Arm® Cortex®-A32 Processor

Revision: r1p0

Technical Reference Manual

arm

100241_0100_00_en

Arm® Cortex®-A32 Processor

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-00	18 March 2016	Confidential	First release for r0p0
0001-00	03 February 2017	Non-Confidential	First release for r0p1
0100-00	28 February 2019	Non-Confidential	First release for r1p0

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

http://www.arm.com

Contents
Arm® Cortex®-A32 Processor Technical Reference Manual

Preface

This preface introduces the *Arm® Cortex®-A32 Processor Technical Reference Manual*.

It contains the following:

About this book

This Technical Reference Manual is for the Cortex®-A32 processor. It provides reference documentation and contains programming details for registers. It also describes the memory system, the caches, the interrupts, and the debug features.

Product revision status

The r*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 manual is written for system designers, system integrators, and programmers who are designing or programming a System-on-Chip (SoC) that uses the Cortex®-A32 processor.

Using this book

This book is organized into the following chapters:

Part A Functional Description
This part describes the main functionality of the Cortex-A32 processor.

Chapter A1 Introduction
This chapter provides an overview of the Cortex-A32 processor and its features.

Chapter A2 Technical Overview
This chapter describes the structure of the Cortex-A32 processor.

Chapter A3 Clocks, Resets, and Input Synchronization
This chapter describes the clocks of the Cortex-A32 processor. It also describes the reset options.

Chapter A4 Power Management
This chapter describes the power domains and the power modes in the Cortex-A32 processor.

Chapter A5 Cache Behavior and Cache Protection
This chapter describes the CPU and SCU cache protection features of the Cortex-A32 processor.

Chapter A6 L1 Memory System
This chapter describes the L1 instruction cache and data cache.

Chapter A7 L2 Memory System
This chapter describes the L2 memory system and the *Snoop Control Unit* (SCU) that is tightly integrated with it.

Chapter A8 AXI Master Interface
This chapter describes the AXI master memory interface.

Chapter A9 ACE Master Interface
This chapter describes the ACE master interface.

Chapter A10 CHI Master Interface
This chapter describes the CHI master memory interface.

Chapter A11 ACP Slave Interface
This chapter describes the ACP slave interface.

Chapter A12 GIC CPU Interface
This chapter describes the *Generic Interrupt Controller* (GIC) CPU interface of the processor.

Part B Register Descriptions
This part describes the non-debug registers of the Cortex-A32 processor.

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

Chapter B2 GIC registers
This chapter describes the GIC registers.

Chapter B3 Generic Timer registers
This chapter describes the Generic Timer registers.

Part C Debug
This part describes the debug functionality and registers of the Cortex-A32 processor.

Chapter C1 Debug
This chapter describes the debug features of the processor.

Chapter C2 PMU
This chapter describes the *Performance Monitor Unit* (PMU) of the processor.

Chapter C3 ETM
This chapter describes the *Embedded Trace Macrocell* (ETM) of the processor.

Chapter C4 CTI
This chapter describes the cross-trigger components of the processor.

Chapter C5 Direct access to internal memory
This chapter describes the direct access to internal memory that caches and TLBs use.

Chapter C6 AArch32 debug registers
This chapter describes the debug registers in the AArch32 execution state and shows examples of how to use them.

Chapter C7 Memory-mapped debug registers
This chapter describes the debug memory-mapped registers and shows examples of how to use them.

Chapter C8 ROM table
This chapter describes the ROM table that debuggers can use to determine which components are implemented. It also describes the ROM table registers.

Chapter C9 PMU registers
This chapter describes the PMU registers.

Chapter C10 ETM registers
This chapter describes the ETM registers.

Chapter C11 CTI registers
This chapter describes the CTI registers.

Part D Appendices

Appendix A Signal Descriptions
This appendix describes the signals at the external interfaces of the processor.

Appendix B AArch32 UNPREDICTABLE Behaviors
The cases in which the Cortex-A32 processor implementation diverges from the preferred behavior described in Armv8 AArch32 UNPREDICTABLE behaviors.

Appendix C 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.

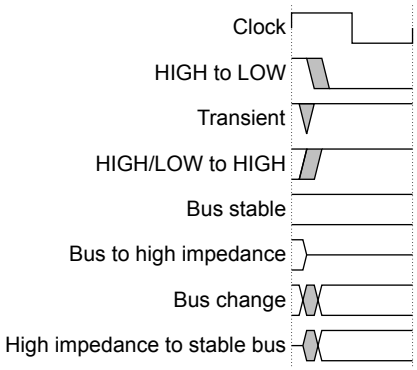

Figure 1 Key to timing diagram conventions

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 denotes an active-LOW signal.

Additional reading

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

Arm publications

- *Arm® AMBA® AXI and ACE Protocol Specification AXI, AXI4, and AXI4-Lite, ACE and ACE-Lite* (IHI 0022).
- *Arm® AMBA® APB Protocol Specification* (IHI 0024).
- *Arm® AMBA® ATB Protocol Specification* (IHI 0032).
- *Arm® Low Power Interface Specification Q-Channel and P-Channel Interfaces* (IHI 0068).
- *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* (DDI 0487).
- *Arm® Generic Interrupt Controller Architecture Specification* (IHI 0069).
- *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* (IHI 0064).
- *Arm® CoreSight™ Architecture Specification* (IHI 0029).
- *Arm® Cortex-A Series Programmer's Guide for Armv8-A* (DEN 0024).

The following confidential books are only available to licensees:

- *Arm® Cortex®-A32 Configuration and Sign-off Guide* (100244).
- *Arm® Cortex®-A32 Processor Integration Manual* (100245).
- *Arm® Cortex®-A32 Processor Cryptographic Extension Technical Reference Manual* (100242).
- *Arm® Cortex®-A32 Processor Advanced SIMD and Floating-point Support Technical Reference Manual* (100243).
- *Arm® AMBA® 5 CHI Protocol Specification* (IHI 0050).
- *Arm® Armv8 AArch32 UNPREDICTABLE behaviors* (PRD03-GENC-010544).

Other publications

This section lists relevant documents published by third parties:

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

——— **Note** ———

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

———————————

Feedback

Feedback on this product

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

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

Feedback on content

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

- The title *Arm Cortex-A32 Processor Technical Reference Manual*.
- The number 100241_0100_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.

———————————

Part A
Functional Description

Chapter A1
Introduction

This chapter provides an overview of the Cortex-A32 processor and its features.

It contains the following sections:

A1.1 About the Cortex®-A32 processor

The Cortex-A32 processor is a product designed to give mid-range instruction execution performance with low power consumption. It is highly configurable, allowing you to select features that are appropriate for the SoC in which it is used.

The major configuration options are:

- One to four Armv8-A compliant cores with automatic data cache coherency.
- One shared L2 cache.
- An AXI, ACE, or CHI system bus interface.

The processor also contains:

- Logic to help with power management.
- GICv4 interrupt capability.
- A complete CoreSight subsystem to support embedded debug in each core.
- An optional ACP that allows for I/O coherent operations with an external master, for example a DMA engine.

The following figure shows an example configuration with four cores, an L2 cache, and a CHI system bus interface.

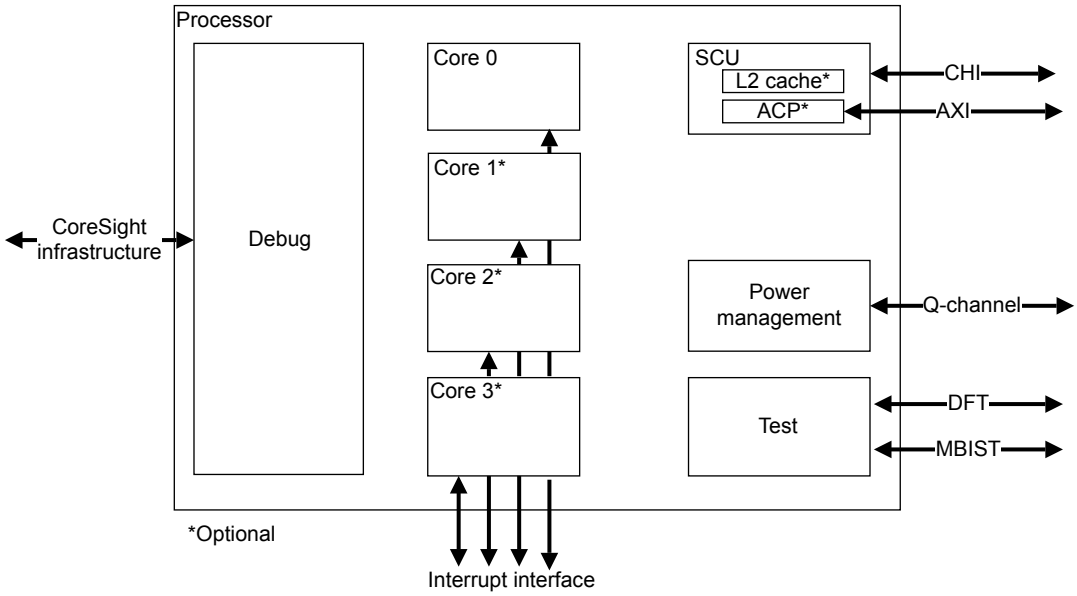

Figure A1-1 Example processor configuration

Related information

A1.2 Features on page A1-27
A1.3 Implementation options on page A1-28
A1.4 Supported standards and specifications on page A1-30
A1.5 Test features on page A1-31
A2.1 Components on page A2-36
A2.2 Interfaces on page A2-40

A1.2 Features

The Cortex-A32 processor includes the following features:

- Full implementation of the Armv8-A A32 and T32 instruction sets.
- The AArch32 execution state at all Exception levels (EL0 to EL3).
- In-order pipeline with direct and indirect branch prediction.
- Separate *Level 1* (L1) data and instruction side memory systems with a *Memory Management Unit* (MMU).
- *Level 2* (L2) memory system that provides cluster memory coherency.
- Optional L2 cache.
- Cache protection in the form of *Error Correction Code* (ECC) or parity on all RAM instances, except for the L2 victim RAM. There are two implementation options:
  - CPU cache protection.
  - *Snoop Control Unit* (SCU)-L2 cache protection.
- TrustZone®.
- Optional data engine that implements the Advanced SIMD and floating-point architecture support.
- Optional Cryptographic Extension. This architectural extension is only available if the data engine is present.
- Armv8 debug logic.
- *Performance Monitoring Unit* (PMU).
- Optional *Embedded Trace Macrocell* (ETM) that supports instruction trace only.
- Optional *Generic Interrupt Controller* (GIC) CPU interface to connect to an external distributor.
- Generic Timers supporting 64-bit count input from an external system counter.

Related information

A1.3 Implementation options on page A1-28
A1.4 Supported standards and specifications on page A1-30
A6.1 About the L1 memory system on page A6-86
A7.1 About the L2 memory system on page A7-92
A5.7 About cache protection on page A5-80

A1.3 Implementation options

The Cortex-A32 processor is highly configurable. Build-time configuration options make it possible to meet functional requirements with the smallest possible area and power.

In a configuration with more than one core, Advanced SIMD and floating-point support can be implemented on all cores, zero cores, or a subset of cores. If the Cryptographic Extension is included, it is included on all cores that implement Advanced SIMD and floating-point support. All cores have the same build-time configuration for other features.

The following table lists the implementation options for a core.

Table A1-1 Implementation options for a core

Feature	Range of options	Notes
L1 instruction cache size	• 8K • 16K • 32K • 64K	
L1 data cache size	• 8K • 16K • 32K • 64K	
CPU cache protection	• Included • Not included	• Not available if the L2 cache is implemented without SCU-L2 cache protection. • Also protects the L1 duplicate tags in the SCU.
GIC CPU interface	• Included • Not included	
ETM	• Included • Not included	
Advanced SIMD and floating-point support	• Included • Not included	Advanced SIMD and floating-point support is configured on a per-core basis.
Cryptographic Extension	• Included • Not included	If the Cryptographic Extension is included, then it is included on all cores that implement Advanced SIMD and floating-point support. There is no option to implement the Cryptographic Extension without Advanced SIMD and floating-point support.

The following table lists the implementation options at build time for the processor.

Table A1-2 Implementation options for the processor

Feature	Range of options	Notes
Number of cores	• 1 • 2 • 3 • 4	All cores have the same build-time configuration.
Main bus interface	• AMBA 4 AXI • AMBA 4 ACE • AMBA 5 CHI	

Table A1-2 Implementation options for the processor (continued)

Feature	Range of options	Notes
L2 cache	• Included • Not included	If it is present, all cores share one L2 cache.
L2 cache size	• 128K • 256K • 512K • 1024K	
L2 data RAM input latency	• 1 cycle • 2 cycles	
L2 data RAM output latency	• 2 cycles • 3 cycles	
SCU-L2 cache protection	• Included • Not included	Protects the L2 tag and L2 data RAMs with ECC.
Accelerator Coherency Port (ACP)	• Included • Not included	Part of the SCU-L2. If the processor does not include an L2 cache, it cannot implement the ACP.
Debug memory map	• v8 debug memory map • v7 debug memory map	

Related information

A2.2 Interfaces on page A2-40
A5.5 Invalidating or cleaning a cache on page A5-78
A6.1 About the L1 memory system on page A6-86
A7.1 About the L2 memory system on page A7-92
A5.7 About cache protection on page A5-80
Chapter A12 GIC CPU Interface on page A12-135
C1.6 Debug memory map on page C1-371
C3.1 About the ETM on page C3-386

A1.4 Supported standards and specifications

The Cortex-A32 processor implements the Armv8-A architecture and some architecture extensions. It also supports various interconnect, interrupt, timer, debug, and trace architectures.

Table A1-3 Compliance with standards and specifications

Architecture specification or standard	Version	Notes
Arm architecture	Armv8-A	• AArch32 execution state at all Exception levels. • A32 and T32 instruction sets.
Arm architecture extensions	• Advanced SIMD and floating-point support • Cryptographic Extension	• You cannot implement floating-point without Advanced SIMD. • You cannot implement the Cryptographic Extension without the Advanced SIMD and floating-point support.
Interconnect	• AMBA 4 AXI • AMBA 4 ACE • AMBA 5 CHI	You can also connect the processor to an AMBA 3 AXI interconnect.
Generic Interrupt Controller	v4	-
Generic Timer	Armv8-A	-
PMU	v3	-
Debug	Armv8	-
CoreSight	v2	-
Embedded Trace Macrocell	ETMv4	-

Related information

Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile
Arm® AMBA® 5 CHI Protocol Specification
Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite
Arm® Generic Interrupt Controller Architecture Specification
Arm® CoreSight™ Architecture Specification
Arm® ETM Architecture Specification, ETMv4

A1.5 Test features

The Cortex-A32 processor provides test signals that enable the use of both ATPG and MBIST to test the processor and its memory arrays.

Related information

A.20 DFT interface signals on page Appx-A-648

A.21 MBIST interface signals on page Appx-A-649

A1.6 Design tasks

The Cortex-A32 processor is delivered as a synthesizable Register Transfer Level (RTL) description in the Verilog HDL. Before you can use it, you must implement, integrate, and program it.

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

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

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

Programming
It is the last task. The system programmer develops the software to configure and initialize the processor and tests the application software.

The operation of the final device depends on the following:

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

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

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

A1.7 Product revisions

This section describes the differences in functionality between product revisions.

r0p0

First release.

r0p1

There are no functional changes in this release.

r1p0

New **CP15SDISABLE2** signal and new asymmetric floating-point/NEON feature.

Chapter A2
Technical Overview

This chapter describes the structure of the Cortex-A32 processor.

It contains the following sections:

A2.1 Components

The Cortex-A32 processor consists of:

- One to four cores, each with its own governor block. The governor block provides functionality that remains required when the core is in retention.
- An SCU-L2 memory system block. The SCU maintains data coherency between the L1 data caches and the L2 cache. It also connects the cores to an external memory system using an AXI, ACE, or CHI master interface. A mini-SCU replaces the SCU in configurations that do not require the SCU functionality. The mini-SCU is instantiated in implementations that are configured with a single core, no L2 cache, no CPU cache protection, and an AXI master interface.

The processor also integrates CoreSight components, and optionally integrates cache protection and the Cryptographic Extension.

The following figure shows a top-level functional diagram of the Cortex-A32 processor.

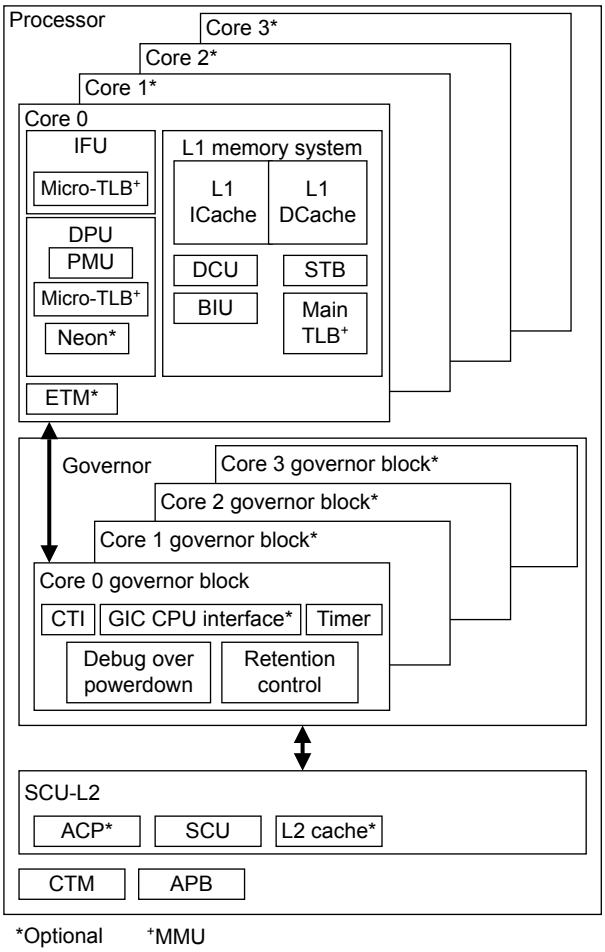

Figure A2-1 Cortex-A32 processor block diagram

Instruction Fetch Unit (IFU)

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

In implementations with CPU cache protection, parity bits protect the L1 Instruction cache data and tag RAMs by enabling the detection of any single-bit error. If an error is detected, the line is invalidated and fetched again.

Data Processing Unit (DPU)

The DPU decodes and executes instructions. It executes instructions that require data transfer to or from the memory system by interfacing to the *Data Cache Unit* (DCU). The DPU includes the *Performance Monitor Unit* (PMU), the Advanced SIMD and floating-point support, and the Cryptographic Extension.

PMU

The PMU provides six performance monitors that can be configured to gather statistics on the operation of each core and the memory system. The information can be used for debug and code profiling.

Advanced SIMD and floating-point support

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

Note

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

Cryptographic Extension

The optional Cortex-A32 processor Cryptographic Extension supports the Armv8 Cryptographic Extensions. It can be configured at implementation time and applies to all cores that implement Advanced SIMD and floating-point support. The Cryptographic Extension adds new instructions to Advanced SIMD that accelerate:

- *Advanced Encryption Standard* (AES) encryption and decryption.
- The *Secure Hash Algorithm* (SHA) functions SHA-1, SHA-224, and SHA-256.
- Finite field arithmetic used in algorithms such as *Galois/Counter Mode* and *Elliptic Curve Cryptography*.

Memory Management Unit (MMU)

The MMU provides fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes that are held in translation tables. These are loaded into the *Translation Lookaside Buffer* (TLB) when a location is accessed. The TLB entries include global and application specific identifiers to prevent context switch TLB flushes. They also include Virtual Machine Identifiers (VMIDs) to prevent TLB flushes on virtual machine switches by the hypervisor.

Micro TLBs

The first level of caching for the translation table information is a micro TLB of ten entries. It is implemented on each of the instruction and data sides. All main TLB related maintenance operations result in flushing both the instruction and data micro TLB.

Main TLB

A unified main TLB handles misses from the micro TLBs.

In implementations with CPU cache protection, parity bits protect the TLB RAMs by enabling the detection of any single-bit error. If an error is detected, the entry is flushed and fetched again.

L1 data-side memory system

The L1 data-side memory system includes the *Data Cache Unit* (DCU), the *Store Buffer* (STB), and the *Bus Interface Unit*.

DCU

The DCU manages all load and store operations.

In implementations with CPU cache protection, parity bits protect the L1 Data cache tag RAMs and dirty RAMs. The L1 Data cache data RAMs are protected using Error Correction Codes (ECC). The ECC scheme is Single Error Correct Double Error Detect (SECDED). The DCU includes a combined local and global exclusive monitor that is used by the Load-Exclusive/Store-Exclusive instructions.

STB

The STB holds store operations when they have left the load/store pipeline in the DCU and have been committed by the DPU. The STB can request access to the cache RAMs in the DCU, request the BIU to initiate linefills, or request the *Bus Interface Unit* (BIU) to write out the data on the external write channel. External data writes are through the SCU.

The STB is also used to queue maintenance operations before they are broadcast to other cores in the processor.

BIU

The BIU contains the SCU interface and buffers to decouple the interface from the L1 Data cache and STB. The BIU and the SCU always operate at the processor frequency.

Governor

The governor block, outside the core, includes all functions that must remain operating while a core is in retention mode.

GIC CPU interface

The GIC CPU interface is a memory-mapped interface through which a core receives an interrupt. The GIC Distributor can read and write the GIC CPU interface registers even while the core is in retention mode.

Generic timer

The Generic Timer has an interface to an external system counter. It provides a consistent view of time, which can be used to schedule events and trigger interrupts. It is also used by the retention circuits in the processor.

L2 Memory System

The L2 memory system contains the L2 cache pipeline and all the logic that maintains memory coherence between the cores in the cluster.

SCU

The SCU connects the cores to the external memory system through the master memory interface. It also maintains data cache coherency between the cores and arbitrates L2 requests from the cores.

mini-SCU

The mini-SCU replaces the SCU in certain uniprocessor configurations that do not require data cache coherency with other masters in the system. That is, implementations that are configured to have a single core, no L2 cache, no CPU cache protection, and an AXI interface. The mini-SCU bridges between the master interface of the core and the AXI master interface of the processor.

L2 cache

Each Cortex-A32 cluster can include an optional L2 cache that participates in the coherency protocol. Each L2 cache is 8-way set associative, supports 64-byte cache lines, and has a configurable cache RAM size between 128KB and 1MB.

ACP

The ACP interface cannot be configured without an L2 cache because it reuses buffering and data paths implemented for the L2 cache to achieve optimal efficiency. The main advantage of the ACP interface is its ability to allocate data in the L2 cache RAMs.

Debug and trace components

Cross-trigger

The *Cross Trigger Matrix* (CTM) combines the CoreSight *Cross Trigger Interface* (CTI) channel signals from all the cores so that a single cross trigger channel interface is presented in the Cortex-A32 processor. This module can combine up to four internal channel interfaces corresponding to each core along with one external channel interface.

Debug ROM

The Cortex-A32 processor has a debug ROM which is a CoreSight feature.

ETM

The ETM trace unit is a build-time configuration option. This module performs real-time instruction flow tracing that complies with the ETM architecture.

Related information

A2.2 Interfaces on page A2-40
A6.1 About the L1 memory system on page A6-86
A7.1 About the L2 memory system on page A7-92
Chapter A3 Clocks, Resets, and Input Synchronization on page A3-45
Chapter A4 Power Management on page A4-53

A2.2 Interfaces

The Cortex-A32 processor has several interfaces to connect it to a SoC.

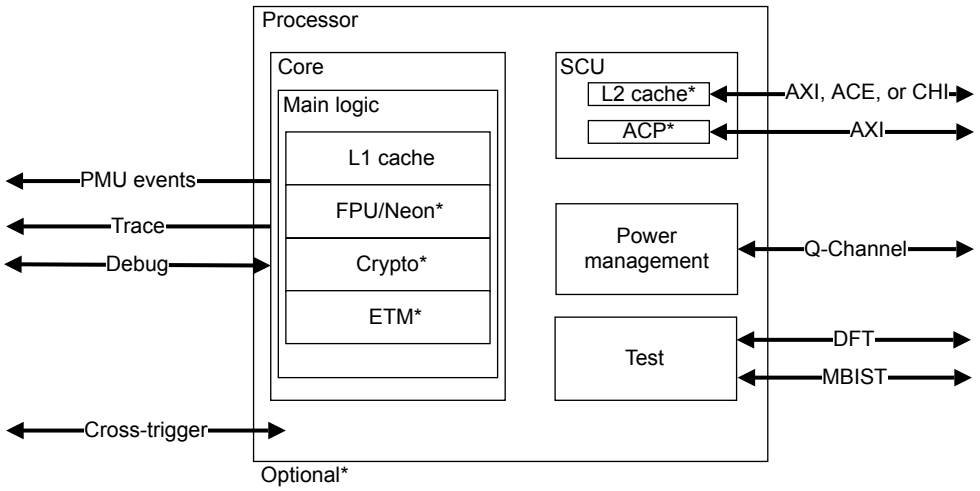

Figure A2-2 Interfaces

Table A2-1 Cortex-A32 interfaces

Purpose	Technology	Notes
PMU events		Performance events provide useful information on the operation of the processor that you can use for debug and code profiling. A subset of available performance events is exported on the PMU event bus.
Trace	ATB	Optional Outputs trace information for debugging. The ATB interface is compatible with the CoreSight architecture.
Memory	AXI, ACE, or CHI	ACE can also be used with AXI peripherals.
ACP	AXI	Optional This slave interface reduces software cache maintenance operations when the cores share memory regions with other masters and allows other masters to allocate data into the L2 cache. It allows an external master to make coherent requests to shared memory, but it does not support cache maintenance, coherency, barrier, or DVM transactions.
Debug	APB	Allows access to debug registers and resources, for example, to set watchpoints and breakpoints.
Cross-trigger	CTI	This external interface is connected to the CoreSight CTI corresponding to each core through a simplified CTM.
Design for Test (DFT)		Allows an industry standard *Automatic Test Pattern Generation* (ATPG) tool to test logic.
Memory Built-In Self Test (MBIST)		Provides support for manufacturing test of the memories embedded in the Cortex-A32 processor.
Power management	Q-channel	Enables communication to an external power controller.

Related information

Chapter A9 ACE Master Interface on page A9-107
Chapter A10 CHI Master Interface on page A10-119
Chapter A8 AXI Master Interface on page A8-99

C1.2 Debug access on page C1-367
C2.1 About the PMU on page C2-376
C3.1 About the ETM on page C3-386
C4.1 About the cross-trigger on page C4-394
Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, and ACE and ACE-Lite
Arm® AMBA® 5 CHI Protocol Specification
Arm® CoreSight™ Architure Specification

A2.3 About system control

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

The main functions of the system registers are:

- Overall system control and configuration.
- MMU configuration and management.
- Configuration and management of the L1 and the L2 caches.
- System performance monitoring.
- GIC configuration and management.

The system registers are accessible in the AArch32 Execution state. Some of the system registers are accessible through the memory-mapped or external debug interfaces. The **CP15SDISABLE2** input disables write access to the following system registers:

- System Control Register (SCTLR).
- Translation Table Base Register 1 (TTBR1).
- Translation Table Base Control Register (TTBCR).
- Domain Access Control Register (DACR).
- Primary Region Remap Register (PRRR).
- Normal Memory Remap Register (NMRR).
- Memory Attribute Indirection Register 0 (MAIR0).
- Memory Attribute Indirection Register 1 (MAIR1).
- Vector Base Address Register (VBAR).
- Monitor Vector Base Address Register (MVBAR).
- Interrupt Controller Monitor System Register Enable (ICC_MSRE).
- Non-Secure Address Control Register (NSACR).
- Secure Debug Enable Register (SDER).
- Secure Debug Control Register (SDCR).

Related reference

B1.1 AArch32 register summary on page B1-144

A2.4 About the Generic Timer

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

The Cortex-A32 processor does not include the system counter. This resides in the SoC. The system counter value is distributed to the Cortex-A32 processor with a synchronous binary encoded 64-bit bus, **CNTVALUEB[63:0]**.

Because **CNTVALUEB[63:0]** is generated from a system counter that typically operates at a slower frequency than the processor clock, **CLKIN**, the **CNTCLKEN** input is provided. **CNTCLKEN** is registered inside the processor and then used as a clock enable for **CNTVALUEB[63:0]**. This allows a multicycle path to be applied to the **CNTVALUEB[63:0]**. The following figure shows the interface.

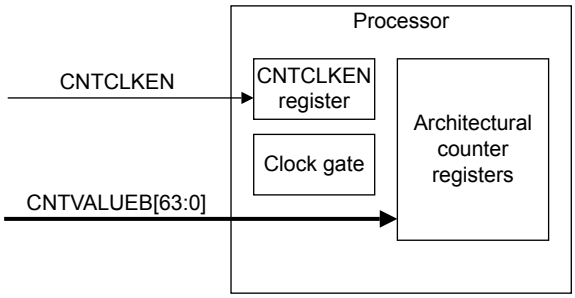

Figure A2-3 Generic Timer interface

The value on **CNTVALUEB[63:0]** is required to be stable whenever the internally registered version of the **CNTCLKEN** clock enable is asserted. **CNTCLKEN** must be synchronous and balanced with respect to **CLKIN** and must toggle at integer ratios of the processor **CLKIN**.

Related information

A.6 Generic Timer signals on page Appx-A-625

A2.5 About the memory model

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

The processor can store words in memory in big-endian or little-endian format. Instructions are always little-endian.

Related information

ARM® Architecture Reference Manual ARMv8, for ARMv8-A architecture profile

Chapter A3
Clocks, Resets, and Input Synchronization

This chapter describes the clocks of the Cortex-A32 processor. It also describes the reset options.

It contains the following sections:

A3.1 Clocks

The Cortex-A32 processor has a single clock input, **CLKIN**. All cores in the Cortex-A32 processor and the SCU are clocked with a distributed version of **CLKIN**.

The Cortex-A32 processor has the following clock enable signals:

- **PCLKENDBG**.
- **ACLKENM**.
- **ACLKENS**.
- **SCLKEN**.
- **ATCLKEN**.
- **CNTCLKEN**.

For more information, see the *Arm® Cortex®-A32 Processor Integration Manual.*

A3.2 Input synchronization

The Cortex-A32 processor synchronizes certain input signals. The SoC can present these inputs asynchronously. All other external signals must be synchronous with reference to **CLKIN**.

Input signals that the Cortex-A32 processor synchronizes:

- **nCORERESET**.
- **nCPUPORESET**.
- **nFIQ**.
- **nIRQ**.
- **nL2RESET**.
- **nMBISTRESET**.
- **nPRESETDBG**.
- **nREI**.
- **nSEI**.
- **nVFIQ**.
- **nVIRQ**.
- **nVSEI**.
- **CLREXMONREQ**.
- **CPUQREQn**.
- **CTICHOUTACK**.
- **CTIIRQACK**.
- **DBGEN**.
- **EDBGRQ**.
- **EVENTI**.
- **L2FLUSHREQ**.
- **L2QREQn**.
- **NEONQREQn**.
- **NIDEN**.
- **SPIDEN**.
- **SPNIDEN**.

Input signals that the Cortex-A32 processor synchronizes under certain conditions:

- **CTICHIN**.

The synchronized **CTICHIN** input signals are used only if the **CISBYPASS** input signal is deasserted LOW. If the **CISBYPASS** signal is asserted HIGH the **CTICHIN** synchronizers are not used, and the SoC must present the **CTICHIN** synchronously to **CLKIN**.

A3.3 Resets

The Cortex-A32 processor has active-LOW reset input signals that can be asynchronously asserted HIGH to LOW, or deasserted LOW to HIGH.

nCPUPORESET[CN:0]

Where **CN** is the number of cores minus one.

Power On reset signals. These primary, cold reset signals initialize all resettable registers in the corresponding core, including debug registers and ETM registers.

nCORERESET[CN:0]
These primary reset signals initialize all resettable registers in the corresponding core, not including debug registers and ETM registers.

nPRESETDBG
This single, processor-wide signal resets the integrated CoreSight components that connect to the external **PCLK** domain, such as debug logic.

nL2RESET
This single, processor-wide signal resets all resettable registers in the L2 memory system and the logic in the SCU or mini-SCU.

nMBISTRESET
An external MBIST controller can use this signal to reset the entire SoC. The **nMBISTRESET** signal resets all resettable registers in the cluster, for entry into, and exit from, MBIST mode.

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

In general, the reset time only requires three processor clock cycles.

——— **Note** ———

The application of a retention state can affect how long reset assertion is required. You must hold the reset signal active-LOW until the power returns and the unit or processor is ready for the reset to take effect if:

- The Advanced SIMD and floating-point unit of a core undergoing a reset is in retention state.
- A core that is being reset is in retention state.

The time that is taken for retention exit and the behavior of the power controller varies by partner and by implementation.

———————————

The following table describes the valid reset signal combinations. All other combinations of reset signals are illegal. In the table, n designates the core that is reset.

Table A3-1 Valid reset combinations

Reset combination	Signals	Value	Description
Cluster cold reset	**nCPUPORESET[CN:0]**	all = 0	All logic is held in reset. **nCORERESET** can be asserted, but is not required.
	nCORERESET[CN:0]	all = X	
	nPRESETDBG	0	
	nL2RESET	0	
	nMBISTRESET	1	
Cluster cold reset with debug active	**nCPUPORESET[CN:0]**	all = 0	All cores are held in reset so they can be powered up. The L2 is held in reset, but must remain powered up. This enables external debug over power down for the cluster. **nCORERESET** can be asserted, but is not required.
	nCORERESET[CN:0]	all = X	
	nPRESETDBG	1	
	nL2RESET	0	
	nMBISTRESET	1	
Individual core cold reset with debug active	**nCPUPORESET[CN:0]**	[n] = 0	Individual core is held in reset, so that the core can be powered up. This enables external debug over power down for the core that is held in reset. **nCORERESET** can be asserted, but is not required.
	nCORERESET[CN:0]	[n] = X	
	nPRESETDBG	1	
	nL2RESET	1	
	nMBISTRESET	1	
Individual core warm reset with trace and debug active	**nCPUPORESET[CN:0]**	[n] = 1	Individual core is held in reset.
	nCORERESET[CN:0]	[n] = 0	
	nPRESETDBG	1	
	nL2RESET	1	
	nMBISTRESET	1	
Debug logic reset	**nCPUPORESET[CN:0]**	all = 1	Cluster debug logic is held in reset.
	nCORERESET[CN:0]	all = 1	
	nPRESETDBG	0	
	nL2RESET	1	
	nMBISTRESET	1	

Table A3-1 Valid reset combinations (continued)

Reset combination	Signals	Value	Description
MBIST reset	**nCPUPORESET[CN:0]**	all = 1	All logic is held in reset.
	nCORERESET[CN:0]	all = 1	
	nPRESETDBG	1	
	nL2RESET	1	
	nMBISTRESET	0	
Normal state	**nCPUPORESET[CN:0]**	all = 1	No logic is held in reset.
	nCORERESET[CN:0]	all = 1	
	nPRESETDBG	1	
	nL2RESET	1	
	nMBISTRESET	1	

Warm reset

The following figure shows the Warm reset sequence for the Cortex-A32 processor.

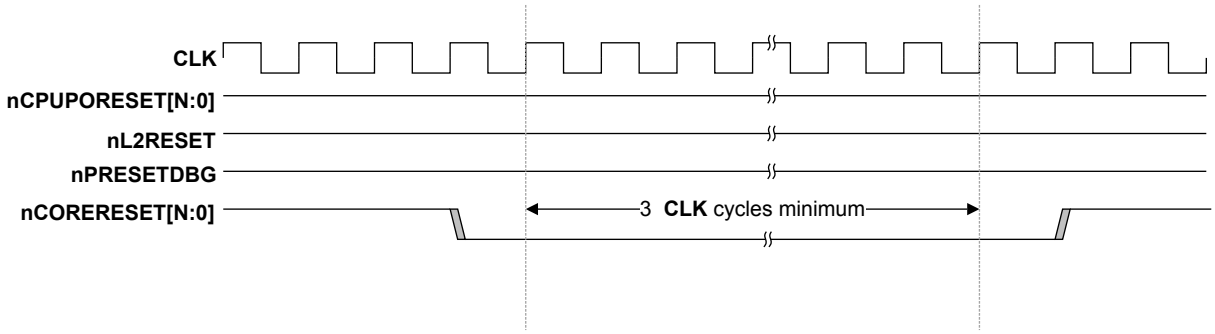

Figure A3-1 Warm reset timing

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

For individual processor Warm reset:

- You must apply steps 1 to 6 in the core powerdown sequence, see *A4.6 Powering down an individual core* on page A4-61, and wait until **STANDBYWFI** is asserted, indicating that the core is idle, before asserting **nCORERESET** for that core.
- **nCORERESET** for that core must assert for at least 3 CLK cycles.
- **nL2RESET** must not assert while any individual core is active.
- **nPRESETDBG** must not assert while any individual core is actively being debugged in normal operating mode.

Note

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

WARMRSTREQ and DBGRSTREQ

When the Reset Request bit in the RMR register is set to 1, the processor asserts the **WARMRSTREQ** signal and the SoC reset controller can use this request to trigger a Warm reset of the core. An external debugger might also request a Warm reset of the core by asserting **DBGRSTREQ**.

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

You must apply steps *1* on page A4-61 to *6* on page A4-61 in the core powerdown sequence in *A4.6 Powering down an individual core* on page A4-61, and wait until **STANDBYWFI** asserts indicating the processor is idle, before asserting **nCORERESET** for that core. **nCORERESET** must satisfy the timing requirements described in the Warm reset section.

Chapter A4
Power Management

This chapter describes the power domains and the power modes in the Cortex-A32 processor.

It contains the following sections:

A4.1 Power domains

A core or a processor can support different power domains. Each power domain has valid and accepted power states.

The Cortex-A32 processor provides mechanisms and support to control both dynamic and static power dissipation. The individual cores in the Cortex-A32 processor support four main levels of power management which correspond to the power domains shown in the following table:

Table A4-1 Power domain description

Power domain	Description
PDMINERVA	Includes the SCU, the optional L2 cache control logic, and debug registers that are described as being in the debug domain.
PDL2	Includes the L2 data RAM, L2 tag RAM, L2 victim RAM, and the SCU duplicate tag RAM.
PDCPU<n>	Includes the optional Advanced SIMD and floating-point support, the L1 cache and TLB RAMs, and the debug registers that are described as being in the processor domain. n is 0, 1, 2, or 3. It represents core 0, core 1, core 2, or core 3. If a core is not present, the corresponding power domain is not present.
PDCPUADVSIMD<n>	Represents the Advanced SIMD and floating-point block of core n. n is 0, 1, 2, or 3. It represents core 0, core 1, core 2, or core 3. If a core is not present, the corresponding power domain is not present.

The separate PDMINERVA and PDL2 power domains can remain active even when all the cores are powered down. It means that the processor can continue to accept snoops from external devices to access the L2 cache.

The following figure shows an example of the domains embedded in a *System-on-Chip* (SoC) power domain.

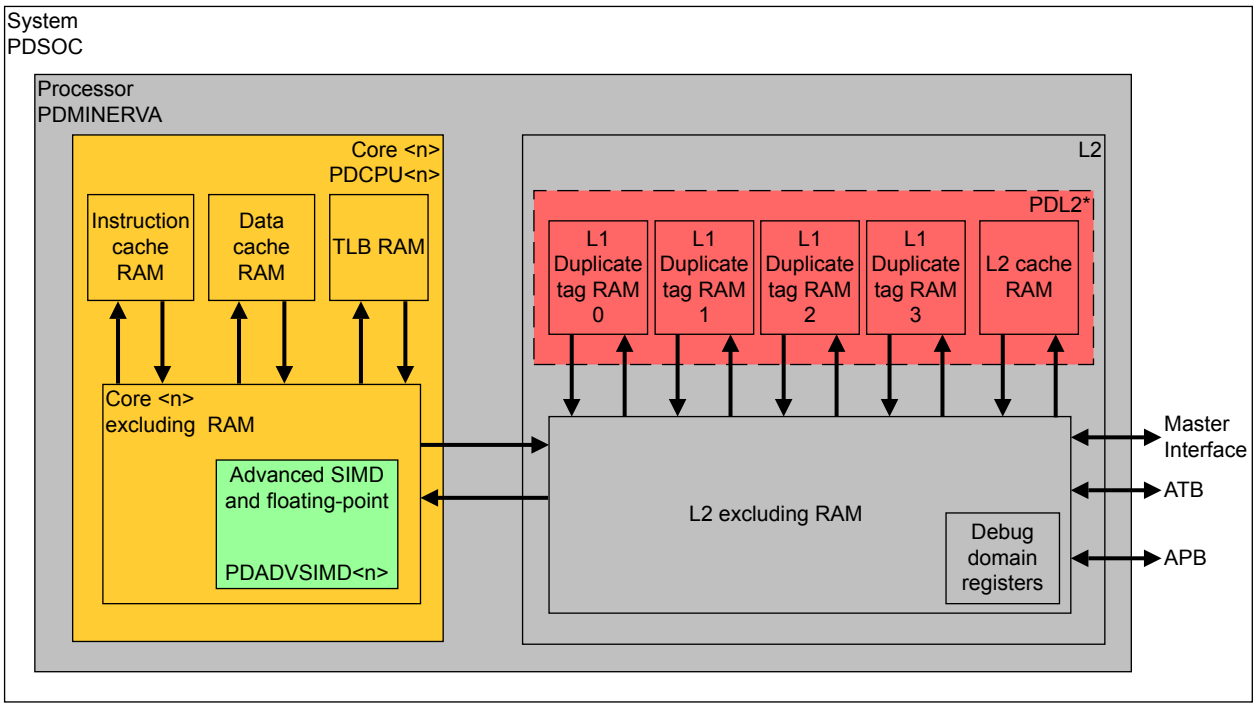

Figure A4-1 Power domains

The power domains can be controlled independently to give different combinations of powered-up and powered-down domains. However, only some powered-up and powered-down domain combinations are valid and supported.

Table A4-2 Power state description

Power state	Description
Off	Block is power gated
Ret	Logic or RAM retention power only
On	Block is active

The following tables show the supported power domain states for the processor.

Caution

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

Table A4-3 Supported processor power states

Power domains			Description
PDMINERVA	**PDL2**	**PDCPU<n>**	
Off	Off	Off	Processor off.
Off	On/Ret	Off	L2 cache dormant mode.

Table A4-3 Supported processor power states (continued)

Power domains			Description
PDMINERVA	**PDL2**	**PDCPU<n>**	
On	Ret	See *Table A4-4 Supported core power states* on page A4-56	Processor on, L2 RAMs retained. All cores either off or in WFx. This is an L2 RAM retention entry or residency condition.
On	Ret	See *Table A4-4 Supported core power states* on page A4-56	Processor on, L2 RAMs retained. At least one core running. This is a transient condition.
On	On	See *Table A4-4 Supported core power states* on page A4-56	Processor on, SCU/L2 RAMs active.

The following table describes the supported power domain states for individual cores. The power domain state in each core is independent of all other cores.

Table A4-4 Supported core power states

Power domains		Description
PDCPU	**PDADVSIMD**	
Off	Off	Core off.
On	On	Core on. Advanced SIMD and floating-point on.
On	Ret	AdvSIMD retention. Advanced SIMD and floating-point in retention.
Ret	Ret	Core retention. Core logic and Advanced SIMD and floating-point in retention.

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

A4.2 Power modes

The processor supports the following power modes:

Normal mode
This is the normal mode of operation in which all of the processor functionality is available. The Cortex-A32 processor uses gated clocks to disable inputs to unused functional blocks. Only the logic used to perform an operation consumes any dynamic power.

Standby mode

When a Cortex-A32 core is in standby mode, it is architecturally clock gated at the top of the clock tree. Each core in the cluster can be put in standby mode separately from the other cores, by executing a *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) instruction.

L2 standby mode
When all the cores are in standby mode and the L2 memory system is idle.

Individual core shutdown mode

The PDCPU power domain for an individual core is shut down and the state held in this domain is lost.

Cluster shutdown mode
The PDMINERVA, PDL2, and all PDCPU power domains are shut down and the state held in these domains is lost.

Dormant mode (optional)
All the cores and L2 control logic are powered down while the L2 cache RAMs are powered up and retain state. The RAM blocks that remain powered up during Dormant mode are:

- L2 tag RAMs.
- L2 data RAMs.
- L2 victim RAM.

Retention mode

Contact Arm for information about retention state.

A4.3 Core Wait for Interrupt

Programmers can use the *Wait for Interrupt* (WFI) instruction to cause the core to enter a low-power state.

Wait for Interrupt is a feature of the Armv8-A architecture that puts the core in a low-power state by disabling most of the clocks in the core while keeping the core powered up. Apart from a small dynamic power overhead on the logic to enable the core to wake up from WFI low-power state, this reduces the power drawn to static leakage current only.

When executing the `WFI` instruction, the core waits for all instructions in the core to retire before entering the idle or 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 received the required data or responses from the L2 memory system:

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

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

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

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

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

On entry into WFI low-power state, **STANDBYWFI** for that core is asserted. Assertion of **STANDBYWFI** guarantees that the core is in idle and low-power state. **STANDBYWFI** continues to assert even if the clocks in the core are temporarily enabled because of an L2 snoop request, cache or TLB maintenance operation, or an APB access.

STANDBYWFI does not indicate completion of L2 memory system transactions initiated by the processor. All Cortex-A32 processor implementations contain an L2 memory system. This includes implementations without an L2 cache.

A4.4 Core Wait for Event

A core can use the *Wait for Event* (WFE) instruction to cause the core to enter a low-power state.

Wait for Event (WFE) is a feature of the Armv8-A architecture. It can be used by a locking mechanism based on events to put the core in a low-power state by disabling most of the clocks in the core while keeping the core powered-up. Apart from a small dynamic power overhead on the logic to enable the core to wake up from WFE low-power state, this reduces the power drawn to static leakage current only.

When executing the `WFE` instruction, the core waits for all instructions in the core to complete before entering the idle or low-power state.

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

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

- An L2 snoop request that must be serviced by the core L1 Data cache.
- A cache or TLB maintenance operation that must be serviced by the core L1 Instruction cache, data cache, or TLB.
- An APB access to the debug or trace registers residing in the core power domain.

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

On entry into WFE low-power state, **STANDBYWFE** for that core is asserted. Assertion of **STANDBYWFE** guarantees that the core is in idle and low-power state. **STANDBYWFE** continues to assert even if the clocks in the core are temporarily enabled because of an L2 snoop request, cache or TLB maintenance operation, or an APB access.

CLREXMON request and acknowledge signaling

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

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

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

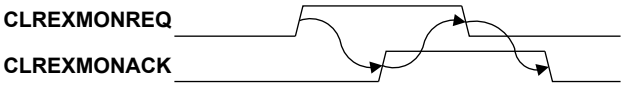

Figure A4-2 CLREXMON request and acknowledge handshake

A4.5 L2 Wait for Interrupt

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

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

- All cores are in WFI low-power state and therefore, the **STANDBYWFI** output for each core is asserted. Assertion of all the cores **STANDBYWFI** outputs guarantees that all the cores are in idle and low-power state. All clocks in the cores, with the exception of a small amount of clock wakeup logic, are disabled.
- If configured with ACE, the SoC asserts the input pin **ACINACTM** to idle the AXI master interface. It indicates that no snoop requests will be made from the external memory system.
- If configured with a CHI interface, the SoC asserts the input pin **SINACT** to idle the CHI master interface. It indicates that no snoop requests will be made from the external memory system.
- If configured with an ACP interface, the SoC asserts the **AINACTS** input pin to idle the ACP interface. It indicates that the SoC sends no more transactions on the ACP interface.

When the L2 memory system completes the outstanding transactions for AXI, ACE, or CHI interfaces, it can then enter the L2 WFI low-power state. On entry into L2 WFI low-power state, **STANDBYWFIL2** is asserted. Assertion of **STANDBYWFIL2** guarantees that the L2 memory system is idle and does not accept new transactions.

Exit from L2 WFI low-power state occurs on one of the following events:

- A physical IRQ or FIQ interrupt.
- A debug event.
- Powerup or warm reset.

When a core exits permanently from WFI low-power state, **STANDBYWFI** for that core is deasserted. When the L2 memory system logic exits from WFI low-power state, **STANDBYWFIL2** is deasserted. The SoC must continue to assert **ACINACTM** or **SINACT** until **STANDBYWFIL2** has deasserted.

The following figure shows the L2 WFI timing for a 4-core configuration.

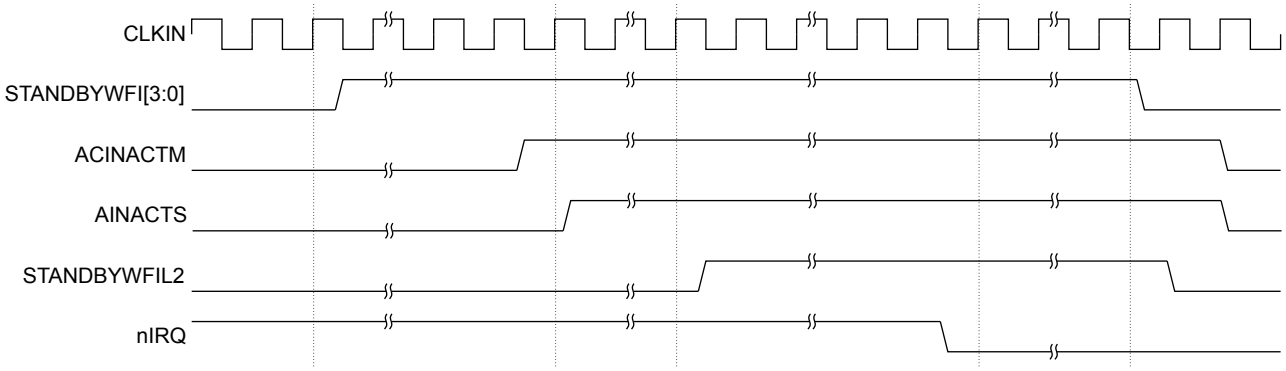

Figure A4-3 L2 Wait For Interrupt timing

A4.6 Powering down an individual core

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

To power down the core, apply the following sequence:

Procedure

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

A4.7 Powering up an individual core

To power up an individual core, apply the following sequence:

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

A4.8 Powering down the processor without system driven L2 flush

When powering down the processor, the PDMINERVA, PDL2, and PDCPU power domains are shut down and all state is lost. In this section, the lead core is defined as the last core to switch off.

To power down the processor, apply the following sequence. For device powerdown, all operations on a lead core must occur after the equivalent step on all non-lead cores.

Procedure

1. Ensure all non-lead cores are in shutdown mode, see *A4.6 Powering down an individual core* on page A4-61.
2. Follow steps *1* on page A4-61 and *2* on page A4-61 in *A4.6 Powering down an individual core* on page A4-61.
3. If the ACP interface is configured, ensure that any master connected to the interface does not send new transactions, then assert **AINACTS**.
4. Clean and invalidate all data from the L2 Data cache.
5. Follow steps *3* on page A4-61 to *10* on page A4-61 in *A4.6 Powering down an individual core* on page A4-61.
6. In an ACE configuration, assert **ACINACTM** or, in a CHI configuration, assert **SINACT**. Then, wait until the **STANDBYWFIL2** output is asserted to indicate that the L2 memory system is idle. All Cortex-A32 processor implementations contain an L2 memory system, including implementations without an L2 cache. This applies to implementations that use the mini-SCU and implementations that use the SCU.
7. Activate the cluster output clamps.
8. Remove power from the PDMINERVA and PDL2 power domains.

A4.9 Powering up the processor without system driven L2 flush

When powering down the processor, the PDMINERVA, PDL2, and PDCPU power domains are shut down and all state is lost.

To power up the processor, apply the following sequence:

Procedure

1. For each core in the cluster, assert **nCPUPORESET** LOW.
2. Assert **nL2RESET** LOW and hold **L2RSTDISABLE** LOW.
3. Apply power to the PDMINERVA and PDL2 domains while keeping the signals described in steps *1* on page A4-64 and *2* on page A4-64 LOW.
4. Release the cluster output clamps.
5. Continue a normal cold reset sequence.

A4.10 Powering down the processor with system driven L2 flush

When powering down the processor, the PDMINERVA, PDL2, and PDCPU power domains are shut down and all state is lost.

To power down the cluster, apply the following sequence:

Procedure

1. Ensure all cores are in shutdown mode, see *A4.6 Powering down an individual core* on page A4-61.
2. If the ACP interface is configured, ensure that any master connected to the interface does not send new transactions, then assert **AINACTS**. This is necessary to prevent ACP transactions from allocating new entries in the L2 cache while the hardware cache flush is occurring.
3. Assert **L2FLUSHREQ** HIGH.
4. Hold **L2FLUSHREQ** HIGH until **L2FLUSHDONE** is asserted.
5. Deassert **L2FLUSHREQ**.
6. In an ACE configuration, assert **ACINACTM** or, in a CHI configuration, assert **SINACT**. Then, wait until the **STANDBYWFIL2** output is asserted to indicate that the L2 memory system is idle. All Cortex-A32 processor implementations contain an L2 memory system, including implementations without an L2 cache. This applies to implementations that use the mini-SCU and implementations that use the SCU.
7. Activate the cluster output clamps.
8. Remove power from the PDMINERVA and PDL2 power domains.

A4.11 Powering up the processor with system driven L2 flush

To power up the processor, apply the following sequence:

Procedure

1. For each core in the cluster, assert **nCPUPORESET** LOW.
2. Assert **nL2RESET** LOW and hold **L2RSTDISABLE** LOW.
3. Apply power to the PDMINERVA and PDL2 domains while keeping the signals described in steps *1* on page A4-66 and *2* on page A4-66 LOW.
4. Release the cluster output clamps.
5. Continue a normal cold reset sequence.

A4.12 Entering Dormant mode

The processor can enter Dormant mode if certain requirements are met.

To support Dormant mode, you must ensure:

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

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

To enter Dormant mode, apply the following sequence:

Procedure

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

    When the L2 has completed the outstanding transactions for the AXI master and slave interfaces, **STANDBYWFIL2** is asserted to indicate that L2 memory system is idle. All Cortex-A32 processor implementations contain an L2 memory system, including implementations without an L2 cache.
11. When **STANDBYWFI** and **STANDBYWFIL2** are asserted for all cores, the cluster is ready to enter Dormant mode. This applies to implementations that use the mini-SCU as well as implementations that use the SCU.
12. Activate the L2 cache RAM input clamps.
13. Remove power from the PDCPU and PDMINERVA power domains.

A4.13 Exiting Dormant mode

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

To exit Dormant mode, apply the following sequence:

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

A4.14 Event communication using WFE or SEV

An external agent can use the **EVENTI** pin to participate in a WFE or SEV event communication with the Cortex-A32 processor.

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

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

A4.15 Communication to the Power Management Controller

Communication between the Cortex-A32 processor and the system power management controller can be performed using one or both of the:

- *A4.16 STANDBYWFI[3:0] and STANDBYWFIL2 signals* on page A4-71.
- *A4.17 Q-channel* on page A4-72.

A4.16 STANDBYWFI[3:0] and STANDBYWFIL2 signals

The **STANDBYWFI[n]** signal indicates when an individual core is in idle and low-power state. The power management controller can remove power from an individual core when **STANDBYWFI[n]** is asserted.

The **STANDBYWFIL2** signal indicates when all individual cores and the L2 memory system are in idle and low-power state. A power management controller can remove power from the Cortex-A32 processor when **STANDBYWFIL2** is asserted. See *A4.8 Powering down the processor without system driven L2 flush* on page A4-63 and *A4.10 Powering down the processor with system driven L2 flush* on page A4-65 for more information.

The Cortex-A32 processor includes a minimal L2 memory system in configurations without an L2 cache. Therefore, the power management controller must always wait for assertion of **STANDBYWFIL2** before removing power from the Cortex-A32 processor. This applies to configurations that use the mini-SCU and configurations that use the SCU.

The following figure shows how **STANDBYWFI[3:0]** and **STANDBYWFIL2** correspond to individual cores and the Cortex-A32 processor.

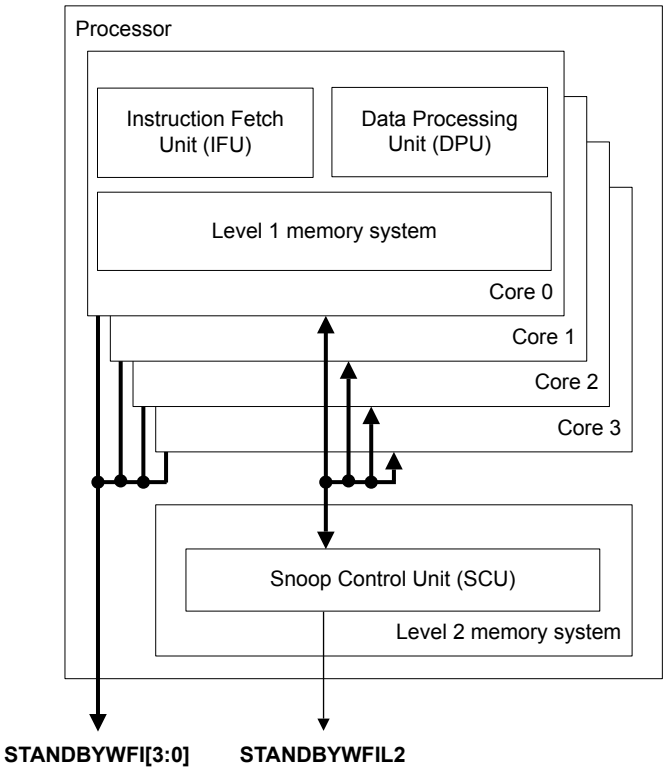

Figure A4-4 STANDBYWFI[3:0] and STANDBYWFIL2 signals

A4.17 Q-channel

Q-channel enables:

- The controller to manage entry to, and exit from, a device quiescent state. Quiescence management is typically of, but not restricted to, clock gated, and power gated retention states, of the device or device partitions.
- The capability to indicate a requirement for exit from the quiescent state. The associated signaling can contain contributions from other devices in the same power domain.
- Optional device capability to deny a quiescence request.
- Safe asynchronous interfacing across clock domains.

For more information, see the *Low Power Interface Specification: Arm Q-Channel and P-Channel Interfaces*.

Chapter A5
Cache Behavior and Cache Protection

This chapter describes the CPU and SCU cache protection features of the Cortex-A32 processor.

It contains the following sections:

A5.1 Cached memory types

The processor can cache data and instructions that meet certain memory attribute criteria.

L1 instruction cache

When the L1 instruction cache is enabled, it caches the following memory types:

- Normal, Inner Write-Back.
- Normal, Inner Write-Through.

L1 data cache

When the L1 data cache is enabled, it can cache the following memory types:

- Normal, Inner Write-Back, Outer Write-Back.

Data might not be allocated if:

- The data is for a non-temporal load.
- The transient hint is set.
- The no-allocate hint is set.
- The processor is in read allocate mode.

L2 cache

If the L2 cache is present and enabled, it can cache the following memory types:

- Normal, Inner Write-Back, Outer Write-Back.

Instruction cache lines are allocated into the L2 cache when they are fetched from the external memory system.

Data cache lines are allocated into the L2 cache when they are evicted from an L1 data cache.

Related information

B1.42 CPU Auxiliary Control Register on page B1-202

A5.2 Coherency between data caches with the MOESI protocol

The processor uses the MOESI protocol to maintain data cache coherency between multiple cores. The DCU stores the MOESI state of the cache line in the tag and dirty RAMs.

MOESI describes the state in which a shareable line can be in an L1 data cache.

Table A5-1 MOESI and AMBA mapping

MOESI	AMBA	Description
Modified	UniqueDirty	The line is in only this cache and is dirty.
Owned	SharedDirty	The line is possibly in more than one cache and is dirty.
Exclusive	UniqueClean	The line is in only this cache and is clean.
Shared	SharedClean	The line is possibly in more than one cache and is clean.
Invalid	Invalid	The line is not in this cache.

Data coherency is enabled only when the CPUECTLR.SMPEN bit is set. You must set the SMPEN bit before enabling the data cache. If you do not, then the cache is not coherent with other cores and data corruption could occur.

Related information

A5.6 About read allocate mode on page A5-79
B1.43 CPU Extended Control Register on page B1-206
C5.3 Encoding for tag and data in the L1 data cache on page C5-400

A5.3 Cache misses, unexpected cache hits, and speculative fetches

The L1 and L2 caches handle problematic cache accesses in predefined ways.

Cache miss

On a cache miss, the processor performs Critical Word First filling of the cache.

Unexpected cache hits

If the cache reports a hit on a memory location that is marked as Non-Cacheable or Device, this is called an unexpected cache hit. In this architecturally UNPREDICTABLE case, the cache might return incorrect data because of the following configurations or settings:

- Improper translation table configuration because the caches are physically addressed.
- The cache is disabled.

Non-Cacheable or Device accesses do not use the result of a cache lookup and therefore ignore any unexpected cache hit.

Speculative fetches

Because there can be several unresolved branches in the pipeline, there is no guarantee that the processor executes an instruction. Instruction fetches are therefore speculative. A branch or exceptional instruction in the code stream can cause a pipeline flush and discard the fetched instructions. Because of the prefetching behavior, you must not place read-sensitive devices in the same page as code. Pages with Device memory type attributes are treated as Non-Cacheable Normal Memory when accessed by instruction fetches. You must use the XN (Execute Never) bit in the page table descriptor for a memory region to stop speculative instructions fetches when such memory region contains read-sensitive devices. To avoid speculative fetches to read-sensitive devices when address translation is disabled, these devices must be separated from code in the physical memory map.

Related information

Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile

A5.4 Disabling a cache

When the instruction cache is disabled:

- Instruction fetches do not access the L1 instruction cache or the L2 unified cache.
- Instruction cache maintenance operations execute normally.
- All instruction fetches to cacheable memory are treated as if they were non-cacheable. This means that instruction fetches might not be coherent with caches in other cores and software must take account of this.

When the data cache is disabled:

- Load and store instructions do not access the L1 data cache or L2 unified cache.
- Instruction fetches do not access the L2 unified cache.
- Data cache maintenance operations to the L1 data cache and the L2 unified cache execute normally.
- All load and store instructions to cacheable memory are treated as if they were non-cacheable. This means that they are not coherent with the caches in this core or the caches in other cores and software must take account of this.

You cannot disable the L1 data cache and the L2 unified cache independently because the same enable bit controls both of them.

If the instruction cache is enabled and the data cache is disabled, then instruction fetches to cacheable memory continue to access the L1 instruction cache but they do not access the L2 unified cache.

Related information

B1.105 System Control Register on page B1-324

A5.5 Invalidating or cleaning a cache

The processor automatically invalidates caches on reset unless suppressed with the **DBGL1RSTDISABLE** or **L2RSTDISABLE** pins. It is therefore not necessary for software to invalidate the caches on start-up.

DCIMVAC operations perform an invalidate of the target address. If the data is dirty within the cluster then a clean is performed before the invalidate.

DCISW operations perform both a clean and invalidate of the target set/way. The values of HCR.SWIO and HCR_EL2.SWIO have no effect.

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

A5.6 About read allocate mode

The processor supports read allocate mode, also called write streaming mode, both for the L1 and the L2 cache.

Read allocate mode is a performance and power-saving optimization for writing a large block of data.

Read allocate mode for the L1 data cache

The L1 data cache supports only a Write-Back policy. It normally allocates a cache line on either a read miss or a write miss, although you can alter this by changing the inner cache allocation hints in the page tables. However, there are some situations where allocating on writes is not wanted, such as executing the C standard library `memset()` function to set a large block of memory to a known value. Writing large blocks of data like this can pollute the cache with unnecessary data. It can also waste power and reduce performance if a linefill must be performed only to discard the linefill data because the entire line was subsequently written by the `memset()` function. Therefore, the core includes logic to detect when the processor has written a full cache line before the linefill completed. If this situation is detected on a threshold number of consecutive linefills, the core switches to read allocate mode.

When the L1 data cache is in read allocate mode:

- Loads behave as normal and can still cause linefills.
- Writes still look up in the cache but if they miss, they write out to L2 rather than starting a linefill.

More than the specified number of linefills might be observed on the master interface, before the core detects that three full cache lines have been written and switches to read allocate mode.

The core continues in read allocate mode until it detects either a cacheable write burst to L2 that is not a full cache line, or there is a load to the same line as is currently being written to L2.

To configure the L1 read allocate mode threshold, use CPUACTLR.L1RADIS.

Read allocate mode for the L2 cache

The L2 cache enters read allocate mode after a threshold number of consecutive cache line sized writes to L2 are detected.

When the L2 cache is in read allocate mode:

- Loads behave as normal and can still cause linefills.
- Writes still lookup in the cache but if they miss, they write out to L3 rather than starting a linefill.

L2 read allocate mode continues until there is a cacheable write burst that is not a full cache line, or there is a load to the same line as is currently being written to L3.

To configure the L2 read allocate mode threshold, use CPUACTLR.RADIS.

Related information

B1.42 CPU Auxiliary Control Register on page B1-202

A5.7 About cache protection

The processor protects against soft errors that result in a RAM bitcell temporarily holding the incorrect value, by writing a new value to the RAM to correct the error. If the error is a hard error that is not corrected by writing to the RAM, for example a physical defect in the RAM, then the processor might get into a livelock because it continually detects and then tries to correct the error.

Some RAMs have *Single Error Detect* (SED) capability, others have *Single Error Correct, Double Error Detect* (SECDED) capability. The L1 data cache dirty RAM is *Single Error Detect, Single Error Correct* (SEDSEC). The processor can make progress and remain functionally correct when there is a single bit error in any RAM. If there are multiple single bit errors in different RAMs or within different protection granules within the same RAM, then the processor also remains functionally correct.

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

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

If there are three or more bit errors the processor might or might not detect the errors, depending on the RAM and the position of the errors within the RAM.

The cache protection feature of the processor has a minimal performance impact when no errors are present. When the processor detects an error, it stalls the access that caused the error while it corrects the error. When the correction is complete, the access either continues with the corrected data or is retried. If the access is retried, it either hits in the cache again with the corrected data or misses in the cache and re-fetches the data from a lower level cache or from main memory.

Table A5-2 Cache protection behavior of each RAM

RAM	Protection type	Configuration option	Protection granule	Correction behavior
L1 instruction cache tag	Parity, SED	CPU_CACHE_PROTECTION	31 bits	The processor invalidates both lines in the cache set then refetches the requested line from the L2 cache or external memory.
L1 instruction cache data	Parity, SED	CPU_CACHE_PROTECTION	18 bits	
TLB	Parity, SED	CPU_CACHE_PROTECTION	31 bits or 51 bits	The processor invalidates the entry and starts a new pagewalk to refetch it.
L1 data cache tag	Parity, SED	CPU_CACHE_PROTECTION	32 bits	The processor cleans the line and invalidates it from the L1 cache. It uses SCU duplicate tags to get the correct address. It refetches the line from the L2 cache or external memory.
L1 data cache data	ECC, SECDED	CPU_CACHE_PROTECTION	32 bits	The processor cleans the line and invalidates it from the L1 cache. It corrects single bit errors as part of the eviction. It refetches the line from the L2 cache or external memory.
L1 data cache dirty	Parity, SEDSEC	CPU_CACHE_PROTECTION	1 bit	The processor cleans the line and invalidates it from the L1 cache. It corrects single bit errors as part of the eviction. Only the dirty bit is protected. The other bits are performance hints, therefore do not cause a functional failure if they are incorrect.

Table A5-2 Cache protection behavior of each RAM (continued)

RAM	Protection type	Configuration option	Protection granule	Correction behavior
SCU L1 duplicate tag	ECC, SECDED	CPU_CACHE_PROTECTION	33 bits	The processor rewrites the tag with the correct value and retries access. If the error is uncorrectable then the processor invalidates the tag.
L2 tag	ECC, SECDED	SCU_CACHE_PROTECTION	32 bits	
L2 victim	None	-	-	The victim RAM only serves as a performance hint. It does not result in a functional failure if the contents are incorrect.
L2 data	ECC, SECDED	SCU_CACHE_PROTECTION	64 bits	The processor corrects the data inline and might stall access for an additional cycle or two while the correction takes place. After correction, the processor might evict the line.

If a correctable ECC error occurs after the first data cache access of a load instruction that takes multiple cycles to complete, and if one of the following conditions has taken place:

- A hardware breakpoint, watchpoint, or vector catch has been set since the first execution that is triggered on re-execution.
- The page tables have been modified since the first execution. This resulted in an instruction or data abort trap being taken on re-execution.

then the register file is updated with data that was successfully read before the correctable ECC error occurred.

A5.8 Error reporting

The processor reports detected errors, including errors that are successfully corrected and those that cannot be corrected, in the CPUMERRSR or L2MERRSR registers. It also signals them on the **PMUEVENT** bus.

If multiple errors occur on the same clock cycle then only one of them is reported. Errors that cannot be corrected, and therefore might result in data corruption, also cause an abort or external pin to be asserted, so that software can be aware that there is an error and can either attempt to recover or can restart the system. Such errors are:

- Uncorrectable errors in the L2 data RAM when read by an instruction fetch, TLB pagewalk, or load instruction, might result in a precise data abort or prefetch abort.
- Uncorrectable errors in the L2 data RAM when read by a fetch into the L1 data cache from a load, store or preload instruction, or by the hardware prefetcher, might result in an asynchronous exception.
- Uncorrectable errors in the L1 or L2 data RAMs when the line is being evicted from a cache results in the processor asserting the **nINTERRIRQ** signal. This might be because of a natural eviction, a cache maintenance operation, or a snoop.
- Uncorrectable errors in the L2 tag RAMs or SCU L1 duplicate tag RAMs result in the processor asserting the **nINTERRIRQ** signal.
- When **nINTERRIRQ** is asserted it remains asserted until the error is cleared by a write of 0 to the L2 internal asynchronous error bit of the L2ECTLR register.
- Arm recommends that the **nINTERRIRQ** signal is connected to the interrupt controller so that an interrupt or system error is generated when the signal is asserted.

When a dirty cache line with an error on the data RAMs is evicted from the processor, the write on the master interface still takes place, however if the error is uncorrectable then:

- On AXI and ACE, the write strobes are not set, therefore the incorrect data is not written externally.
- On CHI, the strobes are set, but the response field indicates that there is a data error.

When a snoop hits on a line with an uncorrectable data error the data is returned, if required by the snoop, but the snoop response indicates that there is an error.

If a snoop hits on a tag that has an uncorrectable error, then it is treated as a snoop miss, because the error means that it is unknown if the cache line is valid or not.

In some cases it is possible for an error to be counted more than once. For example, multiple accesses might read the location with the error before the line is evicted as part of the correction process.

Related reference

B1.44 CPU Memory Error Syndrome Register on page B1-208
B1.93 L2 Extended Control Register on page B1-299
B1.94 L2 Memory Error Syndrome Register on page B1-301

A5.9 Error injection

To support testing of error handling software, the processor provides the capability to force double-bit errors to be injected into the L1 D-cache data RAMs, the L2 data RAMs, and the L2 tag RAMs.

Error injection on the L1 D-cache data RAMs is enabled by setting the CPUACTLR.L1DEIEN bit. While this bit is set, double-bit errors are injected on all writes to the L1 D-cache data RAMs for the first word of each 32-byte region. This corresponds to bytes with an address where bits [4:2] are `0b000`. The L1 D-cache RAMs can be written to because of:

- Explicit stores from the core.
- Cache line fetches into the cache, as a result of:
  - Load instructions.
  - Store instructions.
  - Preload instructions.
  - Data prefetches.
  - Pagewalks.

Error injection on the L2 data RAMs is enabled by setting the L2ACTLR.L2DEIEN bit. While this bit is set, double-bit errors are injected on all writes to the L2 cache data RAMs. The L2 data RAMs can be written to because of:

- Explicit stores from one of the cores.
- Instruction fetches or prefetches.
- Evictions from the L1 Data cache.
- ACP accesses.

Error injection on the L2 tag RAMs is enabled by setting the L2ACTLR.L2TEIEN bit. While this bit is set, double-bit errors are injected on all writes to the L2 tag RAMs. The L2 cache tag RAMs can be written because of:

- Explicit stores from one of the cores.
- L2 allocations caused by instruction fetches or prefetches.
- Evictions from the L1 Data cache.
- ACP accesses.
- Snoop operations.
- Cache maintenance instructions.

Chapter A6
L1 Memory System

This chapter describes the L1 instruction cache and data cache.

It contains the following sections:

A6.1 About the L1 memory system

The L1 memory system includes several power-saving and performance-enhancing features. These include separate instruction and data caches, which can be configured independently during implementation to sizes of 8KB, 16KB, 32KB, or 64KB.

MMU

The MMU provides fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes held in translation tables. These are loaded into the *Translation Lookaside Buffer* (TLB) when a location is accessed. The key features are:

- 10-entry fully-associative instruction micro TLB.
- 10-entry fully-associative data micro TLB.
- 2-way set-associative 512-entry unified main TLB.
- 2-way set-associative 64-entry walk cache.
- 2-way set-associative 64-entry IPA cache.

L1 instruction-side memory system

The L1 instruction-side memory system provides an instruction stream to the DPU. The key features are:

- A dedicated instruction cache that:
  - is virtually indexed and physically tagged.
  - is 2-way set associative.
  - is configurable to be 8KB, 16KB, 32KB, or 64KB.
  - uses a cache line length of 64 bytes.
  - uses a pseudo-random replacement policy.
- A 128-bit read interface to the L2 memory system.
- Dynamic program flow prediction.

L1 data-side memory system

The L1 data-side memory system responds to load and store requests from the DPU. It also responds to snoop requests that have been forwarded by the SCU from other cores or external masters. The key features are:

- A dedicated data cache that:
  - is physically indexed and physically tagged.
  - is 4-way set associative.
  - is configurable to be 8KB, 16KB, 32KB, or 64KB.
  - uses a cache line length of 64 bytes.
  - uses a pseudo-random replacement policy.
- A 128-bit read and 256-bit write interface to the L2 memory system.
- A 64-bit read and 64-bit write path to the DPU.
- Read buffers that service the DCU, the IFU, and the TLB.
- Support for three outstanding data cache misses.
- Support for eight outstanding linefill requests.
- A merging store buffer.
- An internal exclusive monitor.
- An automatic data prefetch engine.
- Write stream detection and optimization (read allocate mode).

Related information

A6.4 About the internal exclusive monitor on page A6-89
A6.5 About data prefetching on page A6-90
A5.6 About read allocate mode on page A5-79

A6.2 TLB Organization

This section describes the organization of the TLB.

Micro TLB

The first level of caching for the translation table information is a micro TLB of ten entries that is implemented on each of the instruction and data sides. All main TLB related maintenance operations result in flushing both the instruction and data micro TLB.

Main TLB

A unified main TLB handles misses from the micro TLBs. It has a 512-entry, 2-way, set-associative structure and supports all VMSAv8 block sizes, except 1GB. If it fetches a 1GB block, the TLB splits it into 512MB blocks and stores the appropriate block for the lookup.

Accesses to the main TLB take a variable number of cycles. The number of cycles depends on the following criteria:

- Competing requests from each of the micro TLBs.
- The TLB maintenance operations in flight.
- The different page size mappings in use.

IPA cache RAM

The *Intermediate Physical Address* (IPA) cache RAM holds mappings between intermediate physical addresses and physical addresses. Only Non-secure EL1 and EL0 stage 2 translations use this cache. When a stage 2 translation is completed, it is updated and checked whenever a stage 2 translation is required.

Similarly to the main TLB, the IPA cache RAM can hold entries for different sizes.

Walk cache RAM

The walk cache RAM holds the result of a stage 1 translation up to but not including the last level. If the stage 1 translation results in a section or larger mapping then nothing is placed in the walk cache.

The walk cache holds entries fetched from Secure and Non-secure state.

A6.3 Program flow prediction

Program flow prediction is always enabled when the MMU is enabled by setting the appropriate control bit in the relevant system control register.

As a general rule, the flow prediction hardware predicts all branch outcomes regardless of the addressing mode. For example, it predicts the outcomes of the following branch types:

- Conditional branches.
- Unconditional branches.
- Indirect branches that are associated with procedure call and return instructions.
- Branches that switch between A32 and T32 states.

However, the flow prediction hardware does not predict the branch outcomes for the following instructions:

- Data-processing instructions that use the PC as a destination register.
- The `BXJ` instruction.
- Exception return instructions.

A T32 instruction set branch that is normally encoded as unconditional can be made conditional by inclusion in an *If-Then* (IT) block. Then it is treated as a conditional branch.

Return stack predictions

The return stack stores the return address and the A32 or T32 instruction set of the instruction after a procedure call type branch instruction. This address is equal to the link register value stored in r14. The following instructions cause a return stack push if predicted:

- `BL`.
- `BLX` (immediate).
- `BLX` (register).

In AArch32 state, the following instructions cause a return stack pop if predicted:

- `BX`
- `LDR pc, [r13], #imm`
- `LDM r13, {…pc}`
- `LDM r13, {…pc}!`

Related information

B1.105 System Control Register on page B1-324

A6.4 About the internal exclusive monitor

The internal exclusive monitor is a state machine that manages Load-Exclusive or Store-Exclusive instructions, and Clear-Exclusive (CLREX) instructions. Its two states are open and exclusive.

You can use the Load-Exclusive or Store-Exclusive accesses and the Clear-Exclusive instructions to construct semaphores to ensure synchronization between different processes running on the core and also between different cores that are using the same coherent memory locations for the semaphore.

A Load-Exclusive instruction tags a small block of memory for exclusive access. The size of the tagged block is defined by CTR.ERG as 16 words, one cache line.

A Load-exclusive/Store-exclusive instruction is any one of the following:

- Any instruction that has a mnemonic starting with `LDREX`, `STREX`, `LDAEX`, or `STLEX`.

A Load-Exclusive instruction that causes a transaction with **ARLOCKM** for AXI/ACE, or **Excl** for CHI, set to HIGH is expected to receive an **EXOKAY** response. An **OKAY** response to a transaction with **ARLOCKM** for AXI/ACE, or **Excl** for CHI, set to HIGH indicates that exclusive accesses are not supported at the address of the transaction and causes a Data Abort exception to be taken with a Data Fault Status Code of:

- `0b110101`, when using the long descriptor format.
- `0b10101`, when using the short descriptor format.

A Load-Exclusive instruction causes **ARLOCKM** for AXI to be set to HIGH if the memory attributes are:

- Device.
- Normal Inner Non-cacheable and Outer Non-cacheable.
- Normal Inner is not Write-Back or Outer is not Write-Back, and Inner Shareable.
- Normal Inner is not Write-Back or Outer is not Write-Back, and Outer Shareable.

A Load-Exclusive instruction causes **ARLOCKM** for ACE or **Excl** for CHI, to be set to HIGH if the memory attributes are:

- Device.
- Normal Inner Non-cacheable and Outer Non-cacheable.
- Normal Inner Write-Back, Outer Write-Back, Outer Shareable, and **BROADCASTOUTER** is set to HIGH.
- Normal Inner Write-Back, Outer Write-Back, Inner Shareable, and **BROADCASTINNER** is set to HIGH.
- Normal Inner is not Write-Back or Outer is not Write-Back, and Inner Shareable.
- Normal Inner is not Write-Back or Outer is not Write-Back, and Outer Shareable.

In cases where there is an intervening store operation between an exclusive load and an exclusive store from the same core, the intermediate store does not produce any direct effect on the internal exclusive monitor. The local monitor is in the Exclusive Access state after the exclusive load, remains in the Exclusive Access state after the store, and returns to the Open Access state only after the exclusive store, a `CLREX` instruction, or an exception return.

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

Related reference

A8.3 AXI transactions on page A8-102
A9.4 ACE transactions on page A9-111
A10.5 CHI transactions on page A10-125

A6.5 About data prefetching

This section describes the software and hardware data prefetching behavior for the processor.

Preload instructions

`PLD` instructions look up in the cache and start a linefill if they miss and are to a cacheable address. These instructions retire as soon as their linefill has started, they do not wait for data to be returned. This enables other instructions to execute while the linefill continues in the background.

`PLDW` instructions are similar to `PLD`, except that if they miss, the linefill causes data to be invalidated in other cores and masters so that the line is ready for writing.

`PLI` instructions are treated as NOPs.

Automatic data prefetching and monitoring

The L1 data-side memory system implements an automatic prefetcher that monitors cache misses in the core. When a pattern is detected, the automatic prefetcher starts linefills in the background. The prefetcher recognizes a sequence of data cache misses at a fixed stride pattern that lies in four cache lines, plus or minus. Any intervening stores or loads that hit in the data cache do not interfere with the recognition of the cache miss pattern.

The CPUACTLR enables you to:

- Deactivate the prefetcher.
- Alter the sequence length required to trigger the prefetcher.
- Alter the number of outstanding requests that the prefetcher can make.

Use `PLD` instructions for data prefetching where short sequences or irregular pattern fetches are required.

Related information

B1.42 CPU Auxiliary Control Register on page B1-202

Chapter A7
L2 Memory System

This chapter describes the L2 memory system and the *Snoop Control Unit* (SCU) that is tightly integrated with it.

It contains the following sections:

A7.1 About the L2 memory system

In most configurations, the L2 memory system consists of an integrated SCU that connects the cores in a cluster, an optional, tightly-coupled L2 cache, and an optional ACP interface. In single core, AXI configurations that do not include CPU cache protection, ACP, or an L2 cache, the SCU is replaced with a more area-efficient mini-SCU.

The same system register control bit enables the L1 data cache and the L2 cache.

SCU

The SCU maintains coherency between the L1 and L2 data caches in the processor. It also arbitrates requests for the L2 cache and the AXI, ACE, or CHI master interface.

A coherent request from a core is one that checks for data in the L1 data caches and, if present, the L2 cache. The SCU might send a request to another core to retrieve or invalidate data, or both, depending on the type of coherent request. This request is referred to as a snoop request. If the processor is implemented with an ACE or CHI master interface then the SCU can issue coherent requests on the master interface, which might result in snoop requests being sent to other masters in the system. The SCU might also receive snoop requests from other masters.

The SCU can handle direct cache-to-cache transfers between cores without having to read or write any data to the external memory system. Cache line migration enables dirty cache lines to be moved between cores, and there is no requirement to write back transferred cache line data to the external memory system.

Each core has tag and dirty RAMs that contain the state of the cache line in the L1 data cache. Rather than sending a snoop request to each core to access these for each coherent request, the SCU contains a set of duplicate tags that allows it to check the contents of each L1 data cache. The duplicate tags filter coherent requests so that a snoop request is only sent to a core if the coherent request hits in the corresponding duplicate tags. The duplicate tags are also used to filter snoop requests from the external memory system. This allows the cores and the system to function efficiently even with a high volume of requests.

The SCU does not support hardware management of coherency of the instruction caches. Instruction cache linefills perform coherent reads, however, there is no coherency management of data held in the instruction cache.

mini-SCU

The mini-SCU replaces the SCU in certain uniprocessor configurations that do not require data cache coherency with other masters in the system. That is, implementations that are configured to have a single CPU, no L2 cache, no CPU cache protection, and an AXI interface. The mini-SCU bridges between the master interface of the core and the AXI master interface of the processor.

L2 cache

Data cache lines are allocated to the L2 cache only when evicted from the L1 memory system, not when first fetched from the system. The only exceptions to this rule are for memory marked with the inner transient hint, or for non-temporal loads that are only ever allocated to the L2 cache. The L1 cache can prefetch data from the system, without data being evicted from the L2 cache.

Instruction cache lines are allocated to the L2 cache when fetched from the system and can be invalidated during maintenance operations.

The L2 cache is 8-way set associative. The L2 cache tags are looked up in parallel with the SCU duplicate tags. If both the L2 tag and SCU duplicate tag hit, a read accesses the L2 cache in preference to snooping one of the other cores.

L2 RAMs are invalidated automatically at reset unless the **L2RSTDISABLE** signal is set HIGH when the **nL2RESET** signal is deasserted.

Further features of the L2 cache are:

- Configurable size of 128KB, 256KB, 512KB, and 1MB.
- Fixed line length of 64 bytes.
- Physically indexed and tagged.
- Optional ECC protection.
- A pseudo-LRU replacement policy.

ACP

Optional 128-bit wide I/O coherent ACP interface that can allocate to the L2 cache.

Master memory interface

The SCU connects the cores to the external memory system through a 128-bit-wide master memory interface that uses ACE, CHI, or AXI technology. The memory interface supports integer ratios of the processor clock period up to and including 1:1 and a 40-bit physical address range.

The L2 memory system has two abort mechanisms, a synchronous one and an asynchronous one.

Related information

A7.5 Handling of external aborts on page A7-97
Chapter A9 ACE Master Interface on page A9-107
Chapter A10 CHI Master Interface on page A10-119
Chapter A8 AXI Master Interface on page A8-99
Chapter A11 ACP Slave Interface on page A11-129

A7.2 Snoop and maintenance requests

In implementations that include an ACE or CHI master interface, the SCU controls snoop and maintenance requests to the external memory system with the **BROADCASTINNER**, **BROADCASTOUTER**, and **BROADCASTCACHEMAINT** configuration inputs.

Table A7-1 Control pins for snoop and maintenance requests

Signal	Setting	Description
BROADCASTINNER	1	The inner shareability domain extends beyond the processor. Inner Shareable snoop and maintenance operations are broadcast externally.
	0	The inner shareability domain does not extend beyond the processor.
BROADCASTOUTER	1	The outer shareability domain extends beyond the processor. Outer shareable snoop and maintenance operations are broadcast externally.
	0	The outer shareability domain does not extend beyond the processor.
BROADCASTCACHEMAINT	1	There are external downstream caches and maintenance operations are broadcast externally.
	0	There are no downstream caches external to the processor.

If you set the **BROADCASTINNER** pin to HIGH you must also set the **BROADCASTOUTER** pin to HIGH.

In a system that contains a Cortex-A32 processor and another processor in a big.LITTLE configuration, you must ensure the **BROADCASTINNER** and **BROADCASTOUTER** pins on both processors are set to HIGH so that both processors are in the same Inner Shareable domain.

Cacheable loads and stores to a shareability domain, that does not extend beyond the processor can allocate data to the L1 and L2 caches. However, they do not make coherent requests on the master for these accesses. Instead, they use only ReadNoSnoop or WriteNoSnoop transactions. This always includes non-shareable memory, and might include inner shareable and outer shareable memory, depending on the setting of the **BROADCASTINNER** and **BROADCASTOUTER** pins.

If the system sends a snoop to the Cortex-A32 processor for an address that is present in the L1 or L2 cache, but the line in the cache is in a shareability domain that does not extend beyond the cluster, then the snoop is treated as missing in the cluster.

A7.3 Support for memory types

The processor simplifies the coherency logic by downgrading some memory types.

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

The attributes provided on **ARCACHE** or **AWCACHE** in AXI and ACE configurations or MemAttr and SnpAttr in CHI configurations are these downgraded attributes and indicate how the interconnect must treat the transaction.

A7.4 Memory type information exported from the processor

The processor makes attribute information about memory types available through signals for interconnects or bus protocols that require such information.

Some interconnects or bus protocols might require more information about the memory type and, for these cases, the cluster exports the memory attribute information from the translation tables stored in the TLB. These signals are for information only, and do not form part of the AXI, ACE, or CHI protocols.

- In an AXI or ACE configuration, there is a **RDMEMATTR** bus for the read channel and a **WRMEMATTR** bus for the write channel.
- In a CHI configuration, there is a single **REQMEMATTR** bus.

Table A7-2 Bus encoding for memory attributes

Bits	Encoding	Notes
[7]	Outer shareable. Always set for device memory or memory that is both inner and outer non-cacheable.	Shareability information is not recorded in the L1 data cache for implementations that use the mini-SCU. **WRMEMATTR[7]** is `0b0` for L1 data cache evictions in these implementations.
[6:3]	Outer memory type, or device type. If bits[1:0] indicate Device, then: `0b0000` nGnRnE. `0b0100` nGnRE. `0b1000` nGRE. `0b1100` GRE. If bits[1:0] indicate Normal, then: `0b0100` NC. `0b10RW` WT. `0b11RW` WB. Where `R` is read allocate hint, `W` is write allocate hint.	If an Armv7 architecture operating system runs on the processor, the Device memory type matches the nGnRE encoding and the Strongly-Ordered memory type matches the nGnRnE memory type. Outer read allocate hint information is not recorded in the L1 or L2 data caches. **WRMEMATTR[4]** and **REQMEMATTR[4]** are set to `0b1` for data cache evictions.
[2]	Inner shareable. Anything with bit[7] set must also have bit[2] set.	Shareability information is not recorded in the L1 data cache for implementations that use the mini-SCU. **WRMEMATTR[2]** is `0b0` for L1 data cache evictions in these implementations.
[1:0]	Inner memory type: `0b00` Device. `0b01` NC. `0b10` WT. `0b11` WB.	

