ARM® Cortex®-A15 MPCore™ Processor

Revision: r4p0

Technical Reference Manual

ARM DDI 0438I (ID073015)

ARM Cortex-A15 MPCore Processor
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
26 April 2011	A	Non-Confidential	First release for r0p0
29 July 2011	B	Non-Confidential	First release for r1p0
28 September 2011	C	Non-Confidential	First release for r2p0
16 December 2011	D	Non-Confidential	First release for r2p1
20 March 2012	E	Non-Confidential	First release for r3p0
04 May 2012	F	Non-Confidential	First release for r3p1
13 July 2012	G	Non-Confidential	First release for r3p2
06 December 2012	H	Non-Confidential	First release for r3p3
24 June 2013	I	Non-Confidential	First release for r4p0

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM® in the EU and other countries, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Where the term ARM is used it means "ARM or any of its subsidiaries as appropriate".

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.

Product Status

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

Web Address

`http://www.arm.com`

Contents

ARM Cortex-A15 MPCore Processor Technical Reference Manual

Chapter 4 System Control

Chapter 5 Memory Management Unit

Chapter 6 Level 1 Memory System

Chapter 7 Level 2 Memory System

Chapter 8 Generic Interrupt Controller

Chapter 9 Generic Timer

Chapter 10 Debug

Chapter 11 Performance Monitor Unit

Preface

This preface introduces the *ARM® Cortex®-A15 MPCore™ Processor Technical Reference Manual*. It contains the following sections:

- *About this book* on page vii.
- *Feedback* on page xi.

Note

- The out-of-order design of the Cortex-A15 MPCore processor pipeline makes it impossible to provide accurate timing information for complex instructions. The timing of an instruction can be affected by factors such as:
  - Other concurrent instructions.
  - Memory system activity.
  - Events outside the instruction flow.
- Timing information has been provided in the past for some ARM processors to assist in detailed hand tuning of performance critical code sequences or in the development of an instruction scheduler within a compiler. This timing information is not required for producing optimized instruction sequences on the Cortex-A15 MPCore processor. The out-of-order pipeline of the Cortex-A15 MPCore processor can schedule and execute the instructions in an optimal fashion without any instruction reordering required.

About this book

This book is for the Cortex-A15 MPCore processor. This is a multiprocessor device that has between one to four Cortex-A15 processors.

Product revision status

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

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the Cortex-A15 MPCore processor.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the processor and descriptions of the major features.

Chapter 2 *Functional Description*

Read this for a description of the functionality of the processor.

Chapter 3 *Programmers Model*

Read this for a description of the programmers model.

Chapter 4 *System Control*

Read this for a description of the system control registers and programming information.

Chapter 5 *Memory Management Unit*

Read this for a description of the *Memory Management Unit* (MMU) and the address translation process.

Chapter 6 *Level 1 Memory System*

Read this for a description of the *Level 1* (L1) memory system that consists of separate instruction and data caches.

Chapter 7 *Level 2 Memory System*

Read this for a description of the *Level 2* (L2) memory system.

Chapter 8 *Generic Interrupt Controller*

Read this for a description of the Generic Interrupt Controller.

Chapter 9 *Generic Timer*

Read this for a description of the Generic Timer.

Chapter 10 *Debug*

Read this for a description of the processor support for debug.

Chapter 11 *Performance Monitor Unit*

Read this for a description of the Cortex-A15 *Performance Monitor Unit* (PMU).

Chapter 12 *Program Trace Macrocell*

Read this for a description of the processor support for instruction trace.

Chapter 13 *Cross Trigger*

Read this for a description of the cross trigger interfaces.

Chapter 14 *NEON and VFP Unit*

Read this for a description of the NEON and *Vector Floating-Point* (VFP) unit.

Appendix A *Signal Descriptions*

Read this for a description of the signals in the Cortex-A15 MPCore processor.

Appendix B *Revisions*

Read this for a description of 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 *ARM® Glossary*, http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html.

Conventions

This book uses the conventions that are described in:

- *Typographical conventions*.
- *Timing diagrams* on page ix.
- *Signals* on page ix.

Typographical conventions

The following table describes the typographical conventions:

Style	Purpose
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.

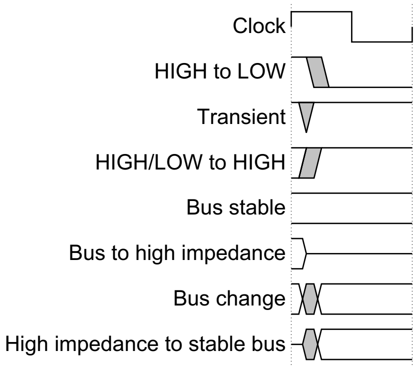
(continued)

Style	Purpose
`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

Key to timing diagram conventions explains the components used in these diagrams. When variations occur they 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.



Key to timing diagram conventions

Timing diagrams sometimes show single-bit signals as HIGH and LOW at the same time and they look similar to the bus change shown in *Key to timing diagram conventions*. If a timing diagram shows a single-bit signal in this way then its value does not affect the accompanying description.

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.

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

Additional reading

This section lists relevant documents published by third parties.

See Infocenter, `http://infocenter.arm.com`, for access to ARM documentation.

ARM publications

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

- *ARM® Cortex®-A15 MPCore™ Processor Configuration and Sign-off Guide* (ARM DII 0219).
- *ARM® AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™, ACE and ACE-Lite™* (ARM IHI 0022).
- *ARM® AMBA® APB™ Protocol Specification* (ARM IHI 0024).
- *ARM® AMBA® 3 ATB Protocol Specification* (ARM IHI 0032).
- *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* (ARM DDI 0406).
- *ARM® Generic Interrupt Controller Architecture Specification* (ARM IHI 0048).
- *ARM® CoreSight™ Architecture Specification* (ARM IHI 0029).
- *ARM® CoreSight™ Program Flow Trace Architecture Specification* (ARM IHI 0035).

Other publications

This section lists relevant documents published by third parties:

- *ANSI/IEEE Std 754-1985, IEEE Standard for Binary Floating-Point Arithmetic*.
- *ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic*.

Chapter 1
Introduction

This chapter introduces the Cortex-A15 MPCore processor and its features. It contains the following sections:

1.1 About the Cortex-A15 MPCore processor

The Cortex-A15 MPCore processor is a high-performance, low-power multiprocessor that implements the ARMv7-A architecture. The Cortex-A15 MPCore processor has one to four Cortex-A15 processors in a single multiprocessor device, or *MPCore device*, with L1 and L2 cache subsystems.

Figure 1-1 shows an example block diagram of a Cortex-A15 MPCore processor configuration with four processors.

See *Components of the processor* on page 2-2 for a description of the Cortex-A15 MPCore processor functional components.

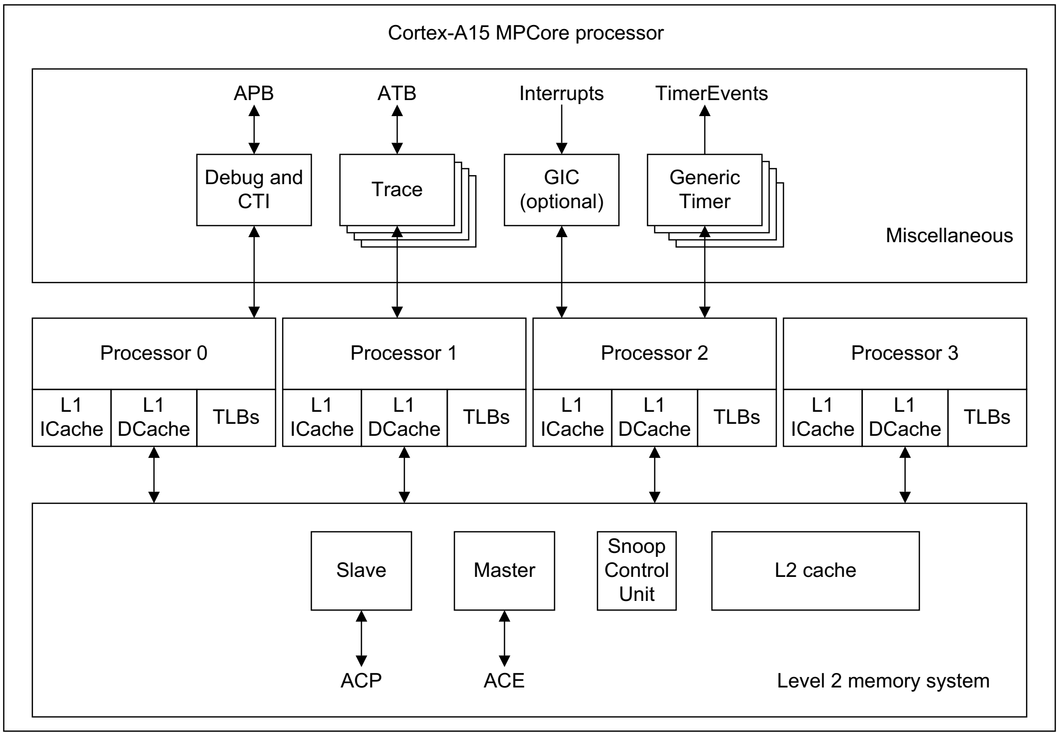


Figure 1-1 Example multiprocessor configuration

1.2 Compliance

The Cortex-A15 MPCore processor complies with, or implements, the specifications described in:

- *ARM architecture*.
- *Advanced Microcontroller Bus Architecture*.
- *Debug architecture* on page 1-4.
- *Generic Interrupt Controller architecture* on page 1-4.
- *Generic Timer architecture* on page 1-4.
- *Program Flow Trace architecture* on page 1-4.

