bits changed for GICR_IIDR to reflect r1p2 release	*Redistributor Implementer Identification Register* on page 9-285	First documentation release for r1p2
Revision bits changed for EDPIDR2 to reflect r1p2 release	*External Debug Peripheral Identification Register 2* on page 11-373	First documentation release for r1p2
Revision bits changed for ROMPIDR2 to reflect r1p2 release	*ROM table Debug Peripheral Identification Register 2* on page 11-392	First documentation release for r1p2
Revision bits changed for PMPIDR2 to reflect r1p2 release	*Performance Monitors Peripheral Identification Register 2* on page 12-413	First documentation release for r1p2
Revision bits changed for CTIPIDR2 to reflect r1p2 release	*Peripheral Identification Register 2* on page 13-440	First documentation release for r1p2

Table E-5 Differences between Issue 0101-01 and Issue 0102-00 (continued)

Change	Location	Affects
REVISION bits changes for TRCIDR1 to reflect r1p2 release	*14.7.28 ID Register 1* on page 14-495	First documentation release for r1p2
TRCPIDR2 bits [7:4] value changed to reflect r1p2 release	*14.7.58 Peripheral Identification Registers* on page 14-524	First documentation release for r1p2
Revision bits changed for FPSID to reflect r1p2 release	*15.6.1 Floating-point System ID Register* on page 15-534	First documentation release for r1p2
Attributes for **MRPATTRx[4:2]** when set to `0b100` and `0b111` changed.	*A.7 MRP signals* on page Appx-A-558	First documentation release for r1p2
PMCR reset value corrected in register summary table	*12.2 PMU register summary* on page 12-401	First documentation release for r1p2
Flash interface short description modified	*7.7 Flash interface* on page 7-241	First documentation release for r1p2
ARUTIDFx signal width has been corrected	*Flash read address interconnect protection signals* on page Appx-A-574	First documentation release for r1p2
RUTIDFx signal width has been corrected	*Flash read data interconnect protection signals* on page Appx-A-575	First documentation release for r1p2
Timer events arrow removed from Cortex-R52 processor interfaces image	*Figure 1-2 Cortex-R52 processor interfaces* on page 1-15	First documentation release for r1p2
Description of Read issuing capability slightly modified	*Table 7-8 AXI4 master interface attributes* on page 7-228	First documentation release for r1p2
Description of Device accesses (data side) slightly modified	*Table 7-9 AXIM read address and read response channel identifiers* on page 7-230	First documentation release for r1p2
Added clarity to REGION bits description in HPSELR for 0-16 EL2-controlled MPU regions	*3.3.50 Hyp Protection Region Selection Register* on page 3-129	First documentation release for r1p2
Added clarity to HCR.AMO, HCR.IMO, and HCR.FMO bit descriptions	*3.3.39 Hyp Configuration Register* on page 3-111	First documentation release for r1p2
TESTR1 register description added	*3.3.98 Test Register 1* on page 3-194	First documentation release for r1p2
Note added to GIC functional description section	*9.2 GIC functional description* on page 9-262	First documentation release for r1p2
Functionality when there are write to the read-only flash port is documented	*7.7 Flash interface* on page 7-241	First documentation release for r1p2
Write acceptance and read acceptance capability values corrected	*7.8.2 AXIS characteristics* on page 7-244	First documentation release for r1p2
Note column added to bit description table for IMP_DCERRx and IMP_ICERRx register descriptions	• *3.3.21 Data Cache Error Record Registers 0 and 1* on page 3-95 • *3.3.57 Instruction Cache Error Record Registers 0 and 1* on page 3-141	First documentation release for r1p2

Table E-6 Differences between Issue 0102-00 and Issue 0103-00

Change	Location	Affects
First release for r1p3	• Document history table • *1.7 Product revisions* on page 1-27	First documentation release for r1p3
Added PMC-R52 to the block diagram	*1.1 About the Cortex®-R52 processor* on page 1-14	First documentation release for r1p3
Information on PMC-R52 added	*1.3.5 MBIST interface* on page 1-23	First documentation release for r1p3
Changed reset values of PMCNTENSET, PMCNTENCLR, PMOVSR, PMSWINC, PMSELR, PMCCNTR, PMXEVCNTR, PMINTENSET, PMINTENCLR, PMOVSSET to UNK	*3.2.11 c9 registers* on page 3-46	First documentation release for r1p3
Variant and Revision bits changed for MIDR to reflect r1p3 release	*3.3.69 Main ID Register* on page 3-156	First documentation release for r1p3
Information added on inner cacheability and outer cacheability domains	*7.3 Level-1 caches* on page 7-222	First documentation release for r1p3
Clarifications added on Write-Back and Write-Through transfers	*7.5.2 AXIM interface transfers* on page 7-228	First documentation release for r1p3
Updated the transaction types for the AXIM write address and write response channel identifiers	*7.5.4 AXIM transaction IDs* on page 7-230	First documentation release for r1p3
Clarification added on WRAP bursts	*7.7 Flash interface* on page 7-241	First documentation release for r1p3
Section on AXI privilege information added for the Flash interface	*7.7.1 AXI privilege information* on page 7-242	First documentation release for r1p3
References added	*7.9.4 Bus protection* on page 7-247	First documentation release for r1p3
Reset value changed for GICD_IIDR to reflect r1p3 release	*9.3.1 Distributor Registers (GICD)* on page 9-266	First documentation release for r1p3
Revision bits and reset value changed for GICD_IIDR to reflect r1p3 release	*Distributor Implementer Identification Register* on page 9-270	First documentation release for r1p3
Reset value changed for GICR_IIDR to reflect r1p3 release	*9.3.2 Redistributor Registers (GICR)* on page 9-282	First documentation release for r1p3
Revision bits and reset value changed for GICR_IIDR to reflect r1p3 release	*Redistributor Implementer Identification Register* on page 9-285	First documentation release for r1p3
Revision bits changed for EDPIDR2 to reflect r1p3 release	*External Debug Peripheral Identification Register 2* on page 11-373	First documentation release for r1p3
Revision bits changed for ROMPIDR2 to reflect r1p3 release	*ROM table Debug Peripheral Identification Register 2* on page 11-392	First documentation release for r1p3
Revision bits changed for PMPIDR2 to reflect r1p3 release	*Performance Monitors Peripheral Identification Register 2* on page 12-413	First documentation release for r1p3
Revision bits changed for CTIPIDR2 to reflect r1p3 release	*Peripheral Identification Register 2* on page 13-440	First documentation release for r1p3

Table E-6 Differences between Issue 0102-00 and Issue 0103-00 (continued)

Change	Location	Affects
Reset value changed for TRCIDR1 to reflect r1p3 release	*14.6 Register summary* on page 14-459	First documentation release for r1p3
REVISION bits and reset value changed for TRCIDR1 to reflect r1p3 release	*14.7.28 ID Register 1* on page 14-495	First documentation release for r1p3
TRCPIDR2 bits [7:4] value changed to reflect r1p3 release	*14.7.58 Peripheral Identification Registers* on page 14-524	First documentation release for r1p3
Revision bits changed for FPSID to reflect r1p3 release	*15.6.1 Floating-point System ID Register* on page 15-534	First documentation release for r1p3
Appendix on PMC-R52 added	*Appendix D PMC-R52* on page Appx-D-629	First documentation release for r1p3