A7.5 Handling of external aborts

The memory system handles external aborts using the synchronous abort mechanism, asynchronous abort mechanism, or the **nEXTERRIRQ** pin as described in this section.

Synchronous abort mechanism

External aborts on the following accesses use the synchronous abort mechanism.

- All load accesses.
- All Store Exclusive accesses (**STREX, STREXB, STREXH, STREXD, STXR, STXRB, STXRH, STXP, STLXR, STLXRB, STLXRH, and STLXP)**.

Asynchronous abort mechanism

External aborts on the following accesses use the asynchronous abort mechanism.

- Stores to Device memory (except Store Exclusive accesses).
- Stores to Normal memory that is Inner Non-cacheable, Inner Write-Through, Outer Non-cacheable, or Outer Write-Through (except Store Exclusive accesses).
- L1 data cache and L2 cache linefills that receive data from the interconnect in the dirty state.

nEXTERRIRQ pin

External aborts on the following accesses cause the **nEXTERRIRQ** pin to be asserted because the aborts might not relate directly back to a specific core in the cluster.

- All store accesses to Normal memory that is both Inner write-back and Outer write-back.
- Evictions from the L1 data cache or L2 cache.
- DVM Complete transactions.

Related reference

B1.93 L2 Extended Control Register on page B1-299

Chapter A8
AXI Master Interface

This chapter describes the AXI master memory interface.

It contains the following sections:

A8.1 About the AXI master interface

You can configure the processor to use the AXI protocol for the master memory interface.

Read responses

The AXI master can delay accepting a read data channel transfer by holding **RREADY** LOW for an indeterminate number of cycles. **RREADY** can be deasserted LOW between read data channel transfers that form part of the same transaction.

Write responses

The AXI master requires that the slave does not return a write response until it has received the write address.

The AXI master always accepts write responses without delay by holding **BREADY** HIGH.

Barriers

You must ensure that your interconnect and any peripherals connected to it do not return a write response for a transaction until that transaction would be considered complete by a later barrier. This means that the write must be observable to all other masters in the system. Arm expects the majority of peripherals to meet this requirement.

Related information

A8.2 AXI privilege information on page A8-101

A8.2 AXI privilege information

AXI provides information about the privilege level of an access on the **ARPROTM[0]** and **AWPROTM[0]** signals. However, when accesses might be cached or merged together, the resulting transaction can have both privileged and unprivileged data combined. If this happens, the processor marks the transaction as privileged, even it if was initiated by an unprivileged process.

The following table shows exception levels and corresponding **ARPROTM[0]** and **AWPROTM[0]** values.

Table A8-1 ARPROT and AWPROT values

Processor exception level	Type of access	Value of ARPROT[0] and AWPROT[0]
EL0, EL1, EL2, EL3	Cacheable read access	Privileged access
EL0	Device, or normal Non-cacheable read access	Unprivileged access
EL1, EL2, EL3		Privileged access
EL0, EL1, EL2, EL3	Cacheable write access	Privileged access
EL0	Device, nGnRnE, nGnRE, and nGRE write	Unprivileged access
EL1, EL2, EL3		Privileged access
EL0	Normal Non-cacheable or Device GRE write, except for STREX, STREXB, STREXH, STREXD, STXR, STXRB, STXRH, STXP, STLXR, STLXRB, STLXRH and STLXP to shareable memory	Privileged access
EL0	Normal Non-cacheable write for STREX, STREXB, STREXH, STREXD, STXR STXRB, STXRH, STXP, STLXR, STLXRB, STLXRH and STLXP to shareable memory	Unprivileged access
EL1, EL2, EL3	Normal Non-cacheable write	Privileged access
EL0, EL1, EL2, EL3	TLB pagewalk	Privileged access

A8.3 AXI transactions

The processor generates only a subset of all possible AXI transactions on the AXI master interface.

The processor does not generate any FIXED bursts and all WRAP bursts fetch a complete cache line starting with the critical word first. A burst does not cross a cache line boundary.

The cache linefill fetch length is always 64 bytes.

For WriteBack transfers the supported transfers are:

- WRAP 4 128-bit for read transfers (linefills).
- INCR 4 128-bit for write transfers (evictions).
- INCR N (N:1, 2, or 4) 128-bit write transfers (read allocate).

For Non-cacheable transactions:

- INCR N (N:1, 2, or 4) 128-bit for write transfers.
- INCR N (N:1, 2, or 4) 128-bit for read transfers.
- INCR 1 32-bit, 64-bit, and 128-bit for read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit for write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit for exclusive write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit for exclusive read transfers.

For Device transactions:

- INCR N (N:1, 2, or 4) 128-bit read transfers.
- INCR N (N:1, 2, or 4) 128-bit write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit exclusive read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit exclusive write transfers.

For translation table walk transactions INCR 1 32-bit, and 64-bit read transfers.

The following points apply to AXI transactions:
- WRAP bursts are only 128-bit.
- INCR 1 can be any size 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.

External memory accesses generate the following transactions in an implementation configured with an AXI master interface.

Table A8-2 AXI transactions

Attributes		**AXI transaction**			
Memory type	**Shareability**	**Load**	**Store**	**Load exclusive**	**Store exclusive**
Device	-	Read	Write	Read with **ARLOCKM** set HIGH	Write with **AWLOCKM** set HIGH
Normal, inner Non-cacheable, outer Non-cacheable	Non-shared	Read	Write	Read	Write
	Inner-shared			Read with **ARLOCKM** set HIGH	Write with **ARLOCKM** set HIGH
	Outer-shared				

Table A8-2 AXI transactions (continued)

Attributes		AXI transaction			
Memory type	**Shareability**	**Load**	**Store**	**Load exclusive**	**Store exclusive**
Normal, inner Non-cacheable, outer Write-Back or Write-Through, or Normal, inner Write-Through, outer Write-Back, Write-Through or Non-cacheable, or Normal inner Write-Back outer Non-cacheable or Write-Through	Non-shared	Read	Write	Read	Write
	Inner-shared			Read with **ARLOCKM** set HIGH	Write with **AWLOCKM** set HIGH
	Outer-shared				
Normal, inner Write-Back, outer Write-Back	Non-shared	Read	Write	Read	Write when the line is evicted
	Inner-shared				
	Outer-shared				

Related information

Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite

A8.4 Attributes of the AXI master interface

The table lists the possible values for the read and write issuing capabilities if the processor includes four cores.

n Number of cores.

m 1 if the processor is configured for the ACP interface, 0 if it is not.

Table A8-3 AXI master interface attributes

Attribute	Value	Comments
Write issuing capability	16	The cluster can issue a maximum of 16 writes: • Up to 16 writes to Normal memory that is both inner and outer write-back cacheable. • Up to 15 writes to all other memory types, including Device, Normal non-cacheable, and Write-through. Any mix of memory types is possible, and each write can be a single write or a write burst.
Read issuing capability	8n + 4m	8 for each core in the cluster including up to: • 8 data linefills. • 4 non-cacheable or Device data reads. • 1 non-cacheable TLB page-walk read. • 3 instruction linefills. If an ACP is configured, up to 4 ACP linefill requests can be generated.
Exclusive thread capability	n	Each core can have 1 exclusive access sequence in progress.
Write ID capability	16	The maximum number of outstanding write IDs is 16. This is the same as the maximum number of outstanding writes. Only Device memory types with nGnRnE or nGnRE can have more than one outstanding transaction with the same AXI ID. All other memory types use a unique AXI ID for every outstanding transaction.
Write ID width	5	The ID encodes the source of the memory transaction. See *Table A9-7 Encoding for AWIDM[4:0]* on page A9-115.
Read ID capability	8n + 4m	8 for each core in the processor and 4 for the ACP. Only Device memory types with nGnRnE or nGnRE can have more than one outstanding transaction with the same AXI ID. All other memory types use a unique AXI ID for every outstanding transaction.
Read ID width	6	The ID encodes the source of the memory transaction. See *Table A9-8 Encoding for ARIDM[5:0]* on page A9-115.

In the following table, nn is the core number `0b00`, `0b01`, `0b10`, or `0b11`.

Table A8-4 Encoding for AWIDM[4:0]

Attribute	Value	Issuing capability per ID	Comments
Write ID	0b000nn	1	Core nn system domain store exclusive
	0b001xx	0	Unused
	0b010xx	0	Unused
	0b011nn	15	Core nn non-re-orderable device write
	0b1xxxx	1	Write to normal memory, or re-orderable device memory

In the following table, nn is the core number 0b00, 0b01, 0b10, or 0b11.

Table A8-5 Encoding for ARIDM[5:0]

Attribute	Value	Issuing capability per ID	Comments
Read ID	0b0000nn	4	Core nn system domain exclusive read or non-reorderable device read
	0b0001xx	0	Unused
	0b001xxx	0	Unused
	0b01xx00	1	ACP read
	0b01xx01	0	Unused
	0b01xx1x	0	Unused
	0b1xxxnn	1	Core nn read

These ID and transaction details are provided for information only. Arm strongly recommends that all interconnects and peripherals are designed to support any type and number of transactions on any ID, to ensure compatibility with future products.

Related information

Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite

Chapter A9
ACE Master Interface

This chapter describes the ACE master interface.

It contains the following sections:

A9.1 About the ACE master interface

You can configure the processor to use the ACE protocol for the master memory interface.

Read responses

The ACE master can delay accepting a read data channel transfer by holding **RREADY** LOW for an indeterminate number of cycles. **RREADY** can be deasserted LOW between read data channel transfers that form part of the same transaction.

The ACE master asserts the read acknowledge signal **RACK** HIGH in the **ACLK** cycle following acceptance of the last read data channel transfer for a transaction. **RACK** is asserted in AXI compatibility mode in addition to ACE configurations.

- For interoperability of system components, Arm recommends that components interfacing with the ACE master are fully ACE compliant with no reliance on the subset of permitted **RACK** behavior described for the processor.
- If the interconnect does not perform hazarding between coherent and non-coherent requests, then, after it has returned the first transfer of read data for a non-coherent read, it must return all the remaining read transfers in the transaction, without requiring progress of any snoops to the cluster that could be to the same address.

Write responses

The ACE master requires that the slave does not return a write response until it has received the write address.

The ACE master always accepts write responses without delay by holding **BREADY** HIGH. It asserts the write acknowledge signal **WACK** HIGH in the **ACLK** cycle following acceptance of a write response. **WACK** is asserted in AXI compatibility mode in addition to ACE configurations.

For interoperability of system components, Arm recommends that components interfacing with the ACE master are fully ACE compliant with no reliance on the subset of permitted **BREADY** and **WACK** behavior described for the processor.

Barriers

The processor does not send barrier transactions to the interconnect. All barriers are terminated within the cluster.

You must ensure that your interconnect and any peripherals connected to it do not return a write response for a transaction until that transaction would be considered complete by a later barrier. This means that the write must be observable to all other masters in the system. Arm expects the majority of peripherals to meet this requirement.

Related information

A8.2 AXI privilege information on page A8-101

A9.2 ACE configurations

This section describes the ACE configurations.

Note

If you want to connect the processor to an AXI interconnect, Arm recommends that you use the AXI processor configuration option. Using the ACE processor configuration option in AXI mode is less area-efficient than the AXI configuration option.

Table A9-1 Supported ACE configurations

Signal	Feature						
	AXI mode	ACE non-coherent		ACE outer coherent		ACE inner coherent	
		No L3 cache	With L3 cache	No L3 cache	With L3 cache	No L3 cache	With L3 cache
BROADCASTCACHEMAINT	0	0	1	0	1	0	1
BROADCASTOUTER	0	0	0	1	1	1	1
BROADCASTINNER	0	0	0	0	0	1	1

The following table shows the key features in each of the supported ACE configurations.

Table A9-2 Supported features in the ACE configurations

Features	Configuration				
	AXI mode	ACE non-coherent, no L3 cache	ACE non-coherent, with L3 cache	ACE outer coherent	ACE inner coherent
AXI3 or AXI4 compliance	Yes	No	No	No	No
ACE compliance	No	Yes	Yes	Yes	Yes
Barriers on AR and AW channels	No	No	No	No	No
Cache maintenance requests on AR channel	No	No	Yes	Yes	Yes
Snoops on AC channel	No	No	No	Yes	Yes
Coherent requests on AR or AW channel	No	No	No	Yes	Yes
DVM requests on AR channel	No	No	No	No	Yes

A9.3 ACE privilege information

ACE provides information about the privilege level of an access on the **ARPROTM[0]** and **AWPROTM[0]** signals. However, when accesses might be cached or merged together, the resulting transaction can have both privileged and unprivileged data combined. If this happens, the processor marks the transaction as privileged, even if it was initiated by an unprivileged process.

The following table shows exception levels and corresponding **ARPROTM[0]** and **AWPROTM[0]** values.

Table A9-3 ARPROT and AWPROT values

Processor exception level	Type of access	Value of ARPROT[0] and AWPROT[0]
EL0, EL1, EL2, EL3	Cacheable read access	Privileged access
EL0	Device, or normal Non-cacheable read access	Unprivileged access
EL1, EL2, EL3		Privileged access
EL0, EL1, EL2, EL3	Cacheable write access	Privileged access
EL0	Device, nGnRnE, nGnRE, and nGRE write	Unprivileged access
EL1, EL2, EL3		Privileged access
EL0	Normal Non-cacheable or Device GRE write, except for `STREX`, `STREXB`, `STREXH`, `STREXD`, `STXR`, `STXRB`, `STXRH`, `STXP`, `STLXR`, `STLXRB`, `STLXRH`, and `STLXP` to shareable memory	Privileged access
EL0	Normal Non-cacheable write for `STREX`, `STREXB`, `STREXH`, `STREXD`, `STXR` `STXRB`, `STXRH`, `STXP`, `STLXR`, `STLXRB`, `STLXRH`, and `STLXP` to shareable memory	Unprivileged access
EL1, EL2, EL3	Normal Non-cacheable write	Privileged access
EL0, EL1, EL2, EL3	TLB pagewalk	Privileged access

A9.4 ACE transactions

The processor generates only a subset of all possible ACE transactions on the ACE master interface.

The processor does not generate any FIXED bursts and all WRAP bursts fetch a complete cache line starting with the critical word first. A burst does not cross a cache line boundary.

The cache linefill fetch length is always 64 bytes.

For WriteBack transfers the supported transfers are:

- WRAP 4 128-bit for read transfers (linefills).
- INCR 4 128-bit for write transfers (evictions).
- INCR N (N:1, 2, or 4) 128-bit write transfers (read allocate).

For Non-cacheable transactions:

- INCR N (N:1, 2, or 4) 128-bit for write transfers.
- INCR N (N:1, 2, or 4) 128-bit for read transfers.
- INCR 1 32-bit, 64-bit, and 128-bit for read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit for write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit for exclusive write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit for exclusive read transfers.

For Device transactions:

- INCR N (N:1, 2, or 4) 128-bit read transfers.
- INCR N (N:1, 2, or 4) 128-bit write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit exclusive read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit exclusive write transfers.

External memory accesses generate the following transactions in an implementation configured with an ACE master interface.

Table A9-4 ACE transactions

<table>
<tr><th colspan="2">Attributes</th><th colspan="5">ACE transaction</th></tr>
<tr><th>Memory type</th><th>Shareability</th><th>Domain</th><th>Load</th><th>Store</th><th>Load exclusive</th><th>Store exclusive</th></tr>
<tr><td>Device</td><td>-</td><td>System</td><td>ReadNoSnoop</td><td>WriteNoSnoop</td><td>ReadNoSnoop and ARLOCKM set to HIGH</td><td>WriteNoSnoop and AWLOCKM set to HIGH</td></tr>
<tr><td rowspan="3">Normal, inner Non-cacheable, outer Non-cacheable</td><td>Non-shared</td><td rowspan="3">System</td><td rowspan="3">ReadNoSnoop</td><td rowspan="3">WriteNoSnoop</td><td rowspan="3">ReadNoSnoop and ARLOCKM set to HIGH</td><td rowspan="3">WriteNoSnoop and AWLOCKM set to HIGH</td></tr>
<tr><td>Inner-shared</td></tr>
<tr><td>Outer-shared</td></tr>
<tr><td rowspan="4">Normal, inner Non-cacheable, outer Write-Back or Write-Through, or Normal, inner Write-Through, outer Write-Back, Write-Through or Non-cacheable, or Normal inner Write-Back outer Non-cacheable or Write-Through</td><td>Non-shared</td><td>System</td><td>ReadNoSnoop</td><td>WriteNoSnoop</td><td>ReadNoSnoop</td><td>ReadNoSnoop</td></tr>
<tr><td>Inner-shared</td><td>System</td><td rowspan="3">ReadNoSnoop</td><td rowspan="3">WriteNoSnoop</td><td rowspan="3">ReadNoSnoop with ARLOCKM set to HIGH</td><td rowspan="3">WriteNoSnoop with ARLOCKM set to HIGH</td></tr>
<tr><td rowspan="2">Outer-shared</td><td rowspan="2">System</td></tr>
</table>

Table A9-4 ACE transactions (continued)

Attributes		ACE transaction				
Memory type	**Shareability**	**Domain**	**Load**	**Store**	**Load exclusive**	**Store exclusive**
Normal, inner Write-Back, outer Write-Back	Non-shared	Non-shareable	ReadNoSnoop	WriteNoSnoop	ReadNoSnoop	WriteNoSnoop
	Inner-shared	Inner Shareable	ReadShared	ReadUnique or CleanUnique if required, then a WriteBack when the line is evicted	ReadShared with **ARLOCKM** set to HIGH	CleanUnique with **ARLOCKM** set to HIGH if required, then a WriteBack when the line is evicted
	Outer-shared	Outer Shareable				

The following table shows the ACE transactions that can be generated, and some typical operations that might cause the transactions to be generated. This is not an exhaustive list of ways to generate each type of transaction, because there are many possibilities.

Table A9-5 ACE transactions and typical operations

Transaction	Operation
ReadNoSnoop	Non-cacheable loads or instruction fetches. Linefills of non-shareable cache lines into L1 or L2.
ReadOnce	Cacheable loads that are not allocating into the cache, or cacheable instruction fetches when there is no L2 cache.
ReadClean	Not used.
ReadNotSharedDirty	Not used.
ReadShared	L1 Data linefills started by a load instruction, or L2 linefills started by an instruction fetch.
ReadUnique	L1 Data linefills started by a store instruction.
CleanUnique	Store instructions that hit in the cache but the line is not in a unique coherence state. Store instructions that are not allocating into the L1 or L2 caches, for example when streaming writes.
MakeUnique	Store instructions of a full cache line of data, that miss in the caches, and are allocating into the L2 cache.
CleanShared	Cache maintenance instructions.
CleanInvalid	Cache maintenance instructions.
MakeInvalid	Cache maintenance instructions.
DVM	TLB and instruction cache maintenance instructions.
DVM complete	DVM sync snoops received from the interconnect.
Barriers	DMB and DSB instructions. DVM sync snoops received from the interconnect.
WriteNoSnoop	Non-cacheable store instructions. Evictions of non-shareable cache lines from L1 and L2.
WriteUnique	Not used.
WriteLineUnique	Not used.
WriteBack	Evictions of dirty lines from the L1 or L2 cache, or streaming writes that are not allocating into the cache.
WriteClean	Evictions of dirty lines from the L2 cache, when the line is still present in an L1 cache. Some cache maintenance instructions.

Table A9-5 ACE transactions and typical operations (continued)

Transaction	Operation
WriteEvict	Evictions of unique clean lines, when configured in the L2ACTLR.
Evict	Evictions of clean lines, when configured in the L2ACTLR.

Related information

Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite

A9.5 Attributes of the ACE master interface

The table lists the maximum possible values for the read and write issuing capabilities if the processor includes four cores.

n Number of cores.

m 1 if the processor is configured for the ACP interface, 0 if it is not.

Table A9-6 ACE master interface attributes

Attribute	Value	Comments
Write issuing capability	16	The cluster can issue a maximum of 16 writes: • Up to 16 writes to Normal memory that is both inner and outer write-back cacheable. • Up to 15 writes to all other memory types, including Device, Normal non-cacheable, and Write-through. Any mix of memory types is possible, and each write can be a single write or a write burst.
Read issuing capability	8n + 4m	8 for each core in the cluster including up to: • 8 data linefills. • 4 non-cacheable or Device data reads. • 1 non-cacheable TLB page-walk read. • 3 instruction linefills. • 5 coherency operations. • 8 DVM messages. The 8 DVM messages per core can each be two-part DVM messages. They result in up to 16 DVM transactions per core. If an ACP is configured, up to 4 ACP linefill requests can be generated.
Exclusive thread capability	n	Each core can have 1 exclusive access sequence in progress.
Write ID capability	16	The maximum number of outstanding write IDs is 16. This is the same as the maximum number of outstanding writes. Only Device memory types with nGnRnE or nGnRE can have more than one outstanding transaction with the same AXI ID. All other memory types use a unique AXI ID for every outstanding transaction.
Write ID width	5	The ID encodes the source of the memory transaction. See *Table A9-7 Encoding for AWIDM[4:0]* on page A9-115.
Read ID capability	8n + 4m	8 for each core in the processor and 4 for the ACP. Only Device memory types with nGnRnE or nGnRE can have more than one outstanding transaction with the same AXI ID. All other memory types use a unique AXI ID for every outstanding transaction. Two part DVMs use the same ID for both parts, and therefore can have two outstanding transactions on the same ID.
Read ID width	6	The ID encodes the source of the memory transaction. See *Table A9-8 Encoding for ARIDM[5:0]* on page A9-115.

In the following table, nn is the core number `0b00`, `0b01`, `0b10`, or `0b11`.

Table A9-7 Encoding for AWIDM[4:0]

Attribute	Value	Issuing capability per ID	Comments
Write ID	`0b000nn`	1	Core nn system domain store exclusive
	`0b001xx`	0	Unused
	`0b010xx`	0	Unused
	`0b011nn`	15	Core nn non-re-orderable device write
	`0b1xxxx`	1	Write to normal memory, or re-orderable device memory

In the following table, nn is the core number `0b00`, `0b01`, `0b10`, or `0b11`.

Table A9-8 Encoding for ARIDM[5:0]

Attribute	Value	Issuing capability per ID	Comments
Read ID	`0b0000nn`	4	Core nn exclusive read or non-reorderable device read
	`0b0001xx`	0	Unused
	`0b001000`	0	Unused
	`0b001001`	1	DVM complete
	`0b00101x`	0	Unused
	`0b0011xx`	0	Unused
	`0b01xx00`	1	ACP read
	`0b01xx01`	0	Unused
	`0b01xx1x`	0	Unused
	`0b1xxxnn`	1	Core nn read

These ID and transaction details are provided for information only. Arm strongly recommends that all interconnects and peripherals are designed to support any type and number of transactions on any ID, to ensure compatibility with future products.

Related information

Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite

A9.6 Snoop channel properties

The table shows the properties of the ACE channels.

Table A9-9 ACE channel properties

Property	Value	Comment
Snoop acceptance capability	8	The SCU can accept and process a maximum of eight snoop requests from the system. It counts requests from the request being accepted on the AC channel to the response being accepted on the CR channel.
Snoop latency	Hit	When there is a hit in L2 cache, the best case for response and data is 13 processor cycles. When there is a miss in L2 cache and a hit in L1 cache, the best case for response and data is 16 processor cycles. Latencies can be higher if hazards occur or if there are not enough buffers to absorb requests.
	Miss	Best case six processor cycles when the SCU duplicate tags and L2 tags indicate the miss.
	DVM	The cluster takes a minimum of six cycles to provide a response to DVM packets.
Snoop filter	Supported	The cluster provides support for an external snoop filter in an interconnect. It indicates when clean lines are evicted from the processor by sending Evict transactions on the write channel. However there are some cases where incorrect software can prevent an Evict transaction from being sent. Therefore you must ensure that you build any external snoop filter to handle a capacity overflow that sends a back-invalidation to the processor if it runs out of storage. Examples of cases where evicts are not produced include: • Linefills that take external aborts. • Store exclusives that fail. • Mismatched aliases.
Supported transactions	-	All transactions described by the ACE protocols: • Are accepted on the master interface from the system. • Can be produced on the ACE master interface except: — WriteUnique. — WriteLineUnique. — ReadNotSharedDirty. — ReadClean.

Related information

Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite

A9.7 AXI compatibility mode

The processor implements an AXI3/AXI4 compatibility mode in ACE configurations. With this mode you can use the processor in a standalone environment where the AMBA 4 ACE interface is not required.

To enable the AXI compatibility mode, you must ensure that the **BROADCASTINNER**, **BROADCASTOUTER**, and **BROADCASTCACHEMAINT** input pins are set to LOW.

Note

The AXI build-time configuration option provides a more area-efficient AXI solution than the AXI compatibility mode in ACE configurations.

Chapter A10
CHI Master Interface

This chapter describes the CHI master memory interface.

It contains the following sections:

A10.1 About the CHI master interface

You can configure the processor to use the CHI protocol for the master memory interface.

A10.2 CHI configurations

This section describes the CHI configurations.

The following table shows the permitted combinations of these signals and the supported configurations in the Cortex-A32 processor, with a CHI bus.

Table A10-1 Supported CHI configurations

Signal	Feature					
	CHI non-coherent		CHI outer coherent		CHI inner coherent	
	No L3 cache	With L3 cache	No L3 cache	With L3 cache	No L3 cache	With L3 cache
BROADCASTCACHEMAINT	0	1	0	1	0	1
BROADCASTOUTER	0	0	1	1	1	1
BROADCASTINNER	0	0	0	0	1	1

The following table shows the key features in each of the supported CHI configurations.

Table A10-2 Supported features in the CHI configurations

Features	Configuration			
	CHI non-coherent, no L3 cache	CHI non-coherent, with L3 cache	CHI outer coherent	CHI inner coherent
Cache maintenance requests on TXREQ channel	No	Yes	Yes	Yes
Snoops on RXREQ channel	No	No	Yes	Yes
Coherent requests on TXREQ channel	No	No	Yes	Yes
DVM requests on TXREQ channel	No	No	No	Yes

A10.3 Attributes of the CHI master interface

The table lists the possible values for the read and write issuing capabilities if the processor includes four cores.

n Number of cores.

m 1 if the processor is configured for the ACP interface, 0 if it is not.

w (write issuing capability)+1.

Table A10-3 Attributes of the CHI master memory interface

Attribute	Value	Comments
Write issuing capability	Configuration dependent	The maximum number of writes varies, depending on the configuration of the processor: • The number of cores. • The presence of the L2 cache. If no L2 cache is configured: **One core** 5 outstanding writes. **2-4 cores** 8 outstanding writes. If an L2 cache is configured: **One core** 7 outstanding writes. **2-4 cores** 10 outstanding write. A cluster with four cores, with L2 cache, can issue 10 outstanding transactions. A processor with one core, without L2 cache, can issue five outstanding transactions. All outstanding transactions use a unique ID.
Read issuing capability	8n + 4m + 1	8 for each core in the cluster including up to: • 8 data linefills. • 4 Non-cacheable or Device data reads. • 1 Non-cacheable TLB page-walk read. • 3 instruction linefills. • 5 coherency operations. • 1 barrier operation. • 8 DVM messages. If an ACP is configured, up to 4 ACP linefill requests can be generated. 1 barrier operation is generated from the cluster.
Exclusive thread capability	n	Each core can have 1 exclusive access sequence in progress.

Table A10-3 Attributes of the CHI master memory interface (continued)

Attribute	Value	Comments
Transaction ID width	8	The ID encodes the source of the memory transaction. See *A9.5 Attributes of the ACE master interface* on page A9-114.
Transaction ID capability	8n + 4m + w + 1	8 for each core in the cluster in addition to: • 4 for the ACP interface. • 1 for barriers. • 6 to 11 writes, depending on the write issuing capability. Unlike in configurations with AXI or ACE, there is never any ID reuse in CHI implementations, regardless of the memory type.

There is no fixed mapping between CHI transaction IDs and cores. Some transaction IDs can be used for either reads or writes.

Related information

A9.5 Attributes of the ACE master interface on page A9-114

Arm® AMBA® 5 CHI Protocol Specification

A10.4 CHI channel properties

You can configure the processor to use the CHI protocol for the master memory interface

Table A10-4 CHI channel properties

Property	Value	Comment
Snoop acceptance capability	10	The SCU can accept and process a maximum of 10 snoop requests from the system.
DVM acceptance capability	4	The SCU can accept and process a maximum of four DVM transactions from the system. Each of these four transactions can be a two part DVM message. The interconnect must be configured to never send more than four DVM messages to a Cortex-A32 processor, otherwise the system might deadlock.
Snoop latency	Hit	When there is a hit in L2 cache, the best case for response and data is 11 processor cycles. When there is a miss in L2 cache and a hit in L1 cache, the best case for response and data is 14 processor cycles. Latencies can be higher if hazards occur or if there are not enough buffers to absorb requests.
	Miss	Best case six processor cycles when the SCU duplicate tags and L2 tags indicate the miss.
	DVM	The cluster takes a minimum of six cycles to provide a response to DVM packets.
Snoop filter	Supported	The cluster provides support for an external snoop filter in an interconnect. It indicates when clean lines are evicted from the processor by sending Evict transactions on the CHI write channel. However, there are some cases where incorrect software can prevent an Evict transaction from being sent, therefore you must ensure that any external snoop filter is built to handle a capacity overflow that sends a back-invalidation to the processor if it runs out of storage.
Supported transactions	-	All transactions described by the CHI protocol: • Are accepted on the CHI master interface from the system. • Can be produced on the CHI master interface except: — ReadClean. — WriteBackPtl. — WriteCleanPtl.

Related information

Arm® AMBA® 5 CHI Protocol Specification

A10.5 CHI transactions

CHI transactions are sent to a specific node in the interconnect based on the following criteria:

- Type of access.
- Address of the access.
- Settings of the System Address Map.

Addresses that map to an HN-F node can be marked as cacheable memory in the page tables, and can take part in the cache coherency protocol. Addresses that map to an HN-I or MN must be marked as device or non-cacheable memory.

Table A10-5 CHI transaction IDs

Transaction ID	Description
000nnxxx	Transaction from core nn. Can be a: • Read transaction. • Write transaction. • Cache maintenance transaction. • DVM transaction. • Barrier transaction.
001001xx	Transaction from the ACP interface. Can be a read or write.
00101110	Barrier generated in response to a DVM sync snoop from the interconnect.
0100xxxx	Eviction from L1 or L2 cache. The number of IDs used depends on the configuration.

Table A10-6 CHI transactions

Transaction	Operation
ReadNoSnp	Non-cacheable loads or instruction fetches. Linefills of non-shareable cache lines into L1 or L2.
ReadOnce	Cacheable loads that are not allocating into the cache, or cacheable instruction fetches when there is no L2 cache.
ReadClean	Not used.
ReadShared	L1 Data linefills started by a load instruction, or L2 linefills started by an instruction fetch.
ReadUnique	L1 Data linefills started by a store instruction.
CleanUnique	Store instructions that hit in the cache but the line is not in a unique coherence state.
MakeUnique	Store instructions of a full cache line of data, that miss in the caches, and are allocating into the L2 cache.
CleanShared	Cache maintenance instructions.
CleanInvalid	Cache maintenance instructions.
MakeInvalid	Cache maintenance instructions.
DVMOp	TLB and instruction cache maintenance instructions.
EOBarrier	DMB instructions.
ECBarrier	DSB instructions. DVM sync snoops received from the interconnect.
WriteNoSnpPtl	Non-cacheable store instructions.
WriteNoSnpFull	Non-cacheable store instructions, or evictions of non-shareable cache lines from the L1 and L2 cache.
WriteUniqueFull	Cacheable writes of a full cache line, that are not allocating into L1 or L2 caches, for example streaming writes.
WriteUniquePtl	Cacheable writes of less than a full cache line that are not allocating into L1 or L2.

Table A10-6 CHI transactions (continued)

Transaction	Operation
WriteBackFull	Evictions of dirty lines from the L1 or L2 cache.
WriteBackPtl	Not used.
WriteCleanFull	Evictions of dirty lines from the L2 cache, when the line is still present in an L1 cache. Some cache maintenance instructions.
WriteCleanPtl	Not used.
WriteEvictFull	Evictions of unique clean lines, when configured in the L2ACTLR.
Evict	Evictions of clean lines, when configured in the L2ACTLR.

External memory accesses generate the following transactions in an implementation configured with a CHI master interface.

Table A10-7 CHI transactions

Attributes		CHI transaction				
Memory type	**Shareability**	**SnpAttr**	**Load**	**Store**	**Load exclusive**	**Store exclusive**
Device	-	Non-snoopable	ReadNoSnp	WriteNoSnp	ReadNoSnp and **Excl** set to HIGH	WriteNoSnp and **Excl** set to HIGH
Normal, inner Non-cacheable, outer Non-cacheable	Non-shared	Non-snoopable	ReadNoSnp	WriteNoSnp	ReadNoSnp and **Excl** set to HIGH	WriteNoSnp and **Excl** set to HIGH
	Inner-shared					
	Outer-shared					
Normal, inner Non-cacheable, outer Write-Back or Write-Through, or Normal, inner Write-Through, outer Write-Back, Write-Through or Non-cacheable, or Normal inner Write-Back outer Non-cacheable or Write-Through	Non-shared	Non-snoopable	ReadNoSnp	WriteNoSnp	ReadNoSnp	ReadNoSnp
	Inner-shared	Non-snoopable	ReadNoSnp	WriteNoSnp	ReadNoSnp with **Excl** set to HIGH	WriteNoSnp with **Excl** set to HIGH
	Outer-shared	Non-snoopable				

Table A10-7 CHI transactions (continued)

Attributes		CHI transaction				
Memory type	**Shareability**	**SnpAttr**	**Load**	**Store**	**Load exclusive**	**Store exclusive**
Normal, inner Write-Back, outer Write-Back	Non-shared	Non-snoopable	ReadNoSnp	WriteNoSnp	ReadNoSnp	WriteNoSnp
	Inner-shared	Inner snoopable	ReadShared	ReadUnique, CleanUnique, or MakeUnique if allocating into the cache, then a WriteBackFull when the line is evicted. WriteUniqueFull or WriteUniquePtl if not allocating into the cache.	ReadShared with **Excl** set to HIGH	CleanUnique with **Excl** set to HIGH if required, then a WriteBackFull when the line is evicted
	Outer-shared	Outer snoopable				

Related information

Arm® AMBA® 5 CHI Protocol Specification

Chapter A11
ACP Slave Interface

This chapter describes the ACP slave interface.

It contains the following sections:

A11.1 About the ACP

The optional *Accelerator Coherency Port* (ACP) is implemented as an AXI4 slave interface with some restrictions.

- 128-bit read and write interfaces.
- **ARCACHE** and **AWCACHE** are restricted to Normal, Write-Back, Read-Write-Allocate, Read-Allocate, Write-Allocate, and No-Allocate memory. **ARCACHE** and **AWCACHE** are limited to the values `0b0111`, `0b1011`, and `0b1111`. Other values cause a SLVERR response on **RRESP** or **BRESP**.
- Exclusive accesses are not supported.
- Barriers are not supported. The **BRESP** handshake for a write transaction indicates global observability for that write.
- **ARSIZE** and **AWSIZE** signals are not present and assume a value of `0b100`, 16 bytes.
- **ARBURST** and **AWBURST** signals are not present and assume a value of INCR.
- **ARLOCK** and **AWLOCK** signals are not present.
- **ARQOS** and **AWQOS** signals are not present.
- **ARLEN** and **AWLEN** are limited to values 0 and 3.

A11.2 Transfer size support

ACP supports the following read-request transfer size and length combinations:

- 64 byte INCR request characterized by:
  - **ARLEN** is `0x03`, 4 beats.
  - **ARADDR** aligned to 64 byte boundary, so **ARADDR[5:0]** is `0b00 0000`.
  - **ARSIZE** and **ARBURST** assume values of `0b100` and INCR respectively.
- 16 byte INCR request characterized by:
  - **ARLEN** is `0x00`, 1 beat.
  - **ARADDR** aligned to 16 byte boundary, so **ARADDR[3:0]** is `0x0`.

ACP supports the following write-request transfer size and length combinations:

- 64 byte INCR request characterized by:
  - **AWLEN** is `0x03`, 4 beats.
  - **AWADDR** aligned to 64 byte boundary, so **AWADDR[5:0]** is `0b00 0000`.
  - **AWSIZE** and **AWBURST** assume values of `0b100` and INCR respectively.
  - **WSTRB** for all beats must be the same and either all asserted or all deasserted.
- 16 byte INCR request characterized by:
  - **AWLEN** is `0x00`, 1 beat.
  - **AWADDR** aligned to 16 byte boundary, so **AWADDR[3:0]** is `0x0`.
  - **AWSIZE** and **AWBURST** assume values of `0b100` and INCR respectively.
  - **WSTRB** can take any value.

Requests not meeting these restrictions cause a SLVERR response on **RRESP** or **BRESP**.

A11.3 ACP performance

The ACP interface can support up to four outstanding transactions. These can be any combination of reads and writes.

The master must avoid sending more than one outstanding transaction on the same AXI ID, to prevent the second transaction stalling the interface until the first has completed. If the master requires explicit ordering between two transactions, Arm recommends that it waits for the response to the first transaction before sending the second transaction.

Writes are higher performance when they contain a full cache line of data.

If SCU cache protection is configured, writes of less than 64 bits incur an overhead of performing a read-modify-write sequence if they hit in the L2 cache.

Some L2 resources are shared between the ACP interface and the cores, therefore heavy traffic on the ACP interface might, in some cases, reduce the performance of the cores.

AXI and ACE You can use the **ARCACHE** and **AWCACHE** signals to control whether the ACP request causes an allocation into the L2 cache if it misses.

CHI To ensure correct ordering of data beats, ACP reads that miss always allocate into the L2 cache.

A11.4 ACP user signals

ACP transactions can cause coherent requests to the system. Therefore ACP requests must pass Inner and Outer Shareable attributes to the L2. Use specific encoding to pass the shareability attribute.

Table A11-1 Encoding of the ACP shareability attribute

AxUSER[1:0]	Attribute
`0b00`	Non-shareable
`0b01`	Inner Shareable
`0b10`	Outer Shareable

This is the same encoding as **AxDOMAIN** on ACE, except that a value of `0b11` is not supported.

Chapter A12
GIC CPU Interface

This chapter describes the *Generic Interrupt Controller* (GIC) CPU interface of the processor.

It contains the following sections:

A12.1 Bypassing the GIC CPU Interface

The processor optionally implements the GIC CPU Interface. If present, you can disable it by asserting the **GICCDISABLE** signal HIGH at reset.

If the GIC is enabled, the input pins **nVIRQ** and **nVFIQ** must be tied off to HIGH because the internal GIC CPU interface generates the virtual interrupt signals to the cores. Software controls the **nIRQ** and **nFIQ** signals, therefore there is no requirement to tie them HIGH. If you disable the GIC CPU interface, a GIC that is external to the processor can drive the input signals **nVIRQ** and **nVFIQ**.

Asserting the **GICCDISABLE** signal HIGH at reset removes access to the memory-mapped and system GIC CPU Interface registers.

Related information

B1.85 Processor Feature Register 1 on page B1-285

A12.2 Memory map for the GIC CPU interface

The GIC CPU Interface is a memory-mapped interface. It is offset from **PERIPHBASE**.

If **GICCDISABLE** is asserted, the registers are not available.

Table A12-1 GIC memory map

Address range	Functional block
0x00000-0x01FFF	CPU Interface
0x02000-0x0FFFF	Reserved
0x10000-0x10FFF	Virtual Interface Control
0x11000-0x1FFFF	Reserved
0x20000-0x21FFF	Virtual CPU Interface
0x22000-0x2EFFF	Reserved
0x2F000-0x30FFF	Alias of Virtual CPU Interface
0x31000-0x3FFFF	Reserved

Related information

B1.85 Processor Feature Register 1 on page B1-285

Part B
Register Descriptions

Chapter B1
AArch32 system registers

This chapter describes the system registers in the AArch32 state.

It contains the following sections:

B1.1 AArch32 register summary

In AArch32 state, you access the system registers through a conceptual coprocessor, identified as CP15, the System Control Coprocessor.

Within CP15, there is a top-level grouping of system registers by a primary coprocessor register number, c0-c15. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about using the conceptual System Control Coprocessor in a VMSA context.

The system register space includes System operations system registers and System operations. The description of the system register space describes the permitted access, RO, WO, or RW, to each register or operation.

The following sections describe the CP15 system control registers grouped by CRn order, and are accessed by the MCR and MRC instructions.

- *B1.2 c0 registers* on page B1-146.
- *B1.3 c1 registers* on page B1-149.
- *B1.4 c2 registers* on page B1-150.
- *B1.5 c3 registers* on page B1-151.
- *B1.6 c4 registers* on page B1-152.
- *B1.7 c5 registers* on page B1-153.
- *B1.8 c6 registers* on page B1-154.
- *B1.9 c7 registers* on page B1-155.
- *B1.10 c7 system operations* on page B1-156.
- *B1.11 c8 system operations* on page B1-159.
- *B1.12 c9 registers* on page B1-161.
- *B1.13 c10 registers* on page B1-162.
- *B1.14 c11 registers* on page B1-163.
- *B1.15 c12 registers* on page B1-164.
- *B1.16 c13 registers* on page B1-166.
- *B1.17 c14 registers* on page B1-167.
- *B1.18 c15 registers* on page B1-168.

The following subsection describes the 64-bit registers and provides cross-references to individual register descriptions:

- *B1.19 64-bit registers* on page B1-169.

In addition to listing the CP15 system registers by CRn ordering, the following subsections describe the CP15 system registers by functional group:

- *B1.20 AArch32 Identification registers* on page B1-170.
- *B1.21 AArch32 Virtual memory control registers* on page B1-172.
- *B1.22 AArch32 Fault handling registers* on page B1-173.
- *B1.23 AArch32 Other System control registers* on page B1-174.
- *B1.24 AArch32 Address registers* on page B1-175.
- *B1.25 AArch32 Thread registers* on page B1-176.
- *B1.26 AArch32 Performance monitor registers* on page B1-177.
- *B1.27 AArch32 Secure registers* on page B1-179.
- *B1.28 AArch32 Virtualization registers* on page B1-180.
- *B1.29 AArch32 GIC system registers* on page B1-182.
- *B1.30 AArch32 Generic Timer registers* on page B1-184.
- *B1.31 AArch32 Implementation defined registers* on page B1-185.

The following table describes the column headings in the CP15 register summary tables used throughout this section.

Table B1-1 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.

B1.2 c0 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value zero are called c0 registers.

Table B1-2 c0 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	MIDR	0x411FD010	*B1.96 Main ID Register* on page B1-307
		1	CTR	0x84448004	*B1.46 Cache Type Register* on page B1-213
		2	TCMTR	0x00000000	*B1.108 TCM Type Register* on page B1-331
		3	TLBTR	0x00000000	*B1.109 TLB Type Register* on page B1-332
		4, 7	MIDR	0x411FD010	Aliases of Main ID Register, *B1.96 Main ID Register* on page B1-307
		5	MPIDR	-	*B1.97 Multiprocessor Affinity Register* on page B1-309 The reset value depends on the primary inputs, CLUSTERIDAFF1 and CLUSTERIDAFF2, and the number of cores that the device implements.
		6	REVIDR	0x00000000	*B1.102 Revision ID Register* on page B1-319
	c1	0	ID_PFR0	0x00000131	*B1.84 Processor Feature Register 0* on page B1-283
		1	ID_PFR1	0x10011011	*B1.85 Processor Feature Register 1* on page B1-285 Bits [31:28] are `0x1` if the GIC CPU interface is implemented and enabled, and `0x0` otherwise.
		2	ID_DFR0	0x03010066	*B1.73 Debug Feature Register 0* on page B1-261 Bits [19:16] are `0x1` if ETM is implemented, and `0x0` otherwise.
		3	ID_AFR0	0x00000000	*B1.72 Auxiliary Feature Register 0* on page B1-260
		4	ID_MMFR0	0x10201105	*B1.80 Memory Model Feature Register 0* on page B1-275
		5	ID_MMFR1	0x40000000	*B1.81 Memory Model Feature Register 1* on page B1-277
		6	ID_MMFR2	0x01260000	*B1.82 Memory Model Feature Register 2* on page B1-279
		7	ID_MMFR3	0x02102211	*B1.83 Memory Model Feature Register 3* on page B1-281
	c2	0	ID_ISAR0	0x02101110	*B1.74 Instruction Set Attribute Register 0* on page B1-263
		1	ID_ISAR1	0x13112111	*B1.75 Instruction Set Attribute Register 1* on page B1-265
		2	ID_ISAR2	0x21232042	*B1.76 Instruction Set Attribute Register 2* on page B1-267
		3	ID_ISAR3	0x01112131	*B1.77 Instruction Set Attribute Register 3* on page B1-269
		4	ID_ISAR4	0x00011142	*B1.78 Instruction Set Attribute Register 4* on page B1-271
		5	ID_ISAR5	0x00011121	*B1.79 Instruction Set Attribute Register 5* on page B1-273 ID_ISAR5 has the value `0x00010001` if the Cryptographic Extension is not implemented and enabled.

Table B1-2 c0 register summary (continued)

Op1	CRm	Op2	Name	Reset	Description
1	c0	0	CCSIDR	-	*B1.39 Cache Size ID Register* on page B1-195
		1	CLIDR	`0x0A200023`	*B1.40 Cache Level ID Register* on page B1-198 The value is `0x09200003` if the L2 cache is not implemented.
		7	AIDR	`0x00000000`	*B1.34 Auxiliary ID Register* on page B1-190
2	c0	0	CSSELR	`0x00000000`	*B1.45 Cache Size Selection Register* on page B1-211
4	c0	0	VPIDR	`0x411FD010`	*B1.121 Virtualization Processor ID Register* on page B1-348
		5	VMPIDR	-	*B1.120 Virtualization Multiprocessor ID Register* on page B1-347 The reset value is the value of the Multiprocessor Affinity Register.

B1.3 c1 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value one are called c1 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c1.

Table B1-3 c1 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	SCTLR	0x00C50838	*B1.105 System Control Register* on page B1-324 The reset value depends on inputs, **CFGTE**, **CFGEND**, and **VINITHI**. The value listed here assumes these signals are set to LOW.
		1	ACTLR	0x00000000	*B1.32 Auxiliary Control Register* on page B1-187
		2	CPACR	0x00000000	*B1.41 Architectural Feature Access Control Register* on page B1-200
	c1	0	SCR	0x00000000	*B1.104 Secure Configuration Register* on page B1-321
		1	SDER	0x00000000	*B1.107 Secure Debug Enable Register* on page B1-330
		2	NSACR	0x00000000	*B1.98 Non-Secure Access Control Register* on page B1-311
	c3	1	SDCR	0x00000000	*B1.106 Secure Debug Control Register* on page B1-328
4	c0	0	HSCTLR	0x03C50838	*B1.67 Hyp System Control Register* on page B1-248
		1	HACTLR	0x00000000	*B1.55 Hyp Auxiliary Control Register* on page B1-225
	c1	0	HCR	0x00000000	*B1.61 Hyp Configuration Register* on page B1-234
		1	HDCR	0x00000006	*B1.63 Hyp Debug Control Register* on page B1-242
		2	HCPTR	0x000033FF	*B1.60 Hyp Architectural Feature Trap Register* on page B1-231 The reset value depends on the FPU and NEON configuration. If Advanced SIMD and floating-point are implemented, the reset value is 0x000033FF. If Advanced SIMD and floating-point are not implemented, the reset value is 0x0000BFFF.
		3	HSTR	0x00000000	*B1.69 Hyp System Trap Register* on page B1-253
		4	HCR2	0x00000000	*B1.62 Hyp Configuration Register 2* on page B1-240
		7	HACR	0x00000000	*B1.54 Hyp Auxiliary Configuration Register* on page B1-224

B1.4 c2 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value two are called c2 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c2.

Table B1-4 c2 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	TTBR0	UNK	*B1.113 Translation Table Base Register 0* on page B1-338
		1	TTBR1	UNK	*B1.116 Translation Table Base Register 1* on page B1-342
		2	TTBCR	`0x00000000`	*B1.110 Translation Table Base Control Register* on page B1-333 The reset value is `0x00000000` for the Secure copy of the register. The reset value for the EAE bit of the Non-secure copy of the register is `0x0`. You must program the Non-secure copy of the register with the required initial value, as part of the processor boot sequence.
4	c0	2	HTCR	UNK	*B1.70 Hyp Translation Control Register* on page B1-257
	c1	2	VTCR	UNK	*B1.122 Virtualization Translation Control Register* on page B1-349

B1.5 c3 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value three are called c3 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c3.

Table B1-5 c3 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	DACR	UNK	*B1.47 Domain Access Control Register* on page B1-215

B1.6 c4 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value four are called c4 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c4.

Table B1-6 c4 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c6	0	ICC_PMR	0x00000000	Priority Mask Register

B1.7 c5 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value five are called c5 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c5.

Table B1-7 c5 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	DFSR	UNK	*B1.49 Data Fault Status Register* on page B1-217
		1	IFSR	UNK	*B1.87 Instruction Fault Status Register* on page B1-288
	c1	0	ADFSR	0x00000000	*B1.33 Auxiliary Data Fault Status Register* on page B1-189
		1	AIFSR	0x00000000	*B1.35 Auxiliary Instruction Fault Status Register* on page B1-191
4	c1	0	HADFSR	0x00000000	*B1.56 Hyp Auxiliary Data Fault Status Syndrome Register* on page B1-227
		1	HAIFSR	0x00000000	*B1.57 Hyp Auxiliary Instruction Fault Status Syndrome Register* on page B1-228
	c2	0	HSR	UNK	*B1.68 Hyp Syndrome Register* on page B1-252

B1.8 c6 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value six are called c6 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c6.

Table B1-8 c6 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	DFAR	UNK	*B1.48 Data Fault Address Register* on page B1-216
		2	IFAR	UNK	*B1.86 Instruction Fault Address Register* on page B1-287
4	c0	0	HDFAR	UNK	*B1.64 Hyp Data Fault Address Register* on page B1-245
		2	HIFAR	UNK	*B1.65 Hyp Instruction Fault Address Register* on page B1-246
		4	HPFAR	UNK	*B1.66 Hyp IPA Fault Address Register* on page B1-247

B1.9 c7 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value seven are called c7 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c7.

Table B1-9 c7 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c4	0	PAR	UNK	*B1.100 Physical Address Register* on page B1-315

B1.10 c7 system operations

System operations are divided into two categories. When the CRn value is c7, these operations are called c7 system operations.

The following table shows the System operations when CRn is c7 and the processor is in AArch32 state. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.* for more information about these operations.

Table B1-10 c7 System operation summary

op1	CRm	op2	Name	Description
0	c1	0	ICIALLUIS	Invalidate all instruction caches Inner Shareable to PoU PoU = Point of Unification. PoU is set by the **BROADCASTINNER** signal and can be in the L1 data cache or outside of the processor, in which case PoU is dependent on the external memory system.
		6	BPIALLIS	Invalidate all entries from branch predictors Inner Shareable
	c5	0	ICIALLU	Invalidate all Instruction Caches to PoU
		1	ICIMVAU	Invalidate Instruction Caches by VA to PoU
		4	CP15ISB	Instruction Synchronization Barrier operation, this operation is deprecated in Armv8-A
		6	BPIALL	Invalidate all entries from branch predictors
		7	BPIMVA	Invalidate VA from branch predictors
	c6	1	DCIMVAC	Invalidate data cache line by VA to PoC PoC = Point of Coherence. The PoC is always outside of the processor and is dependent on the external memory system.
		2	DCISW	Invalidate data cache line by set/way
	c8	0	ATS1CPR	Stage 1 current state PL1 read
		1	ATS1CPW	Stage 1 current state PL1 write
		2	ATS1CUR	Stage 1 current state unprivileged read
		3	ATS1CUW	Stage 1 current state unprivileged write
		4	ATS12NSOPR	Stages 1 and 2 Non-secure only PL1 read
		5	ATS12NSOPW	Stages 1 and 2 Non-secure only PL1 write
		6	ATS12NSOUR	Stages 1 and 2 Non-secure only unprivileged read
		7	ATS12NSOUW	Stages 1 and 2 Non-secure only unprivileged write
	c10	1	DCCMVAC	Clean data cache line by VA to PoC
		2	DCCSW	Clean data cache line by set/way
		4	CP15DSB	Data Synchronization Barrier operation, this operation is deprecated in Armv8-A
		5	CP15DMB	Data Memory Barrier operation, this operation is deprecated in Armv8-A
	c11	1	DCCMVAU	Clean data cache line by VA to PoU
	c14	1	DCCIMVAC	Clean and invalidate data cache line by VA to PoC
		2	DCCISW	Clean and invalidate data cache line by set/way

Table B1-10 c7 System operation summary (continued)

op1	CRm	op2	Name	Description
4	c8	0	ATS1HR	Stage 1 Hyp mode read
		1	ATS1HW	Stage 1 Hyp mode write

B1.11 c8 system operations

System operations are divided into two categories. When the CRn value is c8, these operations are called c8 system operations.

The following table shows the System operations when CRn is c8 and the processor is in AArch32 state. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about these operations.

Table B1-11 c8 System operations summary

op1	CRm	op2	Name	Description
0	c3	0	TLBIALLIS	Invalidate entire TLB Inner Shareable
		1	TLBIMVAIS	Invalidate unified TLB entry by VA and ASID Inner Shareable
		2	TLBIASIDIS	Invalidate unified TLB by ASID match Inner Shareable
		3	TLBIMVAAIS	Invalidate unified TLB entry by VA all ASID Inner Shareable
		5	TLBIMVALIS	Invalidate unified TLB entry by VA Inner Shareable, Last level
		7	TLBIMVAALIS	Invalidate unified TLB by VA all ASID Inner Shareable, Last level
	c5	0	ITLBIALL	Invalidate instruction TLB
		1	ITLBIMVA	Invalidate instruction TLB entry by VA and ASID
		2	ITLBIASID	Invalidate instruction TLB by ASID match
	c6	0	DTLBIALL	Invalidate data TLB
		1	DTLBIMVA	Invalidate data TLB entry by VA and ASID
		2	DTLBIASID	Invalidate data TLB by ASID match
	c7	0	TLBIALL	Invalidate unified TLB
		1	TLBIMVA	Invalidate unified TLB by VA and ASID
		2	TLBIASID	Invalidate unified TLB by ASID match
		3	TLBIMVAA	Invalidate unified TLB entries by VA all ASID
		5	TLBIMVAL	Invalidate last level of stage 1 TLB entry by VA
		7	TLBIMVAAL	Invalidate last level of stage 1 TLB entry by VA all ASID

Table B1-11 c8 System operations summary (continued)

op1	CRm	op2	Name	Description
4	c0	1	TLBIIPAS2IS	TLB Invalidate entry by Intermediate Physical Address, Stage 2, Inner Shareable
		5	TLBIIPAS2LIS	TLB Invalidate entry by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
	c3	0	TLBIALLHIS	Invalidate entire Hyp unified TLB Inner Shareable
		1	TLBIMVAHIS	Invalidate Hyp unified TLB entry by VA Inner Shareable
		4	TLBIALLNSNHIS	Invalidate entire Non-secure non-Hyp unified TLB Inner Shareable
		5	TLBIMVALHIS	Invalidate Unified Hyp TLB entry by VA Inner Shareable, Last level
	c4	1	TLBIIPAS2	TLB Invalidate entry by Intermediate Physical Address, Stage 2
		5	TLBIIPAS2L	TLB Invalidate entry by Intermediate Physical Address, Stage 2, Last level
	c7	0	TLBIALLH	Invalidate entire Hyp unified TLB
		1	TLBIMVAH	Invalidate Hyp unified TLB entry by VA
		4	TLBIALLNSNH	Invalidate entire Non-secure non-Hyp unified TLB
		5	TLBIMVALH	Invalidate Unified Hyp TLB entry by VA, Last level

B1.12 c9 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value nine are called c9 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c9. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

Table B1-12 c9 register summary CRn=c9

Op1	CRm	Op2	Name	Reset	Description
0	c12	0	PMCR	`0x41063000`	Performance Monitors Control Register
		1	PMNCNTENSET	UNK	Performance Monitors Count Enable Set Register
		2	PMNCNTENCLR	UNK	Performance Monitors Count Enable Clear Register
		3	PMOVSR	UNK	Performance Monitor Overflow Flag Status Clear Register
		4	PMSWINC	UNK	Performance Monitors Software Increment Register
		5	PMSELR	UNK	Performance Monitors Event Counter Selection Register
		6	PMCEID0	`0x6FFFBFFF`	*C9.3 Performance Monitors Common Event Identification Register 0* on page C9-477 The reset value is `0x6E3FBFFF` if L2 cache is not implemented.
		7	PMCEID1	`0x00000000`	*C9.4 Performance Monitors Common Event Identification Register 1* on page C9-481
	c13	0	PMCCNTR	UNK	Performance Monitors Cycle Counter
		1	PMXEVTYPER	UNK	Performance Monitors Selected Event Type and Filter Register
		2	PMXEVCNTR	UNK	Performance Monitors Selected Event Counter Register
	c14	0	PMUSERENR	`0x00000000`	Performance Monitors User Enable Register
		1	PMINTENSET	UNK	Performance Monitors Interrupt Enable Set Register
		2	PMINTENCLR	UNK	Performance Monitors Interrupt Enable Clear Register
		3	PMOVSSET	UNK	Performance Monitor Overflow Flag Status Set Register
1	c0	2	L2CTLR	-	*B1.92 L2 Control Register* on page B1-297 The reset value depends on the processor configuration.
		3	L2ECTLR	`0x00000000`	*B1.93 L2 Extended Control Register* on page B1-299

B1.13 c10 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value ten are called c10 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c10.

Table B1-13 c10 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c2	0	PRRR	UNK	*B1.101 Primary Region Remap Register* on page B1-316
		0	MAIR0	UNK	*B1.95 Memory Attribute Indirection Registers 0 and 1* on page B1-304
		1	NMRR	UNK	*B1.99 Normal Memory Remap Register* on page B1-313
		1	MAIR1	UNK	*B1.95 Memory Attribute Indirection Registers 0 and 1* on page B1-304
	c3	0	AMAIR0	`0x00000000`	*B1.36 Auxiliary Memory Attribute Indirection Register 0* on page B1-192
		1	AMAIR1	`0x00000000`	*B1.37 Auxiliary Memory Attribute Indirection Register 1* on page B1-193
4	c2	0	HMAIR0	UNK	Hyp Memory Attribute Indirection Register 0 See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
		1	HMAIR1	UNK	Hyp Memory Attribute Indirection Register 1 See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
	c3	0	HAMAIR0	`0x00000000`	*B1.58 Hyp Auxiliary Memory Attribute Indirection Register 0* on page B1-229
		1	HAMAIR1	`0x00000000`	*B1.59 Hyp Auxiliary Memory Attribute Indirection Register 1* on page B1-230

B1.14 c11 registers

There are no system registers to access when the processor is in AArch32 state and the value of CRn is 11.

B1.15 c12 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value twelve are called c12 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c12.

Table B1-14 c12 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	VBAR	`0x00000000`	*B1.119 Vector Base Address Register* on page B1-346. `0x00000000` is the secure reset value and UNK is the non-secure reset value.
		1	MVBAR	UNK	Monitor Vector Base Address Register. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
		2	RMR	`0x00000000`	*B1.103 Reset Management Register* on page B1-320.
	c1	0	ISR	UNK	*B1.90 Interrupt Status Register* on page B1-293.
	c8	0	ICC_IAR0	-	Interrupt Acknowledge Register 0
		1	ICC_EOIR0	-	End Of Interrupt Register 0
		2	ICC_HPPIR0	-	Highest Priority Pending Interrupt Register 0
		3	ICC_BPR0	`0x00000002`	Binary Point Register 0
		4	ICC_AP0R0	`0x00000000`	Active Priorities 0 Register 0
	c9	0	ICC_AP1R0	`0x00000000`	Active Priorities 1 Register 0
	c11	1	ICC_DIR	-	Deactivate Interrupt Register
		3	ICC_RPR	-	Running Priority Register
	c12	0	ICC_IAR1	-	Interrupt Acknowledge Register 1
		1	ICC_EOIR1	-	End Of Interrupt Register 1
		2	ICC_HPPIR1	-	Highest Priority Pending Interrupt Register 1
		3	ICC_BPR1	`0x00000003`	Binary Point Register 1 This is the reset value in non-secure state. In secure state, the reset value is 0x00000002.
		4	ICC_CTLR	`0x00000400`	Interrupt Control Register
		5	ICC_SRE	`0x00000000`	System Register Enable Register
		6	ICC_IGRPEN0	`0x00000000`	Interrupt Group Enable Register 0
		7	ICC_IGRPEN1	`0x00000000`	Interrupt Group Enable Register 1

Table B1-14 c12 register summary (continued)

Op1	CRm	Op2	Name	Reset	Description
4	c0	0	HVBAR	UNK	*B1.71 Hyp Vector Base Address Register* on page B1-259.
	c8	0	ICH_AP0R0	0x00000000	Interrupt Controller Hyp Active Priorities Register (0,0)
	c9	0	ICH_AP1R0	0x00000000	Interrupt Controller Hyp Active Priorities Register (1,0)
		4	ICH_VSEIR	0x00000000	Interrupt Controller Virtual System Error Interrupt Register
		5	ICC_HSRE	0x00000000	System Register Enable Register for EL2
	c11	0	ICH_HCR	0x00000000	Interrupt Controller Hyp Control Register
		1	ICH_VTR	0x90080003	Interrupt Controller VGIC Type Register
		2	ICH_MISR	0x00000000	Interrupt Controller Maintenance Interrupt State Register
		3	ICH_EISR	0x00000000	Interrupt Controller End of Interrupt Status Register
		7	ICH_VMCR	0x004C0000	Interrupt Controller Virtual Machine Control Register
		5	ICH_ELRSR	0x0000000F	Interrupt Controller Empty List Register Status Register
	c12	0	ICH_LR0	0x00000000	Interrupt Controller List Register 0
		1	ICH_LR1	0x00000000	Interrupt Controller List Register 1
		2	ICH_LR2	0x00000000	Interrupt Controller List Register 2
		3	ICH_LR3	0x00000000	Interrupt Controller List Register 3
	c14	0	ICH_LRC0	0x00000000	Interrupt Controller List Register 0
		1	ICH_LRC1	0x00000000	Interrupt Controller List Register 1
		2	ICH_LRC2	0x00000000	Interrupt Controller List Register 2
		3	ICH_LRC3	0x00000000	Interrupt Controller List Register 3
6	c12	4	ICC_MCTLR	0x00000400	Interrupt Control Register for EL3
		5	ICC_MSRE	0x00000000	System Register Enable Register for EL3
		7	ICC_MGRPEN1	0x00000000	Interrupt Controller Monitor Interrupt Group 1 Enable register

B1.16 c13 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value thirteen are called c13 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c13.

For information on all registers but FCSE Process ID, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table B1-15 c13 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	FCSEIDR	`0x00000000`	*B1.53 FCSE Process ID Register* on page B1-223
		1	CONTEXTIDR	UNK	Context ID Register
		2	TPIDRURW	UNK	User Read/Write Thread ID Register
		3	TPIDRURO	UNK	User Read-Only Thread ID Register
		4	TPIDRPRW	UNK	EL1 only Thread ID Register
4	c0	2	HTPIDR	UNK	Hyp Software Thread ID Register

B1.17 c14 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value fourteen are called c14 registers.

The following table shows the CP15 system registers when the processor is in AArch32 state and the value of CRn is c14. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

Table B1-16 c14 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	CNTFRQ	UNK	Timer Counter Frequency Register
	c1	0	CNTKCTL	-	Timer Control Register The reset value for bits[9:8, 2:0] is `0b00000`.
	c2	0	CNTP_TVAL	UNK	Physical Timer Value Register
		1	CNTP_CTL	-	Physical Timer Control Register The reset value for bit[0] is 0.
	c3	0	CNTV_TVAL	UNK	Virtual Timer TimerValue Register
		1	CNTV_CTL	-	Counter-timer Virtual Timer Control Register The reset value for bit[0] is 0.
	c8	0	PMEVCNTR0	UNK	Performance Monitor Event Count Registers
		1	PMEVCNTR1	UNK	
		2	PMEVCNTR2	UNK	
		3	PMEVCNTR3	UNK	
		4	PMEVCNTR4	UNK	
		5	PMEVCNTR5	UNK	
	c12	0	PMEVTYPER0	UNK	Performance Monitor Event Type Registers
		1	PMEVTYPER1	UNK	
		2	PMEVTYPER2	UNK	
		3	PMEVTYPER3	UNK	
		4	PMEVTYPER4	UNK	
		5	PMEVTYPER5	UNK	
	c15	7	PMCCFILTR	`0x00000000`	Performance Monitor Cycle Count Filter Register.
4	c1	0	CNTHCTL	-	Timer Control Register (EL2) The reset value for bit[2] is 0 and for bits[1:0] is `0b11`.
	c2	0	CNTHP_TVAL	UNK	Physical Timer TimerValue (EL2)
		1	CNTHP_CTL	-	Physical Timer Control Register (EL2) The reset value for bit[0] is 0.

B1.18 c15 registers

The processor can access different 32-bit wide system registers. Registers where CRn has the value fifteen are called c15 registers.

The following table shows the 32-bit wide system registers you can access when the processor is in AArch32 state and the value of CRn is c15.

Table B1-17 c15 register summary

Op1	CRm	Op2	Name	Reset	Description
1	c0	0	L2ACTLR	`0x80000000`	*B1.91 L2 Auxiliary Control Register* on page B1-295 This is the reset value for an ACE interface. For an AXI interface the reset value is `0x80000008`. For a CHI interface the reset value is `0x80004008`.
	c3	0	CBAR	-	*B1.38 Configuration Base Address Register* on page B1-194 The reset value depends on the processor configuration.
3	c0	0	CDBGDR0	UNK	Cache Debug Data Register 0, see *C5.1 About direct access to internal memory* on page C5-398
		1	CDBGDR1	UNK	Cache Debug Data Register 1, see *C5.1 About direct access to internal memory* on page C5-398
		2	CDBGDR2	UNK	Cache Debug Data Register 2, see *C5.1 About direct access to internal memory* on page C5-398
		3	CDBGDR3	UNK	Cache Debug Data Register 3, see *C5.1 About direct access to internal memory* on page C5-398
	c2	0	CDBGDCT	UNK	Cache Debug Data Cache Tag Read Operation Register, see *C5.1 About direct access to internal memory* on page C5-398
		1	CDBGICT	UNK	Cache Debug Instruction Cache Tag Read Operation Register, see *C5.1 About direct access to internal memory* on page C5-398
	c4	0	CDBGDCD	UNK	Cache Debug Cache Debug Data Cache Data Read Operation Register, see *C5.1 About direct access to internal memory* on page C5-398
		1	CDBGICD	UNK	Cache Debug Instruction Cache Data Read Operation Register, see *C5.1 About direct access to internal memory* on page C5-398
		2	CDBGTD	UNK	Cache Debug TLB Data Read Operation Register, see *C5.1 About direct access to internal memory* on page C5-398

B1.19 64-bit registers

The processor can access 64-bit wide CP15 system registers in AArch32 state.

The following table shows the 64-bit wide CP15 system registers, accessed by the `MCRR` and `MRRC` instructions. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

Table B1-18 64-bit register summary

Op1	CRm	Name	Reset	Description
0	c2	TTBR0	UNK	Translation Table Base Register 0
1	c2	TTBR1	UNK	Translation Table Base Register 1
4	c2	HTTBR	UNK	Hyp Translation Table Base Register
6	c2	VTTBR	UNK	Virtualization Translation Table Base Register
0	c7	PAR	UNK	*B1.100 Physical Address Register* on page B1-315
0	c14	CNTPCT	UNK	Physical Timer Count Register
1	c14	CNTVCT	UNK	Virtual Timer Count Register
2	c14	CNTP_CVAL	UNK	Physical Timer CompareValue Register
3	c14	CNTV_CVAL	UNK	Virtual Timer CompareValue Register
4	c14	CNTVOFF	UNK	Virtual Timer Offset Register
6	c14	CNTHP_CVAL	UNK	Physical Timer CompareValue Register
0	c15	CPUACTLR	`0x00000000090CA000`	*B1.42 CPU Auxiliary Control Register* on page B1-202
1	c15	CPUECTLR	`0x0000000000000000`	*B1.43 CPU Extended Control Register* on page B1-206
2	c15	CPUMERRSR	-	*B1.44 CPU Memory Error Syndrome Register* on page B1-208
3	c15	L2MERRSR	-	*B1.94 L2 Memory Error Syndrome Register* on page B1-301

B1.20 AArch32 Identification registers

The following table shows the identification registers.

Table B1-19 Identification registers

Name	CRn	Op1	CRm	Op2	Reset	Description
MIDR	c0	0	c0	0	0x411FD010	*B1.96 Main ID Register* on page B1-307
CTR				1	0x84448004	*B1.46 Cache Type Register* on page B1-213
TCMTR				2	0x00000000	*B1.108 TCM Type Register* on page B1-331
TLBTR				3	0x00000000	*B1.109 TLB Type Register* on page B1-332
MPIDR				5	-	*B1.97 Multiprocessor Affinity Register* on page B1-309 The reset value depends on the primary inputs, CLUSTERIDAFF1 and CLUSTERIDAFF2, and the number of cores that the device implements.
REVIDR				6	0x00000000	*B1.102 Revision ID Register* on page B1-319
ID_PFR0			c1	0	0x00000131	*B1.84 Processor Feature Register 0* on page B1-283
ID_PFR1				1	0x10011011	*B1.85 Processor Feature Register 1* on page B1-285 Bits [31:28] are `0x1` if the GIC CPU interface is implemented and enabled, `and 0x0` otherwise.
ID_DFR0				2	0x03010066	*B1.73 Debug Feature Register 0* on page B1-261 Bits [19:16] are `0x1` if ETM is implemented, and `0x0` otherwise.
ID_AFR0				3	0x00000000	*B1.72 Auxiliary Feature Register 0* on page B1-260
ID_MMFR0				4	0x10201105	*B1.80 Memory Model Feature Register 0* on page B1-275
ID_MMFR1				5	0x40000000	*B1.81 Memory Model Feature Register 1* on page B1-277
ID_MMFR2				6	0x01260000	*B1.82 Memory Model Feature Register 2* on page B1-279
ID_MMFR3				7	0x02102211	*B1.83 Memory Model Feature Register 3* on page B1-281
ID_ISAR0			c2	0	0x02101110	*B1.74 Instruction Set Attribute Register 0* on page B1-263
ID_ISAR1				1	0x13112111	*B1.75 Instruction Set Attribute Register 1* on page B1-265
ID_ISAR2				2	0x21232042	*B1.76 Instruction Set Attribute Register 2* on page B1-267
ID_ISAR3				3	0x01112131	*B1.77 Instruction Set Attribute Register 3* on page B1-269
ID_ISAR4				4	0x00011142	*B1.78 Instruction Set Attribute Register 4* on page B1-271

Table B1-19 Identification registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
ID_ISAR5	c0	0	c2	5	0x00011121	*B1.79 Instruction Set Attribute Register 5* on page B1-273 ID_ISAR5 has the value 0x00010001 if the Cryptographic Extension is not implemented and enabled.
CCSIDR		1	c0	0	-	*B1.39 Cache Size ID Register* on page B1-195
CLIDR				1	0x0A200023	*B1.40 Cache Level ID Register* on page B1-198 The value is 0x09200003 if the L2 cache is not implemented.
AIDR				7	0x00000000	*B1.34 Auxiliary ID Register* on page B1-190
CSSELR		2	c0	0	0x00000000	*B1.45 Cache Size Selection Register* on page B1-211

B1.21 AArch32 Virtual memory control registers

The following table shows the virtual memory control registers.

Table B1-20 Virtual memory control registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
SCTLR	c1	0	c0	0	`0x00C50838`	32-bit	*B1.105 System Control Register* on page B1-324 The reset value depends on inputs, **CFGTE**, **CFGEND**, and **VINITHI**. The value listed here assumes these signals are set to LOW.
TTBR0	c2	0	c0	0	UNK	32-bit	Translation Table Base Register 0, see *B1.113 Translation Table Base Register 0* on page B1-338
	-	0	c2	-		64-bit	
TTBR1	c2	0	c0	1	UNK	32-bit	Translation Table Base Register 1, see *B1.116 Translation Table Base Register 1* on page B1-342
	-	1	c2	-		64-bit	
TTBCR	c2	0	c0	2	`0x00000000`	32-bit	*B1.110 Translation Table Base Control Register* on page B1-333 The reset value is `0x00000000` for the Secure copy of the register. The reset value for the EAE bit of the Non-secure copy of the register is `0x0`. You must program the Non-secure copy of the register with the required initial value, as part of the processor boot sequence.
DACR	c3	0	c0	0	UNK	32-bit	*B1.47 Domain Access Control Register* on page B1-215
PRRR	c10	0	c2	0	UNK	32-bit	*B1.101 Primary Region Remap Register* on page B1-316
MAIR0				0	UNK	32-bit	*B1.95 Memory Attribute Indirection Registers 0 and 1* on page B1-304
NMRR				1	UNK	32-bit	*B1.99 Normal Memory Remap Register* on page B1-313
MAIR1				1	UNK	32-bit	*B1.95 Memory Attribute Indirection Registers 0 and 1* on page B1-304
AMAIR0			c3	0	`0x00000000`	32-bit	*B1.36 Auxiliary Memory Attribute Indirection Register 0* on page B1-192
AMAIR1				1	`0x00000000`	32-bit	*B1.37 Auxiliary Memory Attribute Indirection Register 1* on page B1-193
CONTEXTIDR	c13	0	c0	1	UNK	32-bit	Process ID Register, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*

B1.22 AArch32 Fault handling registers

The following table shows the Fault handling registers in the AArch32 Execution state.

Table B1-21 Fault handling registers

Name	CRn	Op1	CRm	Op2	Reset	Description
DFSR	c5	0	c0	0	UNK	*B1.49 Data Fault Status Register* on page B1-217
IFSR				1	UNK	*B1.87 Instruction Fault Status Register* on page B1-288
ADFSR			c1	0	`0x00000000`	*B1.33 Auxiliary Data Fault Status Register* on page B1-189
AIFSR				1	`0x00000000`	*B1.35 Auxiliary Instruction Fault Status Register* on page B1-191
DFAR	c6	0	c0	0	UNK	Data Fault Address Register, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*
IFAR				2	UNK	Instruction Fault Address Register, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*

The Virtualization registers include additional fault handling registers. See *B1.28 AArch32 Virtualization registers* on page B1-180 for more information.

B1.23 AArch32 Other System control registers

The following table shows the other system registers.

Table B1-22 Other system registers

Name	CRn	Op1	CRm	Op2	Reset	Description
ACTLR	c1	0	c0	1	`0x00000000`	*B1.32 Auxiliary Control Register* on page B1-187
CPACR				2	`0x00000000`	*B1.41 Architectural Feature Access Control Register* on page B1-200
FCSEIDR	c13	0	c0	0	`0x00000000`	*B1.53 FCSE Process ID Register* on page B1-223

B1.24 AArch32 Address registers

The following table shows the address translation register and operations.

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

Table B1-23 Address translation operations

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
PAR	c7	0	c4	0	UNK	32-bit	*B1.100 Physical Address Register* on page B1-315
	-	0	c7	-		64-bit	

B1.25 AArch32 Thread registers

The following table shows the miscellaneous operations.

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

Table B1-24 Miscellaneous System instructions

Name	CRn	Op1	CRm	Op2	Reset	Description
TPIDRURW	c13	0	c0	2	UNK	User Read/Write Thread ID Register
TPIDRURO				3	UNK	User Read-Only Thread ID Register
TPIDRPRW				4	UNK	EL1 only Thread ID Register
HTPIDR		4	c0	2	UNK	Hyp Software Thread ID Register

B1.26 AArch32 Performance monitor registers

The following table shows the performance monitor registers.

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

Table B1-25 Performance monitor registers

Name	CRn	Op1	CRm	Op2	Reset	Description
PMCR	c9	0	c12	0	`0x41063000`	*C9.2 Performance Monitors Control Register* on page C9-474
PMCNTENSET				1	UNK	Performance Monitors Count Enable Set Register
PMCNTENCLR				2	UNK	Performance Monitors Count Enable Clear Register
PMOVSR				3	UNK	Performance Monitors Overflow Flag Status Register
PMSWINC				4	UNK	Performance Monitors Software Increment Register
PMSELR				5	UNK	Performance Monitors Event Counter Selection Register
PMCEID0				6	`0x6FFFBFFF`	*C9.3 Performance Monitors Common Event Identification Register 0* on page C9-477 The reset value is `0x6E3FBFFF` if L2 cache is not implemented.
PMCEID1				7	`0x00000000`	*C9.4 Performance Monitors Common Event Identification Register 1* on page C9-481
PMCCNTR			c13	0	UNK	Performance Monitors Cycle Count Register
PMXEVTYPER				1	UNK	Performance Monitors Selected Event Type Register
PMXEVCNTR				2	UNK	Performance Monitors Event Count Registers
PMUSERENR			c14	0	`0x00000000`	Performance Monitors User Enable Register
PMINTENSET				1	UNK	Performance Monitors Interrupt Enable Set Register
PMINTENCLR				2	UNK	Performance Monitors Interrupt Enable Clear Register
PMOVSSET				3	UNK	Performance Monitor Overflow Flag Status Set Register

Table B1-25 Performance monitor registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
PMEVCNTR0	c14	0	c8	0	UNK	Performance Monitors Event Count Register 0
PMEVCNTR1				1	UNK	
PMEVCNTR2				2	UNK	
PMEVCNTR3				3	UNK	
PMEVCNTR4				4	UNK	
PMEVCNTR5				5	UNK	
PMEVTYPER0			c12	0	UNK	Performance Monitors Selected Event Type Register 0
PMEVTYPER1				1	UNK	
PMEVTYPER2				2	UNK	
PMEVTYPER3				3	UNK	
PMEVTYPER4				4	UNK	
PMEVTYPER5				5	UNK	
PMCCFILTR			c15	7	`0x00000000`	Performance Monitors Cycle Count Filter Register

B1.27 AArch32 Secure registers

The following table shows the Secure registers.

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

Table B1-26 Security registers

Name	CRn	Op1	CRm	Op2	Reset	Description
SCR	c1	0	c1	0	`0x00000000`	*B1.104 Secure Configuration Register* on page B1-321
SDER				1	UNK	Secure Debug Enable Register
NSACR				2	`0x00000000`	*B1.98 Non-Secure Access Control Register* on page B1-311
VBAR	c12	0	c0	0	`0x00000000`	*B1.119 Vector Base Address Register* on page B1-346 `0x00000000` is the secure reset value and UNK is the non-secure reset value.
MVBAR				1	UNK	Monitor Vector Base Address Register
ISR			c1	0	UNK	Interrupt Status Register

B1.28 AArch32 Virtualization registers

The following table shows the Virtualization registers.

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

Table B1-27 Virtualization registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
VPIDR	c0	4	c0	0	`0x411FD010`	32-bit	*B1.121 Virtualization Processor ID Register* on page B1-348
VMPIDR				5	-	32-bit	*B1.120 Virtualization Multiprocessor ID Register* on page B1-347 The reset value is the value of the Multiprocessor Affinity Register.
HSCTLR	c1	4	c0	0	`0x30C50838`	32-bit	*B1.67 Hyp System Control Register* on page B1-248
HACTLR				1	UNK		*B1.55 Hyp Auxiliary Control Register* on page B1-225
HCR			c1	0	`0x00000000`	32-bit	Hyp Configuration Register
HDCR				1	`0x00000006`	32-bit	*B1.63 Hyp Debug Control Register* on page B1-242
HCPTR				2	`0x000033FF`	32-bit	*B1.60 Hyp Architectural Feature Trap Register* on page B1-231 The reset value depends on the FPU and NEON configuration. If Advanced SIMD and floating-point are implemented, the reset value is `0x000033FF`. If Advanced SIMD and floating-point are not implemented, the reset value is `0x0000BFFF`.
HSTR				3	`0x00000000`	32-bit	Hypervisor System Trap Register
HTCR	c2	4	c0	2	UNK	32-bit	*B1.70 Hyp Translation Control Register* on page B1-257
VTCR			c1	2	UNK	32-bit	Virtualization Translation Control Register
HTTBR	-	4	c2	-	UNK	64-bit	Hyp Translation Table Base Register
VTTBR	-	6	c2	-	UNK	64-bit	Virtualization Translation Table Base Register
HADFSR	c5	4	c1	0	`0x00000000`	32-bit	*B1.56 Hyp Auxiliary Data Fault Status Syndrome Register* on page B1-227
HAIFSR				1	`0x00000000`	32-bit	*B1.57 Hyp Auxiliary Instruction Fault Status Syndrome Register* on page B1-228
HSR			c2	0	UNK	32-bit	*B1.68 Hyp Syndrome Register* on page B1-252
HDFAR	c6	4	c0	0	UNK	32-bit	Hyp Data Fault Address Register
HIFAR				2	UNK	32-bit	Hyp Instruction Fault Address Register
HPFAR				4	UNK	32-bit	Hyp IPA Fault Address Register

Table B1-27 Virtualization registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
HMAIR0	c10	4	c2	0	UNK	32-bit	Hyp Memory Attribute Indirection Register 0
HMAIR1				1	UNK	32-bit	Hyp Memory Attribute Indirection Register 1
HAMAIR0			c3	0	0x00000000	32-bit	*B1.58 Hyp Auxiliary Memory Attribute Indirection Register 0* on page B1-229
HAMAIR1				1	0x00000000	32-bit	*B1.59 Hyp Auxiliary Memory Attribute Indirection Register 1* on page B1-230
HVBAR	c12	4	c0	0	UNK	32-bit	Hyp Vector Base Address Register

B1.29 AArch32 GIC system registers

The following table shows the GIC system registers in AArch32 state.

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

Table B1-28 AArch32 GIC system registers

Name	CRn	Op1	CRm	Op2	Type	Reset	Width	Description
ICC_SGI1R	-	0	c12	-	WO	-	64-bit	SGI Generation Register 1
ICC_ASGI1R	-	1	c12	-	WO	-	64-bit	Alternate SGI Generation Register 1
ICC_SGI0R	-	2	c12	-	WO	-	64-bit	SGI Generation Register 0
ICC_PMR	c4	0	c6	0	RW	0x00000000	32-bit	Priority Mask Register
ICC_IAR0	c12	0	c8	0	RO	-	32-bit	Interrupt Acknowledge Register 0
ICC_EOIR0				1	WO	-	32-bit	End Of Interrupt Register 0
ICC_HPPIR0				2	RO	-	32-bit	Highest Priority Pending Interrupt Register 0
ICC_BPR0				3	RW	0x00000002	32-bit	Binary Point Register 0
ICC_AP0R0				4	RW	0x00000000	32-bit	Active Priorities 0 Register 0
ICC_AP1R0			c9	0	RW	0x00000000	32-bit	Active Priorities 1 Register 0
ICC_DIR			c11	1	WO	-	32-bit	Deactivate Interrupt Register
ICC_RPR				3	RO	-	32-bit	Running Priority Register
ICC_IAR1			c12	0	RO	-	32-bit	Interrupt Acknowledge Register 1
ICC_EOIR1				1	WO	-	32-bit	End Of Interrupt Register 1
ICC_HPPIR1				2	RO	-	32-bit	Highest Priority Pending Interrupt Register 1
ICC_BPR1				3	RW	0x00000003	32-bit	Binary Point Register 1 This is the reset value in non-secure state. In secure state, the reset value is 0x00000002.
ICC_CTLR				4	RW	0x00000400	32-bit	Interrupt Control Register
ICC_SRE				5	RW	0x00000000	32-bit	System Register Enable Register
ICC_IGRPEN0				6	RW	0x00000000	32-bit	Interrupt Group Enable Register 0
ICC_IGRPEN1				7	RW	0x00000000	32-bit	Interrupt Group Enable Register 1
ICH_AP0R0		4	c8	0	RW	0x00000000	32-bit	Interrupt Controller Hyp Active Priorities Register (0,0)
ICH_AP1R0			c9	0	RW	0x00000000	32-bit	Interrupt Controller Hyp Active Priorities Register (1,0)
ICC_HSRE				5	RW	0x00000000	32-bit	System Register Enable Register for EL2
ICH_HCR			c11	0	RW	0x00000000	32-bit	Interrupt Controller Hyp Control Register
ICH_VTR				1	RO	0x90080003	32-bit	Interrupt Controller VGIC Type Register
ICH_MISR				2	RO	0x00000000	32-bit	Interrupt Controller Maintenance Interrupt State Register

Table B1-28 AArch32 GIC system registers (continued)

Name	CRn	Op1	CRm	Op2	Type	Reset	Width	Description
ICH_EISR	c12	4	c8	3	RO	0x00000000	32-bit	Interrupt Controller End of Interrupt Status Register
ICH_VMCR				7	RW	0x004C0000	32-bit	Interrupt Controller Virtual Machine Control Register
ICH_ELRSR				5	RO	0x0000000F	32-bit	Interrupt Controller Empty List Register Status Register
ICH_LR0			c12	0	RW	0x00000000	32-bit	Interrupt Controller List Register 0
ICH_LR1				1	RW	0x00000000	32-bit	Interrupt Controller List Register 1
ICH_LR2				2	RW	0x00000000	32-bit	Interrupt Controller List Register 2
ICH_LR3				3	RW	0x00000000	32-bit	Interrupt Controller List Register 3
ICH_LRC0			c14	0	RW	0x00000000	32-bit	Interrupt Controller List Register 0
ICH_LRC1				1	RW	0x00000000	32-bit	Interrupt Controller List Register 1
ICH_LRC2				2	RW	0x00000000	32-bit	Interrupt Controller List Register 2
ICH_LRC3				3	RW	0x00000000	32-bit	Interrupt Controller List Register 3
ICC_MCTLR		6	c12	4	RW	0x00000400	32-bit	Interrupt Control Register for EL3
ICC_MSRE				5	RW	0x00000000	32-bit	System Register Enable Register for EL3
ICC_MGRPEN1				7	RW	0x00000000	32-bit	Interrupt Controller Monitor Interrupt Group 1 Enable register

B1.30 AArch32 Generic Timer registers

The processor implements the architecturally defined Generic Timer registers.

The AArch32 Generic Timer registers are described in *B3.1 AArch32 Generic Timer register summary* on page B3-362. See also the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.* for more information.

B1.31 AArch32 Implementation defined registers

IMPLEMENTATION DEFINED registers provide test features and any required configuration options specific to the Cortex-A32 processor.

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

Table B1-29 Memory access registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
L2CTLR	c9	1	c0	2	-	32-bit	*B1.92 L2 Control Register* on page B1-297 The reset value depends on the processor configuration.
L2ECTLR				3	`0x00000000`	32-bit	*B1.93 L2 Extended Control Register* on page B1-299
L2ACTLR	c15	1	c0	0	`0x80000000`	32-bit	*B1.91 L2 Auxiliary Control Register* on page B1-295 This is the reset value for an ACE interface. For an AXI interface the reset value is `0x80000008`. For a CHI interface the reset value is `0x80004008`.
CBAR			c3	0	-	32-bit	*B1.38 Configuration Base Address Register* on page B1-194 The reset value depends on the processor configuration.
CDBGDR0		3	c0	0	UNK	32-bit	Data Register 0, see *C5.1 About direct access to internal memory* on page C5-398 for information on how these registers are used.
CDBGDR1				1	UNK	32-bit	Data Register 1, see *C5.1 About direct access to internal memory* on page C5-398
CDBGDR2				2	UNK	32-bit	Data Register 2, see *C5.1 About direct access to internal memory* on page C5-398
CDBGDR3				3	UNK	32-bit	Data Register 3, see *C5.1 About direct access to internal memory* on page C5-398.
CDBGDCT			c2	0	UNK	32-bit	Data Cache Tag Read Operation Register, see *C5.1 About direct access to internal memory* on page C5-398
CDBGICT				1	UNK	32-bit	Instruction Cache Tag Read Operation Register, see *C5.1 About direct access to internal memory* on page C5-398
CDBGDCD			c4	0	UNK	32-bit	Data Cache Data Read Operation Register, see *C5.1 About direct access to internal memory* on page C5-398
CDBGICD				1	UNK	32-bit	Instruction Cache Data Read Operation Register, see *C5.1 About direct access to internal memory* on page C5-398
CDBGTD				2	UNK	32-bit	TLB Data Read Operation Register, see *C5.1 About direct access to internal memory* on page C5-398

Table B1-29 Memory access registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
CPUACTLR	-	0	c15	-	0x00000000090CA000	64-bit	*B1.42 CPU Auxiliary Control Register* on page B1-202
CPUECTLR	-	1	c15	-	0x0000000000000000	64-bit	*B1.43 CPU Extended Control Register* on page B1-206
CPUMERRSR	-	2	c15	-	-	64-bit	*B1.44 CPU Memory Error Syndrome Register* on page B1-208
L2MERRSR	-	3	c15	-	-	64-bit	*B1.94 L2 Memory Error Syndrome Register* on page B1-301

B1.32 Auxiliary Control Register

The ACTLR characteristics are:

Purpose

Controls write access to IMPLEMENTATION DEFINED registers in EL2, such as CPUACTLR, CPUECTLR, L2CTLR, L2ECTLR, and L2ACTLR.