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

1.2.1 ARM architecture

The Cortex-A15 MPCore processor implements the ARMv7-A architecture with the following architecture extensions:

- Advanced *Single Instruction Multiple Data version 2* (SIMDv2) architecture extension for integer and floating-point vector operations.

  Note

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

- *Vector Floating-Point version 4* (VFPv4) architecture extension for floating-point computation that is fully compliant with the IEEE 754 standard.
- Security Extensions for implementation of enhanced security.
- Virtualization Extensions for the development of virtualized systems that enables the switching of guest operating systems.
- *Large Physical Address Extension* (LPAE) for address translation of up to 40 bits physical addresses.
- Multiprocessing Extensions for multiprocessing functionality.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

1.2.2 Advanced Microcontroller Bus Architecture

The Cortex-A15 MPCore processor complies with the:

- AMBA 4 *Advanced eXtensible Interface* (AXI) and *AXI Coherency Extensions* (ACE) protocol. See the *ARM® AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™, ACE and ACE-Lite™*.
- AMBA 3 *Advanced Peripheral Bus* (APB) protocol. See the *ARM® AMBA® APB Protocol Specification*.
- AMBA 3 *Advanced Trace Bus* (ATB) protocol. See the *ARM® AMBA® 3 ATB Protocol Specification*.

1.2.3 Debug architecture

The Cortex-A15 MPCore processor implements version 7.1 of the ARM Debug architecture that complies with the CoreSight architecture. For more information, see the:

- *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition.*
- *ARM® CoreSight™ Architecture Specification*.

1.2.4 Generic Interrupt Controller architecture

The Cortex-A15 MPCore processor implements version 2.0 of the ARM *Generic Interrupt Controller* (GICv2) architecture that includes support for the Virtualization Extensions. See the *ARM® Generic Interrupt Controller Architecture Specification*.

1.2.5 Generic Timer architecture

The Cortex-A15 MPCore processor implements the ARM Generic Timer architecture that includes support for the Virtualization Extensions. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

1.2.6 Program Flow Trace architecture

The Cortex-A15 MPCore processor implements the *Program Trace Macrocell* (PTM) based on version 1.1 of the *Program Flow Trace* (PFTv1.1) architecture. See the *ARM® CoreSight™ Program Flow Trace Architecture Specification*.

1.3 Features

The Cortex-A15 MPCore processor includes the following features:

- Full implementation of the ARMv7-A architecture instruction set with the architecture extensions listed in *Compliance* on page 1-3.
- Superscalar, variable-length, out-of-order pipeline.
- Dynamic branch prediction with *Branch Target Buffer* (BTB) and *Global History Buffer* (GHB), a return stack, and an indirect predictor.
- Three separate 32-entry fully-associative *Level 1* (L1) *Translation Look-aside Buffers* (TLBs), one for instruction, one for data loads, and one for data stores.
- 4-way set-associative 512-entry *Level 2* (L2) TLB in each processor.
- Fixed 32KB L1 instruction and data caches.
- Shared L2 cache of 512KB, 1MB, 2MB, or 4MB configurable size.
- Optional *Error Correction Code* (ECC) protection for L1 data cache and L2 cache, and parity protection for L1 instruction cache.
- AMBA 4 *AXI Coherency Extension*s (ACE) master interface.
- *Accelerator Coherency Port* (ACP) that is implemented as an AXI3 slave interface.
- *Program Trace Macrocell* (PTM) based on version 1.1 of the *Program Flow Trace* (PFTv1.1) architecture.
- *Performance Monitor Unit* (PMU) based on PMUv2 architecture.
- *Cross trigger interfaces* for multi-processor debugging.
- VFP component only or optionally implemented VFP and NEON components.
- Optional *Generic Interrupt Controller* (GIC) that supports up to 224 *Shared Peripheral Interrupts* (SPIs).
- ARM generic 64-bit timers for each processor.
- Support for power management with multiple power domains.

1.3.1 Test features

The Cortex-A15 MPCore processor provides several test signals that enable the use of both ATPG and MBIST to test the Cortex-A15 MPCore processor and its memory arrays. See Appendix A *Signal Descriptions* for more information.

1.4 Interfaces

The processor has the following external interfaces:

- Memory interface that implements an ACE master inferface.
- ACP that implements as an AXI slave interface.
- Debug interface that implements an APB slave interface.
- Trace interface that implements an ATB interface.
- Cross trigger interface.
- DFT interface.
- MBIST controller interface.

See *Interfaces* on page 2-6 for more information on each of these interfaces.

1.5 Implementation options

Table 1-1 lists the implementation options for the Cortex-A15 MPCore processor.

Table 1-1 Cortex-A15 MPCore processor implementation options

Feature	Range of options
Number of processors	Up to four processors
L2 cache size	L2 cache size of: • 512KB. • 1MB. • 2MB. • 4MB.
L2 tag RAM register slice	0 or 1
L2 data RAM register slice	0, 1 or 2
L2 arbitration register slice	Included or Not
L2 logic idle gated clock	Included or Not
Regional gated clocks[a]	Included or Not
ECC/parity support	Supported in L1 and L2, L2 only, or none
NEON	Included or Not
VFP	Included or Not
Generic Interrupt Controller	Included or Not
Shared Peripheral Interrupts	0 to 224, in steps of 32
Processor clock stop pins[a]	Included or Not
Power switch and clamp pins	Included or Not
L1 data TLB page table size	L1 data TLB page table size of: • 4KB entries only. • 4KB or 1MB entries.

[a] This feature is not available in revisions prior to r3p0.

Note

- All the processors share an integrated L2 cache and GIC. Each processor has the same configuration for NEON, VFP, and L1 ECC or parity.
- If you configure the design for one processor, it retains the system level coherency support and the ACP slave port.
- If you configure the design to exclude VFP, NEON is not available. You cannot configure the design to exclude VFP but include NEON.
- If you configure the design to exclude the GIC, SPIs and the remaining GIC signals are not available, except **PERIPHBASE[39:15]**.
- The L2 tag RAM register slice option adds register slices to the L2 tag RAMs. The L2 data RAM register slice option adds register slices to the L2 data RAMs. Table 1-2 lists valid combinations of the L2 tag RAM and L2 data RAM register slice options.

- If L2 arbitration register slice is included, an additional pipeline stage for the CPU-L2 arbitration logic interface is added to the L2 arbitration logic.
- If L2 logic idle clock gating is present, most of the L2 logic is dynamically clock gated with a different clock than the GIC and Generic Timer. If L2 logic idle clock gating is not present, the L2 logic is not dynamically clock gated, and shares the same clock as the GIC and Generic Timer. The clock gate generator for the L2 logic is also removed. Having dynamic clock gating of the L2 logic can provide lower power dissipation, but at the cost of a more complex clock tree implementation.
- If regional clock gating is present, an additional level of clock gating occurs for several logic blocks such as the register banks. Having regional clock gating can potentially provide lower power dissipation, but at the cost of a more complex clock tree implementation.

Table 1-2 shows valid combinations of the L2 tag RAM and L2 data RAM register slice options.

Table 1-2 Valid combinations of L2 tag and data RAM register slice

L2 tag RAM register slice	L2 data RAM register slice
0	0
0	1
0	2
1	1
1	2

1.6 Product documentation and design flow

This section describes the Cortex-A15 MPCore processor books and how they relate to the design flow in:

- *Documentation*.
- *Design flow* on page 1-10.

See *Additional reading* on page ix for more information about the books described in this section. For information on the relevant architectural standards and protocols, see *Compliance* on page 1-3.