Usage constraints

This register is accessible as follows:

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

Configurations

The processor does not implement the ACTLR (NS) register. This register is always RES0.

Attributes

ACTLR is a 32-bit register.

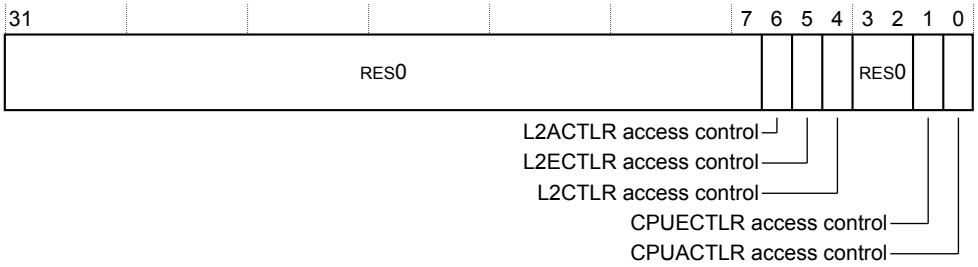

Figure B1-1 ACTLR bit assignments

[31:7]

Reserved, RES0.

L2ACTLR access control, [6]

L2ACTLR write access control. The possible values are:

`0` The register is not write accessible from a lower exception level. This is the reset value.

`1` The register is write accessible from EL2.

L2ECTLR access control, [5]

L2ECTLR write access control. The possible values are:

`0` The register is not write accessible from a lower exception level. This is the reset value.

`1` The register is write accessible from EL2.

L2CTLR access control, [4]

L2CTLR write access control. The possible values are:

`0` The register is not write accessible from a lower exception level. This is the reset value.

`1` The register is write accessible from EL2.

[3:2]

Reserved, RES0.

CPUECTLR access control, [1]

CPUECTLR write access control. The possible values are:

`0` The register is not write accessible from a lower exception level. This is the reset value.

`1` The register is write accessible from EL2.

CPUACTLR access control, [0]

CPUACTLR write access control. The possible values are:

`0` The register is not write accessible from a lower exception level. This is the reset value.

`1` The register is write accessible from EL2.

To access the ACTLR:

```
MRC p15, 0, <Rt>, c1, c0, 1 ; Read ACTLR into Rt
MCR p15, 0, <Rt>, c1, c0, 1 ; Write Rt to ACTLR
```

Register access is encoded as follows:

Table B1-30 ACTLR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0001	0000	001

B1.33 Auxiliary Data Fault Status Register

ADFSR

The processor does not implement ADFSR. This register is always RES0.

B1.34 Auxiliary ID Register

AIDR

The processor does not implement AIDR. This register is always RES0.

B1.35 Auxiliary Instruction Fault Status Register

AIFSR

The processor does not implement AIFSR. This register is always RES0.

B1.36 Auxiliary Memory Attribute Indirection Register 0

AMAIR0

The processor does not implement AMAIR0. This register is always RES0.

B1.37 Auxiliary Memory Attribute Indirection Register 1

AMAIR1

The processor does not implement AMAIR1. This register is always RES0.

B1.38 Configuration Base Address Register

The CBAR characteristics are:

Purpose

Holds the physical base address of the memory-mapped GIC CPU interface registers.

Usage constraints

This register is accessible as follows:

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

Configurations

The CBAR is Common to the Secure and Non-secure states.

Attributes

CBAR is a 32-bit register.

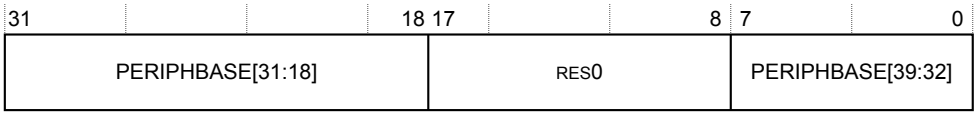

Figure B1-2 CBAR bit assignments

PERIPHBASE[31:18], [31:18]

If the processor is implemented with the GIC CPU interface, the input **PERIPHBASE[31:18]** determines the reset value. If the GIC CPU interface is not implemented, this field is RAZ.

[17:8]

Reserved, RES0.

PERIPHBASE[39:32], [7:0]

If the processor is implemented with the GIC CPU interface, the input **PERIPHBASE[39:32]** determines the reset value. If the GIC CPU interface is not implemented, this field is RAZ.

To access the CBAR:

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

Register access is encoded as follows:

Table B1-31 CBAR access encoding

coproc	opc1	CRn	CRm	opc2
1111	001	1111	0011	000

B1.39 Cache Size ID Register

The CCSIDR characteristics are:

Purpose

Provides information about the architecture of the caches.

Usage constraints

This register is accessible as follows:

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

If CSSELR indicates a cache that is not implemented, then on a read of the CCSIDR the behavior is CONSTRAINED UNPREDICTABLE, and can be one of the following:

- The CCSIDR read is treated as `NOP`.
- The CCSIDR read is UNDEFINED.
- The CCSIDR read returns an UNKNOWN value (preferred).

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

The implementation includes one CCSIDR for each cache that it can access. CSSELR selects which Cache Size ID Register is accessible.

Attributes

CCSIDR is a 32-bit register.

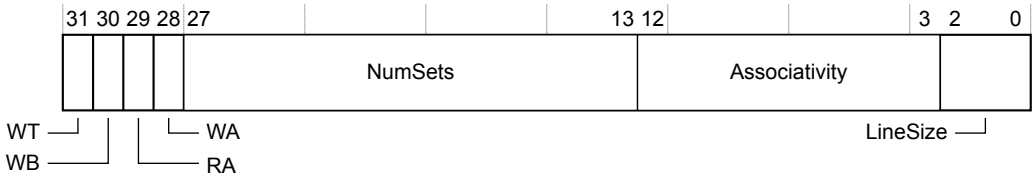

Figure B1-3 CCSIDR bit assignments

WT, [31]

Indicates support for Write-Through:

`0` Cache level does not support Write-Through.

WB, [30]

Indicates support for Write-Back:

`0` Cache level does not support Write-Back.

`1` Cache level supports Write-Back.

RA, [29]

Indicates support for Read-Allocation:

`0` Cache level does not support Read-Allocation.

`1` Cache level supports Read-Allocation.

WA, [28]

Indicates support for Write-Allocation:

`0` Cache level does not support Write-Allocation.

`1` Cache level supports Write-Allocation.

NumSets, [27:13]

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.

For more information about encoding, see *Table B1-32 CCSIDR encodings* on page B1-196.

Associativity, [12:3]

Indicates the associativity of cache - 1. Therefore, a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2.

For more information about encoding, see *Table B1-32 CCSIDR encodings* on page B1-196.

LineSize, [2:0]

Indicates the ($\log_2$ (number of words in cache line)) - 2:

`0b010` 16 words per line.

For more information about encoding, see *Table B1-32 CCSIDR encodings* on page B1-196.

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

Table B1-32 CCSIDR encodings

CSSELR	Cache	Size	Complete register encoding	Register bit field encoding						
				WT	WB	RA	WA	NumSets	Associativity	LineSize
0x0	L1 Data cache	8KB	0x7003E01A	0	1	1	1	0x001F	0x003	0x2
		16KB	0x7007E01A					0x003F	0x003	0x2
		32KB	0x700FE01A					0x007F	0x003	0x2
		64KB	0x701FE01A					0x00FF	0x003	0x2
0x1	L1 Instruction cache	8KB	0x2007E00A	0	0	1	0	0x003F	0x001	0x2
		16KB	0x200FE00A					0x007F	0x001	0x2
		32KB	0x201FE00A					0x00FF	0x001	0x2
		64KB	0x203FE00A					0x001F	0x001	0x2
0x2	L2 cache	128KB	0x701FE03A	0	1	1	1	0x00FF	0x007	0x2
		256KB	0x703FE03A					0x01FF	0x007	0x2
		512KB	0x707FE03A					0x03FF	0x007	0x2
		1024KB	0x70FFE03A					0x07FF	0x007	0x2
0x3-0xF	Reserved	-	-	-	-	-	-	-	-	-

To access the CCSIDR:

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

Register access is encoded as follows:

Table B1-33 CCSIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	001	0000	0000	000

B1.40 Cache Level ID Register

The CLIDR characteristics are:

Purpose

Identifies:

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

Usage constraints

This register is accessible as follows:

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

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

CLIDR is a 32-bit register.

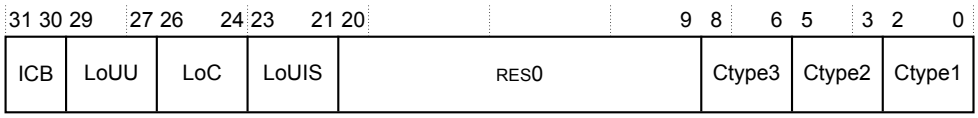

Figure B1-4 CLIDR bit assignments

ICB, [31:30]

Inner cache boundary. This field indicates the boundary between the inner and the outer domain.

`0b00` Not disclosed in this mechanism.

LoUU, [29:27]

Indicates the Level of Unification Uniprocessor for the cache hierarchy:

`0b001` L1 cache is the last level of cache that must be cleaned or invalidated when cleaning or invalidating to the point of unification for the processor.

LoC, [26:24]

Indicates the Level of Coherency for the cache hierarchy:

`0b001` L2 cache not implemented.

`0b010` A clean to the point of coherency operation requires the L1 and L2 caches to be cleaned.

LoUIS, [23:21]

Indicates the Level of Unification Inner Shareable for the cache hierarchy:

`0b001` L2 cache not implemented or **BROADCASTINNER** set to 0.

The L1 cache is the last level of cache that must be cleaned or invalidated when cleaning or invalidating to the point of unification for the Inner Shareable shareability domain.

`0b010` L2 cache implemented and **BROADCASTINNER** set to 1.

The L2 cache is the last level of cache that must be cleaned or invalidated when cleaning or invalidating to the point of unification for the Inner Shareable shareability domain.

[20:9]

Reserved, RES0.

Ctype3, [8:6]

Indicates the type of cache if the processor implements L3 cache:

`0b000` L3 cache not implemented.

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

Ctype2, [5:3]

Indicates the type of cache if the processor implements L2 cache:

`0b000` L2 cache is not implemented.

`0b100` L2 cache is implemented as a unified cache.

Ctype1, [2:0]

Indicates the type of cache implemented at L1:

`0b011` Separate instruction and data caches at L1.

To access the CLIDR:

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

Register access is encoded as follows:

Table B1-34 CLIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	001	0000	0000	001

B1.41 Architectural Feature Access Control Register

The CPACR characteristics are:

Purpose

Controls access to CP0 to CP13, and indicates which of CP0 to CP13 are implemented.

Usage constraints

This register is accessible as follows:

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

The CPACR has no effect on instructions executed at EL2.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Bits in the NSACR control Non-secure access to the CPACR fields. See the field descriptions cp10 and cp11.

Attributes

CPACR is a 32-bit register.

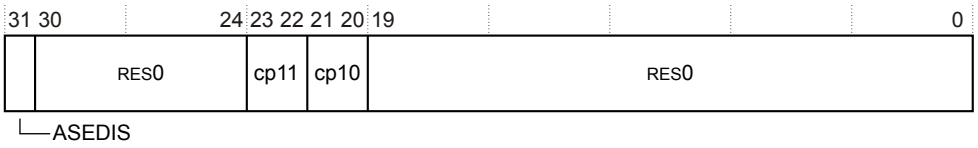

Figure B1-5 CPACR bit assignments

ASEDIS, [31]

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.

If Advanced SIMD and floating-point are not implemented, this bit is RES0.

[30:24]

Reserved, RES0.

cp11, [23:22]

Defines the access rights for CP11, that control the Advanced SIMD and floating-point features. Possible values of the fields are:

`0b00` Access denied. Any attempt to access Advanced SIMD and floating-point registers or instructions generates an Undefined Instruction exception. This is the reset value.

`0b01` Access at EL1 only. Any attempt to access Advanced SIMD and floating-point registers or instructions from software executing at EL0 generates an Undefined Instruction exception.

`0b10` Reserved.

`0b11` Full access.

If Advanced SIMD and floating-point are not implemented, this field is RES0.

The Advanced SIMD and floating-point features controlled by these fields are:

- Floating-point instructions.
- Advanced SIMD instructions, both integer and floating-point.
- Advanced SIMD and floating-point registers D0-D31 and their views as S0-S31 and Q0-Q15.
- FPSCR, FPSID, MVFR0, MVFR1, MVFR2, FPEXC system registers.

If the cp11 and cp10 fields are set to different values, the behavior is the same as if both fields were set to the value of cp10, in all respects other than the value read back by explicitly reading cp11.

cp10, [21:20]

Defines the access rights for CP10, that control the Advanced SIMD and floating-point features. Possible values of the fields are:

`0b00` Access denied. Any attempt to access Advanced SIMD and floating-point registers or instructions generates an Undefined Instruction exception. This is the reset value.

`0b01` Access at EL1 only. Any attempt to access Advanced SIMD and floating-point registers or instructions from software executing at EL0 generates an Undefined Instruction exception.

`0b10` Reserved.

`0b11` Full access.

If Advanced SIMD and floating-point are not implemented, this bit is RES0.

The Advanced SIMD and floating-point features controlled by these fields are:

- Floating-point instructions.
- Advanced SIMD instructions, both integer and floating-point.
- Advanced SIMD and floating-point registers D0-D31 and their views as S0-S31 and Q0-Q15.
- FPSCR, FPSID, MVFR0, MVFR1, MVFR2, FPEXC system registers.

[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
```

Register access is encoded as follows:

Table B1-35 CPACR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0001	0000	010

B1.42 CPU Auxiliary Control Register

The CPUACTLR characteristics are:

Purpose

Provides IMPLEMENTATION DEFINED configuration and control options for the processor. There is one 64-bit CPU Auxiliary Control Register for each core in the processor.

Usage constraints

This register is accessible as follows:

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

The CPU Auxiliary Control Register can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled, and before any master interface or ACP traffic begins.

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.

Configurations

CPUACTLR is common to the Secure and Non-secure states.

Attributes

CPUACTLR is a 64-bit register.

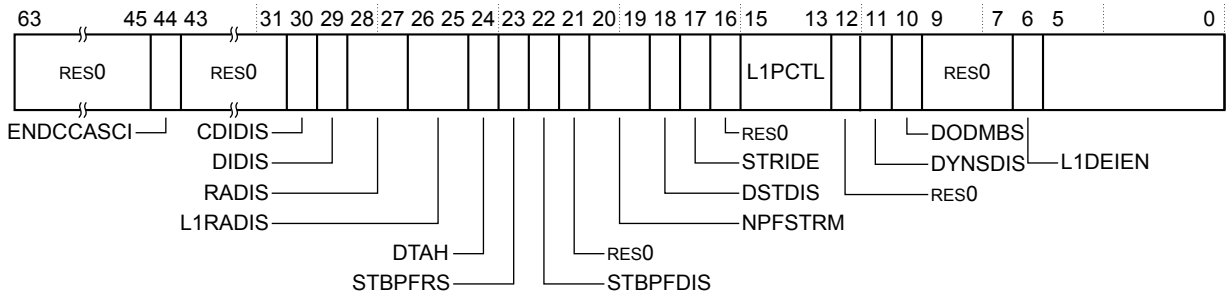

Figure B1-6 CPUACTLR bit assignments

[63:45]

Reserved, RES0.

ENDCCASCI, [44]

Enable data cache clean as data cache clean/invalidate. The possible values are:

`0` Normal behavior, data cache clean operations are unaffected. This is the reset value.

`1` Executes data cache clean operations as data cache clean and invalidate. The following operations are affected:

- DCCSW is executed as DCCISW, DCCMVAU and DCCMVAC are executed as DCCIMVAC.

[43:31]

Reserved, RES0.

CDIDIS, [30]

Disable Cryptographic dual issue. The possible values are:

`0` Enable dual issue of Advanced SIMD and Cryptographic instructions. This is the reset value.

`1` Disable dual issue of Advanced SIMD and Cryptographic instructions.

DIDIS, [29]

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.

RADIS, [28:27]

Write streaming no-allocate threshold. The possible values are:

`0b00` 16th consecutive streaming cache line does not allocate in the L1 or L2 cache.

`0b01` 128th consecutive streaming cache line does not allocate in the L1 or L2 cache. This is the reset value.

`0b10` 512th consecutive streaming cache line does not allocate in the L1 or L2 cache.

`0b11` Disables streaming. All write-allocate lines allocate in the L1 or L2 cache.

L1RADIS, [26:25]

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.

DTAH, [24]

Disable transient and no-read-allocate hints for loads. The possible values are:

`0` Normal operation.

`1` Transient and no-read-allocate hints in the MAIR are ignored and treated the same as non-transient, read-allocate types for loads. This is the reset value.

STBPFRS, [23]

Disable ReadUnique request for prefetch streams initiated by STB accesses:

`0` ReadUnique used for prefetch streams initiated from STB accesses. This is the reset value.

`1` ReadShared used for prefetch streams initiated from STB accesses.

STBPFDIS, [22]

Disable prefetch streams initiated from STB accesses:

`0` Enable Prefetch streams initiated from STB accesses. This is the reset value.

`1` Disable Prefetch streams initiated from STB accesses.

[21]

Reserved, RES0.

NPFSTRM, [20:19]

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.

DSTDIS, [18]

Enable device split throttle. The possible values are:

`0` Device split throttle disabled.

`1` Device split throttle enabled. This is the reset value.

STRIDE, [17]

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.

L1PCTL, [15:13]

L1 Data prefetch control. The value of the 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.

DYNSDIS, [11]

Disable dynamic stride adjustment for prefetch streams. The possible values are:

0 Enable dynamic stride adjustment. This is the reset value.

1 Disable dynamic stride adjustment.

DODMBS, [10]

Disable optimized Data Memory Barrier behavior. The possible values are:

`0` Enable optimized Data Memory Barrier behavior. This is the reset value.

`1` Disable optimized Data Memory Barrier behavior.

[9:7]

Reserved, RES0.

L1DEIEN, [6]

L1 D-cache data RAM error injection enable. The possible values are;

`0` Normal behavior, errors are not injected. This is the reset value.

`1` Double-bit errors are injected on all writes to the L1 D-cache data RAMs for the first word of each 32-byte region.

[5:0]

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
```

Register access is encoded as follows:

Table B1-36 CPUACTLR access encoding

coproc	opc1	CRm
1111	0000	1111

B1.43 CPU Extended Control Register

The CPUECTLR characteristics are:

Purpose

Provides additional IMPLEMENTATION DEFINED configuration and control options for the processor.

Usage constraints

This register is accessible as follows:

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

The CPUECTLR can be written dynamically.

Configurations

There are no configuration notes.

Attributes

CPUECTLR is a 64-bit register.

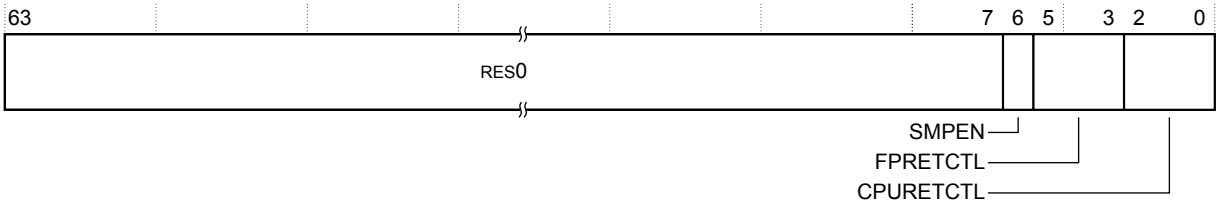

Figure B1-7 CPUECTLR bit assignments

[63:7]

Reserved, RES0.

SMPEN, [6]

Enable hardware management of data coherency with other cores in the cluster. The possible values are:

`0` Disables data coherency with other cores in the cluster. This is the reset value.

`1` Enables data coherency with other cores in the cluster.

Set the SMPEN bit before enabling the caches, even if there is only one core in the system.

FPRETCTL, [5:3]

Advanced SIMD and floating-point retention control. The possible values are:

`0b000` Disable the retention circuit. This is the reset value.

`0b001` 2 Architectural Timer ticks are required before retention entry.

`0b010` 8 Architectural Timer ticks are required before retention entry.

`0b011` 32 Architectural Timer ticks are required before retention entry.

`0b100` 64 Architectural Timer ticks are required before retention entry.

`0b101` 128 Architectural Timer ticks are required before retention entry.

`0b110` 256 Architectural Timer ticks are required before retention entry.

`0b111` 512 Architectural Timer ticks are required before retention entry.

This field is present only if the Advanced SIMD and floating-point support is implemented. Otherwise, it is RES0.

CPURETCTL, [2:0]

CPU retention control. The possible values are:

`0b000` Disable the retention circuit. This is the reset value.

`0b001` 2 Architectural Timer ticks are required before retention entry.

`0b010` 8 Architectural Timer ticks are required before retention entry.

`0b011` 32 Architectural Timer ticks are required before retention entry.

`0b100` 64 Architectural Timer ticks are required before retention entry.

`0b101` 128 Architectural Timer ticks are required before retention entry.

`0b110` 256 Architectural Timer ticks are required before retention entry.

`0b111` 512 Architectural Timer ticks are required before retention entry.

To access the CPUECTLR:

```
MRRC p15, 1, <Rt>, <Rt2>, c15; Read CPU Extended Control Register
MCRR p15, 1, <Rt>, <Rt2>, c15; Write CPU Extended Control Register
```

Register access is encoded as follows:

Table B1-37 CPUECTLR access encoding

coproc	opc1	CRm
1111	0001	1111

B1.44 CPU Memory Error Syndrome Register

The CPUMERRSR characteristics are:

Purpose

Holds ECC errors on the:

- L1 data RAMs.
- L1 tag RAMs.
- TLB RAMs.

This register is used for recording ECC errors on all processor RAMs.

Usage constraints

This register is accessible as follows:

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

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

A write of any value to the register updates the register to `0000000000000000`.

Attributes

CPUMERRSR is a 64-bit register.

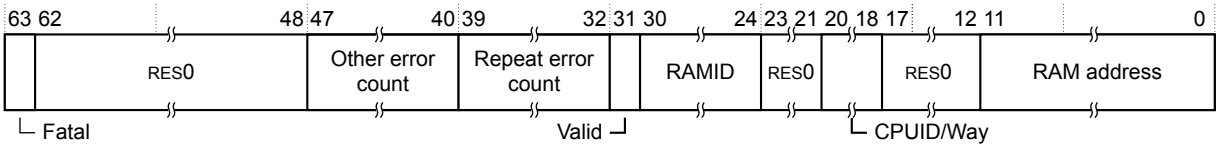

Figure B1-8 CPUMERRSR bit assignments

Fatal, [63]

Fatal bit. This bit is set to 1 on the first memory error that caused a data abort. It is a sticky bit so that after it is set, it remains set until the register is written.

The reset value is 0.

[62:48]

Reserved, RES0.

Other error count, [47:40]

This field is set to 0 on the first memory error and is incremented on any memory error that does not match the RAMID and Bank/Way information in this register while the sticky Valid bit is set.

The reset value is 0.

Repeat error count, [39:32]

This field is set to 0 on the first memory error and is incremented on any memory error that exactly matches the RAMID and Bank/Way information in this register while the sticky Valid bit is set.

The reset value is 0.

Valid, [31]

Valid bit. This bit is set to 1 on the first memory error. It is a sticky bit so that after it is set, it remains set until the register is written.

The reset value is 0.

RAMID, [30:24]

RAM Identifier. Indicates the RAM in which the first memory error. The possible values are:

`0x00` L1 Instruction tag RAM.

`0x01` L1 Instruction data RAM.

`0x08` L1 Data tag RAM.

`0x09` L1 Data data RAM.

`0x0A` L1 Data dirty RAM.

`0x18` TLB RAM.

[23:21]

Reserved, RES0.

CPUID/Way, [20:18]

Indicates the RAM where the first memory error occurred.

L1 I-tag RAM

`0x0`	Way 0
`0x1`	Way 1
`0x2-0x7`	Unused

L1 I-data RAM

`0x0`	Bank 0
`0x1`	Bank 1
`0x2-0x7`	Unused

TLB RAM

`0x0`	Way 0
`0x1`	Way 1
`0x2-0x7`	Unused

L1 D-dirty RAM

`0x0`	Dirty RAM
`0x1-0x7`	Unused

L1 D-tag RAM

`0x0`	Way 0
`0x1`	Way 1
`0x2`	Way 2
`0x3`	Way 3
`0x4-0x7`	Unused

L1 D-data RAM

`0x0`	Way0-Bank0
`0x1`	Way0-Bank1

0x2	Way1-Bank0
0x3	Way1-Bank1
...	
0x7	Way3-Bank1

[17:12]

Reserved, RES0.

RAM address, [11:0]

Indicates the index address of the first memory error.

- A fatal error results in the RAMID, Way, and RAM address recording the fatal error, even if the sticky bit is set.
- Only L1 Data data and L1 Data dirty RAMs can signal fatal errors, because all other RAM instances are protected only by parity.
- If two or more memory errors in the same RAM occur in the same cycle, only one error is reported.
- If two or more first memory error events from different RAMs occur in the same cycle, one of the errors is selected arbitrarily.
- If two or more memory error events from different RAMs, that do not match the RAMID, Way, and index information in this register while the sticky Valid bit is set, occur in the same cycle, then the Other error count field is incremented only by one.

To access the CPUMERRSR:

```
MRRC p15, 2, <Rt>, <Rt2>, c15;  Read CPUMERRSR into Rt and Rt2
MCRR p15, 2, <Rt>, <Rt2>, c15;  Write Rt and Rt2 to CPUMERRSR
```

Register access is encoded as follows:

Table B1-38 CPUMERRSR access encoding

coproc	opc1	CRm
1111	0010	1111

B1.45 Cache Size Selection Register

The CSSELR characteristics are:

Purpose

Selects the current CCSIDR, see *B1.39 Cache Size ID Register* on page B1-195, by specifying:

- The required cache level.
- The cache type, either instruction or data cache.

Usage constraints

This register is accessible as follows:

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

If the CSSELR level field is programmed to a cache level that is not implemented, then a read of CSSELR returns an UNKNOWN value in CSSELR.Level.

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Attributes

CSSELR is a 32-bit register.

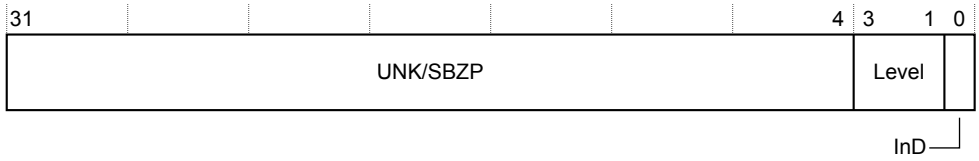

Figure B1-9 CSSELR bit assignments

[31:4]

Reserved, RES0.

Level, [3:1]

Cache level of required cache:

`0b000`	L1.
`0b001`	L2.
`0b010-0b111`	Reserved.

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

InD, [0]

Instruction not Data bit:

`0`	Data or unified cache.
`1`	Instruction cache.

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

To access the CSSELR:

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

Register access is encoded as follows:

Table B1-39 CSSELR access encoding

coproc	opc1	CRn	CRm	opc2
1111	010	0000	0001	000

B1.46 Cache Type Register

The CTR characteristics are:

Purpose

Provides information about the architecture of the caches.

Usage constraints

This register is accessible as follows:

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

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

CTR is a 32-bit register.

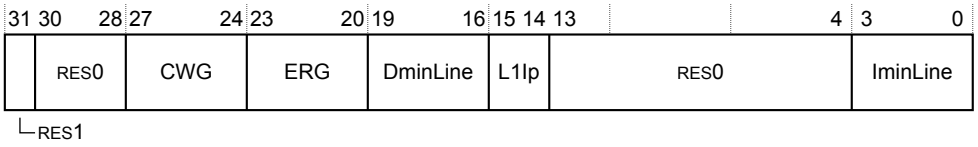

Figure B1-10 CTR bit assignments

[31]

Reserved, RES1.

[30:28]

Reserved, RES0.

CWG, [27:24]

Cache Write-Back granule. Log_2 of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified:

`0x4` Cache Write-Back granule size is 16 words.

ERG, [23:20]

Exclusives Reservation Granule. Log_2 of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions:

`0x4` Exclusive reservation granule size is 16 words.

DminLine, [19:16]

Log_2 of the number of words in the smallest cache line of all the data and unified caches that the processor controls:

`0x4` Smallest data cache line size is 16 words.

L1Ip, [15:14]

L1 Instruction cache policy. Indicates the indexing and tagging policy for the L1 Instruction cache:

`0b10` *Virtually Indexed Physically Tagged* (VIPT).

[13:4]

Reserved, RES0.

IminLine, [3:0]

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
```

Register access is encoded as follows:

Table B1-40 CTR access encoding

coproc	opc1	CRn	CRm	opc2
1111	010	0000	0000	001

B1.47 Domain Access Control Register

The DACR characteristics are:

Purpose

Defines the access permission for each of the sixteen memory domains.

Usage constraints

This register is accessible as follows:

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

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Write access to DACR(S) is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

DACR has no function when TTBCR.EAE is set to 1 to select the Long-descriptor translation table format.

Attributes

DACR is a 32-bit register.

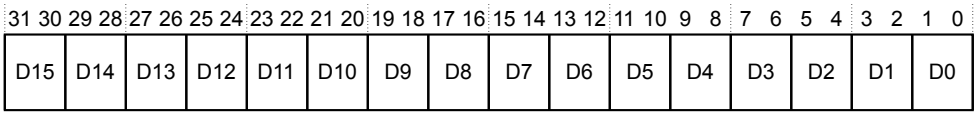

Figure B1-11 DACR bit assignments

D<*n*>, bits [2*n*+1:2*n*], for *n* = 0 to 15, [31:0]

Domain *n* access permission, where *n* = 0 to 15. Permitted values are:

`0b00` No access. Any access to the domain generates a Domain fault.

`0b01` Client. Accesses are checked against the permission bits in the translation tables.

`0b11` Manager. Accesses are not checked against the permission bits in the translation tables.

The value `0b10` is reserved.

To access the DACR:

```
MRC p15, 0, <Rt>, c3, c0, 0 ; Read DACR into Rt
MCR p15, 0, <Rt>, c3, c0, 0 ; Write Rt to DACR
```

Register access is encoded as follows:

Table B1-41 DACR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0011	0000	000

B1.48 Data Fault Address Register

The DFAR characteristics are:

Purpose

Holds the virtual address of the faulting address that caused a synchronous Data Abort exception.

Usage constraints

This register is accessible as follows:

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

Configurations

DFAR (S) is architecturally mapped to HDFAR. See *B1.64 Hyp Data Fault Address Register on page B1-245*.

Attributes

DFAR is a 32-bit register.

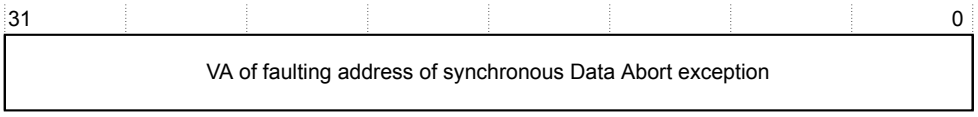

Figure B1-12 DFAR bit assignments

VA, [31:0]

The Virtual Address of faulting address of synchronous Data Abort exception.

To access the DFAR:

```
MRC p15, 0, <Rt>, c6, c0, 0 ; Read DFAR into Rt
MCR p15, 0, <Rt>, c6, c0, 0 ; Write Rt to DFAR
```

Register access is encoded as follows:

Table B1-42 DFAR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0110	0000	000

B1.49 Data Fault Status Register

The DFSR characteristics are:

Purpose

Holds status information about the last data fault.

Usage constraints

This register is accessible as follows:

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

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

There are two formats for this register. The current translation table format determines which format of the register is used.

Attributes

DFSR is a 32-bit register.

B1.50 DFSR with Short-descriptor translation table format

DFSR has a specific format when using the Short-descriptor translation table format.

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

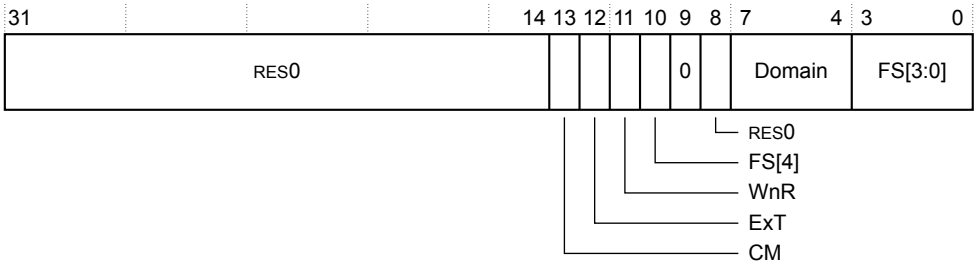

Figure B1-13 DFSR bit assignments for Short-descriptor translation table format

[31:14]
Reserved, RES0.

CM, [13]

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.

ExT, [12]

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.

WnR, [11]

Write not Read bit. This field indicates whether the abort was caused by a write or a read access:

0 Abort caused by a read access.

1 Abort caused by a write access.

For faults on CP15 cache maintenance operations, including the VA to PA translation operations, this bit always returns a value of 1.

FS[4], [10]
Part of the Fault Status field. See bits [3:0] in this table.

[9]
RAZ.

[8]
Reserved, RES0.

Domain, [7:4]

Specifies which of the 16 domains, D15-D0, was being accessed when a data fault occurred.

For permission faults that generate Data Abort exception, this field is UNKNOWN. Armv8 deprecates any use of the domain field in the DFSR.

FS[3:0], [3:0]

Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved:

`0b00001` Alignment fault.

`0b00010` Debug event.

`0b00011` Access flag fault, section.

`0b00100` Instruction cache maintenance fault.

`0b00101` Translation fault, section.

`0b00110` Access flag fault, page.

`0b00111` Translation fault, page.

`0b01000` Synchronous external abort, non-translation.

`0b01001` Domain fault, section.

`0b01011` Domain fault, page.

`0b01100` Synchronous external abort on translation table walk, first level.

`0b01101` Permission fault, section.

`0b01110` Synchronous external abort on translation table walk, second level.

`0b01111` Permission fault, second level.

`0b10000` TLB conflict abort.

`0b10101` `LDREX` or `STREX` abort.

`0b10110` Asynchronous external abort.

`0b11000` Asynchronous parity error on memory access.

`0b11001` Synchronous parity error on memory access.

`0b11100` Synchronous parity error on translation table walk, first level.

`0b11110` Synchronous parity error on translation table walk, second level.

B1.51 DFSR with Long-descriptor translation table format

DFSR has a specific format when using the Long-descriptor translation table format.

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

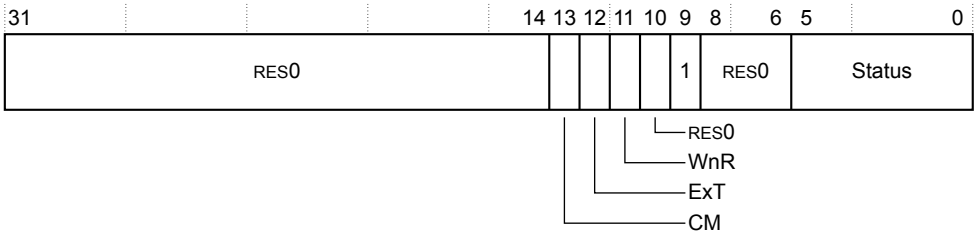

Figure B1-14 DFSR bit assignments for Long-descriptor translation table format

[31:14]

Reserved, RES0.

CM, [13]

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.

ExT, [12]

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.

WnR, [11]

Write not Read bit. This field indicates whether the abort was caused by a write or a read access:

`0` Abort caused by a read access.

`1` Abort caused by a write access.

For faults on CP15 cache maintenance operations, including the VA to PA translation operations, this bit always returns a value of 1.

[10]

Reserved, RES0.

[9]

RAO.

[8:6]

Reserved, RES0.

Status, [5:0]

Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved.

`0b000000` Address size fault in TTBR0 or TTBR1.

`0b0001LL` Translation fault, LL bits indicate level.

`0b0010LL` Access fault flag, LL bits indicate level.

`0b0011LL` Permission fault, LL bits indicate level.

`0b010000` Synchronous external abort.

`0b010001` Asynchronous external abort.

`0b0101LL` Synchronous external abort on translation table walk, LL bits indicate level.

`0b011000` Synchronous parity error on memory access.

`0b011001` Asynchronous parity error on memory access (DFSR only).

`0b0111LL` Synchronous parity error on memory access on translation table walk, first level, LL bits indicate level.

`0b100001` Alignment fault.

`0b100010` Debug event.

`0b110000` TLB conflict abort.

`0b110101` `LDREX` or `STREX` abort.

Table B1-43 Encodings of LL bits associated with the MMU fault

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

To access the DFSR:

```
MRC p15, 0, <Rt>, c5, c0, 0; Read DFSR into Rt
MCR p15, 0, <Rt>, c5, c0, 0; Write Rt to DFSR
```

B1.52 Encoding of ISS[24:20] when HSR[31:30] is 0b00

For EC values that are nonzero and have the two most-significant bits 0b00, ISS[24:20] provides the condition field for the trapped instruction, together with a valid flag for this field.

The encoding of this part of the ISS field is:

CV, ISS[24]

Condition valid. Possible values of this bit are:

`0` The COND field is not valid.

`1` The COND field is valid.

When an instruction is trapped, CV is set to 1.

COND, ISS[23:20]

The Condition field for the trapped instruction. This field is valid only when CV is set to 1.

If CV is set to 0, this field is RES0.

When an instruction is trapped, the COND field is set to the condition the instruction was executed with.

B1.53 FCSE Process ID Register

FCSEIDR

The processor does not implement *Fast Context Switch Extension* (FCSE). This register is always RES0.

B1.54 Hyp Auxiliary Configuration Register

HACR

The processor does not implement HACR. This register is always RES0.

B1.55 Hyp Auxiliary Control Register

The HACTLR characteristics are:

Purpose

Controls write access to IMPLEMENTATION DEFINED registers in Non-secure EL1 modes, such as CPUACTLR, CPUECTLR, L2CTLR, L2ECTLR, and L2ACTLR.

Usage constraints

This register is accessible as follows:

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

Configurations

There are no configuration notes.

Attributes

HACTLR is a 32-bit register.

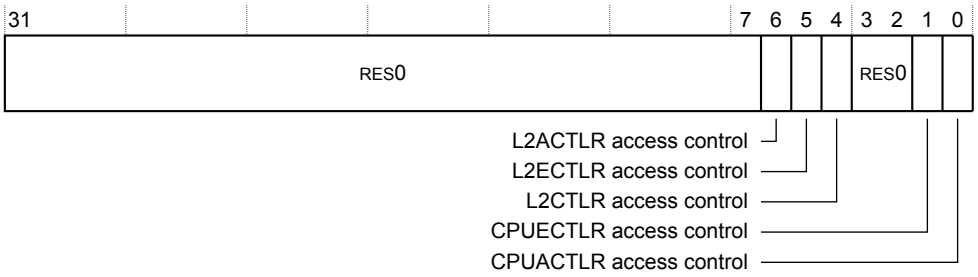

Figure B1-15 HACTLR bit assignments

[31:7]

Reserved, RES0.

L2ACTLR access control, [6]

L2ACTLR write access control. The possible values are:

`0` The register is not write accessible from Non-secure EL1.

This is the reset value.

`1` The register is write accessible from Non-secure EL1.

Write access from Non-secure EL1 also requires ACTLR(S)[6] to be set.

L2ECTLR access control, [5]

L2ECTLR write access control. The possible values are:

`0` The register is not write accessible from Non-secure EL1.

This is the reset value.

`1` The register is write accessible from Non-secure EL1.

Write access from Non-secure EL1 also requires ACTLR(S)[5] to be set.

L2CTLR access control, [4]

L2CTLR write access control. The possible values are:

0 The register is not write accessible from Non-secure EL1.

This is the reset value.

1 The register is write accessible from Non-secure EL1.

Write access from Non-secure EL1 also requires ACTLR(S)[4] to be set.

[3:2]

Reserved, RES0.

CPUECTLR access control, [1]

CPUECTLR write access control. The possible values are:

0 The register is not write accessible from Non-secure EL1.

This is the reset value.

1 The register is write accessible from Non-secure EL1.

Write access from Non-secure EL1 also requires ACTLR(S)[1] to be set.

CPUACTLR access control, [0]

CPUACTLR write access control. The possible values are:

0 The register is not write accessible from Non-secure EL1.

This is the reset value.

1 The register is write accessible from Non-secure EL1.

Write access from Non-secure EL1 also requires ACTLR(S)[0] to be set.

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
```

Register access is encoded as follows:

Table B1-44 HACTLR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0001	0000	001

B1.56 Hyp Auxiliary Data Fault Status Syndrome Register

HADFSR

The processor does not implement HADFSR. This register is always RES0.

B1.57 Hyp Auxiliary Instruction Fault Status Syndrome Register

HAIFSR

The processor does not implement HAIFSR. This register is always RES0.

B1.58 Hyp Auxiliary Memory Attribute Indirection Register 0

HAMAIR0

The processor does not implement HAMAIR0. This register is always RES0.

B1.59 Hyp Auxiliary Memory Attribute Indirection Register 1

HAMAIR1

The processor does not implement HAMAIR1. This register is always RES0.

B1.60 Hyp Architectural Feature Trap Register

The HCPTR characteristics are:

Purpose

Controls trapping to Hyp mode of Non-secure access, at EL1 or lower, to coprocessors other than CP14 and CP15 and to floating-point and Advanced SIMD functionality. Also controls access from Hyp mode to this functionality.

Usage constraints

This register is accessible as follows:

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

If a bit in the NSACR prohibits a Non-secure access, then the corresponding bit in the HCPTR behaves as RAO/WI for Non-secure accesses. See the bit description for TASE.

Configurations

There are no configuration notes.

Attributes

HCPTR is a 32-bit register.

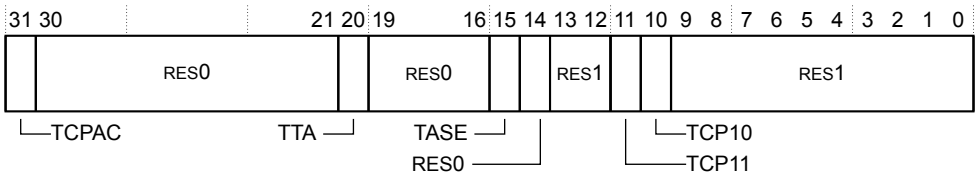

Figure B1-16 HCPTR bit assignments

TCPAC, [31]

Trap CPACR accesses. The possible values of this bit are:

`0` Has no effect on CPACR accesses.

`1` Trap valid Non-secure EL1 CPACR accesses to Hyp mode.

When this bit is set to 1, any valid Non-secure EL1 access to the CPACR is trapped to Hyp mode.

Resets to 0.

[30:21]

Reserved, RES0.

TTA, [20]

Trap Trace Access.

Not implemented. RES0.

[19:16]

Reserved, RES0.

TASE, [15]

Trap Advanced SIMD use:

0 If the NSACR settings permit Non-secure use of the Advanced SIMD functionality then Hyp mode can access that functionality, regardless of any settings in the CPACR. This bit value has no effect on possible use of the Advanced SIMD functionality from Non-secure EL1 and EL0 modes.

1 Trap valid Non-secure accesses to Advanced SIMD functionality to Hyp mode.

If Advanced SIMD and floating-point are not implemented, this bit is RAO/WI.

If NSACR.NSASEDIS is set to 1, then on Non-secure accesses to the HCPTR, the TASE bit behaves as RAO/WI.

[14]

Reserved, RES0.

[13:12]

Reserved, RES1.

TCP11, [11]

Trap CP11. The possible values of each of this bit is:

0 If NSACR.cp11 is set to 1, then Hyp mode can access CP11, regardless of the value of CPACR.cp11. This bit value has no effect on possible use of CP11 from Non-secure EL1 and EL0 modes.

1 Trap valid Non-secure accesses to CP11 to Hyp mode.

Any otherwise-valid access to CP11 from:
- A Non-secure EL1 or EL0 state is trapped to Hyp mode.
- Hyp mode generates an Undefined Instruction exception, taken in Hyp mode.

Resets to 0.

If the TCP11 and TCP10 fields are set to different values, the behavior is the same as if both fields were set to the value of TCP10, in all respects other than the value read back by explicitly reading TCP11.

TCP10, [10]

Trap CP10. The possible values of each of this bit is:

0 If NSACR.cp10 is set to 1, then Hyp mode can access CP10, regardless of the value of CPACR.cp10. This bit value has no effect on possible use of CP10 from Non-secure EL1 and EL0 modes.

1 Trap valid Non-secure accesses to CP10 to Hyp mode.

Any otherwise-valid access to CP10 from:
- A Non-secure EL1 or EL0 state is trapped to Hyp mode.
- Hyp mode generates an Undefined Instruction exception, taken in Hyp mode.

Resets to 0.

If the TCP11 and TCP10 fields are set to different values, the behavior is the same as if both fields were set to the value of TCP10, in all respects other than the value read back by explicitly reading TCP11.

[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
```

Register access is encoded as follows:

Table B1-45 HCPTR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0001	0001	010

B1.61 Hyp Configuration Register

The HCR characteristics are:

Purpose

Provides configuration controls for virtualization, including defining whether various Non-secure operations are trapped to Hyp mode.

Usage constraints

This register is accessible as follows:

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

Configurations

There are no configuration notes.

Attributes

HCR is a 32-bit register.

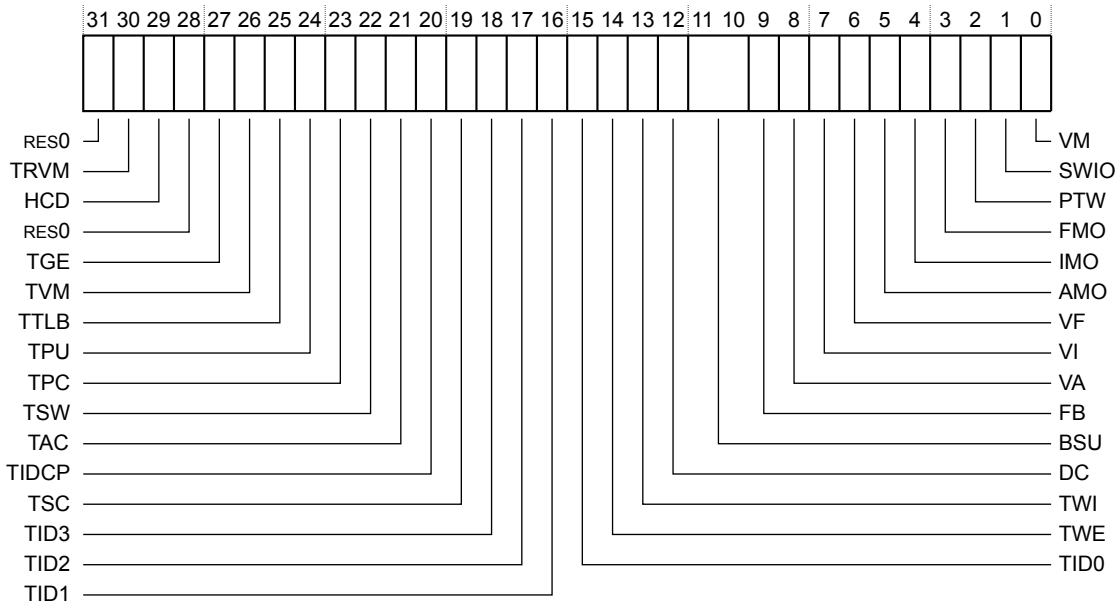

Figure B1-17 HCR bit assignments

[31]

Reserved, RES0.

TRVM, [30]

Trap Read of Virtual Memory controls.

When 1, this causes Reads to the EL1 virtual memory control registers from EL1 to be trapped to EL2. This covers the following registers:

SCTLR, TTBR0, TTBR1, TTBCR, DACR, DFSR, IFSR, DFAR, IFAR, ADFSR, AIFSR, PRRR/MAIR0, NMRR/MAIR1, AMAIR0, AMAIR1, and CONTEXTIDR.

The reset value is 0.

HCD, [29]

Hyp Call Disable. The HCD value is:

0 HVC is enabled at EL1 or EL2.

1 HVC is UNDEFINED at all exception levels.

[28]

Reserved, RES0.

TGE, [27]

Trap General Exceptions. If this bit is set, and SCR_EL3.NS is set, then:

All exceptions that would be routed to EL1 are routed to EL2.

- The SCTLR.M bit is treated as 0 regardless of its actual state, other than for the purpose of reading the bit.
- The HCR.FMO, IMO, and AMO bits are treated as 1 regardless of their actual state, other than for the purpose of reading the bits.
- All virtual interrupts are disabled.
- Any implementation defined mechanisms for signaling virtual interrupts are disabled.
- An exception return to EL1 is treated as an illegal exception return.

Additionally, if HCR.TGE is 1, the HDCR.{TDRA,TDOSA,TDA} bits are ignored and the processor behaves as if they are set to 1, other than for the value read back from HDCR.

The reset value is 0.

TVM, [26]

Trap Virtual Memory controls. When 1, this causes Writes to the EL1 virtual memory control registers from EL1 to be trapped to EL2. This covers the following registers:

SCTLR, TTBR0, TTBR1, TTBCR, DACR, DFSR, IFSR, DFAR, IFAR, ADFSR, AIFSR, PRRR/MAIR0, NMRR/MAIR1, AMAIR0, AMAIR1, and CONTEXTIDR.

The reset value is 0.

TTLB, [25]

Trap TLB maintenance instructions. When 1, this causes TLB maintenance instructions executed from EL1 that are not UNDEFINED to be trapped to EL2. This covers the following instructions:

TLBIALLIS, TLBIMVAIS, TLBIASIDIS, TLBIMVAAIS, TLBIALL, TLBIMVA, TLBIASID, TLBIMVAA, TLBIMVALIS, TLBIMVAALIS, TLBIMVAL, and TLBIMVAAL.

The reset value is 0.

TPU, [24]

Trap Cache maintenance instructions to Point of Unification. When 1, this causes Cache maintenance instructions to the point of unification executed from EL1 or EL0 that are not UNDEFINED to be trapped to EL2. This covers the following instructions:

ICIMVAU, ICIALLU, ICIALLUIS, and DCCMVAU.

The reset value is 0.

TPC, [23]

Trap Data/Unified Cache maintenance operations to Point of Coherency. When 1, this causes Data or Unified Cache maintenance instructions by address to the point of coherency executed from EL1 or EL0 that are not UNDEFINED to be trapped to EL2. This covers the following instructions:

DCIMVAC, DCCIMVAC, and DCCMVAC.

The reset value is 0.

TSW, [22]

Trap Data/Unified Cache maintenance operations by Set/Way. When 1, this causes Data or Unified Cache maintenance instructions by set/way executed from EL1 that are not UNDEFINED to be trapped to EL2. This covers the following instructions:

`DCISW`, `DCCSW`, and `DCCISW`.

The reset value is 0.

TAC, [21]

Trap ACTLR accesses. When this bit is set to 1, any valid Non-secure access to the ACTLR is trapped to Hyp mode.

The reset value is 0.

TIDCP, [20]

Trap Implementation Dependent functionality. When 1, this causes accesses to all CP15 MCR and MRC instructions executed from EL1, to be trapped to EL2 as follows:

- CRn is 9, Opcode1 is 0 to 7, CRm is c0, c1, c2, c5, c6, c7, c8, opcode2 is 0 to 7.
- CRn is 10, Opcode1 is 0 to 7, CRm is c0, c1, c4, c8}, opcode2 is 0 to 7.
- CRn is 11, Opcode1 is 0 to 7, CRm is c0 to c8, or c15, opcode2 is 0 to 7.

Accesses from EL0 are UNDEFINED.

Resets to 0.

TSC, [19]

Trap SMC instruction. When this bit is set to 1, any attempt from a Non-secure EL1 state to execute an SMC instruction, that passes its condition check if it is conditional, is trapped to Hyp mode.

The reset value is 0.

TID3, [18]

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:

- CP15, OPC1 is 0, CRn is 0, CRm is c3, c4, c5, c6, or c7, and Opc2 is 0 or 1.
- CP15, Opc1 is 0, CRn is 0, CRm is c3, and Opc2 is 2.
- CP15, Opc1 is 0, CRn is 0, CRm is 5, and Opc2 is 4 or 5.

The reset value is 0.

TID2, [17]

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.

TID1, [16]

Trap ID Group 1. When 1, this causes reads to the following registers executed from EL1 to be trapped to EL2:

TCMTR, TLBTR, AIDR, and REVIDR.

The reset value is 0.

TID0, [15]

Trap ID Group 0. When 1, this causes reads to the following registers executed from EL1 or EL0 if not UNDEFINED to be trapped to EL2:

FPSID and JIDR.

The reset value is 0.

TWE, [14]

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.

TWI, [13]

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.

DC, [12]

Default cacheable. When this bit is set to 1, and the Non-secure EL1 and EL0 stage 1 MMU is disabled, the memory type and attributes determined by the stage 1 translation is Normal, Non-shareable, Inner Write-Back Write-Allocate, Outer Write-Back Write-Allocate.

The reset value is 0.

BSU, [11:10]

Barrier Shareability upgrade. The value in this field determines the minimum shareability domain that is applied to any barrier executed from EL1 or EL0. The possible values are:

`0b00` No effect.

`0b01` Inner Shareable.

`0b10` Outer Shareable.

`0b11` Full System.

The reset value is 0.

FB, [9]

Force broadcast. When 1, this causes the following instructions to be broadcast within the Inner Shareable domain when executed from Non-secure EL1:

`TLBIALL`, `TLBIMVA`, `TLBIASID`, `TLBIMVAA`, `BPIALL`, and `ICIALLU`.

The reset value is 0.

VA, [8]

Virtual Asynchronous Abort exception. When the AMO bit is set to 1, setting this bit signals a virtual Asynchronous Abort exception to the Guest OS, when the processor is executing in Non-secure state at EL0 or EL1.

The Guest OS cannot distinguish the virtual exception from the corresponding physical exception.

The reset value is 0.

VI, [7]

Virtual IRQ exception. When the IMO bit is set to 1, setting this bit signals a virtual IRQ exception to the Guest OS, when the processor is executing in Non-secure state at EL0 or EL1.

The Guest OS cannot distinguish the virtual exception from the corresponding physical exception.

The reset value is 0.

VF, [6]

Virtual FIQ exception. When the FMO bit is set to 1, setting this bit signals a virtual FIQ exception to the Guest OS, when the processor is executing in Non-secure state at EL0 or EL1.

The Guest OS cannot distinguish the virtual exception from the corresponding physical exception.

The reset value is 0.

AMO, [5]

Asynchronous Abort Mask Override. When this is set to 1, it overrides the effect of CPSR.A, and enables virtual exception signaling by the VA bit.

The reset value is 0.

IMO, [4]

IRQ Mask Override. When this is set to 1, it overrides the effect of CPSR.I, and enables virtual exception signaling by the VI bit.

The reset value is 0.

FMO, [3]

FIQ Mask Override. When this is set to 1, it overrides the effect of CPSR.F, and enables virtual exception signaling by the VF bit.

The reset value is 0.

PTW, [2]

Protected Table Walk. When 1, if the stage 2 translation of a translation table access made as part of a stage 1 translation table walk at EL0 or EL1 maps that translation table access to Device memory, the access is faulted as a stage 2 Permission fault.

The reset value is 0.

SWIO, [1]

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.

VM, [0]

Second stage of Translation enable. When 1, this enables the second stage of translation for execution in EL1 and EL0.

The reset value is 0.

To access the HCR:

```
MRC p15, 4, <Rt>, c1, c1, 0; Read Hyp Configuration Register
MCR p15, 4, <Rt>, c1, c1, 0; Write Hyp Configuration Register
```

Register access is encoded as follows:

Table B1-46 HCR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0001	0001	000

B1.62 Hyp Configuration Register 2

The HCR2 characteristics are:

Purpose

Provides additional configuration controls for virtualization.

Usage constraints

This register is accessible as follows:

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

Configurations

This register is accessible only at EL2 or EL3.

Attributes

HCR2 is a 32-bit register.

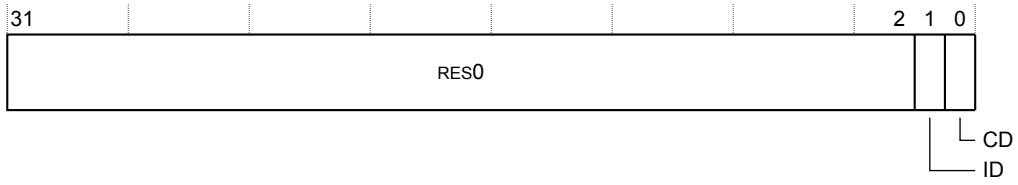

Figure B1-18 HCR2 bit assignments

[31:2]

Reserved, RES0.

ID, [1]

Stage 2 Instruction cache disable. When HCR.VM is 1, this forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable for the EL1/EL0 translation regime. The possible values are:

`0` No effect on the stage 2 of the EL1/EL0 translation regime for instruction accesses.

`1` Forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable for the EL0/EL1 translation regime.

CD, [0]

Stage 2 Data cache disable. When HCR.VM is 1, this forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable for the EL1/EL0 translation regime. The possible values are:

`0` No effect on the stage 2 of the EL1/EL0 translation regime for data accesses and translation table walks.

`1` Forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable for the EL0/EL1 translation regime.

To access the HCR2:

```
MRC p15,4,<Rt>,c1,c1,4 ; Read HCR2 into Rt
MCR p15,4,<Rt>,c1,c1,4 ; Write Rt to HCR2
```

Register access is encoded as follows:

Table B1-47 HCR2 access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0001	0001	100

B1.63 Hyp Debug Control Register

The HDCR characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

Configurations

This register is accessible only at EL2 or EL3.

Attributes

HDCR is a 32-bit register.

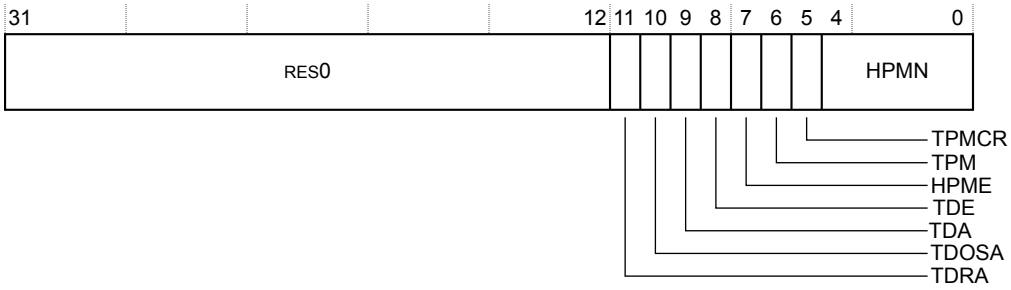

Figure B1-19 HDCR bit assignments

[31:12]

Reserved, RES0.

TDRA, [11]

Trap debug ROM address register access.

`0` Has no effect on accesses to debug ROM address registers from EL1 and EL0.

`1` Trap valid Non-secure EL1 and EL0 access to debug ROM address registers to Hyp mode.

When this bit is set to 1, any valid Non-secure access to the following registers is trapped to Hyp mode:

- DBGDRAR.
- DBGDSAR.

If HCR.TGE is 1 or HDCR.TDE is 1, then this bit is ignored and treated as though it is 1 other than for the value read back from HDCR.

On Warm reset, the field resets to 0.

TDOSA, [10]

Trap Debug OS-related register access:

`0` Has no effect on accesses to CP14 Debug registers.

1 Trap valid Non-secure accesses to CP14 OS-related Debug registers to Hyp mode.

When this bit is set to 1, any valid Non-secure CP14 access to the following OS-related Debug registers is trapped to Hyp mode:
- DBGOSLSR.
- DBGOSLAR.
- DBGOSDLR.
- DBGPRCR.

If HCR.TGE is 1 or HDCR.TDE is 1, then this bit is ignored and treated as though it is 1 other than for the value read back from HDCR.

On Warm reset, the field resets to 0.

TDA, [9]

Trap Debug Access:

0 Has no effect on accesses to CP14 Debug registers.

1 Trap valid Non-secure accesses to CP14 Debug registers to Hyp mode.

When this bit is set to 1, any valid access to the CP14 Debug registers, other than the registers trapped by the TDRA and TDOSA bits, is trapped to Hyp mode.

If HCR.TGE is 1 or HDCR.TDE is1, then this bit is ignored and treated as though it is 1 other than for the value read back from HDCR.

On Warm reset, the field resets to 0.

TDE, [8]

Trap Debug Exceptions:

0 Has no effect on Debug exceptions.

1 Route Non-secure Debug exceptions to Hyp mode.

When this bit is set to 1, any Debug exception taken in Non-secure state is trapped to Hyp mode.

If HCR.TGE is 1, then this bit is ignored and treated as though it is 1 other than for the value read back from HDCR.This bit resets to 0.

HPME, [7]

Hypervisor Performance Monitor Enable:

0 Hyp mode performance monitor counters disabled.

1 Hyp mode performance monitor counters enabled.

When this bit is set to 1, access to the performance monitors that are reserved for use from Hyp mode is enabled. For more information, see the description of the HPMN field.

The reset value of this bit is UNKNOWN.

TPM, [6]

Trap Performance Monitor accesses:

0 Has no effect on performance monitor accesses.

1 Trap valid Non-secure performance monitor accesses to Hyp mode.

When this bit is set to 1, any valid Non-secure access to the Performance Monitor registers is trapped to Hyp mode. This bit resets to 0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

TPMCR, [5]

Trap Performance Monitor Control Register accesses:

`0` Has no effect on PMCR accesses.

`1` Trap valid Non-secure PMCR accesses to Hyp mode.

When this bit is set to 1, any valid Non-secure access to the PMCR is trapped to Hyp mode. This bit resets to 0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

HPMN, [4:0]

Hyp Performance Monitor count. Defines the number of Performance Monitors counters that are accessible from Non-secure EL1 and EL0 modes if unprivileged access is enabled.

In Non-secure state, HPMN divides the Performance Monitors counters as follows. If software is accessing Performance Monitors counter n then, in Non-secure state:

For example, If PMnEVCNTR is performance monitor counter *n* then, in Non-secure state:

- If *n* is in the range $0 \le n <$ 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. The HPME bit enables access to the counters in this range.

If this field is set to 0, or to a value larger than PMCR.N, then the behavior in Non-secure EL0 and EL1 is CONSTRAINED UNPREDICTABLE, and one of the following must happen:

- The number of counters accessible is an UNKNOWN non-zero value less than PMCR.N.
- There is no access to any counters.

For reads of HDCR.HPMN by EL2 or higher, if this field is set to 0 or to a value larger than PMCR.N, the processor must return a CONSTRAINED UNPREDICTABLE value being one of:

- PMCR.N.
- The value that was written to HDCR.HPMN.
- (The value that was written to HDCR.HPMN) modulo 2h, where h is the smallest number of bits required for a value in the range 0 to PMCR.N.

This field resets to `0x6`.

To access the HDCR:

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

Register access is encoded as follows:

Table B1-48 HDCR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0001	0001	001

B1.64 Hyp Data Fault Address Register

The HDFAR characteristics are:

Purpose

Holds the virtual address of the faulting address that caused a synchronous Data Abort exception that is taken to Hyp mode.

Usage constraints

This register is accessible as follows:

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

An execution in a Non-secure EL1 state, or in Secure state, makes the HDFAR UNKNOWN.

Configurations

HDFAR (S) is architecturally mapped to DFAR (S). See *B1.48 Data Fault Address Register* on page B1-216.

Attributes

HDFAR is a 32-bit register.

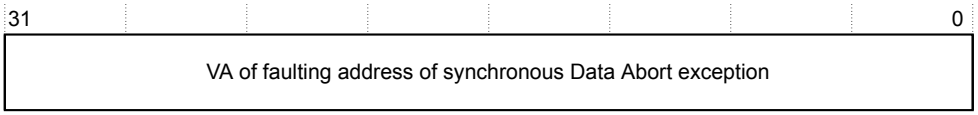

Figure B1-20 HDFAR bit assignments

VA, [31:0]

The Virtual Address of faulting address of synchronous Data Abort exception.

To access the HDFAR:

```
MRC p15, 4, <Rt>, c6, c0, 0 ; Read HDFAR into Rt
MCR p15, 4, <Rt>, c6, c0, 0 ; Write Rt to HDFAR
```

Register access is encoded as follows:

Table B1-49 HDFAR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0110	0000	000

B1.65 Hyp Instruction Fault Address Register

The HIFAR characteristics are:

Purpose

Holds the virtual address of the faulting address that caused a synchronous Prefetch Abort exception that is taken to Hyp mode.

Usage constraints

This register is accessible as follows:

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

Execution in any Non-secure mode other than Hyp mode makes HPFAR UNKNOWN.

Configurations

HIFAR is architecturally mapped to IFAR (S). See *B1.86 Instruction Fault Address Register* on page B1-287.

Attributes

HIFAR is a 32-bit register.

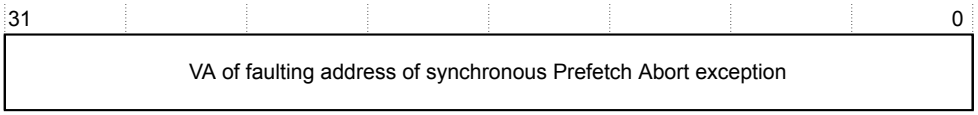

Figure B1-21 HIFAR bit assignments

VA, [31:0]

The Virtual Address of faulting address of synchronous Prefetch Abort exception.

To access the HIFAR:

```
MRC p15, 4, <Rt>, c6, c0, 2 ; Read HIFAR into Rt
MCR p15, 4, <Rt>, c6, c0, 2 ; Write Rt to HIFAR
```

Register access is encoded as follows:

Table B1-50 HIFAR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0110	0000	010

B1.66 Hyp IPA Fault Address Register

The HPFAR characteristics are:

Purpose

Holds the faulting IPA for some aborts on a stage 2 translation taken to Hyp mode.

Usage constraints

This register is accessible as follows:

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

Execution in any Non-secure mode other than Hyp mode makes HPFAR UNKNOWN.

Configurations

There are no configuration notes.

Attributes

HPFAR is a 32-bit register.

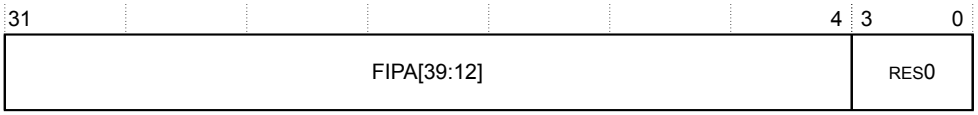

Figure B1-22 HPFAR bit assignments

FIPA[39:12], [31:4]

Bits [39:12] of the faulting intermediate physical address

[3:0]

Reserved, RES0

To access the HPFAR:

```
MRC p15, 4, <Rt>, c6, c0, 4 ; Read HPFAR into Rt
MCR p15, 4, <Rt>, c6, c0, 4 ; Write Rt to HPFAR
```

Register access is encoded as follows:

Table B1-51 HPFAR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0110	0000	100

B1.67 Hyp System Control Register

The HSCTLR characteristics are:

Purpose

Provides top level control of the system operation in Hyp mode. This register provides Hyp mode control of features controlled by the Banked SCTLR bits, and shows the values of the non-Banked SCTLR bits.

Usage constraints

This register is accessible as follows:

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

Configurations

There are no configuration notes.

Attributes

HSCTLR is a 32-bit register.

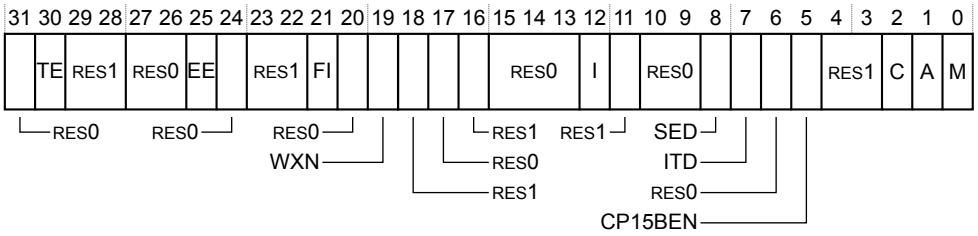

Figure B1-23 HSCTLR bit assignments

[31]

Reserved, RES0.

TE, [30]

Thumb Exception enable. This bit controls whether exceptions taken in Hyp mode are taken in A32 or T32 state:

0 Exceptions taken in A32 state.

1 Exceptions taken in T32 state.

[29:28]

Reserved, RES1.

[27:26]

Reserved, RES0.

EE, [25]

Exception Endianness. The value of this bit defines the value of the CPSR.E bit on entry to an exception vector, including reset. This value also indicates the endianness of the translation table data for translation table lookups:

0 Little endian.

1 Big endian.

[24]

Reserved, RES0.

[23:22]

Reserved, RES1.

FI, [21]

Fast Interrupts configuration enable bit. This bit can be used to reduce interrupt latency by disabling implementation-defined performance features.

This bit is not implemented, RES0.

[20]

Reserved, RES0.

WXN, [19]

Write permission implies *Execute Never* (XN). This bit can be used to require all memory regions with write permission to be treated as XN:

`0` Regions with write permission are not forced to XN.

`1` Regions with write permission are forced to XN.

The WXN bit is permitted to be cached in a TLB.

[18]

Reserved, RES1.

[17]

Reserved, RES0.

[16]

Reserved, RES1.

[15:13]

Reserved, RES0.

I, [12]

Instruction cache enable. This is an enable bit for instruction caches at EL2:

`0` Instruction caches disabled at EL2. If HSCTLR.M is set to 0, instruction accesses from stage 1 of the EL2 translation regime are to Normal memory, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable.

`1` Instruction caches enabled at EL2. If HSCTLR.M is set to 0, instruction accesses from stage 1 of the EL2 translation regime are to Normal memory, Outer Shareable, Inner Write-Through, Outer Write-Through.

When this bit is 0, all EL2 Normal memory instruction accesses are Non-cacheable.

If this register is at the highest exception level implemented, field resets to 0. Otherwise, its reset value is UNKNOWN.

[11]

Reserved, RES1.

[10:9]

Reserved, RES0.

SED, [8]

SETEND Disable:

0 The SETEND instruction is available.

1 The SETEND instruction is UNALLOCATED.

ITD, [7]

IT Disable:

0 The IT instruction functionality is available.

1 All encodings of the IT instruction with hw1[3:0]!=1000 are UNDEFINED and treated as unallocated. All encodings of the subsequent instruction with the following values for hw1 are UNDEFINED (and treated as unallocated):

`11xxxxxxxxxxxxx`	All 32-bit instructions, B(2), B(1), Undefined, SVC, Load/Store multiple
`1x11xxxxxxxxxxx`	Miscellaneous 16-bit instructions
`1x100xxxxxxxxx`	ADD Rd, PC, #imm
`01001xxxxxxxxxx`	LDR Rd, [PC, #imm]
`0100x1xxx1111xxx`	ADD(4),CMP(3), MOV, BX pc, BLX pc
`010001xx1xxxx111`	ADD(4),CMP(3), MOV

[6]

Reserved, RES0.

CP15BEN, [5]

CP15 barrier enable:

0 CP15 barrier operations disabled. Their encodings are UNDEFINED.

1 CP15 barrier operations enabled.

[4:3]

Reserved, RES1.

C, [2]

Cache enable. This is an enable bit for data and unified caches at EL2:

0 Data and unified caches disabled at EL2.

1 Data and unified caches enabled at EL2.

When this bit is 0, all EL2 Normal memory data accesses and all accesses to the EL2 translation tables are Non-cacheable.

If this register is at the highest exception level implemented, field resets to 0. Otherwise, its reset value is UNKNOWN.

A, [1]

Alignment check enable. This is the enable bit for Alignment fault checking:

0 Alignment fault checking disabled.

1 Alignment fault checking enabled.

When this bit is 1, all instructions that load or store one or more registers, other than load/store exclusive and load-acquire/store-release, have an alignment check that the address being accessed is aligned to the size of the data element(s) being accessed. If this check fails it causes an Alignment fault, that is taken as a Data Abort exception.

Load/store exclusive and load-acquire/store-release instructions have this alignment check regardless of the value of the A bit.

If this register is at the highest exception level implemented, field resets to 0. Otherwise, its reset value is UNKNOWN.

M, [0]

MMU enable. This is a global enable bit for the EL2 stage 1 MMU:

`0` EL2 stage 1 MMU disabled.

`1` EL2 stage 1 MMU enabled.

If this register is at the highest exception level implemented, field resets to 0. Otherwise, its reset value is UNKNOWN.

To access the HSCTLR:

```
MRC p15,4,<Rt>,c1,c0,0 ; Read HSCTLR into Rt
MCR p15,4,<Rt>,c1,c0,0 ; Write Rt to HSCTLR
```

Register access is encoded as follows:

Table B1-52 HSCTLR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0001	0000	000

B1.68 Hyp Syndrome Register

The HSR characteristics are:

Purpose

Holds syndrome information for an exception taken to Hyp mode.

Usage constraints

This register is accessible as follows:

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

Configurations

This register is accessible only at EL2 or EL3.

Attributes

HSR is a 32-bit register.

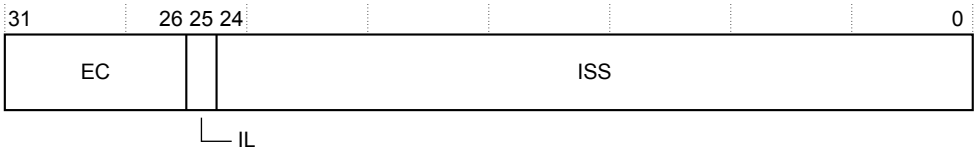

Figure B1-24 HSR bit assignments

EC, [31:26]

Exception class. The exception class for the exception that is taken in Hyp mode. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

IL, [25]

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

ISS, [24:0]

Instruction specific syndrome. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information. The interpretation of this field depends on the value of the EC field. See *B1.52 Encoding of ISS[24:20] when HSR[31:30] is 0b00* on page B1-222.

B1.69 Hyp System Trap Register

The HSTR characteristics are:

Purpose

Controls trapping to Hyp mode of Non-secure accesses, at EL1 or lower, of use of T32EE, or the CP15 primary registers, {c0-c3,c5-c13,c15}.

Usage constraints

This register is accessible as follows:

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

Configurations

This register is accessible only at EL2 or EL3.

Attributes

HSTR is a 32-bit register.

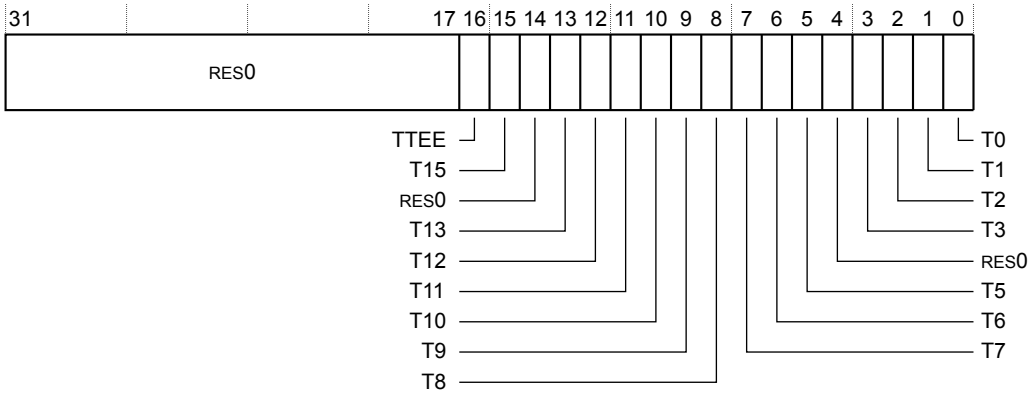

Figure B1-25 HSTR bit assignments

[31:17]

Reserved, RES0.

TTEE, [16]

Trap T32EE. This value is:

`0` T32EE is not supported.

T15, [15]

Trap coprocessor primary register CRn = 15. The possible values are:

`0` Has no effect on Non-secure accesses to CP15 registers.

`1` Trap valid Non-secure accesses to coprocessor primary register CRn = 15 to Hyp mode.

The reset value is 0.

[14]

Reserved, RES0.

T13, [13]

Trap coprocessor primary register CRn = 13. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 13 to Hyp mode.

The reset value is 0.

T12, [12]

Trap coprocessor primary register CRn = 12. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 12 to Hyp mode.

The reset value is 0.

T11, [11]

Trap coprocessor primary register CRn = 11. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 11 to Hyp mode.

The reset value is 0.

T10, [10]

Trap coprocessor primary register CRn = 10. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 10 to Hyp mode.

The reset value is 0.

T9, [9]

Trap coprocessor primary register CRn = 9. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 9 to Hyp mode.

The reset value is 0.

T8, [8]

Trap coprocessor primary register CRn = 8. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 8 to Hyp mode.

The reset value is 0.

T7, [7]

Trap coprocessor primary register CRn = 7. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 7 to Hyp mode.

The reset value is 0.

T6, [6]

Trap coprocessor primary register CRn = 6. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 6 to Hyp mode.

The reset value is 0.

T5, [5]

Trap coprocessor primary register CRn = 5. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 5 to Hyp mode.

The reset value is 0.

[4]

Reserved, RES0.

T3, [3]

Trap coprocessor primary register CRn = 3. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 3 to Hyp mode.

The reset value is 0.

T2, [2]

Trap coprocessor primary register CRn = 2. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 2 to Hyp mode.

The reset value is 0.

T1, [1]

Trap coprocessor primary register CRn = 1. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 1 to Hyp mode.

The reset value is 0.

T0, [0]

Trap coprocessor primary register CRn = 0. The possible values are:

0 Has no effect on Non-secure accesses to CP15 registers.

1 Trap valid Non-secure accesses to coprocessor primary register CRn = 0 to Hyp mode.

The reset value is 0.

To access the HSTR:

```
MRC p15, 4, <Rt>, c1, c1, 3 ; Read HSTR into Rt
MCR p15, 4, <Rt>, c1, c1, 3 ; Write Rt to HSTR
```

Register access is encoded as follows:

Table B1-53 HSTR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0001	0001	011

B1.70 Hyp Translation Control Register

The HTCR characteristics are:

Purpose

Controls translation table walks required for the stage 1 translation of memory accesses from Hyp mode, and holds cacheability and shareability information for the accesses.

Usage constraints

This register is accessible as follows:

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

Configurations

There are no configuration notes.

Attributes

HTCR is a 32-bit register.

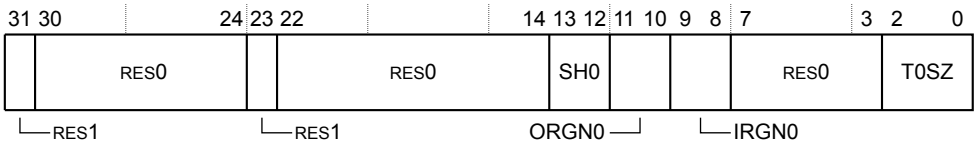

Figure B1-26 HTCR bit assignments

[31]

Reserved, RES1.

[30:24]

Reserved, RES0.

[23]

Reserved, RES1.

[22:14]

Reserved, RES0.

SH0, [13:12]

Shareability attribute for memory associated with translation table walks using TTBR0. The possible values are:

`0b00` Non-shareable.

`0b01` Reserved.

`0b10` Outer shareable.

`0b11` Inner shareable.

ORGN0, [11:10]

Outer cacheability attribute for memory associated with translation table walks using TTBR0. The possible values are:

`0b00` Normal memory, Outer Non-cacheable.

`0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Outer Write-Through Cacheable.

`0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.

IRGN0, [9:8]

Inner cacheability attribute for memory associated with translation table walks using TTBR0. The possible values are:

`0b00` Normal memory, Inner Non-cacheable.

`0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Inner Write-Through Cacheable.

`0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.

[7:3]

Reserved, RES0.

T0SZ, [2:0]

Size offset of the memory region addressed by TTBR0. The region size is $2^{(32-TSIZE)}$ bytes.

The processor does not use the implementation-defined bit, HTCR[30], so this bit is RES0.

To access the HTCR:

```
MRC p15, 4, <Rt>, c2, c0, 2; Read HTCR into Rt
MCR p15, 4, <Rt>, c2, c0, 2; Write Rt to HTCR
```

Register access is encoded as follows:

Table B1-54 HTCR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0010	0000	010

B1.71 Hyp Vector Base Address Register

The HVBAR characteristics are:

Purpose

Holds the exception base address for any exception that is taken to Hyp mode.

Usage constraints

This register is accessible as follows:

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

Configurations

There are no configuration notes.

Attributes

HVBAR is a 32-bit register.

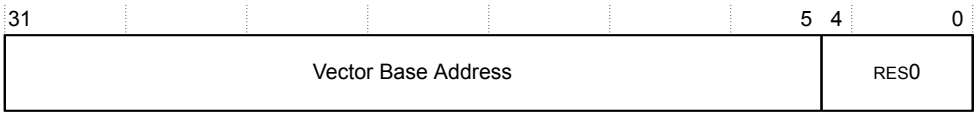

Figure B1-27 HVBAR bit assignments

Vector Base Address, [31:5]

Bits[31:5] of the base address of the exception vectors, for exceptions taken in this exception level. Bits[4:0] of an exception vector are the exception offset.

[4:0]

Reserved, RES0.

To access the HVBAR:

```
MRC p15, 4, <Rt>, c12, c0, 0 ; Read HVBAR into Rt
MCR p15, 4, <Rt>, c12, c0, 0 ; Write Rt to HVBAR
```

Register access is encoded as follows:

Table B1-55 HVBAR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	1100	0000	000

B1.72 Auxiliary Feature Register 0

ID_AFR0

The processor does not implement ID_AFR0. This register is always RES0.

B1.73 Debug Feature Register 0

The ID_DFR0 characteristics are:

Purpose

Provides top level information about the debug system in AArch32.

Usage constraints

This register is accessible as follows:

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

Must be interpreted with the Main ID Register, MIDR.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_DFR0 is a 32-bit register.

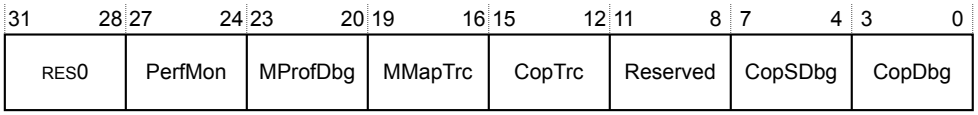

Figure B1-28 ID_DFR0 bit assignments

[31:28]

Reserved, RES0.

PerfMon, [27:24]

Indicates support for performance monitor model:

`0x3` Support for *Performance Monitor Unit version 3* (PMUv3) system registers.

MProfDbg, [23:20]

Indicates support for memory-mapped debug model for M profile processors:

`0x0` Processor does not support M profile Debug architecture.

MMapTrc, [19:16]

Indicates support for memory-mapped trace model:

`0x0` ETM is not implemented.

`0x1` Support for Arm trace architecture, with memory-mapped access.

In the Trace registers, the ETMIDR gives more information about the implementation.

CopTrc, [15:12]

Indicates support for coprocessor-based trace model:

`0x0` Processor does not support Arm trace architecture, with CP14 access.

[11:8]

Reserved, RAZ.

CopSDbg, [7:4]

Indicates support for coprocessor-based Secure debug model:

`0x6` Processor supports v8 Debug architecture, with CP14 access.

CopDbg, [3:0]

Indicates support for coprocessor-based debug model:

`0x6` Processor supports v8 Debug architecture, with CP14 access.

To access the ID_DFR0:

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

Register access is encoded as follows:

Table B1-56 ID_DFR0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	010

B1.74 Instruction Set Attribute Register 0

The ID_ISAR0 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

Must be interpreted with ID_ISAR1, ID_ISAR2, ID_ISAR3, ID_ISAR4, and ID_ISAR5. See:

- *B1.75 Instruction Set Attribute Register 1* on page B1-265
- *B1.76 Instruction Set Attribute Register 2* on page B1-267
- *B1.77 Instruction Set Attribute Register 3* on page B1-269
- *B1.78 Instruction Set Attribute Register 4* on page B1-271
- *B1.79 Instruction Set Attribute Register 5* on page B1-273

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_ISAR0 is a 32-bit register.

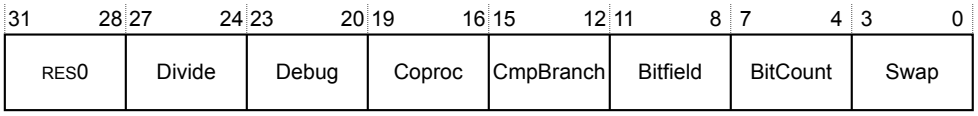

Figure B1-29 ID_ISAR0 bit assignments

[31:28]

Reserved, RES0.

Divide, [27:24]

Indicates the implemented Divide instructions:

`0x2` • `SDIV` and `UDIV` in the T32 instruction set.
• `SDIV` and `UDIV` in the A32 instruction set.

Debug, [23:20]

Indicates the implemented Debug instructions:

`0x1` `BKPT`.

Coproc, [19:16]

Indicates the implemented Coprocessor instructions:

`0x0` None implemented, except for separately attributed by the architecture including CP15, CP14, Advanced SIMD and floating-point.

CmpBranch, [15:12]

Indicates the implemented combined Compare and Branch instructions in the T32 instruction set:

`0x1` `CBNZ` and `CBZ`.

Bitfield, [11:8]

Indicates the implemented bit field instructions:

`0x1` `BFC`, `BFI`, `SBFX`, and `UBFX`.

BitCount, [7:4]

Indicates the implemented Bit Counting instructions:

`0x1` `CLZ`.

Swap, [3:0]

Indicates the implemented Swap instructions in the A32 instruction set:

`0x0` None implemented.

To access the ID_ISAR0:

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

Register access is encoded as follows:

Table B1-57 ID_ISAR0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	000

B1.75 Instruction Set Attribute Register 1

The ID_ISAR1 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

Must be interpreted with ID_ISAR0, ID_ISAR2, ID_ISAR3, ID_ISAR4 and ID_ISAR5. See:

- *B1.74 Instruction Set Attribute Register 0* on page B1-263
- *B1.76 Instruction Set Attribute Register 2* on page B1-267
- *B1.77 Instruction Set Attribute Register 3* on page B1-269
- *B1.78 Instruction Set Attribute Register 4* on page B1-271
- *B1.79 Instruction Set Attribute Register 5* on page B1-273

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_ISAR1 is a 32-bit register.

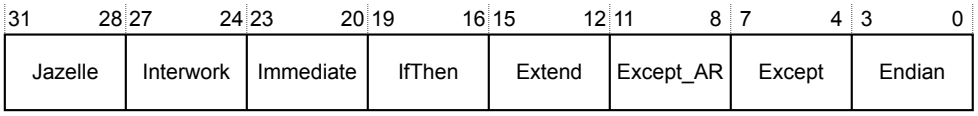

Figure B1-30 ID_ISAR1 bit assignments

Jazelle, [31:28]

Indicates the implemented Jazelle state instructions:

`0x1` The `BXJ` instruction, and the J bit in the PSR.

Interwork, [27:24]

Indicates the implemented Interworking instructions:

`0x3`
- The `BX` instruction, and the T bit in the PSR.
- The `BLX` instruction. The PC loads have `BX`-like behavior.
- Data-processing instructions in the A32 instruction set with the PC as the destination and the S bit clear, have `BX`-like behavior.

Immediate, [23:20]

Indicates the implemented data-processing instructions with long immediates:

`0x1`
- The `MOVT` instruction.
- The `MOV` instruction encodings with zero-extended 16-bit immediates.
- The T32 `ADD` and `SUB` instruction encodings with zero-extended 12-bit immediates, and other `ADD`, `ADR`, and `SUB` encodings cross-referenced by the pseudocode for those encodings.

IfThen, [19:16]

Indicates the implemented `If-Then` instructions in the T32 instruction set:

`0x1` The `IT` instructions, and the IT bits in the PSRs.

Extend, [15:12]

Indicates the implemented Extend instructions:

`0x2`
- The `SXTB`, `SXTH`, `UXTB`, and `UXTH` instructions.
- The `SXTB16`, `SXTAB`, `SXTAB16`, `SXTAH`, `UXTB16`, `UXTAB`, `UXTAB16`, and `UXTAH` instructions.

Except_AR, [11:8]

Indicates the implemented A profile exception-handling instructions:

`0x1` The `SRS` and `RFE` instructions, and the A profile forms of the `CPS` instruction.

Except, [7:4]

Indicates the implemented exception-handling instructions in the A32 instruction set:

`0x1` The `LDM` (exception return), `LDM` (user registers), and `STM` (user registers) instruction versions.

Endian, [3:0]

Indicates the implemented Endian instructions:

`0x1` The `SETEND` instruction, and the E bit in the PSRs.

To access the ID_ISAR1:

```
MRC p15, 0, <Rt>, c0, c2, 1 ; Read ID_ISAR1 into Rt
```

Register access is encoded as follows:

Table B1-58 ID_ISAR1 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	001

B1.76 Instruction Set Attribute Register 2

The ID_ISAR2 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

Must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR3, ID_ISAR4 and ID_ISAR5. See.

- *B1.74 Instruction Set Attribute Register 0* on page B1-263
- *B1.75 Instruction Set Attribute Register 1* on page B1-265
- *B1.77 Instruction Set Attribute Register 3* on page B1-269
- *B1.78 Instruction Set Attribute Register 4* on page B1-271
- *B1.79 Instruction Set Attribute Register 5* on page B1-273

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_ISAR2 is a 32-bit register.

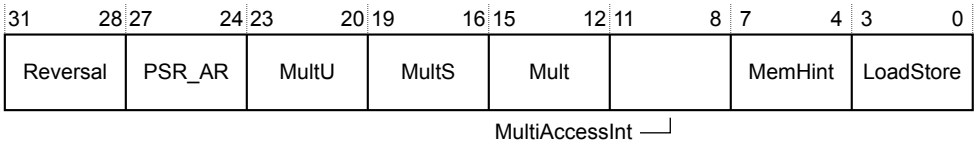

Figure B1-31 ID_ISAR2 bit assignments

Reversal, [31:28]

Indicates the implemented Reversal instructions:

`0x2` The `REV`, `REV16`, and `REVSH` instructions.

The `RBIT` instruction.

PSR_AR, [27:24]

Indicates the implemented A and R profile instructions to manipulate the PSR:

`0x1` The `MRS` and `MSR` instructions, and the exception return forms of data-processing instructions.

The exception return forms of the data-processing instructions are:

- In the A32 instruction set, data-processing instructions with the PC as the destination and the S bit set.
- In the T32 instruction set, the `SUBS PC, LR, #N` instruction.

MultU, [23:20]

Indicates the implemented advanced unsigned Multiply instructions:

`0x2` The `UMULL` and `UMLAL` instructions.

The `UMAAL` instruction.

MultS, [19:16]

Indicates the implemented advanced signed Multiply instructions.

`0x3`
- The `SMULL` and `SMLAL` instructions.
- The `SMLABB`, `SMLABT`, `SMLALBB`, `SMLALBT`, `SMLALTB`, `SMLALTT`, `SMLATB`, `SMLATT`, `SMLAWB`, `SMLAWT`, `SMULBB`, `SMULBT`, `SMULTB`, `SMULTT`, `SMULWB`, `SMULWT` instructions, and the Q bit in the PSRs.
- The `SMLAD`, `SMLADX`, `SMLALD`, `SMLALDX`, `SMLSD`, `SMLSDX`, `SMLSLD`, `SMLSLDX`, `SMMLA`, `SMMLAR`, `SMMLS`, `SMMLSR`, `SMMUL`, `SMMULR`, `SMUAD`, `SMUADX`, `SMUSD`, and `SMUSDX` instructions.

Mult, [15:12]

Indicates the implemented additional Multiply instructions:

`0x2` The `MUL` instruction.

The `MLA` instruction.

The `MLS` instruction.

MultiAccessInt, [11:8]

Indicates the support for interruptible multi-access instructions:

`0x0` No support. This means the LDM and STM instructions are not interruptible.

MemHint, [7:4]

Indicates the implemented memory hint instructions:

`0x4` The `PLD` instruction.

The `PLI` instruction.

The `PLDW` instruction.

LoadStore, [3:0]

Indicates the implemented additional load/store instructions:

`0x2` The `LDRD` and `STRD` instructions.

The Load Acquire (`LDAB`, `LDAH`, `LDA`, `LDAEXB`, `LDAEXH`, `LDAEX`, and `LDAEXD`) and Store Release (`STLB`, `STLH`, `STL`, `STLEXB`, `STLEXH`, `STLEX`, and `STLEXD`) instructions.

To access the ID_ISAR2:

```
MRC p15, 0, <Rt>, c0, c2, 2 ; Read ID_ISAR2 into Rt
```

Register access is encoded as follows:

Table B1-59 ID_ISAR2 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	010

B1.77 Instruction Set Attribute Register 3

The ID_ISAR3 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

Must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR2, ID_ISAR4, and ID_ISAR5. See:

- *B1.74 Instruction Set Attribute Register 0* on page B1-263
- *B1.75 Instruction Set Attribute Register 1* on page B1-265
- *B1.76 Instruction Set Attribute Register 2* on page B1-267
- *B1.78 Instruction Set Attribute Register 4* on page B1-271
- *B1.79 Instruction Set Attribute Register 5* on page B1-273

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_ISAR3 is a 32-bit register.

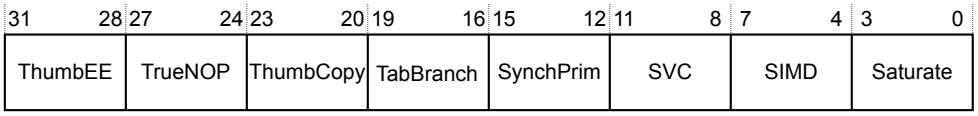

Figure B1-32 ID_ISAR3 bit assignments

ThumbEE, [31:28]

Indicates the implemented Thumb Execution Environment (T32EE) instructions:

`0x0` None implemented.

TrueNOP, [27:24]

Indicates support for True NOP instructions:

`0x1` True `NOP` instructions in both the A32 and T32 instruction sets, and additional NOP-compatible hints.

ThumbCopy, [23:20]

Indicates the support for T32 non flag-setting `MOV` instructions:

`0x1` Support for T32 instruction set encoding T1 of the `MOV` (register) instruction, copying from a low register to a low register.

TabBranch, [19:16]

Indicates the implemented Table Branch instructions in the T32 instruction set.

`0x1` The `TBB` and `TBH` instructions.

SynchPrim, [15:12]

Indicates the implemented Synchronization Primitive instructions.

`0x2` • The `LDREX` and `STREX` instructions.
• The `CLREX`, `LDREXB`, `STREXB`, and `STREXH` instructions.
• The `LDREXD` and `STREXD` instructions.

SVC, [11:8]

Indicates the implemented SVC instructions:

`0x1` The `SVC` instruction.

SIMD, [7:4]

Indicates the implemented *Single Instruction Multiple Data* (SIMD) instructions.

`0x3` • The `SSAT` and `USAT` instructions, and the Q bit in the PSRs.
• The `PKHBT`, `PKHTB`, `QADD16`, `QADD8`, `QASX`, `QSUB16`, `QSUB8`, `QSAX`, `SADD16`, `SADD8`, `SASX`, `SEL`, `SHADD16`, `SHADD8`, `SHASX`, `SHSUB16`, `SHSUB8`, `SHSAX`, `SSAT16`, `SSUB16`, `SSUB8`, `SSAX`, `SXTAB16`, `SXTB16`, `UADD16`, `UADD8`, `UASX`, `UHADD16`, `UHADD8`, `UHASX`, `UHSUB16`, `UHSUB8`, `UHSAX`, `UQADD16`, `UQADD8`, `UQASX`, `UQSUB16`, `UQSUB8`, `UQSAX`, `USAD8`, `USADA8`, `USAT16`, `USUB16`, `USUB8`, `USAX`, `UXTAB16`, `UXTB16` instructions, and the GE[3:0] bits in the PSRs.

Saturate, [3:0]

Indicates the implemented Saturate instructions:

`0x1` The `QADD`, `QDADD`, `QDSUB`, `QSUB` and the Q bit in the PSRs.

To access the ID_ISAR3:

```
MRC p15, 0, <Rt>, c0, c2, 3 ; Read ID_ISAR3 into Rt
```

Register access is encoded as follows:

Table B1-60 ID_ISAR3 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	011

B1.78 Instruction Set Attribute Register 4

The ID_ISAR4 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

Must be interpreted with ID_ISAR0, ID_ISAR1, ID_ISAR2, ID_ISAR3, and ID_ISAR5. See:

- *B1.74 Instruction Set Attribute Register 0* on page B1-263
- *B1.75 Instruction Set Attribute Register 1* on page B1-265
- *B1.76 Instruction Set Attribute Register 2* on page B1-267
- *B1.77 Instruction Set Attribute Register 3* on page B1-269
- *B1.79 Instruction Set Attribute Register 5* on page B1-273

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_ISAR4 is a 32-bit register.

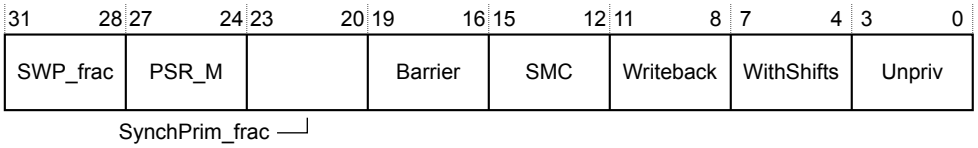

Figure B1-33 ID_ISAR4 bit assignments

SWP_frac, [31:28]

Indicates support for the memory system locking the bus for `SWP` or `SWPB` instructions:

`0x0` `SWP` and `SWPB` instructions not implemented.

PSR_M, [27:24]

Indicates the implemented M profile instructions to modify the PSRs:

`0x0` None implemented.

SynchPrim_frac, [23:20]

This field is used with the ID_ISAR3.SynchPrim field to indicate the implemented Synchronization Primitive instructions:

`0x0`
- The `LDREX` and `STREX` instructions.
- The `CLREX`, `LDREXB`, `LDREXH`, `STREXB`, and `STREXH` instructions.
- The `LDREXD` and `STREXD` instructions.

Barrier, [19:16]

Indicates the supported Barrier instructions in the A32 and T32 instruction sets:

`0x1` The `DMB`, `DSB`, and `ISB` barrier instructions.

SMC, [15:12]

Indicates the implemented SMC instructions:

0x1 The SMC instruction.

Writeback, [11:8]

Indicates the support for writeback addressing modes:

0x1 Processor supports all of the writeback addressing modes defined in Armv8.

WithShifts, [7:4]

Indicates the support for instructions with shifts:

0x4
- Support for shifts of loads and stores over the range LSL 0-3.
- Support for other constant shift options, both on load/store and other instructions.
- Support for register-controlled shift options.

Unpriv, [3:0]

Indicates the implemented unprivileged instructions:

0x2
- The LDRBT, LDRT, STRBT, and STRT instructions.
- The LDRHT, LDRSBT, LDRSHT, and STRHT instructions.

To access the ID_ISAR4:

```
MRC p15, 0, <Rt>, c0, c2, 4 ; Read ID_ISAR4 into Rt
```

Register access is encoded as follows:

Table B1-61 ID_ISAR4 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	100

B1.79 Instruction Set Attribute Register 5

The ID_ISAR5 characteristics are:

Purpose

Provides information about the instruction sets that the processor implements.

Usage constraints

This register is accessible as follows:

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

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

- *B1.74 Instruction Set Attribute Register 0* on page B1-263
- *B1.75 Instruction Set Attribute Register 1* on page B1-265
- *B1.76 Instruction Set Attribute Register 2* on page B1-267
- *B1.77 Instruction Set Attribute Register 3* on page B1-269
- *B1.78 Instruction Set Attribute Register 4* on page B1-271

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_ISAR5 is a 32-bit register.

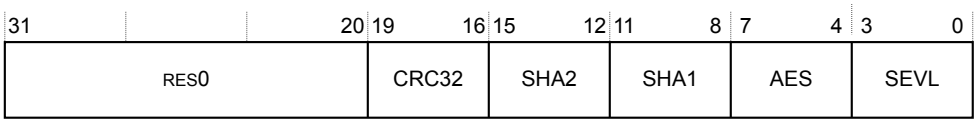

Figure B1-34 ID_ISAR5 bit assignments

[31:20]

Reserved, RES0.

CRC32, [19:16]

Indicates whether CRC32 instructions are implemented in AArch32 state:

`0x1` CRC32 instructions are implemented.

SHA2, [15:12]

Indicates whether SHA2 instructions are implemented in AArch32 state:

`0x0` Cryptographic Extensions are not implemented or are disabled.

`0x1` `SHA256H`, `SHA256H2`, `SHA256SU0`, and `SHA256SU1` instructions are implemented.

See the *Cortex®-A32 Processor Cryptographic Extension Technical Reference Manual* for more information.

SHA1, [11:8]

Indicates whether SHA1 instructions are implemented in AArch32 state:

`0x0` Cryptographic Extensions are not implemented or are disabled.

`0x1` `SHA1C`, `SHA1P`, `SHA1M`, `SHA1H`, `SHA1SU0`, and `SHA1SU1` instructions are implemented.

See the *Cortex®-A32 Processor Cryptographic Extension Technical Reference Manual* for more information.

AES, [7:4]

Indicates whether AES instructions are implemented in AArch32 state:

`0x0` Cryptographic Extensions are not implemented or are disabled.

`0x2` `AESE`, `AESD`, `AESMC` and `AESIMC`, plus `PMULL` and `PMULL2` instructions operating on 64-bit data.

See the *Cortex®-A32 Processor Cryptographic Extension Technical Reference Manual* for more information.

SEVL, [3:0]

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
```

Register access is encoded as follows:

Table B1-62 ID_ISAR5 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0010	101

B1.80 Memory Model Feature Register 0

The ID_MMFR0 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

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

- *B1.81 Memory Model Feature Register 1* on page B1-277
- *B1.82 Memory Model Feature Register 2* on page B1-279
- *B1.83 Memory Model Feature Register 3* on page B1-281

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_MMFR0 is a 32-bit register.

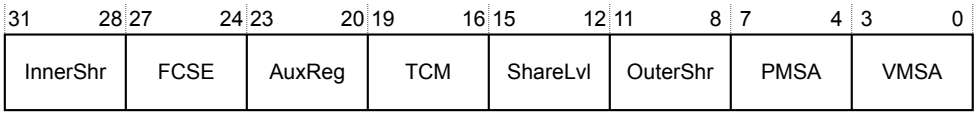

Figure B1-35 ID_MMFR0 bit assignments

InnerShr, [31:28]

Indicates the innermost shareability domain implemented:

`0x1` Implemented with hardware coherency support.

FCSE, [27:24]

Indicates support for *Fast Context Switch Extension* (FCSE):

`0x0` Not supported.

AuxReg, [23:20]

Indicates support for Auxiliary registers:

`0x2` Support for Auxiliary Fault Status Registers (AIFSR and ADFSR) and Auxiliary Control Register.

TCM, [19:16]

Indicates support for TCMs and associated DMAs:

`0x0` Not supported.

ShareLvl, [15:12]

Indicates the number of shareability levels implemented:

`0x1` Two levels of shareability implemented.

OuterShr, [11:8]

Indicates the outermost shareability domain implemented:

`0x1` Implemented with hardware coherency support.

PMSA, [7:4]

Indicates support for a *Protected Memory System Architecture* (PMSA):

`0x0` Not supported.

VMSA, [3:0]

Indicates support for a *Virtual Memory System Architecture* (VMSA).

`0x5` Support for:
- VMSAv7, with support for remapping and the Access flag.
- The PXN bit in the Short-descriptor translation table format descriptors.
- The Long-descriptor translation table format.

To access the ID_MMFR0:

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

Register access is encoded as follows:

Table B1-63 ID_MMFR0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	100

B1.81 Memory Model Feature Register 1

The ID_MMFR1 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

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

- *B1.80 Memory Model Feature Register 0* on page B1-275
- *B1.82 Memory Model Feature Register 2* on page B1-279
- *B1.83 Memory Model Feature Register 3* on page B1-281

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_MMFR1 is a 32-bit register.

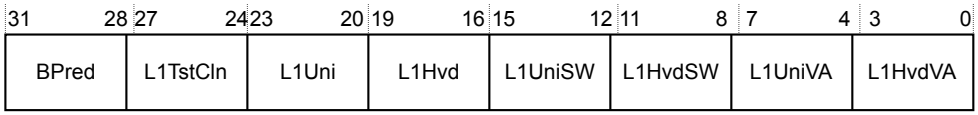

Figure B1-36 ID_MMFR1 bit assignments

BPred, [31:28]

Indicates branch predictor management requirements:

`0x4` For execution correctness, branch predictor requires no flushing at any time.

L1TstCln, [27:24]

Indicates the supported L1 Data cache test and clean operations, for Harvard or unified cache implementation:

`0x0` None supported.

L1Uni, [23:20]

Indicates the supported entire L1 cache maintenance operations, for a unified cache implementation:

`0x0` None supported.

L1Hvd, [19:16]

Indicates the supported entire L1 cache maintenance operations, for a Harvard cache implementation:

`0x0` None supported.

L1UniSW, [15:12]

Indicates the supported L1 cache line maintenance operations by set/way, for a unified cache implementation:

`0x0` None supported.

L1HvdSW, [11:8]

Indicates the supported L1 cache line maintenance operations by set/way, for a Harvard cache implementation:

`0x0` None supported.

L1UniVA, [7:4]

Indicates the supported L1 cache line maintenance operations by MVA, for a unified cache implementation:

`0x0` None supported.

L1HvdVA, [3:0]

Indicates the supported L1 cache line maintenance operations by MVA, for a Harvard cache implementation:

`0x0` None supported.

To access the ID_MMFR1:

```
MRC p15, 0, <Rt>, c0, c1, 5; Read ID_MMFR1 into Rt
```

Register access is encoded as follows:

Table B1-64 ID_MMFR1 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	101

B1.82 Memory Model Feature Register 2

The ID_MMFR2 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

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

- *B1.80 Memory Model Feature Register 0* on page B1-275
- *B1.81 Memory Model Feature Register 1* on page B1-277
- *B1.83 Memory Model Feature Register 3* on page B1-281

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_MMFR2 is a 32-bit register.

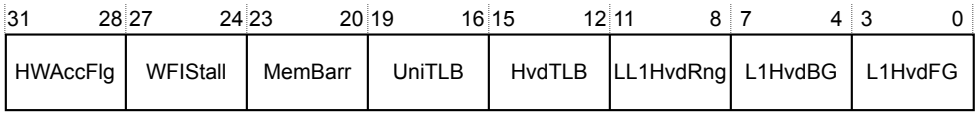

Figure B1-37 ID_MMFR2 bit assignments

HWAccFlg, [31:28]

Hardware Access Flag. Indicates support for a Hardware Access flag, as part of the VMSAv7 implementation:

`0x0` Not supported.

WFIStall, [27:24]

Wait For Interrupt Stall. Indicates the support for *Wait For Interrupt* (WFI) stalling:

`0x1` Support for WFI stalling.

MemBarr, [23:20]

Memory Barrier. Indicates the supported CP15 memory barrier operations.

`0x2` Supported CP15 memory barrier operations are:

- *Data Synchronization Barrier* (DSB).
- *Instruction Synchronization Barrier* (ISB).
- *Data Memory Barrier* (DMB).

UniTLB, [19:16]

Unified TLB. Indicates the supported TLB maintenance operations, for a unified TLB implementation.

`0x6` Supported unified TLB maintenance operations are:

- Invalidate all entries in the TLB.
- Invalidate TLB entry by MVA.
- Invalidate TLB entries by ASID match.
- Invalidate instruction TLB and data TLB entries by MVA All ASID. This is a shared unified TLB operation.
- Invalidate Hyp mode unified TLB entry by MVA.
- Invalidate entire Non-secure EL1 and EL0 unified TLB.
- Invalidate entire Hyp mode unified TLB.
- `TLBIMVALIS`, `TLBIMVAALIS`, `TLBIMVALHIS`, `TLBIMVAL`, `TLBIMVAAL`, and `TLBIMVALH`.
- `TLBIIPAS2IS`, `TLBIIPAS2LIS`, `TLBIIPAS2`, and `TLBIIPAS2L`.

HvdTLB, [15:12]

Harvard TLB. Indicates the supported TLB maintenance operations, for a Harvard TLB implementation:

`0x0` Not supported.

LL1HvdRng, [11:8]

L1 Harvard cache Range. Indicates the supported L1 cache maintenance range operations, for a Harvard cache implementation:

`0x0` Not supported.

L1HvdBG, [7:4]

L1 Harvard cache Background fetch. Indicates the supported L1 cache background prefetch operations, for a Harvard cache implementation:

`0x0` Not supported.

L1HvdFG, [3:0]

L1 Harvard cache Foreground fetch. Indicates the supported L1 cache foreground prefetch operations, for a Harvard cache implementation:

`0x0` Not supported.

To access the ID_MMFR2:

```
MRC p15,0,<Rt>,c0,c1,6 ; Read ID_MMFR2 into Rt
```

Register access is encoded as follows:

Table B1-65 ID_MMFR2 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	110

B1.83 Memory Model Feature Register 3

The ID_MMFR3 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

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

- *B1.80 Memory Model Feature Register 0* on page B1-275
- *B1.81 Memory Model Feature Register 1* on page B1-277
- *B1.82 Memory Model Feature Register 2* on page B1-279

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_MMFR3 is a 32-bit register.

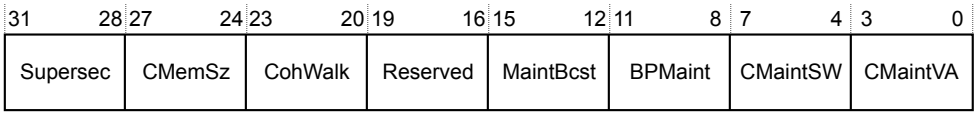

Figure B1-38 ID_MMFR3 bit assignments

Supersec, [31:28]

Supersections. Indicates support for supersections:

`0x0` Supersections supported.

CMemSz, [27:24]

Cached Memory Size. Indicates the size of physical memory supported by the processor caches:

`0x2` 1TByte, corresponding to a 40-bit physical address range.

CohWalk, [23:20]

Coherent walk. Indicates whether translation table updates require a clean to the point of unification:

`0x1` Updates to the translation tables do not require a clean to the point of unification to ensure visibility by subsequent translation table walks.

[19:16]

Reserved, RES0.

MaintBcst, [15:12]

Maintenance broadcast. Indicates whether cache, TLB and branch predictor operations are broadcast:

`0x2` Cache, TLB and branch predictor operations affect structures according to shareability and defined behavior of instructions.

BPMaint, [11:8]

Branch predictor maintenance. Indicates the supported branch predictor maintenance operations.

`0x2` Supported branch predictor maintenance operations are:
- Invalidate all branch predictors.
- Invalidate branch predictors by MVA.

CMaintSW, [7:4]

Cache maintenance by set/way. Indicates the supported cache maintenance operations by set/way.

`0x1` Supported hierarchical cache maintenance operations by set/way are:
- Invalidate data cache by set/way.
- Clean data cache by set/way.
- Clean and invalidate data cache by set/way.

CMaintVA, [3:0]

Cache maintenance by MVA. Indicates the supported cache maintenance operations by MVA.

`0x1` Supported hierarchical cache maintenance operations by MVA are:
- Invalidate data cache by MVA.

  Invalidate data cache by MVA operations are treated as clean and invalidate data cache by MVA operations on the executing core. If the operation is broadcast to another core then it is broadcast as an invalidate data cache by MVA operation.
- Clean data cache by MVA.
- Clean and invalidate data cache by MVA.
- Invalidate instruction cache by MVA.
- Invalidate all instruction cache entries.

To access the ID_MMFR3:

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

Register access is encoded as follows:

Table B1-66 ID_MMFR3 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	111

B1.84 Processor Feature Register 0

The ID_PFR0 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

ID_PFR0 must be interpreted with ID_PFR1.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_PFR0 is a 32-bit register.

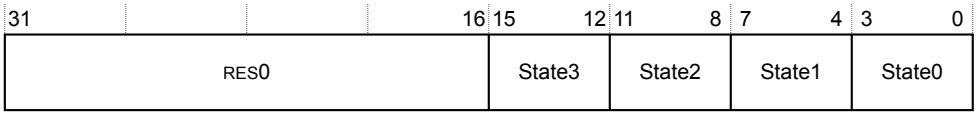

Figure B1-39 ID_PFR0 bit assignments

[31:16]

Reserved, RES0.

State3, [15:12]

Indicates support for *Thumb Execution Environment* (T32EE) instruction set. This value is:

`0x0` Processor does not support the T32EE instruction set.

State2, [11:8]

Indicates support for Jazelle. This value is:

`0x1` Processor supports trivial implementation of Jazelle.

State1, [7:4]

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.

State0, [3:0]

Indicates support for A32 instruction set. This value is:

`0x1` A32 instruction set implemented.

To access the ID_PFR0:

```
MRC p15,0,<Rt>,c0,c1,0 ; Read ID_PFR0 into Rt
```

Register access is encoded as follows:

Table B1-67 ID_PFR0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	000

B1.85 Processor Feature Register 1

The ID_PFR1 characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

Must be interpreted with ID_PFR0.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ID_PFR1 is a 32-bit register.

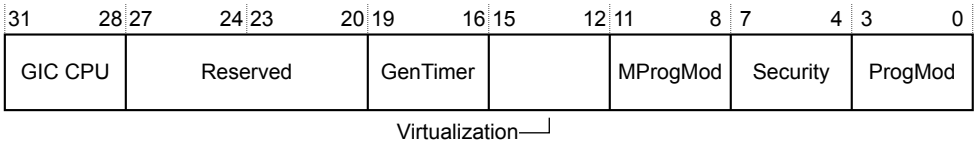

Figure B1-40 ID_PFR1 bit assignments

GIC CPU, [31:28]

GIC CPU support:

`0x0` GIC CPU interface is disabled, **GICCDISABLE** is HIGH, or not implemented.

`0x1` GIC CPU interface is implemented and enabled, **GICCDISABLE** is LOW.

[27:20]

Reserved, RAZ.

GenTimer, [19:16]

Generic Timer support:

`0x1` Generic Timer implemented.

Virtualization, [15:12]

Indicates support for Virtualization:

`0x1` Virtualization implemented.

MProgMod, [11:8]

M profile programmers model support:

`0x0` Not supported.

Security, [7:4]

Security support:

`0x1` Security implemented. This includes support for Monitor mode and the SMC instruction.

ProgMod, [3:0]

Indicates support for the standard programmers model for Armv4 and later.

Model must support User, FIQ, IRQ, Supervisor, Abort, Undefined and System modes:

`0x1` Supported.

To access the ID_PFR1:

```
MRC p15,0,<Rt>,c0,c1,1 ; Read ID_PFR1 into Rt
```

Register access is encoded as follows:

Table B1-68 ID_PFR1 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0001	001

B1.86 Instruction Fault Address Register

The IFAR characteristics are:

Purpose

Holds the virtual address of the faulting address that caused a synchronous Prefetch Abort exception.

Usage constraints

This register is accessible as follows:

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

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

IFAR (S) is architecturally mapped to HIFAR. See *B1.65 Hyp Instruction Fault Address Register* on page B1-246.

Attributes

IFAR is a 32-bit register.

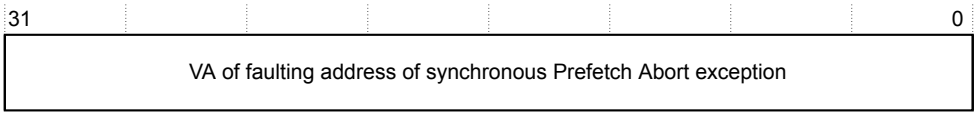

Figure B1-41 IFAR bit assignments

VA, [31:0]

The Virtual Address of faulting address of synchronous Prefetch Abort exception.

To access the IFAR:

```
MRC p15, 0, <Rt>, c6, c0, 2; Read IFAR into Rt
MCR p15, 0, <Rt>, c6, c0, 2; Write Rt to IFAR
```

B1.87 Instruction Fault Status Register

The IFSR characteristics are:

Purpose

Holds status information about the last instruction fault.

Usage constraints

This register is accessible as follows:

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

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Attributes

IFSR is a 32-bit register.

There are two formats for this register. The current translation table format determines which format of the register is used.

B1.88 IFSR with Short-descriptor translation table format

IFSR has a specific format when using the Short-descriptor translation table format.

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

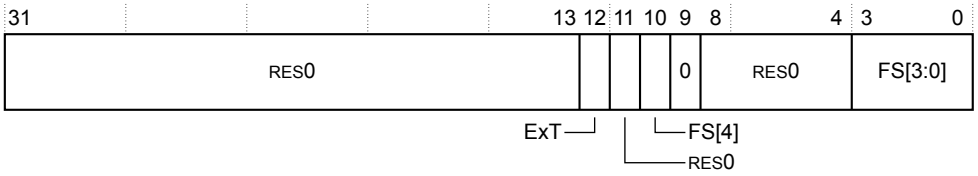

Figure B1-42 IFSR bit assignments for Short-descriptor translation table format

[31:13]
Reserved, RES0.

ExT,[12]

External abort type. This field indicates whether an AXI Decode or Slave error caused an abort:

`0` External abort marked as DECERR.

`1` External abort marked as SLVERR.

For aborts other than external aborts this bit always returns 0.

[11]
Reserved, RES0.

FS[4],[10]
Part of the Fault Status field. See bits [3:0].

[9]
RAZ.

[8:5]
Reserved, RES0.

FS[3:0],[4:0]

Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved.

`0b00010` Debug event.

`0b00011` Access flag fault, section.

`0b00101` Translation fault, section.

`0b00110` Access flag fault, page.

`0b00111` Translation fault, page.

`0b01000` Synchronous external abort, non-translation.

`0b01001` Domain fault, section.

`0b01011` Domain fault, page.

`0b01100` Synchronous external abort on translation table walk, first level.

`0b01101` Permission Fault, Section.

`0b01110` Synchronous external abort on translation table walk, second Level.

`0b01111` Permission fault, page.

`0b10000` TLB conflict abort.

`0b11001` Synchronous parity error on memory access.

`0b11100` Synchronous parity error on translation table walk, first level.

`0b11110` Synchronous parity error on translation table walk, second level.

B1.89 IFSR with Long-descriptor translation table format

IFSR has a specific format when using the Long-descriptor translation table format.

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

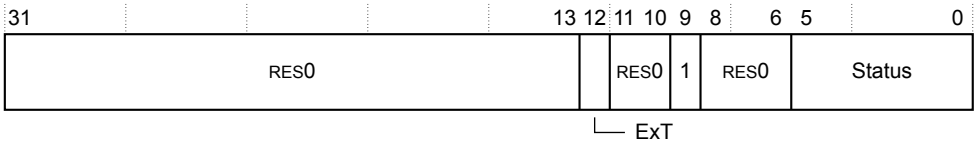

Figure B1-43 IFSR bit assignments for Long-descriptor translation table format

[31:13]

Reserved, RES0.

ExT,[12]

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]

RAO.

[8:6]

Reserved, RES0.

Status,[5:0]

Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved.

`0b000000` Address size fault in TTBR0 or TTBR1.

`0b0001LL` Translation fault, LL bits indicate level.

`0b0010LL` Access fault flag, LL bits indicate level.

`0b0011LL` Permission fault, LL bits indicate level.

`0b010000` Synchronous external abort.

`0b0101LL` Synchronous external abort on translation table walk, LL bits indicate level.

`0b011000` Synchronous parity error on memory access.

`0b0111LL` Synchronous parity error on memory access on translation table walk, LL bits indicate level.

`0b100001` Alignment fault.

`0b100010` Debug event.

`0b110000` TLB conflict abort.

Table B1-69 Encodings of LL bits associated with the MMU fault

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

If a Data Abort exception is generated by an instruction cache maintenance operation when the Long-descriptor translation table format is selected, the fault is reported as a Cache Maintenance fault in the DFSR or HSR with the appropriate Fault Status code. For such exceptions reported in the DFSR, the corresponding IFSR is UNKNOWN.

To access the IFSR:

```
MRC p15, 0, <Rt>, c5, c0, 1; Read IFSR into Rt
MCR p15, 0, <Rt>, c5, c0, 1; Write Rt to IFSR
```

Register access is encoded as follows:

Table B1-70 IFSR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0101	0000	001

B1.90 Interrupt Status Register

The ISR characteristics are:

Purpose

Shows whether an IRQ, FIQ, or external abort is pending. An indicated pending abort might be a physical abort or a virtual abort.

Usage constraints

This register is accessible as follows:

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

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

ISR is a 32-bit register.

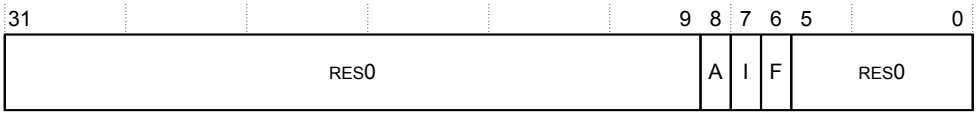

Figure B1-44 ISR bit assignments

[31:9]

Reserved, RES0.

A, [8]

External abort pending bit:

0 No pending external abort.

1 An external abort is pending.

I, [7]

IRQ pending bit. Indicates whether an IRQ interrupt is pending:

0 No pending IRQ.

1 An IRQ interrupt is pending.

F, [6]

FIQ pending bit. Indicates whether an FIQ interrupt is pending:

0 No pending FIQ.

1 An FIQ interrupt is pending.

[5:0]

Reserved, RES0.

To access the ISR:

```
MRC p15, 0, <Rt>, c12, c1, 1; Read ISR into Rt
```

Register access is encoded as follows:

Table B1-71 ISR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	1100	0001	000

B1.91 L2 Auxiliary Control Register

The L2ACTLR characteristics are:

Purpose

Provides configuration and control options for the L2 memory system.

Usage constraints

This register is accessible as follows:

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

You can write to this register only when the L2 memory system is idle. Arm recommends that you write to this register after a powerup reset before the MMU is enabled and before any AXI, ACE, CHI, or ACP traffic has begun.

If the register must be modified after a powerup reset sequence, to idle the L2 memory system, you must take the following steps:

1. Disable the MMU from each core followed by an ISB to ensure the MMU disable operation is complete, then followed by a DSB to drain previous memory transactions.
2. Ensure that the system has no outstanding AC channel coherence requests to the Cortex-A32 processor.
3. Ensure that the system has no outstanding ACP requests to the Cortex-A32 processor.

When the L2 is idle, the processor can update the L2ACTLR followed by an `ISB`. After the L2ACTLR is updated, the MMUs can be enabled and normal ACE and ACP traffic can resume.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

L2ACTLR is a 32-bit register.

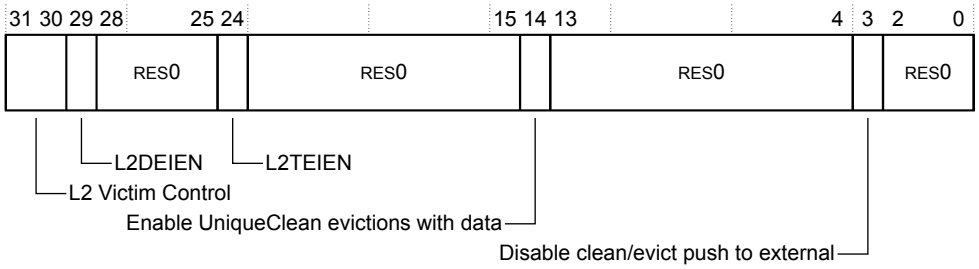

Figure B1-45 L2ACTLR bit assignments

[31:30]

L2 Victim Control.

`0b10` This is the default value. Software must not change it.

L2DEIEN, [29]

L2 cache data RAM error injection enable. The possible values are:

`0` Normal behavior, errors are not injected. This is the reset value.

`1` Double-bit errors are injected on all writes to the L2 cache data RAMs.

[28:25]

Reserved, RES0.

L2TEIEN, [24]

L2 cache tag RAM error injection enable. The possible values are:

0 Normal behavior, errors are not injected. This is the reset value.

1 Double-bit errors are injected on all writes to the L2 cache tag RAMs.

[23:15]

Reserved, RES0.

Enable UniqueClean evictions with data, [14]

Enables UniqueClean evictions with data. The possible values are:

0 Disables UniqueClean evictions with data. This is the reset value for ACE.

1 Enables UniqueClean evictions with data. This is the reset value for CHI.

In AXI implementations, this field is RES0.

[13:4]

Reserved, RES0.

Disable clean/evict push to external, [3]

Disables clean/evict push to external. The possible values are:

0 Enables clean/evict to be pushed out to external. This is the reset value for ACE.

1 Disables clean/evict from being pushed to external. This is the reset value for CHI.

In AXI implementations, this field is RES1.

[2:0]

Reserved, RES0.

To access the L2ACTLR:

```
MRC p15, 1, <Rt>, c15, c0, 0; Read L2ACTLR into Rt
MCR p15, 1, <Rt>, c15, c0, 0; Write Rt to L2ACTLR
```

Register access is encoded as follows:

Table B1-72 L2ACTLR access encoding

coproc	opc1	CRn	CRm	opc2
1111	001	1111	0000	000

B1.92 L2 Control Register

The L2CTLR characteristics are:

Purpose

Provides IMPLEMENTATION DEFINED control options for the L2 memory system.

Usage constraints

This register is accessible as follows:

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

L2CTLR is writable. However, all writes to this register are ignored.

Configurations

There is one L2CTLR for the Cortex-A32 processor.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

L2CTLR is a 32-bit register.

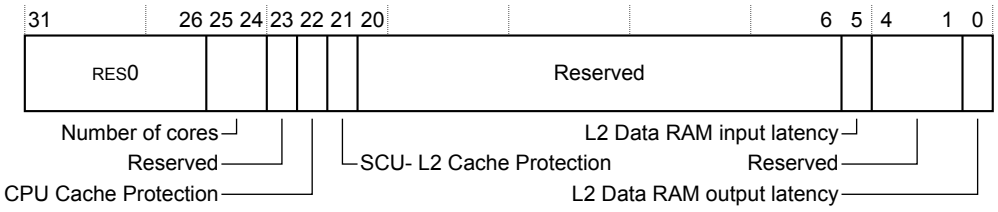

Figure B1-46 L2CTLR bit assignments

[31:26]

Reserved, RES0.

Number of cores, [25:24]

Number of cores present:

`0b00` One core, core 0.

`0b01` Two cores, core 0 and core 1.

`0b10` Three cores, cores 0 to 2.

`0b11` Four cores, cores 0 to 3.

These bits are read-only and the value of this field is set to the number of cores present in the configuration.

[23]

Reserved, RAZ.

CPU Cache Protection, [22]

CPU Cache Protection. Core RAMs are implemented:

`0` Without ECC.

`1` With ECC.

This field is RO.

SCU-L2 Cache Protection, [21]

SCU-L2 Cache Protection. L2 cache is implemented:

`0` Without ECC.

`1` With ECC.

This field is RO.

[20:6]

Reserved, RAZ.

L2 Data RAM input latency, [5]

L2 data RAM input latency

`0` 1-cycle input delay from L2 data RAMs.

`1` 2-cycle input delay from L2 data RAMs.

This field is RO.

[4:1]

Reserved, RAZ.

L2 Data RAM output latency, [0]

L2 data RAM output latency:

`0` 2-cycle output delay from L2 data RAMs.

`1` 3-cycle output delay from L2 data RAMs.

This field is RO.

To access the L2CTLR:

```
MRC p15, 1, <Rt>, c9, c0, 2; Read L2CTLR into Rt
```

Register access is encoded as follows:

Table B1-73 L2CTLR access encoding

coproc	opc1	CRn	CRm	opc2
1111	001	1001	0000	010

B1.93 L2 Extended Control Register

The L2ECTLR characteristics are:

Purpose

Provides additional IMPLEMENTATION DEFINED control options for the L2 memory system. This register is used for dynamically changing, but implementation specific, control bits.

Usage constraints

This register is accessible as follows:

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

The L2ECTLR can be written dynamically.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

There is one L2ECTLR for the Cortex-A32 processor.

Attributes

L2ECTLR is a 32-bit register.

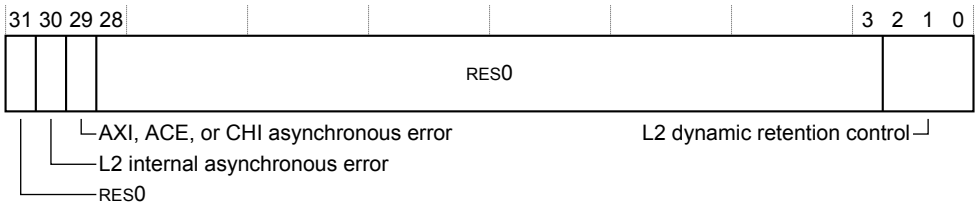

Figure B1-47 L2ECTLR bit assignments

[31]

Reserved, RES0.

L2 internal asynchronous error, [30]

L2 internal asynchronous error caused by L2 RAM double-bit ECC error. The possible values are:

`0` No pending asynchronous error. This is the reset value.

`1` An asynchronous error has occurred.

A write of 0 clears this bit. A write of 1 is ignored.

AXI, ACE, or CHI asynchronous error, [29]

AXI, ACE, or CHI asynchronous error indication. The possible values are:

`0` No pending asynchronous error.

`1` An asynchronous error has occurred.

A write of 0 clears this bit. A write of 1 is ignored.

[28:3]

Reserved, RES0.

L2 dynamic retention control, [2:0]

L2 dynamic retention control. The possible values are:

`0b000` L2 dynamic retention disabled. This is the reset value.

`0b001` 2 Generic Timer ticks required before retention entry.

`0b010` 8 Generic Timer ticks required before retention entry.

`0b011` 32 Generic Timer ticks required before retention entry.

`0b100` 64 Generic Timer ticks required before retention entry.

`0b101` 128 Generic Timer ticks required before retention entry.

`0b110` 256 Generic Timer ticks required before retention entry.

`0b111` 512 Generic Timer ticks required before retention entry.

To access the L2ECTLR:

```
MRC p15, 1, <Rt>, c9, c0, 3; Read L2ECTLR into Rt
MCR p15, 1, <Rt>, c9, c0, 3; Write Rt to L2ECTLR
```

Register access is encoded as follows:

Table B1-74 L2ECTLR access encoding

coproc	opc1	CRn	CRm	opc2
1111	001	1001	0000	011

B1.94 L2 Memory Error Syndrome Register

The L2MERRSR characteristics are:

Purpose

Holds ECC errors on the:

- L2 data RAMs.
- L2 tag RAMs.
- SCU snoop filter RAMs.

Usage constraints

This register is accessible as follows:

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

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

A write of any value to the register updates the register to `0x0000000000000000`.

Attributes

L2MERRSR is a a 64-bit register.

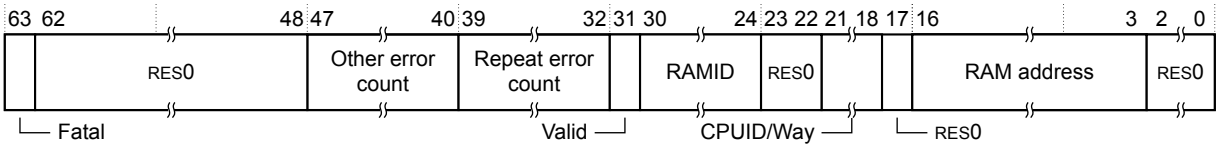

Figure B1-48 L2MERRSR bit assignments

Fatal, [63]

Fatal bit. This bit is set to 1 on the first memory error that caused a data abort. It is a sticky bit so that after it is set, it remains set until the register is written.

The reset value is 0.

[62:48]

Reserved, RES0.

Other error count, [47:40]

This field is set to 0 on the first memory error and is incremented on any memory error that does not match the RAMID and Bank/Way information in this register while the sticky Valid bit is set.

The reset value is 0.

Repeat error count, [39:32]

This field is set to 0 on the first memory error and is incremented on any memory error that exactly matches the RAMID and Bank/Way information in this register while the sticky Valid bit is set.

The reset value is 0.

Valid, [31]

Valid bit. This bit is set to 1 on the first memory error. It is a sticky bit so that after it is set, it remains set until the register is written.

The reset value is 0.

RAMID, [30:24]

RAM Identifier. Indicates the RAM in which the first memory error occurred. The possible values are:

`0x10` L2 tag RAM.

`0x11` L2 data RAM.

`0x12` SCU snoop filter RAM.

[23:22]

Reserved, RES0.

CPUID/Way, [21:18]

Indicates the RAM where the first memory error occurred.

L2 tag RAM

`0x0` Way 0

`0x1` Way 1

...

`0x6` Way 6

`0x7` Way 7

L2 data RAM

`0x0` Bank 0

`0x1` Bank 1

...

`0x7` Bank 7

`0x8-0xF` Unused

SCU snoop filter RAM

`0x0` CPU0:Way0

`0x1` CPU0:Way1

...

`0xE` CPU3:Way2

`0xF` CPU3:Way3

[17]

Reserved, RES0.

RAM Address, [16:3]

Indicates the index address of the first memory error.

[2:0]

Reserved, RES0.

- A fatal error results in the RAMID, CPU ID/Way and RAM address recording the fatal error, even if the sticky bit was set.
- If two or more memory errors in the same RAM occur in the same cycle, only one error is reported.

- If two or more first memory error events from different RAMs occur in the same cycle, one of the errors is selected arbitrarily, while the Other error count field is incremented only by one.
- If two or more memory error events from different RAMs, that do not match the RAMID, bank, way, or index information in this register while the sticky Valid bit is set, occur in the same cycle, the Other error count field is incremented only by one.

To access the L2MERRSR:

```
MRRC p15, 3, <Rt>, <Rt2>, c15;  Read L2MERRSR into Rt and Rt2
MCRR p15, 3, <Rt>, <Rt2>, c15;  Write Rt and Rt2 to L2MERRSR
```

Register access is encoded as follows:

Table B1-75 L2MERRSR access encoding

coproc	opc1	CRm
1111	0011	1111

B1.95 Memory Attribute Indirection Registers 0 and 1

The MAIR0 and MAIR1 characteristics are:

Purpose

To provide the memory attribute encodings corresponding to the possible AttrIndx values in a Long-descriptor format translation table entry for stage 1 translations.

Usage constraints

These registers are accessible as follows:

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

Accessible only when using the Long-descriptor translation table format. When using the Short-descriptor format see, instead, *B1.101 Primary Region Remap Register* on page B1-316 and *B1.99 Normal Memory Remap Register* on page B1-313.

AttrIndx[2], from the translation table descriptor, selects the appropriate MAIR. Setting AttrIndx[2] to 0 selects MAIR0 and setting AttrIndx[2] to 1 selects MAIR1.

The Secure instance of the register gives the value for memory accesses from Secure state.

The Non-secure instance of the register gives the value for memory accesses from Non-secure states other than Hyp mode.

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Write access to MAIR0(S) and MAIR1(S) is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

Attributes

MAIR0 is a 32-bit register when TTBCR.EAE==1.

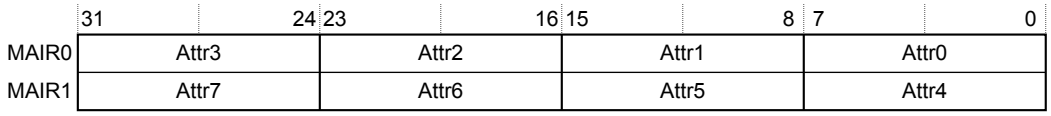

Figure B1-49 MAIR0 and MAIR1 bit assignments

Attr*m*, [7:0]

Where m is 0-7.

The memory attribute encoding for an AttrIndx[2:0] entry in a Long descriptor format translation table entry, where:

- AttrIndx[2] selects the appropriate MAIR:
  - Setting AttrIndx[2] to 0 selects MAIR0.
  - Setting AttrIndx[2] to 1 selects MAIR1.
- AttrIndx[2:0] gives the value of <n> in Attr<n>.

Table B1-76 Attr<n>[7:4] bit assignments

Bits	Meaning
0b0000	Device memory. See *Attr<n>[3:0] bit assignments* on page B1-305 for the type of Device memory.
0b00RW, RW not 00	Normal Memory, Outer Write-through transient. The transient hint is ignored.
0b0100	Normal Memory, Outer Non-Cacheable.
0b01RW, RW not 00	Normal Memory, Outer Write-back transient. The transient hint is ignored.
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 B1-77 Attr<n>[3:0] bit assignments

Bits	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
0b0000	Device-nGnRnE memory	UNPREDICTABLE
0b00RW, RW not 00	UNPREDICTABLE	Normal Memory, Inner Write-through transient
0b0100	Device-nGnRE memory	Normal memory, Inner Non-Cacheable
0b01RW, RW not 00	UNPREDICTABLE	Normal Memory, Inner Write-back transient
0b1000	Device-nGRE memory	Normal Memory, Inner Write-through non-transient (RW=00)
0b10RW, RW not 00	UNPREDICTABLE	Normal Memory, Inner Write-through non-transient
0b1100	Device-GRE memory	Normal Memory, Inner Write-back non-transient (RW=00)
0b11RW, RW not 00	UNPREDICTABLE	Normal Memory, Inner Write-back non-transient

The following table shows the encoding of the R and W bits that are used, in some Attr<n> encodings in *Table B1-76 Attr<n>[7:4] bit assignments* on page B1-305 and *Table B1-77 Attr<n>[3:0] bit assignments* on page B1-305, to define the read-allocate and write-allocate policies:

Table B1-78 Encoding of R and W bits in some Attr*m* fields

R or W	Meaning
0	Do not allocate
1	Allocate

To access the MAIR0:

```
MRC p15, 0, <Rt>, c10, c2, 0    ; Read MAIR0 into Rt
MCR p15, 0, <Rt>, c10, c2, 0    ; Write Rt to MAIR0
```

Register access is encoded as follows:

Table B1-79 MAIR0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	1010	0010	000

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
```

Register access is encoded as follows:

Table B1-80 MAIR1 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	1010	0010	001

B1.96 Main ID Register

The MIDR characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

Configurations

MIDR is architecturally mapped to the external register MIDR_EL1.

Attributes

MIDR is a 32-bit register.

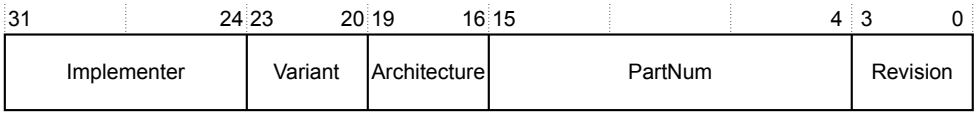

Figure B1-50 MIDR bit assignments

Implementer, [31:24]

Indicates the implementer code. This value is:

`0x41` ARM.

Variant, [23:20]

Indicates the variant number of the processor. This is the major revision number *n* in the *rn* part of the rnpn description of the product revision status. This value is:

`0x1` r1p0.

Architecture, [19:16]

Indicates the architecture code. This value is:

`0xF` Defined in the CPUID scheme.

PartNum, [15:4]

Indicates the primary part number. This value is:

`0xD01` Cortex-A32 processor.

Revision, [3:0]

Indicates the minor revision number of the processor. This is the minor revision number *n* in the *pn* part of the *rnpn* description of the product revision status. This value is:

`0x0` r1p0.

To access the MIDR:

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

Register access is encoded as follows:

Table B1-81 MIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0000	000

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

B1.97 Multiprocessor Affinity Register

The MPIDR characteristics are:

Purpose

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

EDDEVAFF0 is a read-only copy of MPIDR accessible from the external debug interface.

Usage constraints

This register is accessible as follows:

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

Configurations

The MPIDR is mapped to external EDDEVAFF0 register.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

MPIDR is a 32-bit register.

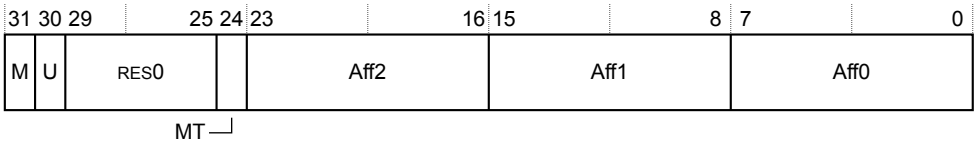

Figure B1-51 MPIDR bit assignments

M, [31]

Reserved, RES1.

U, [30]

Indicates a uniprocessor system, as distinct from core 0 in a multiprocessor system. This value is one of:

0 Processor is part of a multiprocessor system. This is the value for implementations with more than one core, and for implementations with an ACE or CHI interface.

1 Processor is part of a uniprocessor system. This is the value for single core implementations with an AXI master interface.

[29:25]

Reserved, RES0.

MT, [24]

Indicates whether the lowest level of affinity consists of logical cores that are implemented using a multi-threading type approach. This value is:

0 Performance of cores at the lowest affinity level is largely independent.

Aff2, [23:16]

Affinity level 2. Second highest level affinity field.

Indicates the value read in the **CLUSTERIDAFF2** configuration signal.

Aff1, [15:8]

Affinity level 1. Third highest level affinity field.

Indicates the value read in the **CLUSTERIDAFF1** configuration signal.

Aff0, [7:0]

Affinity level 0. Lowest level affinity field.

Indicates the core number in the Cortex-A32 processor. The possible values are:

`0x0`	A processor with one core only.
`0x0`, `0x1`	A cluster with two cores.
`0x0`, `0x1`, `0x2`	A cluster with three cores.
`0x0`, `0x1`, `0x2`, `0x3`	A cluster with four cores.

To access the MPIDR:

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

Register access is encoded as follows:

Table B1-82 MPIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0000	101

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

B1.98 Non-Secure Access Control Register

The NSACR characteristics are:

Purpose

Defines the Non-secure access permission to CP0 to CP13.

Usage constraints

This register is accessible as follows:

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

Any read or write to NSACR in Secure EL1 state in AArch32 is trapped as an exception to EL3.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Write access to NSACR is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

Attributes

NSACR is a 32-bit register.

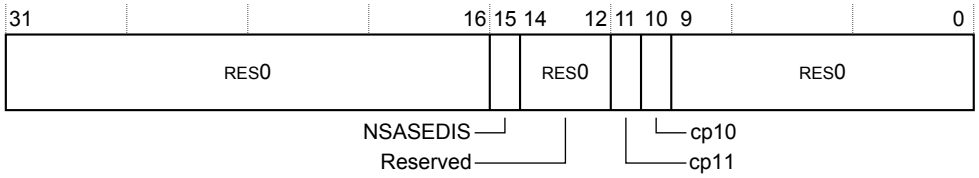

Figure B1-52 NSACR bit assignments

[31:16]

Reserved, RES0.

NSASEDIS, [15]

Disable Non-secure Advanced SIMD functionality:

`0` This bit has no effect on the ability to write CPACR.ASEDIS, this is the reset value.

`1` When executing in Non-secure state, the CPACR.ASEDIS bit has a fixed value of 1 and writes to it are ignored.

If Advanced SIMD and floating-point are not implemented, this bit is RES0.

[14:12]

Reserved, RES0.

cp11, [11]

Non-secure access to CP11 enable:

`0` Secure access only. Any attempt to access CP11 in Non-secure state results in an Undefined Instruction exception. If the processor is in Non-secure state, the corresponding bits in the CPACR ignore writes and read as `0b00`, access denied. This is the reset value.

`1` Secure or Non-secure access.

If Advanced SIMD and Floating-point are not implemented, this bit is RES0.

cp10, [10]

Non-secure access to CP10 enable:

`0` Secure access only. Any attempt to access CP10 in Non-secure state results in an Undefined Instruction exception. If the processor is in Non-secure state, the corresponding bits in the CPACR ignore writes and read as `0b00`, access denied. This is the reset value.

`1` Secure or Non-secure access.

If Advanced SIMD and floating-point are not implemented, this bit is RES0.

[9:0]

Reserved, RES0.

If the CP11 and CP10 fields are set to different values, the behavior is CONSTRAINED UNPREDICTABLE. It is the same as if both fields were set to the value of CP10, in all respects other than the value read back by explicitly reading CP11.

To access the NSACR:

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

Register access is encoded as follows:

Table B1-83 NSACR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0001	0001	010

B1.99 Normal Memory Remap Register

The NMRR characteristics are:

Purpose

Provides additional mapping controls for memory regions that are mapped as Normal memory by their entry in the PRRR.

Usage constraints

This register is accessible as follows:

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

The register is:

- Used in conjunction with the PRRR.
- Not accessible when using the Long-descriptor translation table format.

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Write access to NMRR(S) is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

Attributes

NMRR is a 32-bit register when TTBCR.EAE is 0.

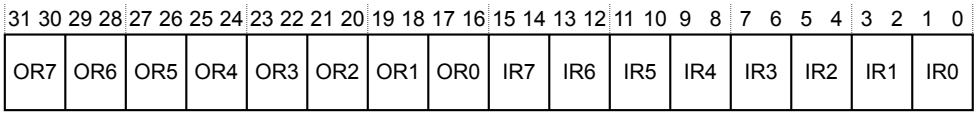

Figure B1-53 NMRR bit assignments

ORn, [2n+17:2n+16]

Outer Cacheable property mapping for memory attributes *n*, where n is 0-7, if the region is mapped as Normal memory by the PRRR.TR*n* entry. *n* is the value of the TEX[0], C and B bits, see *Memory attributes and the n value for the PRRR field descriptions* on page B1-317. The possible values of this field are:

`0b00` Region is Non-cacheable.

`0b01` Region is Write-Back, Write-Allocate.

`0b10` Region is Write-Through, no Write-Allocate.

`0b11` Region is Write-Back, no Write-Allocate.

IRn, [2n+1:2n]

Inner Cacheable property mapping for memory attributes *n*, where n is 0-7, if the region is mapped as Normal Memory by the PRRR.TR*n* entry. *n* is the value of the TEX[0], C and B bits, see *Memory attributes and the n value for the PRRR field descriptions* on page B1-317. The possible values of this field are the same as those given for the OR*n* field.

To access the NMRR:

```
MRC p15, 0, <Rt>, c10, c2, 1    ; Read NMRR into Rt
MCR p15, 0, <Rt>, c10, c2, 1    ; Write Rt to NMRR
```

Register access is encoded as follows:

Table B1-84 NMRR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	1010	0010	001

B1.100 Physical Address Register

PAR

The processor does not use any implementation-defined bits in the 32-bit format or 64-bit format PAR. Bit[8] is RES0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

B1.101 Primary Region Remap Register

The PRRR characteristics are:

Purpose

Controls the top level mapping of the TEX[0], C, and B memory region attributes.

Usage constraints

This register is accessible as follows:

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

PRRR is not accessible when the Long-descriptor translation table format is in use. See, instead, *B1.95 Memory Attribute Indirection Registers 0 and 1* on page B1-304.

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Write access to PRRR(S) is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

Attributes

PRRR is a 32-bit register when TTBCR.EAE==0.

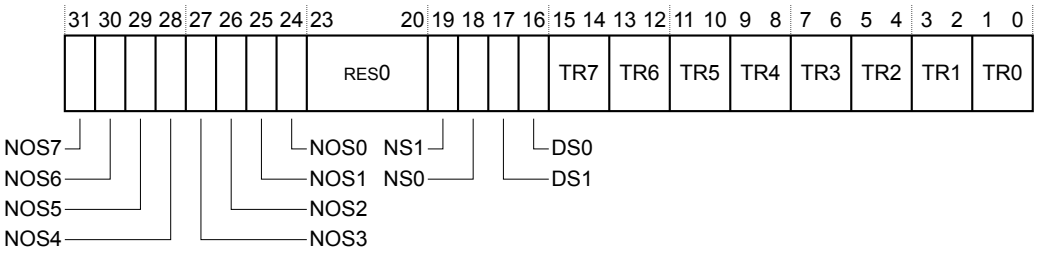

Figure B1-54 PRRR bit assignments

NOS*n*, [24+*n*]

Outer Shareable property mapping for memory attributes *n*, where n is 0-7, if the region is mapped as Normal Shareable. *n* is the value of the TEX[0], C and B bits concatenated. The possible values of each NOS*n* bit are:

`0` Memory region is Outer Shareable.

`1` Memory region is Inner Shareable.

The value of this bit is ignored if the region is Normal or Device memory that is not Shareable.

[23:20]

Reserved, RES0.

NS1, [19]

Mapping of S = 1 attribute for Normal memory. This bit gives the mapped Shareable attribute for a region of memory that:

- Is mapped as Normal memory.
- Has the S bit set to 1.

The possible values of the bit are:

`0` Region is not Shareable.

`1` Region is Shareable.

NS0, [18]

Mapping of S = 0 attribute for Normal memory. This bit gives the mapped Shareable attribute for a region of memory that:

- Is mapped as Normal memory.
- Has the S bit set to 0.

The possible values of the bit are the same as those given for the NS1 bit, bit[19].

DS1, [17]

Mapping of S = 1 attribute for Device memory. This bit gives the mapped Shareable attribute for a region of memory that:

- Is mapped as Device memory.
- Has the S bit set to 1.

This field has no significance in the processor.

DS0, [16]

Mapping of S = 0 attribute for Device memory. This bit gives the mapped Shareable attribute for a region of memory that:

- Is mapped as Device memory.
- Has the S bit set to 0.

This field has no significance in the processor.

TR*n*, [2*n*+1:2*n*]

Primary TEX mapping for memory attributes *n*, where n is 0-7. *n* is the value of the TEX[0], C and B bits, see *Memory attributes and the n value for the PRRR field descriptions* on page B1-317. This field defines the mapped memory type for a region with attributes *n*. The possible values of the field are:

`0b00` Device (nGnRnE).

`0b01` Device (not nGnRnE).

`0b10` Normal Memory.

`0b11` Reserved, effect is UNPREDICTABLE.

The following table shows the mapping between the memory region attributes and the *n* value used in the PRRR.nOS*n* and PRRR.TR*n* field descriptions.

Table B1-85 Memory attributes and the *n* value for the PRRR field descriptions

Attributes			*n* value
TEX[0]	C	B	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Large physical address translations use Long-descriptor translation table formats and MAIR0 replaces the PRRR, and MAIR1 replaces the NMRR. For more information see *B1.95 Memory Attribute Indirection Registers 0 and 1* on page B1-304.

To access the PRRR:

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

Register access is encoded as follows:

Table B1-86 PRRR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	1010	0010	000

B1.102 Revision ID Register

The REVIDR characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

REVIDR is a 32-bit register.

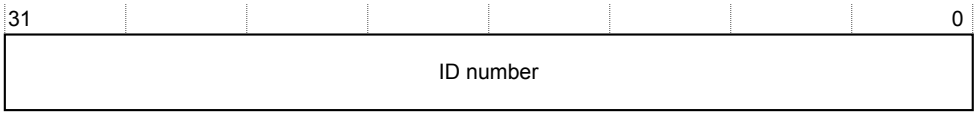

Figure B1-55 REVIDR bit assignments

ID number, [31:0]

Implementation-specific revision information. The reset value is determined by the specific Cortex-A32 processor implementation.

`0x00000000` Revision code is zero.

To access the REVIDR:

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

Register access is encoded as follows:

Table B1-87 REVIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0000	110

B1.103 Reset Management Register

The RMR characteristics are:

Purpose

Controls the execution state that the processor boots into and allows request of a warm reset.

Usage constraints

This register is accessible as follows:

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

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

RMR is a 32-bit register.

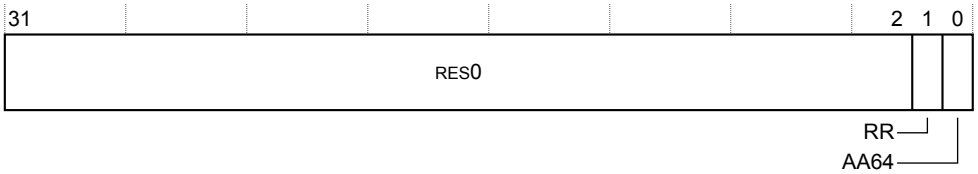

Figure B1-56 RMR bit assignments

[31:2]

Reserved, RES0.

RR, [1]

Reset Request. The possible values are:

`0` This is the reset value.

`1` Requests a warm reset. This bit is set to 0 by either a cold or warm reset.

The bit is strictly a request.

The RR bit drives the **WARMRSTREQ** output signal.

AA64, [0]

Reserved, RES0.

To access the RMR:

```
MRC p15,0,<Rt>,c12,c0,2 ; Read RMR into Rt
MCR p15,0,<Rt>,c12,c0,2 ; Write Rt to RMR
```

Register access is encoded as follows:

Table B1-88 RMR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	1100	0000	010

B1.104 Secure Configuration Register

The SCR characteristics are:

Purpose

Defines the configuration of the current security state. It specifies:

- The security state of the processor, Secure or Non-secure.
- What state the processor branches to, if an IRQ, FIQ or external abort occurs.
- Whether the CPSR.F and CPSR.A bits can be modified when SCR.NS = 1.

Usage constraints

This register is accessible as follows:

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

Any read or write to SCR in Secure EL1 state in AArch32 is trapped as an exception to EL3.

Configurations

The SCR is a Restricted access register that exists only in the Secure state.

Attributes

SCR is a 32-bit register.

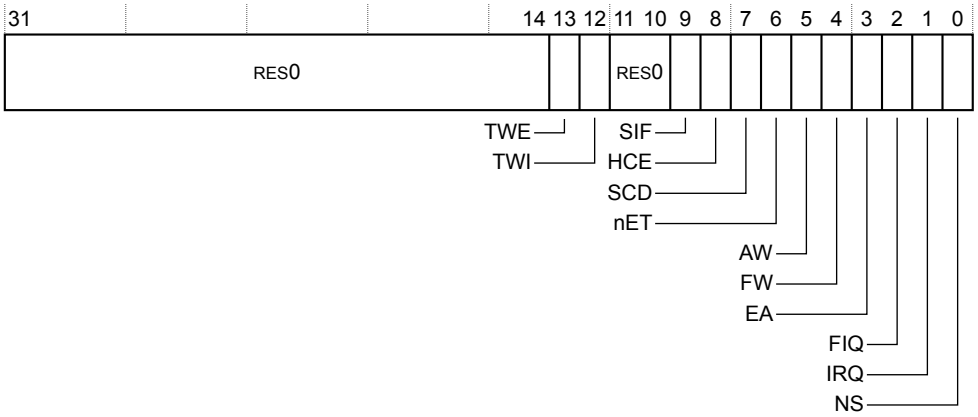

Figure B1-57 SCR bit assignments

[31:14]

Reserved, RES0.

TWE, [13]

Trap `WFE` instructions. The possible values are:

`0` `WFE` instructions are not trapped. This is the reset value.

`1` `WFE` instructions executed in any mode other than Monitor mode are trapped to Monitor mode as UNDEFINED if the instruction would otherwise cause suspension of execution, that is if:

- The event register is not set.
- There is not a pending WFE wakeup event.
- The instruction does not cause another exception.

TWI, [12]

Trap WFI instructions. The possible values are:

0 WFI instructions are not trapped. This is the reset value.

1 WFI instructions executed in any mode other than Monitor mode are trapped to Monitor mode as UNDEFINED if the instruction would otherwise cause suspension of execution.

[11:10]

Reserved, RES0.

SIF, [9]

Secure Instruction Fetch. When the processor is in Secure state, this bit disables instruction fetches from Non-secure memory. The possible values are:

0 Secure state instruction fetches from Non-secure memory permitted. This is the reset value.

1 Secure state instruction fetches from Non-secure memory not permitted.

HCE, [8]

Hyp Call enable. This bit enables use of the HVC instruction from Non-secure EL1 modes. The possible values are:

0 The HVC instruction is UNDEFINED in any mode. This is the reset value.

1 The HVC instruction enabled in Non-secure EL1, and performs a Hyp Call.

SCD, [7]

Secure Monitor Call disable. Makes the SMC instruction UNDEFINED in Non-secure state. The possible values are:

0 SMC executes normally in Non-secure state, performing a Secure Monitor Call. This is the reset value.

1 The SMC instruction is UNDEFINED in Non-secure state.

A trap of the SMC instruction to Hyp mode takes priority over the value of this bit.

nET, [6]

Not Early Termination. This bit disables early termination.

This bit is not implemented, RES0.

AW, [5]

A bit writable. This bit controls whether CPSR.A can be modified in Non-secure state.

- CPSR.A can be modified only in Secure state. This is the reset value.
- CPSR.A can be modified in any security state.

FW, [4]

F bit writable. This bit controls whether CPSR.F can be modified in Non-secure state:

- CPSR.F can be modified only in Secure state. This is the reset value.
- CPSR.F can be modified in any security state.

EA, [3]

External Abort handler. This bit controls which mode takes external aborts. The possible values are:

0 External aborts taken in abort mode. This is the reset value.

1 External aborts taken in Monitor mode.

FIQ, [2]

FIQ handler. This bit controls which mode takes FIQ exceptions. The possible values are:

0 FIQs taken in FIQ mode. This is the reset value.

1 FIQs taken in Monitor mode.

IRQ, [1]

IRQ handler. This bit controls which mode takes IRQ exceptions. The possible values are:

0 IRQs taken in IRQ mode. This is the reset value.

1 IRQs taken in Monitor mode.

NS, [0]

Non-secure bit. Except when the processor is in Monitor mode, this bit determines the security state of the processor. The possible values are:

0 Processor is in secure state. This is the reset value.

1 Processor is in non-secure state.

To access the SCR:

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

Register access is encoded as follows:

Table B1-89 SCR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0001	0001	000

B1.105 System Control Register

The SCTLR characteristics are:

Purpose

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

Usage constraints

The SCTLR is accessible as follows:

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

Control bits in the SCTLR that are not applicable to a VMSA implementation read as the value that most closely reflects that implementation, and ignore writes.

Some bits in the register are read-only. These bits relate to non-configurable features of an implementation, and are provided for compatibility with previous versions of the architecture.

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Write access to SCTLR(S) is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

Attributes

SCTLR is a 32-bit register.

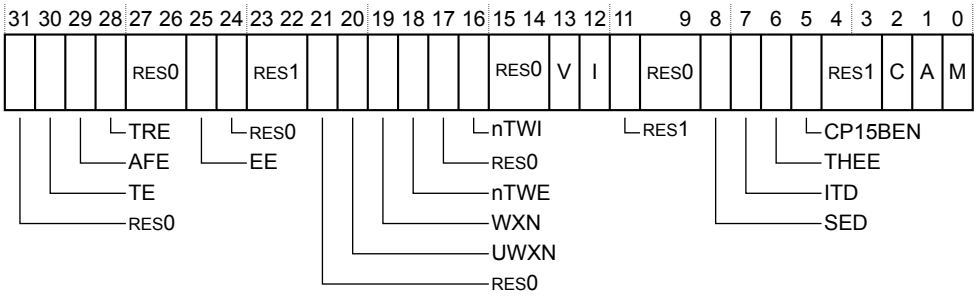

Figure B1-58 SCTLR bit assignments

[31]

Reserved, RES0.

TE, [30]

T32 Exception enable. This bit controls whether exceptions are taken in A32 or T32 state:

`0` Exceptions, including reset, taken in A32 state.

`1` Exceptions, including reset, taken in T32 state.

The input **CFGTE** defines the reset value of the TE bit.

AFE, [29]

Access Flag Enable. This bit enables use of the AP[0] bit in the translation descriptors as the Access flag. It also restricts access permissions in the translation descriptors to the simplified model:

0 In the translation table descriptors, AP[0] is an access permissions bit. The full range of access permissions is supported. No Access flag is implemented. This is the reset value.

1 In the translation table descriptors, AP[0] is the Access flag. Only the simplified model for access permissions is supported.

TRE, [28]

TEX remap enable. This bit enables remapping of the TEX[2:1] bits for use as two translation table bits that can be managed by the operating system. Enabling this remapping also changes the scheme used to describe the memory region attributes in the VMSA:

0 TEX remap disabled. TEX[2:0] are used, with the C and B bits, to describe the memory region attributes. This is the reset value.

1 TEX remap enabled. TEX[2:1] are reassigned for use as bits managed by the operating system. The TEX[0], C and B bits are used to describe the memory region attributes, with the MMU remap registers.

[27:26]

Reserved, RES0.

EE, [25]

Exception Endianness bit. The value of this bit defines the value of the CPSR.E bit on entry to an exception vector, including reset. This value also indicates the endianness of the translation table data for translation table lookups:

0 Little endian.

1 Big endian.

The input **CFGEND** defines the reset value of the EE bit.

[24]

Reserved, RES0.

[23:22]

Reserved, RES1.

[21]

Reserved, RES0.

UWXN, [20]

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.

WXN, [19]

Write permission implies *Execute Never* (XN). This bit can be used to require all memory regions with write permissions to be treated as XN.

0 Regions with write permission are not forced to be XN, this is the reset value.

1 Regions with write permissions are forced to be XN.

nTWE, [18]

Not trap WFE.

`0` If a WFE instruction executed at EL0 would cause execution to be suspended, such as if the event register is not set and there is not a pending WFE wakeup event, it is taken as an exception to EL1 using the 0x1 ESR code.

`1` WFE instructions are executed as normal.

[17]

Reserved, RES0.

nTWI, [16]

Not trap WFI.

`0` If a WFI instruction executed at EL0 would cause execution to be suspended, such as if there is not a pending WFI wakeup event, it is taken as an exception to EL1 using the 0x1 ESR code.

`1` WFI instructions are executed as normal.

[15:14]

Reserved, RES0.

V, [13]

Vectors bit. This bit selects the base address of the exception vectors:

`0` Normal exception vectors, base address `0x00000000`. Software can remap this base address using the VBAR.

`1` High exception vectors, base address `0xFFFF0000`. This base address is never remapped.

The input **VINITHI** defines the reset value of the V bit.

I, [12]

Instruction cache enable bit. This is a global enable bit for instruction caches:

`0` Instruction caches disabled. If SCTLR.M is set to 0, instruction accesses from stage 1 of the EL0/EL1 translation regime are to Normal memory, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable.

`1` Instruction caches enabled. If SCTLR.M is set to 0, instruction accesses from stage 1 of the EL0/EL1 translation regime are to Normal memory, Outer Shareable, Inner Write-Through, Outer Write-Through.

[11]

Reserved, RES1

[10:9]

Reserved, RES0

SED, [8]

SETEND Disable:

`0` The SETEND instruction is available.

`1` The SETEND instruction is UNALLOCATED.

ITD, [7]

IT Disable:

`0` The IT instruction functionality is available.

`1` All encodings of the IT instruction with hw1[3:0]!=1000 are UNDEFINED and treated as unallocated. All encodings of the subsequent instruction with the following values for hw1 are UNDEFINED (and treated as unallocated):

`11xxxxxxxxxxxxx`	All 32-bit instructions, B(2), B(1), Undefined, SVC, Load/Store multiple
`1x11xxxxxxxxxxx`	Miscellaneous 16-bit instructions
`1x100xxxxxxxxxx`	ADD Rd, PC, #imm
`01001xxxxxxxxxx`	LDR Rd, [PC, #imm]
`0100x1xxx1111xxx`	ADD(4),CMP(3), MOV, BX pc, BLX pc
`010001xx1xxxx111`	ADD(4),CMP(3), MOV

THEE, [6]
Reserved, RES0.

CP15BEN, [5]

CP15 barrier enable.

`0` CP15 barrier operations disabled. Their encodings are UNDEFINED.

`1` CP15 barrier operations enabled.

[4:3]
Reserved, RES1.

C, [2]

Cache enable. This is a global enable bit for data and unified caches:

`0` Data and unified caches disabled, this is the reset value.

`1` Data and unified caches enabled.

A, [1]

Alignment check enable. This is the enable bit for Alignment fault checking:

`0` Alignment fault checking disabled, this is the reset value.

`1` Alignment fault checking enabled.

M, [0]

MMU enable. This is a global enable bit for the MMU stage 1 address translation:

`0` EL1 and EL0 stage 1 MMU disabled.

`1` EL1 and EL0 stage 1 MMU enabled.

To access the SCTLR:

```
MRC p15, 0, <Rt>, c1, c0, 0 ; Read SCTLR into Rt
MCR p15, 0, <Rt>, c1, c0, 0 ; Write Rt to SCTLR
```

Register access is encoded as follows:

Table B1-90 SCTLR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0001	0000	000

B1.106 Secure Debug Control Register

The SDCR characteristics are:

Purpose

Controls debug and performance monitors functionality in Secure state.

Usage constraints

This register is accessible as follows:

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

Configurations

Write access to SDCR is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

Attributes

SDCR is a 32-bit register.

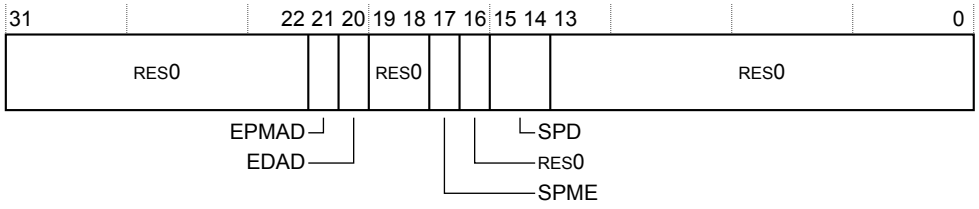

Figure B1-59 SDCR bit assignments

[31:22]

Reserved, RES0.

EPMAD, [21]

External debugger access to Performance Monitors registers disabled. This disables access to these registers by an external debugger:

0 Access to Performance Monitors registers from external debugger is permitted. This is the reset value.

1 Access to Performance Monitors registers from external debugger is disabled, unless overridden by authentication interface.

EDAD, [20]

External debugger access to breakpoint and watchpoint registers disabled. This disables access to these registers by an external debugger:

0 Access to breakpoint and watchpoint registers from external debugger is permitted. This is the reset value.

1 Access to breakpoint and watchpoint registers from external debugger is disabled, unless overridden by authentication interface.

[19:18]

Reserved, RES0.

SPME, [17]

Secure performance monitors enable. This allows event counting in Secure state:

`0` Event counting prohibited in Secure state, unless overridden by the authentication interface. This is the reset value.

`1` Event counting allowed in Secure state.

[16]

Reserved, RES0.

SPD, [15:14]

AArch32 secure privileged debug. Enables or disables debug exceptions in Secure state, other than Software breakpoint instructions. The possible values are:

`0b00` Legacy mode. Debug exceptions from Secure EL1 are enabled by the authentication interface.

`0b10` Secure privileged debug disabled. Debug exceptions from Secure EL1 are disabled.

`0b11` Secure privileged debug enabled. Debug exceptions from Secure EL1 are enabled.

The value `0b01` is reserved.

If debug exceptions from Secure EL1 are enabled, then debug exceptions from Secure EL0 are also enabled.

Otherwise, debug exceptions from Secure EL0 are enabled only if SDER32_EL3.SUIDEN is 1.

SPD is ignored in Non-secure state. Debug exceptions from Software breakpoint instruction debug events are always enabled.

[13:0]

Reserved, RES0.

To access the SDCR:

```
MRC p15,0,<Rt>,c1,c3,1 ; Read SDCR into Rt
MCR p15,0,<Rt>,c1,c3,1 ; Write Rt to SDCR
```

Register access is encoded as follows:

Table B1-91 SDCR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0001	0011	001

B1.107 Secure Debug Enable Register

The SDER characteristics are:

Purpose

Controls invasive and non-invasive debug in the Secure EL0 state.

Usage constraints

This register is accessible as follows:

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

Configurations

This register is accessible only in Secure state.

Write access to SDER is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

Attributes

SDER is a 32-bit register.

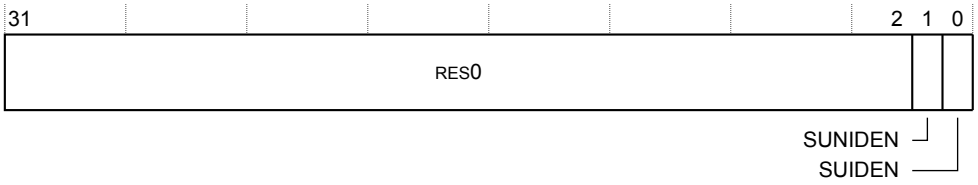

Figure B1-60 SDER bit assignments

[31:2]

Reserved, RES0.

SUNIDEN, [1]

Secure User Non-invasive Debug Enable. The possible values are:

`0` Non-invasive debug not permitted in Secure EL0 state. This is the Warm reset value.

`1` Non-invasive debug permitted in Secure EL0 state.

SUIDEN, [0]

Secure User Invasive Debug Enable. The possible values are:

`0` Invasive debug not permitted in Secure EL0 state. This is the Warm reset value.

`1` Invasive debug permitted in Secure EL0 state.

To access the SDER:

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

Register access is encoded as follows:

Table B1-92 SDER access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0001	0001	001

B1.108 TCM Type Register

TCMTR

The processor does not implement the features described by the TCMTR. This register is always RAZ.

B1.109 TLB Type Register

TLBTR

The processor does not implement the features described by the TLBTR. This register is always RAZ.

B1.110 Translation Table Base Control Register

The TTBCR characteristics are:

Purpose

Determines which of the Translation Table Base Registers defines the base address for a translation table walk required for the stage 1 translation of a memory access from any mode other than Hyp mode. Also controls the translation table format and, when using the Long-descriptor translation table format, holds cacheability and shareability information.

Usage constraints

This register is accessible as follows:

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

The processor does not use the implementation-defined bit, TTBCR[30], when using the Long-descriptor translation table format, so this bit is RES0.

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Write access to TTBCR(S) is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

Attributes

TTBCR is a 32-bit register.

There are two formats for this register. TTBCR.EAE determines which format of the register is used.

B1.111 TTBCR with Short-descriptor translation table format

TTBCR has a specific format when using the Short-descriptor translation table format. TTBCR.EAE determines which format of the register is in use.

The following figure shows the TTBCR bit assignments when TTBCR.EAE is 0.

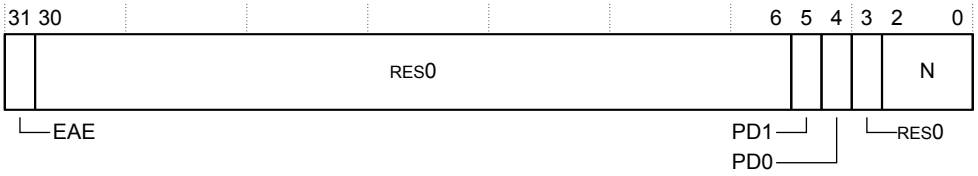

Figure B1-61 TTBCR bit assignments, TTBCR.EAE is 0

EAE, [31]

Extended Address Enable.

0 Use the 32-bit translation system, with the Short-descriptor translation table format.

[30:6]

Reserved, RES0.

PD1, [5]

Translation table walk disable for translations using TTBR1. This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using TTBR1. The possible values are:

0 Perform translation table walks using TTBR1.

1 A TLB miss on an address that is translated using TTBR1 generates a Translation fault. No translation table walk is performed.

PD0, [4]

Translation table walk disable for translations using TTBR0. This bit controls whether a translation table walk is performed on a TLB miss for an address that is translated using TTBR0. The possible values are:

0 Perform translation table walks using TTBR0.

1 A TLB miss on an address that is translated using TTBR0 generates a Translation fault. No translation table walk is performed.

[3]

Reserved, RES0.

N, [2:0]

Indicate the width of the base address held in TTBR0. In TTBR0, the base address field is bits[31:14-N]. The value of N also determines:

- Whether TTBR0 or TTBR1 is used as the base address for translation table walks.
- The size of the translation table pointed to by TTBR0.

N can take any value from 0 to 7, that is, from 0b000 to 0b111.

When N has its reset value of 0, the translation table base is compatible with Armv5 and Armv6.

Resets to 0.

B1.112 TTBCR with Long-descriptor translation table format

TTBCR has a specific format when using the Long-descriptor translation table format. TTBCR.EAE determines which format of the register is in use.

The following figure shows the TTBCR bit assignments when TTBCR.EAE is 1.

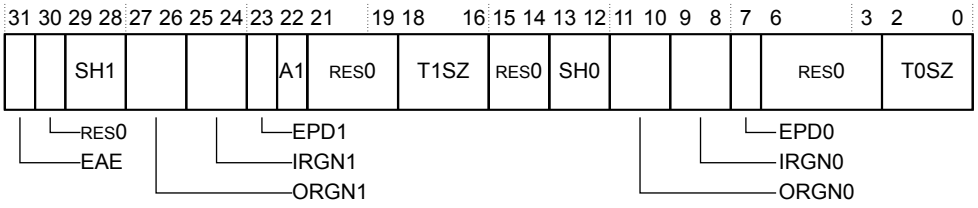

Figure B1-62 TTBCR bit assignments, TTBCR.EAE is 1

EAE, [31]

Extended Address Enable:

`1` Use the 40-bit translation system, with the Long-descriptor translation table format.

[30]

Reserved, RES0.

SH1, [29:28]

Shareability attribute for memory associated with translation table walks using TTBR1:

`0b00` Non-shareable.

`0b10` Outer Shareable.

`0b11` Inner Shareable.

Other values are reserved.

Resets to 0.

ORGN1, [27:26]

Outer cacheability attribute for memory associated with translation table walks using TTBR1:

`0b00` Normal memory, Outer Non-cacheable.

`0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Outer Write-Through Cacheable.

`0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.

Resets to 0.

IRGN1, [25:24]

Inner cacheability attribute for memory associated with translation table walks using TTBR1:

`0b00` Normal memory, Inner Non-cacheable.

`0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Inner Write-Through Cacheable.

`0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.

Resets to 0.

EPD1, [23]

Translation table walk disable for translations using TTBR1. This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using TTBR1:

`0`
Perform translation table walks using TTBR1.

`1`
A TLB miss on an address that is translated using TTBR1 generates a Translation fault. No translation table walk is performed.

A1, [22]

Selects whether TTBR0 or TTBR1 defines the ASID:

`0` TTBR0.ASID defines the ASID.

`1` TTBR1.ASID defines the ASID.

[21:19]

Reserved, RES0.

T1SZ, [18:16]

The size offset of the memory region addressed by TTBR1. The region size is $2^{32-T1SZ}$ bytes.

Resets to 0.

[15:14]

Reserved, RES0.

SH0, [13:12]

Shareability attribute for memory associated with translation table walks using TTBR0:

`0b00` Non-shareable.

`0b10` Outer Shareable.

`0b11` Inner Shareable.

Other values are reserved.

Resets to 0.

ORGN0, [11:10]

Outer cacheability attribute for memory associated with translation table walks using TTBR0:

`0b00` Normal memory, Outer Non-cacheable.

`0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Outer Write-Through Cacheable.

`0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.

Resets to 0.

IRGN0, [9:8]

Inner cacheability attribute for memory associated with translation table walks using TTBR0:

`0b00` Normal memory, Inner Non-cacheable.

`0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Inner Write-Through Cacheable.

`0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.

Resets to 0.

EPD0, [7]

Translation table walk disable for translations using TTBR0. This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using TTBR0:

`0` Perform translation table walks using TTBR0.

`1` A TLB miss on an address that is translated using TTBR0 generates a Translation fault. No translation table walk is performed.

[6:3]

Reserved, RES0.

T0SZ, [2:0]

The size offset of the memory region addressed by TTBR0. The region size is $2^{32-T0SZ}$ bytes.

Resets to 0.

To access the TTBCR:

```
MRC p15,0,<Rt>,c2,c0,0 ; Read TTBR0 into Rt
MCR p15,0,<Rt>,c2,c0,0 ; Write Rt to TTBR0
```

Register access is encoded as follows:

Table B1-93 TTBCR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0010	0000	010

B1.113 Translation Table Base Register 0

The TTBR0 characteristics are:

Purpose

Holds the base address of translation table 0, and information about the memory it occupies. This is one of the translation tables for the stage 1 translation of memory accesses from modes other than Hyp mode.

Usage constraints

This register is accessible as follows:

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

Used in conjunction with the TTBCR. When the 64-bit TTBR0 format is used, cacheability and shareability information is held in the TTBCR and not in TTBR0.

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Attributes

TTBR0 is:

- A 32-bit register when TTBCR.EAE is 0.
- A 64-bit register when TTBCR.EAE is 1.

There are different formats for this register. TTBCR.EAE determines which format of the register is used.

B1.114 TTBR0 with Short-descriptor translation table format

TTBR0 has a specific format when using the Short-descriptor translation table format. TTBCR.EAE determines which format of the register is in use.

The following figure shows the TTBR0 bit assignments when TTBCR.EAE is 0.

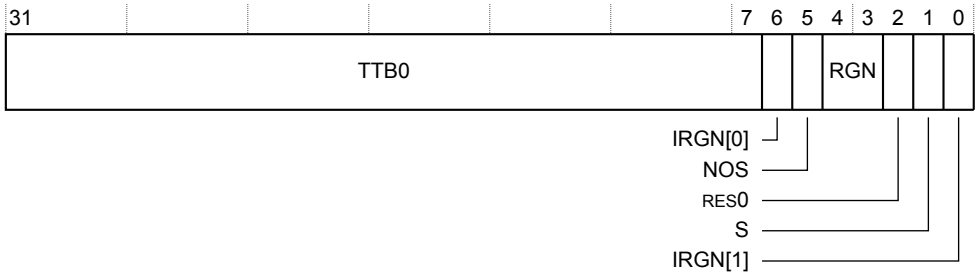

Figure B1-63 TTBR0 bit assignments, TTBCR.EAE is 0

TTB0, [31:7]

Translation table base 0 address, bits[31:x], where x is 14-(TTBCR.N). Bits [x-1:7] are RES0.

The value of x determines the required alignment of the translation table, that must be aligned to 2^x bytes.

If bits [x-1:7] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:7] are treated as if all the bits are zero. The value read back from those bits is the value written.

IRGN[0], [6]

See bit[0] for description of the IRGN field.

NOS, [5]

Not Outer Shareable bit. Indicates the Outer Shareable attribute for the memory associated with a translation table walk that has the Shareable attribute, indicated by TTBR0.S is 1. The possible values are:

`0` Outer Shareable.

`1` Inner Shareable.

This bit is ignored when TTBR0.S is 0.

RGN, [4:3]

Region bits. Indicates the Outer cacheability attributes for the memory associated with the translation table walks. The possible values are:

`0b00` Normal memory, Outer Non-cacheable.

`0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Outer Write-Through Cacheable.

`0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.

[2]

Reserved, RES0.

S, [1]

Shareable bit. Indicates the Shareable attribute for the memory associated with the translation table walks. The possible values are:

`0` Non-shareable.

`1` Shareable.

IRGN[1], [0]

Inner region bits. Indicates the Inner Cacheability attributes for the memory associated with the translation table walks. The possible values of IRGN[1:0] are:

`0b00` Normal memory, Inner Non-cacheable.

`0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Inner Write-Through Cacheable.

`0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.

The encoding of the IRGN bits is counter-intuitive, with register bit[6] being IRGN[0] and register bit[0] being IRGN[1]. This encoding is chosen to give a consistent encoding of memory region types and to ensure that software written for Armv7 without the Multiprocessing Extensions can run unmodified on an implementation that includes the functionality introduced by the Armv7 Multiprocessing Extensions.

To access the TTBR0 when TTBCR.EAE is 0:

```
MRC p15,0,<Rt>,c2,c0,0 ; Read TTBR0 into Rt
MCR p15,0,<Rt>,c2,c0,0 ; Write Rt to TTBR0
```

Register access is encoded as follows:

Table B1-94 TTBR0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0010	0000	000

B1.115 TTBR0 with Long-descriptor translation table format

TTBR0 has a specific format when using the Long-descriptor translation table format. TTBCR.EAE determines which format of the register is in use.

The following figure shows the TTBR0 bit assignments when TTBCR.EAE is 1.

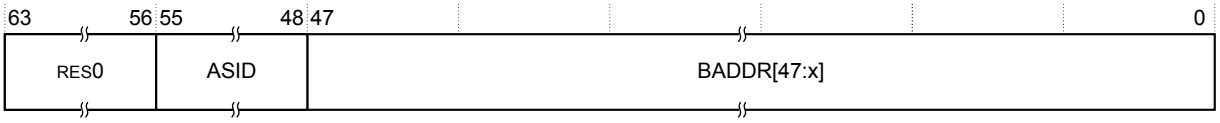

Figure B1-64 TTBR0 bit assignments, TTBCR.EAE is 1

[63:56]

Reserved, RES0.

ASID, [55:48]

An ASID for the translation table base address. The TTBCR.A1 field selects either TTBR0.ASID or TTBR1.ASID.

BADDR[47:x], [47:0]

Translation table base address, bits[47:x]. Bits [x-1:0] are RES0.

x is based on the value of TTBCR.T0SZ, and is calculated as follows:

- If TTBCR.T0SZ is 0 or 1, x = 5 - TTBCR.T0SZ.
- If TTBCR.T0SZ is greater than 1, x = 14 - TTBCR.T0SZ.

The value of x determines the required alignment of the translation table, that must be aligned to 2x bytes.

If bits [x-1:3] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:0] are treated as if all the bits are zero. The value read back from those bits is the value written.

To access the TTBR0 when TTBCR.EAE==1:

```
MRRC p15,0,<Rt>,<Rt2>,c2 ; Read 64-bit TTBR0 into Rt (low word) and Rt2 (high word)
MCRR p15,0,<Rt>,<Rt2>,c2 ; Write Rt (low word) and Rt2 (high word) to 64-bit TTBR0
```

Register access is encoded as follows:

Table B1-95 TTBR0 access encoding

coproc	opc1	CRm
1111	0000	0010

B1.116 Translation Table Base Register 1

The TTBR1 characteristics are:

Purpose

Holds the base address of translation table 1, and information about the memory it occupies. This is one of the translation tables for the stage 1 translation of memory accesses from modes other than Hyp mode.

Usage constraints

This register is accessible as follows:

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

Used in conjunction with the TTBCR. When the 64-bit TTBR1 format is used, cacheability and shareability information is held in the TTBCR and not in TTBR1. See *B1.110 Translation Table Base Control Register* on page B1-333.

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Write access to TTBR1(S) is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

Attributes

TTBR1 is:

- A 32-bit register when TTBCR.EAE is 0.
- A 64-bit register when TTBCR.EAE is 1.

There are two formats for this register. TTBCR.EAE determines which format of the register is used.

B1.117 TTBR1 with Short-descriptor translation table format

TTBR1 has a specific format when using the Short-descriptor translation table format. TTBCR.EAE determines which format of the register is in use.

The following figure shows the TTBR1 bit assignments when TTBCR.EAE is 0.

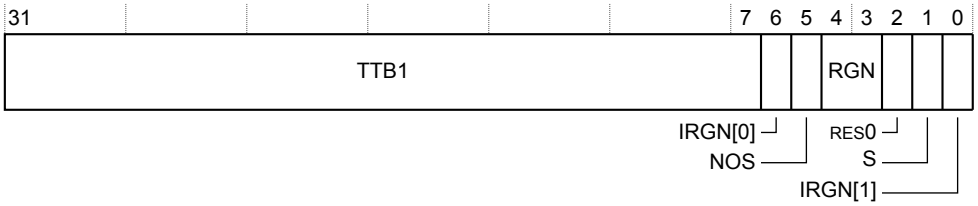

Figure B1-65 TTBR1 bit assignments, TTBCR.EAE is 0

TTB1, [31:7]

Translation table base 1 address, bits[31:x], where x is 14-(TTBCR.N). Bits [x-1:7] are RES0.

The translation table must be aligned on a 16KByte boundary.

If bits [x-1:7] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:7] are treated as if all the bits are zero. The value read back from those bits is the value written.

IRGN[0], [6]

See IRGN[1] below for description of the IRGN field.

NOS, [5]

Not Outer Shareable bit. Indicates the Outer Shareable attribute for the memory associated with a translation table walk that has the Shareable attribute, indicated by TTBR0.S is 1. The possible values are:

`0` Outer Shareable.

`1` Inner Shareable.

This bit is ignored when TTBR0.S is 0.

RGN, [4:3]

Region bits. Indicates the Outer cacheability attributes for the memory associated with the translation table walks. The possible values are:

`0b00` Normal memory, Outer Non-cacheable.

`0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Outer Write-Through Cacheable.

`0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.

IMP, [2]

Reserved, RES0.

S, [1]

Shareable bit. Indicates the Shareable attribute for the memory associated with the translation table walks. The possible values are:

`0` Non-shareable.

`1` Shareable.

IRGN[1], [0]

Inner region bits. Indicates the Inner Cacheability attributes for the memory associated with the translation table walks. The possible values of IRGN[1:0] are:

`0b00` Normal memory, Inner Non-cacheable.

`0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Inner Write-Through Cacheable.

`0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.

The encoding of the IRGN bits is counter-intuitive, with register bit[6] being IRGN[0] and register bit[0] being IRGN[1]. This encoding is chosen to give a consistent encoding of memory region types and to ensure that software written for Armv7 without the Multiprocessing Extensions can run unmodified on an implementation that includes the functionality introduced by the Armv7 Multiprocessing Extensions.

To access the TTBR1 when TTBCR.EAE is 0:

```
MRC p15, 0, <Rt>, c2, c0, 1 ; Read TTBR1 into Rt
MCR p15, 0, <Rt>, c2, c0, 1 ; Write Rt to TTBR1
```

Register access is encoded as follows:

Table B1-96 TTBR1 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0010	0000	001

B1.118 TTBR1 with Long-descriptor translation table format

TTBR1 has a specific format when using the Long-descriptor translation table format. TTBCR.EAE determines which format of the register is in use.

The following figure shows the TTBR1 bit assignments when TTBCR.EAE is 1.

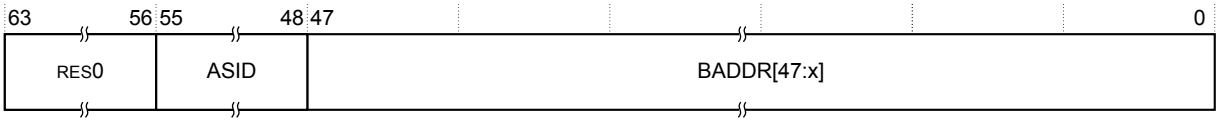

Figure B1-66 TTBR1 bit assignments, TTBCR.EAE is 1

[63:56]

Reserved, RES0.

ASID, [55:48]

An ASID for the translation table base address. The TTBCR.A1 field selects either TTBR0.ASID or TTBR1.ASID.

BADDR[47:x], [47:0]

Translation table base address, bits[47:x]. Bits [x-1:0] are RES0.

x is based on the value of TTBCR.T0SZ, and is calculated as follows:

- If TTBCR.T0SZ is 0 or 1, x = 5 - TTBCR.T0SZ.
- If TTBCR.T0SZ is greater than 1, x = 14 - TTBCR.T0SZ.

The value of x determines the required alignment of the translation table, that must be aligned to 2x bytes.

If bits [x-1:3] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:0] are treated as if all the bits are zero. The value read back from those bits is the value written.

To access the 64-bit TTBR1 when TTBCR.EAE = 1:

```
MRRC p15, 1, <Rt>, <Rt2>, c2 ; Read 64-bit TTBR1 into Rt (low word) and Rt2 (high word)
MCRR p15, 1, <Rt>, <Rt2>, c2 ; Write Rt (low word) and Rt2 (high word) to 64-bit TTBR1
```

Register access is encoded as follows:

Table B1-97 TTBR0 access encoding

coproc	opc1	CRm
1111	0001	0010

B1.119 Vector Base Address Register

The VBAR characteristics are:

Purpose

Holds the exception base address for exceptions that are not taken to Monitor mode or to Hyp mode when high exception vectors are not selected.

Usage constraints

This register is accessible as follows:

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

Software must program the Non-secure instance of the register with the required initial value as part of the processor boot sequence.

Configurations

There are separate Secure and Non-secure instances of this register at EL3.

Write access to VBAR(S) is disabled when the **CP15SDISABLE2** signal is asserted HIGH.

Attributes

VBAR is a 32-bit register.

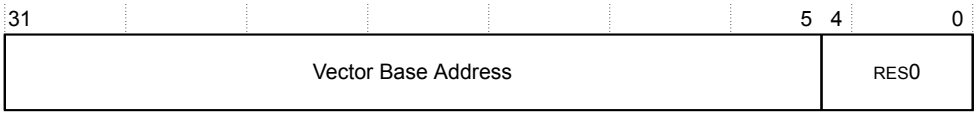

Figure B1-67 VBAR bit assignments

Vector Base Address, [31:5]

Bits[31:5] of the base address of the exception vectors, for exceptions taken in this exception level. Bits[4:0] of an exception vector are the exception offset.

[4:0]

Reserved, RES0.

To access the VBAR:

```
MRC p15, 0, <Rt>, c12, c0, 0 ; Read VBAR into Rt
MCR p15, 0, <Rt>, c12, c0, 0 ; Write Rt to VBAR
```

Register access is encoded as follows:

Table B1-98 VBAR access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	1100	0000	000

B1.120 Virtualization Multiprocessor ID Register

The VMPIDR characteristics are:

Purpose

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

Usage constraints

This register is accessible as follows:

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

Configurations

This register is accessible only at EL2 or EL3.

Attributes

VMPIDR is a 32-bit register.

VMPIDR resets to the value of MPIDR.

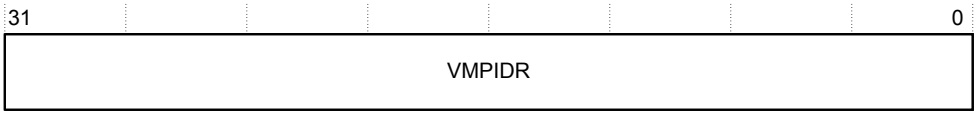

Figure B1-68 VMPIDR bit assignments

VMPIDR, [31:0]

MPIDR value returned by Non-secure EL1 reads of the MPIDR. The MPIDR description defines the subdivision of this value. See *Figure B1-51 MPIDR bit assignments on page B1-309*.

To access the VMPIDR:

```
MRC p15,4,<Rt>,c0,c0,5 ; Read VMPIDR into Rt
MCR p15,4,<Rt>,c0,c0,5 ; Write Rt to VMPIDR
```

Register access is encoded as follows:

Table B1-99 VMPIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0000	0000	101

B1.121 Virtualization Processor ID Register

The VPIDR characteristics are:

Purpose

Holds the value of the Virtualization Processor ID. This is the value returned by Non-secure EL1 reads of MIDR. See *Figure B1-50 MIDR bit assignments* on page B1-307.

Usage constraints

This register is accessible as follows:

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

Configurations

There are no configuration notes.

Attributes

VPIDR is a 32-bit register.

VPIDR resets to the value of MIDR.

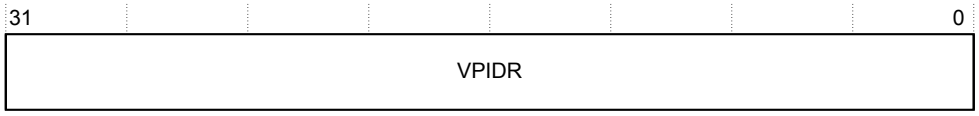

Figure B1-69 VPIDR bit assignments

VPIDR, [31:0]

MIDR value returned by Non-secure PL1 reads of the MIDR. The MIDR description defines the subdivision of this value. See *Figure B1-50 MIDR bit assignments* on page B1-307.

To access the VPIDR:

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

Register access is encoded as follows:

Table B1-100 VPIDR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0000	0000	000

B1.122 Virtualization Translation Control Register

The VTCR characteristics are:

Purpose

Controls the translation table walks required for the stage 2 translation of memory accesses from Non-secure modes other than Hyp mode, and holds cacheability and shareability information for the accesses.

Usage constraints

This register is accessible as follows:

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

Used in conjunction with VTTBR, that defines the translation table base address for the translations.

Configurations

This register is accessible only at EL2 or EL3.

Attributes

VTCR is a 32-bit register.

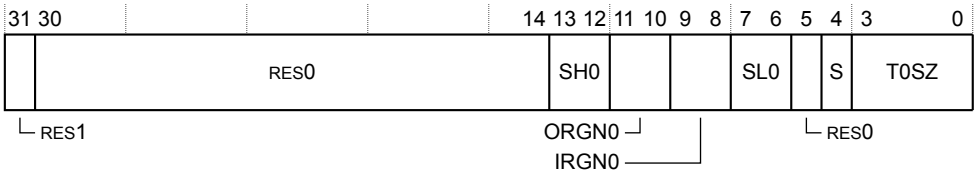

Figure B1-70 VTCR bit assignments

[31]

Reserved, RES1.

[30:14]

Reserved, RES0.

SH0, [13:12]

Shareability attribute for memory associated with translation table walks using TTBR0.

`0b00` Non-shareable.

`0b01` Reserved.

`0b10` Outer Shareable.

`0b11` Inner Shareable.

ORGN0, [11:10]

Outer cacheability attribute for memory associated with translation table walks using TTBR0.

`0b00` Normal memory, Outer Non-cacheable.

`0b01` Normal memory, Outer Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Outer Write-Through Cacheable.

`0b11` Normal memory, Outer Write-Back no Write-Allocate Cacheable.

IRGN0, [9:8]

Inner cacheability attribute for memory associated with translation table walks using TTBR0.

`0b00` Normal memory, Inner Non-cacheable.

`0b01` Normal memory, Inner Write-Back Write-Allocate Cacheable.

`0b10` Normal memory, Inner Write-Through Cacheable.

`0b11` Normal memory, Inner Write-Back no Write-Allocate Cacheable.

SL0, [7:6]

Starting level for translation table walks using VTTBR:

`0b00` Start at second level.

`0b01` Start at first level.

[5]

Reserved, RES0.

S, [4]

Sign extension bit. This bit must be programmed to the value of T0SZ[3]. If it is not, then the stage 2 T0SZ value is treated as an UNKNOWN value within the legal range that can be programmed.

T0SZ, [3:0]

The size offset of the memory region addressed by TTBR0. The region size is $2^{32-T0SZ}$ bytes.

To access the VTCR:

```
MRC p15, 4, <Rt>, c2, c1, 2; Read VTCR into Rt
MCR p15, 4, <Rt>, c2, c1, 2; Write Rt to VTCR
```

Register access is encoded as follows:

Table B1-101 VTCR access encoding

coproc	opc1	CRn	CRm	opc2
1111	100	0010	0001	010

Chapter B2
GIC registers

This chapter describes the GIC registers.

It contains the following sections:

B2.1 CPU interface register summary

Each CPU interface block provides the interface for a Cortex-A32 processor that interfaces with a GIC distributor within the system.

Each CPU interface provides a programming interface for:

- Enabling the signaling of interrupt requests by the CPU interface.
- Acknowledging an interrupt.
- Indicating completion of the processing of an interrupt.
- Setting an interrupt priority mask for the processor.
- Defining the preemption policy for the processor.
- Determining the highest priority pending interrupt for the processor.
- Generating SGIs.

For more information on the CPU interface, see the *Arm® Generic Interrupt Controller Architecture Specification.*

The following table lists the registers for the CPU interface.

All the registers in the following table are word-accessible. Registers not described in this table are RES0. See the *Arm® Generic Interrupt Controller Architecture Specification* for more information.

Table B2-1 CPU interface register summary

Offset	Name	Type	Reset	Description
0x0000	GICC_CTLR	RW	0x00000000	CPU Interface Control Register
0x0004	GICC_PMR	RW	0x00000000	Interrupt Priority Mask Register
0x0008	GICC_BPR	RW	0x00000002 (Secure) 0x00000003 (Non-secure)	Binary Point Register
0x000C	GICC_IAR	RO	-	Interrupt Acknowledge Register
0x0010	GICC_EOIR	WO	-	End Of Interrupt Register
0x0014	GICC_RPR	RO	0x000000FF	Running Priority Register
0x0018	GICC_HPPIR	RO	0x000003FF	Highest Priority Pending Interrupt Register
0x001C	GICC_ABPR	RW	0x00000003	Aliased Binary Point Register
0x0020	GICC_AIAR	RO	-	Aliased Interrupt Acknowledge Register
0x0024	GICC_AEOIR	WO	-	Aliased End of Interrupt Register
0x0028	GICC_AHPPIR	RO	0x000003FF	Aliased Highest Priority Pending Interrupt Register
0x00D0	GICC_APR0	RW	0x00000000	*B2.2 Active Priority Register* on page B2-353
0x00E0	GICC_NSAPR0	RW	0x00000000	Non-secure Active Priority Register
0x00FC	GICC_IIDR	RO	0x0014243B	*B2.3 CPU Interface Identification Register* on page B2-354
0x1000	GICC_DIR	WO	-	Deactivate Interrupt Register

B2.2 Active Priority Register

The GICC_APR0 characteristics are:

Purpose

Provides support for preserving and restoring state in power management applications.

Usage constraints

This register is banked to provide Secure and Non-secure copies. This ensures that Non-secure accesses do not interfere with Secure operation.

Configurations

Available in all configurations.

Attributes

See the register summary in *B2.1 CPU interface register summary* on page B2-352.

The Cortex-A32 processor implements the GICC_APR0 according to the recommendations described in the *Arm Generic Interrupt Controller Architecture Specification*.

Table B2-2 Active Priority Register implementation

Number of group priority bits	Preemption levels	Minimum value of Secure GICC_BPR	Minimum legal value of Non-secure GICC_BPR	Active Priority Registers implemented	View of Active Priority Registers for Non-secure accesses
5	32	2	3	GICC_APR0 [31:0]	GICC_NSAPR0 [31:16] appears as GICC_APRo [15:0]

B2.3 CPU Interface Identification Register

The GICC_IIDR characteristics are:

Purpose

Provides information about the implementer and revision of the CPU interface.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See the register summary in *B2.1 CPU interface register summary* on page B2-352.

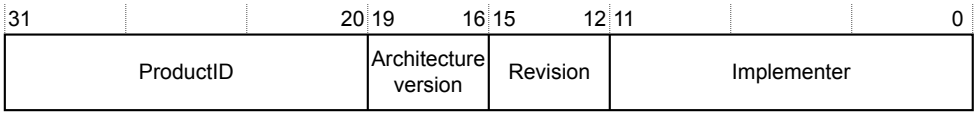

Figure B2-1 GICC_IIDR bit assignments

ProductID, [31:20]

Identifies the product:

`0x001` Cortex-A32 processor.

Architecture version, [19:16]

Identifies the architecture version of the GIC CPU interface:

`0x4` GICv4.

Revision, [15:12]

Identifies the revision number for the CPU interface:

`0x2` r1p0.

Implementer, [11:0]

Contains the JEP106 code of the company that implements the CPU interface:

`0x43B` Arm.

B2.4 Virtual interface control register summary

The virtual interface control registers are management registers. Configuration software on the Cortex-A32 processor must ensure they are accessible only by a hypervisor, or similar software.

The following table describes the registers for the virtual interface control registers. All these registers are word-accessible. Registers not described in this table are RES0. See the *Arm® Generic Interrupt Controller Architecture Specification* for more information.

Table B2-3 Virtual interface control register summary

Offset	Name	Type	Reset	Description
0x000	GICH_HCR	RW	0x00000000	Hypervisor Control Register
0x004	GICH_VTR	RO	0x90000003	*B2.5 VGIC Type Register* on page B2-356
0x008	GICH_VMCR	RW	0x004C0000	Virtual Machine Control Register
0x010	GICH_MISR	RO	0x00000000	Maintenance Interrupt Status Register
0x020	GICH_EISR0	RO	0x00000000	End of Interrupt Status Registers
0x030	GICH_ELRSR0	RO	0x0000000F	Empty List Register Status Registers
0x0F0	GICH_APR0	RW	0x00000000	Active Priorities Register
0x100	GICH_LR0	RW	0x00000000	List Register 0
0x104	GICH_LR1	RW	0x00000000	List Register 1
0x108	GICH_LR2	RW	0x00000000	List Register 2
0x10C	GICH_LR3	RW	0x00000000	List Register 3

B2.5 VGIC Type Register

The GICH_VTR characteristics are:

Purpose

Holds information on number of priority bits, number of preemption bits, and number of List Registers implemented.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See the register summary in *B2.4 Virtual interface control register summary* on page B2-355.

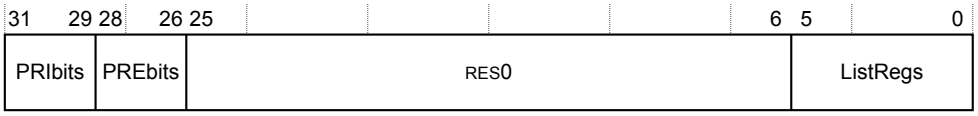

Figure B2-2 GICH_VTR bit assignments

PRIbits, [31:29]

Indicates the number of priority bits implemented, minus one:

`0x4` Five bits of priority and 32 priority levels.

PREbits, [28:26]

Indicates the number of preemption bits implemented, minus one:

`0x4` Five bits of preemption and 32 preemption levels.

[25:6]

Reserved, RES0.

ListRegs, [5:0]

Indicates the number of implemented List Registers, minus one:

`0x3` Four List Registers.

B2.6 Virtual CPU interface register summary

The virtual CPU interface forwards virtual interrupts to a connected Cortex-A32 processor, subject to the normal GIC handling and prioritization rules.

The virtual interface control registers control virtual CPU interface operations. In particular, the virtual CPU interface uses the contents of the List registers to determine when to signal virtual interrupts. When a core accesses the virtual CPU interface, the List registers are updated. For more information on the virtual CPU interface, see the *Arm® Generic Interrupt Controller Architecture Specification*.

The following table describes the registers for the virtual CPU interface.

All the registers in the following table are word-accessible. Registers not described in this table are RES0. See the *Arm® Generic Interrupt Controller Architecture Specification* for more information.

Table B2-4 Virtual CPU interface register summary

Name	Type	Reset	Description
GICV_CTLR	RW	0x00000000	VM Control Register
GICV_PMR	RW	0x00000000	VM Priority Mask Register
GICV_BPR	RW	0x00000002	VM Binary Point Register
GICV_IAR	RO	-	VM Interrupt Acknowledge Register
GICV_EOIR	WO	-	VM End Of Interrupt Register
GICV_RPR	RO	0x000000FF	VM Running Priority Register
GICV_HPPIR	RO	0x000003FF	VM Highest Priority Pending Interrupt Register
GICV_ABPR	RW	0x00000003	VM Aliased Binary Point Register
GICV_AIAR	RO	-	VM Aliased Interrupt Acknowledge Register
GICV_AEOIR	WO	-	VM Aliased End of Interrupt Register
GICV_AHPPIR	RO	0x000003FF	VM Aliased Highest Priority Pending Interrupt Register
GICV_APR0	RW	0x00000000	*B2.7 VM Active Priority Register* on page B2-358
GICV_IIDR	RO	0x0014243B	*B2.8 VM CPU Interface Identification Register* on page B2-359
GICV_DIR	WO	-	VM Deactivate Interrupt Register

B2.7 VM Active Priority Register

GICV_APR0

For software compatibility, this register is present in the virtual CPU interface. However, in a virtualized system, it is not used when preserving and restoring state.

The GICV_APR0 characteristics are:

Purpose

For software compatibility, this register is present in the virtual CPU interface. However, in a virtualized system, it is not used when preserving and restoring state.

Usage constraints

Reading the content of this register and then writing the same values must not change any state because there is no requirement to preserve and restore state during a powerdown.

Configurations

Available in all configurations.

Attributes

See the register summary in *B2.6 Virtual CPU interface register summary* on page B2-357.

The Cortex-A32 processor implements the GICV_APR0 as an alias of GICH_APR0.

B2.8 VM CPU Interface Identification Register

The GICV_IIDR characteristics are:

Purpose

Provides information about the implementer and revision of the virtual CPU interface.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See the register summary in *B2.6 Virtual CPU interface register summary* on page B2-357.

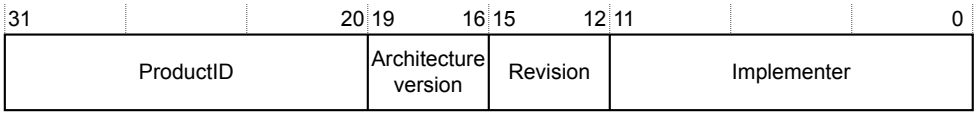

Figure B2-3 GICV_IIDR bit assignments

ProductID, [31:20]

Identifies the product:

`0x001` Cortex-A32 processor.

Architecture version, [19:16]

Identifies the architecture version of the GIC CPU Interface:

`0x4` GICv4.

Revision, [15:12]

Identifies the revision number for the CPU interface:

`0x2` r1p0.

Implementer, [11:0]

Contains the JEP106 code of the company that implements the CPU interface:

`0x43B` Arm.

Chapter B3
Generic Timer registers

This chapter describes the Generic Timer registers.

It contains the following section:

B3.1 AArch32 Generic Timer register summary

The following table shows the AArch32 Generic Timer registers.

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

Table B3-1 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	-	32-bit	Counter-timer Kernel Control register The reset value for bits[9:8, 2:0] is `0b00000`.
CNTP_TVAL			c2	0	UNK	32-bit	Counter-timer Physical Timer TimerValue register
CNTP_CTL				1	-	32-bit	Counter-timer Physical Timer Control register The reset value for bit[0] is 0.
CNTV_TVAL			c3	0	UNK	32-bit	Counter-timer Virtual Timer TimerValue register
CNTV_CTL				1		32-bit	Counter-timer Virtual Timer Control register The reset value for bit[0] is 0.
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	-	32-bit	Counter-timer Hyp Control register The reset value for bit[2] is 0 and for bits[1:0] is `0b11`.
CNTHP_TVAL			c2	0	UNK	32-bit	Counter-timer Hyp Physical Timer TimerValue register
CNTHP_CTL				1		32-bit	Counter-timer Hyp Physical Timer Control register The reset value for bit[0] is 0.
CNTHP_CVAL	-	6	c14	-	UNK	64-bit	Counter-timer Hyp Physical CompareValue register

Part C
Debug

Chapter C1
Debug

This chapter describes the debug features of the processor.

It contains the following sections:

C1.1 About debug methods

The processor is part of a debug system and supports both self-hosted and external debug.

The following figure shows a typical external debug system.

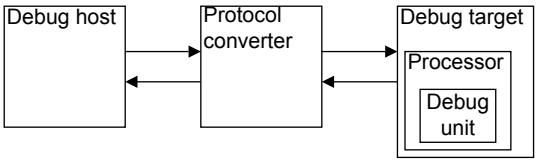

Figure C1-1 External debug system

Debug host

A computer, for example a personal computer, that is running a software debugger such as the DS-5 Debugger. With the debug host, you can issue high-level commands, such as setting a breakpoint at a certain location or examining the contents of a memory address.

Protocol converter

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

Debug target

The lowest level of the system implements system support for the protocol converter to access the debug unit using the *Advanced Peripheral Bus* (APB) slave interface. An example of a debug target is a development system with a test chip or a silicon part with a processor.

Debug unit

Helps debugging software that is running on the processor:

- Hardware systems that are based on the processor.
- Operating systems.
- Application software.

With the debug unit, you can:

- Stop program execution.
- Examine and alter process and coprocessor state.
- Examine and alter memory and the state of the input or output peripherals.
- Restart the processor.

For self-hosted debug, the debug target runs additional debug monitor software that runs on the Cortex-A32 processor itself. This way, it does not require expensive interface hardware to connect a second host computer.

Related information

C1.2 Debug access on page C1-367

C1.2 Debug access

The processor implements the Armv8 Debug architecture and debug events. Accessing system registers allows the processor to access certain debug registers directly. The external debug interface enables both external and self-hosted debug agents to access debug registers.

The partitioning of debug register access is as follows:

Debug registers
Access is based on the system registers and is memory-mapped. You can access the debug register map using the APB slave port.

PMU
Access is based on the system registers and is memory-mapped. You can access the performance monitor registers using the APB slave port.

ETM
Access is memory-mapped.

CTI
Access is based on the debug registers and is memory-mapped.

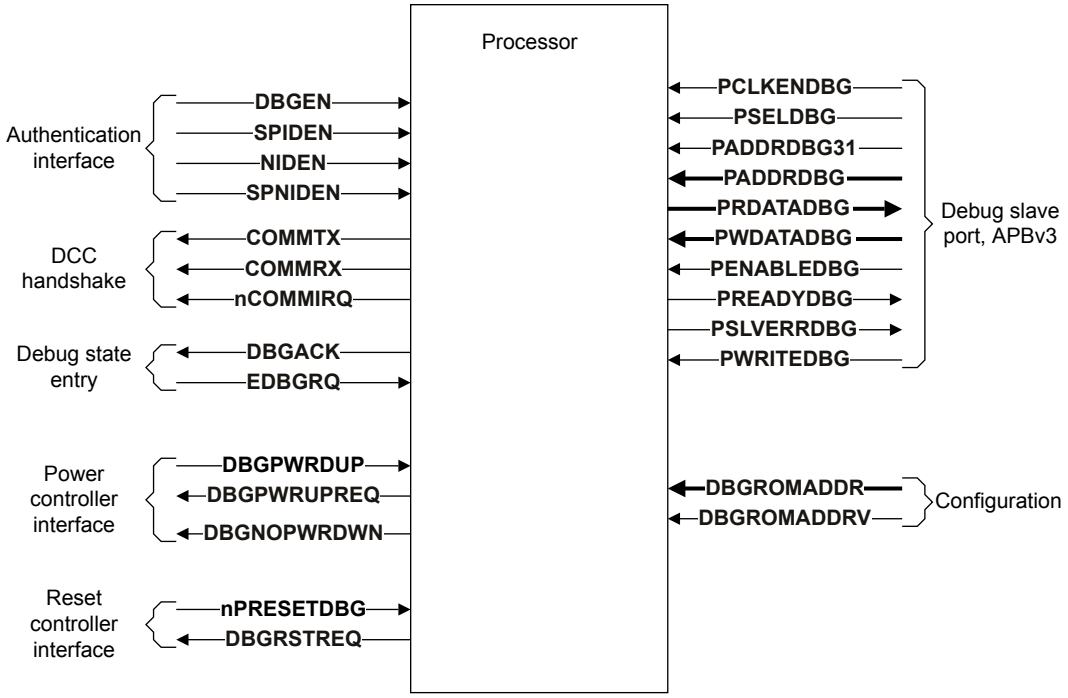

Figure C1-2 External debug interface

Related information

Chapter C4 CTI on page C4-393
Appendix A Signal Descriptions on page Appx-A-617

C1.3 Effects of resets on debug registers

Some of the processor reset signals affect the debug registers.

nCPUPORESET

Cold reset that covers reset of the processor logic and the integrated debug functionality.

The signal initializes the processor logic, including the debug, *Embedded Trace Macrocell* (ETM) trace unit, breakpoint, watchpoint logic, and performance monitors logic.

nCORERESET

Warm reset that covers reset of the processor logic.

The signal resets some of the debug and performance monitor logic.

nPRESETDBG

External debug reset that covers the resetting of the external debug interface and has no impact on the processor functionality.

The signal initializes the shared debug APB, *Cross Trigger Interface* (CTI), and *Cross Trigger Matrix* (CTM) logic.

Related information

A3.3 Resets on page A3-48
Appendix A Signal Descriptions on page Appx-A-617

C1.4 External access permissions to debug registers

External access permission to the debug registers is subject to the conditions at the time of the access.

The following table describes the processor response to accesses to the debug registers.

Table C1-1 Conditions on external register access to debug registers

Name	Condition	Description
Off	EDPRSR.PU is 0	The processor power domain is completely off or in a low-power state in which the processor power domain registers cannot be accessed. If debug power is off, then all external debug and memory-mapped register accesses return an error.
DLK	EDPRSR.DLK is 1	OS Double Lock is locked.
OSLK	OSLSR_EL1.OSLK is 1	OS Lock is locked.
EDAD	`AllowExternalDebugAccess() == FALSE`	External debug access is disabled. When an error is returned because of an EDAD condition code, the highest priority error condition, EDPRSR.SDAD is set to 1. Otherwise SDAD is unchanged.
SLK	Memory-mapped interface only	Software lock is locked. For the external debug interface, ignore this column.
Default	-	None of the conditions apply, normal access.

The following table shows an example of external register condition codes for access to a debug register. To determine the access permission for the register, scan the columns from left to right. Stop at the first column a condition is true, the entry gives the access permission of the register and scanning stops.

Table C1-2 Code example for the conditions on external register access to debug registers

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	RO/WI	RO

C1.5 Debug events

A debug event can be either a software debug event or a halting debug event. A core responds to a debug event in one of the following ways: ignores it, takes a debug exception, or enters debug state.

In the processor, watchpoint debug events are always synchronous. Memory hint instructions and cache clean operations, except DC ZVA, DC IVAC, and DCIMVAC, do not generate watchpoint debug events. Store exclusive instructions generate a watchpoint debug event even when the check for the control of exclusive monitor fails. For watchpoint debug events, except those resulting from cache maintenance operations, the value reported in DFAR is guaranteed to be no lower than the address of the watchpointed location rounded down to a multiple of 16 bytes.

The powerup reset signal, **nCPUPORESET**, sets the Debug OS Lock. For the debug events and debug register accesses to operate normally, the Debug OS Lock must be cleared.

Related information

A3.3 Resets on page A3-48

A.4 Reset signals on page Appx-A-621

Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile

C1.6 Debug memory map

The basic memory map supports up to four cores in the cluster. You can configure the mapping as a v8 or as a v7 Debug memory map.

The following table shows the address mapping for the debug APB components when you configure them for a v8 Debug memory map. Each component in the table requires 4KB and uses the bottom 4KB of each 64KB region. The remaining 60KB of each region is reserved. The table indicates the mapped component if it is present, otherwise the field is reserved.

Table C1-3 Address mapping for APB components in a v8 Debug memory map

Address offset [21:0]	Mapped to
0x000000 - 0x000FFF	APB ROM table for the processor
0x010000 - 0x010FFF	Core 0 Debug
0x020000 - 0x020FFF	Core 0 CTI
0x030000 - 0x030FFF	Core 0 PMU
0x040000 - 0x040FFF	Core 0 Trace
0x041000 - 0x10FFFF	Reserved
0x110000 - 0x110FFF	Core 1 Debug
0x120000 - 0x120FFF	Core 1 CTI
0x130000 - 0x130FFF	Core 1 PMU
0x140000 - 0x140FFF	Core 1 Trace
0x141000 - 0x20FFFF	Reserved
0x210000 - 0x210FFF	Core 2 Debug
0x220000 - 0x220FFF	Core 2 CTI
0x230000 - 0x230FFF	Core 2 PMU
0x240000 - 0x240FFF	Core 2 Trace
0x241000 - 0x30FFFF	Reserved
0x310000 - 0x310FFF	Core 3 Debug
0x320000 - 0x320FFF	Core 3 CTI
0x330000 - 0x330FFF	Core 3 PMU
0x340000 - 0x340FFF	Core 3 Trace
0x341000 - 0x3FFFFF	Reserved

The following table shows the address mapping for the debug components when you configure them for a v7 Debug memory map. The table indicates the mapped component if it is present, otherwise the field is reserved.

Table C1-4 Address mapping for APB components in a v7 Debug memory map

Address offset [21:0]	Mapped to
0x00000 - 0x00FFF	APB ROM table for the processor
0x01000 - 0x07FFF	Reserved for other debug components
0x08000 - 0x0FFFF	Reserved for future expansion
0x10000 - 0x10FFF	Core 0 Debug
0x11000 - 0x11FFF	Core 0 PMU
0x12000 - 0x12FFF	Core 1 Debug
0x13000 - 0x13FFF	Core 1 PMU
0x14000 - 0x14FFF	Core 2 Debug
0x15000 - 0x15FFF	Core 2 PMU
0x16000 - 0x16FFF	Core 3 Debug
0x17000 - 0x17FFF	Core 3 PMU
0x18000 - 0x18FFF	Core 0 CTI
0x19000 - 0x19FFF	Core 1 CTI
0x1A000 - 0x1AFFF	Core 2 CTI
0x1B000 - 0x1BFFF	Core 3 CTI
0x1C000 - 0x1CFFF	Core 0 Trace
0x1D000 - 0x1DFFF	Core 1 Trace
0x1E000 - 0x1EFFF	Core 2 Trace
0x1F000 - 0x1FFFF	Core 3 Trace

C1.7 Debug signals

The **DBGPWRDUP** and **DBGL1RSTDISABLE** signals are subject to particular rules.

You must set the **DBGPWRDUP** signal LOW before removing power to the processor domain. After power is restored to the processor domain, the **DBGPWRDUP** signal must be asserted HIGH. The EDPRSR.PU bit reflects the value of this **DBGPWRDUP** signal. **DBGPWRDUP** must be tied HIGH if the particular implementation does not support separate processor and SCU power domains.

When you set it HIGH, the **DBGL1RSTDISABLE** input signal disables the automatic, hardware-controlled invalidation with **nCORERESET** or **nCPUPORESET** of the L1 data cache after the processor is reset. You must use **DBGL1RSTDISABLE** only to debug a reset that an external watchdog triggered. This signal makes the contents of the L1 data cache from before the reset observable after the reset. If reset is asserted while an L1 data cache eviction or L1 data cache fetch is performed, the accuracy of those cache entries is not guaranteed. You must not use the **DBGL1RSTDISABLE** signal to disable the automatic, hardware controlled invalidation of the L1 data cache in normal processor powerup sequences. This is because there is no guarantee that the L1 data cache invalidation sequence is synchronized to the duplicate L1 tags in the SCU. The **DBGL1RSTDISABLE** signal applies to all cores in the cluster. Each core samples the signal when **nCORERESET** or **nCPUPORESET** is asserted. If the functionality offered by the **DBGL1RSTDISABLE** input signal is not required, the input must be tied to LOW.

Related information

A.14 Debug signals on page Appx-A-641

C1.8 Changing the authentication signals for debug

The **NIDEN**, **DBGEN**, **SPIDEN**, and **SPNIDEN** input signals are either tied off to some fixed value or controlled by some external device. If software running on the processor has control over an external device that drives the authentication signals, it must change the signal value using the specified procedure.

Procedure

1. Execute an implementation-specific sequence of instructions to change the signal value. For example, a single `STR` instruction might write certain values to a control register in a system peripheral.
2. If step *1* on page C1-374 involves any memory operation, issue a `DSB` instruction.
3. Issue an `ISB` instruction or exception entry or exception return.
4. Poll the DBGAUTHSTATUS_EL1 register to check whether the processor has already detected the changed value of these signals. This check is required because the system might not issue the signal change to the processor until several cycles after the `DSB` instruction completes.

Next Steps

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. You can determine the relevant combinations of the **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN** values by polling DBGAUTHSTATUS_EL1.

Related information

A.14 Debug signals on page Appx-A-641

Chapter C2
PMU

This chapter describes the *Performance Monitor Unit* (PMU) of the processor.

It contains the following sections:

C2.1 About the PMU

The processor includes performance monitors that enable you to gather various statistics on the operation of the processor and its memory system during runtime. They provide useful information that you can use when debugging or profiling code.

The PMU provides six counters. Each counter can count any of the events available in the processor. The absolute counts that the PMU records might vary because of pipeline effects. This variability only has an impact on the operation of the PMU when a counter is enabled for a short time.

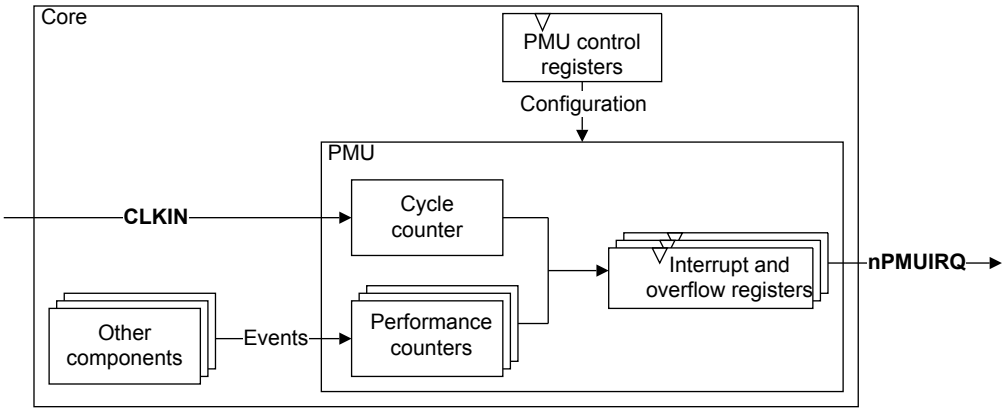

Figure C2-1 PMU block diagram

Event interface

Events from all other units from across the design are provided to the PMU.

System register and APB interface

You can program the PMU registers using the system registers or external APB interface.

Counters

The PMU has six 32-bit performance counters and one 64-bit cycle counter. The performance counters increment when they are enabled based on events.

PMU register interfaces

The processor supports access to the performance monitor registers from the internal system register interface or external debug interface.

Related information

C2.3 Performance monitoring events on page C2-378
C4.2 Cross-trigger inputs and outputs on page C4-395

C2.2 External register access permissions to the PMU registers

External access permission to the PMU registers is subject to conditions at the time of the access.

The following table describes the processor response to accesses through the external debug interface.

Table C2-1 Conditions on external register access to PMU

Name	Condition	Description
Off	EDPRSR.PU is 0	Processor power domain is completely off or in a low-power state where the processor power domain registers cannot be accessed.
DLK	EDPRSR.DLK is 1	OS Double Lock is locked.
OSLK	OSLSR_EL1.OSLK is 1	OS Lock is locked.
EPMAD	`AllowExternalPMUAccess() == FALSE`	External performance monitors access is disabled. When an error is returned because of an EPMAD condition code, and it is the highest priority error condition, EDPRSR.SPMAD is set to 1. Otherwise SPMAD is unchanged.
SLK	Memory-mapped interface only	Software lock is locked. For the external debug interface, ignore this column.
Default	-	None of the conditions apply, normal access.

To determine the access permission for the register, scan the columns from left to right in the register usage constraints table. An example is shown in *Table C2-2 Example for external register condition code* on page C2-377 . Stop at the first column in which the condition is true. The entry gives the register access permission and scanning stops.

Table C2-2 Example for external register condition code

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO/WI	RO

C2.3 Performance monitoring events

The PMU monitors events in the processor and uses reference numbers for significant ones.

The following table shows the bit position of each event on the event bus. Event reference numbers that are not listed are reserved.

Table C2-3 Performance monitoring events

Number	Event mnemonic	PMU event bus to external	PMU event bus to trace	Event name
0x00	SW_INCR	-	-	Software increment. The register is incremented only on writes to the Software Increment Register.
0x01	L1I_CACHE_REFILL	[0]	[0]	L1 Instruction cache refill.
0x02	L1I_TLB_REFILL	[1]	[1]	L1 Instruction TLB refill.
0x03	L1D_CACHE_REFILL	[2]	[2]	L1 Data cache refill.
0x04	L1D_CACHE	[3]	[3]	L1 Data cache access.
0x05	L1D_TLB_REFILL	[4]	[4]	L1 Data TLB refill.
0x06	LD_RETIRED	[5]	[5]	Instruction that is architecturally executed, condition check pass - load.
0x07	ST_RETIRED	[6]	[6]	Instruction that is architecturally executed, condition check pass - store.
0x08	INST_RETIRED	[7]	[7]	Instruction that is architecturally executed.
-	-	[8]	[8]	Two instructions are architecturally executed. Counts every cycle in which two instructions are architecturally retired. Event 0x08, INST_RETIRED, always counts when this event counts.
0x09	EXC_TAKEN	[9]	[9]	Exception taken.
0x0A	EXC_RETURN	[10]	[10]	Exception return.
0x0B	CID_WRITE_RETIRED	[11]	[11]	Change to Context ID retired.
0x0C	PC_WRITE_RETIRED	[12]	[12]	Instruction that is architecturally executed, condition check pass, software change of the PC.
0x0D	BR_IMMED_RETIRED	[13]	[13]	Instruction that is architecturally executed, immediate branch.
0x0E	BR_RETURN_RETIRED	-	-	Instruction that is architecturally executed, condition code check pass, procedure return.
0x0F	UNALIGNED_LDST_RETIRED	[14]	[14]	Instruction that is architecturally executed, condition check pass, unaligned load or store.
0x10	BR_MIS_PRED	[15]	[15]	Mispredicted or not predicted branch that is speculatively executed.
0x11	CPU_CYCLES	-	-	Cycle.
0x12	BR_PRED	[16]	[16]	Predictable branch that is speculatively executed.
0x13	MEM_ACCESS	[17]	[17]	Data memory access.

Table C2-3 Performance monitoring events (continued)

Number	Event mnemonic	PMU event bus to external	PMU event bus to trace	Event name
0x14	L1I_CACHE	[18]	[18]	L1 Instruction cache access.
0x15	L1D_CACHE_WB	[19]	[19]	L1 Data cache writeback.
0x16	L2D_CACHE	[20]	[20]	L2 Data cache access.
0x17	L2D_CACHE_REFILL	[21]	[21]	L2 Data cache refill.
0x18	L2D_CACHE_WB	[22]	[22]	L2 Data cache write-back.
0x19	BUS_ACCESS	-	-	Bus access.
0x1A	MEMORY_ERROR	-	-	Local memory error.
0x1D	BUS_CYCLES	-	-	Bus cycle.
0x1E	CHAIN	-	-	Odd performance counter chain mode.
0x60	BUS_ACCESS_LD	-	-	Bus access - Read.
0x61	BUS_ACCESS_ST	-	-	Bus access - Write.
0x7A	BR_INDIRECT_SPEC	-	-	Branch that is speculatively executed - Indirect branch.
0x86	EXC_IRQ	-	-	Exception taken, IRQ.
0x87	EXC_FIQ	-	-	Exception taken, FIQ.
0xC0	-	-	-	External memory request.
0xC1	-	-	-	Non-cacheable external memory request.
0xC2	-	-	-	Linefill because of prefetch.
0xC4	-	-	-	Entering read allocate mode.
0xC5	-	-	-	Read allocate mode.
0xC6	-	-	-	Pre-decode error.
0xC7	-	-	-	Data Write operation that stalls the pipeline because the store buffer is full.
0xC8	-	-	-	SCU Snooped data from another core for this core.
0xC9	-	-	-	Conditional branch that is executed.
0xCA	-	-	-	Indirect branch that is mispredicted.
0xCB	-	-	-	Indirect branch that is mispredicted because of address miscompare.
0xCC	-	-	-	Conditional branch that is mispredicted.
0xD0	-	[23]	[23]	L1 Instruction Cache (data or tag) memory error.
0xD1	-	[24]	[24]	L1 Data Cache (data, tag, or dirty) memory error, correctable or non-correctable.
0xD2	-	[25]	[25]	TLB memory error.

Table C2-3 Performance monitoring events (continued)

Number	Event mnemonic	PMU event bus to external	PMU event bus to trace	Event name
0xE0	-	-	-	Attributable Performance Impact Event. Counts every cycle that the DPU IQ is empty and that is not because of a recent micro-TLB miss, an instruction cache miss or a pre-decode error.
0xE1	-	-	-	Attributable Performance Impact Event. Counts every cycle the DPU IQ is empty and there is an instruction cache miss being processed.
0xE2	-	-	-	Attributable Performance Impact Event. Counts every cycle the DPU IQ is empty and there is an instruction micro-TLB miss being processed.
0xE3	-	-	-	Attributable Performance Impact Event. Counts every cycle the DPU IQ is empty and there is a pre-decode error being processed.
0xE4	-	-	-	Attributable Performance Impact Event. 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.
0xE5	-	-	-	Attributable Performance Impact Event. 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.
0xE6	-	-	-	Attributable Performance Impact Event. 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 awaiting load data, are excluded.
0xE7	-	-	-	Attributable Performance Impact Event Counts every cycle there is a stall in the Wr stage because of a load miss.
0xE8	-	-	-	Attributable Performance Impact Event. Counts every cycle there is a stall in the Wr stage because of a store.

Table C2-3 Performance monitoring events (continued)

Number	Event mnemonic	PMU event bus to external	PMU event bus to trace	Event name
-	-	[26]	[26]	L2 (data or tag) memory error, correctable or non-correctable.
-	-	[27]	[27]	SCU snoop filter memory error, correctable or non-correctable.
-	-	[28]	-	Advanced SIMD and floating-point retention active.
-	-	[29]	-	Core retention active.

C2.4 PMU interrupts

The processor asserts the **nPMUIRQ** signal when an interrupt occurs that the PMU generated.

You can route this signal to an external interrupt controller for prioritization and masking. It is the only mechanism that signals this interrupt to the processor.

This interrupt is also driven as a trigger input to the CTI.

Related information

C4.2 Cross-trigger inputs and outputs on page C4-395

C2.5 Exporting PMU events

Some of the PMU events are exported to the ETM trace unit and the *Cross Trigger Interface* (CTI) to enable them to be monitored. Furthermore, the processor exports some PMU events on the **PMUEVENT** bus to external hardware.

Related information

Chapter C3 ETM on page C3-385
Chapter C4 CTI on page C4-393

Chapter C3
ETM

This chapter describes the *Embedded Trace Macrocell* (ETM) of the processor.

It contains the following sections:

C3.1 About the ETM

The ETM trace unit is a build-time configuration option. This module performs real-time instruction flow tracing that complies with the ETM architecture. As a CoreSight component, it is part of the Arm real-time debug solution.

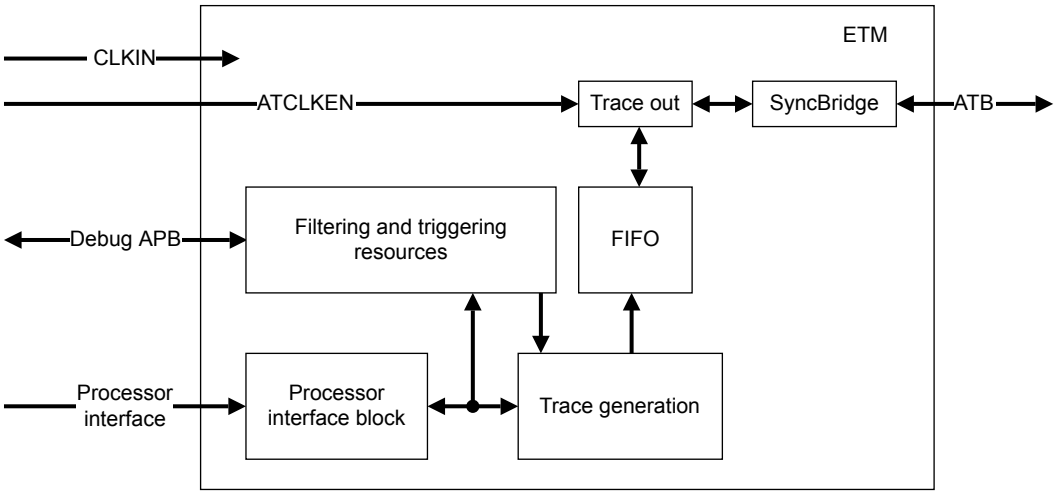

Figure C3-1 ETM functional blocks

Filtering and triggering resources

You can limit the amount of trace data that the ETM generates through filtering. For example, you can configure the ETM to generate trace only in a certain address range. More complicated filtering options, in the style of a logic analyzer, are also available.

The ETM trace unit can also generate a trigger signal to the Trace Capture Device to stop capturing trace.

Processor interface

Monitors the behavior of the processor and generates P0 elements that are executed instructions and exceptions that the ETM traces in program order.

Trace generation

Generates various trace packets that are based on P0 elements.

FIFO

The ETM generates trace in a highly compressed form. The FIFO can flatten trace bursts. When it becomes full, it signals overflow so that the trace generation logic does not generate any new trace until the FIFO becomes empty. The period without trace generation results in a gap in the trace in the debugger view.

Trace out

Trace from FIFO is output on the synchronous ATB interface.

Syncbridge

The ATB interface from the trace out block goes through an ATB synchronous bridge.

Interaction with the *Performance Monitoring Unit* (PMU)

The processor includes a PMU that enables counting events, such as cache misses, over a period. All PMU architectural events are available to the ETM trace unit through the extended input facility. The ETM trace unit uses four extended external input selectors to access the PMU events. Each selector can independently select one of the PMU events, which is then active for the cycles where the relevant events occur. Any ETM event register can access the selected event.

The processor supports only a memory-mapped interface to trace registers.

All trace register accesses through the external debug interface behave as if the processor power domain is powered down when debug double lock is set.

Related information

Arm® CoreSight Architecture Specification

Chapter C2 PMU on page C2-375

C3.2 Configuration options for the ETM unit and trace resources

The ETM unit is configurable. The processor implements options for the ETM unit and its resources.

Table C3-1 Configuration of trace generation

Description	Configuration
Instruction address size in bytes	8
Data address size in bytes	0
Data value size in bytes	0
Virtual Machine ID size in bytes	1
Context ID size in bytes	4
Support for conditional instruction tracing	Not implemented
Support for tracing of data	Not implemented
Support for tracing of load and store instructions as P0 elements	Not implemented
Support for cycle counting in the instruction trace	Implemented
Support for branch broadcast tracing	Implemented
Exception levels implemented in Non-secure state	EL2, EL1, EL0
Exception levels implemented in Secure state	EL3, EL1, EL0
Number of events supported in the trace	4
Return stack support	Implemented
Tracing of SError exception support	Implemented
Instruction trace cycle counting minimum threshold	1
Size of Trace ID	7 bits
Synchronization period support	Read-write
Global timestamp size	64 bits
Number of cores available for tracing	1
ATB trigger support	Implemented
Low-power behavior override	Implemented
Stall control support	Implemented
Support for no overflows in the trace	Not implemented

Table C3-2 Configuration of trace resources

Description	Configuration
Number of resource selection pairs implemented	8
Number of external input selectors implemented	4
Number of external inputs implemented	30, 4 CTI + 26 PMU
Number of counters implemented	2
Reduced function counter implemented	Not implemented

Table C3-2 Configuration of trace resources (continued)

Description	Configuration
Number of sequencer states implemented	4
Number of Virtual Machine ID comparators implemented	1
Number of Context ID comparators implemented	1
Number of address comparator pairs implemented	4
Number of single-shot comparator controls	1
Number of processor comparator inputs implemented	0
Data address comparisons implemented	Not implemented
Number of data value comparators implemented	0

C3.3 Resetting the ETM

The reset for the ETM trace unit is the same as a cold reset for the processor. If the ETM trace unit is reset, tracing stops until the ETM trace unit is reprogrammed and re-enabled.

A warm reset of the processor does not reset the ETM trace unit. Therefore it is possible to trace through warm processor reset. However, if the processor is reset using warm reset, the trace unit might not be able to trace the last few instructions before the reset.

C3.4 Programming and reading ETM trace unit registers

You program and read the ETM trace unit registers using the Debug APB interface.

The processor does not have to be in the debug state while you program the ETM trace unit registers.

When you are programming the ETM trace unit registers, you must enable all the changes at the same time. Otherwise if you reprogram the counter, it might start to count based on incorrect events before the correct setup is in place for the trigger condition.

To disable the ETM trace unit, use the TRCPRGCTLR.EN bit.

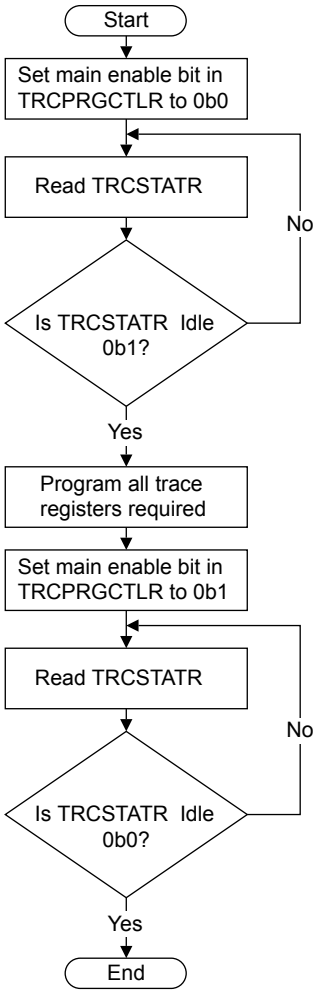

Figure C3-2 Programming ETM trace unit registers

Related information

C10.2 Programming Control Register on page C10-506

C10.3 Status Register on page C10-507

Chapter C4
CTI

This chapter describes the cross-trigger components of the processor.

It contains the following sections:

C4.1 About the cross-trigger

The *Cross-Trigger Interface* (CTI) enables the debug logic, ETM trace unit, and PMU to interact with each other and with other CoreSight components. For example, you configure the CTI to generate an interrupt when the ETM trace unit trigger event occurs.

The single external cross-trigger channel interface of the processor connects to the CTI of each core through a *Cross Trigger Matrix* (CTM). Trigger inputs and trigger outputs connect debug components in the processor and CoreSight CTI blocks. The external CTM output is driven by the OR of the internal CTI outputs. Each internal CTI input is driven by the OR of the other internal CTI outputs and the external CTM input.

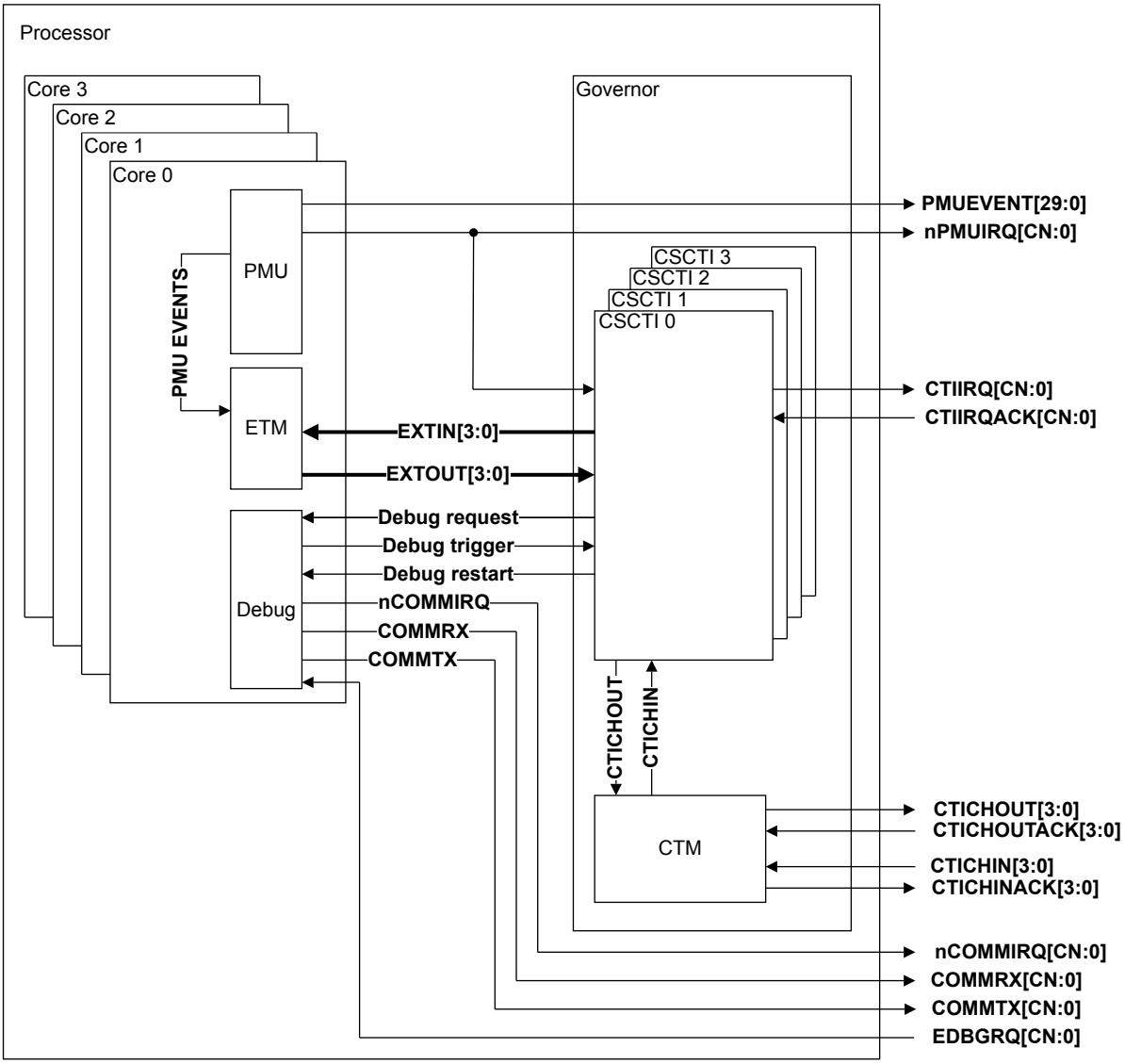

Figure C4-1 Cross-trigger components

C4.2 Cross-trigger inputs and outputs

Outputs from the PMU, the ETM, and the debug subsystem are cross-trigger inputs to the processor CTI. The CTI then generates cross-trigger outputs to other components in the SoC.

Table C4-1 Cross-trigger inputs

CTI input	Name	Description
0	**DBGTRIGGER**, pulsed	Pulsed on entry to debug state.
1	**PMUIRQ**	PMU-generated interrupt. This signal is the same as **nPMUIRQ** with inverted polarity.
2	-	-
3	-	-
4	**EXTOUT[0]**	Output from the ETM unit of Core<N>.
5	**EXTOUT[1]**	
6	**EXTOUT[2]**	
7	**EXTOUT[3]**	

Table C4-2 Cross-trigger outputs

CTI output	Name	Description
0	**EDBGRQ**	Causes the processor to enter debug state
1	**DBGRESTART**	Causes the processor to exit debug state
2	**CTIIRQ**	CTI interrupt
3	-	-
4	**EXTIN[0]**	ETM trace unit external input
5	**EXTIN[1]**	ETM trace unit external input
6	**EXTIN[2]**	ETM trace unit external input
7	**EXTIN[3]**	ETM trace unit external input

Chapter C5
Direct access to internal memory

This chapter describes the direct access to internal memory that caches and TLBs use.

It contains the following sections:

C5.1 About direct access to internal memory

System registers provide access to the internal memory that the caches and TLBs use. This functionality can be useful when investigating issues where the coherency between the data in the cache and data in system memory is broken.

The appropriate memory block and location are selected using write-only CP15 registers and the data is read from read-only CP15 registers. These operations are available only in EL3. In all other modes, executing the CP15 instruction results in an Undefined Instruction exception.

Table C5-1 AArch32 CP15 registers used to access internal memory

Function	Access	CP15 operation	Rd Data
Data Register 0	Read-only	`MRC p15, 3, <Rd>, c15, c0, 0`	Data
Data Register 1	Read-only	`MRC p15, 3, <Rd>, c15, c0, 1`	Data
Data Register 2	Read-only	`MRC p15, 3, <Rd>, c15, c0, 2`	Data
Data Register 3	Read-only	`MRC p15, 3, <Rd>, c15, c0, 3`	Data
Data Cache Tag Read Operation Register	Write-only	`MCR p15, 3, <Rd>, c15, c2, 0`	Set/Way
Instruction Cache Tag Read Operation Register	Write-only	`MCR p15, 3, <Rd>, c15, c2, 1`	Set/Way
Data Cache Data Read Operation Register	Write-only	`MCR p15, 3, <Rd>, c15, c4, 0`	Set/Way/Offset
Instruction Cache Data Read Operation Register	Write-only	`MCR p15, 3, <Rd>, c15, c4, 1`	Set/Way/Offset
TLB Data Read Operation Register	Write-only	`MCR p15, 3, <Rd>, c15, c4, 2`	Index/Way

Related information

C5.4 Encoding for the main TLB RAM on page C5-402

C5.2 Encoding for tag and data in the L1 instruction cache

The following table shows the format of the Instruction Cache Tag Read Operation Register and the Instruction Cache Data Read Operation Register.

The set-index range parameter (S) is determined by the following formula:

$S = \log_2(\text{size of the instruction cache in bytes} / 2)$

Table C5-2 Instruction cache tag and data location encoding

Bit field	Description
[31]	Cache Way
[30:S]	Unused
[S-1:6]	Set index
[5:2]	Line offset
[1:0]	Unused

The following table shows the format of the information in Data Register 0 following an Instruction Cache Tag Read Operation.

Table C5-3 Instruction cache tag data format

Bits	Description
[31]	Unused.
[30:29]	Valid and set mode: `0b00` A32. `0b01` T32. `0b11` Invalid.
[28]	Non-secure state (NS).
[27:0]	Tag address.

The Instruction Cache Data Read Operation returns two entries from the cache in Data Register 0 and Data Register 1 corresponding to the 16-bit aligned offset in the cache line:

Data Register 0 Bits[19:0] data from cache offset+ `0b00`.

Data Register 1 Bits[19:0] data from cache offset+ `0b10`.

In A32 state these two fields combined always represent a single predecoded instruction. In T32 state, they can represent any combination of 16-bit and partial or full 32-bit instructions.

C5.3 Encoding for tag and data in the L1 data cache

The following table shows the format of the Data Cache Tag Read Operation Register and the Data Cache Data Read Operation Register. The Data Cache Tag Read Operation and the Data Cache Data Read Operation use the same encoding but the latter includes an additional field to locate the appropriate doubleword in the cache line.

The set-index range parameter (S) is determined by the following formula:

$S = \log_2(\text{size of the data cache in bytes}/4)$.

Table C5-4 Data cache tag and data location encoding

Bit field	Description
[31:30]	Cache way
[29:S]	Unused
[S-1:6]	Set index
[5:3]	Cache doubleword data offset (Data Cache Data Read Operation Register only)
[2:0]	Unused, RAZ.

The Data Cache Tag Read Operation returns 64 bits of data in Data Register 0 and Data Register 1. This includes the tag information, MOESI coherency state, outer attributes, and valid bit, for the selected cache line. The following table shows the format of the return value. The Cortex-A32 processor encodes the 4-bit MOESI coherency state across two fields of Data Register 0 and Data Register 1.

Table C5-5 Data cache tag data format

Register	Bit field	Description
Data Register 1	[31]	Parity bit if ECC is implemented, otherwise RES0.
Data Register 1	[30:29]	Partial MOESI State, from tag RAM. See *Table C5-6 MOESI state* on page C5-401.
Data Register 1	[28]	Non-secure state (NS).
Data Register 1	[27:0]	Tag Address [39:12].
Data Register 0	[31]	Tag Address [11].
Data Register 0	[30:5]	Reserved, RES0.
Data Register 0	[4]	Parity bit if ECC is implemented, otherwise RES0.
Data Register 0	[3]	Dirty copy bit if ECC is implemented, otherwise RES0.
Data Register 0	[2]	Outer Allocation Hint.
Data Register 0	[1]	Outer Shareability, from Dirty RAM.
Data Register 0	[0]	Partial MOESI state, from Dirty RAM. See *Table C5-6 MOESI state* on page C5-401.

The Data Cache Data Read Operation returns two entries from the cache in Data Register 0 and Data Register 1 corresponding to the 16-bit aligned offset in the cache line:

Data Register 0 Bits[31:0] data from cache offset+ `0b000`.

Data Register 1 Bits[31:0] data from cache offset+ `0b100`.

Table C5-6 MOESI state

Tag RAM partial MOESI bits	Dirty RAM partial MOESI bits	MOESI state
00	x	Invalid (I)
01	0	SharedClean (S)
	1	SharedDirty (O)
1x	0	UniqueClean (E)
	1	UniqueDirty (M)

Related information

A5.2 Coherency between data caches with the MOESI protocol on page A5-75

C5.4 Encoding for the main TLB RAM

The Cortex-A32 processor unified TLB is built from a 2-way set-associative RAM based structure. To read the individual entries into the data registers, software must write to the TLB Data Read Operation Register.

The following table shows the format of the TLB Data Read Operation Register.

Table C5-7 Location encoding for the TLB Data Read Operation Register

Bits	Description
[31]	Unused
[30]	TLB way
[29:9]	Unused
[8:0]	TLB index **0-255** Main TLB RAM **256-287** Walk cache RAM **288-319** IPA cache RAM **320-511** Unused

The TLB Read Data Operation returns the selected entry in Data Register 0-3. The entry uses a 116-bit encoding when parity is enabled and a 113-bit encoding when parity is disabled.

Data Register 0[31:0] TLB Descriptor[31:0].
Data Register 1[31:0] TLB Descriptor[63:32].
Data Register 2[31:0] TLB Descriptor[95:64].
Data Register 3[20:0] TLB Descriptor[115:96].

The following table shows the data fields in the TLB descriptor.

Table C5-8 Main TLB descriptor data fields

Bits	Name	Description
[115:113]	Parity	If CPU cache protection is not implemented, these bits are absent.
[112:111]	S2 Level	The stage 2 level that gave this translation: `0b00` No stage 2 translation performed. `0b01` Level 1. `0b10` Level 2. `0b11` Level 3.
[110:108]	S1 Size	The stage 1 size that gave this translation. VMSAv8-32 Short-descriptor translation table format: `0b000` 4KB. `0b010` 64KB. `0b011` 1MB. `0b101` 16MB.

Table C5-8 Main TLB descriptor data fields (continued)

Bits	Name	Description
[107:104]	Domain	In VMSAv7 format, indicates one of sixteen memory regions. In non-VMSAv7 formats: • Domain[0] stores the contiguous bit information. • Domain[1] stores the page size MSB for the combined page size. • Domain[2] stores the page size MSB for the stage 1 page size.
[103:96]	Memory Type and shareability	See *TLB encoding for memory types and shareability*.
[95]	XS2	Stage2 executable permissions.
[94]	XS1Nonusr	Non user mode executable permissions.
[93]	XS1Usr	User mode executable permissions.
[92-65]	PA	Physical Address.
[64]	NS, descriptor	Security state allocated to memory region.
[63:62]	HAP	Hypervisor access permissions.
[61:59]	AP or HYP	Access permissions from stage-1 translation, or select EL2 or flag.
[58]	nG	Not global.
[57:55]	Size	This field shows the encoding for the combined page size for stage 1 and stage 2. VMSAv8-32 Short-descriptor translation table format: `0b000` 4KB. `0b010` 64KB. `0b100` 1MB. `0b110` 16MB.
[54:39]	ASID	Address Space Identifier.
[38:31]	VMID	Virtual Machine Identifier.
[30]	NS (walk)	Security state that the entry was fetched in.
[29:2]	VA	Virtual Address.
[1]	Address Sign bit	VA[48] sign bit.
[0]	Valid	Valid bit: `0` Entry does not contain valid data. `1` Entry contains valid data.

The following table shows the main TLB memory types and shareability.

Table C5-9 TLB encoding for memory types and shareability

Bits	Memory type	Description
[7]	Device Non-coherent, Outer WB Non-coherent, Outer NC Non-coherent, Outer WT	0
	Coherent, Inner WB and Outer WB	1
[6]	Device Non-coherent, Outer WB	0
	Non-coherent, Outer NC Non-coherent, Outer WT	1
	Coherent, Inner WB and Outer WB	Transience: `0` Non-transient `1` Transient.
[5:4]	Device	Stage 1 (Non-device) overridden by stage 2 (Device) `00` Not overridden `01` Overridden.
	Non-coherent, Outer WB	Inner type: `10` NC. `11` WT.
	Non-coherent, Outer NC	`11`
	Non-coherent, Outer WT	Inner type: `00` NC. `01` WB. `10` WT.
	Coherent, Inner WB and Outer WB	Inner allocation hint: `00` NA. `01` WA. `10` RA. `11` WRA.

Table C5-9 TLB encoding for memory types and shareability (continued)

Bits	Memory type	Description
[3:2]	Device	Device type: `00` nGnRnE. `01` nGnRE. `10` nGRE. `11` GRE.
	Non-coherent, Outer WB Non-coherent, Outer WT Coherent, Inner WB and Outer WB	Outer allocation hint: `00` NA. `01` WA. `10` RA. `11` WRA.
	Non-coherent, Outer NC	Inner type: `00` NC. `01` WB. `10` WT. `11` Unused.
[1:0]	Device Non-coherent, Outer WB Non-coherent, Outer NC Non-coherent, Outer WT Coherent, Inner WB and Outer WB	Shareability: `00` Non-shareable. `01` Unused. `10` Outer shareable. `11` Inner shareable.

Related information

C5.5 Encoding for walk cache on page C5-406
C5.6 Encoding for IPA cache on page C5-407
C5.1 About direct access to internal memory on page C5-398

C5.5 Encoding for walk cache

The following table shows the data fields in the walk cache descriptor.

Table C5-10 Walk cache descriptor fields

Bits	Name	Description
[115:113]	Parity bits	If CPU cache protection is not implemented, these bits are absent.
[112:83]	PA	Physical Address of the second, last, translation level.
[82:60]	VA	Virtual address.
[59]	VA sign	Virtual address sign bit.
[58:55]	-	Reserved, must be zero.
[54:39]	ASID	Address Space Identifier.
[38:31]	VMID	Virtual Machine Identifier.
[30]	NS, walk	Security state that the entry was fetched in.
[29:23]	-	Reserved, must be zero.
[22:19]	Domain	Valid only if the entry was fetched in VMSAv7 format.
[18:16]	Entry size	Memory size to which entry maps: `0b100` 1MB. `0b101` 2MB. `0b010` 8MB. `0b110`, `0b011` 32MB. `0b001` 128MB. `0b111` 512MB.
[15]	NSTable	Combined NSTable bits from first and second-level stage 1 tables or NS descriptor (VMSA).
[14]	PXNTable	Combined PXNTable bits from stage1 descriptors up to last level.
[13]	XNTable	Combined XNTable bit from stage1 descriptors up to last level.
[12:11]	APTable	Combined APTable bits from stage1 descriptors up to last level.
[10]	-	Reserved, must be zero.
[9]	EL2	Set if the entry was fetched in EL2 mode.
[8:1]	Attrs	Physical attributes of the final level stage 1 table.
[0]	Valid	Valid bit: `0` Entry does not contain valid data. `1` Entry contains valid data.

C5.6 Encoding for IPA cache

The following table shows the data fields in the IPA cache descriptor.

Table C5-11 IPA cache descriptor fields

Bits	Name	Description
[115:113]	Parity bits	If CPU cache protection is not implemented, these bits are absent.
[112:85]	PA	Physical address.
[84:62]	IPA	Unused lower bits, page size dependent, must be zero.
[61:59]	-	Reserved, must be zero.
[58:55]	Size	Stage 2 page size. The values are: `0b0001` 4KB. `0b1001` 16KB. `0b0011` 64KB. `0b0101` 2MB. `0b1011` 32MB. `0b0111` 512MB.
[54:39]	-	Reserved, must be zero.
[38:31]	VMID	Virtual Machine Identifier.
[30:12]	-	Reserved, must be zero.
[11:10]	Entry granule	The values are: `0b00` 4KB. `0b10` 16KB. `0b01` 64KB.
[9:6]	Memattrs	Memory attributes.
[5]	XN	Execute Never.
[4:3]	HAP	Hypervisor access permissions.
[2:1]	SH	Shareability.
[0]	Valid	The entry contains valid data.

Chapter C6
AArch32 debug registers

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

It contains the following sections:

C6.1 AArch32 debug register summary

This section summarizes the 32-bit and 64-bit debug control registers that are accessible in the AArch32 Execution state from the internal CP14 interface.

These registers, listed in the following table, are accessed by the `MCR` and `MRC` instructions in the order of CRn, op2, CRm, Op1 or `MCRR` and `MRRC` instructions in the order of CRm, Op1. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

See the *C7.1 Memory-mapped debug register summary* on page C7-424 for a complete list of registers accessible from the external debug interface.

Table C6-1 AArch32 debug register summary

CRn	Op2	CRm	Op1	Name	Type	Description
c0	0	c0	0	DBGDIDR	RO	*C6.4 Debug ID Register* on page C6-418
c0	0	c1	0	DBGDSCRint	RO	Debug Status and Control Register, Internal View
c0	0	c2	0	DBGDCCINT	RW	Debug Comms Channel Interrupt Enable Register
c0	0	c5	0	DBGDTRTXint	WO	Debug Data Transfer Register, Transmit, Internal View
				DBGDTRRXint	RO	Debug Data Transfer Register, Receive, Internal View
c0	0	c6	0	DBGWFAR	RW	Watchpoint Fault Address Register, RES0 Previously returned information about the address of the instruction that accessed a watchpoint address. This register is now deprecated and is RES0.
c0	0	c7	0	DBGVCR	RW	Debug Vector Catch Register
c0	2	c0	0	DBGDTRRXext	RW	Debug Data Transfer Register, Receive, External View
c0	2	c2	0	DBGDSCRext	RW	Debug Status and Control Register, External View
c0	2	c3	0	DBGDTRTXext	RW	Debug Data Transfer Register, Transmit, External View
c0	2	c6	0	DBGOSECCR	RW	Debug OS Lock Exception Catch Control Register
c0	4	c0	0	DBGBVR0	RW	Debug Breakpoint Value Register 0
c0	4	c1	0	DBGBVR1	RW	Debug Breakpoint Value Register 1
c0	4	c2	0	DBGBVR2	RW	Debug Breakpoint Value Register 2
c0	4	c3	0	DBGBVR3	RW	Debug Breakpoint Value Register 3
c0	4	c4	0	DBGBVR4	RW	Debug Breakpoint Value Register 4
c0	4	c5	0	DBGBVR5	RW	Debug Breakpoint Value Register 5
c0	5	c0	0	DBGBCR0	RW	*C6.2 Debug Breakpoint Control Registers* on page C6-412
c0	5	c1	0	DBGBCR1	RW	
c0	5	c2	0	DBGBCR2	RW	
c0	5	c3	0	DBGBCR3	RW	
c0	5	c4	0	DBGBCR4	RW	
c0	5	c5	0	DBGBCR5	RW	
c0	6	c0	0	DBGWVR0	RW	Debug Watchpoint Value Register 0
c0	6	c1	0	DBGWVR1	RW	Debug Watchpoint Value Register 1

Table C6-1 AArch32 debug register summary (continued)

CRn	Op2	CRm	Op1	Name	Type	Description
c0	6	c2	0	DBGWVR2	RW	Debug Watchpoint Value Register 2
c0	6	c3	0	DBGWVR3	RW	Debug Watchpoint Value Register 3
c0	7	c0	0	DBGWCR0	RW	*C6.3 Debug Watchpoint Control Registers* on page C6-415
c0	7	c1	0	DBGWCR1	RW	
c0	7	c2	0	DBGWCR2	RW	
c0	7	c3	0	DBGWCR3	RW	
c1	0	c0	0	DBGDRAR[31:0]	RO	Debug ROM Address Register
-	-	c1	-	DBGDRAR[63:0]	RO	
c1	1	c4	0	DBGBXVR4	RW	Debug Breakpoint Extended Value Register 4
c1	1	c5	0	DBGBXVR5	RW	Debug Breakpoint Extended Value Register 5
c1	4	c0	0	DBGOSLAR	WO	Debug OS Lock Access Register
c1	4	c1	0	DBGOSLSR	RO	Debug OS Lock Status Register
c1	4	c3	0	DBGOSDLR	RW	Debug OS Double Lock Register
c1	4	c4	0	DBGPRCR	RW	Debug Power/Reset Control Register
c2	2	c0	0	DBGDSAR[31:0]	RO	Debug Self Address Register RES0
-	0	c2	-	DBGDSAR[63:0]	RO	Previously defined the offset from the base address defined in DBGDRAR of the physical base address of the debug registers for the processor. This register is now deprecated and RES0.
c7	7	c0	0	DBGDEVID2	RO	Debug Device ID Register 2, RES0
c7	7	c1	0	DBGDEVID1	RO	*C6.6 Debug Device ID Register 1* on page C6-422
c7	7	c2	0	DBGDEVID	RO	*C6.5 Debug Device ID Register* on page C6-420
c7	6	c8	0	DBGCLAIMSET	RW	Debug Claim Tag Set Register
c7	6	c9	0	DBGCLAIMCLR	RW	Debug Claim Tag Clear Register
c7	6	c14	0	DBGAUTHSTATUS	RO	Debug Authentication Status Register

C6.2 Debug Breakpoint Control Registers

The DBGBCR*n* characteristics are:

Purpose

Holds control information for a breakpoint. Each DBGBVR is associated with a DBGBCR to form a *Breakpoint Register Pair* (BRP). DBGBVR*n* is associated with DBGBCR*n* to form BRP*n*.

The range of *n* for DBGBCR*n* is 0 to 5.

Usage constraints

These registers are accessible as follows:

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

Configurations

DBGBCR*n* are architecturally mapped to the external DBGBCR*n*_EL1 registers.

Attributes

See *C6.1 AArch32 debug register summary* on page C6-410.

The debug logic reset value of a DBGBCR*n* is UNKNOWN.

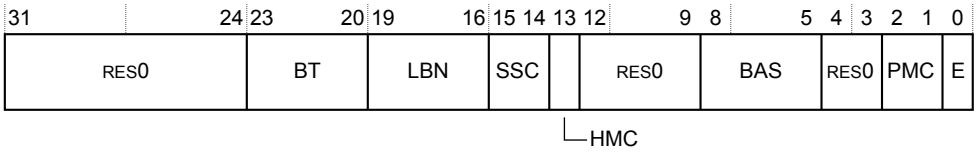

Figure C6-1 DBGBCR

[31:24]

Reserved, RES0.

BT, [23:20]

Breakpoint Type. This field controls the behavior of Breakpoint debug event generation. This includes the meaning of the value held in the associated DBGBVR*n*, indicating whether it is an instruction address match or mismatch or a Context match. It also controls whether the breakpoint is linked to another breakpoint. The possible values are:

`0b0000` Unlinked instruction address match.

`0b0001` Linked instruction address match.

`0b0010` Unlinked Context ID match.

`0b0011` Linked Context ID match.

`0b0100` Unlinked instruction address mismatch.

`0b0101` Linked instruction address mismatch.

`0b1000` Unlinked VMID match.

`0b1001` Linked VMID match.

`0b1010` Unlinked VMID + Context ID match.

`0b1011` Linked VMID + Context ID match.

All other values are reserved.

The field break down is:

- BT[3:1]: Base type. If the breakpoint is not context-aware, these bits are RES0. Otherwise, the possible values are:

  `0b000` Match address. DBGBVR*n* is the address of an instruction.

  `0b001` Match context ID. DBGBVR*n*[31:0] is a context ID.

  `0b010` Mismatch address. Behaves as type 0b000 if halting debug-mode is enabled and halting is allowed.

  Otherwise, DBGBVR*n* is the address of an instruction to be stepped.

  `0b100` Match VMID. DBGBVR*n*[7:0] is a VMID.

  `0b101` Match VMID and context ID. DBGBVR*n*[31:0] is a context ID, and DBGBVR*n*[7:0] is a VMID.

- BT[0]: Enable linking.

LBN, [19:16]

Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to.

SSC, [15:14]

Security State Control. Determines the security states that a breakpoint debug event for breakpoint *n* is generated.

This field must be interpreted with the *Higher Mode Control* (HMC), and *Privileged Mode Control* (PMC), fields to determine the mode and security states that can be tested.

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

HMC, [13]

Hyp Mode Control bit. Determines the debug perspective for deciding when a breakpoint debug event for breakpoint *n* is generated.

This bit must be interpreted with the SSC and PMC fields to determine the mode and security states that can be tested.

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

[12:9]

Reserved, RES0.

BAS, [8:5]

Byte Address Select. Defines which half-words a regular breakpoint matches, regardless of the instruction set and execution state. A debugger must program this field as follows:

`0x3` Match the T32 instruction at DBGBVR*n*.

`0xC` Match the T32 instruction at DBGBVR*n*+2.

`0xF` Match the A32 instruction at DBGBVR*n*, or context match.

All other values are reserved.

The Armv8-A architecture does not support direct execution of Java bytecodes. BAS[3] and BAS[1] ignore writes and on reads return the values of BAS[2] and BAS[0] respectively.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information on how the BAS field is interpreted by hardware.

[4:3]

Reserved, RES0.

PMC, [2:1]

Privileged Mode Control. Determines the exception level or levels that a breakpoint debug event for breakpoint *n* is generated.

This field must be interpreted with the SSC and HMC fields to determine the mode and security states that can be tested.

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

Bits[2:1] have no effect for accesses made in Hyp mode.

E, [0]

Enable breakpoint. This bit enables the BRP:

`0` BRP disabled.

`1` BRP enabled.

A BRP never generates a breakpoint debug event when it is disabled.

The value of DBGBCR*n*.E is UNKNOWN on reset. A debugger must ensure that DBGBCR*n*.E has a defined value before it enables debug.

To access the DBGBCR*n*:

```
MRC p14, 0, <Rt>, c0, cn, 4; Read Debug Breakpoint Control Register nMCR p14, 0, <Rt>, c0,
cn, 4; Write Debug Breakpoint Control Register n
```

The DBGBCR*n*_EL1 can be accessed through the external debug interface, offset `0x4n8`. The range of bits for DBGBCR*n*_EL1 is 0 to 5.

C6.3 Debug Watchpoint Control Registers

The DBGWCR*n* characteristics are:

Purpose

Holds control information for a watchpoint. Each DBGWCR is associated with a DBGWVR_EL1 to form a *Watchpoint Register Pair* (WRP). DBGWCR*n* is associated with DBGWVR*n*_EL1 to form WRP*n*.

The range of *n* for DBGBCR*n* is 0 to 3.

Usage constraints

These registers are accessible as follows:

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

Configurations

The DBGWCR*n* are architecturally mapped to the external DBGWCR*n*_EL1 registers.

Attributes

See *C6.1 AArch32 debug register summary* on page C6-410.

The debug logic reset value of a DBGWCR_EL1 is UNKNOWN.

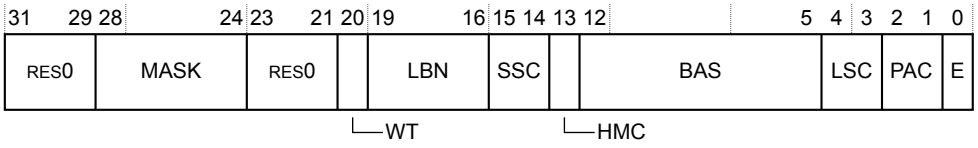

Figure C6-2 DBGWCR

[31:29]

Reserved, RES0.

MASK, [28:24]

Address mask. Only objects up to 2GB can be watched using a single mask.

`0b0000` No mask

`0b0001` Reserved

`0b0010` Reserved

Other values mask the corresponding number of address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).

[23:21]

Reserved, RES0.

WT, [20]

Watchpoint type. Possible values are:

`0` Unlinked data address match.

`1` Linked data address match.

On Cold reset, the field reset value is architecturally UNKNOWN.

LBN, [19:16]

Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.

On Cold reset, the field reset value is architecturally UNKNOWN.

SSC, [15:14]

Security state control. Determines the Security states under which a watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.

On Cold reset, the field reset value is architecturally UNKNOWN.

HMC, [13]

Higher mode control. Determines the debug perspective for deciding when a watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PACfields.

On Cold reset, the field reset value is architecturally UNKNOWN.

BAS, [12:5]

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR*n* is being watched. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

LSC, [4:3]

Load/store access control. This field enables watchpoint matching on the type of access being made. The possible values are:

`0b01` Match instructions that load from a watchpointed address.

`0b10` Match instructions that store to a watchpointed address.

`0b11` Match instructions that load from or store to a watchpointed address.

All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

Ignored if E is 0.

On Cold reset, the field reset value is architecturally UNKNOWN.

PAC, [2:1]

Privilege of access control. Determines the exception level or levels at which a watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMCfields.

On Cold reset, the field reset value is architecturally UNKNOWN.

E, [0]

Enable watchpoint n. Possible values are:

`0` Watchpoint disabled.

`1` Watchpoint enabled.

On Cold reset, the field reset value is architecturally UNKNOWN.

To access the DBGWCR*n*:

```
MRC p14, 0, <Rt>, c0, cn, 7; Read Debug Watchpoint Control Register nMCR p14, 0, <Rt>, c0,
cn, 7; Write Debug Watchpoint Control Register n
```

The DBGWCR*n*_EL1 can be accessed through the external debug interface, offset `0x8n8`. The range of *n* for DBGWCRn_EL1 is 0 to 3.

C6.4 Debug ID Register

The DBGDIDR characteristics are:

Purpose

Specifies:

- The version of the Debug architecture that is implemented.
- Some features of the debug implementation.

Usage constraints

This register is accessible as follows:

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

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

See *C6.1 AArch32 debug register summary* on page C6-410.

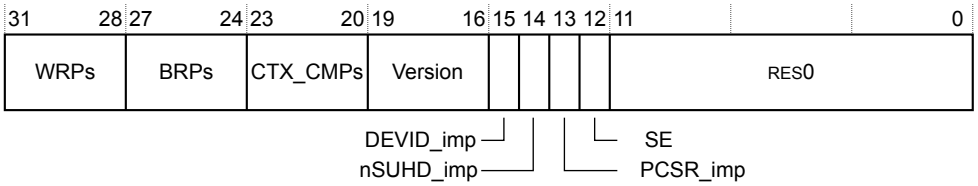

Figure C6-3 DBGDIDR bit assignments

WRPs, [31:28]

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:

`0x3` The processor implements 4 WRPs.

BRPs, [27:24]

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:

`0x5` The processor implements 6 BRPs.

CTX_CMPs, [23:20]

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 4 and 5.

Version, [19:16]

The Debug architecture version.

`0x6` The processor implements Armv8 Debug architecture.

DEVID_imp, [15]

Reserved, RAO.

nSUHD_imp, [14]

Secure User Halting Debug not implemented bit. The value is:

1 The processor does not implement Secure User Halting Debug.

PCSR_imp, [13]

Reserved, RAZ.

SE, [12]

EL3 implemented. The value is:

1 The processor implements EL3.

[11:0]

Reserved, RES0.

To access the DBGDIDR:

```
MRC p14, 0, <Rt>, c0, c0, 0; Read Debug ID Register
```

C6.5 Debug Device ID Register

The DBGDEVID characteristics are:

Purpose

Specifies the version of the Debug architecture is implemented, and some features of the debug implementation.

Usage constraints

This register is accessible as follows:

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

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

See *C6.1 AArch32 debug register summary* on page C6-410.

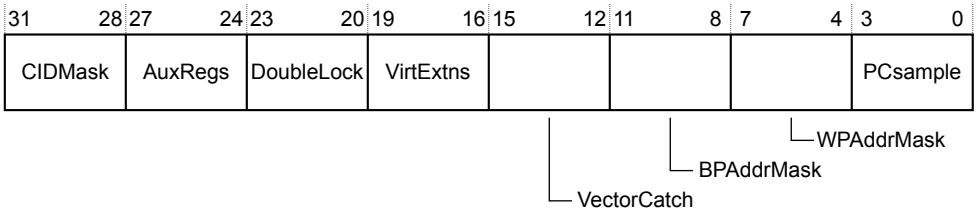

Figure C6-4 DBGDEVID bit assignments

CIDMask, [31:28]

Specifies the level of support for the Context ID matching breakpoint masking capability. This value is:

`0x0` Context ID masking is not implemented.

AuxRegs, [27:24]

Specifies support for the Debug External Auxiliary Control Register. This value is:

`0x0` None supported.

DoubleLock, [23:20]

Specifies support for the Debug OS Double Lock Register. This value is:

`0x1` The processor supports Debug OS Double Lock Register.

VirtExtns, [19:16]

Specifies whether EL2 is implemented. This value is:

`0x1` The processor implements EL2.

VectorCatch, [15:12]

Defines the form of the vector catch event implemented. This value is:

`0x0` The processor implements address matching form of vector catch.

BPAddrMask, [11:8]

Indicates the level of support for the *Immediate Virtual Address* (IVA) matching breakpoint masking capability. This value is:

`0xF` Breakpoint address masking not implemented. DBGBCRn[28:24] are RES0.

WPAddrMask, [7:4]

Indicates the level of support for the DVA matching watchpoint masking capability. This value is:

`0x1` Watchpoint address mask implemented.

PCSample, [3:0]

Indicates the level of support for Program Counter sampling using debug registers 40 and 41. 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 Debug Device ID Register 0
```

C6.6 Debug Device ID Register 1

The DBGDEVID1 characteristics are:

Purpose

Adds to the information given by the DBGDIDR by describing other features of the debug implementation.