1.6.1 Documentation

The Cortex-A15 MPCore processor documentation is as follows:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the Cortex-A15 MPCore processor. It is required at all stages of the design flow. The choices made in the design flow can mean that some behavior described in the TRM is not relevant. If you are programming the Cortex-A15 MPCore processor, additional information must be obtained from:

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

Note

- The out-of-order design of the Cortex-A15 MPCore processor pipeline makes it impossible to provide accurate timing information for complex instructions. The timing of an instruction can be affected by factors such as:
  - Other concurrent instructions.
  - Memory system activity.
  - Events outside the instruction flow.
- Timing information has been provided in the past for some ARM processors to assist in detailed hand tuning of performance critical code sequences or in the development of an instruction scheduler within a compiler. This timing information is not required for producing optimized instruction sequences on the Cortex-A15 MPCore processor. The out-of-order pipeline of the Cortex-A15 MPCore processor can schedule and execute the instructions in an optimal fashion without any instruction reordering required.

Configuration and Sign-off Guide

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

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

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

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

1.6.2 Design flow

The Cortex-A15 MPCore processor is delivered as synthesizable RTL. Before the processor can be used in a product, it must go through the following process:

Implementation

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

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

Programming

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

- Software required to configure the Cortex-A15 MPCore processor.
- Software required to initialize the Cortex-A15 MPCore processor.
- Application software and the SoC tests.

Each process:

- Can be performed by a different party.
- Can include implementation and integration choices that affect the behavior and features of the Cortex-A15 MPCore processor.

The operation of the final device depends on:

Build configuration

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

Configuration inputs

The integrator configures some features of the Cortex-A15 MPCore processor by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made. They can also limit the options available to the software.

Software configuration

The programmer configures the Cortex-A15 MPCore processor by programming particular values into registers. This affects the behavior of the Cortex-A15 MPCore processor.

Note

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

1.7 Product revisions

This section describes the differences in functionality between product revisions.

1.7.1 r0p0 - r0p1

The following change has been made in this release:

- ID register value changed to reflect product revision status:

  Main ID Register `0x410FC0F1`

  Debug ID Register `0x3515F001`

- Various engineering errata fixes.

1.7.2 r0p1 - r0p2

The following change has been made in this release:

- ID register value changed to reflect product revision status:

  Main ID Register `0x410FC0F2`

  Debug ID Register `0x3515F002`

- Various engineering errata fixes.

1.7.3 r0p2 - r0p3

The following change has been made in this release:

- ID register value changed to reflect product revision status:

  Main ID Register `0x410FC0F3`

  Debug ID Register `0x3515F003`

- Various engineering errata fixes.

1.7.4 r0p3 - r1p0

The following changes have been made in this release:

- ID register values changed to reflect product revision status:

  Main ID Register `0x411FC0F0`

  Debug ID Register `0x3515F010`

  ETM ID Register `0x411CF311`

  Peripheral ID2 Register

  `0x0000001B`

- Various engineering errata fixes.

1.7.5 r1p0 - r2p0

The following changes have been made in this release:

- ID register values changed to reflect product revision status:

Main ID Register	`0x412FC0F0`
Debug ID Register	`0x3515F020`
ETM ID Register	`0x411CF312`
Peripheral ID2 Register	`0x0000002B`

- The input signals, **nVIRQ** and **nVFIQ**, are always present regardless of whether the GIC is present or not. See *GIC configuration* on page 8-6.
- L2ACTLR bit[5] is now reserved, RAZ/WI. See *L2 Auxiliary Control Register* on page 4-100.
- Renamed PMCCFILTR to PMXEVTYPER31 in the PMU register summary table. See Table 11-1 on page 11-4.
- Various engineering errata fixes.

1.7.6 r2p0 - r2p1

The following changes have been made in this release:

- ID register values changed to reflect product revision status:

Main ID Register	`0x412FC0F1`
Debug ID Register	`0x3515F021`

- Various engineering errata fixes.

1.7.7 r2p1 - r2p2

The following changes have been made in this release:

- ID register values changed to reflect product revision status:

Main ID Register	`0x412FC0F2`
Debug ID Register	`0x3515F022`

- Various engineering errata fixes.

1.7.8 r2p2 - r3p0

The following changes have been made in this release:

- ID register values changed to reflect product revision status:

Main ID Register	`0x413FC0F0`
Debug ID Register	`0x3515F030`
ETM ID Register	`0x411CF313`
Peripheral ID2 Register	`0x0000003B`

- Added processor clock stop pins, **CPUCLKOFF**, configurable option. See *Implementation options* on page 1-7 and *Clocks* on page 2-8.

- Added regional clock gating configurable option. See *Implementation options* on page 1-7 and *Dynamic power management* on page 2-21.
- Added processor retention in WFI and WFE low-power state. See *Processor retention in WFI and WFE low-power state* on page 2-24.
- Added L2ACTLR bits[26, 16:11]. See *L2 Auxiliary Control Register* on page 4-100.
- Added ACTLR2 Register. See *Auxiliary Control Register 2* on page 4-105.
- Various engineering errata fixes.

1.7.9 r3p0 - r3p1

- Added L1 data TLB support for 1MB page table entries. See *Implementation options* on page 1-7 and *L1 data TLB* on page 5-3.
- Various engineering errata fixes.

Chapter 2
Functional Description

This chapter describes the functionality of the Cortex-A15 MPCore processor. It contains the following sections:

- *About the Cortex-A15 MPCore processor functions* on page 2-2.
- *Interfaces* on page 2-6.
- *Clocking and resets* on page 2-8.
- *Power management* on page 2-21.

2.1 About the Cortex-A15 MPCore processor functions

Figure 2-1 shows a top-level functional diagram of the Cortex-A15 MPCore processor.

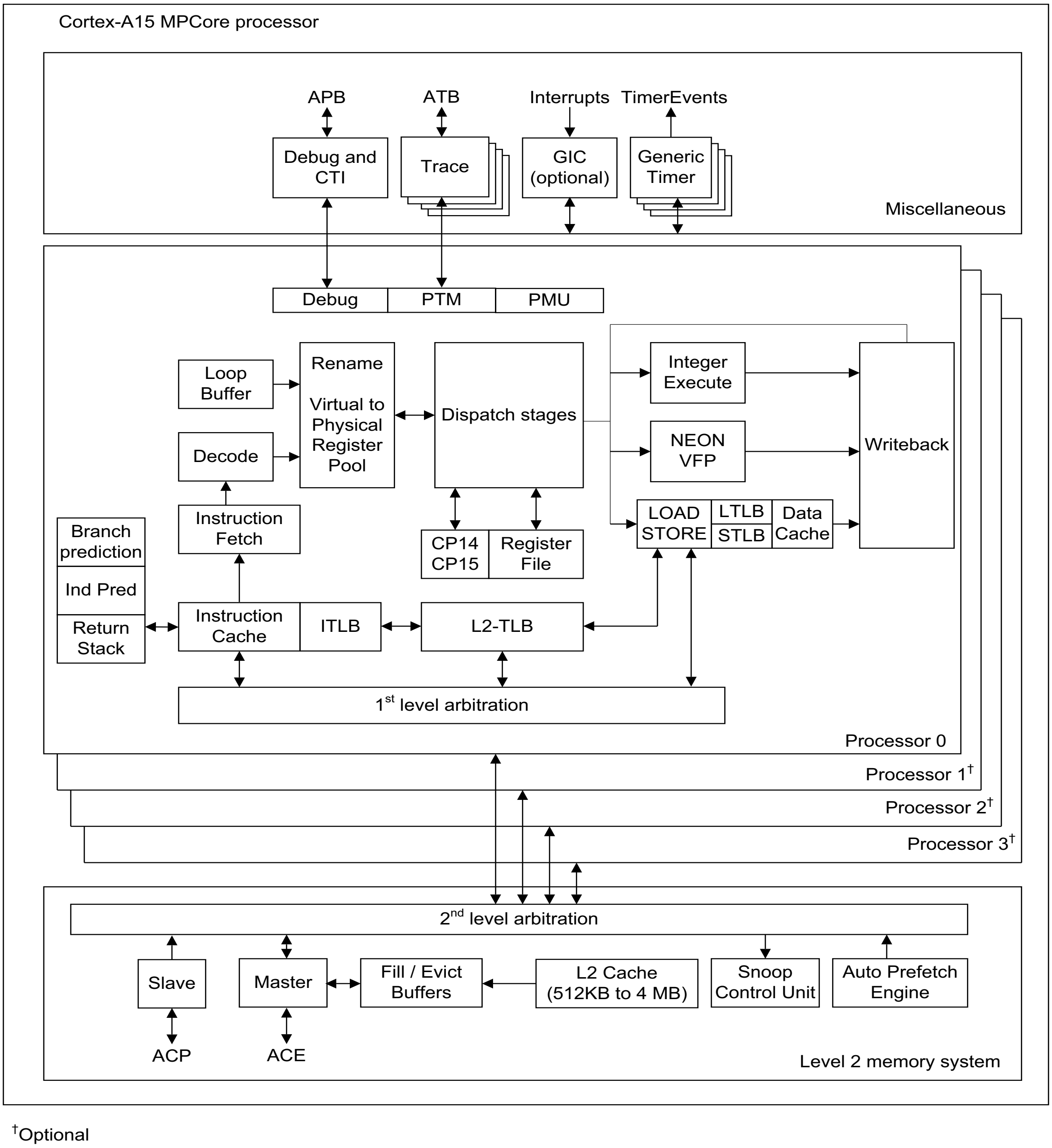


Figure 2-1 Block diagram

2.1.1 Components of the processor

The main components of the processor are:

- *Instruction fetch* on page 2-3.
- *Instruction decode* on page 2-3.
- *Instruction dispatch* on page 2-3.
- *Integer execute* on page 2-3.
- *Load/Store unit* on page 2-4.
- *L2 memory system* on page 2-4.

- *NEON and VFP unit* on page 2-4.
- *Generic Interrupt Controller* on page 2-4.
- *Generic Timer* on page 2-4.
- *Debug and trace* on page 2-4.

Instruction fetch

The instruction fetch unit fetches instructions from the L1 instruction cache and delivers up to three instructions per cycle to the instruction decode unit. It supports dynamic and static branch prediction. The instruction fetch unit includes:

- L1 instruction cache that is a 32KB 2-way set-associative cache with 64 bytes cache line and optional parity protection per 16-bits.
- 2-level dynamic predictor with BTB for fast target generation.
- Return stack.
- Static branch predictor.
- Indirect predictor.
- 32-entry fully-associative L1 instruction TLB.

Instruction decode

The instruction decode unit decodes the following instructions:

- ARM.
- Thumb.
- ThumbEE.
- Advanced SIMD.
- CP14.
- CP15.

The instruction decode unit also performs register renaming to facilitate out-of-order execution by removing *Write-After-Write* (WAW) and *Write-After-Read* (WAR) hazards. A loop buffer provides additional power savings while executing small instruction loops.

Instruction dispatch

The instruction dispatch unit controls when the decoded instructions can be dispatched to the execution pipelines and when the returned results can be retired. It includes:

- The ARM core general purpose registers.
- The Advanced SIMD and VFP extension register set.
- The CP14 and CP15 registers.
- The APSR and FPSCR flag bits.

Integer execute

The integer execute unit includes:

- Two symmetric *Arithmetic Logical Unit* (ALU) pipelines.
- Integer multiply-accumulate pipeline.
- Iterative integer divide hardware.
- Branch and instruction condition codes resolution logic.
- Result forwarding and comparator logic.

Load/Store unit

The load/store unit executes load and store instructions and encompasses the L1 data side memory system. It also services memory coherency requests from the L2 memory system. The load/store unit includes:

- L1 data cache that is a 32KB 2-way set-associative cache with 64 bytes cache line and optional ECC protection per 32-bits.
- Two separate 32-entry fully-associative L1 TLBs, one for data loads and one for data stores.

See Chapter 5 *Memory Management Unit* and Chapter 6 *Level 1 Memory System* for more information.

L2 memory system

The L2 memory system services L1 instruction and data cache misses from each processor. It handles requests on the AMBA 4 ACE master interface and AXI3 ACP slave interface. The L2 memory system includes:

- L2 cache that is:
  - 512KB, 1MB, 2MB, or 4MB configurable size.
  - 16-way set-associative cache with optional ECC protection per 64-bits.
- Duplicate copy of L1 data cache tag RAMs from each processor for handling snoop requests.
- 4-way set-associative of 512-entry L2 TLB in each processor.
- Automatic hardware prefetcher with programmable instruction fetch and load/store data prefetch distances.

See Chapter 7 *Level 2 Memory System* for more information.

NEON and VFP unit

The NEON and VFP unit provides support for the ARMv7 Advanced SIMDv2 and VFPv4 instruction sets. See Chapter 14 *NEON and VFP Unit* for more information.

Generic Interrupt Controller

The GIC provides support for handling multiple interrupt sources. See Chapter 8 *Generic Interrupt Controller* for more information.

Generic Timer

The Generic Timer provides the ability to schedule events and trigger interrupts. See Chapter 9 *Generic Timer* for more information.

Debug and trace

The debug and trace unit includes:

- Support for ARMv7.1 Debug architecture with an APB slave interface for access to the debug registers.
- Performance Monitor Unit based on PMUv2 architecture.

- Program Trace Macrocell based on the CoreSight PFTv1.1 architecture and dedicated ATB interface per processor.
- Cross trigger interfaces for multi-processor debugging.

See the following for more information:

- Chapter 10 *Debug*.
- Chapter 11 *Performance Monitor Unit*.
- Chapter 12 *Program Trace Macrocell*.
- Chapter 13 *Cross Trigger* on page 13-1.

2.2 Interfaces

The processor has the following external interfaces:

- *ACE master interface*.
- *ACP slave interface*.
- *APB slave interface*.
- *ATB interface*.
- *Cross trigger interface*.
- *DFT interface* on page 2-7.
- *MBIST controller interface* on page 2-7.

2.2.1 ACE master interface

The processor implements an AMBA 4 *AXI Coherency Extension*s (ACE) master interface. See the *ARM® AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™, ACE and ACE-Lite™* for more information.

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

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

2.2.2 ACP slave interface

The processor implements an AMBA 3 AXI *Accelerator Coherency Port* (ACP) slave interface. See the *ARM® AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™, ACE and ACE-Lite™* for more information.

ACP is an implementation of an AMBA 3 AXI slave interface. It supports memory coherent accesses to the processor memory system, but cannot receive coherent requests, barriers or distributed virtual memory messages.

2.2.3 APB slave interface

The processor implements an AMBA 3 APB slave interface that enables access to the debug registers. See the *ARM CoreSight Architecture Specification* for more information.

2.2.4 ATB interface

The processor implements dedicated AMBA 3 ATB interfaces for each processor that outputs trace information for debugging. The ATB interface is compatible with the CoreSight architecture. See the *ARM® CoreSight™ Architecture Specification* for more information.

2.2.5 Cross trigger interface

The processor implements a single cross trigger channel interface. This external interface is connected to the CoreSight *Cross Trigger Interface* (CTI) corresponding to each processor through a simplified *Cross Trigger Matrix* (CTM). See Chapter 13 *Cross Trigger* for more information.

2.2.6 DFT interface

The processor implements a *Design For Test* (DFT) interface that enables an industry standard *Automatic Test Pattern Generation* (ATPG) tool to test logic outside of the embedded memories. See *DFT and MBIST interfaces* on page A-26 for information on these test signals.

2.2.7 MBIST controller interface

The *Memory Built In Self Test* (MBIST) controller interface provides support for manufacturing testing of the memories embedded in the Cortex-A15 MPCore processor. MBIST is the industry standard method of testing embedded memories. MBIST works by performing sequences of reads and writes to the memory based on test algorithms. See *MBIST interface* on page A-26 for information on the MBIST signals.

2.3 Clocking and resets

This section describes the clocks and resets of the processor in:

- *Clocks*.
- *Resets* on page 2-11.

2.3.1 Clocks

The processor has the following clock inputs:

CLK This is the main clock of the Cortex-A15 MPCore processor. All processors, the shared L2 memory system logic, the GIC, and the Generic Timer are clocked with a distributed version of **CLK**.

PCLKDBG This is the APB clock that controls the Debug APB, CTI and CTM logic in the **PCLKDBG** domain. **PCLKDBG** is asynchronous to **CLK**.

The processor has the following clock enable inputs:

ACLKENM The AXI master interface is a synchronous AXI interface that can operate at any integer multiple that is equal to or slower than the main processor clock, **CLK**, using the **ACLKENM** signal. For example, you can set the **CLK** to **ACLKM** frequency ratio to 1:1, 2:1, or 3:1, where **ACLKM** is the AXI master clock. **ACLKENM** asserts one **CLK** cycle prior to the rising edge of **ACLKM.** Software can change the **CLK** to **ACLKM** frequency ratio dynamically using **ACLKENM**.

Figure 2-2 shows a timing example of **ACLKENM** that changes the **CLK** to **ACLKM** frequency ratio from 3:1 to 1:1.