Usage constraints

This register is accessible as follows:

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

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

See *C6.1 AArch32 debug register summary* on page C6-410.

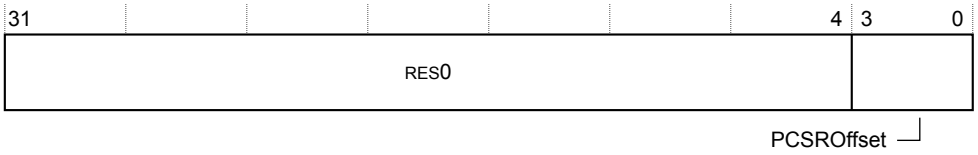

Figure C6-5 DBGDEVID1 bit assignments

[31:4]

Reserved, RES0.

PCSROffset, [3:0]

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 in the AArch32 state.

To access the DBGDEVID1:

```
MRC p14, 0, <Rt>, c7, c1, 47 Read Debug Device ID Register 1
```

Chapter C7
Memory-mapped debug registers

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

It contains the following sections:

C7.1 Memory-mapped debug register summary

The following table shows the offset address for the registers that are accessible from the external debug interface.

For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table C7-1 Memory-mapped debug register summary

Offset	Name	Type	Width	Description
0x000-0x01C	-	-		Reserved
0x020	EDESR	RW	32	External Debug Event Status Register
0x024	EDECR	RW	32	External Debug Execution Control Register
0x028-0x02C	-	-	-	Reserved
0x030	EDWAR[31:0]	RO	64	External Debug Watchpoint Address Register
0x034	EDWAR[63:32]			
0x038-0x07C	-	-	-	Reserved
0x080	DBGDTRRX_EL0	RW	32	Debug Data Transfer Register, Receive
0x084	EDITR	WO	32	External Debug Instruction Transfer Register
0x088	EDSCR	RW	32	External Debug Status and Control Register
0x08C	DBGDTRTX_EL0	RW	32	Debug Data Transfer Register, Transmit
0x090	EDRCR	WO	32	*C7.2 External Debug Reserve Control Register* on page C7-428
0x094	EDACR	RW	32	External Debug Auxiliary Control Register
0x098	EDECCR	RW	32	External Debug Exception Catch Control Register
0x09C	-	-	32	Reserved
0x0A0	EDPCSRlo	RO	32	External Debug Program Counter Sample Register, low word
0x0A4	EDCIDSR	RO	32	External Debug Context ID Sample Register
0x0A8	EDVIDSR	RO	32	External Debug Virtual Context Sample Register
0x0AC	EDPCSRhi	RO	32	External Debug Program Counter Sample Register, high word
0x0B0-0x2FC	-	-	-	Reserved
0x300	OSLAR_EL1	WO	32	OS Lock Access Register
0x304-0x30C	-	-	-	Reserved
0x310	EDPRCR	RW	32	External Debug Power/Reset Control Register
0x314	EDPRSR	RO	32	External Debug Processor Status Register
0x318-0x3FC	-	-	--	Reserved
0x400	DBGBVR0_EL1[31:0]	RW	64	Debug Breakpoint Value Register 0
0x404	DBGBVR0_EL1[63:32]			
0x408	DBGBCR0_EL1	RW	32	*C6.2 Debug Breakpoint Control Registers* on page C6-412

Table C7-1 Memory-mapped debug register summary (continued)

Offset	Name	Type	Width	Description
0x40C	-	-	-	Reserved
0x410	DBGBVR1_EL1[31:0]	RW	64	Debug Breakpoint Value Register 1
0x414	DBGBVR1_EL1[63:32]			
0x418	DBGBCR1_EL1	RW	32	*C6.2 Debug Breakpoint Control Registers* on page C6-412
0x41C	-	-	-	Reserved
0x420	DBGBVR2_EL1[31:0]	RW	64	Debug Breakpoint Value Register 2
0x424	DBGBVR2_EL1[63:32]			
0x428	DBGBCR2_EL1	RW	32	*C6.2 Debug Breakpoint Control Registers* on page C6-412
0x42C	-	-	-	Reserved
0x430	DBGBVR3_EL1[31:0]	RW	64	Debug Breakpoint Value Register 3
0x434	DBGBVR3_EL1[63:32]			
0x438	DBGBCR3_EL1	RW	32	*C6.2 Debug Breakpoint Control Registers* on page C6-412
0x43C	-	-	-	Reserved
0x440	DBGBVR4_EL1[31:0]	RW	64	Debug Breakpoint Value Register 4
0x444	DBGBVR4_EL1[63:32]			
0x448	DBGBCR4_EL1	RW	32	*C6.2 Debug Breakpoint Control Registers* on page C6-412
0x44C	-	-	-	Reserved
0x450	DBGBVR5_EL1[31:0]	RW	64	Debug Breakpoint Value Register 5
0x454	DBGBVR5_EL1[63:32]			
0x458	DBGBCR5_EL1	RW	32	*C6.2 Debug Breakpoint Control Registers* on page C6-412
0x45C-0x7FC	-	-	-	Reserved
0x800	DBGWVR0_EL1[31:0]	RW	64	Debug Watchpoint Value Register 0
0x804	DBGWVR0_EL1[63:32]			
0x808	DBGWCR0_EL1	RW	32	Debug Watchpoint Control Register 0
0x80C	-	-	-	Reserved
0x810	DBGWVR1_EL1[31:0]	RW	64	Debug Watchpoint Value Register 1
0x814	DBGWVR1_EL1[63:32]			
0x818	DBGWCR1_EL1	RW	32	Debug Watchpoint Control Register 1
0x81C	-	-	-	Reserved
0x820	DBGWVR2_EL1[31:0]	RW	64	Debug Watchpoint Value Register 2
0x824	DBGWVR2_EL1[63:32]			
0x828	DBGWCR2_EL1	RW	32	Debug Watchpoint Control Register 2
0x82C	-	-	-	Reserved
0x830	DBGWVR3_EL1[31:0]	RW	64	Debug Watchpoint Value Register 0,
0x834	DBGWVR3_EL1[63:32]			

Table C7-1 Memory-mapped debug register summary (continued)

Offset	Name	Type	Width	Description
0x838	DBGWCR3_EL1	RW	32	Debug Watchpoint Control Register 2
0x83C-0xCFC	-	-	-	Reserved
0xD00	MIDR_EL1	RO	32	*B1.96 Main ID Register* on page B1-307
0xD04-0xD1C	-	-	-	Reserved
0xD20	EDPFR[31:0]	RO	64	*C7.6 External Debug Processor Feature Register* on page C7-433
0xD24	EDPFR[63:32]			
0xD28	EDDFR[31:0]	RO	64	*C7.7 External Debug Feature Register* on page C7-435
0xD2C	EDDFR[63:32]			
0xD30-0xD34	-	-	-	Reserved
0xD38-0xD3C	-	-	-	Reserved
0xD60	EDAA32PFR	RO	64	*C7.8 External Debug AArch32 Processor Feature Register* on page C7-437
0xD64-0xEFC	-	-	-	Reserved
0xF00	EDITCTRL	RW	32	*C7.3 External Debug Integration Mode Control Register* on page C7-430
0xF04-0xF9C	-	-	-	Reserved
0xFA0	DBGCLAIMSET_EL1	RW	32	Debug Claim Tag Set Register
0xFA4	DBGCLAIMCLR_EL1	RW	32	Debug Claim Tag Clear Register
0xFA8	EDDEVAFF0	RO	32	*B1.97 Multiprocessor Affinity Register* on page B1-309
0xFAC	EDDEVAFF1	RO	32	External Debug Device Affinity Register 1, RES0
0xFB0	EDLAR	WO	32	External Debug Lock Access Register
0xFB4	EDLSR	RO	32	External Debug Lock Status Register
0xFB8	DBGAUTHSTATUS_EL1	RO	32	Debug Authentication Status Register
0xFBC	EDDEVARCH	RO	32	External Debug Device Architecture Register
0xFC0	EDDEVID2	RO	32	External Debug Device ID Register 2, RES0
0xFC4	EDDEVID1	RO	32	*C7.5 External Debug Device ID Register 1* on page C7-432
0xFC8	EDDEVID	RO	32	*C7.4 External Debug Device ID Register 0* on page C7-431
0xFCC	EDDEVTYPE	RO	32	External Debug Device Type Register
0xFD0	EDPIDR4	RO	32	*C7.14 External Debug Peripheral Identification Register 4* on page C7-443
0xFD4-0xFDC	EDPIDR5-7	RO	32	*C7.15 External Debug Peripheral Identification Register 5-7* on page C7-444
0xFE0	EDPIDR0	RO	32	*C7.10 External Debug Peripheral Identification Register 0* on page C7-439
0xFE4	EDPIDR1	RO	32	*C7.11 External Debug Peripheral Identification Register 1* on page C7-440

Table C7-1 Memory-mapped debug register summary (continued)

Offset	Name	Type	Width	Description
0xFE8	EDPIDR2	RO	32	*C7.12 External Debug Peripheral Identification Register 2* on page C7-441
0xFEC	EDPIDR3	RO	32	*C7.13 External Debug Peripheral Identification Register 3* on page C7-442
0xFF0	EDCIDR0	RO	32	*C7.17 External Debug Component Identification Register 0* on page C7-446
0xFF4	EDCIDR1	RO	32	*C7.18 External Debug Component Identification Register 1* on page C7-447
0xFF8	EDCIDR2	RO	32	*C7.19 External Debug Component Identification Register 2* on page C7-448
0xFFC	EDCIDR3	RO	32	*C7.20 External Debug Component Identification Register 3* on page C7-449

C7.2 External Debug Reserve Control Register

The EDRCR characteristics are:

Purpose

This register is used to allow imprecise entry to Debug state and clear sticky bits in EDSCR.

This register is part of the Debug registers functional group.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	WI	WO

Configurations

EDRCR is in the Core power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

EDRCR is a 32-bit register.

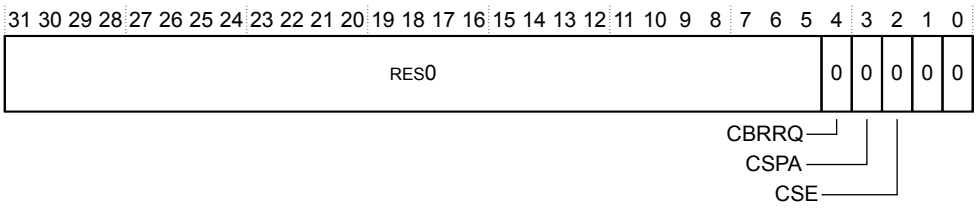

Figure C7-1 EDRCR bit assignments

[31:5]

Reserved, RES0.

CBRRQ, [4]

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.

CSPA, [3]

Clear Sticky Pipeline Advance. This bit is used to clear the EDSCR.PipeAdv bit to 0. The actions on writing to this bit are:

`0` No action.

`1` Clear the EDSCR.PipeAdv bit to 0.

CSE, [2]

Clear Sticky Error. Used to clear the EDSCR cumulative error bits to 0. The actions on writing to this bit are:

`0` No action

1 Clear the EDSCR.{TXU, RXO, ERR} bits, and, if the processor is in Debug state, the EDSCR.ITO bit, to 0.

[1:0]

Reserved, RES0.

The EDRCR can be accessed through the external debug interface, offset `0x090`.

C7.3 External Debug Integration Mode Control Register

The EDITCTRL characteristics are:

Purpose

Enables the external debug to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the processor, for integration testing or topology detection.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	RO	RW

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

EDITCTRL is in the processor power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

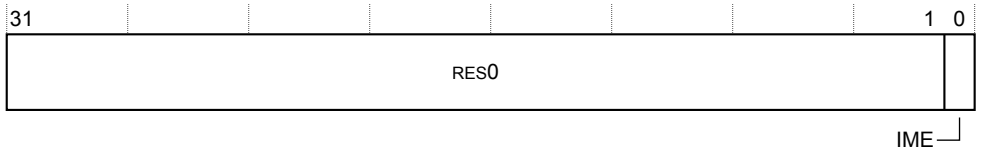

Figure C7-2 EDITCTRL bit assignments

[31:1]

Reserved, RES0.

IME, [0]

Integration Mode Enable.

RES0. The device does not revert to an integration mode to enable integration testing or topology detection.

The EDITCTRL can be accessed through the external debug interface, offset `0xF00`.

C7.4 External Debug Device ID Register 0

The EDDEVID characteristics are:

Purpose

Provides extra information for external debuggers about features of the debug implementation.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDDEVID is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

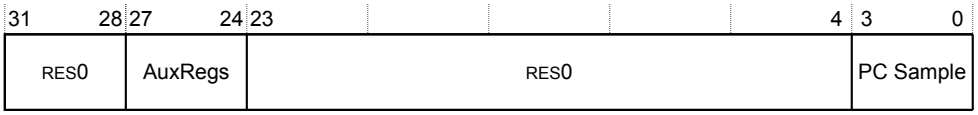

Figure C7-3 EDDEVID bit assignments

[31:28]

Reserved, RES0.

AuxRegs, [27:24]

Indicates support for Auxiliary registers:

`0x0` None supported.

[23:4]

Reserved, RES0.

PC Sample, [3:0]

Indicates the level of sample-based profiling support using external debug registers 40 to 43:

`0x3` EDPCSR, EDCIDSR, and EDVIDSR are implemented.

The EDDEVID can be accessed through the external debug interface, offset `0xFC8`.

C7.5 External Debug Device ID Register 1

The EDDEVID1 characteristics are:

Purpose

Provides extra information for external debuggers about features of the debug implementation.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDDEVID1 is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

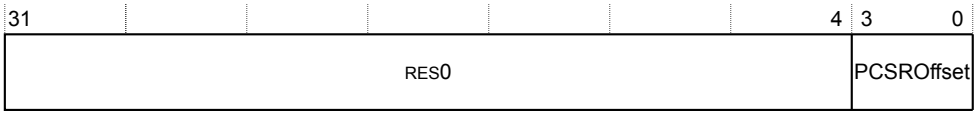

Figure C7-4 EDDEVID1 bit assignments

[31:4]

Reserved, RES0.

PCSROffset, [3:0]

Indicates the offset applied to PC samples returned by reads of EDPCSR:

`0x2` EDPCSR samples have no offset applied and do not sample the instruction set state.

The EDDEVID1 can be accessed through the external debug interface, offset `0xFC4`.

C7.6 External Debug Processor Feature Register

The EDPFR characteristics are:

Purpose

Provides additional information about implemented PE features in AArch64.

Usage constraints

This register is accessible as follows:

Default
RO

Configurations

The EDPFR is in the Debug power domain.

Attributes

EDPFR is a 64-bit register.

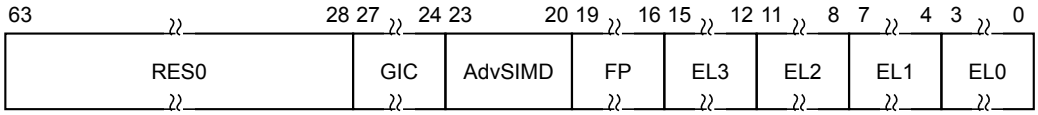

Figure C7-5 EDPFR bit assignments

[63:28]

Reserved, RES0.

GIC, [27:24]

System register GIC interface. Defined values are:

`0x0` No System register interface to the GIC is supported.

`0x1` System register interface to the GIC CPU interface is supported.

All other values are reserved.

AdvSIMD, [23:20]

Advanced SIMD. Defined values are:

`0x0` Advanced SIMD is implemented.

`0xF` Advanced SIMD is not implemented.

All other values are reserved.

FP, [19:16]

Floating-point. Defined values are:

`0x0` Floating-point is implemented.

`0xF` Floating-point is not implemented.

All other values are reserved.

EL3 handling, [15:12]

EL3 exception handling:

`0x0` Instructions can be executed at EL3 in AArch32 state.

EL2 handling, [11:8]

EL2 exception handling:

`0x0` Instructions can be executed at EL2 in AArch32 state.

EL1 handling, [7:4]

EL1 exception handling. The possible values are:

`0x0` Instructions can be executed at EL1 in AArch32 state.

EL0 handling, [3:0]

EL0 exception handling. The possible values are:

`0x0` Instructions can be executed at EL0 in AArch32 state.

The EDPFR[31:0] can be accessed through the external debug interface, offset `0xD20`.

The EDPFR[63:32] can be accessed through the external debug interface, offset `0xD24`.

C7.7 External Debug Feature Register

The EDDFR characteristics are:

Purpose

Provides top level information about the debug system in AArch64.

Usage constraints

This register is accessible as follows:

Default
RO

Configurations

EDDFR is in the Debug power domain.

Attributes

EDDFR is a 64-bit register.

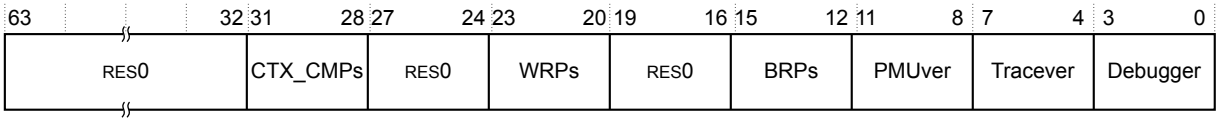

Figure C7-6 EDDFR bit assignments

[63:32]

Reserved, RES0.

CTX_CMPs, [31:28]

Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints.

[27:24]

Reserved, RES0.

WRPs, [23:20]

Number of watchpoints, minus 1. The value of `0b0000` is reserved.

[19:16]

Reserved, RES0.

BRPs, [15:12]

Number of breakpoints, minus 1. The value of `0b0000` is reserved.

PMUVer, [11:8]

Performance Monitors extension version. Indicates whether system register interface to Performance Monitors extension is implemented. Defined values are:

`0x0000` Performance Monitors extension system registers not implemented.

`0x0001` Performance Monitors extension system registers implemented, PMUv3.

`0x1111` IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported.

All other values are reserved.

TraceVer [7:4]

Trace support. Indicates whether system register interface to a trace macrocell is implemented. Defined values are:

`0x0000` Trace macrocell system registers not implemented.

`0x0001` Trace macrocell system registers implemented.

All other values are reserved.

A value of `0x0000` only indicates that no system register interface to a trace macrocell is implemented. A trace macrocell might nevertheless be implemented without a system register interface.

UNKOWN, [7:4]

Reserved, UNKNOWN.

EDDFR[31:0] can be accessed through the external debug interface, offset `0xD28`.

EDDFR[63:32] can be accessed through the external debug interface, offset `0xD2C`.

C7.8 External Debug AArch32 Processor Feature Register

The EDAA32PFR characteristics are:

Purpose

Provides additional information about implemented PE features in AArch32.

Usage constraints

This register is accessible as follows:

Default
RO

Configurations

EDAA32PFR is in the Debug power domain.

Attributes

EDAA32PFR is a 64-bit register.

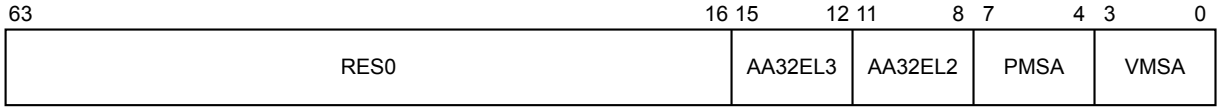

Figure C7-7 EDAA32PFR bit assignments

[63:16]

Reserved, RES0.

AA32EL3, [15:12]

AArch32 EL3 Exception level handling.

`0001` EL3 can be executed in AArch32 Execution State only.

All other values are reserved.

AA32EL2, [11:8]

AArch32 EL2 Exception level handling.

`0001` EL2 can be executed in AArch32 Execution State only.

All other values are reserved.

PMSA, [7:4]

Indicates support for an Armv8-R PMSA.

In Armv8-A, the only permitted value is `0x0`.

VMSA, [7:4]

Indicates support for an Armv8-R PMSA.

In Armv8-A, the only permitted value is `0x0`.

The EDAA32PFR [31:0] can be accessed through the internal memory-mapped interface and the external debug interface, offset `0xD60`.

The EDAA32PFR [63:32] can be accessed through the internal memory-mapped interface and the external debug interface, offset `0xD64`.

C7.9 External Debug Peripheral Identification Registers

The External Debug Peripheral Identification Registers provide standard information required for all components that conform to the Arm Debug Interface v5 specification.

The following table lists the External Debug Peripheral Identification Registers.

Table C7-2 Summary of the External Debug Peripheral Identification Registers

Register	Value	Offset
Peripheral ID4	0x04	0xFD0
Peripheral ID5	0x00	0xFD4
Peripheral ID6	0x00	0xFD8
Peripheral ID7	0x00	0xFDC
Peripheral ID0	0x11	0xFE0
Peripheral ID1	0xBD	0xFE4
Peripheral ID2	0x2B	0xFE8
Peripheral ID3	0x00	0xFEC

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

C7.10 External Debug Peripheral Identification Register 0

The EDPIDR0 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDPIDR0 is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

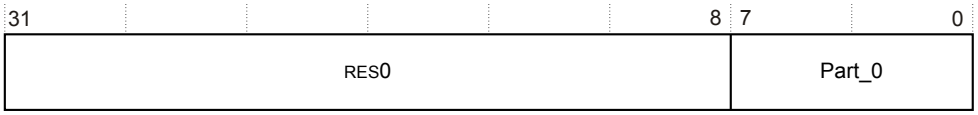

Figure C7-8 EDPIDR0 bit assignments

[31:8]

Reserved, RES0.

Part_0, [7:0]

`0x01` Least significant byte of the debug part number.

The EDPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

C7.11 External Debug Peripheral Identification Register 1

The EDPIDR1 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDPIDR1 is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

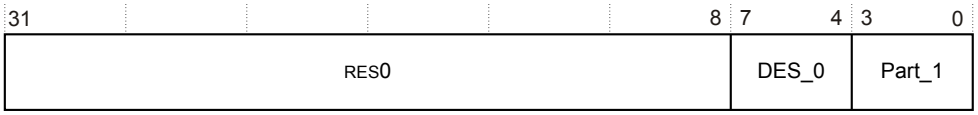

Figure C7-9 EDPIDR1 bit assignments

[31:8]

Reserved, RES0.

DES_0, [7:4]

`0xB` Arm Limited. This is the least significant nibble of JEP106 ID code.

Part_1, [3:0]

`0xD` Most significant nibble of the debug part number.

The EDPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

C7.12 External Debug Peripheral Identification Register 2

The EDPIDR2 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDPIDR2 is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

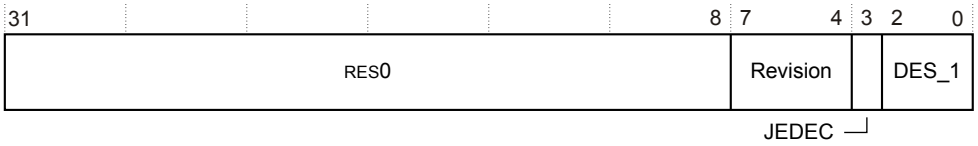

Figure C7-10 EDPIDR2 bit assignments

[31:8]

Reserved, RES0.

Revision, [7:4]

`0x2` r1p0.

JEDEC, [3]

`0b1` RAO. Indicates a JEP106 identity code is used.

DES_1, [2:0]

`0b011` Arm Limited. This is the most significant nibble of JEP106 ID code.

The EDPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

C7.13 External Debug Peripheral Identification Register 3

The EDPIDR3 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDPIDR3 is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

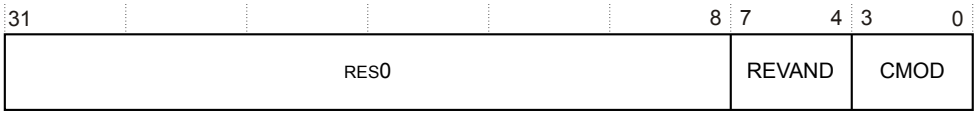

Figure C7-11 EDPIDR3 bit assignments

[31:8]

Reserved, RES0.

REVAND, [7:4]

`0x0` Part minor revision.

CMOD, [3:0]

`0x0` Customer modified.

The EDPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

C7.14 External Debug Peripheral Identification Register 4

The EDPIDR4 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDPIDR4 is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

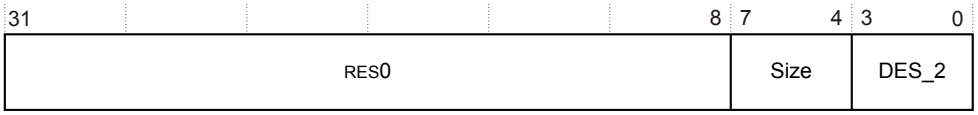

Figure C7-12 EDPIDR4 bit assignments

[31:8]

Reserved, RES0.

Size, [7:4]

`0x0` Size of the component. Log_2 the number of 4KB pages from the start of the component to the end of the component ID registers.

DES_2, [3:0]

`0x4` Arm Limited. This is the least significant nibble JEP106 continuation code.

The EDPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

C7.15 External Debug Peripheral Identification Register 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

They are reserved for future use and are RES0.

C7.16 External Debug Component Identification Registers

There are four read-only External Debug Component Identification Registers, Component ID0 through Component ID3.

Table C7-3 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.

C7.17 External Debug Component Identification Register 0

The EDCIDR0 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDCIDR0 is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

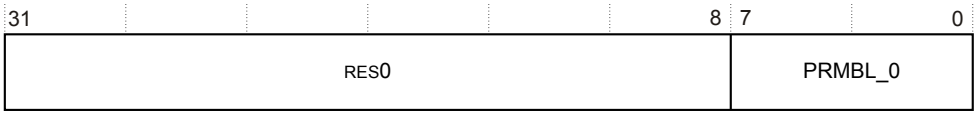

Figure C7-13 EDCIDR0 bit assignments

[31:8]

Reserved, RES0.

PRMBL_0, [7:0]

`0x0D` Preamble byte 0.

The EDCIDR0 can be accessed through the external debug interface, offset `0xFF0`.

C7.18 External Debug Component Identification Register 1

The EDCIDR1 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDCIDR1 is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

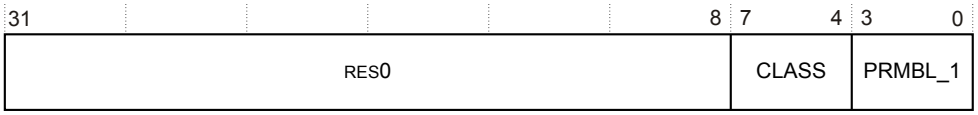

Figure C7-14 EDCIDR1 bit assignments

[31:8]

Reserved, RES0.

CLASS, [7:4]

`0x9` Debug component.

PRMBL_1, [3:0]

`0x0` Preamble.

The EDCIDR1 can be accessed through the external debug interface, offset `0xFF4`.

C7.19 External Debug Component Identification Register 2

The EDCIDR2 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDCIDR2 is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

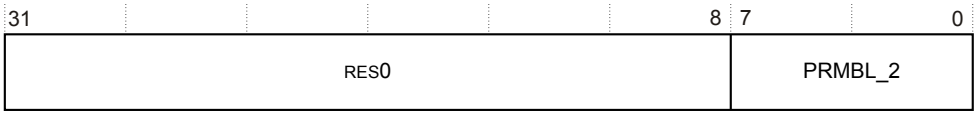

Figure C7-15 EDCIDR2 bit assignments

[31:8]

Reserved, RES0.

PRMBL_2, [7:0]

`0x05` Preamble byte 2.

The EDCIDR2 can be accessed through the external debug interface, offset `0xFF8`.

C7.20 External Debug Component Identification Register 3

The EDCIDR3 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

Table C1-1 Conditions on external register access to debug registers on page C1-369 describes the condition codes.

Configurations

The EDCIDR3 is in the Debug power domain.

Attributes

See *C7.1 Memory-mapped debug register summary* on page C7-424.

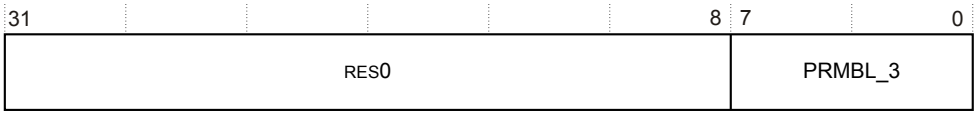

Figure C7-16 EDCIDR3 bit assignments

[31:8]

Reserved, RES0.

PRMBL_3, [7:0]

`0xB1` Preamble byte 3.

The EDCIDR3 can be accessed through the external debug interface, offset `0xFFC`.

Chapter C8
ROM table

This chapter describes the ROM table that debuggers can use to determine which components are implemented. It also describes the ROM table registers.

It contains the following sections:

C8.1 About the ROM table

The processor includes a ROM table that complies with the Arm CoreSight Architecture Specification.

This table contains a list of components such as processor debug units, processor *Cross Trigger Interfaces* (CTIs), processor *Performance Monitoring Units* (PMUs), and processor *Embedded Trace Macrocell* (ETM) trace units. Debuggers can use the ROM table to determine which components are implemented inside the Cortex-A32 processor.

If a component is not included in your configuration of the Cortex-A32 processor, the corresponding debug APB ROM table entry is still present but the component is marked as not present.

C8.2 ROM table register interface

The interface to the ROM table entries is the APB slave port.

See *C1.2 Debug access* on page C1-367.

C8.3 ROM table register summary

The following table shows the offsets from the physical base address of the ROM table.

Table C8-1 ROM table registers

Offset	Name	Type	Description
0x000	ROMENTRY0	RO	*C8.4 ROM entry registers* on page C8-455
0x004	ROMENTRY1	RO	
0x008	ROMENTRY2	RO	
0x00C	ROMENTRY3	RO	
0x010	ROMENTRY4	RO	
0x014	ROMENTRY5	RO	
0x018	ROMENTRY6	RO	
0x01C	ROMENTRY7	RO	
0x020	ROMENTRY8	RO	
0x024	ROMENTRY9	RO	
0x028	ROMENTRY10	RO	
0x02C	ROMENTRY11	RO	
0x030	ROMENTRY12	RO	
0x034	ROMENTRY13	RO	
0x038	ROMENTRY14	RO	
0x03C	ROMENTRY15	RO	
0x040-0xFCC	-	RO	Reserved, RES0
0xFD0	ROMPIDR4	RO	*C8.10 ROM Table Peripheral Identification Register 4* on page C8-464
0xFD4	ROMPIDR5	RO	*C8.11 ROM Table Peripheral Identification Register 5-7* on page C8-465
0xFD8	ROMPIDR6	RO	
0xFDC	ROMPIDR7	RO	
0xFE0	ROMPIDR0	RO	*C8.6 ROM Table Peripheral Identification Register 0* on page C8-460
0xFE4	ROMPIDR1	RO	*C8.7 ROM Table Peripheral Identification Register 1* on page C8-461
0xFE8	ROMPIDR2	RO	*C8.8 ROM Table Peripheral Identification Register 2* on page C8-462
0xFEC	ROMPIDR3	RO	*C8.9 ROM Table Peripheral Identification Register 3* on page C8-463
0xFF0	ROMCIDR0	RO	*C8.13 ROM Table Component Identification Register 0* on page C8-467
0xFF4	ROMCIDR1	RO	*C8.14 ROM Table Component Identification Register 1* on page C8-468
0xFF8	ROMCIDR2	RO	*C8.15 ROM Table Component Identification Register 2* on page C8-469
0xFFC	ROMCIDR3	RO	*C8.16 ROM Table Component Identification Register 3* on page C8-470

C8.4 ROM entry registers

The characteristics of the ROMENTRY*n* are:

Purpose

Indicates to a debugger whether the debug component is present in the debug logic of the processor. There are 16 ROMENTRY registers in the Cortex-A32 processor.

Usage constraints

These registers are accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

C1.4 External access permissions to debug registers on page C1-369 describes the condition codes.

Configurations

Attributes

See *C8.3 ROM table register summary* on page C8-454.

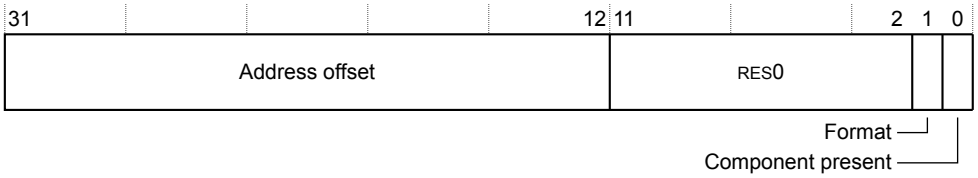

Figure C8-1 ROMENTRY bit assignments

Address offset, [31:12]There is one copy of this register that is used in both Secure and Non-secure states.

Address offset for the debug component.

There is one copy of this register that is used in both Secure andNegative values of address offsets are permitted using the two's complement of the offset.

[11:2]

Reserved, RES0.

Format, [1]

Format of the ROM table entry. The value for all ROMENTRY registers is:

`0` End marker.

`1` 32-bit format.

Component present, [0]

Indicates whether the component is present:

`0` Component is not present.

`1` Component is present.

The debug, CTI, and PMU components for core 0 are always present.

The Physical Address of a debug component is determined by shifting the address offset 12 places to the left and adding the result to the Physical Address of the Cortex-A32 processor ROM table.

The following tables show the offset values for all ROMENTRY values when a v8 memory map is implemented and the offset values for all ROMENTRY values when a legacy v7 memory map is implemented.

If a core is not implemented, the ROMENTRY registers for its debug, CTI, PMU, and ETM trace unit components are `0x00000000` when a v8 memory map is implemented and `0x00000002` when a v7 memory map is implemented.

If a core is implemented but the ETM trace unit is not implemented then the corresponding ROMENTRY register is `0x00000002` in both the v7 and v8 memory maps.

Table C8-2 v8 ROMENTRY values

Name	Debug component	Address offset [31:12]	ROMENTRY value
ROMENTRY0	Core 0 Debug	`0x00010`	`0x00010003`
ROMENTRY1	Core 0 CTI	`0x00020`	`0x00020003`
ROMENTRY2	Core 0 PMU	`0x00030`	`0x00030003`
ROMENTRY3	Core 0 ETM	`0x00040`	`0x00040003` If the component is present.
ROMENTRY4	Core 1 Debug	`0x00110`	`0x00110003` If the component is present.
ROMENTRY5	Core 1 CTI	`0x00120`	`0x00120003` If the component is present.
ROMENTRY6	Core 1 PMU	`0x00130`	`0x00130003` If the component is present.
ROMENTRY7	Core 1 ETM	`0x00140`	`0x00140003` If the component is present.
ROMENTRY8	Core 2 Debug	`0x00210`	`0x00210003` If the component is present.
ROMENTRY9	Core 2 CTI	`0x00220`	`0x00220003` If the component is present.
ROMENTRY10	Core 2 PMU	`0x00230`	`0x00230003` If the component is present.
ROMENTRY11	Core 2 ETM	`0x00240`	`0x00240003` If the component is present.
ROMENTRY12	Core 3 Debug	`0x00310`	`0x00310003` If the component is present.
ROMENTRY13	Core 3 CTI	`0x00320`	`0x00320003` If the component is present.

Table C8-2 v8 ROMENTRY values (continued)

Name	Debug component	Address offset [31:12]	ROMENTRY value
ROMENTRY14	Core 3 PMU	0x00330	0x00330003 If the component is present.
ROMENTRY15	Core 3 ETM	0x00340	0x00340003 If the component is present.

Table C8-3 Legacy v7 ROMENTRY values

Name	Debug component	Address offset [31:12]	ROMENTRY value
ROMENTRY0	Core 0 Debug	0x00010	0x00010003
ROMENTRY1	Core 0 PMU	0x00011	0x00011003
ROMENTRY2	Core 1 Debug	0x00012	0x00012003
ROMENTRY3	Core 1 PMU	0x00013	0x00013003 If the component is present.
ROMENTRY4	Core 2 Debug	0x00014	0x00014003 If the component is present.
ROMENTRY5	Core 2 PMU	0x00015	0x00015003 If the component is present.
ROMENTRY6	Core 3 Debug	0x00016	0x00016003 If the component is present.
ROMENTRY7	Core 3 PMU	0x00017	0x00017003 If the component is present.
ROMENTRY8	Core 0 CTI	0x00018	0x00018003 If the component is present.
ROMENTRY9	Core 1 CTI	0x00019	0x00019003 If the component is present.
ROMENTRY10	Core 2 CTI	0x0001A	0x0001A003 If the component is present.
ROMENTRY11	Core 3 CTI	0x0001B	0x0001B003 If the component is present.
ROMENTRY12	Core 0 ETM	0x0001C	0x0001C003 If the component is present.
ROMENTRY13	Core 1 ETM	0x0001D	0x0001D003 If the component is present.

Table C8-3 Legacy v7 ROMENTRY values (continued)

Name	Debug component	Address offset [31:12]	ROMENTRY value
ROMENTRY14	Core 2 ETM	`0x0001E`	`0x0001E003` If the component is present.
ROMENTRY15	Core 3 ETM	`0x0001F`	`0x0001F003` If the component is present.

C8.5 ROM Table Peripheral Identification Registers

The ROM Table 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 are eight registers listed in register number order in the following table.

Table C8-4 Summary of the ROM Table Peripheral Identification Registers

Register	Value	Offset
ROMPIDR4	`0x04`	`0xFD0`
ROMPIDR5	`0x00`	`0xFD4`
ROMPIDR6	`0x00`	`0xFD8`
ROMPIDR7	`0x00`	`0xFDC`
ROMPIDR0	`0xAB` for v8 memory map. `0xE1` for v7 memory map.	`0xFE0`
ROMPIDR1	`0xB4`	`0xFE4`
ROMPIDR2	`0x2B`	`0xFE8`
ROMPIDR3	`0x00`	`0xFEC`

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

C8.6 ROM Table Peripheral Identification Register 0

The ROMPIDR0 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

C1.4 External access permissions to debug registers on page C1-369 describes the condition codes.

Configurations

The ROMPIDR0 is in the Debug power domain.

Attributes

See *C8.3 ROM table register summary* on page C8-454.

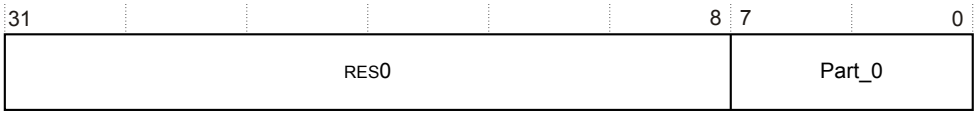

Figure C8-2 ROMPIDR0 bit assignments

[31:8]

Reserved, RES0.

Part_0, [7:0]

Least significant byte of the ROM table part number.

`0xAB` For v8 memory map.

`0xE1` For v7 memory map.

The ROMPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

C8.7 ROM Table Peripheral Identification Register 1

The ROMPIDR1 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

C1.4 External access permissions to debug registers on page C1-369 describes the condition codes.

Configurations

The ROMPIDR1 is in the Debug power domain.

Attributes

See *C8.3 ROM table register summary* on page C8-454.

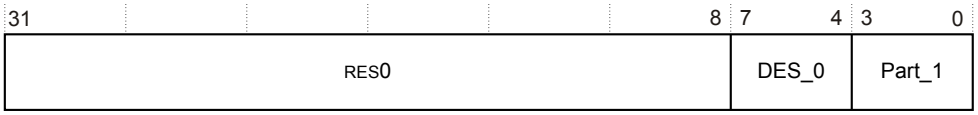

Figure C8-3 ROMPIDR1 bit assignments

[31:8]

Reserved, RES0.

DES_0, [7:4]

`0xB` Least significant nibble of JEP106 ID code. For Arm Limited.

Part_1, [3:0]

`0x4` Most significant nibble of the ROM table part number.

The ROMPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

C8.8 ROM Table Peripheral Identification Register 2

The ROMPIDR2 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

C1.4 External access permissions to debug registers on page C1-369 describes the condition codes.

Configurations

The ROMPIDR2 is in the Debug power domain.

Attributes

See *C8.3 ROM table register summary* on page C8-454.

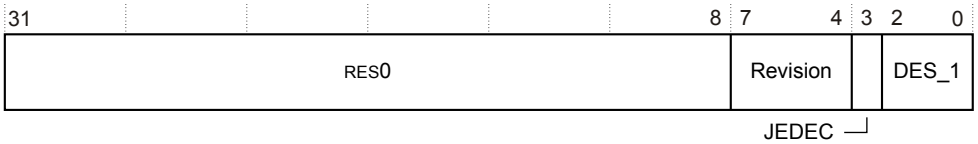

Figure C8-4 ROMPIDR2 bit assignments

[31:8]

Reserved, RES0.

Revision, [7:4]

`0x2` r1p0.

JEDEC, [3]

`0b1` RAO. Indicates a JEP106 identity code is used.

DES_1, [2:0]

`0b011` Designer, most significant bits of JEP106 ID code. For Arm Limited.

The ROMPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

C8.9 ROM Table Peripheral Identification Register 3

The ROMPIDR3 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

C1.4 External access permissions to debug registers on page C1-369 describes the condition codes.

Configurations

The ROMPIDR3 is in the Debug power domain.

Attributes

See *C8.3 ROM table register summary* on page C8-454.

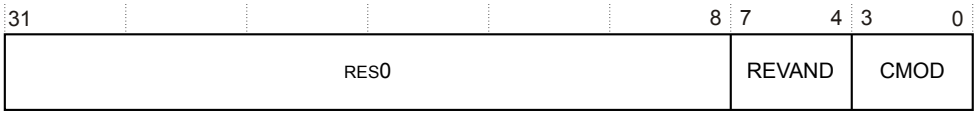

Figure C8-5 ROMPIDR3 bit assignments

[31:8]

Reserved, RES0.

REVAND, [7:4]

`0x0` Part minor revision.

CMOD, [3:0]

`0x0` Customer modified.

The ROMPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

C8.10 ROM Table Peripheral Identification Register 4

The ROMPIDR4 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

C1.4 External access permissions to debug registers on page C1-369 describes the condition codes.

Configurations

The ROMPIDR4 is in the Debug power domain.

Attributes

See *C8.3 ROM table register summary* on page C8-454.

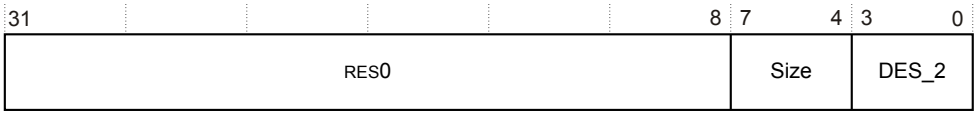

Figure C8-6 ROMPIDR4 bit assignments

[31:8]

Reserved, RES0.

Size, [7:4]

`0x0` Size of the component. Log2 the number of 4KB pages from the start of the component to the end of the component ID registers.

DES_2, [3:0]

`0x4` Designer, JEP106 continuation code, least significant nibble. For Arm Limited.

The ROMPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

C8.11 ROM Table Peripheral Identification Register 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

They are reserved for future use and are RES0.

C8.12 ROM Table Component Identification Registers

There are four read-only ROM Table Component Identification Registers, Component ID0 through Component ID3.

Table C8-5 Summary of the ROM table component Identification registers

Register	Value	Offset
ROMCIDR0	0x0D	0xFF0
ROMCIDR1	0x10	0xFF4
ROMCIDR2	0x05	0xFF8
ROMCIDR3	0xB1	0xFFC

The ROM Table Component Identification Registers identify Debug as an Arm Debug Interface v5 component.

C8.13 ROM Table Component Identification Register 0

The ROMCIDR0 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

C1.4 External access permissions to debug registers on page C1-369 describes the condition codes.

Configurations

The ROMCIDR0 is in the Debug power domain.

Attributes

See *C8.3 ROM table register summary* on page C8-454.

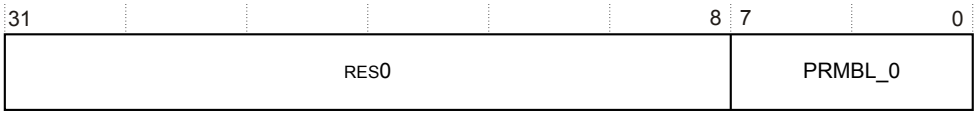

Figure C8-7 ROMCIDR0 bit assignments

[31:8]

Reserved, RES0.

Size, [7:0]

`0x0D` Preamble byte 0.

The ROMCIDR0 can be accessed through the external debug interface, offset `0xFF0`.

C8.14 ROM Table Component Identification Register 1

The ROMCIDR1 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

C1.4 External access permissions to debug registers on page C1-369 describes the condition codes.

Configurations

The ROMCIDR1 is in the Debug power domain.

Attributes

See *C8.3 ROM table register summary* on page C8-454.

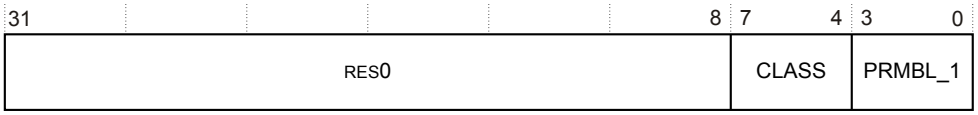

Figure C8-8 ROMCIDR1 bit assignments

[31:8]

Reserved, RES0.

CLASS, [7:4]

`0x1` Component Class. For a ROM table.

PRMBL_1, [3:0]

`0x0` Preamble.

The ROMCIDR1 can be accessed through the external debug interface, offset `0xFF4`.

C8.15 ROM Table Component Identification Register 2

The ROMCIDR2 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

C1.4 External access permissions to debug registers on page C1-369 describes the condition codes.

Configurations

The ROMCIDR2 is in the Debug power domain.

Attributes

See *C8.3 ROM table register summary* on page C8-454.

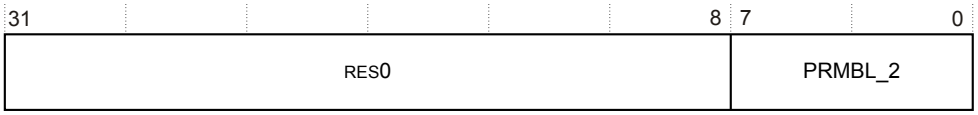

Figure C8-9 ROMCIDR2 bit assignments

[31:8]

Reserved, RES0.

PRMBL_2, [7:0]

`0x05` Preamble byte 2.

The ROMCIDR2 can be accessed through the external debug interface, offset `0xFF8`.

C8.16 ROM Table Component Identification Register 3

The ROMCIDR3 characteristics are:

Purpose

Provides information to identify an external debug component.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	-	RO

C1.4 External access permissions to debug registers on page C1-369 describes the condition codes.

Configurations

The ROMCIDR3 is in the Debug power domain.

Attributes

See *C8.3 ROM table register summary* on page C8-454.

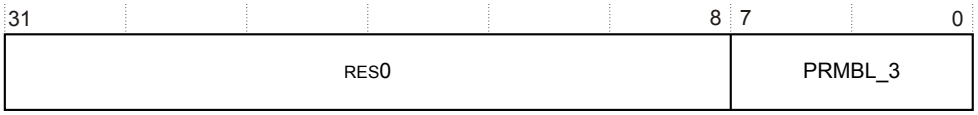

Figure C8-10 ROMCIDR3 bit assignments

[31:8]

Reserved, RES0.

PRMBL_3, [7:0]

`0xB1` Preamble byte 3.

The ROMCIDR3 can be accessed through the external debug interface, offset `0xFFC`.

Chapter C9
PMU registers

This chapter describes the PMU registers.

It contains the following sections:

C9.1 AArch32 PMU register summary

The PMU counters and their associated control registers are accessible in the AArch32 Execution state from the internal CP15 system register interface with `MCR` and `MRC` instructions for 32-bit registers and `MCRR` and `MRRC` for 64-bit registers.

The following table gives a summary of the Cortex-A32 PMU registers in the AArch32 Execution state. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

See the *C9.5 Memory-mapped PMU register summary* on page C9-483 for a complete list of registers that are accessible from the external debug interface.

Table C9-1 PMU register summary in the AArch32 Execution state

CRn	Op1	CRm	Op2	Name	Type	Width	Description
c9	0	c12	0	PMCR	RW	32	*C9.2 Performance Monitors Control Register* on page C9-474
c9	0	c12	1	PMCNTENSET	RW	32	Performance Monitors Count Enable Set Register
c9	0	c12	2	PMCNTENCLR	RW	32	Performance Monitors Count Enable Clear Register
c9	0	c12	3	PMOVSR	RW	32	Performance Monitors Overflow Flag Status Register
c9	0	c12	4	PMSWINC	WO	32	Performance Monitors Software Increment Register
c9	0	c12	5	PMSELR	RW	32	Performance Monitors Event Counter Selection Register
c9	0	c12	6	PMCEID0	RO	32	*C9.3 Performance Monitors Common Event Identification Register 0* on page C9-477
c9	0	c12	7	PMCEID1	RO	32	*C9.4 Performance Monitors Common Event Identification Register 1* on page C9-481
c9	0	c13	0	PMCCNTR[31:0]	RW	32	Performance Monitors Cycle Count Register
-	0	c9	-	PMCCNTR[63:0]	RW	64	
c9	0	c13	1	PMXEVTYPER	RW	32	Performance Monitors Selected Event Type Register
				PMCCFILTR	RW	32	Performance Monitors Cycle Count Filter Register
c9	0	c13	2	PMXEVCNTR	RW	32	Performance Monitors Selected Event Count Register
c9	0	c14	0	PMUSERENR	RW	32	Performance Monitors User Enable Register
c9	0	c14	1	PMINTENSET	RW	32	Performance Monitors Interrupt Enable Set Register
c9	0	c14	2	PMINTENCLR	RW	32	Performance Monitors Interrupt Enable Clear Register
c9	0	c14	3	PMOVSSET	RW	32	Performance Monitor Overflow Flag Status Set Register
c14	0	c8	0	PMEVCNTR0	RW	32	Performance Monitor Event Count Registers
c14	0	c8	1	PMEVCNTR1	RW	32	
c14	0	c8	2	PMEVCNTR2	RW	32	
c14	0	c8	3	PMEVCNTR3	RW	32	
c14	0	c8	4	PMEVCNTR4	RW	32	
c14	0	c8	5	PMEVCNTR5	RW	32	

Table C9-1 PMU register summary in the AArch32 Execution state (continued)

CRn	Op1	CRm	Op2	Name	Type	Width	Description
c14	0	c12	0	PMEVTYPER0	RW	32	Performance Monitors Event Type Registers
c14	0	c12	1	PMEVTYPER1	RW	32	
c14	0	c12	2	PMEVTYPER2	RW	32	
c14	0	c12	3	PMEVTYPER3	RW	32	
c14	0	c12	4	PMEVTYPER4	RW	32	
c14	0	c12	5	PMEVTYPER5	RW	32	
c14	0	c15	7	PMCCFILTR	RW	32	Performance Monitors Cycle Count Filter Register

C9.2 Performance Monitors Control Register

The PMCR characteristics are:

Purpose

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Usage constraints

This register is accessible as follows:

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

This register is accessible at EL0 when PMUSERENR_EL0.EN is set to 1.

Configurations

PMCR[6:0] is architecturally mapped to external PMCR_EL0 register.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

PMCR is a 32-bit register.

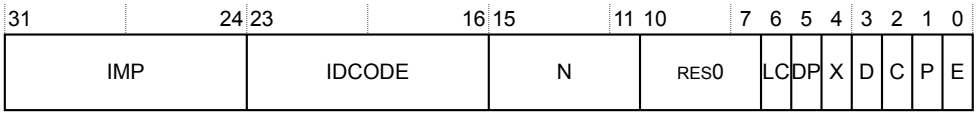

Figure C9-1 PMCR bit assignments

IMP, [31:24]

Implementer code:

`0x41` Arm.

This is a read-only field.

IDCODE, [23:16]

Identification code:

`0x06` Cortex-A32.

This is a read-only field.

N, [15:11]

Number of event counters.

`0b00110` Six counters.

[10:7]

Reserved, RES0.

LC, [6]

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.

`1` Overflow on increment that changes PMCCNTR[63] from 1 to 0.

DP, [5]

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.

X, [4]

Export enable. This bit permits events to be exported to another debug device, such as a trace macrocell, over an event bus:

`0` Export of events is disabled. This is the reset value.

`1` Export of events is enabled.

This bit is read/write and does not affect the generation of Performance Monitors interrupts on the **nPMUIRQ** pin.

D, [3]

Clock divider:

`0` When enabled, PMCCNTR counts every clock cycle. This is the reset value.

`1` When enabled, PMCCNTR counts every 64 clock cycles.

This bit is read/write.

C, [2]

Clock counter reset. This bit is WO. The effects of writing to this bit are:

`0` No action. This is the reset value.

`1` Reset PMCCNTR_EL0 to 0.

This bit is always RAZ.

Resetting PMCCNTR does not clear the PMCCNTR overflow bit to 0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

P, [1]

Event counter reset. This bit is WO. The effects of writing to this bit are:

`0` No action. This is the reset value.

`1` Reset all event counters, not including PMCCNTR, to zero.

This bit is always RAZ.

In Non-secure EL0 and EL1, a write of 1 to this bit does not reset event counters that HDCR.HPMN reserves for EL2 use.

In EL2 and EL3, a write of 1 to this bit resets all the event counters.

Resetting the event counters does not clear any overflow bits to 0.

E, [0]

Enable. The possible values of this bit are:

`0` All counters, including PMCCNTR, are disabled. This is the reset value.

`1` All counters are enabled.

This bit is RW.

In Non-secure EL0 and EL1, this bit does not affect the operation of event counters that HDCR_EL2.HPMN reserves for EL2 use.

On Warm reset, the field resets to 0.

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 external debug interface, offset `0xE04`.

C9.3 Performance Monitors Common Event Identification Register 0

The PMCEID0 characteristics are:

Purpose

Defines which common architectural and common microarchitectural feature events are implemented.

Usage constraints

This register is accessible as follows:

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

This register is accessible at EL0 when PMUSERENR_EL0.EN is set to 1.

Configurations

The PMCEID0 is architecturally mapped to the external register PMCEID0_EL0.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

PMCEID0 is a 32-bit register.

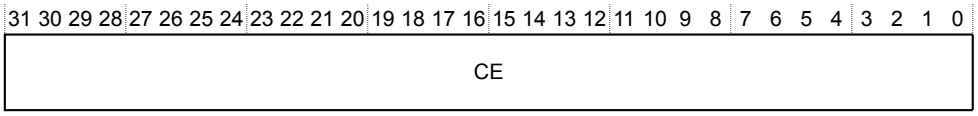

Figure C9-2 PMCEID0 bit assignments

CE[31:0], [31:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

The following table shows the PMCEID0 bit assignments with event implemented or not implemented when the associated bit is set to 1 or 0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about these events.

Table C9-2 PMU events

Bit	Event number	Event mnemonic	Description
[31]	0x1F	L1D_CACHE_ALLOCATE	L1 Data cache allocate: 0 This event is not implemented.
[30]	0x1E	CHAIN	Chain. For odd-numbered counters, counts once for each overflow of the preceding even-numbered counter. For even-numbered counters, does not count: 1 This event is implemented.
[29]	0x1D	BUS_CYCLES	Bus cycle: 1 This event is implemented.

Table C9-2 PMU events (continued)

Bit	Event number	Event mnemonic	Description
[28]	0x1C	TTBR_WRITE_RETIRED	TTBR write, architecturally executed, condition check pass - write to translation table base: 0 This event is not implemented.
[27]	0x1B	INST_SPEC	Instruction speculatively executed: 1 This event is implemented.
[26]	0x1A	MEMORY_ERROR	Local memory error: 1 This event is implemented.
[25]	0x19	BUS_ACCESS	Bus access: 1 This event is implemented.
[24]	0x18	L2D_CACHE_WB	L2 Data cache Write-Back: 0 This event is not implemented if the Cortex-A32 processor has been configured without an L2 cache. 1 This event is implemented if the Cortex-A32 processor has been configured with an L2 cache.
[23]	0x17	L2D_CACHE_REFILL	L2 Data cache refill: 0 This event is not implemented if the Cortex-A32 processor has been configured without an L2 cache. 1 This event is implemented if the Cortex-A32 processor has been configured with an L2 cache.
[22]	0x16	L2D_CACHE	L2 Data cache access: 0 This event is not implemented if the Cortex-A32 processor has been configured without an L2 cache. 1 This event is implemented if the Cortex-A32 processor has been configured with an L2 cache.
[21]	0x15	L1D_CACHE_WB	L1 Data cache Write-Back: 1 This event is implemented.
[20]	0x14	L1I_CACHE	L1 Instruction cache access: 1 This event is implemented.
[19]	0x13	MEM_ACCESS	Data memory access: 1 This event is implemented.
[18]	0x12	BR_PRED	Predictable branch speculatively executed: 1 This event is implemented.

Table C9-2 PMU events (continued)

Bit	Event number	Event mnemonic	Description
[17]	0x11	CPU_CYCLES	Cycle: 1 This event is implemented.
[16]	0x10	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed: 1 This event is implemented.
[15]	0x0F	UNALIGNED_LDST_RETIRED	Instruction architecturally executed, condition check pass - unaligned load or store: 1 This event is implemented.
[14]	0x0E	BR_RETURN_RETIRED	Instruction architecturally executed, condition check pass - procedure return: 0 This event is not implemented.
[13]	0x0D	BR_IMMED_RETIRED	Instruction architecturally executed - immediate branch: 1 This event is implemented.
[12]	0x0C	PC_WRITE_RETIRED	Instruction architecturally executed, condition check pass - software change of the PC: 1 This event is implemented.
[11]	0x0B	CID_WRITE_RETIRED	Instruction architecturally executed, condition check pass - write to CONTEXTIDR: 1 This event is implemented.
[10]	0x0A	EXC_RETURN	Instruction architecturally executed, condition check pass - exception return: 1 This event is implemented.
[9]	0x09	EXC_TAKEN	Exception taken: 1 This event is implemented.
[8]	0x08	INST_RETIRED	Instruction architecturally executed: 1 This event is implemented.
[7]	0x07	ST_RETIRED	Instruction architecturally executed, condition check pass - store: 1 This event is implemented.
[6]	0x06	LD_RETIRED	Instruction architecturally executed, condition check pass - load: 1 This event is implemented.
[5]	0x05	L1D_TLB_REFILL	L1 Data TLB refill: 1 This event is implemented.

Table C9-2 PMU events (continued)

Bit	Event number	Event mnemonic	Description
[4]	0x04	L1D_CACHE	L1 Data cache access: 1 This event is implemented.
[3]	0x03	L1D_CACHE_REFILL	L1 Data cache refill: 1 This event is implemented.
[2]	0x02	L1I_TLB_REFILL	L1 Instruction TLB refill: 1 This event is implemented.
[1]	0x01	L1I_CACHE_REFILL	L1 Instruction cache refill: 1 This event is implemented.
[0]	0x00	SW_INCR	Instruction architecturally executed, condition check pass - software increment: 1 This event is implemented.

To access the PMCEID0:

```
MRC p15,0,<Rt>,c9,c12,6 ; Read PMCEID0 into Rt
```

The PMCEID0 can be accessed through the external debug interface, offset 0xE20.

C9.4 Performance Monitors Common Event Identification Register 1

The PMCEID1 characteristics are:

Purpose

Defines which common architectural and common microarchitectural feature events are implemented.

Usage constraints

This register is accessible as follows:

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

This register is accessible at EL0 when PMUSERENR_EL0.EN is set to 1

Configurations

The PMCEID1 is architecturally mapped to the external register PMCEID1_EL0.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

PMCEID1 is a 32-bit register.

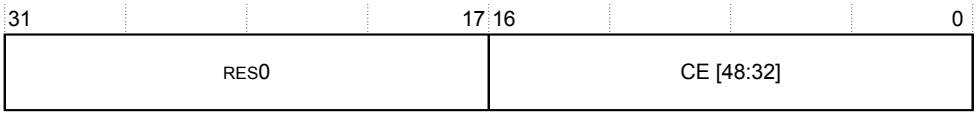

Figure C9-3 PMCEID1 bit assignments

[31:17]

CE[48:32], [16:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

For each bit described in The following table, the event is implemented if the bit is set to 1, or not implemented if the bit is set to 0.

Table C9-3 PMU common events

Bit	Event number	Event mnemonic	Description
[16]	0x30	L2I_TLB	Attributable Level 2 instruction TLB access. 0 This event is not implemented.
[15]	0x2F	L2D_TLB	Attributable Level 2 data or unified TLB access. 0 This event is not implemented.
[14]	0x2E	L2I_TLB_REFILL	Attributable Level 2 instruction TLB refill. 0 This event is not implemented.
[13]	0x2D	L2D_TLB_REFIL	Attributable Level 2 data or unified TLB refill. 0 This event is not implemented.

Table C9-3 PMU common events (continued)

Bit	Event number	Event mnemonic	Description
[12]	0x2C	L3D_CACHE_WB	Attributable Level 3 data or unified cache write-back. 0 This event is not implemented.
[11]	0x2B	L3D_CACHE	Attributable Level 3 data or unified cache access. 0 This event is not implemented.
[10]	0x2A	L3D_CACHE_REFILL	Attributable Level 3 data or unified cache refill. 0 This event is not implemented.
[9]	0x29	L3D_CACHE_ALLOCATE	Attributable Level 3 data or unified cache allocation without refill. 0 This event is not implemented.
[8]	0x28	L2I_CACHE_REFILL	Attributable Level 2 instruction cache refill. 0 This event is not implemented.
[7]	0x27	L2I_CACHE	Attributable Level 2 instruction cache access. 0 This event is not implemented.
[6]	0x26	L1I_TLB	Level 1 instruction TLB access. 0 This event is not implemented.
[5]	0x25	L1D_TLB	Level 1 data or unified TLB access. 0 This event is not implemented.
[4]	0x24	STALL_BACKEND	No operation issued due to backend. 0 This event is not implemented.
[3]	0x23	STALL_FRONTEND	No operation issued due to the frontend. 0 This event is not implemented.
[2]	0x22	BR_MIS_PRED_RETIRED	Instruction architecturally executed, mispredicted branch. 0 This event is not implemented.
[1]	0x21	BR_RETIRED	Instruction architecturally executed, branch. 0 This event is not implemented.
[0]	0x20	L2D_CACHE_ALLOCATE	Level 2 data cache allocation without refill. 0 This event is not implemented.

To access the PMCEID1:

```
MRC p15,0,<Rt>,c9,c12,7 ; Read PMCEID1 into Rt
```

The PMCEID1 can be accessed through the external debug interface, offset 0xE24.

C9.5 Memory-mapped PMU register summary

There are PMU registers that are accessible through the external debug interface.

These registers are listed in the following table. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table C9-4 Memory-mapped PMU register summary

Offset	Name	Type	Description
0x000	PMEVCNTR0_EL0	RW	Performance Monitors Event Count Register 0
0x004	-	-	Reserved
0x008	PMEVCNTR1_EL0	RW	Performance Monitors Event Count Register 1
0x00C	-	-	Reserved
0x010	PMEVCNTR2_EL0	RW	Performance Monitors Event Count Register 2
0x014	-	-	Reserved
0x018	PMEVCNTR3_EL0	RW	Performance Monitors Event Count Register 3
0x01C	-	-	Reserved
0x020	PMEVCNTR4_EL0	RW	Performance Monitors Event Count Register 4
0x024	-	-	Reserved
0x028	PMEVCNTR5_EL0	RW	Performance Monitors Event Count Register 5
0x02C-0xF4	-	-	Reserved
0x0F8	PMCCNTR_EL0[31:0]	RW	Performance Monitors Cycle Count Register
0x0FC	PMCCNTR_EL0[63:32]	RW	
0x100-0x3FC	-	-	Reserved
0x400	PMEVTYPER0_EL0	RW	Performance Monitors Event Type Register
0x404	PMEVTYPER1_EL0	RW	
0x408	PMEVTYPER2_EL0	RW	
0x40C	PMEVTYPER3_EL0	RW	
0x410	PMEVTYPER4_EL0	RW	
0x414	PMEVTYPER5_EL0	RW	
0x418-0x478	-	-	Reserved
0x47C	PMCCFILTR_EL0	RW	Performance Monitors Cycle Count Filter Register
0x480-0xBFC	-	-	Reserved
0xC00	PMCNTENSET_EL0	RW	Performance Monitors Count Enable Set Register
0xC04-0xC1C	-	-	Reserved
0xC20	PMCNTENCLR_EL0	RW	Performance Monitors Count Enable Clear Register
0xC24-0xC3C	-	-	Reserved

Table C9-4 Memory-mapped PMU register summary (continued)

Offset	Name	Type	Description
0xC40	PMINTENSET_EL1	RW	Performance Monitor Interrupt Enable Set Register
0xC44-0xC5C	-	-	Reserved
0xC60	PMINTENCLR_EL1	RW	Performance Monitors Interrupt Enable Clear Register
0xC64-0xC7C	-	-	Reserved
0xC80	PMOVSCLR_EL0	RW	Performance Monitors Overflow Flag Status Register
0xC84-0xC9C	-	-	Reserved
0xCA0	PMSWINC_EL0	WO	Performance Monitor Software Increment Register
0xCA4-0xCBC	-	-	Reserved
0xCC0	PMOVSSET_EL0	RW	Performance Monitors Overflow Flag Status Set Register
0xCC4-0xDFC	-	-	Reserved
0xE00	PMCFGR	RO	*C9.6 Performance Monitors Configuration Register* on page C9-486
0xE04	PMCR_EL0	RW	Performance Monitors Control Register
0xE08-0xE1C	-	-	Reserved
0xE20	PMCEID0_EL0	RO	*C9.3 Performance Monitors Common Event Identification Register 0* on page C9-477
0xE24	PMCEID1_EL0	RO	*C9.4 Performance Monitors Common Event Identification Register 1* on page C9-481
0xE28-0xFA4	-	-	Reserved
0xFA8	PMDEVAFF0	RO	Performance Monitors Device Affinity Register 0, see *B1.97 Multiprocessor Affinity Register* on page B1-309
0xFAC	PMDEVAFF1	RO	Performance Monitors Device Affinity Register 1, RES0
0xFB0	PMLAR	WO	Performance Monitors Lock Access Register
0xFB4	PMLSR	RO	Performance Monitors Lock Status Register
0xFB8	PMAUTHSTATUS	RO	Performance Monitors Authentication Status Register
0xFBC	PMDEVARCH		Performance Monitors Device Architecture Register
0xFC0-0xFC8	-	-	Reserved
0xFCC	PMDEVTYPE	RO	Performance Monitors Device Type Register
0xFD0	PMPIDR4	RO	*C9.12 Performance Monitors Peripheral Identification Register 4* on page C9-493
0xFD4	PMPIDR5	RO	*C9.13 Performance Monitors Peripheral Identification Register 5-7* on page C9-494
0xFD8	PMPIDR6	RO	
0xFDC	PMPIDR7	RO	
0xFE0	PMPIDR0	RO	*C9.8 Performance Monitors Peripheral Identification Register 0* on page C9-489
0xFE4	PMPIDR1	RO	*C9.9 Performance Monitors Peripheral Identification Register 1* on page C9-490
0xFE8	PMPIDR2	RO	*C9.10 Performance Monitors Peripheral Identification Register 2* on page C9-491

Table C9-4 Memory-mapped PMU register summary (continued)

Offset	Name	Type	Description
0xFEC	PMPIDR3	RO	*C9.11 Performance Monitors Peripheral Identification Register 3* on page C9-492
0xFF0	PMCIDR0	RO	*C9.15 Performance Monitors Component Identification Register 0* on page C9-496
0xFF4	PMCIDR1	RO	*C9.16 Performance Monitors Component Identification Register 1* on page C9-497
0xFF8	PMCIDR2	RO	*C9.17 Performance Monitors Component Identification Register 2* on page C9-498
0xFFC	PMCIDR3	RO	*C9.18 Performance Monitors Component Identification Register 3* on page C9-499

C9.6 Performance Monitors Configuration Register

The PMCFGR characteristics are:

Purpose

Contains PMU specific configuration data.

Usage constraints

The accessibility to the PMCFGR by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
Error	Error	Error	Error	RO	RO

C2.2 External register access permissions to the PMU registers on page C2-377 describes the condition codes.

Configurations

The PMCFGR is in the processor power domain.

Attributes

See *C9.5 Memory-mapped PMU register summary* on page C9-483.

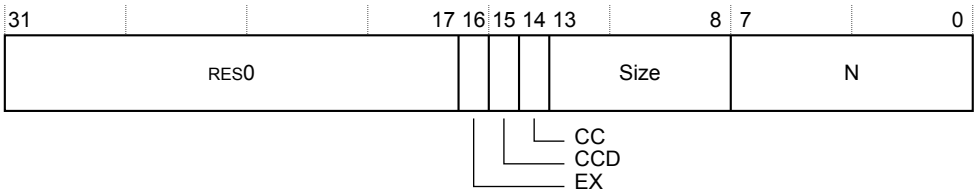

Figure C9-4 PMCFGR bit assignments

[31:17]

Reserved, RES0.

EX, [16]

Export supported. The value is:

`1` Export is supported. PMCR_EL0.EX is read/write.

CCD, [15]

Cycle counter has pre-scale. The value is:

`1` PMCR_EL0.D is read/write.

CC, [14]

Dedicated cycle counter supported. The value is:

`1` Dedicated cycle counter is supported.

Size, [13:8]

Counter size. The value is:

`0b111111` 64-bit counters.

N, [7:0]

Number of event counters. The value is:

`0x06` Six counters.

The PMCFGR can be accessed through the external debug interface, offset `0xE00`.

C9.7 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.

The following table lists the Performance Monitors Peripheral Identification Registers.

Table C9-5 Summary of the Performance Monitors Peripheral Identification Registers

Register	Value	Offset
Peripheral ID4	0x04	0xFD0
Peripheral ID5	0x00	0xFD4
Peripheral ID6	0x00	0xFD8
Peripheral ID7	0x00	0xFDC
Peripheral ID0	0xDB	0xFE0
Peripheral ID1	0xB9	0xFE4
Peripheral ID2	0x2B	0xFE8
Peripheral ID3	0x00	0xFEC

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

C9.8 Performance Monitors Peripheral Identification Register 0

The PMPIDR0 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

Usage constraints

The PMPIDR0 can be accessed through the external debug interface.

The accessibility to the PMPIDR0 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C2.2 External register access permissions to the PMU registers on page C2-377 describes the condition codes.

Configurations

The PMPIDR0 is in the Debug power domain.

Attributes

See the register summary in *C9.5 Memory-mapped PMU register summary* on page C9-483.

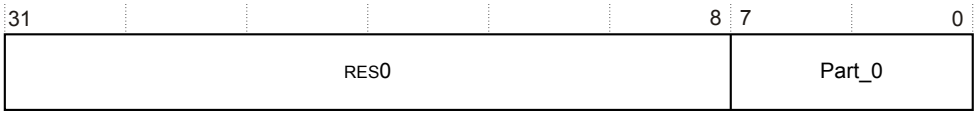

Figure C9-5 PMPIDR0 bit assignments

[31:8]

Reserved, RES0.

Part_0, [7:0]

`0xDB` Least significant byte of the performance monitor part number.

The PMPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

C9.9 Performance Monitors Peripheral Identification Register 1

The PMPIDR1 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

Usage constraints

The PMPIDR1 can be accessed through the external debug interface.

The accessibility to the PMPIDR1 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C2.2 External register access permissions to the PMU registers on page C2-377 describes the condition codes.

Configurations

The PMPIDR1 is in the Debug power domain.

Attributes

See the register summary in *C9.5 Memory-mapped PMU register summary* on page C9-483.

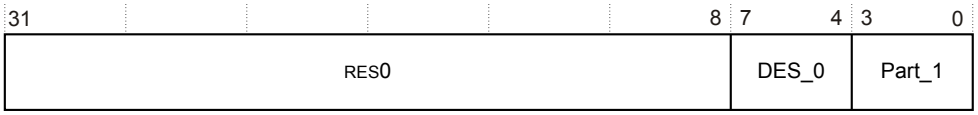

Figure C9-6 PMPIDR1 bit assignments

[31:8]

Reserved, RES0.

DES_0, [7:4]

`0xB` Arm Limited. This is the least significant nibble of JEP106 ID code.

Part_1, [3:0]

`0x9` Most significant nibble of the performance monitor part number.

The PMPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

C9.10 Performance Monitors Peripheral Identification Register 2

The PMPIDR2 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

Usage constraints

The accessibility to the PMPIDR2 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C2.2 External register access permissions to the PMU registers on page C2-377 describes the condition codes.

The PMPIDR2 can be accessed through the external debug interface.

Configurations

The PMPIDR2 is in the Debug power domain.

Attributes

See the register summary in *C9.5 Memory-mapped PMU register summary* on page C9-483.

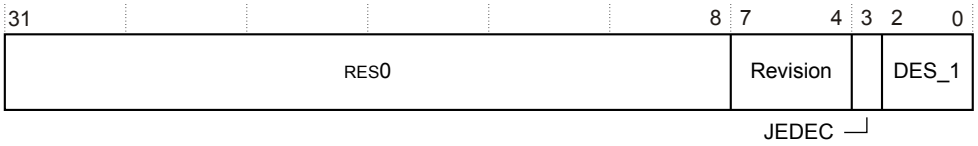

Figure C9-7 PMPIDR2 bit assignments

[31:8]

Reserved, RES0.

Revision, [7:4]

`0x2` r1p0.

JEDEC, [3]

`0b1` RAO. Indicates a JEP106 identity code is used.

DES_1, [2:0]

`0b011` Arm Limited. This is the most significant nibble of JEP106 ID code.

The PMPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

C9.11 Performance Monitors Peripheral Identification Register 3

The PMPIDR3 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

Usage constraints

The PMPIDR3 can be accessed through the external debug interface.

The accessibility to the PMPIDR3 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C2.2 External register access permissions to the PMU registers on page C2-377 describes the condition codes.

Configurations

The PMPIDR3 is in the Debug power domain.

Attributes

See the register summary in *C9.5 Memory-mapped PMU register summary* on page C9-483.

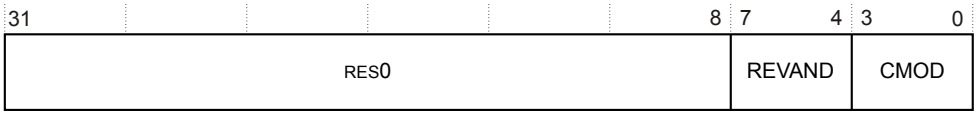

Figure C9-8 PMPIDR3 bit assignments

[31:8]

Reserved, RES0.

REVAND, [7:4]

`0x0` Part minor revision.

CMOD, [3:0]

`0x0` Customer modified.

The PMPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

C9.12 Performance Monitors Peripheral Identification Register 4

The PMPIDR4 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

Usage constraints

The PMPIDR4 can be accessed through the external debug interface.

The accessibility to the PMPIDR4 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C2.2 External register access permissions to the PMU registers on page C2-377 describes the condition codes.

Configurations

The PMPIDR4 is in the Debug power domain.

Attributes

See the register summary in *C9.5 Memory-mapped PMU register summary* on page C9-483.

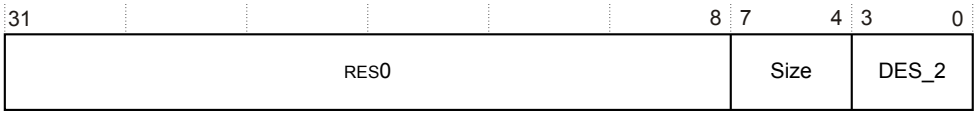

Figure C9-9 PMPIDR4 bit assignments

[31:8]

Reserved, RES0.

Size, [7:4]

`0x0` Size of the component. Log2 the number of 4KB pages from the start of the component to the end of the component ID registers.

DES_2, [3:0]

`0x4` Arm Limited. This is the least significant nibble JEP106 continuation code.

The PMPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

C9.13 Performance Monitors Peripheral Identification Register 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

They are reserved for future use and are RES0.

C9.14 Performance Monitors Component Identification Registers

There are four read-only PMU Component Identification Registers, Component ID0 through Component ID3.

Table C9-6 Summary of the Performance Monitors Component Identification Registers

Register	Value	Offset
Component ID0	0x0D	0xFF0
Component ID1	0x90	0xFF4
Component ID2	0x05	0xFF8
Component ID3	0xB1	0xFFC

The Performance Monitors Component Identification Registers identify Performance Monitor as Arm PMUv3 architecture.

C9.15 Performance Monitors Component Identification Register 0

The PMCIDR0 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

Usage constraints

The PMCIDR0 can be accessed through the external debug interface.

The accessibility to the PMCIDR0 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C2.2 External register access permissions to the PMU registers on page C2-377 describes the condition codes.

Configurations

The PMCIDR0 is in the Debug power domain.

Attributes

See the register summary in *C9.5 Memory-mapped PMU register summary* on page C9-483.

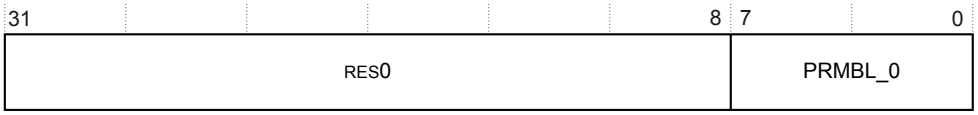

Figure C9-10 PMCIDR0 bit assignments

[31:8]

Reserved, RES0.

Size, [7:0]

`0x0D` Preamble byte 0.

The PMCIDR0 can be accessed through the external debug interface, offset `0xFF0`.

C9.16 Performance Monitors Component Identification Register 1

The PMCIDR1 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

Usage constraints

The PMCIDR1 can be accessed through the external debug interface.

The accessibility to the PMCIDR1 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C2.2 External register access permissions to the PMU registers on page C2-377 describes the condition codes.

Configurations

The PMCIDR1 is in the Debug power domain.

Attributes

See the register summary in *C9.5 Memory-mapped PMU register summary* on page C9-483.

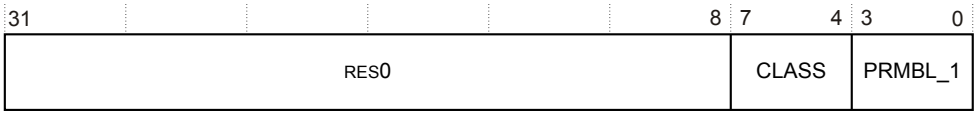

Figure C9-11 PMCIDR1 bit assignments

[31:8]

Reserved, RES0.

CLASS, [7:4]

`0x9` Debug component.

PRMBL_1, [3:0]

`0x0` Preamble byte 1.

The PMCIDR1 can be accessed through the external debug interface, offset `0xFF4`.

C9.17 Performance Monitors Component Identification Register 2

The PMCIDR2 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

Usage constraints

The PMCIDR2 can be accessed through the external debug interface.

The accessibility to the PMCIDR2 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C2.2 External register access permissions to the PMU registers on page C2-377 describes the condition codes.

Configurations

The PMCIDR2 is in the Debug power domain.

Attributes

See the register summary in *C9.5 Memory-mapped PMU register summary* on page C9-483.

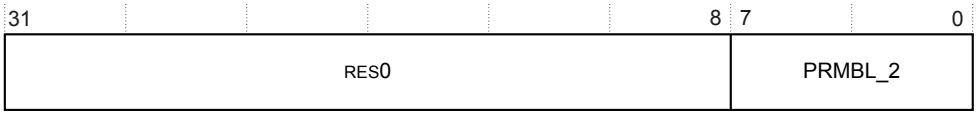

Figure C9-12 PMCIDR2 bit assignments

[31:8]

Reserved, RES0.

PRMBL_2, [7:0]

`0x05` Preamble byte 2.

The PMCIDR2 can be accessed through the external debug interface, offset `0xFF8`.

C9.18 Performance Monitors Component Identification Register 3

The PMCIDR3 characteristics are:

Purpose

Provides information to identify a Performance Monitor component.

Usage constraints

The PMCIDR3 can be accessed through the external debug interface.

The accessibility to the PMCIDR3 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C2.2 External register access permissions to the PMU registers on page C2-377 describes the condition codes.

Configurations

The PMCIDR3 is in the Debug power domain.

Attributes

See the register summary in *C9.5 Memory-mapped PMU register summary* on page C9-483.

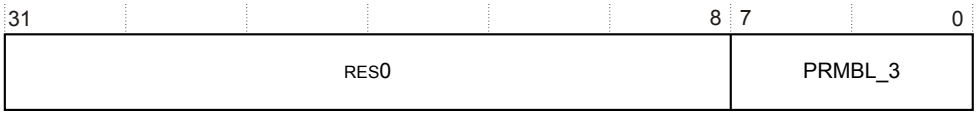

Figure C9-13 PMCIDR3 bit assignments

[31:8]

Reserved, RES0.

PRMBL_3, [7:0]

`0xB1` Preamble byte 3.

The PMCIDR3 can be accessed through the external debug interface, offset `0xFFC`.

Chapter C10
ETM registers

This chapter describes the ETM registers.

It contains the following sections:

C10.1 ETM register summary

This section summarizes the ETM trace unit registers.

All ETM trace unit registers are 32 bits wide. The description of each register includes its offset from a base address. The base address is defined by the system integrator when placing the ETM trace unit in the Debug-APB memory map.