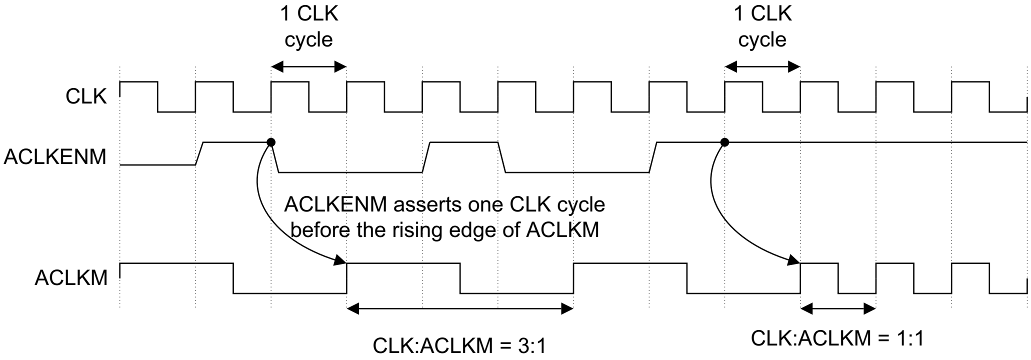


Figure 2-2 ACLKENM with CLK:ACLKM ratio changing from 3:1 to 1:1

Note

Figure 2-2 shows the timing relationship between the AXI master clock, **ACLKM** and **ACLKENM**, where **ACLKENM** asserts one **CLK** cycle before the rising edge of **ACLKM**. It is important that the relationship between **ACLKM** and **ACLKENM** is maintained.

ACLKENS ACP is a synchronous AXI slave interface that can operate at any integer multiple that is equal to or slower than the main processor clock, **CLK**, using the **ACLKENS** signal. For example, the **CLK** to **ACLKS** frequency ratio can be 1:1, 2:1, or 3:1, where **ACLKS** is the AXI slave clock. **ACLKENS** asserts one **CLK** cycle before the rising edge of **ACLKS**. The **CLK** to **ACLKS** frequency ratio can be changed dynamically using **ACLKENS**.

Figure 2-3 shows a timing example of **ACLKENS** that changes the **CLK** to **ACLKS** frequency ratio from 3:1 to 1:1.

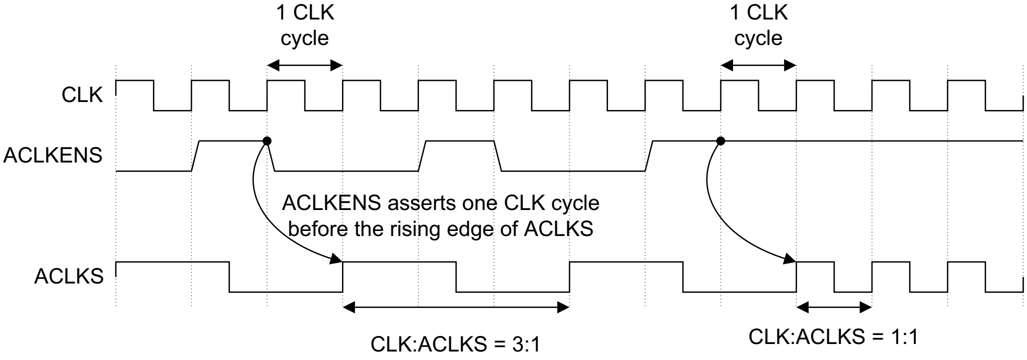


Figure 2-3 ACLKENS with CLK:ACLKS ratio changing from 3:1 to 1:1

Note

Figure 2-3 shows the timing relationship between the ACP clock, **ACLKS** and **ACLKENS**, where **ACLKENS** asserts one **CLK** cycle before the rising edge of **ACLKS**. It is important that the relationship between **ACLKS** and **ACLKENS** is maintained.

PCLKENDBG

The Debug APB interface is an asynchronous interface that can operate at any integer multiple that is equal to or slower than the APB clock, **PCLKDBG**, using the **PCLKENDBG** signal. For example, the **PCLKDBG** to internal **PCLKDBG** frequency ratio can be 1:1, 2:1, or 3:1. **PCLKENDBG** asserts one **PCLKDBG** cycle before the rising edge of the internal **PCLKDBG**. The **PCLKDBG** to internal **PCLKDBG** frequency ratio can be changed dynamically using **PCLKENDBG**.

Figure 2-4 shows a timing example of **PCLKENDBG** that changes the **PCLKDBG** to internal **PCLKDBG** frequency ratio from 2:1 to 1:1.

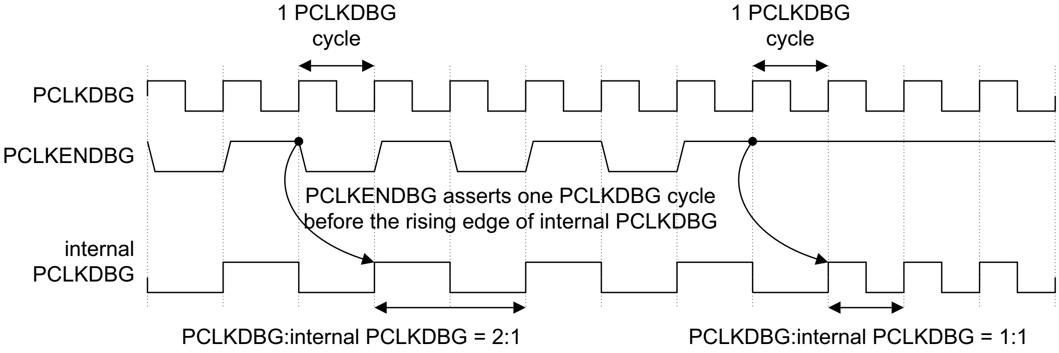

1 PCLKDBG cycle
1 PCLKDBG cycle
PCLKDBG
PCLKENDBG
PCLKENDBG asserts one PCLKDBG cycle before the rising edge of internal PCLKDBG
internal PCLKDBG
PCLKDBG:internal PCLKDBG = 2:1
PCLKDBG:internal PCLKDBG = 1:1

Figure 2-4 PCLKENDBG with PCLKDBG:internal PCLKDBG ratio changing from 2:1 to 1:1

ATCLKEN The ATB interface is a synchronous interface that can operate at any integer multiple that is slower than the main processor clock, **CLK**, using the **ATCLKEN** signal. For example, the **CLK** to **ATCLK** frequency ratio can be 2:1, 3:1, or 4:1, where **ATCLK** is the ATB bus clock. **ATCLKEN** asserts three **CLK** cycles before the rising edge of **ATCLK**. Three **CLK** cycles are required to permit propagation delay from the **ATCLKEN** input to the processor. The **CLK** to **ATCLK** frequency ratio can be changed dynamically using **ATCLKEN**.

Figure 2-5 shows a timing example of **ATCLKEN** where the **CLK** to **ATCLK** frequency ratio is 2:1.

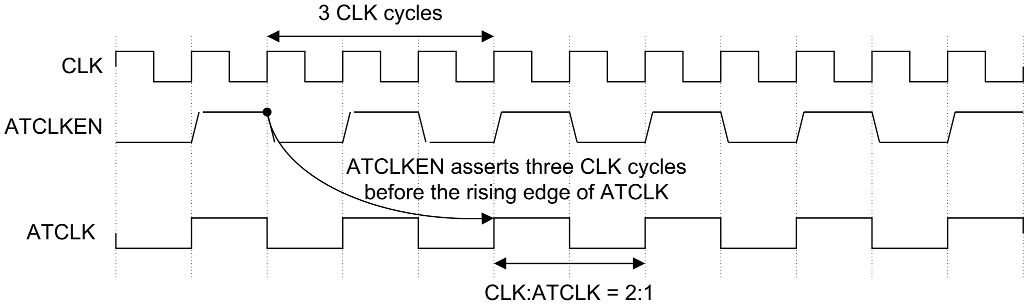

3 CLK cycles
CLK
ATCLKEN
ATCLKEN asserts three CLK cycles
before the rising edge of ATCLK
ATCLK
CLK:ATCLK = 2:1

Figure 2-5 ATCLKEN with CLK:ATCLK ratio at 2:1

PERIPHCLKEN

This is the synchronous clock enable signal for the GIC. The GIC can operate at any integer multiple that is slower than the main processor clock, **CLK**, using the **PERIPHCLKEN** signal. For example, the **CLK** to internal GIC clock frequency ratio can be 2:1 or 3:1. The GIC does not support a 1:1 clock frequency ratio. **PERIPHCLKEN** asserts one **CLK** cycle prior to the rising edge of the internal IC clock. The **CLK** to internal IC clock frequency ratio can be changed dynamically using **PERIPHCLKEN**.

Note

If you configure your design to exclude the GIC, this signal does not exist.

Figure 2-6 shows a timing example of **PERIPHCLKEN** where the **CLK** to internal GIC frequency ratio is 2:1.

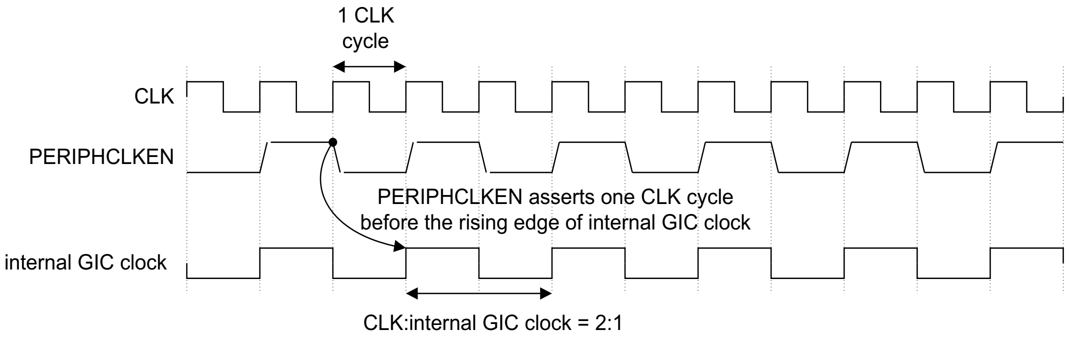

1 CLK
cycle
CLK
PERIPHCLKEN
PERIPHCLKEN asserts one CLK cycle
before the rising edge of internal GIC clock
internal GIC clock
CLK:internal GIC clock = 2:1

Figure 2-6 PERIPHCLKEN with CLK:internal IC clock ratio at 2:1

CLKEN This is the main clock enable for all internal clocks in the Cortex-A15 MPCore processor that are derived from **CLK**. There is one **CLK** cycle delay between the assertion of **CLKEN** and the internal clocks that are enabled. When all the processors and L2 are in WFI low-power state, you can place the processor in a low-power state using the **CLKEN** input. This disables all internal clocks, excluding the asynchronous Debug APB **PCLKDBG** domain. See *L2 Wait for Interrupt* on page 2-23.

CPUCLKOFF

Note

This configuration option is not available in revisions prior to r3p0.

These pins are only present if you configure the Cortex-A15 MPCore processor to include them. If you configure the processor to include the **CPUCLKOFF** pins, there is one **CPUCLKOFF** input pin and a new top-level clock gate instantiated for each processor. When **CPUCLKOFF** is asserted, the processor clock is stopped. This pin must be tied LOW or 1'b0 in normal functional operation, and can only be asserted under strict conditions. Having external control of the processor clock enable permits the SoC to assert **CPUCLKOFF** when the processor is already powered down, or when the processor is powered up. However, **CPUCLKOFF** must be deasserted after power has been completely restored, at least 16 cycles before the deassertion of the processor reset, and at least 16 cycles before releasing the clamps on the processor outputs, to permit the powerup reset sequence to complete.

Note

Because configuring the Cortex-A15 MPCore processor with the **CPUCLKOFF** pins adds a new top-level clock gate for each processor, it might increase the clock skew between the processors.

2.3.2 Resets

The processor has the following reset inputs:

nCPUPORESET[3:0]

The **nCPUPORESET** signal initializes all the processor logic, including the NEON and VFP logic, Debug, PTM, breakpoint and watchpoint logic in the processor **CLK** domain. Each processor has one **nCPUPORESET** reset input.

nCORERESET[3:0]

The **nCORERESET** signal initializes the processor logic, including the NEON and VFP logic but excludes the Debug, PTM, breakpoint and watchpoint logic. Each processor has one **nCORERESET** reset input.

nCXRESET[3:0]

The **nCXRESET** signal initializes the NEON and VFP logic. This reset can be used to hold the NEON and VFP unit in a reset state so that the power to the unit can be safely applied during power up. Each processor has one **nCXRESET** reset input.

nDBGRESET[3:0]

The **nDBGRESET** signal initializes the Debug, PTM, breakpoint and watchpoint logic in the processor **CLK** domain. Each processor has one **nDBGRESET** reset input.

nPRESETDBG

The **nPRESETDBG** signal initializes the shared Debug APB, CTI, and CTM logic in the **PCLKDBG** domain.

nL2RESET

The **nL2RESET** signal initializes the shared L2 memory system, Interrupt Controller, and Timer logic.

All resets are active-LOW inputs. The reset signals lets you reset different parts of the processor independently. Table 2-12the pr c Tw[(2ntrolll re4(ed by-5(s variouTw[(25 Tw6.2(ll re4on)-8.3-.6(27-19.747 -1.27 0 7-0

In Table 2-2, [3:0] specifies the processor configuration and [n] designates the processor that is reset.

Table 2-2 Valid reset combinations

Reset combination	Signals	Value	Description

Full powerup reset for entire Cortex-A15 MPCore processor	**nCPUPORESET [3:0]**
	nCORERESET [3:0]
	nCXRESET [3:0]
	nDBGRESET [3:0]
	nPRESETDBG
	nL2RESET
		all = 0
		all = 0[a]
		all = 0[a]
		all = 0[a]
		0
		0
			All logic is held in reset.

Individual processor powerup reset with Debug (**PCLKDBG**) reset	**nCPUPORESET [3:0]** **nCORERESET [3:0]** **nCXRESET [3:0]** **nDBGRESET [3:0]** **nPRESETDBG** **nL2RESET**	[n] = 0 [n] = 0[a] [n] = 0[a] [n] = 0[a] 0 1	Individual processor and Debug (**PCLKDBG**) are held in reset, so that the processor and Debug (**PCLKDBG**) can be powered up.

All processor and L2 reset with Debug (**PCLKDBG**) active	**nCPUPORESET [3:0]**
	nCORERESET [3:0]
	nCXRESET [3:0]
	nDBGRESET [3:0]
	nPRESETDBG
	nL2RESET
		all = 0
		all = 0[a]
		all = 0[a]
		all = 0[a]
		1
		0
			All processors and L2 are held in reset, so they can be powered up. This enables external debug over power down for all processors.

Individual processor powerup reset with Debug (**PCLKDBG**) active	**nCPUPORESET [3:0]** **nCORERESET [3:0]** **nCXRESET [3:0]** **nDBGRESET [3:0]** **nPRESETDBG** **nL2RESET**	[n] = 0 [n] = 0[a] [n] = 0[a] [n] = 0[a] 1 1	Individual processor is held in reset, so that the processor can be powered up. This enables external debug over power down for the processor that is held in reset.

All processors software reset	**nCPUPORESET [3:0]**
	nCORERESET [3:0]
	nCXRESET [3:0]
	nDBGRESET [3:0]
	nPRESETDBG
	nL2RESET
		all = 1
		all = 0
		all = 0[b]
		all = 1
		1
		1
			All logic excluding Debug and PTM (**CLK** and **PCLKDBG**) and L2 are held in reset. All breakpoints and watchpoints are retained.

All processors software reset and L2 reset	**nCPUPORESET [3:0]**
	nCORERESET [3:0]
	nCXRESET [3:0]
	nDBGRESET [3:0]
	nPRESETDBG
	nL2RESET
		all = 1
		all = 0
		all = 0[b]
		all = 1
		1
		0
			All logic excluding Debug and PTM (**CLK** and **PCLKDBG**) is held in reset. All breakpoints and watchpoints are retained.

Table 2-2 Valid reset combinations (continued)

Reset combination	Signals	Value	Description
Individual processor software reset	**nCPUPORESET [3:0]** **nCORERESET [3:0]** **nCXRESET [3:0]** **nDBGRESET [3:0]** **nPRESETDBG** **nL2RESET**	[n] = 1 [n] = 0 [n] = 0[b] [n] = 1 1 1	Individual processor logic excluding Debug and PTM (**CLK**) is held in reset. Breakpoints and watchpoints for that processor are retained.
NEON and VFP reset	**nCPUPORESET [3:0]** **nCORERESET [3:0]** **nCXRESET [3:0]** **nDBGRESET [3:0]** **nPRESETDBG** **nL2RESET**	[n] = 1 [n] = 1 [n] = 0 [n] = 1 1 1	NEON and VFP unit is held in reset, so that the unit can be powered up.