Table C10-1 ETM trace unit register summary

Name	Type	Description
TRCPRGCTLR	RW	*C10.2 Programming Control Register* on page C10-506
TRCSTATR	RO	*C10.3 Status Register* on page C10-507
TRCCONFIGR	RW	*C10.4 Trace Configuration Register* on page C10-508
TRCAUXCTLR	RW	*C10.6 Auxiliary Control Register* on page C10-511
TRCEVENTCTL0R	RW	*C10.7 Event Control 0 Register* on page C10-513
TRCEVENTCTL1R	RW	*C10.8 Event Control 1 Register* on page C10-515
TRCSTALLCTLR	RW	*C10.9 Stall Control Register* on page C10-516
TRCTSCTLR	RW	*C10.10 Global Timestamp Control Register* on page C10-517
TRCSYNCPR	RW	*C10.11 Synchronization Period Register* on page C10-518
TRCCCCTLR	RW	*C10.12 Cycle Count Control Register* on page C10-519
TRCBBCTLR	RW	*C10.5 Branch Broadcast Control Register* on page C10-510
TRCTRACEIDR	RW	*C10.13 Trace ID Register* on page C10-520
TRCVICTLR	RW	*C10.14 ViewInst Main Control Register* on page C10-521
TRCVIIECTLR	RW	*C10.15 ViewInst Include-Exclude Control Register* on page C10-523
TRCVISSCTLR	RW	*C10.16 ViewInst Start-Stop Control Register* on page C10-524
TRCSEQEVR0	RW	*C10.17 Sequencer State Transition Control Registers 0-2* on page C10-525
TRCSEQEVR1	RW	*C10.17 Sequencer State Transition Control Registers 0-2* on page C10-525
TRCSEQEVR2	RW	*C10.17 Sequencer State Transition Control Registers 0-2* on page C10-525
TRCSEQRSTEVR	RW	*C10.18 Sequencer Reset Control Register* on page C10-527
TRCSEQSTR	RW	*C10.19 Sequencer State Register* on page C10-528
TRCEXTINSELR	RW	*C10.20 External Input Select Register* on page C10-529
TRCCNTRLDVR0	RW	*C10.21 Counter Reload Value Registers 0-1* on page C10-530
TRCCNTRLDVR1	RW	*C10.21 Counter Reload Value Registers 0-1* on page C10-530
TRCCNTCTLR0	RW	*C10.22 Counter Control Register 0* on page C10-531
TRCCNTCTLR1	RW	*C10.23 Counter Control Register 1* on page C10-533
TRCCNTVR0	RW	*C10.24 Counter Value Registers 0-1* on page C10-535
TRCCNTVR1	RW	*C10.24 Counter Value Registers 0-1* on page C10-535
TRCIDR8	RO	*C10.25 ID Register 8* on page C10-536
TRCIDR9	RO	*C10.26 ID Register 9* on page C10-537

Table C10-1 ETM trace unit register summary (continued)

Name	Type	Description
TRCIDR10	RO	*C10.27 ID Register 10* on page C10-538
TRCIDR11	RO	*C10.28 ID Register 11* on page C10-539
TRCIDR12	RO	*C10.29 ID Register 12* on page C10-540
TRCIDR13	RO	*C10.30 ID Register 13* on page C10-541
TCRIMSPEC0	RW	*C10.31 Implementation Specific Register 0* on page C10-542
TRCIDR0	RO	*C10.32 ID Register 0* on page C10-543
TRCIDR1	RO	*C10.33 ID Register 1* on page C10-545
TRCIDR2	RO	*C10.34 ID Register 2* on page C10-546
TRCIDR3	RO	*C10.35 ID Register 3* on page C10-548
TRCIDR4	RO	*C10.36 ID Register 4* on page C10-550
TRCIDR5	RO	*C10.37 ID Register 5* on page C10-552
TRCRSCTLRn	RW	*C10.38 Resource Selection Control Registers 2-16* on page C10-554, n is 2, 15
TRCSSCCR0	RW	*C10.39 Single-Shot Comparator Control Register 0* on page C10-555
TRCSSCSR0	RW, RO	*C10.40 Single-Shot Comparator Status Register 0* on page C10-556
TRCOSLAR	WO	*C10.41 OS Lock Access Register* on page C10-557
TRCOSLSR	RO	*C10.42 OS Lock Status Register* on page C10-558
TRCPDCR	RW	*C10.43 Power Down Control Register* on page C10-559
TRCPDSR	RO	*C10.44 Power Down Status Register* on page C10-560
TRCACVRn	RW	*C10.45 Address Comparator Value Registers 0-7* on page C10-561
TRCACATRn	RW	*C10.46 Address Comparator Access Type Registers 0-7* on page C10-562>
TRCCIDCVR0	RW	*C10.47 Context ID Comparator Value Register 0* on page C10-564
TRCVMIDCVR0	RW	*C10.48 VMID Comparator Value Register 0* on page C10-565
TRCCIDCCTLR0	RW	*C10.49 Context ID Comparator Control Register 0* on page C10-566
TRCITATBIDR	RW	*C10.50 Integration ATB Identification Register* on page C10-567
TRCITIDATAR	WO	*C10.51 Integration Instruction ATB Data Register* on page C10-568
TRCITIATBINR	RO	*C10.52 Integration Instruction ATB In Register* on page C10-569
TRCITIATBOUTR	WO	*C10.53 Integration Instruction ATB Out Register* on page C10-570
TRCITCTRL	RW	*C10.54 Integration Mode Control Register* on page C10-571
TRCCLAIMSET	RW	*C10.55 Claim Tag Set Register* on page C10-572
TRCCLAIMCLR	RW	*C10.56 Claim Tag Clear Register* on page C10-573
TRCDEVAFF0	RO	*C10.57 Device Affinity Register 0* on page C10-574
TRCDEVAFF1	RO	*C10.58 Device Affinity Register 1* on page C10-576
TRCLAR	WO	*C10.59 Software Lock Access Register* on page C10-577
TRCLSR	RO	*C10.60 Software Lock Status Register* on page C10-578
TRCAUTHSTATUS	RO	*C10.61 Authentication Status Register* on page C10-579

Table C10-1 ETM trace unit register summary (continued)

Name	Type	Description
TRCDEVARCH	RO	*C10.62 Device Architecture Register* on page C10-580
TRCDEVID	RO	*C10.63 Device ID Register* on page C10-581
TRCDEVTYPE	RO	*C10.64 Device Type Register* on page C10-582
TRCPIDR4	RO	*C10.70 ETM Peripheral Identification Register 4* on page C10-588
TRCPIDR5	RO	*C10.71 ETM Peripheral Identification Register 5-7* on page C10-589
TRCPIDR6	RO	
TRCPIDR7	RO	
TRCPIDR0	RO	*C10.66 ETM Peripheral Identification Register 0* on page C10-584
TRCPIDR1	RO	*C10.67 ETM Peripheral Identification Register 1* on page C10-585
TRCPIDR2	RO	*C10.68 ETM Peripheral Identification Register 2* on page C10-586
TRCPIDR3	RO	*C10.69 ETM Peripheral Identification Register 3* on page C10-587
TRCCIDR0	RO	*C10.73 ETM Component Identification Register 0* on page C10-591
TRCCIDR1	RO	*C10.74 ETM Component Identification Register 1* on page C10-592
TRCCIDR2	RO	*C10.75 ETM Component Identification Register 2* on page C10-593
TRCCIDR3	RO	*C10.76 ETM Component Identification Register 3* on page C10-594

C10.2 Programming Control Register

The TRCPRGCTLR characteristics are:

Purpose

Enables the ETM trace unit.

Usage constraints

See *C3.4 Programming and reading ETM trace unit registers* on page C3-391.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

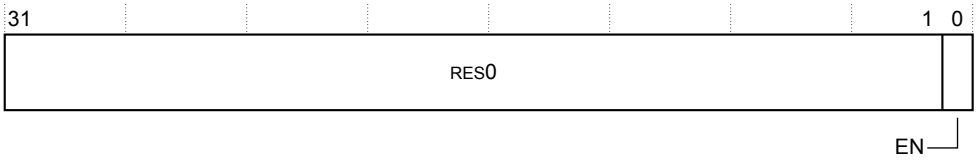

Figure C10-1 TRCPRGCTLR bit assignments

[31:1]

Reserved, RES0.

EN, [0]

Trace program enable:

`0` The ETM trace unit interface in the processor is disabled, and clocks are enabled only when necessary to process APB accesses, or drain any already generated trace. This is the reset value.

`1` The ETM trace unit interface in the processor is enabled, and clocks are enabled. Writes to most trace registers are ignored.

The TRCPRGCTLR can be accessed through the external debug interface, offset `0x004`.

C10.3 Status Register

The TRCSTATR characteristics are:

Purpose

Indicates the ETM trace unit status.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

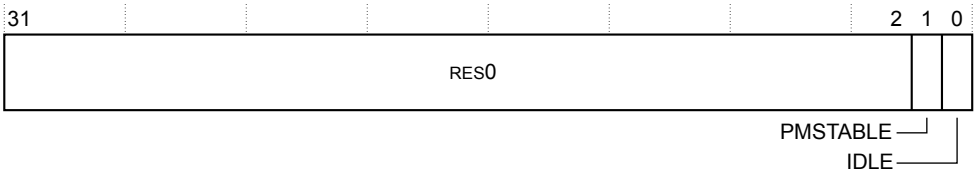

Figure C10-2 TRCSTATR bit assignments

[31:2]

Reserved, RES0.

PMSTABLE, [1]

Indicates whether the ETM trace unit registers are stable and can be read:

`0` The programmers model is not stable.

`1` The programmers model is stable.

IDLE, [0]

Idle status:

`0` The ETM trace unit is not idle.

`1` The ETM trace unit is idle.

The TRCSTATR can be accessed through the external debug interface, offset `0x00C`.

C10.4 Trace Configuration Register

The TRCCONFIGR characteristics are:

Purpose

Controls the tracing options.

Usage constraints

- This register must always be programmed as part of trace unit initialization.
- Only accepts writes when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

TRCCONFIGR is a 32-bit RW trace register.

See *C10.1 ETM register summary* on page C10-503.

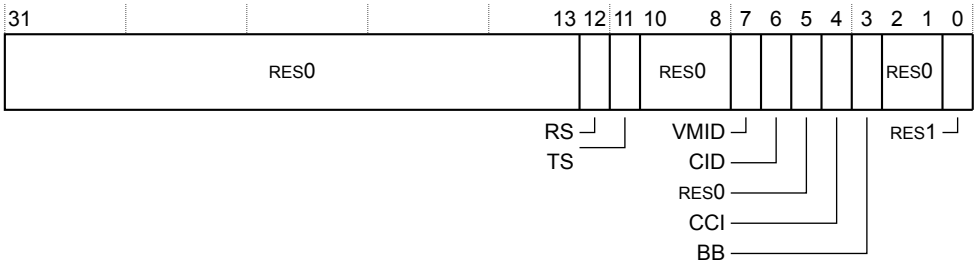

Figure C10-3 TRCCONFIGR bit assignments

[31:13]

Reserved, RES0.

RS, [12]

Enables the return stack. The possible values are:

0 Disables the return stack.

1 Enables the return stack.

TS, [11]

Enables global timestamp tracing. The possible values are:

0 Disables global timestamp tracing.

1 Enables global timestamp tracing.

[10:8]

Reserved, RES0.

VMID, [7]

Enables VMID tracing. The possible values are:

0 Disables VMID tracing.

1 Enables VMID tracing.

CID, [6]

Enables context ID tracing. The possible values are:

0 Disables context ID tracing.

1 Enables context ID tracing.

[5]

Reserved, RES0.

CCI, [4]

Enables cycle counting instruction trace. The possible values are:

`0` Disables cycle counting instruction trace.

`1` Enables cycle counting instruction trace.

BB, [3]

Enables branch broadcast mode. The possible values are:

`0` Disables branch broadcast mode.

`1` Enables branch broadcast mode.

[2:1]

Reserved, RES0.

[0]

Reserved, RES1.

The TRCCONFIGR can be accessed through the external debug interface, offset `0x010`.

C10.5 Branch Broadcast Control Register

The TRCBBCTLR characteristics are:

Purpose

Controls how branch broadcasting behaves, and enables branch broadcasting to be enabled for certain memory regions.

Usage constraints

- Only accepts writes when the trace unit is disabled.
- Must be programmed if TRCCONFIGR.BB == 1.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

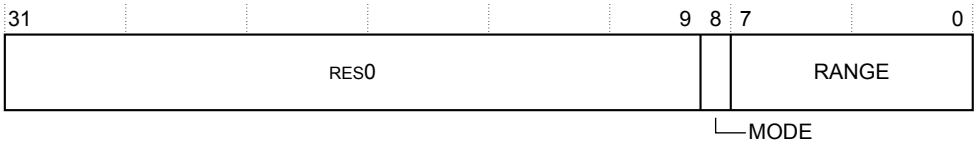

Figure C10-4 TRCBBCTLR bit assignments

[31:9]

Reserved, RES0.

MODE, [8]

Mode bit:

`0` Exclude mode. Branch broadcasting is not enabled in the address range that RANGE defines.

If RANGE==0 then branch broadcasting is enabled for the entire memory map.

`1` Include mode. Branch broadcasting is enabled in the address range that RANGE defines.

If RANGE==0 then the behavior of the trace unit is constrained UNPREDICTABLE. That is, the trace unit might or might not consider any instructions to be in a branch broadcast region.

RANGE, [7:0]

Address range field.

Selects which address range comparator pairs are in use with branch broadcasting. Each bit represents an address range comparator pair, so bit[*n*] controls the selection of address range comparator pair *n*. If bit[*n*] is:

`0` The address range that address range comparator pair *n* defines, is not selected.

`1` The address range that address range comparator pair *n* defines, is selected.

The TRCBBCTLR can be accessed through the external debug interface, offset `0x03C`.

C10.6 Auxiliary Control Register

The TRCAUXCTLR characteristics are:

Purpose

The function of this register is to provide IMPLEMENTATION DEFINED configuration and control options.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

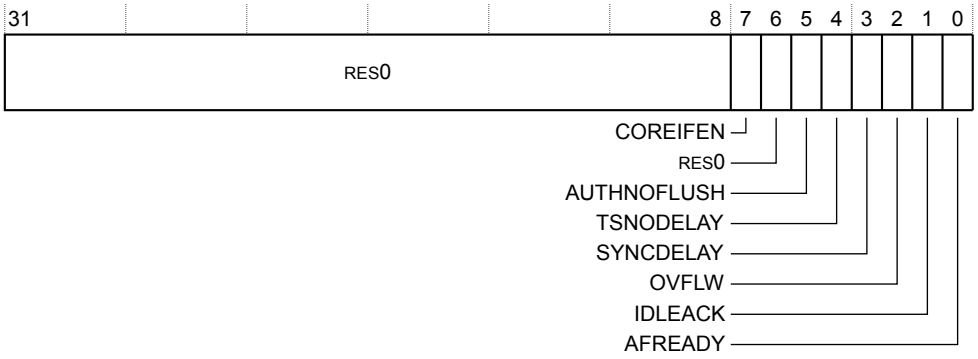

Figure C10-5 TRCAUXCTLR bit assignments

[31:8]

Reserved, RES0.

COREIFEN, [7]

Keep core interface enabled regardless of trace enable register state. The possible values are:

`0` Core interface enabled is set by trace enable register state.

`1` Enable core interface, regardless of trace enable register state.

[6]

Reserved, RES0.

AUTHNOFLUSH, [5]

Do not flush trace on de-assertion of authentication inputs. The possible values are:

`0` ETM trace unit FIFO is flushed and ETM trace unit enters idle state when **DBGEN** or **NIDEN** is LOW.

`1` ETM trace unit FIFO is not flushed and ETM trace unit does not enter idle state when **DBGEN** or **NIDEN** is LOW.

When this bit is set to 1, the trace unit behavior deviates from architecturally-specified behavior.

TSNODELAY, [4]

Do not delay timestamp insertion based on FIFO depth. The possible values are:

`0` Timestamp packets are inserted into FIFO only when trace activity is LOW.

`1` Timestamp packets are inserted into FIFO irrespective of trace activity.

SYNCDELAY, [3]

Delay periodic synchronization if FIFO is more than half-full. The possible values are:

`0` SYNC packets are inserted into FIFO only when trace activity is low.

`1` SYNC packets are inserted into FIFO irrespective of trace activity.

OVFLW, [2]

Force overflow if synchronization is not completed when second synchronization becomes due. The possible values are:

`0` No FIFO overflow when SYNC packets are delayed.

`1` Forces FIFO overflow when SYNC packets are delayed.

When this bit is set to 1, the trace unit behavior deviates from architecturally-specified behavior.

IDLEACK, [1]

Force idle-drain acknowledge high, CPU does not wait for trace to drain before entering WFX state. The possible values are:

`0` ETM trace unit idle acknowledge is asserted only when the ETM trace unit is in idle state.

`1` ETM trace unit idle acknowledge is asserted irrespective of the ETM trace unit idle state.

When this bit is set to 1, trace unit behavior deviates from architecturally-specified behavior.

AFREADY, [0]

Always respond to AFREADY immediately. Does not have any interaction with FIFO draining, even in WFI state. The possible values are:

`0` ETM trace unit **AFREADYM** output is asserted only when the ETM trace unit is in idle state or when all the trace bytes in FIFO before a flush request are output.

`1` ETM trace unit **AFREADYM** output is always asserted HIGH. When this bit is set to 1, trace unit behavior deviates from architecturally-specified behavior.

The TRCAUXCTLR can be accessed through the external debug interface, offset `0x018`.

C10.7 Event Control 0 Register

The TRCEVENTCTL0R characteristics are:

Purpose

Controls the tracing of events in the trace stream. The events also drive the external outputs from the ETM trace unit. The events are selected from the Resource Selectors.

Usage constraints

- You must always program this register as part of trace unit initialization.
- Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

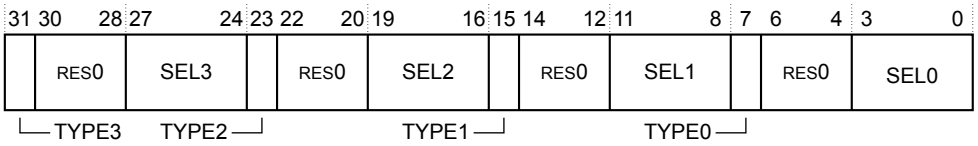

Figure C10-6 TRCEVENTCL0R bit assignments

TYPE3, [31]

Selects the resource type for trace event 3:

0 Single selected resource.

1 Boolean combined resource pair.

[30:28]

Reserved, RES0.

SEL3, [27:24]

Selects the resource number, based on the value of TYPE3:

When TYPE3 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE3 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TYPE2, [23]

Selects the resource type for trace event 2:

0 Single selected resource.

1 Boolean combined resource pair.

[22:20]

Reserved, RES0.

SEL2, [19:16]

Selects the resource number, based on the value of TYPE2:

When TYPE2 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE2 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TYPE1, [15]

Selects the resource type for trace event 1:

0 Single selected resource.

`1` Boolean combined resource pair.

[14:12]

Reserved, RES0.

SEL1, [11:8]

Selects the resource number, based on the value of TYPE1:

When TYPE1 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE1 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TYPE0, [7]

Selects the resource type for trace event 0:

`0` Single selected resource.

`1` Boolean combined resource pair.

[6:4]

Reserved, RES0.

SEL0, [3:0]

Selects the resource number, based on the value of TYPE0:

When TYPE0 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE0 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

The TRCEVENTCTL0R can be accessed through the external debug interface, offset `0x020`.

C10.8 Event Control 1 Register

The TRCEVENTCTL1R characteristics are:

Purpose

Controls the behavior of the events that TRCEVENTCTL0R selects.

Usage constraints

- You must always program this register as part of trace unit initialization.
- Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

TRCEVENTCTL1R is a 32-bit RW trace register.

See *C10.1 ETM register summary* on page C10-503.

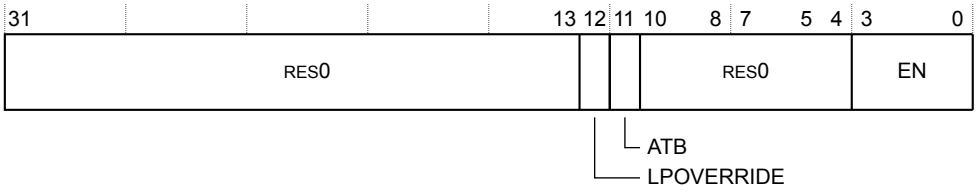

Figure C10-7 TRCEVENTCL1R bit assignments

[31:13]

Reserved, RES0.

LPOVERRIDE, [12]

Low-power state behavior override:

`0` Low-power state behavior unaffected.

`1` Low-power state behavior overridden. The resources and Event trace generation are unaffected by entry to a low-power state.

ATB, [11]

ATB trigger enable:

`0` ATB trigger disabled.

`1` ATB trigger enabled.

[10:4]

Reserved, RES0.

EN, [3:0]

One bit per event, to enable generation of an event element in the instruction trace stream when the selected event occurs:

`0` Event does not cause an event element.

`1` Event causes an event element.

The TRCEVENTCTL1R can be accessed through the external debug interface, offset `0x024`.

C10.9 Stall Control Register

The TRCSTALLCTLR characteristics are:

Purpose

Enables the ETM trace unit to stall the Cortex-A32 processor if the ETM trace unit FIFO overflows.

Usage constraints

- You must always program this register as part of trace unit initialization.
- Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

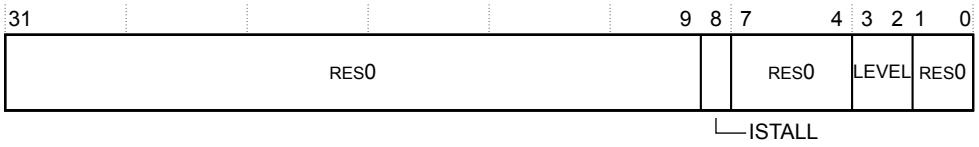

Figure C10-8 TRCSTALLCTLR bit assignments

[31:9]

Reserved, RES0.

ISTALL, [8]

Instruction stall bit. Controls if the trace unit can stall the processor when the instruction trace buffer space is less than LEVEL:

`0` The trace unit does not stall the processor.

`1` The trace unit can stall the processor.

[7:4]

Reserved, RES0.

LEVEL, [3:2]

Threshold level field. The field can support 4 monotonic levels from `0b00` to `0b11`, where:

`0b00` Zero invasion. This setting has a greater risk of an ETM trace unit FIFO overflow.

`0b11` Maximum invasion occurs but there is less risk of a FIFO overflow.

[1:0]

Reserved, RES0.

The TRCSTALLCTLR can be accessed through the external debug interface, offset `0x02c`.

C10.10 Global Timestamp Control Register

The TRCTSCTLR characteristics are:

Purpose

Controls the insertion of global timestamps in the trace streams. When the selected event is triggered, the trace unit inserts a global timestamp into the trace streams. The event is selected from one of the Resource Selectors.

Usage constraints

- Accepts writes only when the trace unit is disabled.
- Must be programmed if TRCCONFIGR.TS==1.

Configurations

Available in all configurations.

Attributes

TRCTSCTLR is a 32-bit RW trace register.

The register is set to an UNKNOWN value on a trace unit reset. See also *C10.1 ETM register summary* on page C10-503.

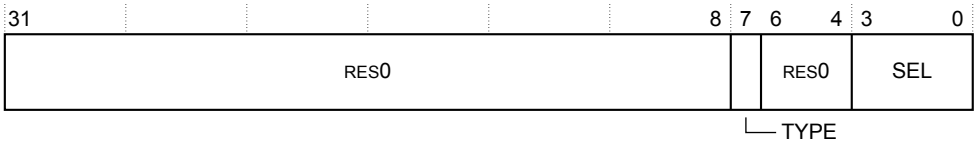

Figure C10-9 TRCTSCTLR bit assignments

[31:8]

Reserved, RES0

TYPE, [7]

Single or combined resource selector.

[6:4]

Reserved.

SEL, [3:1]

Identifies the resource selector to use.

The TRCTSCTLR can be accessed through the external debug interface, offset `0x030`.

C10.11 Synchronization Period Register

The TRCSYNCPR characteristics are:

Purpose

Controls how often periodic trace synchronization requests occur.

Usage constraints

- You must always program this register as part of trace unit initialization.
- Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

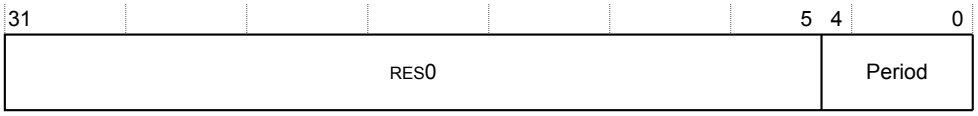

Figure C10-10 TRCSYNCPR bit assignments

[31:5]

Reserved, RES0.

PERIOD, [4:0]

Defines the number of bytes of trace between synchronization requests as a total of the number of bytes generated by both the instruction and data streams. The number of bytes is 2^N where N is the value of this field:

- A value of zero disables these periodic synchronization requests, but does not disable other synchronization requests.
- The minimum value that can be programmed, other than zero, is 8, providing a minimum synchronization period of 256 bytes.
- The maximum value is 20, providing a maximum synchronization period of 2^{20} bytes.

The TRCSYNCPR can be accessed through the external debug interface, offset `0x034`.

C10.12 Cycle Count Control Register

The TRCCCCTLR characteristics are:

Purpose

Sets the threshold value for cycle counting.

Usage constraints

- Accepts writes only when the trace unit is disabled.
- Must be programmed if TRCCONFIGR.CCI==1.
- Minimum value that can be programmed is defined in TRCIDR3.CCITMIN.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

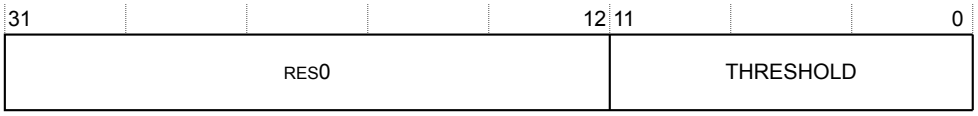

Figure C10-11 TRCCCCTLR bit assignments

[31:12]

Reserved, RES0.

THRESHOLD, [11:0]

Instruction trace cycle count threshold.

The TRCCCCTLR can be accessed through the external debug interface, offset `0x038`.

C10.13 Trace ID Register

The TRCTRACEIDR characteristics are:

Purpose

Sets the trace ID for instruction trace.

Usage constraints

- You must always program this register as part of trace unit initialization.
- Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

TRCTRACEIDR is a 32-bit RW trace register.

See *C10.1 ETM register summary* on page C10-503.

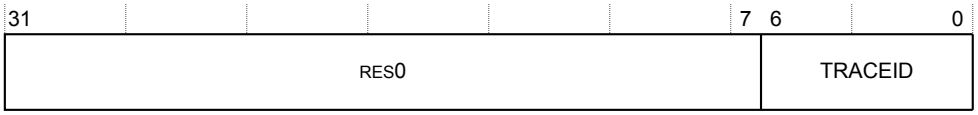

Figure C10-12 TRCTRACEIDR bit Assignments

[31:7]

Reserved, RES0.

TRACEID, [6:0]

Trace ID value. When only instruction tracing is enabled, this provides the trace ID.

The TRCTRACEIDR can be accessed through the external debug interface, offset `0x040`.

C10.14 ViewInst Main Control Register

The TRCVICTLR characteristics are:

Purpose

Controls instruction trace filtering.

Usage constraints

- Accepts writes only when the trace unit is disabled.
- Returns stable data only when TRCSTATR.PMSTABLE==1.
- Must be programmed, particularly to set the value of the SSSTATUS bit, that sets the state of the start-stop logic.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

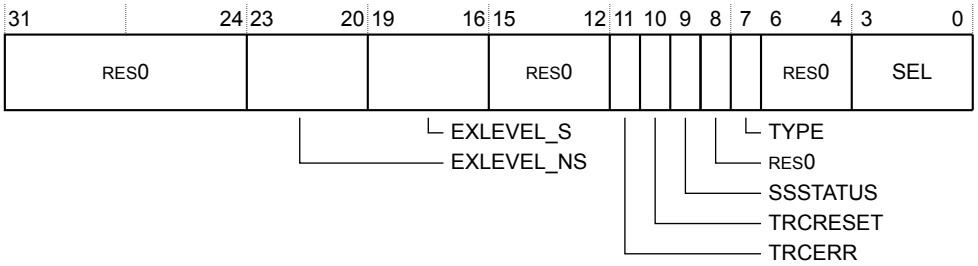

Figure C10-13 TRCVICTLR bit assignments

[31:24]

Reserved, RES0.

EXLEVEL_NS, [23:20]

In Non-secure state, each bit controls whether instruction tracing is enabled for the corresponding exception level:

`0` Trace unit generates instruction trace, in Non-secure state, for exception level *n*.

`1` Trace unit does not generate instruction trace, in Non-secure state, for exception level *n*.

The exception levels are:

Bit[20] Exception level 0.

Bit[21] Exception level 1.

Bit[22] Exception level 2.

Bit[23] RAZ/WI. Instruction tracing is not implemented for exception level 3.

EXLEVEL_S, [19:16]

In Secure state, each bit controls whether instruction tracing is enabled for the corresponding exception level:

`0` Trace unit generates instruction trace, in Secure state, for exception level *n*.

`1` Trace unit does not generate instruction trace, in Secure state, for exception level *n*.

The exception levels are:

Bit[16] Exception level 0.

Bit[17] Exception level 1.

Bit[18] RAZ/WI. Instruction tracing is not implemented for exception level 2.

Bit[19] Exception level 3.

[15:12]

Reserved, RES0.

TRCERR, [11]

Selects whether a system error exception must always be traced:

`0` System error exception is traced only if the instruction or exception immediately before the system error exception is traced.

`1` System error exception is always traced regardless of the value of ViewInst.

TRCRESET, [10]

Selects whether a reset exception must always be traced:

`0` Reset exception is traced only if the instruction or exception immediately before the reset exception is traced.

`1` Reset exception is always traced regardless of the value of ViewInst.

SSSTATUS, [9]

Indicates the current status of the start/stop logic:

`0` Start/stop logic is in the stopped state.

`1` Start/stop logic is in the started state.

[8]

Reserved, RES0.

TYPE, [7]

Selects the resource type for the viewinst event:

`0` Single selected resource.

`1` Boolean combined resource pair.

[6:4]

Reserved, RES0.

SEL, [3:0]

Selects the resource number to use for the viewinst event, based on the value of TYPE:

When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

The TRCVICTLR can be accessed through the external debug interface, offset `0x080`.

C10.15 ViewInst Include-Exclude Control Register

The TRCVIIECTLR characteristics are:

Purpose

Defines the address range comparators that control the ViewInst Include/Exclude control.

Usage constraints

- You must always program this register as part of trace unit initialization.
- Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

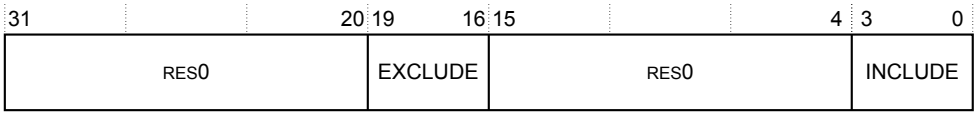

Figure C10-14 TRCVIIECTLR bit assignments

[31:20]

Reserved, RES0.

EXCLUDE, [19:16]

Defines the address range comparators for ViewInst exclude control. One bit is provided for each implemented Address Range Comparator.

[15:4]

Reserved, RES0.

INCLUDE, [3:0]

Defines the address range comparators for ViewInst include control.

Selecting no include comparators indicates that all instructions must be included. The exclude control indicates which ranges must be excluded.

One bit is provided for each implemented Address Range Comparator.

The TRCVIIECTLR can be accessed through the external debug interface, offset `0x084`.

C10.16 ViewInst Start-Stop Control Register

The TRCVISSCTLR characteristics are:

Purpose

Defines the single address comparators that control the ViewInst Start/Stop logic.

Usage constraints

- You must always program this register as part of trace unit initialization.
- Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

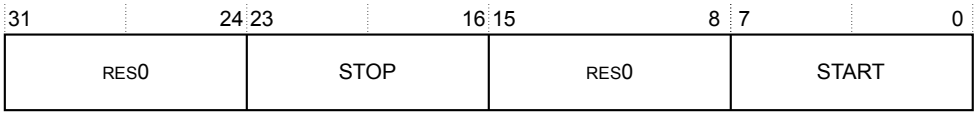

Figure C10-15 TRCVISSCTLR bit assignments

[31:24]

Reserved, RES0.

STOP, [23:16]

Defines the single address comparators to stop trace with the ViewInst Start/Stop control.

One bit is provided for each implemented single address comparator.

[15:8]

Reserved, RES0.

START, [7:0]

Defines the single address comparators to start trace with the ViewInst Start/Stop control.

One bit is provided for each implemented single address comparator.

The TRCVISSCTLR can be accessed through the external debug interface, offset `0x088`.

C10.17 Sequencer State Transition Control Registers 0-2

The TRCSEQEVRn characteristics are:

Purpose

Defines the sequencer transitions that progress to the next state or backwards to the previous state. The ETM trace unit implements a sequencer state machine with up to four states.

Usage constraints

- Accepts writes only when the trace unit is disabled.
- Returns stable data only when TRCSTATR.PMSTABLE==1.
- Software must use this register to set the initial state of the sequencer before the sequencer is used.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

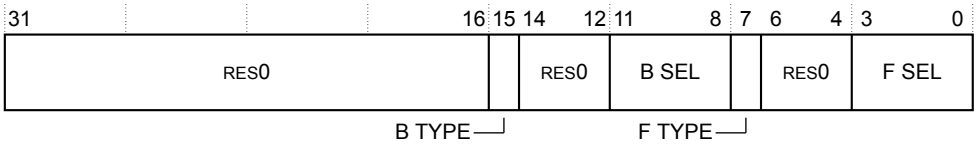

Figure C10-16 TRCSEQEVRn bit assignments

[31:16]

Reserved, RES0.

B TYPE, [15]

Selects the resource type to move backwards to this state from the next state:

0 Single selected resource.

1 Boolean combined resource pair.

[14:12]

Reserved, RES0

B SEL, [11:8]

Selects the resource number, based on the value of B TYPE:

When B TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When B TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

F TYPE, [7]

Selects the resource type to move forwards from this state to the next state:

0 Single selected resource.

1 Boolean combined resource pair.

[6:4]

Reserved, RES0.

F SEL, [3:0]

Selects the resource number, based on the value of F TYPE:

When F TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When F TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

The TRCSEQEVRn registers can be accessed through the external debug interface, offsets:

TRCSEQEVR0
`0x100`.

TRCSEQEVR1
`0x104`.

TRCSEQEVR2
`0x108`.

C10.18 Sequencer Reset Control Register

The TRCSEQRSTEVR characteristics are:

Purpose

Resets the sequencer to state 0.

Usage constraints

- Accepts writes only when the trace unit is disabled.
- If the sequencer is used, you must program all sequencer state transitions with a valid event.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

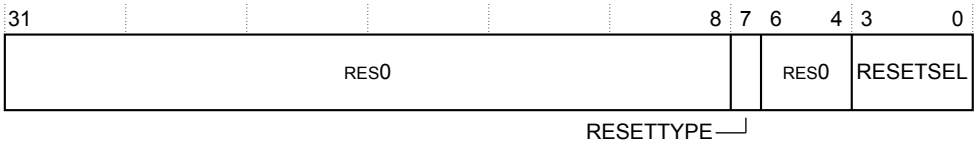

Figure C10-17 TRCSEQRSTEVR bit assignments

[31:8]

Reserved, RES0.

RESETTYPE, [7]

Selects the resource type to move back to state 0:

`0` Single selected resource.

`1` Boolean combined resource pair.

[6:4]

Reserved, RES0.

RESETSEL, [3:0]

Selects the resource number, based on the value of RESETTYPE:

When RESETTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When RESETTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

The TRCSEQRSTEVR can be accessed through the external debug interface, offset `0x118`.

C10.19 Sequencer State Register

The TRCSEQSTR characteristics are:

Purpose

Holds the value of the current state of the sequencer.

Usage constraints

- Accepts writes only when the trace unit is disabled.
- Returns stable data only when TRCSTATR.PMSTABLE==1.
- Software must use this register to set the initial state of the sequencer before the sequencer is used.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

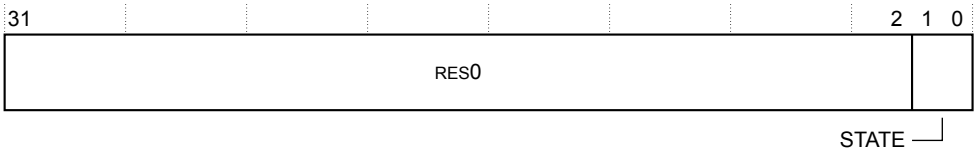

Figure C10-18 TRCSEQSTR bit assignments

[31:2]

Reserved, RES0.

STATE, [1:0]

Current sequencer state:

`0b00` State 0.

`0b01` State 1.

`0b10` State 2.

`0b11` State 3.

The TRCSEQSTR can be accessed through the external debug interface, offset `0x11c`.

C10.20 External Input Select Register

The TRCEXTINSELR characteristics are:

Purpose

Controls the selectors that choose an external input as a resource in the ETM trace unit. You can use the Resource Selectors to access these external input resources.

Usage constraints

Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

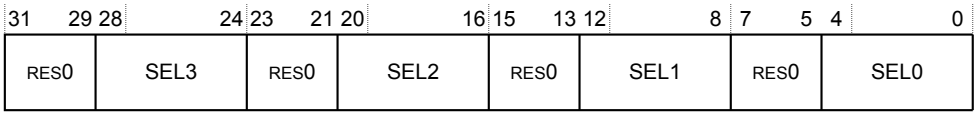

Figure C10-19 TRCEXTINSELR bit assignments

[31:29]

Reserved, RES0.

SEL3, [28:24]

Selects an event from the external input bus for External Input Resource 3.

[23:21]

Reserved, RES0.

SEL2, [20:16]

Selects an event from the external input bus for External Input Resource 2.

[15:13]

Reserved, RES0.

SEL1, [12:8]

Selects an event from the external input bus for External Input Resource 1.

[7:5]

Reserved, RES0.

SEL0, [4:0]

Selects an event from the external input bus for External Input Resource 0.

The TRCEXTINSELR can be accessed through the external debug interface, offset `0x120`.

C10.21 Counter Reload Value Registers 0-1

The TRCCNTRLDVRn characteristics are:

Purpose
Defines the reload value for the counter.

Usage constraints
Accepts writes only when the trace unit is disabled.

Configurations
Available in all configurations.

Attributes
See *C10.1 ETM register summary* on page C10-503.

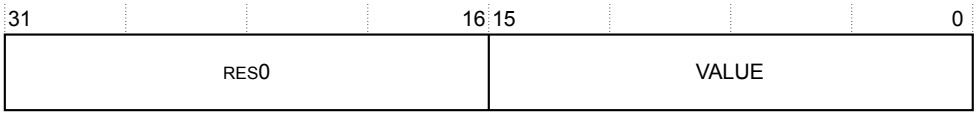

Figure C10-20 TRCCNTRLDVRn bit assignments

[31:16]
Reserved, RES0.

VALUE, [15:0]
Defines the reload value for the counter. This value is loaded into the counter each time the reload event occurs.

The TRCCNTRLDVRn registers can be accessed through the external debug interface, offsets:

TRCCNTRLDVR0
`0x140`.

TRCCNTRLDVR1
`0x144`.

C10.22 Counter Control Register 0

The TRCCNTCTLR0 characteristics are:

Purpose

Controls the counter.

Usage constraints

Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

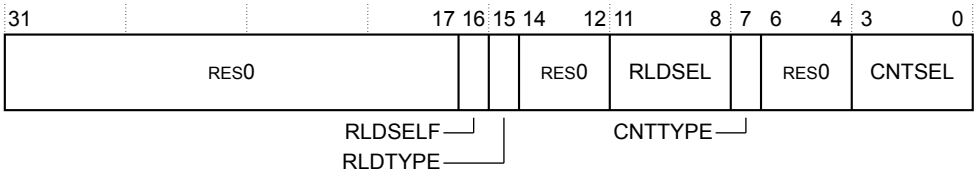

Figure C10-21 TRCCNTCTLR0 bit assignments

[31:17]

Reserved, RES0.

RLDSELF, [16]

Defines whether the counter reloads when it reaches zero:

- `0` The counter does not reload when it reaches zero. The counter only reloads based on RLDTYPE and RLDSEL.
- `1` The counter reloads when it reaches zero and the resource selected by CNTTYPE and CNTSEL is also active. The counter also reloads based on RLDTYPE and RLDSEL.

RLDTYPE, [15]

Selects the resource type for the reload:

- `0` Single selected resource.
- `1` Boolean combined resource pair.

[14:12]

Reserved, RES0.

RLDSEL, [11:8]

Selects the resource number, based on the value of RLDTYPE:

When RLDTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When RLDTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

CNTTYPE, [7]

Selects the resource type for the counter:

- `0` Single selected resource.
- `1` Boolean combined resource pair.

[6:4]

Reserved, RES0.

CNTSEL, [3:0]

Selects the resource number, based on the value of CNTTYPE:

When CNTTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When CNTTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

The TRCCNTCTLR0 can be accessed through the external debug interface, offset `0x150`.

C10.23 Counter Control Register 1

The TRCCNTCTLR1 characteristics are:

Purpose

Controls the counter.

Usage constraints

Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

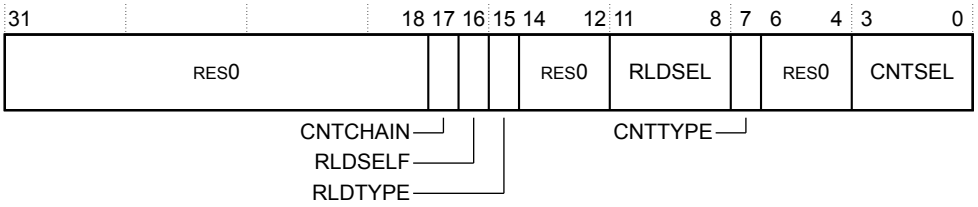

Figure C10-22 TRCCNTCTLR1 bit assignments

[31:18]

Reserved, RES0.

CNTCHAIN, [17]

Defines whether the counter decrements when the counter reloads. This enables two counters to be used in combination to provide a larger counter:

0 The counter operates independently from the counter. The counter only decrements based on CNTTYPE and CNTSEL.

1 The counter decrements when the counter reloads. The counter also decrements when the resource selected by CNTTYPE and CNTSEL is active.

RLDSELF, [16]

Defines whether the counter reloads when it reaches zero:

0 The counter does not reload when it reaches zero. The counter only reloads based on RLDTYPE and RLDSEL.

1 The counter reloads when it is zero and the resource selected by CNTTYPE and CNTSEL is also active. The counter also reloads based on RLDTYPE and RLDSEL.

RLDTYPE, [15]

Selects the resource type for the reload:

0 Single selected resource.

1 Boolean combined resource pair.

[14:12]

Reserved, RES0.

RLDSEL, [11:8]

Selects the resource number, based on the value of RLDTYPE:

When RLDTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When RLDTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

CNTTYPE, [7]

Selects the resource type for the counter:

`0` Single selected resource.

`1` Boolean combined resource pair.

[6:4]

Reserved, RES0.

CNTSEL, [3:0]

Selects the resource number, based on the value of CNTTYPE:

When CNTTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When CNTTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

The TRCCNTCTLR1 can be accessed through the external debug interface, offset `0x154`.

C10.24 Counter Value Registers 0-1

The TRCCNTVRn characteristics are:

Purpose

Contains the current counter value.

Usage constraints

- Can be written only when the ETM trace unit is disabled.
- The count value is stable only when TRCSTATR.PMSTABLE==1.
- If software uses counter <*n*>, then it must write to this register to set the initial counter value.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

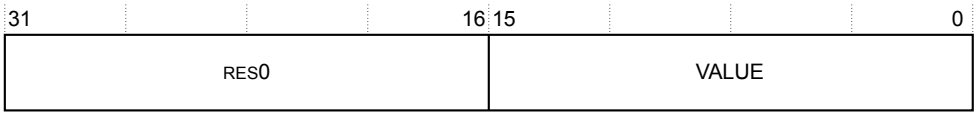

Figure C10-23 TRCCNTVRn bit assignments

[31:16]

Reserved, RES0.

VALUE, [15:0]

Contains the current counter value.

The TRCCNTVRn registers can be accessed through the external debug interface, offsets:

TRCCNTVR0

`0x160`.

TRCCNTVR1

`0x164`.

C10.25 ID Register 8

The TRCIDR8 characteristics are:

Purpose

Returns the maximum speculation depth of the instruction trace stream.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

Figure C10-24 TRCIDR8 bit assignments

MAXSPEC, [31:0]

The maximum number of P0 elements in the trace stream that can be speculative at any time.

`0` Maximum speculation depth of the instruction trace stream.

The TRCIDR8 can be accessed through the external debug interface, offset `0x180`.

C10.26 ID Register 9

The TRCIDR9 characteristics are:

Purpose

Returns the number of P0 right-hand keys that the trace unit can use.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

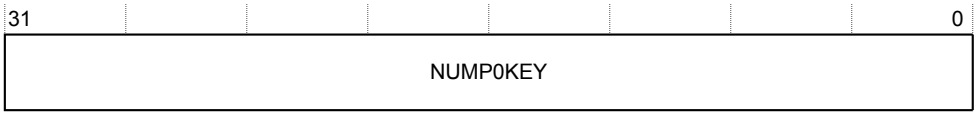

Figure C10-25 TRCID9 bit assignments

NUMP0KEY, [31:0]

The number of P0 right-hand keys that the trace unit can use.

`0` Number of P0 right-hand keys.

The TRCIDR9 can be accessed through the external debug interface, offset `0x184`.

C10.27 ID Register 10

The TRCIDR10 characteristics are:

Purpose

Returns the number of P1 right-hand keys that the trace unit can use.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

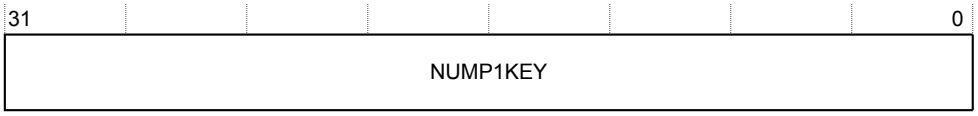

Figure C10-26 TRCIDR10 bit assignments

NUMP1KEY, [31:0]

The number of P1 right-hand keys that the trace unit can use.

`0` Number of P1 right-hand keys.

The TRCIDR10 can be accessed through the external debug interface, offset `0x188`.

C10.28 ID Register 11

The TRCIDR11 characteristics are:

Purpose

Returns the number of special P1 right-hand keys that the trace unit can use.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

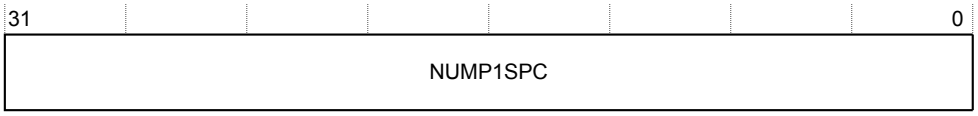

Figure C10-27 TRCIDR11 bit assignments

NUMP1SPC, [31:0]

The number of special P1 right-hand keys that the trace unit can use.

`0` Number of special P1 right-hand keys.

The TRCIDR11 can be accessed through the external debug interface, offset `0x18C`.

C10.29 ID Register 12

The TRCIDR12 characteristics are:

Purpose

Returns the number of conditional instruction right-hand keys that the trace unit can use.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

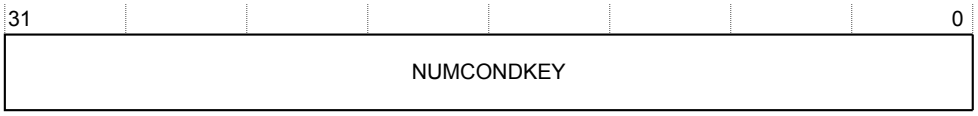

Figure C10-28 TRCIDR12 bit assignments

NUMCONDKEY, [31:0]

The number of conditional instruction right-hand keys that the trace unit can use, including normal and special keys.

`0` Number of conditional instruction right-hand keys.

The TRCIDR12 can be accessed through the external debug interface, offset `0x190`.

C10.30 ID Register 13

The TRCIDR13 characteristics are:

Purpose

Returns the number of special conditional instruction right-hand keys that the trace unit can use.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

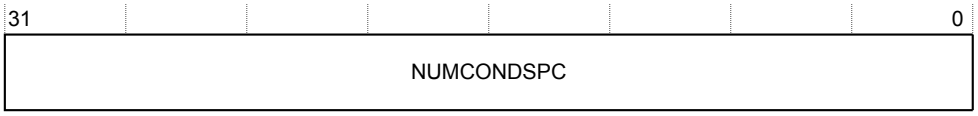

Figure C10-29 TRCIDR13 bit assignments

NUMCONDSPC, [31:0]

The number of special conditional instruction right-hand keys that the trace unit can use, including normal and special keys.

`0` Number of special conditional instruction right-hand keys.

The TRCIDR13 can be accessed through the external debug interface, offset `0x194`.

C10.31 Implementation Specific Register 0

The TRCIMSPEC0 characteristics are:

Purpose

Shows the presence of any implementation specific features, and enables any features that are provided.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

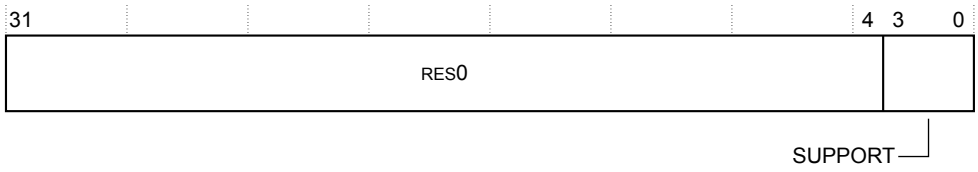

Figure C10-30 TRCIMSPEC0 bit assignments

[31:4]

Reserved, RES0.

SUPPORT, [3:0]

`0` No implementation specific extensions are supported.

The TRCIMSPEC0 can be accessed through the external debug interface, offset `0x1C0`.

C10.32 ID Register 0

The TRCIDR0 characteristics are:

Purpose

Returns the tracing capabilities of the ETM trace unit.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

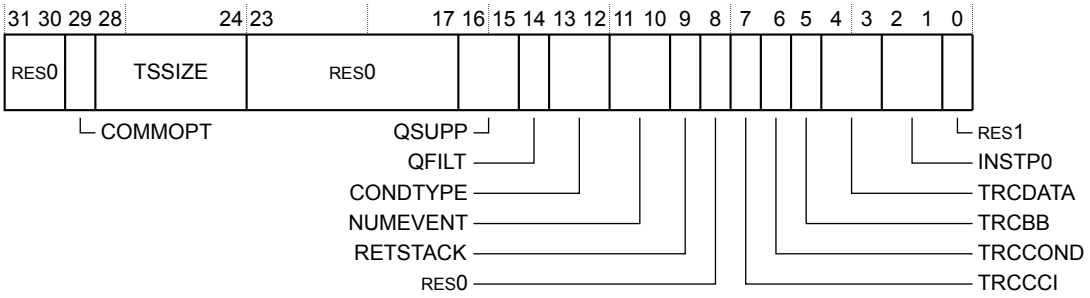

Figure C10-31 TRCIDR0 bit assignments

[31:30]

Reserved, RES0.

COMMOPT, [29]

Indicates the meaning of the commit field in some packets:

`1` Commit mode 1.

TSSIZE, [28:24]

Global timestamp size field:

`0b01000` Implementation supports a maximum global timestamp of 64 bits.

[23:17]

Reserved, RES0.

QSUPP, [16:15]

Indicates Q element support:

`0b00` Q elements not supported.

QFILT, [14]

Indicates Q element filtering support:

`0b0` Q element filtering not supported.

CONDTYPE, [13:12]

Indicates how conditional results are traced:

`0b00` Conditional trace not supported.

NUMEVENT, [11:10]

Number of events supported in the trace, minus 1:

`0b11` Four events supported.

RETSTACK, [9]

Return stack support:

`1` Return stack implemented.

[8]

Reserved, RES0.

TRCCCI, [7]

Support for cycle counting in the instruction trace:

`1` Cycle counting in the instruction trace is implemented.

TRCCOND, [6]

Support for conditional instruction tracing:

`0` Conditional instruction tracing is not supported.

TRCBB, [5]

Support for branch broadcast tracing:

`1` Branch broadcast tracing is implemented.

TRCDATA, [4:3]

Conditional tracing field:

`0b00` Tracing of data addresses and data values is not implemented.

INSTP0, [2:1]

P0 tracing support field:

`0b00` Tracing of load and store instructions as P0 elements is not supported.

[0]

Reserved, RES1.

The TRCIDR0 can be accessed through the external debug interface, offset `0x1E0`.

C10.33 ID Register 1

The TRCIDR1 characteristics are:

Purpose

Returns the base architecture of the trace unit.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

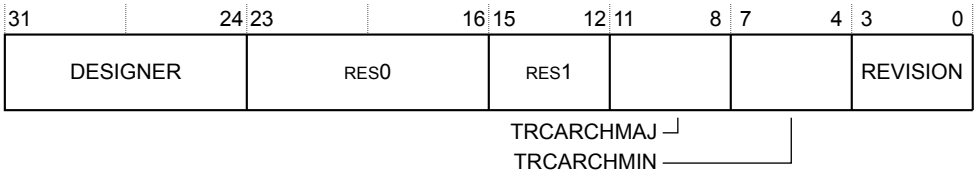

Figure C10-32 TRCIDR1 bit assignments

DESIGNER, [31:24]

Indicates which company designed the trace unit:

`0x41` Arm.

[23:16]

Reserved, RES0.

[15:12]

Reserved, RES1.

TRCARCHMAJ, [11:8]

Major trace unit architecture version number:

`0b0100` ETMv4.

TRCARCHMIN, [7:4]

Minor trace unit architecture version number:

`0b0000` Minor revision 0.

REVISION, [3:0]

Implementation revision number:

`0x2` r1p0.

The TRCIDR1 can be accessed through the external debug interface, offset `0x1E4`.

C10.34 ID Register 2

The TRCIDR2 characteristics are:

Purpose

Returns the maximum size of the following parameters in the trace unit:

- Cycle counter.
- Data value.
- Data address.
- VMID.
- Context ID.
- Instruction address.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

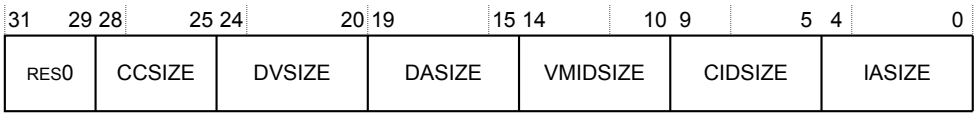

Figure C10-33 TRCIDR2 bit assignments

[31:29]

Reserved, RES0.

CCSIZE, [28:25]

Size of the cycle counter in bits minus 12:

`0x0` The cycle counter is 12 bits in length.

DVSIZE, [24:20]

Data value size in bytes:

`0x00` Data value tracing is not implemented.

DASIZE, [19:15]

Data address size in bytes:

`0x00` Data address tracing is not implemented.

VMIDSIZE, [14:10]

Virtual Machine ID size:

`0x1` Virtual Machine ID is 8 bits.

CIDSIZE, [9:5]

Context ID size in bytes:

`0x4` Maximum of 32-bit Context ID size.

IASIZE, [4:0]

Instruction address size in bytes:

`0x8` Maximum of 64-bit address size.

The TRCIDR2 can be accessed through the external debug interface, offset `0x1E8`.

C10.35 ID Register 3

The TRCIDR3 characteristics are:

Purpose

Indicates:

- Whether TRCVICTLR is supported.
- The number of cores available for tracing.
- If an exception level supports instruction tracing.
- The minimum threshold value for instruction trace cycle counting.
- Whether the synchronization period is fixed.
- Whether TRCSTALLCTLR is supported and if so whether it supports trace overflow prevention and supports stall control of the processor.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

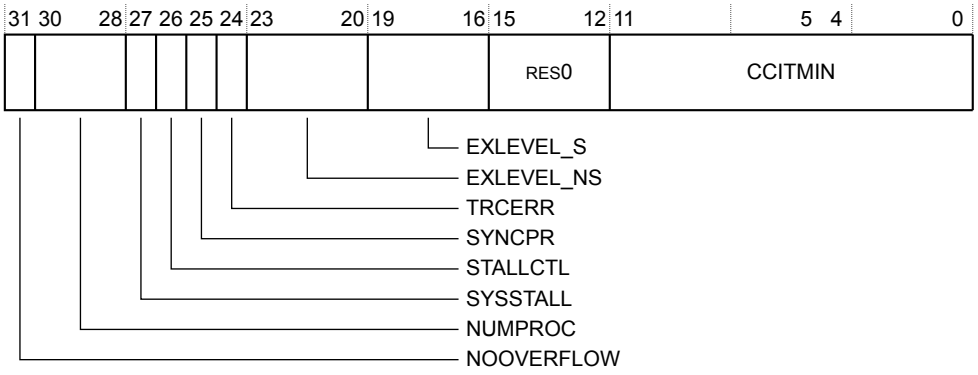

Figure C10-34 TRCIDR3 bit assignments

NOOVERFLOW, [31]

Indicates whether TRCSTALLCTLR.NOOVERFLOW is implemented:

`0` TRCSTALLCTLR.NOOVERFLOW is not implemented.

NUMPROC, [30:28]

Indicates the number of cores available for tracing:

`0b000` The trace unit can trace one processor, ETM trace unit sharing not supported.

SYSSTALL, [27]

Indicates whether stall control is implemented:

`1` The system supports processor stall control.

STALLCTL, [26]

Indicates whether TRCSTALLCTLR is implemented:

`1` TRCSTALLCTLR is implemented.

This field is used in conjunction with SYSSTALL.

SYNCPR, [25]

Indicates whether there is a fixed synchronization period:

`0` TRCSYNCPR is read-write so software can change the synchronization period.

TRCERR, [24]

Indicates whether TRCVICTLR.TRCERR is implemented:

`1` TRCVICTLR.TRCERR is implemented.

EXLEVEL_NS, [23:20]

Each bit controls whether instruction tracing in Non-secure state is implemented for the corresponding exception level:

`0b0111` Instruction tracing is implemented for Non-secure EL0, EL1 and EL2 exception levels.

EXLEVEL_S, [19:16]

Each bit controls whether instruction tracing in Secure state is implemented for the corresponding exception level:

`0b1011` Instruction tracing is implemented for Secure EL0, EL1 and EL3 exception levels.

[15:12]

Reserved, RES0.

CCITMIN, [11:0]

The minimum value that can be programmed in TRCCCCTLR.THRESHOLD:

`0x004` Instruction trace cycle counting minimum threshold is 4.

The TRCIDR3 can be accessed through the external debug interface, offset `0x1EC`.

C10.36 ID Register 4

The TRCIDR4 characteristics are:

Purpose

Indicates the resources available in the ETM trace unit.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

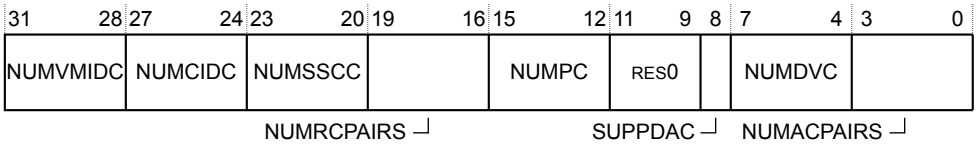

Figure C10-35 TRCIDR4 bit assignments

NUMVMIDC, [31:28]

Indicates the number of VMID comparators available for tracing:

`0x1` One VMID comparator is available.

NUMCIDC, [27:24]

Indicates the number of CID comparators available for tracing:

`0x1` One Context ID comparator is available.

NUMSSCC, [23:20]

Indicates the number of single-shot comparator controls available for tracing:

`0x1` One single-shot comparator control is available.

NUMRSPAIR, [19:16]

Indicates the number of resource selection pairs available for tracing:

`0x7` Eight resource selection pairs are available.

NUMPC, [15:12]

Indicates the number of processor comparator inputs available for tracing:

`0x0` Processor comparator inputs are not implemented.

[11:9]

Reserved, RES0.

SUPPDAC, [8]

Indicates whether the implementation supports data address comparisons: This value is:

`0` Data address comparisons are not implemented.

NUMDVC, [7:4]

Indicates the number of data value comparators available for tracing:

`0x0` Data value comparators not implemented.

NUMACPPAIRS, [3:0]

Indicates the number of address comparator pairs available for tracing:

`0x4` Four address comparator pairs are implemented.

The TRCIDR4 can be accessed through the external debug interface, offset `0x1F0`.

C10.37 ID Register 5

The TRCIDR5 characteristics are:

Purpose

Returns how many resources the trace unit supports.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

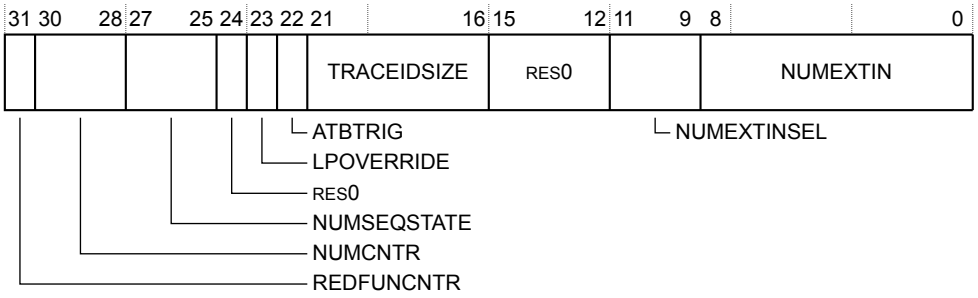

Figure C10-36 TRCIDR5 bit assignments

REDFUNCNTR, [31]

Reduced Function Counter implemented:

`0` Reduced Function Counter not implemented.

NUMCCNTR, [30:28]

Number of counters implemented:

`0b010` Two counters implemented.

NUMSEQSTATE, [27:25]

Number of sequencer states implemented:

`0b100` Four sequencer states implemented.

[24]

Reserved, RES0.

LPOVERRIDE, [23]

Low-power state override support:

`1` Low-power state override support implemented.

ATBTRIG, [22]

ATB trigger support:

`1` ATB trigger support implemented.

TRACEIDSIZE, [21:16]

Number of bits of trace ID:

`0x07` Seven-bit trace ID implemented.

[15:12]

Reserved, RES0.

NUMEXTINSEL, [11:9]

Number of external input selectors implemented:

`0b100` Four external input selectors implemented.

NUMEXTIN, [8:0]

Number of external inputs implemented:

`0x1E` 30 external inputs implemented.

The TRCIDR5 can be accessed through the external debug interface, offset `0x1F4`.

C10.38 Resource Selection Control Registers 2-16

The TRCRSCTLRn characteristics are:

Purpose

Controls the trace resources.

There are eight resource pairs, the first pair is predefined as {0,1,pair=0} and having reserved select registers. This leaves seven pairs to be implemented as programmable selectors.

Usage constraints

Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

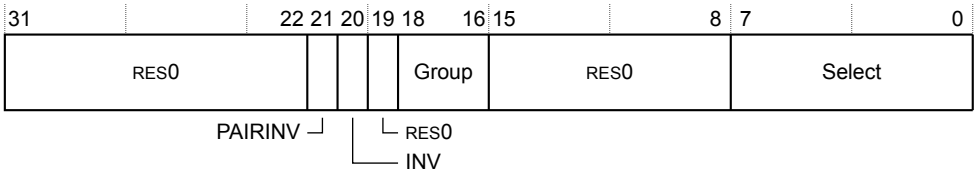

Figure C10-37 TRCRSCTLRn bit assignments

[31:22]

Reserved, RES0.

PAIRINV, [21]

Inverts the result of a combined pair of resources.

This bit is implemented only on the lower register for a pair of resource selectors.

INV, [20]

Inverts the selected resources:

`0` Resource is not inverted.

`1` Resource is inverted.

[19]

Reserved, RES0.

GROUP, [18:16]

Selects a group of resources. See the *Arm® ETM Architecture Specification, ETMv4* for more information.

[15:8]

Reserved, RES0.

SELECT, [7:0]

Selects one or more resources from the required group. One bit is provided for each resource from the group.

The TRCRSCTLRn can be accessed through the external debug interface, offset `0x208-023C`.

C10.39 Single-Shot Comparator Control Register 0

The TRCSSCCR0 characteristics are:

Purpose

Controls the single-shot comparator.

Usage constraints

Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

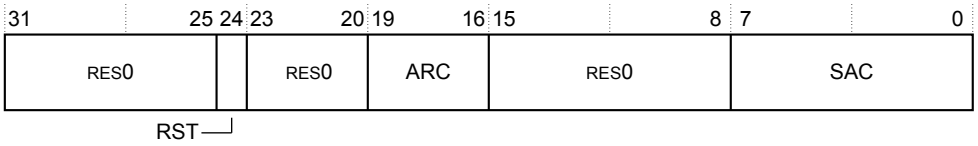

Figure C10-38 TRCSSCCR0 bit assignments

[31:25]

Reserved, RES0.

RST, [24]

Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected:

1 Reset enabled. Multiple matches can occur.

[23:20]

Reserved, RES0.

ARC, [19:16]

Selects one or more address range comparators for single-shot control.

One bit is provided for each implemented address range comparator.

[15:8]

Reserved, RES0.

SAC, [7:0]

Selects one or more single address comparators for single-shot control.

One bit is provided for each implemented single address comparator.

The TRCSSCCR0 can be accessed through the external debug interface, offset `0x280`.

C10.40 Single-Shot Comparator Status Register 0

The TRCSSCSR0 characteristics are:

Purpose

Indicates the status of the single-shot comparator. TRCSSCSR0 is sensitive to instruction addresses.

Usage constraints

- Accepts writes only when the trace unit is disabled.
- The STATUS bit value is stable only when TRCSTATR.PMSTABLE==1.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

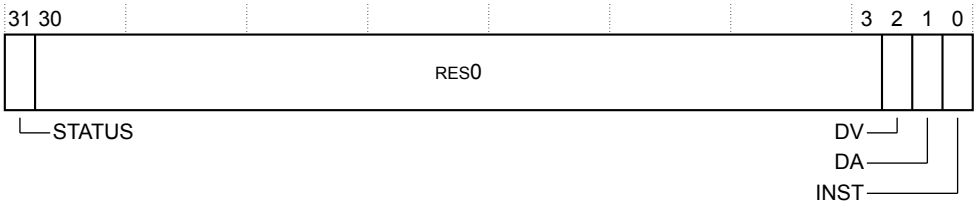

Figure C10-39 TRCSSCSR0 bit assignments

STATUS, [31]

Single-shot status. This indicates whether any of the selected comparators have matched:

`0` Match has not occurred.

`1` Match has occurred at least once.

When programming the ETM trace unit, if TRCSSCCRn.RST is b0, the STATUS bit must be explicitly written to 0 to enable this single-shot comparator control.

[30:3]

Reserved, RES0.

DV, [2]

Data value comparator support:

`0` Single-shot data value comparisons not supported.

DA, [1]

Data address comparator support:

`0` Single-shot data address comparisons not supported.

INST, [0]

Instruction address comparator support:

`1` Single-shot instruction address comparisons supported.

The TRCSSCSR0 can be accessed through the external debug interface, offset `0x2A0`.

C10.41 OS Lock Access Register

The TRCOSLAR characteristics are:

Purpose

Sets and clears the OS Lock, to lock out external debugger accesses to the ETM trace unit registers.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

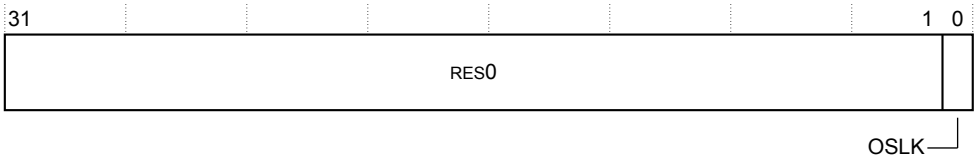

Figure C10-40 TRCOSLAR bit assignments

[31:1]

TRCRSCTLRn

OSLK, [0]

OS Lock key value:

`0` Unlock the OS Lock.

`1` Lock the OS Lock.

The TRCOSLAR can be accessed through the external debug interface, offset `0x300`.

C10.42 OS Lock Status Register

The TRCOSLSR characteristics are:

Purpose

Returns the status of the OS Lock.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

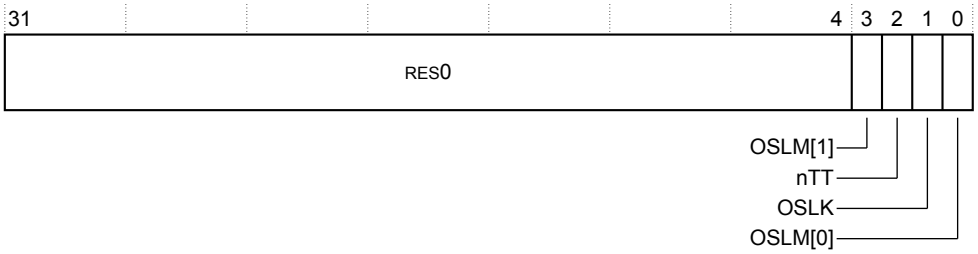

Figure C10-41 TRCOSLSR bit assignments

[31:4]

TRCRSCTLRn

OSLM[1], [3]

OS Lock model [1] bit. This bit is combined with OSLM[0] to form a two-bit field that indicates the OS Lock model is implemented.

The value of this field is always `0b10`, indicating that the OS Lock is implemented.

nTT, [2]

This bit is RAZ, that indicates that software must perform a 32-bit write to update the TRCOSLAR.

OSLK, [1]

OS Lock status bit:

`0` OS Lock is unlocked.

`1` OS Lock is locked.

OSLM[0], [0]

OS Lock model [0] bit. This bit is combined with OSLM[1] to form a two-bit field that indicates the OS Lock model is implemented.

The value of this field is always `0b10`, indicating that the OS Lock is implemented.

The TRCOSLSR can be accessed through the external debug interface, offset `0x304`.

C10.43 Power Down Control Register

The TRCPDCR characteristics are:

Purpose

Request to the system power controller to keep the ETM trace unit powered up.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

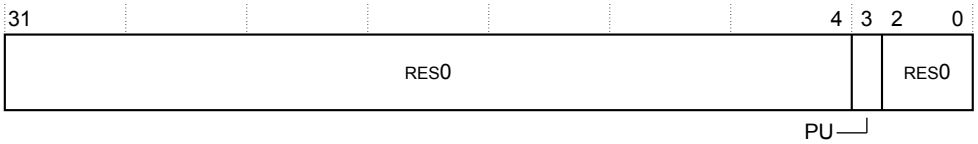

Figure C10-42 TRCPDCR bit assignments

[31:4]

Reserved, RES0.

PU, [3]

Powerup request, to request that power to the ETM trace unit and access to the trace registers is maintained:

`0` Power not requested.

`1` Power requested.

This bit is reset to 0 on a trace unit reset.

[2:0]

Reserved, RES0.

The TRCPDCR can be accessed through the external debug interface, offset `0x310`.

C10.44 Power Down Status Register

The TRCPDSR characteristics are:

Purpose

Indicates the power down status of the ETM trace unit.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

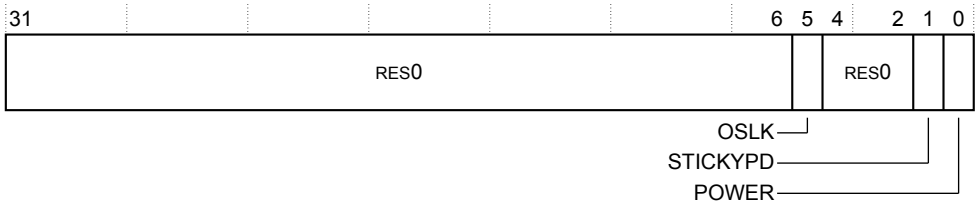

Figure C10-43 TRCPDSR bit assignments

[31:6]

Reserved, RES0.

OSLK, [5]

OS lock status.

`0` The OS Lock is unlocked.

`1` The OS Lock is locked.

[4:2]

Reserved, RES0.

STICKYPD, [1]

Sticky power down state.

`0` Trace register power has not been removed since the TRCPDSR was last read.

`1` Trace register power has been removed since the TRCPDSR was last read.

This bit is set to 1 when power to the ETM trace unit registers is removed, to indicate that programming state has been lost. It is cleared after a read of the TRCPDSR.

POWER, [0]

Indicates the ETM trace unit is powered:

`0` ETM trace unit is not powered. The trace registers are not accessible and they all return an error response.

`1` ETM trace unit is powered. All registers are accessible.

If a system implementation allows the ETM trace unit to be powered off independently of the debug power domain, the system must handle accesses to the ETM trace unit appropriately.

The TRCPDSR can be accessed through the external debug interface, offset `0x314`.

C10.45 Address Comparator Value Registers 0-7

The TRCACVRn characteristics are:

Purpose

Indicates the address for the address comparators.

Usage constraints

Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

Figure C10-44 TRCACVRn bit assignments

ADDRESS, [63:0]

The address value to compare against.

The TRCACVRn can be accessed through the external debug interface, offset `0x400-0x43C`.

C10.46 Address Comparator Access Type Registers 0-7

The TRCACATRn characteristics are:

Purpose

Controls the access for the corresponding address comparators.

Usage constraints

- Accepts writes only when the trace unit is disabled.
- If software uses two single address comparators as an address range comparator then it must program the corresponding TRCACATR registers with identical values in the following fields:
  - TYPE
  - CONTEXTTYPE
  - EXLEVEL_S
  - EXLEVEL_NS

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

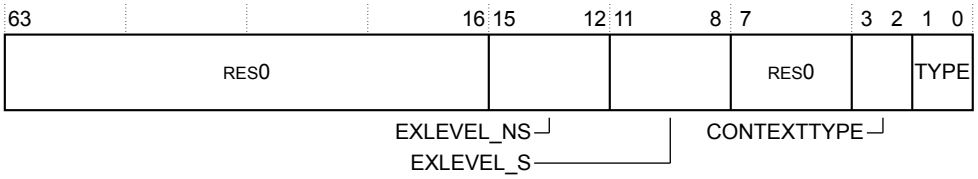

Figure C10-45 TRCACATRn bit assignments

[63:16]

Reserved, RES0.

EXLEVEL_NS, [15:12]

Each bit controls whether a comparison can occur in Non-secure state for the corresponding exception level. The possible values are:

`0` The trace unit can perform a comparison, in Non-secure state, for exception level *n*.

`1` The trace unit does not perform a comparison, in Non-secure state, for exception level *n*.

The exception levels are:

Bit[12] Exception level 0.

Bit[13] Exception level 1.

Bit[14] Exception level 2.

Bit[15] Always RES0.

EXLEVEL_S, [11:8]

Each bit controls whether a comparison can occur in Secure state for the corresponding exception level. The possible values are:

`0` The trace unit can perform a comparison, in Secure state, for exception level *n*.

`1` The trace unit does not perform a comparison, in Secure state, for exception level *n*.

The exception levels are:

Bit[8] Exception level 0.

Bit[9] Exception level 1.

Bit[10] Always RES0.

Bit[11] Exception level 3.

[7:4]

Reserved, RES0.

Context type, [3:2]

Controls whether the trace unit performs a Context ID comparison, a VMID comparison, or both comparisons:

`0b00` The trace unit does not perform a Context ID comparison.

`0b01` The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.

`0b10` The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.

`0b11` The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

Type, [1:0]

The type of comparison:

`0b00` Instruction address, RES0.

The TRCACATR*n* can be accessed through the external debug interface, offset `0x480-0x4B8`.

C10.47 Context ID Comparator Value Register 0

The TRCCIDCVR0 characteristics are:

Purpose

Contains a Context ID value.

Usage constraints

Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

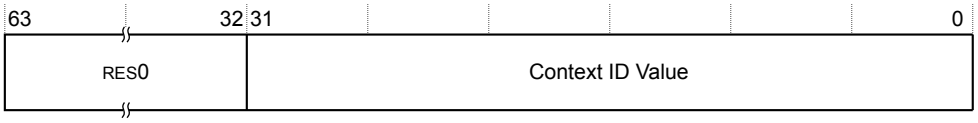

Figure C10-46 TRCCIDCVR0 bit assignments

[63:32]

Reserved, RES0.

VALUE, [31:0]

The data value to compare against.

The TRCCIDCVR0 can be accessed through the external debug interface, offset `0x600`.

C10.48 VMID Comparator Value Register 0

The TRCVMIDCVR0 characteristics are:

Purpose

Contains a VMID value.

Usage constraints

Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

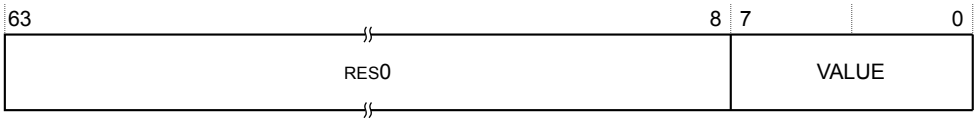

Figure C10-47 TRCVMIDCVR0 bit assignments

[63:8]

Reserved, RES0.

VALUE, [7:0]

The VMID value.

The TRCVMIDCVR0 can be accessed through the external debug interface, offset `0x640`.

C10.49 Context ID Comparator Control Register 0

The TRCCIDCCTLR0 characteristics are:

Purpose

Controls the mask value for the context ID comparators.

Usage constraints

- Accepts writes only when the trace unit is disabled.
- If software uses the TRCCIDCVR*n* registers, where *n*=0 to 3, then it must program this register.
- If software sets a mask bit to 1 then it must program the relevant byte in TRCCIDCVR*n* to `0x00`.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

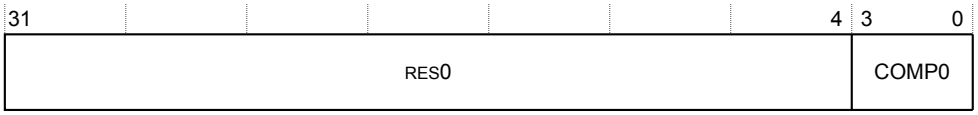

Figure C10-48 TRCCIDCCTLR0 bit assignments

[31:4]

Reserved, RES0.

COMP0, [3:0]

Controls the mask value that the trace unit applies to TRCCIDCVR0. Each bit in this field corresponds to a byte in TRCCIDCVR0. When a bit is:

`0` The trace unit includes the relevant byte in TRCCIDCVR0 when it performs the Context ID comparison.

`1` The trace unit ignores the relevant byte in TRCCIDCVR0 when it performs the Context ID comparison.

The TRCCIDCCTLR0 can be accessed through the external debug interface, offset `0x680`.

C10.50 Integration ATB Identification Register

The TRCITATBIDR characteristics are:

Purpose

Sets the state of output pins shown in the following table.

Usage constraints

- Available when bit[0] of TRCITCTRL is set to 1.
- The value of the register sets the signals on the output pins when the register is written.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

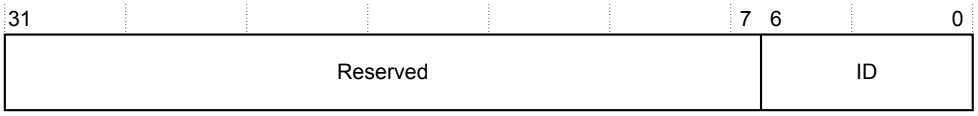

Figure C10-49 TRCITATBIDR bit assignments

[31:7]

Reserved. Read undefined.

ID, [6:0]

Drives the **ATIDMn[6:0]** output pins.

When a bit is set to 0, the corresponding output pin is LOW.

When a bit is set to 1, the corresponding output pin is HIGH.

The TRCITATBIDR bit values correspond to the physical state of the output pins.

The TRCITATBIDR can be accessed through the external debug interface, offset `0xEE4`.

C10.51 Integration Instruction ATB Data Register

The TRCITIDATAR characteristics are:

Purpose

Sets the state of the **ATDATAMn** output pins shown in the following table.

Usage constraints

- Available when bit[0] of TRCITCTRL is set to 1.
- The value of the register sets the signals on the output pins when the register is written.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

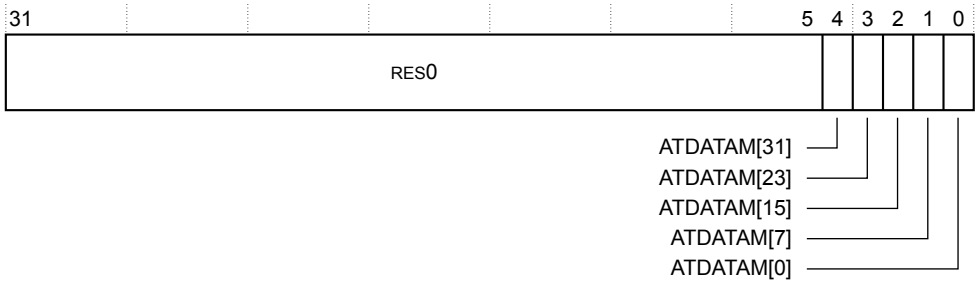

Figure C10-50 TRCITIDATAR bit assignments

For all non-reserved bits:

- When a bit is set to 0, the corresponding output pin is LOW.
- When a bit is set to 1, the corresponding output pin is HIGH.
- The TRCITDDATAR bit values correspond to the physical state of the output pins.

[31:5]

Reserved, RES0.

ATDATAM[31], [4]

Drives the **ATDATAM[31]** output.

ATDATAM[23], [3]

Drives the **ATDATAM[23]** output.

ATDATAM[15], [2]

Drives the **ATDATAM[15]** output.

ATDATAM[7], [1]

Drives the **ATDATAM[7]** output.

ATDATAM[0], [0]

Drives the **ATDATAM[0]** output.

The TRCITIDATAR can be accessed through the external debug interface, offset `0xEEC`.

C10.52 Integration Instruction ATB In Register

The TRCITIATBINR characteristics are:

Purpose

Reads the state of the input pins shown in the following table.

Usage constraints

- Available when bit[0] of TRCITCTRL is set to 1.
- The values of the register bits depend on the signals on the input pins when the register is read.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

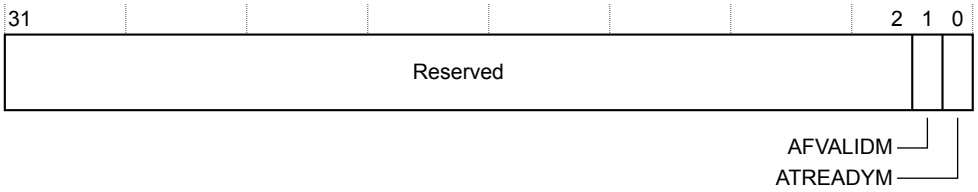

Figure C10-51 TRCITIATBINR bit assignments

For all non-reserved bits:

- When an input pin is LOW, the corresponding register bit is 0.
- When an input pin is HIGH, the corresponding register bit is 1.
- The TRCITIATBINR bit values always correspond to the physical state of the input pins.

[31:2]

Reserved. Read undefined.

AFVALIDM, [1]

Returns the value of the **AFVALIDMn** input pin.

ATREADYM, [0]

Returns the value of the **ATREADYMn** input pin.

The TRCITIATBINR can be accessed through the external debug interface, offset `0xEF4`.

C10.53 Integration Instruction ATB Out Register

The TRCITIATBOUTR characteristics are:

Purpose

Sets the state of the output pins shown in the following table.

Usage constraints

- Available when bit[0] of TRCITCTRL is set to 1.
- The value of the register sets the signals on the output pins when the register is written.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

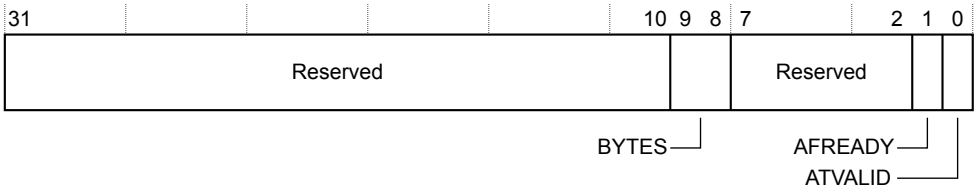

Figure C10-52 TRCITIATBOUTR bit assignments

For all non-reserved bits:

- When a bit is set to 0, the corresponding output pin is LOW.
- When a bit is set to 1, the corresponding output pin is HIGH.
- The TRCITIATBOUTR bit values always correspond to the physical state of the output pins.

[31:10]

Reserved. Read undefined.

BYTES, [9:8]

Drives the **ATBYTESMn[1:0]** output pins.

[7:2]

Reserved. Read undefined.

AFREADY, [1]

Drives the **AFREADYMn** output pin.

ATVALID, [0]

Drives the **ATVALIDMn** output pin.

The TRCITIATBOUTR can be accessed through the external debug interface, offset `0xEFC`.

C10.54 Integration Mode Control Register

The TRCITCTRL characteristics are:

Purpose

Enables topology detection or integration testing, by putting the ETM trace unit into integration mode.

Usage constraints

Arm recommends that you perform a debug reset after using integration mode.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

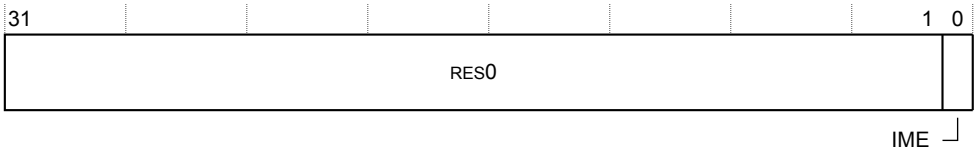

Figure C10-53 TRCITCTRL bit assignments

[31:1]

Reserved, RES0.

IME, [0]

Integration mode enable bit. The possible values are:

`0` The trace unit is not in integration mode.

`1` The trace unit is in integration mode. This mode enables:

- A debug agent to perform topology detection.
- SoC test software to perform integration testing.

The TRCITCTRL can be accessed through the external debug interface, offset `0xF00`.

C10.55 Claim Tag Set Register

The TRCCLAIMSET characteristics are:

Purpose

Sets bits in the claim tag and determines the number of claim tag bits implemented.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

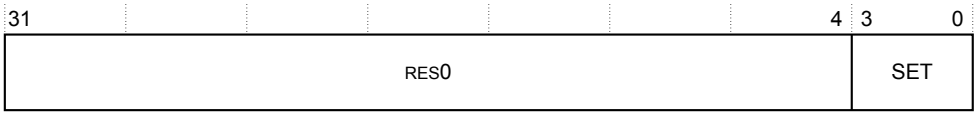

Figure C10-54 TRCCLAIMSET bit assignments

[31:4]

Reserved, RES0.

SET, [3:0]

On reads, for each bit:

`0` Claim tag bit is not implemented.

`1` Claim tag bit is implemented.

On writes, for each bit:

`0` Has no effect.

`1` Sets the relevant bit of the claim tag.

The TRCCLAIMSET can be accessed through the external debug interface, offset `0xFA0`.

C10.56 Claim Tag Clear Register

The TRCCLAIMCLR characteristics are:

Purpose

Clears bits in the claim tag and determines the current value of the claim tag.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

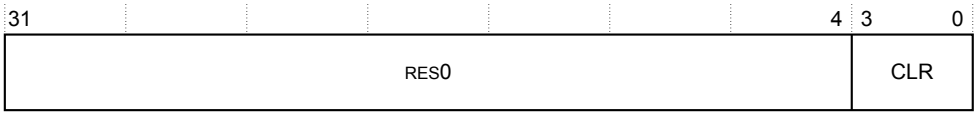

Figure C10-55 TRCCLAIMCLR bit assignments

[31:4]

Reserved, RES0.

CLR, [3:0]

On reads, for each bit:

`0` Claim tag bit is not set.

`1` Claim tag bit is set.

On writes, for each bit:

`0` Has no effect.

`1` Clears the relevant bit of the claim tag.

The TRCCLAIMCLR can be accessed through the external debug interface, offset `0xFA4`.

C10.57 Device Affinity Register 0

The TRCDEVAFF0 characteristics are:

Purpose

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

TRCDEVAFF0 is a read-only copy of MPIDR accessible from the external debug interface.

Usage constraints

This register is accessible as follows:

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

Configurations

The TRCDEVAFF0 is architecturally mapped to external TRCDEVAFF0 register.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

TRCDEVAFF0 is a 32-bit register.

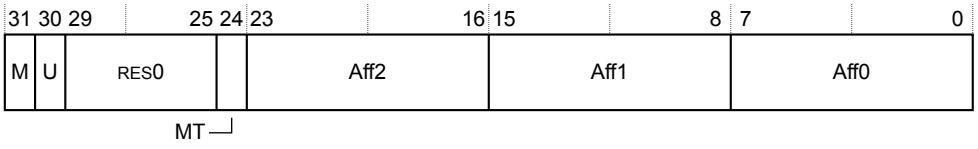

Figure C10-56 TRCDEVAFF0 bit assignments

M, [31]

Reserved, RES1.

U, [30]

Indicates a single core system, as distinct from core 0 in a cluster. This value is:

0 Processor is part of a multiprocessor system. This is the value for implementations with more than one core, and for implementations with an ACE or CHI master interface.

1 Processor is part of a uniprocessor system. This is the value for single core implementations with an AXI master interface.

[29:25]

Reserved, RES0.

MT, [24]

Indicates whether the lowest level of affinity consists of logical cores that are implemented using a multi-threading type approach. This value is:

0 Performance of cores at the lowest affinity level is largely independent.

Aff2, [23:16]

Affinity level 2. Second highest level affinity field.