All processors Debug (**CLK**) and Debug (**PCLKDBG**) reset	**nCPUPORESET [3:0]**
	nCORERESET [3:0]
	nCXRESET [3:0]
	nDBGRESET [3:0]
	nPRESETDBG
	nL2RESET
		all = 1
		all = 1
		all = 1
		all = 0
		0
		1
			Debug and PTM (**CLK** and **PCLKDBG**) are held in reset.

Individual processor Debug (**CLK**) reset	**nCPUPORESET [3:0]** **nCORERESET [3:0]** **nCXRESET [3:0]** **nDBGRESET [3:0]** **nPRESETDBG** **nL2RESET**	[n] = 1 [n] = 1 [n] = 1 [n] = 0 1 1	Individual processor Debug and PTM (**CLK**) is held in reset.
Debug (**PCLKDBG**) reset	**nCPUPORESET [3:0]** **nCORERESET [3:0]** **nCXRESET [3:0]** **nDBGRESET [3:0]** **nPRESETDBG** **nL2RESET**	all = 1 all = 1 all = 1 all = 1 0 1	Debug (**PCLKDBG**) is held in reset, so that Debug (**PCLKDBG**) can be powered up.

Table 2-2 Valid reset combinations (continued)

Reset combination	Signals	Value	Description
NEON and VFP and Debug (**PCLKDBG**) reset	**nCPUPORESET [3:0]** **nCORERESET [3:0]** **nCXRESET [3:0]** **nDBGRESET [3:0]** **nPRESETDBG** **nL2RESET**	[n] = 1 [n] = 1 [n] = 0 [n] = 1 0 1	NEON and VFP and Debug (**PCLKDBG**) are held in reset, so that NEON and VFP and Debug (**PCLKDBG**) can be powered up.
NEON and VFP and Debug (**CLK**) reset	**nCPUPORESET [3:0]** **nCORERESET [3:0]** **nCXRESET [3:0]** **nDBGRESET [3:0]** **nPRESETDBG** **nL2RESET**	[n] = 1 [n] = 1 [n] = 0 [n] = 0 1 1	NEON and VFP and Debug (**CLK**) are held in reset.
Run mode	**nCPUPORESET [3:0]** **nCORERESET [3:0]** **nCXRESET [3:0]** **nDBGRESET [3:0]** **nPRESETDBG** **nL2RESET**	1 1 1 1 1 1	No logic is held in reset.

a. For powerup reset or processor reset, **nCPUPORESET** must be asserted. The remaining processor resets, **nCORERESET**, **nCXRESET**, and **nDBGRESET** can be asserted, but is not required.

b. For soft reset, **nCORERESET** must be asserted, **nCXRESET** can be asserted, but is not required.

Note

- **nL2RESET** resets the shared L2 memory system logic, GIC and Generic Timer that is common to all processors. This reset must not be asserted while any individual processor is active.
- **nPRESETDBG** resets the shared Debug in **PCLKDBG** domain that is common to all processors. This reset must not be asserted while any individual processor is actively being debugged in normal operating mode or during external debug over power down.
- If your implementation does not include the NEON and VFP unit, you can tie the **nCXRESET** input HIGH.

There are specific requirements that you must meet to reset each reset domain listed in Table 2-1 on page 2-12. Not adhering to these requirements can lead to a reset domain that is not functional.

The reset sequences in the following sections are the only reset sequences that ARM recommends. Any deviation from these sequences might cause an improper reset of that reset domain.

The supported reset sequences are:

- *Powerup reset* on page 2-16.
- *Soft reset* on page 2-17.
- *NEON and VFP reset* on page 2-18.
- *Debug PCLKDBG reset* on page 2-19.
- *Memory arrays reset* on page 2-19.

Powerup reset

The full powerup reset initializes all logic in the Cortex-A15 MPCore processor. You must apply powerup reset to the Cortex-A15 MPCore processor when power is first applied to the SoC. Logic in all clock domains are placed in a benign state following the deassertion of the reset sequence.

Figure 2-7 shows the full powerup reset sequence for the Cortex-A15 MPCore processor.

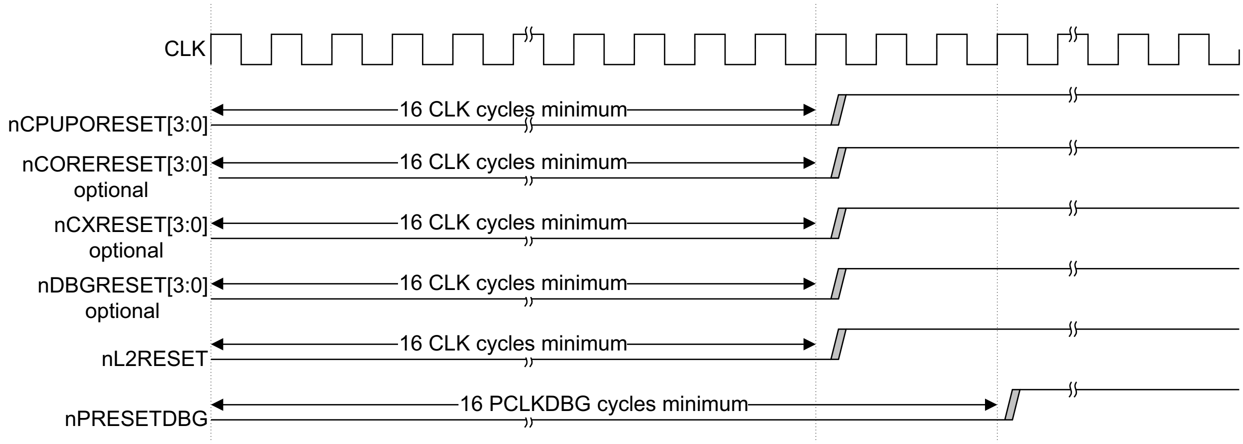


Figure 2-7 Powerup reset timing

On full powerup reset for the Cortex-A15 MPCore processor, perform the following reset sequence:

1. Apply **nCPUPORESET**, **nL2RESET**, and **nPRESETDBG**. The remaining processor resets, **nCORERESET**, **nCXRESET**, and **nDBGRESET** can be asserted, but is not required.
2. **nCPUPORESET** and **nL2RESET** must be asserted for at least 16 **CLK** cycles. **nPRESETDBG** must be asserted for at least16 **PCLKDBG** cycles. Holding the resets for this duration ensures that the resets have propagated to all locations within the processor.
3. **nL2RESET** must be deasserted in the same cycle as the processor resets, or before any of the processor resets are deasserted.

Individual processor powerup reset initializes all logic in a single processor. You must apply the powerup reset when the individual processor is in powered state. In implementations where each processor has its own power supply, the powerup reset holds the processor in a reset state so that power to the processor can be safely applied. You must apply the correct sequence before applying a powerup reset to that processor.

For individual processor powerup reset:

- **nCPUPORESET** for that processor must be asserted for at least 16 **CLK** cycles.
- **nL2RESET** must not be asserted while any individual processor is active.
- **nPRESETDBG** must not be asserted while any individual processor is actively being debugged in normal operating mode or during external debug over power down.

Soft reset

The full soft reset initializes all logic in each of the individual processor apart from the Debug and PTM logic in the **CLK** domain. All breakpoints and watchpoints are retained during a soft reset sequence. By asserting only **nCORERESET**, the reset domains controlled by **nDBGRESET**, **nPRESETDBG**, and **nL2RESET**, that is, the Debug and PTM in **CLK**, Debug APB in **PCLKDBG**, and the shared L2 memory system, GIC, and Generic Timer domains, are not reset.

Figure 2-8 shows the full soft reset sequence for the Cortex-A15 MPCore processor.