Indicates the value read in the **CLUSTERIDAFF2** configuration signal.

Aff1, [15:8]

Affinity level 1. Third highest level affinity field.

Indicates the value read in the **CLUSTERIDAFF1** configuration signal.

Aff0, [7:0]

Affinity level 0. Lowest level affinity field.

Indicates the core number in the Cortex-A32 processor. The possible values are:

`0x0`	A processor with one core only.
`0x0`, `0x1`	A cluster with two cores.
`0x0`, `0x1`, `0x2`	A cluster with three cores.
`0x0`, `0x1`, `0x2`, `0x3`	A cluster with four cores.

To access the TRCDEVAFF0:

```
MRC p15,0,<Rt>,c0,c0,5 ; Read TRCDEVAFF0 into Rt
```

Register access is encoded as follows:

Table C10-2 TRCDEVAFF0 access encoding

coproc	opc1	CRn	CRm	opc2
1111	000	0000	0000	101

The TRCDEVAFF0 can be accessed through the external debug interface, offset `0xFA8`.

C10.58 Device Affinity Register 1

The TRCDEVAFF1 characteristics are:

Purpose

The value is a read-only copy of MPIDR_EL1[63:32] as seen from EL3, unaffected by VMPIDR_EL2.

Usage constraints

Accessible only from the external debug interface.

Configurations

Available in all configurations.

Attributes

TRCDEVAFF1 is a 32-bit RO management register.

For the Cortex-A32 processor, MPIDR_EL1[63:32] is RES0.

See *C10.1 ETM register summary* on page C10-503.

The TRCDEVAFF1 can be accessed through the external debug interface, offset `0xFAC`.

C10.59 Software Lock Access Register

The TRCLAR characteristics are:

Purpose

Controls access to registers using the memory-mapped interface, when **PADDRDBG31** is LOW.

When the software lock is set, write accesses using the memory-mapped interface to all ETM trace unit registers are ignored, except for write accesses to the TRCLAR.

When the software lock is set, read accesses of TRCPDSR do not change the TRCPDSR.STICKYPD bit. Read accesses of all other registers are not affected.

Usage constraints

Accessible only from the memory-mapped interface.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

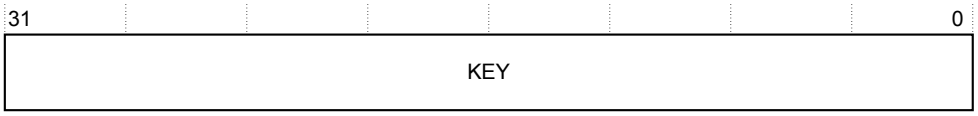

Figure C10-57 TRCLAR bit assignments

KEY, [31:0]

Software lock key value:

`0xC5ACCE55` Clear the software lock.

All other write values set the software lock.

The TRCLAR can be accessed through the external debug interface, offset `0xFB0`.

C10.60 Software Lock Status Register

The TRCLSR characteristics are:

Purpose

Determines whether the software lock is implemented, and indicates the current status of the software lock.

Usage constraints

Accessible only from the external debug interface.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

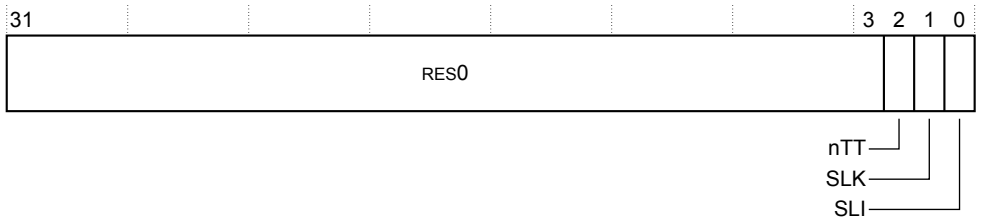

Figure C10-58 TRCLSR bit assignments

[31:3]

Reserved, RES0.

nTT, [2]

Indicates size of TRCLAR:

`0` TRCLAR is always 32 bits.

SLK, [1]

Software lock status:

`0` Software lock is clear.

`1` Software lock is set.

SLI, [0]

Indicates whether the software lock is implemented on this interface.

`1` Software lock is implemented on this interface.

The TRCLSR can be accessed through the external debug interface, offset `0xFB4`.

C10.61 Authentication Status Register

The TRCAUTHSTATUS characteristics are:

Purpose

Indicates the current level of tracing permitted by the system.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

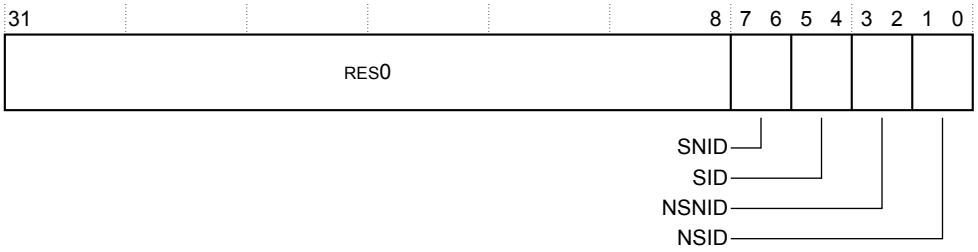

Figure C10-59 TRCAUTHSTATUS bit assignments

[31:8]

Reserved, RES0.

SNID, [7:6]

Secure Non-invasive Debug:

`0b10` Secure Non-invasive Debug implemented but disabled.

`0b11` Secure Non-invasive Debug implemented and enabled.

SID, [5:4]

Secure Invasive Debug:

`0b00` Secure Invasive Debug is not implemented.

NSNID, [3:2]

Non-secure Non-invasive Debug:

`0b10` Non-secure Non-invasive Debug implemented but disabled, **NIDEN**=0.

`0b11` Non-secure Non-invasive Debug implemented and enabled, **NIDEN**=1.

NSID, [1:0]

Non-secure Invasive Debug:

`0b00` Non-secure Invasive Debug is not implemented.

The TRCAUTHSTATUS can be accessed through the external debug interface, offset `0xFB8`.

C10.62 Device Architecture Register

The TRCDEVARCH characteristics are:

Purpose

Identifies the ETM trace unit as an ETMv4 component.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

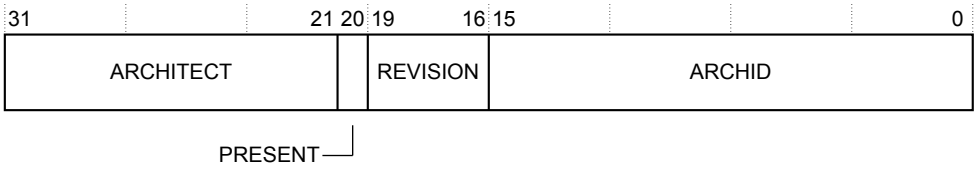

Figure C10-60 TRCDEVARCH bit assignments

ARCHITECT, [31:21]

Defines the architect of the component:

`0x4` Arm JEP continuation.

`0x3B` Arm JEP 106 code.

PRESENT, [20]

Indicates the presence of this register:

`0b1` Register is present.

REVISION, [19:16]

Architecture revision:

`0b0000` Architecture revision 0.

ARCHID, [15:0]

Architecture ID:

`0x4A13` ETMv4 component.

The TRCDEVARCH can be accessed through the external debug interface, offset `0xFBC`.

C10.63 Device ID Register

The TRCDEVID characteristics are:

Purpose

Indicates the capabilities of the ETM trace unit.

Usage constraints

There are no usage constraints.

Configurations

Available in all configurations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

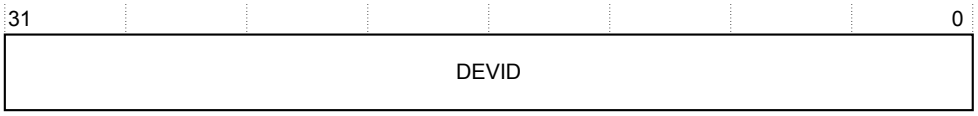

Figure C10-61 TRCDEVID bit assignments

DEVID, [31:0]

RAZ. There are no component-defined capabilities.

The TRCDEVID can be accessed through the external debug interface, offset `0xFC8`.

C10.64 Device Type Register

The TRCDEVTYPE characteristics are:

Purpose

Indicates the type of the component.

Usage constraints

Accessible only from the external debug interface.

Configurations

Available in all configurations.

Attributes

C10.1 ETM register summary on page C10-503.

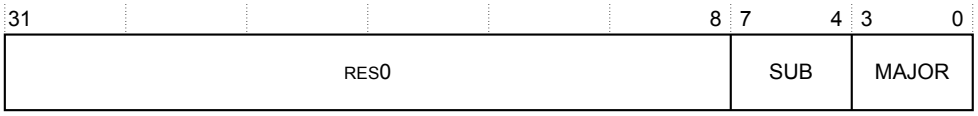

Figure C10-62 TRCDEVTYPE bit assignments

[31:8]

Reserved, RES0.

SUB, [7:4]

The sub-type of the component:

`0b0001` Processor trace.

MAJOR, [3:0]

The main type of the component:

`0b0011` Trace source.

The TRCDEVTYPE can be accessed through the external debug interface, offset `0xFCC`.

C10.65 ETM Peripheral Identification Registers

The ETM Peripheral Identification Registers provide standard information required for all CoreSight components.

The following table lists the Peripheral ID Registers.

Table C10-3 Summary of the Peripheral ID Registers

Register	Value	Offset
Peripheral ID4	0x04	0xFD0
Peripheral ID5	0x00	0xFD4
Peripheral ID6	0x00	0xFD8
Peripheral ID7	0x00	0xFDC
Peripheral ID0	0xDB	0xFE0
Peripheral ID1	0xB9	0xFE4
Peripheral ID2	0x2B	0xFE8
Peripheral ID3	0x00	0xFEC

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

C10.66 ETM Peripheral Identification Register 0

The TRCPIDR0 characteristics are:

Purpose

Provides information to identify a trace component.

Usage constraints

- Only bits[7:0] are valid.
- Accessible only from the external debugger interface, offset `0xFE0`.

Configurations

Available in all implementations.

Attributes

TRCPIDR0 is a 32-bit RO management register.

See *C10.1 ETM register summary* on page C10-503.

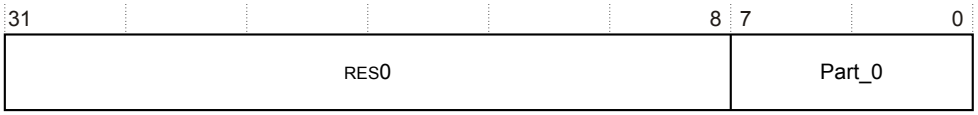

Figure C10-63 TRCPIDR0 bit assignments

[31:8]

Reserved, RES0.

Part_0, [7:0]

`0xDB` Least significant byte of the ETM trace unit part number.

C10.67 ETM Peripheral Identification Register 1

The TRCPIDR1 characteristics are:

Purpose

Provides information to identify a trace component.

Usage constraints

- Only bits[7:0] are valid.
- Accessible only from the external debug interface, offset `0xFE4`.

Configurations

Available in all implementations.

Attributes

TRCPIDR1 is a 32-bit RO management register.

See *C10.1 ETM register summary* on page C10-503.

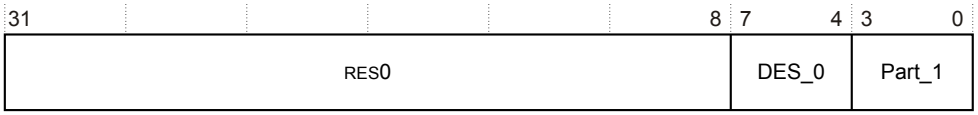

Figure C10-64 TRCPIDR1 bit assignments

[31:8]

Reserved, RES0.

DES_0, [7:4]

`0xB` Arm Limited. This is bits[3:0] of JEP106 ID code.

Part_1, [3:0]

`0x9` Most significant four bits of the ETM trace unit part number.

C10.68 ETM Peripheral Identification Register 2

The TRCPIDR2 characteristics are:

Purpose

Provides information to identify a trace component.

Usage constraints

- Only bits[7:0] are valid.
- Accessible only from the external debug interface, offset `0xFE8`.

Configurations

Available in all implementations.

Attributes

TRCPIDR2 is a 32-bit RO management register.

See *C10.1 ETM register summary* on page C10-503.

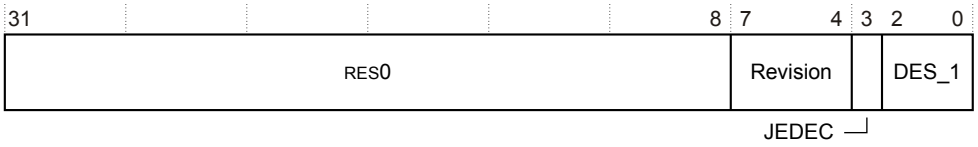

Figure C10-65 TRCPIDR2 bit assignments

[31:8]

Reserved, RES0.

Revision, [7:4]

`0x2` r1p0.

JEDEC, [3]

`0b1` RES1. Indicates a JEP106 identity code is used.

DES_1, [2:0]

`0b011` Arm Limited. This is bits[6:4] of JEP106 ID code.

C10.69 ETM Peripheral Identification Register 3

The TRCPIDR3 characteristics are:

Purpose

Provides information to identify a trace component.

Usage constraints

- Only bits[7:0] are valid.
- Accessible only from the external debug interface, offset `0xFEC`.

Configurations

Available in all implementations.

Attributes

TRCPIDR3 is a 32-bit RO management register.

See *C10.1 ETM register summary* on page C10-503.

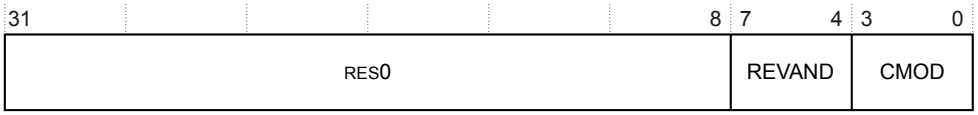

Figure C10-66 TRCPIDR3 bit assignments

[31:8]

Reserved, RES0.

REVAND, [7:4]

`0x0` Part minor revision.

CMOD, [3:0]

`0x0` Not customer modified.

C10.70 ETM Peripheral Identification Register 4

The TRCPIDR4 characteristics are:

Purpose

Provides information to identify a trace component.

Usage constraints

- Only bits[7:0] are valid.
- Accessible only from the external debug interface, offset `0xFD0`.

Configurations

Available in all implementations.

Attributes

TRCPIDR4 is a 32-bit RO management register.

See *C10.1 ETM register summary* on page C10-503.

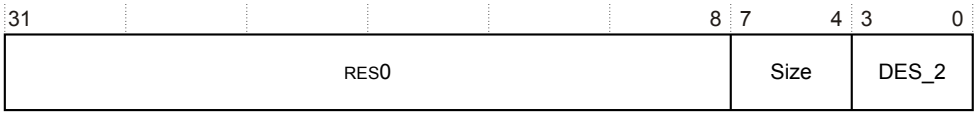

Figure C10-67 TRCPIDR4 bit assignments

[31:8]

Reserved, RES0.

Size, [7:4]

`0x0` Size of the component. Log2 the number of 4KB pages from the start of the component to the end of the component ID registers.

DES_2, [3:0]

`0x4` Arm Limited. This is bits[3:0] of the JEP106 continuation code.

C10.71 ETM Peripheral Identification Register 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

They are reserved for future use and are RES0.

C10.72 ETM Component Identification Registers

There are four read-only ETM Component Identification Registers, Component ID0 to Component ID3.

Table C10-4 Summary of the ETM Component Identification Registers

Register	Value	Offset
Component ID0	0x0D	0xFF0
Component ID1	0x90	0xFF4
Component ID2	0x05	0xFF8
Component ID3	0xB1	0xFFC

The ETM Component Identification Registers identify ETM trace unit as a CoreSight component.

C10.73 ETM Component Identification Register 0

The TRCCIDR0 characteristics are:

Purpose

Provides information to identify a trace component.

Usage constraints

- Only bits[7:0] are valid.
- Accessible only from the external debug interface, offset `0xFF0`.

Configurations

Available in all implementations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

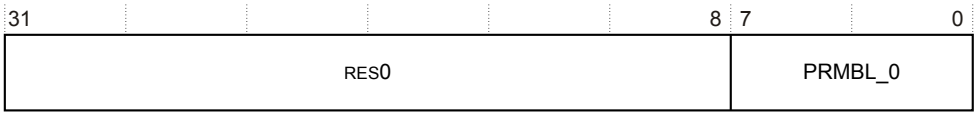

Figure C10-68 TRCCIDR0 bit assignments

[31:8]

Reserved, RES0.

PRMBL_0, [7:0]

`0x0D` Preamble byte 0.

C10.74 ETM Component Identification Register 1

The TRCCIDR1 characteristics are:

Purpose

Provides information to identify a trace component.

Usage constraints

- Only bits[7:0] are valid.
- Accessible only from the external debug interface, offset `0xFF4`.

Configurations

Available in all implementations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

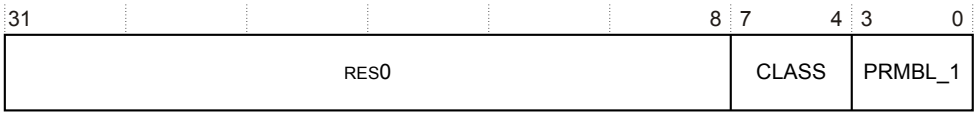

Figure C10-69 TRCCIDR1 bit assignments

[31:8]

Reserved, RES0

CLASS, [7:4]

`0x9` Debug component.

PRMBL_1, [3:0]

`0x0` Preamble byte 1.

C10.75 ETM Component Identification Register 2

The TRCCIDR2 characteristics are:

Purpose

Provides information to identify a CTI component.

Usage constraints

- Only bits[7:0] are valid.
- Accessible only from the external debug interface, offset `0xFF8`.

Configurations

Available in all implementations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

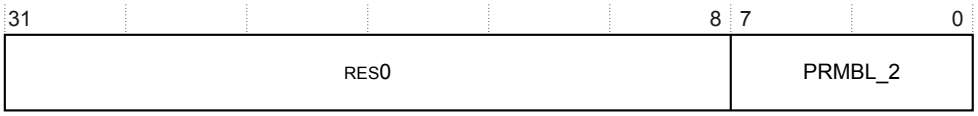

Figure C10-70 TRCCIDR2 bit assignments

[31:8]

Reserved, RES0.

PRMBL_2, [7:0]

`0x05` Preamble byte 2.

C10.76 ETM Component Identification Register 3

The TRCCIDR3 characteristics are:

Purpose

Provides information to identify a trace component.

Usage constraints

- Only bits[7:0] are valid.
- Accessible only from the external debug interface, offset `0xFFC`.

Configurations

Available in all implementations.

Attributes

See *C10.1 ETM register summary* on page C10-503.

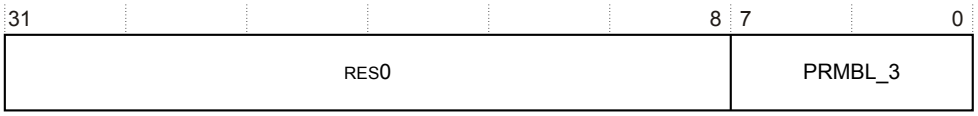

Figure C10-71 TRCCIDR3 bit assignments

[31:8]

Reserved, RES0.

PRMBL_3, [7:0]

`0xB1` Preamble byte 3.

Chapter C11
CTI registers

This chapter describes the CTI registers.

It contains the following sections:

C11.1 Cross trigger register summary

This section describes the cross trigger registers in the Cortex-A32 processor. These registers are accessed through the external debug interface.

The following table gives a summary of the Cortex-A32 cross trigger registers. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table C11-1 Cross trigger register summary

Offset	Name	Type	Description
0x000	CTICONTROL	RW	CTI Control Register
0x000-0x00C	-	-	Reserved
0x010	CTIINTACK	WO	CTI Output Trigger Acknowledge Register
0x014	CTIAPPSET	RW	CTI Application Trigger Set Register
0x018	CTIAPPCLEAR	WO	CTI Application Trigger Clear Register
0x01C	CTIAPPPULSE	WO	CTI Application Pulse Register
0x020	CTIINEN0	RW	CTI Input Trigger to Output Channel Enable Registers
0x024	CTIINEN1	RW	
0x028	CTIINEN2	RW	
0x02C	CTIINEN3	RW	
0x030	CTIINEN4	RW	
0x034	CTIINEN5	RW	
0x038	CTIINEN6	RW	
0x03C	CTIINEN7	RW	
0x040-0x09C	-	-	Reserved
0x0A0	CTIOUTEN0	RW	CTI Input Channel to Output Trigger Enable Registers
0x0A4	CTIOUTEN1	RW	
0x0A8	CTIOUTEN2	RW	
0x0AC	CTIOUTEN3	RW	
0x0B0	CTIOUTEN4	RW	
0x0B4	CTIOUTEN5	RW	
0x0B8	CTIOUTEN6	RW	
0x0BC	CTIOUTEN7	RW	
0x0C0-0x12C	-	-	Reserved
0x130	CTITRIGINSTATUS	RO	CTI Trigger In Status Register
0x134	CTITRIGOUTSTATUS	RO	CTI Trigger Out Status Register
0x138	CTICHINSTATUS	RO	CTI Channel In Status Register
0x13C	CTICHOUTSTATUS	RO	CTI Channel Out Status Register
0x140	CTIGATE	RW	CTI Channel Gate Enable Register

Table C11-1 Cross trigger register summary (continued)

Offset	Name	Type	Description
0x144	ASICCTL	RW	CTI External Multiplexer Control Register
0x148-0xF7C	-	-	Reserved
0xF00	CTIITCTRL	RW	*C11.4 CTI Integration Mode Control Register* on page C11-601
0xF04-0xFA4	-	-	Reserved
0xFA0	CTICLAIMSET	RW	CTI Claim Tag Set Register
0xFA4	CTICLAIMCLR	RW	CTI Claim Tag Clear Register
0xFA8	CTIDEVAFF0	RO	CTI Device Affinity Register 0
0xFAC	CTIDEVAFF1	RO	CTI Device Affinity Register 1
0xFB0	CTILAR	WO	CTI Lock Access Register
0xFB4	CTILSR	RO	CTI Lock Status Register
0xFB8	CTIAUTHSTATUS	RO	CTI Authentication Status Register
0xFBC	CTIDEVARCH	RO	CTI Device Architecture Register
0xFC0	CTIDEVID2	RO	CTI Device ID Register 2
0xFC4	CTIDEVID1	RO	CTI Device ID Register 1
0xFC8	CTIDEVID	RO	*C11.3 CTI Device Identification Register* on page C11-599
0xFCC	CTIDEVTYPE	RO	CTI Device Type Register
0xFD0	CTIPIDR4	RO	*C11.10 CTI Peripheral Identification Register 4* on page C11-607
0xFD4	CTIPIDR5	RO	*C11.11 CTI Peripheral Identification Register 5-7* on page C11-608
0xFD8	CTIPIDR6	RO	
0xFDC	CTIPIDR7	RO	
0xFE0	CTIPIDR0	RO	*C11.6 CTI Peripheral Identification Register 0* on page C11-603
0xFE4	CTIPIDR1	RO	*C11.7 CTI Peripheral Identification Register 1* on page C11-604
0xFE8	CTIPIDR2	RO	*C11.8 CTI Peripheral Identification Register 2* on page C11-605
0xFEC	CTIPIDR3	RO	*C11.9 CTI Peripheral Identification Register 3* on page C11-606
0xFF0	CTICIDR0	RO	*C11.13 CTI Component Identification Register 0* on page C11-610
0xFF4	CTICIDR1	RO	*C11.14 CTI Component Identification Register 1* on page C11-611
0xFF8	CTICIDR2	RO	*C11.15 CTI Component Identification Register 2* on page C11-612
0xFFC	CTICIDR3	RO	*C11.16 CTI Component Identification Register 3* on page C11-613

C11.2 External register access permissions to the CTI registers

External access permission to the cross trigger registers is subject to the conditions at the time of the access.

The following table describes the processor response to accesses through the external debug and memory-mapped interfaces.

Table C11-2 External register conditions

Name	Condition	Description
Off	EDPRSR.PU is 0	Processor power domain is completely off, or in a low-power state where the processor power domain registers cannot be accessed.
DLK	EDPRSR.DLK is 1	OS Double Lock is locked.
OSLK	OSLSR_EL1.OSLK is 1	OS Lock is locked.
EDAD	`AllowExternalDebugAccess() ==FALSE`	External debug access is disabled. When an error is returned because of an EDAD condition code, and this is the highest priority error condition, EDPRSR.SDAD is set to 1. Otherwise EDPRSR.SDAD is unchanged.
SLK	Memory-mapped interface only	Software lock is locked. For the external debug interface, ignore this row.
Default	-	None of the conditions apply, normal access.

The following table shows an example of external register condition codes for access to a cross trigger register. To determine the access permission for the register, scan the columns from left to right. Stop at the first column a condition is true, the entry gives the access permission of the register and scanning stops.

Table C11-3 External register condition code example

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	RO/WI	RO

C11.3 CTI Device Identification Register

The CTIDEVID characteristics are:

Purpose

Describes the CTI component to the debugger.

Usage constraints

The accessibility of CTIDEVID by condition code is:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	RO	RO

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTIDEVID is in the Debug power domain.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

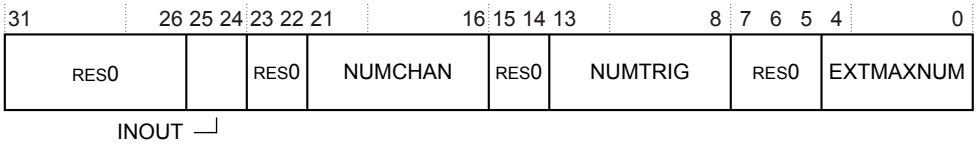

Figure C11-1 CTIDEVID bit assignments

[31:26]

Reserved, RES0.

INOUT, [25:24]

Input and output options. Indicates the presence of an input gate. The possible values are:

`0b00` CTIGATE does not mask propagation of input events from external channels.

`0b01` CTIGATE masks propagation of input events from external channels.

[23:22]

Reserved, RES0.

NUMCHAN, [21:16]

Number of channels implemented. This value is:

`0b00100` Four channels implemented.

[15:14]

Reserved, RES0.

NUMTRIG, [13:8]

Number of triggers implemented. This value is:

`0b01000` Eight triggers implemented.

[7:5]

Reserved, RES0.

EXTMAXNUM, [4:0]

Maximum number of external triggers implemented. This value is:

`0b00000` No external triggers implemented.

CTIDEVID can be accessed through the external debug interface, offset `0xFC8`.

C11.4 CTI Integration Mode Control Register

The CTIITCTRL characteristics are:

Purpose

The CTIITCTRL shows that the Cortex-A32 processor does not implement an integration mode.

Usage constraints

The accessibility of CTIITCTRL by condition code is:

Off	DLK	OSLK	EDAD	SLK	Default
-	-	-	-	RO/WI	RW

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTIITCTRL is in the Debug power domain.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

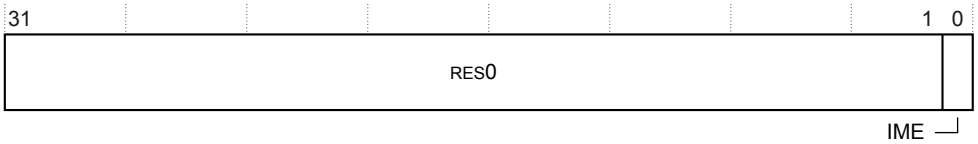

Figure C11-2 CTIITCTRL bit assignments

[31:1]

Reserved, RES0.

IME, [0]

Integration mode enable. The possible value is:

`0` Normal operation.

CTIITCTRL can be accessed through the external debug interface, offset `0xF00`.

C11.5 CTI Peripheral Identification Registers

The CTI Peripheral Identification Registers provide standard information required for all components that conform to the Arm CoreSight architecture.

The following table lists the CTI Peripheral Identification Registers.

Table C11-4 Summary of the CTI Peripheral Identification Registers

Register	Value	Offset
Peripheral ID4	0x04	0xFD0
Peripheral ID5	0x00	0xFD4
Peripheral ID6	0x00	0xFD8
Peripheral ID7	0x00	0xFDC
Peripheral ID0	0xDB	0xFE0
Peripheral ID1	0xB9	0xFE4
Peripheral ID2	0x2B	0xFE8
Peripheral ID3	0x00	0xFEC

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

C11.6 CTI Peripheral Identification Register 0

The CTIPIDR0 characteristics are:

Purpose

Provides information to identify a CTI component.

Usage constraints

The accessibility of CTIPIDR0 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTIPIDR0 is in the Debug power domain.

CTIPIDR0 is optional to implement in the external register interface.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

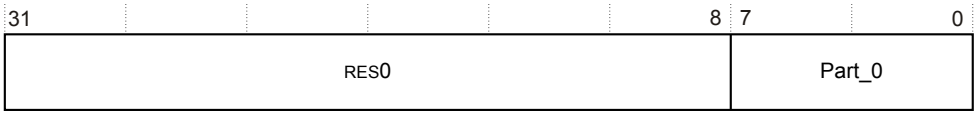

Figure C11-3 CTIPIDR0 bit assignments

[31:8]

Reserved, RES0.

Part_0, [7:0]

`0xDB` Least significant byte of the cross trigger part number.

CTIPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

C11.7 CTI Peripheral Identification Register 1

The CTIPIDR1 characteristics are:

Purpose

Provides information to identify a CTI component.

Usage constraints

The accessibility of CTIPIDR1 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTIPIDR1 is in the Debug power domain.

CTIPIDR1 is optional to implement in the external register interface.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

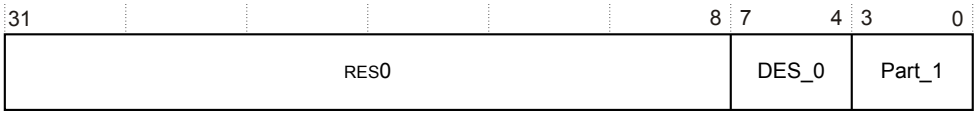

Figure C11-4 CTIPIDR1 bit assignments

[31:8]

Reserved, RES0.

DES_0, [7:4]

`0xB` Arm Limited. This is the least significant nibble of JEP106 ID code.

Part_1, [3:0]

`0x9` Most significant nibble of the CTI part number.

CTIPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

C11.8 CTI Peripheral Identification Register 2

The CTIPIDR2 characteristics are:

Purpose

Provides information to identify a CTI component.

Usage constraints

The accessibility of CTIPIDR2 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTIPIDR2 is in the Debug power domain.

CTIPIDR2 is optional to implement in the external register interface.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

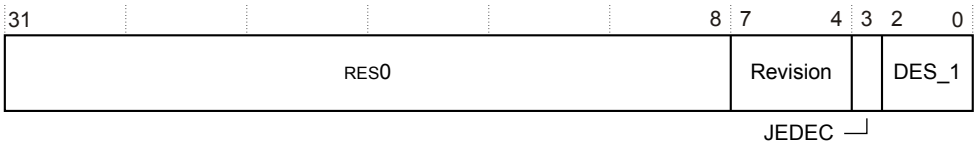

Figure C11-5 CTIPIDR2 bit assignments

[31:8]

Reserved, RES0.

Revision, [7:4]

`0x2` r1p0.

JEDEC, [3]

`0b1` RES1. Indicates a JEP106 identity code is used.

DES_1, [2:0]

`0b011` Arm Limited. This is the most significant nibble of JEP106 ID code.

CTIPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

C11.9 CTI Peripheral Identification Register 3

The CTIPIDR3 characteristics are:

Purpose

Provides information to identify a CTI component.

Usage constraints

The accessibility of CTIPIDR3 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTIPIDR3 is in the Debug power domain.

CTIPIDR3 is optional to implement in the external register interface.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

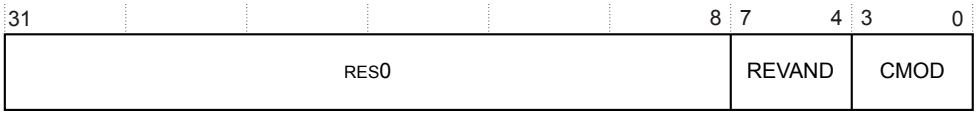

Figure C11-6 CTIPIDR3 bit assignments

[31:8]

Reserved, RES0.

REVAND, [7:4]

`0x0` Part minor revision.

CMOD, [3:0]

`0x0` Customer modified.

CTIPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

C11.10 CTI Peripheral Identification Register 4

The CTIPIDR4 characteristics are:

Purpose

Provides information to identify a CTI component.

Usage constraints

The accessibility of CTIPIDR4 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTIPIDR4 is in the Debug power domain.

CTIPIDR4 is optional to implement in the external register interface.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

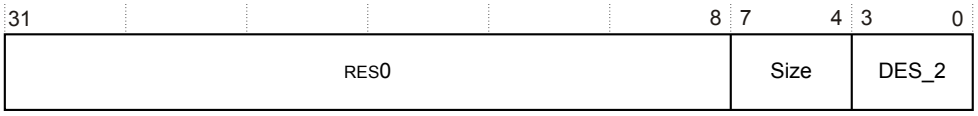

Figure C11-7 CTIPIDR4 bit assignments

[31:8]

Reserved, RES0.

Size, [7:4]

`0x0` Size of the component. Log2 the number of 4KB pages from the start of the component to the end of the component ID registers.

DES_2, [3:0]

`0x4` Arm Limited. This is the least significant nibble JEP106 continuation code.

CTIPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

C11.11 CTI Peripheral Identification Register 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

They are reserved for future use and are RES0.

C11.12 CTI Component Identification Registers

There are four read-only CTI Component Identification Registers, Component ID0 through Component ID3.

Table C11-5 Summary of the CTI Component Identification Registers

Register	Value	Offset
Component ID0	0x0D	0xFF0
Component ID1	0x90	0xFF4
Component ID2	0x05	0xFF8
Component ID3	0xB1	0xFFC

C11.13 CTI Component Identification Register 0

The CTICIDR0 characteristics are:

Purpose

Provides information to identify a CTI component.

Usage constraints

The accessibility of CTICIDR0 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTICIDR0 is in the Debug power domain.

CTICIDR0 is optional to implement in the external register interface.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

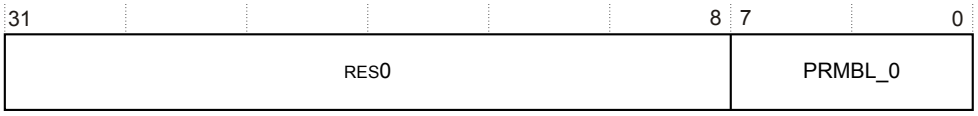

Figure C11-8 CTICIDR0 bit assignments

[31:8]

Reserved, RES0.

PRMBL_0, [7:0]

`0x0D` Preamble byte 0.

CTICIDR0 can be accessed through the external debug interface, offset `0xFF0`.

C11.14 CTI Component Identification Register 1

The CTICIDR1 characteristics are:

Purpose

Provides information to identify a CTI component.

Usage constraints

The accessibility of CTICIDR1 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTICIDR1 is in the Debug power domain.

CTICIDR1 is optional to implement in the external register interface.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

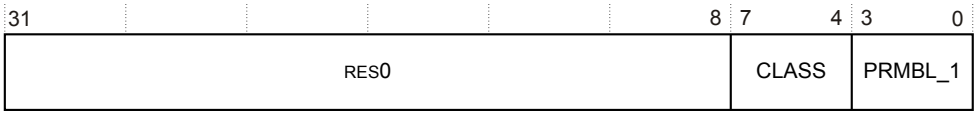

Figure C11-9 CTICIDR1 bit assignments

[31:8]

Reserved, RES0.

CLASS, [7:4]

`0x9` Debug component.

PRMBL_1, [3:0]

`0x0` Preamble byte 1.

CTICIDR1 can be accessed through the external debug interface, offset `0xFF4`.

C11.15 CTI Component Identification Register 2

The CTICIDR2 characteristics are:

Purpose

Provides information to identify a CTI component.

Usage constraints

The accessibility of CTICIDR2 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTICIDR2 is in the Debug power domain.

CTICIDR2 is optional to implement in the external register interface.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

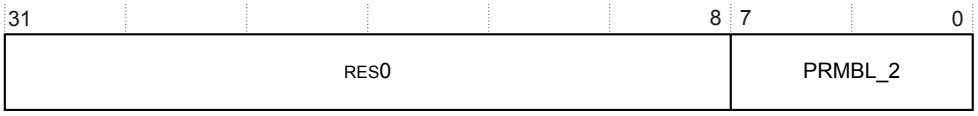

Figure C11-10 CTICIDR2 bit assignments

[31:8]

Reserved, RES0.

PRMBL_2, [7:0]

`0x05` Preamble byte 2.

CTICIDR2 can be accessed through the external debug interface, offset `0xFF8`.

C11.16 CTI Component Identification Register 3

The CTICIDR3 characteristics are:

Purpose

Provides information to identify a CTI component.

Usage constraints

The accessibility of CTICIDR3 by condition code is:

Off	DLK	OSLK	EPMAD	SLK	Default
-	-	-	-	RO	RO

C11.2 External register access permissions to the CTI registers on page C11-598 describes the condition codes.

Configurations

CTICIDR3 is in the Debug power domain.

CTICIDR3 is optional to implement in the external register interface.

Attributes

See the register summary in *C11.1 Cross trigger register summary* on page C11-596.

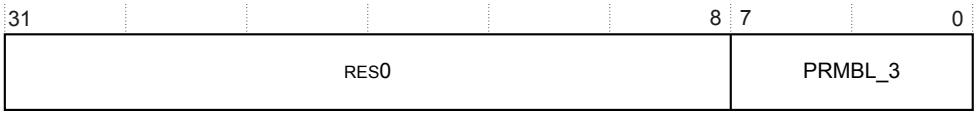

Figure C11-11 CTICIDR3 bit assignments

[31:8]

Reserved, RES0.

PRMBL_3, [7:0]

`0xB1` Preamble byte 3.

CTICIDR3 can be accessed through the external debug interface, offset `0xFFC`.

Part D
Appendices

Appendix A
Signal Descriptions

This appendix describes the signals at the external interfaces of the processor.

It contains the following sections:

A.1 About the signal descriptions

The tables in this appendix provide direction information and high-level descriptions about the signals at the external interfaces of the processor.

Some of the buses include a configurable width field, <Signal>[CN:0], where CN = 0, 1, 2, or 3, to encode up to four cores. For example:

- **nIRQ[0]** represents a core 0 interrupt request.
- **nIRQ[2]** represents a core 2 interrupt request.

Some signals are specified in the form **<signal>x** where x = 0, 1, 2 or 3 to reference core 0, core 1, core 2, core 3. If a core is not present, the corresponding pin is removed. For example:

- **PMUEVENT0[29:0]** represents the core 0 PMU event bus.
- **PMUEVENT3[29:0]** represents the core 3 PMU event bus.

The number of signals changes depending on the configuration. For example, the CHI interface signals are not present when the processor is configured to have an ACE memory interface.

A.2 Processor configuration signals

The processor samples configuration signals only during cluster reset.

Table A-1 Processor configuration signals

Signal	Direction	Description
CFGEND[CN:0]	Input	Endianness configuration at reset. It sets the initial value of SCTLR_EL3.EE and SCTR_S.EE: `0` LOW. `1` HIGH.
CFGTE[CN:0]	Input	Enabling T32 exceptions. It sets the initial value of SCTLR.TE: `0` LOW. `1` HIGH.
CLUSTERIDAFF1[7:0]	Input	Value read in MPIDR.Aff1. This signal is sampled only during processor reset.
CLUSTERIDAFF2[7:0]	Input	Value read in MPIDR.Aff2. This signal is sampled only during processor reset.
CP15SDISABLE2[CN:0]	Input	Disables write access to some Secure CP15 registers.
CRYPTODISABLE[CN:0]	Input	Disables the Cryptographic Extensions. ——— **Note** ——— If a subset of cores in the processor is configured to include the Cryptographic Extensions, then **CRYPTODISABLE[CN:0]** is replaced with separate **CRYPTOxDISABLE** signals, where x indicates the core that each signal relates to. ———
VINITHI[CN:0]	Input	Location of the exception vectors at reset. It sets the initial value of SCTLR.V: `0` Exception vectors start at address `0x00000000`. `1` Exception vectors start at address `0xFFFF0000`.

Related information

B1.97 Multiprocessor Affinity Register on page B1-309
B1.105 System Control Register on page B1-324

A.3 Clock signals

The processor uses a single standard clock signal.

Table A-2 Clock signal

Signal	Direction	Description
CLKIN	Input	Global clock

Related information

A3.1 Clocks on page A3-46

A.4 Reset signals

The processor uses a set of reset signals.

Table A-3 Reset and reset control signals

Signal	Direction	Description
nCPUPORESET[CN:0]	Input	Processor powerup reset: 0 Apply reset to all processor logic. 1 Do not apply reset to all processor logic. Processor logic includes Advanced SIMD and floating-point, debug, ETM trace unit, breakpoint and watchpoint logic.
nCORERESET[CN:0]	Input	Individual core resets excluding debug and ETM trace unit: 0 Apply reset to processor logic. 1 Do not apply reset to processor logic.
nPRESETDBG	Input	See *A.15 APB interface signals* on page Appx-A-643.
nL2RESET	Input	L2 memory system reset: 0 Apply reset to the shared L2 memory system controller. 1 Do not apply reset to the shared L2 memory system controller.
nMBISTRESET	Input	See *A.21 MBIST interface signals* on page Appx-A-649.
L2RSTDISABLE	Input	Disable the automatic invalidation of the L2 cache at reset: 0 Hardware resets the L2 cache. 1 Hardware does not reset the L2 cache. This signal is sampled only during processor reset.
WARMRSTREQ[CN:0]	Output	Request for a processor warm reset: 0 Do not apply warm reset. 1 Apply warm reset.
DBGRSTREQ[CN:0]	Output	Warm reset request.
DBGL1RSTDISABLE	Input	Disable the automatic invalidation of the L1 data cache at processor reset: 0 Enable automatic invalidation of L1 data cache on reset. 1 Disable automatic invalidation of L1 data cache on reset. This signal is sampled only during processor reset.

Related information

A3.3 Resets on page A3-48

A.5 GIC signals

The processor uses a range of signals for global disable, base address definition, interrupt types, and Distributor messaging.

This interface exists only if the processor is configured to use the GIC CPU interface. However, the first seven signals in the following table are present even when the processor is configured without a GIC CPU interface.

Table A-4 GIC signals

Signal	Direction	Description
nFIQ[CN:0]	Input	FIQ request. Active-LOW, level sensitive, asynchronous FIQ interrupt request: `0` Activate FIQ interrupt. `1` Do not activate FIQ interrupt. The processor treats the **nFIQ** input as level-sensitive. The **nFIQ** input must be asserted until the processor acknowledges the interrupt.
nIRQ[CN:0]	Input	IRQ request input lines. Active-LOW, level sensitive, asynchronous interrupt request: `0` Activate interrupt. `1` Do not activate interrupt. The processor treats the **nIRQ** input as level-sensitive. The **nIRQ** input must be asserted until the processor acknowledges the interrupt.
nSEI[CN:0]	Input	System Error Interrupt request. Active-LOW, edge sensitive: `0` Activate SEI request. `1` Do not activate SEI request. The processor treats **nSEI** as edge-sensitive. The **nSEI** signal must be sent as a pulse to the processor. Asserting the **nSEI** input causes an Asynchronous Data Abort. The DFSR.FS field is set to indicate an Asynchronous External Abort.
nVFIQ[CN:0]	Input	Virtual FIQ request. Active-LOW, level sensitive, asynchronous FIQ interrupt request: `0` Activate FIQ interrupt. `1` Do not activate FIQ interrupt. The processor treats the **nVFIQ** input as level-sensitive. The **nVFIQ** input must be asserted until the processor acknowledges the interrupt. If the GIC is enabled by tying **GICCDISABLE** LOW, **nVFIQ** must be tied off to HIGH. If the GIC is disabled by tying **GICCDISABLE** HIGH, **nVFIQ** can be driven by an external GIC in the SoC.
nVIRQ[CN:0]	Input	Virtual IRQ request. Active-LOW, level sensitive, asynchronous interrupt request: `0` Activate interrupt. `1` Do not activate interrupt. The processor treats **nVIRQ** as level-sensitive. **nVIRQ** must be asserted until the processor acknowledges the interrupt. If the GIC is enabled by tying **GICCDISABLE** LOW, **nVIRQ** must be tied off to HIGH. If the GIC is disabled by tying **GICCDISABLE** HIGH, **nVIRQ** can be driven by an external GIC in the SoC.

Table A-4 GIC signals (continued)

Signal	Direction	Description
nVSEI[CN:0]	Input	Virtual System Error Interrupt request. Active-LOW, edge sensitive: `0` Activate virtual SEI request. `1` Do not activate virtual SEI request. The processor treats **nVSEI** as edge-sensitive. The **nVSEI** signal must be sent as a pulse to the processor. Asserting **nVSEI** causes an asynchronous Data Abort. The DFSR.FS field is set to indicate an Asynchronous External Abort.
nREI[CN:0]	Input	RAM Error Interrupt request. Active-LOW, edge sensitive: `0` Activate REI request. Reports an asynchronous RAM error in the system. `1` Do not activate REI request. The processor treats **nREI** as edge-sensitive. The **nREI** signal must be sent as a pulse to the processor. Asserting the **nREI** input causes an asynchronous Data Abort. The DFSR.FS field is set to indicate an Asynchronous parity error on memory access.
nVCPUMNTIRQ[CN:0]	Output	Virtual CPU interface maintenance interrupt PPI output.
PERIPHBASE[39:18]	Input	Specifies the base address for the GIC registers. This value is sampled into CBAR at reset.
GICCDISABLE	Input	Globally disables the GIC CPU interface logic and routes the external signals directly to the processor: `0` Enable the GIC CPU interface logic. `1` Disable the GIC CPU interface logic and route the legacy **nIRQ**, **nFIQ**, **nVIRQ**, and **nVFIQ** signals directly to the processor. Drive this signal HIGH when you are using a legacy interrupt controller such as the GIC-400 which does not support GICv3 or GICv4.

Table A-5 AXI4 Stream Protocol signals for messages from the Distributor to the GIC CPU Interface

Signal	Direction	Description
ICDTVALID	Input	Indicates that the master is driving a valid transfer.
ICDTREADY	Output	Indicates that the slave can accept a transfer in the current cycle.
ICDTDATA[15:0]	Input	Primary payload for the data that is passing across the interface.
ICDTLAST	Input	Indicates the boundary of a packet.
ICDTDEST[1:0]	Input	Routing information for the data stream.

Table A-6 AXI4 Stream Protocol signals for messages from the GIC CPU Interface to the Distributor

Signal	Direction	Description
ICCTVALID	Output	Indicates that the master is driving a valid transfer.
ICCTREADY	Input	Indicates that the slave can accept a transfer in the current cycle.
ICCTDATA[15:0]	Output	Primary payload for the data that is passing across the interface.

Table A-6 AXI4 Stream Protocol signals for messages from the GIC CPU Interface to the Distributor (continued)

Signal	Direction	Description
ICCTLAST	Output	Indicates the boundary of a packet.
ICCTID[1:0]	Output	Data stream identifier.

Related information

B1.49 Data Fault Status Register on page B1-217
B1.38 Configuration Base Address Register on page B1-194
Chapter A12 GIC CPU Interface on page A12-135

A.6 Generic Timer signals

The processor uses a standard set of signals for the Generic Timer.

Table A-7 Generic Timer signals

Signal	Direction	Description
nCNTHPIRQ[CN:0]	Output	Hypervisor physical timer event.
nCNTPNSIRQ[CN:0]	Output	Non-secure physical timer event.
nCNTPSIRQ[CN:0]	Output	Secure physical timer event.
nCNTVIRQ[CN:0]	Output	Virtual physical timer event.
CNTCLKEN	Input	Counter clock enable. This clock enable must be asserted one cycle before the **CNTVALUEB** bus.
CNTVALUEB[63:0]	Input	Global system counter value in binary format.

Related information

A2.4 About the Generic Timer on page A2-43

A.7 Power management signals

The processor has retention and non-retention signals for power management.

Table A-8 Non-Retention power management signals

Signal	Direction	Description
CLREXMONREQ	Input	Clearing of the external global exclusive monitor request. When this signal is asserted, it acts as a WFE wake-up event to all the cores in the processor device.
CLREXMONACK	Output	Clearing of the external global exclusive monitor acknowledge.
EVENTI	Input	Event input for processor wake-up from WFE state.
EVENTO	Output	Event output. Active when a `SEV` instruction is executed.
STANDBYWFI[CN:0]	Output	Indicates whether a core is in WFI low-power state: `0` Core not in WFI low-power state. `1` Core in WFI low-power state. This is the reset condition.
STANDBYWFE[CN:0]	Output	Indicates whether a core is in WFE low-power state: `0` Core not in WFE low-power state. `1` Core in WFE low-power state.
STANDBYWFIL2	Output	Indicates whether the L2 memory system is in WFI low-power state. This signal is active when the following conditions are met: • All cores are in WFI low-power state, held in reset, or **nL2RESET** is asserted LOW. • In an ACE configuration, **ACINACTM** is asserted HIGH. • In a CHI configuration, **SINACT** is asserted HIGH. • If ACP has been configured, **AINACTS** is asserted HIGH. • L2 memory system is idle.
L2FLUSHREQ	Input	L2 hardware flush request.
L2FLUSHDONE	Output	L2 hardware flush complete.
SMPEN[CN:0]	Output	Indicates whether a core is taking part in coherency.
DBGNOPWRDWN[CN:0]	Output	Request not to power down the core: `0` Do not request that the core stays powered-up. `1` Request that the core stays powered-up.
DBGPWRUPREQ[CN:0]	Output	Core power-up request: `0` Do not request that the core is powered up. `1` Request that the core is powered up.
DBGPWRDUP[CN:0]	Input	Core powered up `0` Core is powered down. `1` Core is powered up.

Table A-9 Retention power management signals

Signal	Direction	Description
CPUQACTIVE[CN:0]	Output	Indicates whether the referenced core is active
CPUQREQn[CN:0]	Input	Indicates that the power controller is ready to enter or exit retention for the referenced core
CPUQDENY[CN:0]	Output	Indicates that the referenced core denies the power controller retention request
CPUQACCEPTn[CN:0]	Output	Indicates that the referenced core accepts the power controller retention request
NEONQACTIVE[CN:0]	Output	Indicates whether the referenced Advanced SIMD and Floating-point block is active **Note** If a subset of cores in the processor is configured with Advanced SIMD and floating-point functionality, then **NEONQACTIVE[CN:0]** is replaced with separate **NEONxQACTIVE** signals, where x indicates the core that each signal relates to.
NEONQREQn[CN:0]	Input	Indicates that the power controller is ready to enter or exit retention for the referenced Advanced SIMD and Floating-point block **Note** If a subset of cores in the processor is configured with Advanced SIMD and floating-point functionality, then **NEONQREQn[CN:0]** is replaced with separate **NEONxQREQn** signals, where x indicates the core that each signal relates to.
NEONQDENY[CN:0]	Output	Indicates that the referenced Advanced SIMD and Floating-point block denies the power controller retention request **Note** If a subset of cores in the processor is configured with Advanced SIMD and floating-point functionality, then **NEONQDENY[CN:0]** is replaced with separate **NEONxQDENY** signals, where x indicates the core that each signal relates to.
NEONQACCEPTn[CN:0]	Output	Indicates that the referenced Advanced SIMD and Floating-point block accepts the power controller retention request **Note** If a subset of cores in the processor is configured with Advanced SIMD and floating-point functionality, then **NEONQACCEPTn[CN:0]** is replaced with separate **NEONxQACCEPTn** signals, where x indicates the core that each signal relates to.
L2QACTIVE	Output	Indicates whether the L2 data RAMs are active
L2QREQn	Input	Indicates that the power controller is ready to enter or exit retention for the L2 data RAMs
L2QDENY	Output	Indicates that the L2 data RAMs deny the power controller retention request
L2QACCEPTn	Output	Indicates that the L2 data RAMs accept the power controller retention request

Related information

Chapter A4 Power Management on page A4-53

A.8 L2 error signals

The processor has signals for errors in the Level 2 cache.

Table A-10 L2 error signals

Signal	Direction	Description
nEXTERRIRQ	Output	Error indicator for memory transactions with a write response error condition.
nINTERRIRQ	Output	Error indicator for L2 RAM double-bit ECC error.

Related information

A7.5 Handling of external aborts on page A7-97

A.9 ACP interface signals

The AXI protocol supports clock, configuration, and data handling signals if the processor implements the ACP slave interface to an external master to make coherent request to the shared L2 cache of the cluster.

This interface exists only if the processor is configured to have the ACP interface.

All ACP channels must be balanced with respect to **CLKIN** and timed relative to **ACLKENS**.

Table A-11 ACP clock and Configuration signals

Signal	Direction	Description
ACLKENS	Input	AXI slave bus clock enable.
AINACTS	Input	ACP master is inactive and is not participating in coherency. There must be no outstanding transactions when the master asserts this signal, and while it is asserted the master must not send any new transactions: `0` ACP Master is active. `1` ACP Master is inactive. This signal must be asserted before the processor enters the low-power L2 WFI state.

Table A-12 ACP write address channel signals

Signal	Direction	Description
AWREADYS	Output	Write address ready
AWVALIDS	Input	Write address valid
AWIDS[4:0]	Input	Write address ID
AWADDRS[39:0]	Input	Write address
AWLENS[7:0]	Input	Write burst length
AWCACHES[3:0]	Input	Write cache type
AWUSERS[1:0]	Input	Write attributes: **[0]** Inner Shareable. **[1]** Outer Shareable.
AWPROTS[2:0]	Input	Write protection type

Table A-13 ACP write data channel signals

Signal	Direction	Description
WREADYS	Output	Write data ready
WVALIDS	Input	Write data valid
WDATAS[127:0]	Input	Write data
WSTRBS[15:0]	Input	Write byte-lane strobes
WLASTS	Input	Write data last transfer indication

Table A-14 ACP write response channel signals

Signal	Direction	Description
BREADYS	Input	Write response ready
BVALIDS	Output	Write response valid
BIDS[4:0]	Output	Write response ID
BRESPS[1:0]	Output	Write response

Table A-15 ACP read address channel signals

Signal	Direction	Description
ARREADYS	Output	Read address ready
ARVALIDS	Input	Read address valid
ARIDS[4:0]	Input	Read address ID
ARADDRS[39:0]	Input	Read address
ARLENS[7:0]	Input	Read burst length
ARCACHES[3:0]	Input	Read cache type
ARUSERS[1:0]	Input	Read attributes: **[0]** Inner Shareable. **[1]** Outer Shareable.
ARPROTS[2:0]	Input	Read protection type

Table A-16 ACP read data channel signals

Signal	Direction	Description
RREADYS	Input	Read data ready
RVALIDS	Output	Read data valid
RIDS[4:0]	Output	Read data ID
RDATAS[127:0]	Output	Read data
RRESPS[1:0]	Output	Read response
RLASTS	Output	Read data last transfer indication

A.10 Broadcast signals for the memory interface

The processor has broadcast signals for the memory interface. These signals are only sampled at processor reset.

The signals in the following table only exist if the processor is configured to have an ACE or CHI memory interface.

Table A-17 Broadcast signals

Signal	Direction	Description
BROADCASTCACHEMAINT	Input	Enable broadcasting of cache maintenance operations to downstream caches: `0` Cache maintenance operations are not broadcast to downstream caches. `1` Cache maintenance operations are broadcast to downstream caches.
BROADCASTINNER	Input	Enable broadcasting of Inner Shareable transactions: `0` Inner Shareable transactions are not broadcast externally. `1` Inner Shareable transactions are broadcast externally. If **BROADCASTINNER** is tied HIGH, you must also tie **BROADCASTOUTER** HIGH.
BROADCASTOUTER	Input	Enable broadcasting of outer shareable transactions: `0` Outer Shareable transactions are not broadcast externally. `1` Outer Shareable transactions are broadcast externally.

A.11 AXI interface signals

The AXI protocol supports clock, configuration, and data handling signals when the processor uses this protocol for the master memory interface.

This interface exists only if the processor is configured to have the AXI interface.

All AXI channels must be balanced with respect to **CLKIN** and timed relative to **ACLKENM**.

Table A-18 AXI clock and configuration signals

Signal	Direction	Description
ACLKENM	Input	AXI Master bus clock enable. See *A3.1 Clocks* on page A3-46 for more information.
RDMEMATTR[7:0]	Output	Read request memory attributes.
WRMEMATTR[7:0]	Output	Write request memory attributes.

Table A-19 AXI write address channel signals

Signal	Direction	Description
AWADDRM[39:0]	Output	Write address.
AWBURSTM[1:0]	Output	Write burst type.
AWCACHEM[3:0]	Output	Write cache type.
AWIDM[4:0]	Output	Write address ID.
AWLENM[7:0]	Output	Write burst length.
AWLOCKM	Output	Write lock type.
AWPROTM[2:0]	Output	Write protection type.
AWREADYM	Input	Write address ready.
AWSIZEM[2:0]	Output	Write burst size.
AWVALIDM	Output	Write address valid.

Table A-20 AXI write data channel signals

Signal	Direction	Description
WDATAM[127:0]	Output	Write data
WIDM[4:0]	Output	Write data ID
WLASTM	Output	Write data last transfer indication
WREADYM	Input	Write data ready
WSTRBM[15:0]	Output	Write byte-lane strobes
WVALIDM	Output	Write data valid

Table A-21 AXI write data response channel signals

Signal	Direction	Description
BIDM[4:0]	Input	Write response ID
BREADYM	Output	Write response ready
BRESPM[1:0]	Input	Write response
BVALIDM	Input	Write response valid

Table A-22 AXI read address channel signals

Signal	Direction	Description
ARADDRM[39:0]	Output	Read address.
ARBURSTM[1:0]	Output	Read burst type.
ARCACHEM[3:0]	Output	Read cache type
ARIDM[5:0]	Output	Read address ID
ARLENM[7:0]	Output	Read burst length
ARLOCKM	Output	Read lock type
ARPROTM[2:0]	Output	Read protection type
ARREADYM	Input	Read address ready
ARSIZEM[2:0]	Output	Read burst size
ARVALIDM	Output	Read address valid

Table A-23 AXI read data channel signals

Signal	Direction	Description
RDATAM[127:0]	Input	Read data
RIDM[5:0]	Input	Read data ID
RLASTM	Input	Read data last transfer indication
RREADYM	Output	Read data ready
RRESPM[1:0]	Input	Read data response
RVALIDM	Input	Read data valid

Related information

Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, and ACE and ACE-Lite

A.12 ACE interface signals

The ACE protocol supports clock, configuration, and data handling signals when the processor uses this protocol for the master memory interface.

This interface exists only if the Cortex-A32 processor is configured to have the ACE interface.

All ACE channels must be balanced with respect to **CLKIN** and timed relative to **ACLKENM**.

Table A-24 ACE clock and configuration signals

Signal	Direction	Description
ACLKENM	Input	ACE Master bus clock enable.
ACINACTM	Input	Snoop interface is inactive and not participating in coherency: `0` Snoop interface is active. `1` Snoop interface is inactive.
RDMEMATTR[7:0]	Output	Read request memory attributes.
WRMEMATTR[7:0]	Output	Write request memory attributes.

Table A-25 ACE write address channel signals

Signal	Direction	Description
AWADDRM[43:0]	Output	Write address.
AWBARM[1:0]	Output	Write barrier type.
AWBURSTM[1:0]	Output	Write burst type.
AWCACHEM[3:0]	Output	Write cache type.
AWDOMAINM[1:0]	Output	Write shareability domain type.
AWIDM[4:0]	Output	Write address ID.
AWLENM[7:0]	Output	Write burst length.
AWLOCKM	Output	Write lock type.
AWPROTM[2:0]	Output	Write protection type.
AWREADYM	Input	Write address ready.
AWSIZEM[2:0]	Output	Write burst size.
AWSNOOPM[2:0]	Output	Write snoop request type.
AWUNIQUEM	Output	For WriteBack, WriteClean and WriteEvict transactions. Indicates that the write is: `0` Shared. `1` Unique.
AWVALIDM	Output	Write address valid.

Table A-26 ACE write data channel signals

Signal	Direction	Description
WDATAM[127:0]	Output	Write data
WIDM[4:0]	Output	Write data ID
WLASTM	Output	Write data last transfer indication
WREADYM	Input	Write data ready
WSTRBM[15:0]	Output	Write byte-lane strobes
WVALIDM	Output	Write data valid

Table A-27 ACE write data response channel signals

Signal	Direction	Description
BIDM[4:0]	Input	Write response ID
BREADYM	Output	Write response ready
BRESPM[1:0]	Input	Write response
BVALIDM	Input	Write response valid

Table A-28 ACE read address channel signals

Signal	Direction	Description
ARADDRM[43:0]	Output	Read address. The top 4 bits communicate only the ACE virtual address for DVM messages. The top 4 bits are Read-as-Zero if a DVM message is not being broadcast.
ARBARM[1:0]	Output	Read barrier type.
ARBURSTM[1:0]	Output	Read burst type.
ARCACHEM[3:0]	Output	Read cache type.
ARDOMAINM[1:0]	Output	Read shareability domain type.
ARIDM[5:0]	Output	Read address ID.
ARLENM[7:0]	Output	Read burst length.
ARLOCKM	Output	Read lock type.
ARPROTM[2:0]	Output	Read protection type.
ARREADYM	Input	Read address ready.
ARSIZEM[2:0]	Output	Read burst size.
ARSNOOPM[3:0]	Output	Read snoop request type.
ARVALIDM	Output	Read address valid.

Table A-29 ACE read data channel signals

Signal	Direction	Description
RDATAM[127:0]	Input	Read data
RIDM[5:0]	Input	Read data ID
RLASTM	Input	Read data last transfer indication
RREADYM	Output	Read data ready
RRESPM[3:0]	Input	Read data response
RVALIDM	Input	Read data valid

Table A-30 ACE coherency address channel signals

Signal	Direction	Description
ACADDRM[43:0]	Input	Snoop address. The top 4 bits communicate only the ACE virtual address for DVM messages.
ACPROTM[2:0]	Input	Snoop protection type.
ACREADYM	Output	Master ready to accept snoop address.
ACSNOOPM[3:0]	Input	Snoop request type.
ACVALIDM	Input	Snoop address valid.

Table A-31 ACE coherency response channel signals

Signal	Direction	Description
CRREADYM	Input	Slave ready to accept snoop response
CRVALIDM	Output	Snoop response
CRRESPM[4:0]	Output	Snoop response valid

Table A-32 ACE coherency data channel handshake signals

Signal	Direction	Description
CDDATAM[127:0]	Output	Snoop data
CDLASTM	Output	Snoop data last transform
CDREADYM	Input	Slave ready to accept snoop data
CDVALIDM	Output	Snoop data valid

Table A-33 ACE read and write acknowledge signals

Signal	Direction	Description
RACKM	Output	Read acknowledge
WACKM	Output	Write acknowledge

Related information

A3.1 Clocks on page A3-46

Arm® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, and ACE and ACE-Lite

A.13 CHI interface signals

The CHI protocol supports clock, configuration, data handling, and address map signals when the processor uses this protocol for the master memory interface.

This interface exists only if the processor is configured to have the CHI interface.

Table A-34 CHI clock and configuration signals

Signal	Direction	Description
SCLKEN	Input	CHI interface bus clock enable
SINACT	Input	CHI snoop active
NODEID[6:0]	Input	Cortex-A32 CHI Node Identifier
RXSACTIVE	Input	Receive pending activity indicator
TXSACTIVE	Output	Transmit pending activity indicator
RXLINKACTIVEREQ	Input	Receive link active request
RXLINKACTIVEACK	Output	Receive link active acknowledge
TXLINKACTIVEREQ	Output	Transmit link active request
TXLINKACTIVEACK	Input	Transmit link active acknowledge
REQMEMATTR[7:0]	Output	Request memory attributes

Table A-35 CHI transmit request virtual channel signals

Signal	Direction	Description
TXREQFLITPEND	Output	Transmit request flit pending
TXREQFLITV	Output	Transmit request flit valid
TXREQFLIT[99:0]	Output	Transmit request flit payload
TXREQLCRDV	Input	Transmit request link-layer credit valid

Table A-36 CHI transmit response virtual channel signals

Signal	Direction	Description
TXRSPFLITPEND	Output	Transmit response flit pending
TXRSPFLITV	Output	Transmit response flit valid
TXRSPFLIT[44:0]	Output	Transmit response flit
TXRSPLCRDV	Input	Transmit response link-layer credit valid

Table A-37 CHI transmit data virtual channel signals

Signal	Direction	Description
TXDATFLITPEND	Output	Transmit data flit pending
TXDATFLITV	Output	Transmit data flit valid
TXDATFLIT[193:0]	Output	Transmit data flit
TXDATLCRDV	Input	Transmit data link-layer credit valid

Table A-38 CHI receive snoop virtual channel signals

Signal	Direction	Description
RXSNPFLITPEND	Input	Receive snoop flit pending
RXSNPFLITV	Input	Receive snoop flit valid
RXSNPFLIT[64:0]	Input	Receive snoop flit
RXSNPLCRDV	Output	Receive snoop link-layer credit valid

Table A-39 CHI receive response virtual channel signals

Signal	Direction	Description
RXRSPFLITPEND	Input	Receive response flit pending
RXRSPFLITV	Input	Receive response flit valid
RXRSPFLIT[44:0]	Input	Receive response flit
RXRSPLCRDV	Output	Receive response link-layer credit valid

Table A-40 CHI receive Data virtual channel signals

Signal	Direction	Description
RXDATFLITPEND	Input	Receive data flit pending
RXDATFLITV	Input	Receive data flit valid
RXDATFLIT[193:0]	Input	Receive data flit
RXDATLCRDV	Output	Receive data link-layer credit valid

Table A-41 CHI system address map signals

Signal	Direction	Description
SAMADDRMAP0[1:0]	Input	Region mapping, 0 – 512MB
SAMADDRMAP1[1:0]	Input	Region mapping, 512MB – 1GB
SAMADDRMAP2[1:0]	Input	Region mapping, 1GB – 1.5GB
SAMADDRMAP3[1:0]	Input	Region mapping, 1.5GB – 2GB
SAMADDRMAP4[1:0]	Input	Region mapping, 2GB – 2.5GB
SAMADDRMAP5[1:0]	Input	Region mapping, 2.5GB – 3GB
SAMADDRMAP6[1:0]	Input	Region mapping, 3GB – 3.5GB
SAMADDRMAP7[1:0]	Input	Region mapping, 3.5GB – 4GB
SAMADDRMAP8[1:0]	Input	Region mapping, 4GB – 8GB
SAMADDRMAP9[1:0]	Input	Region mapping, 8GB – 16GB
SAMADDRMAP10[1:0]	Input	Region mapping, 16GB – 32GB
SAMADDRMAP11[1:0]	Input	Region mapping, 32GB – 64GB
SAMADDRMAP12[1:0]	Input	Region mapping, 64GB – 128GB
SAMADDRMAP13[1:0]	Input	Region mapping, 128GB – 256GB

Table A-41 CHI system address map signals (continued)

Signal	Direction	Description
SAMADDRMAP14[1:0]	Input	Region mapping, 256GB – 512GB
SAMADDRMAP15[1:0]	Input	Region mapping, 512GB – 1TB
SAMMNBASE[39:24]	Input	MN base address **SAMMNBASE** must reside in a **SAMADDRMAPx[1:0]** that corresponds to the HN-I.
SAMMNNODEID[6:0]	Input	MN node ID
SAMHNI0NODEID[6:0]	Input	HN-I 0 node ID
SAMHNI1NODEID[6:0]	Input	HN-I 1 node ID
SAMHNF0NODEID[6:0]	Input	HN-F 0 node ID
SAMHNF1NODEID[6:0]	Input	HN-F 1 node ID
SAMHNF2NODEID[6:0]	Input	HN-F 2 node ID
SAMHNF3NODEID[6:0]	Input	HN-F 3 node ID
SAMHNF4NODEID[6:0]	Input	HN-F 4 node ID
SAMHNF5NODEID[6:0]	Input	HN-F 5 node ID
SAMHNF6NODEID[6:0]	Input	HN-F 6 node ID
SAMHNF7NODEID[6:0]	Input	HN-F 7 node ID
SAMHNFMODE[2:0]	Input	HN-F interleaving module

A.14 Debug signals

The processor has the following debug signals.

Table A-42 Debug signals

Signal	Direction	Description
DBGROMADDR[39:12]	Input	Debug ROM base address. Specifies bits[39:12] of the ROM table physical address. If the address cannot be determined, tie this signal LOW. This signal is sampled only during processor reset.
DBGROMADDRV	Input	Debug ROM base address valid. If the debug ROM address cannot be determined, tie this signal LOW. This signal is sampled only during processor reset.
DBGACK[CN:0]	Output	Debug acknowledge: 0 External debug request not acknowledged. 1 External debug request acknowledged.
nCOMMIRQ[CN:0]	Output	Communications channel receive or transmit interrupt request 0 Request interrupt. 1 No interrupt request.
COMMRX[CN:0]	Output	Communications channel receive. Receive portion of Data Transfer Register full flag: 0 Empty. 1 Full.
COMMTX[CN:0]	Output	Communication transmit channel. Transmit portion of Data Transfer Register empty flag: 0 Full. 1 Empty.
EDBGRQ[CN:0]	Input	External debug request: 0 No external debug request. 1 External debug request. The processor treats the **EDBGRQ** input as level-sensitive. The **EDBGRQ** input must be asserted until the processor asserts **DBGACK.**
DBGEN[CN:0]	Input	Invasive debug enable: 0 Not enabled. 1 Enabled.
NIDEN[CN:0]	Input	Non-invasive debug enable: 0 Not enabled. 1 Enabled.

Table A-42 Debug signals (continued)

Signal	Direction	Description
SPIDEN[CN:0]	Input	Secure privileged invasive debug enable: `0` Not enabled. `1` Enabled.
SPNIDEN[CN:0]	Input	Secure privileged non-invasive debug enable: `0` Not enabled. `1` Enabled.
DBGRSTREQ[CN:0]	Output	Warm reset request.
DBGNOPWRDWN[CN:0]	Output	Request not to power down the core: `0` Do not request that the core stays powered up. `1` Request that the core stays powered up.
DBGPWRUPREQ[CN:0]	Output	Core power- up request: `0` Do not request that the core is powered up. `1` Request that the core is powered up.
DBGPWRDUP[CN:0]	Input	Core powered up: `0` Core is powered down. `1` Core is powered up.
DBGL1RSTDISABLE	Input	Disable the automatic invalidation of the L1 data cache at processor reset: `0` Enable automatic invalidation of L1 data cache on reset. `1` Disable automatic invalidation of L1 data cache on reset. This signal is sampled only during processor reset.

Related information

Chapter C1 Debug on page C1-365
Chapter C6 AArch32 debug registers on page C6-409
Chapter C7 Memory-mapped debug registers on page C7-423
Debug memory mapped registers
ROM table

A.15 APB interface signals

The debug APB bus supports clock, reset, addressing, and data handling signals when the processor includes an APB interface to provide access to the debug and performance monitoring registers.

You must balance all APB interface signals with respect to **CLKIN** and time them relative to **PCLKENDBG**.

Table A-43 APB interface signals

Signal	Direction	Description
nPRESETDBG	Input	APB reset, active-LOW: `0` Apply reset to APB interface. `1` Do not apply reset to APB interface.
PADDRDBG[21:2]	Input	APB address bus.
PADDRDBG31	Input	APB address bus bit[31]: `0` Not an external debugger access. `1` External debugger access.
PCLKENDBG	Input	APB clock enable.
PENABLEDBG	Input	Indicates the second and subsequent cycles of an APB transfer.
PRDATADBG[31:0]	Output	APB read data.
PREADYDBG	Output	APB slave ready. An APB slave can deassert **PREADYDBG** to extend a transfer by inserting wait states.
PSELDBG	Input	Debug bus access.
PSLVERRDBG	Output	APB slave transfer error: `0` No transfer error. `1` Transfer error.
PWDATADBG[31:0]	Input	APB write data.
PWRITEDBG	Input	APB read or write signal: `0` Reads from APB. `1` Writes to APB.

A.16 ATB interface signals

The ATB bus supports clock and data handling signals when the processor includes an ATB interface for each core to output trace information for debugging.

This interface exists only if the processor is configured to have one or more ETMs.

You must balance all ATB interface signals with respect to **CLKIN** and time them relative to **ATCLKEN**.

Table A-44 ATB interface signals

Signal	Direction	Description
ATCLKEN	Input	ATB clock enable
ATREADYMx	Input	ATB device ready
AFVALIDMx	Input	FIFO flush request
ATDATAMx[31:0]	Output	Data
ATVALIDMx	Output	Data valid
ATBYTESMx[1:0]	Output	Data size
AFREADYMx	Output	FIFO flush finished
ATIDMx[6:0]	Output	Trace source ID
SYNCREQMx	Input	Synchronization request from the trace sink

A.17 ETM signals

The processor has signals to receive information from an external trace device.

This interface exists only if the processor is configured to have one or more ETMs.

Table A-45 ETM signals

Signal	Direction	Description
TSVALUEB[63:0]	Input	Timestamp in binary encoding

A.18 PMU interface signals

The processor uses the PMU signals to communicate with the external performance monitoring device.

Table A-46 PMU interface signals

Signal	Direction	Description
PMUEVENTx[29:0]	Output	PMU event bus
nPMUIRQ[CN:0]	Output	PMU interrupt request

A.19 CTI interface signals

The processor supports various cross-trigger signals when it implements the CTI.

Table A-47 CTI interface signals

Signal	Direction	Description
CTICHIN[3:0]	Input	Channel In
CTICHOUTACK[3:0]	Input	Channel Out acknowledge
CTICHOUT[3:0]	Output	Channel Out
CTICHINACK[3:0]	Output	Channel In acknowledge
CISBYPASS	Input	Channel interface sync bypass
CIHSBYPASS[3:0]	Input	Channel interface H/S bypass
CTIIRQ[CN:0]	Output	CTI interrupt, active-HIGH
CTIIRQACK[CN:0]	Input	CTI interrupt acknowledge

A.20 DFT interface signals

The processor uses the DFT signals to communicate with the external test device.

Table A-48 DFT interface signals

Signal	Direction	Description
DFTRAMHOLD	Input	Disable the RAM chip select during scan testing
DFTRSTDISABLE	Input	Disable internal synchronized reset during scan shift
DFTCGEN	Input	Clock gate enable, forces on the clock grids during scan shift
DFTMCPHOLD	Input	Disable Multicycle Paths on RAM interfaces

A.21 MBIST interface signals

The processor does not include an external MBIST address and data interface port. The MBIST address and data interface ports are internally tied-off in the design, and you can insert MBIST into the design before synthesis. The process of adding MBIST into the design can be done automatically by an EDA MBIST tool.

Table A-49 MBIST interface signals

Signal	Direction	Description
MBISTREQ	Input	MBIST test request
nMBISTRESET	Input	MBIST reset

Appendix B
AArch32 UNPREDICTABLE Behaviors

The cases in which the Cortex-A32 processor implementation diverges from the preferred behavior described in Armv8 AArch32 UNPREDICTABLE behaviors.

It contains the following sections:

B.1 Use of R15 by Instruction

If the use of R15 as a base register for a load or store is UNPREDICTABLE, the value used by the load or store using R15 as a base register is the *Program Counter* (PC) with its usual offset and, in the case of T32 instructions, with the forced word alignment. In this case, if the instruction specifies Writeback, then the load or store is performed without Writeback.

The Cortex-A32 core does not implement a *Read 0* or *Ignore Write* policy on UNPREDICTABLE use of R15 by instruction. Instead, the Cortex-A32 core takes an UNDEFINED exception trap.

B.2 UNPREDICTABLE instructions within an IT Block

Conditional instructions within an IT Block, described as being unpredictable in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* pseudo-code, are executed unconditionally.

The Cortex-A32 core does not implement an unconditional execution policy for the following instructions. Instead all execute conditionally:

- NEON instructions new to Armv8.
- All instructions in the Armv8 Cryptographic Extensions.
- CRC32.

B.3 Load/Store accesses crossing page boundaries

The Cortex-A32 processor implements a set of behaviors for load or store accesses that cross page boundaries.

Crossing a page boundary with different memory types or shareability attributes

The *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*, states that a memory access from a load or store instruction that crosses a page boundary to a memory location that has a different memory type or shareability attribute results in CONSTRAINED UNPREDICTABLE behavior.

Crossing a 4KB boundary with a Device access

The *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*, states that a memory access from a load or store instruction to Device memory that crosses a 4KB boundary results in CONSTRAINED UNPREDICTABLE behavior.

Implementation (for both page boundary specifications)

For an access that crosses a page boundary, the Cortex-A32 processor implements the following behaviors:

- Store crossing a page boundary:
  - No alignment fault.
  - The access is split into two stores.
  - Each store uses the memory type and shareability attributes associated with its own address.
- Load crossing a page boundary (Device to Device and Normal to Normal):
  - No alignment fault.
  - The access is split into two loads.
  - Each load uses the memory type and shareability attributes associated with its own address.
- Load crossing a page boundary (Device to Normal and Normal to Device):
  - The instruction might generate an alignment fault.
  - If no fault is generated, the access is split into two loads.
  - Each load uses the memory type and shareability attributes associated with its own address.

B.4 Armv8 Debug UNPREDICTABLE behaviors

This section describes the behavior that the Cortex-A32 processor implements when:

- A topic has multiple options.
- The behavior differs from either or both of the Options and Preferences behaviors.

Note

This section does not describe the behavior when a topic only has a single option and the processor implements the preferred behavior.

Table B-1 Armv8 Debug UNPREDICTABLE behaviors

Scenario	Behavior
A32 BKPT instruction with condition code not AL	The processor implements the following preferred option: • Executed unconditionally.
Address match breakpoint match only on second halfword of an instruction	The processor generates a breakpoint on the instruction, unless it is a breakpoint on the second half of the first 32-bit instruction. In this case the breakpoint is taken on the following instruction.
Address matching breakpoint on A32 instruction with DBGBCRn.BAS=1100	The processor implements the following option: • Does match.
Address match breakpoint match on T32 instruction at DBGBCRn+2 with DBGBCRn.BAS=1111	The processor implements the following option: • Does match.
Address mismatch breakpoint match on T32 instruction at DBGBCRn +2 with DBGBCRn.BAS=1111	The processor implements the following option: • Does match.
Other mismatch breakpoint matches any address in current mode and state	The processor implements the following option: • Immediate breakpoint debug event.
Mismatch breakpoint on branch to self	The processor implements the following option: • Instruction is stepped an UNKNOWN number of times, while it continues to branch to itself.
Link to non-existent breakpoint or breakpoint that is not context-aware	The processor implements the following option: • No Breakpoint or Watchpoint debug event is generated, and the LBN field of the *linker* reads UNKNOWN.
DBGWCRn_EL1.MASK!=00000 and DBGWCRn_EL1.BAS!=11111111	The processor behaves as indicated in the sole Preference: • DBGWCRn_EL1.BAS is ignored and treated as if `0x11111111`.
Address-matching Vector catch on 32-bit T32 instruction at (vector-2)	The processor implements the following option: • Does match.
Address-matching Vector catch on 32-bit T32 instruction at (vector+2)	The processor implements the following option: • Does match.
Address-matching Vector catch and Breakpoint on same instruction	The processor implements the following option: • Report Breakpoint.

Table B-1 Armv8 Debug UNPREDICTABLE behaviors (continued)

Scenario	Behavior
Address match breakpoint with DBGBCRn_EL1.BAS=0000	The processor implements the following option: • As if disabled.
DBGWCRn_EL1.BAS specifies a non-contiguous set of bytes within a double-word	The processor implements the following option: • A Watchpoint debug event is generated for each byte.
A32 HLT instruction with condition code not AL	The processor implements the following option: • Executed unconditionally.
Execute instruction at a given EL when the corresponding EDECCR bit is 1 and Halting is allowed	The processor behaves as follows: • Generates debug event and Halt no later than the instruction following the next Context Synchronization operation (CSO) excluding ISB instruction.
Unlinked Context matching and Address mismatch breakpoints taken to Abort mode	The processor implements the following option: • A Prefetch Abort debug exception is generated. Because the breakpoint is configured to generate a breakpoint at PL1, the instruction at the Prefetch Abort vector 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.
Vector catch on Data or Prefetch abort, and taken to Abort mode	The 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 Non-secure EL1 and EL0, including value read from PMCR_EL0.N	The processor implements: • A simple implementation where all of HPMN[4:0] are implemented, and In Non-secure EL1 and EL0: — If H > N then M = N. — If H = 0 then M = 0.
H > N or H = 0: value read back in MDCR_EL2.HPMN	The processor implements: • A simple implementation where all of HPMN[4:0] are implemented and for reads of MDCR_EL2.HPMN, return H.
P ≥ M and P ≠ 31: reads and writes of PM XEVTYPER_EL0 and PMXEVCNTR_EL0	The 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 ≥ M and P ≠ 31: value read in PMSELR_EL0.SEL	The 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.

Table B-1 Armv8 Debug UNPREDICTABLE behaviors (continued)

Scenario	Behavior
P = 31: reads and writes of PMXEVCNTR_EL0	The processor implements: • RES0.
n ≥ M: Direct access to PMEVCNTRn_EL0 and PMEVTYPERn_EL0	The 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 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 processor behaves as indicated in the sole Preference: • Does unaligned accesses, faulting if these are not permitted for the memory type.
Access to memory-mapped registers mapped to Normal memory	The processor behaves as indicated in the sole Preference: • The access is generated, and accesses might be repeated, gathered, split or resized, in accordance with the rules for Normal memory, meaning the effect is UNPREDICTABLE.
Not word-sized accesses	The processor behaves as indicated in the sole Preference: • Reads occur and return UNKNOWN data. • Writes set the accessed register(s) to UNKNOWN.
External debug write to register that is being reset	The processor behaves as indicated in the sole Preference: • Takes reset value.

Table B-1 Armv8 Debug UNPREDICTABLE behaviors (continued)

Scenario	Behavior
Accessing reserved debug registers	The processor deviates from Preferred behavior because the hardware cost to decode some of these addresses in debug power domain is significantly high: **Actual behavior:** 1. For reserved debug and Performance Monitors registers the response is CONSTRAINED UNPREDICTABLE Error or RES0, when any of the following error instead of preferred RES0 for reserved debug registers `0x000-0xCFC` and reserved PMU registers `0x000-0xF00`: **Off** 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, that is, EDPRSR.DLK is 1. **OSLK** OSLSR_EL1.OSLK is1, OS lock is locked. 2. In addition, for reserved debug registers in the address ranges `0x400` to `0x4FC` and `0x800` to `0x8FC`, the response is CONSTRAINED UNPREDICTABLE Error or RES0 when the conditions in *1* do not apply and: **EDAD** `AllowExternalDebugAccess()` is FALSE, external debug access is disabled. 3. For reserved Performance Monitor registers in the address ranges `0x000` to `0x0FC` and `0x400` to `0x47C`, the response is CONSTRAINED UNPREDICTABLE Error, or RES0 when the conditions in *1* and *2* do not apply, and the following errors instead of preferred res0 for the these registers: **EPMAD** `AllowExternalPMUAccess()` is FALSE (external Performance Monitors access is disabled).
Clearing the *clear-after-read* EDPRSR bits when Core power domain is on, and `DoubleLockStatus()` is TRUE	The processor behaves as indicated in the sole Preference: • Bits are not cleared to zero.

B.5 Other UNPREDICTABLE behaviors

This section describes other UNPREDICTABLE behaviors.

Table B-2 Other UNPREDICTABLE behaviors

Scenario	Description
CSSELR indicates a cache that is not implemented.	If CSSELR indicates a cache that is not implemented, then on a read of the CCSIDR the behavior is CONSTRAINED UNPREDICTABLE, and can be one of the following: • The CCSIDR read is treated as `NOP`. • The CCSIDR read is UNDEFINED. • The CCSIDR read returns an UNKNOWN value (preferred).
HDCR.HPMN is set to 0, or to a value larger than PMCR.N.	If HDCR.HPMN is set to 0, or to a value larger than PMCR.N, then the behavior in Non-secure EL0 and EL1 is CONSTRAINED UNPREDICTABLE, and one of the following must happen: • The number of counters accessible is an UNKNOWN non-zero value less than PMCR.N. • There is no access to any counters. For reads of HDCR.HPMN by EL2 or higher, if this field is set to 0 or to a value larger than PMCR.N, the core must return a CONSTRAINED UNPREDICTABLE value that is one of: • PMCR.N. • The value that was written to HDCR.HPMN. • (The value that was written to HDCR.HPMN) modulo 2h, where h is the smallest number of bits required for a value in the range 0 to PMCR.N.

Appendix C
Revisions

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

It contains the following section:

- *C.1 Revisions* on page Appx-C-662.

C.1 Revisions

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

Table C-1 Issue 0000-00

Change	Location	Affects
First release for r0p0.	-	-

Table C-2 Issue 0001-00

Change	Location	Affects
First release for r0p1.	On front page and in document history table. *B1.96 Main ID Register* on page B1-307. *B2.3 CPU Interface Identification Register* on page B2-354. *B2.8 VM CPU Interface Identification Register* on page B2-359. *C7.11 External Debug Peripheral Identification Register 1* on page C7-440. *C8.8 ROM Table Peripheral Identification Register 2* on page C8-462. *C9.10 Performance Monitors Peripheral Identification Register 2* on page C9-491. *C10.13 Trace ID Register* on page C10-520. *C10.68 ETM Peripheral Identification Register 2* on page C10-586. *C11.8 CTI Peripheral Identification Register 2* on page C11-605.	r0p1
References to internal memory-mapped removed.	*B1.96 Main ID Register* on page B1-307. *B1.97 Multiprocessor Affinity Register* on page B1-309. *C7.1 Memory-mapped debug register summary* on page C7-424. *C9.1 AArch32 PMU register summary* on page C9-472. *C9.5 Memory-mapped PMU register summary* on page C9-483. *C11.1 Cross trigger register summary* on page C11-596.	All versions
nSEI, nREI, and nVSEI clarified in GIC signals table.	*A.5 GIC signals* on page Appx-A-622.	All versions

Table C-3 Issue 0100-00

Change	Location	Affects
First release for r1p0	Revision history table *B1.96 Main ID Register* on page B1-307 *B2.3 CPU Interface Identification Register* on page B2-354. *B2.8 VM CPU Interface Identification Register* on page B2-359. *C8.8 ROM Table Peripheral Identification Register 2* on page C8-462. *C9.10 Performance Monitors Peripheral Identification Register 2* on page C9-491. *C10.68 ETM Peripheral Identification Register 2* on page C10-586. *C11.8 CTI Peripheral Identification Register 2* on page C11-605. MIDR and VPIDR reset values in *B1.2 c0 registers* on page B1-146, *B1.20 AArch32 Identification registers* on page B1-170 and *B1.28 AArch32 Virtualization registers* on page B1-180. GICV_IIDR reset value in *B2.6 Virtual CPU interface register summary* on page B2-357. GICC_IIDR reset value in *B2.1 CPU interface register summary* on page B2-352.	r1p0
Updated company name from Arm to Arm	Entire manual	All versions
Updated all sections affected by the addition of a new asymmetric floating-point/ NEON feature	• *A1.3 Implementation options* on page A1-28 • *A2.1 Components* on page A2-36 • *A.2 Processor configuration signals* on page Appx-A-619 • *A.7 Power management signals* on page Appx-A-626	r1p0
Updated all sections affected by the additional of the **CP15SDISABLE2** signal	• *A2.3 About system control* on page A2-42 • *B1.47 Domain Access Control Register* on page B1-215 • *B1.95 Memory Attribute Indirection Registers 0 and 1* on page B1-304 • *B1.98 Non-Secure Access Control Register* on page B1-311 • *B1.99 Normal Memory Remap Register* on page B1-313 • *B1.101 Primary Region Remap Register* on page B1-316 • *B1.105 System Control Register* on page B1-324 • *B1.106 Secure Debug Control Register* on page B1-328 • *B1.107 Secure Debug Enable Register* on page B1-330 • *B1.110 Translation Table Base Control Register* on page B1-333 • *B1.116 Translation Table Base Register 1* on page B1-342 • *B1.119 Vector Base Address Register* on page B1-346	r1p0
Fixed incorrect MRC/MCR encodings	*C9.2 Performance Monitors Control Register* on page C9-474	All versions
Clarified the L2 cache behavior when disabled	*A5.4 Disabling a cache* on page A5-77	All versions
Added warm reset information	*A3.3 Resets* on page A3-48	All versions