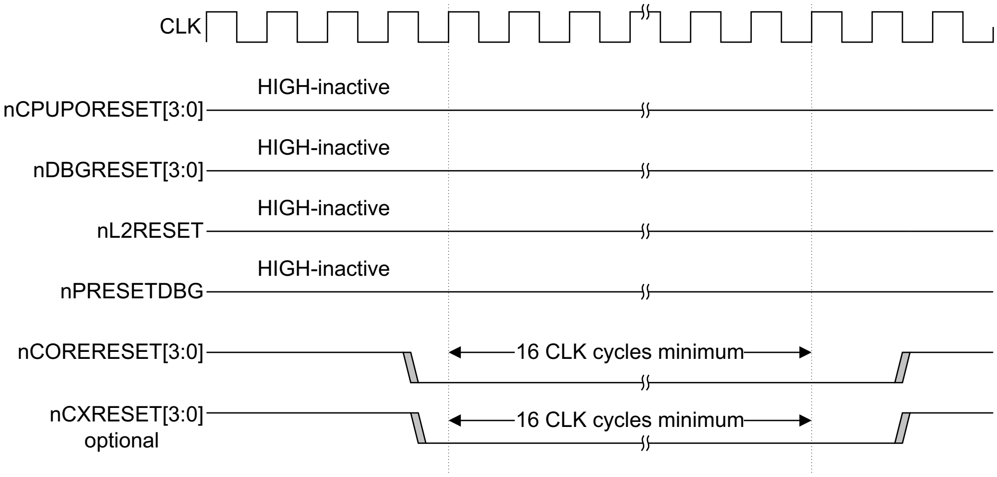


Figure 2-8 Soft reset timing

On full soft reset for the Cortex-A15 MPCore processor, perform the following reset sequence:

1. You must apply steps 1 to 8 in the processor powerdown sequence, see *Processor power domain* on page 2-30, and wait until **STANDBYWFI** is asserted, indicating that the processor is idle, before asserting **nCORERESET** for that processor.
2. Apply **nCORERESET**, **nCXRESET** can be asserted, but is not required.
3. After both resets have been asserted for 5 cycles, the clamps can be released.
4. **nCORERESET** must be asserted for at least 16 **CLK** cycles.
5. If **nCXRESET** is asserted, both resets must be deasserted in the same cycle.

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

For individual processor soft reset:

- You must apply steps 1 to 8 in the processor powerdown sequence, see *Processor power domain* on page 2-30, and wait until **STANDBYWFI** is asserted, indicating that the processor is idle, before asserting **nCORERESET** for that processor.

  Note

  For a single processor configuration you can omit step 3 that clears the ACTLR SMP bit.

- **nCORERESET** for that processor must be asserted for at least 16 **CLK** cycles.
- **nL2RESET** must not be asserted while any individual processor is active.

- **nPRESETDBG** must not be asserted while any individual processor is actively being debugged in normal operating mode.

NEON and VFP reset

An additional reset controls the NEON and VFP unit, independently of the processor reset. You can use this reset to hold the NEON and VFP unit in a reset state so that the power to this unit can be safely applied during power up.

The reset cycle timing requirements for **nCXRESET** are identical to those for **nCORERESET**. **nCXRESET** must be held for a minimum of 16 **CLK** cycles when asserted to guarantee that the NEON and VFP unit has entered a reset state.

Figure 2-9 shows the NEON and VFP reset sequence.

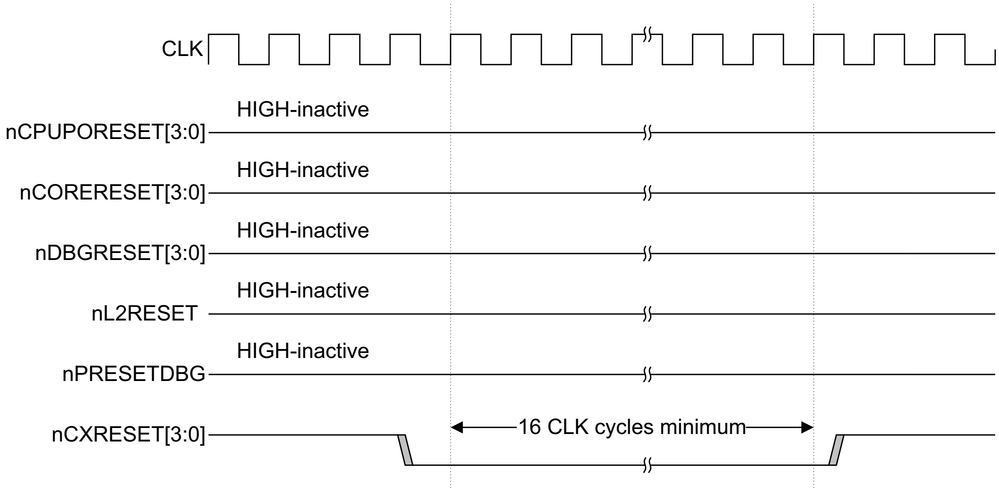


Figure 2-9 NEON and VFP reset timing

Note

If your implementation does not include the NEON and VFP unit, you can tie the **nCXRESET** inputs HIGH.

Debug CLK reset

Use **nDBGRESET** to reset the processor, Debug, PTM, breakpoint and watchpoint logic in the **CLK** domain.

To safely reset the Debug CLK unit, **nDBGRESET** must be asserted for a minimum of 16 **CLK** cycles.

Figure 2-10 on page 2-19 shows the Debug CLK reset sequence.

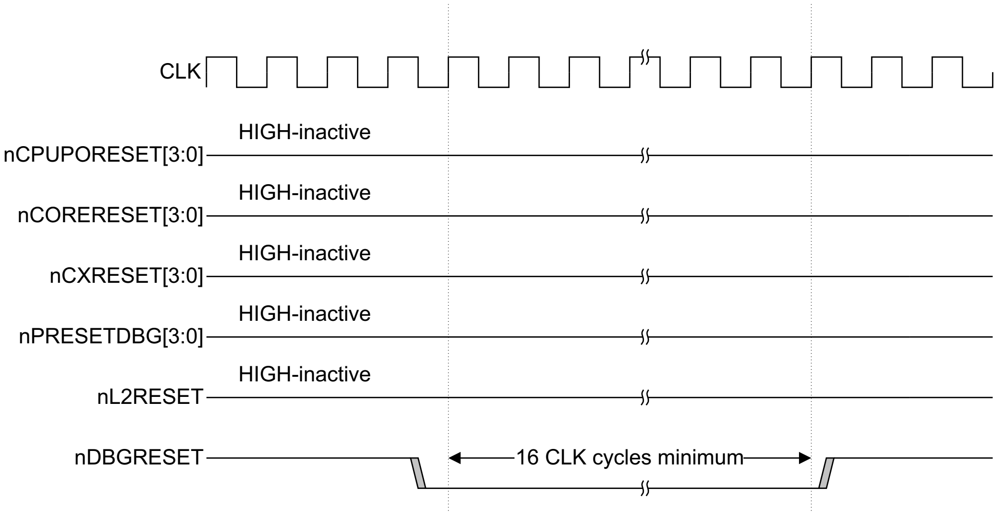


Figure 2-10 Debug CLK reset timing

Debug PCLKDBG reset

Use **nPRESETDBG** to reset the Debug APB, CTI and CTM logic in the **PCLKDBG** domain. This reset holds the Debug PCLKDBG unit in a reset state so that the power to the unit can be safely applied during power up.

To safely reset the Debug PCLKDBG unit, **nPRESETDBG** must be asserted for a minimum of 16 **PCLKDBG** cycles.

Figure 2-11 shows the Debug PCLKDBG reset sequence.

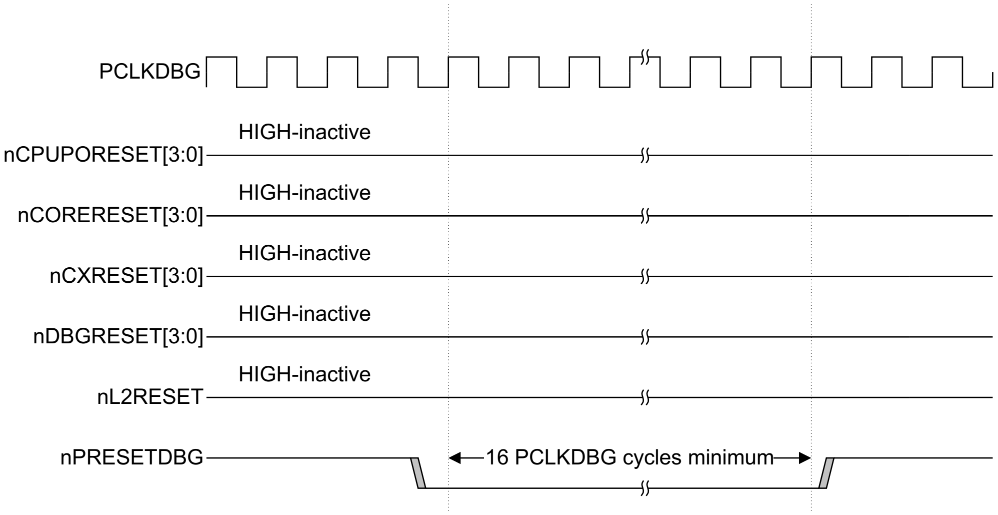


Figure 2-11 Debug PCLKDBG reset timing

Memory arrays reset

During a processor reset, the following memory arrays in the processor are invalidated at reset:

- Branch Prediction arrays such as BTB, GHB, and Indirect Predictor.
- L1 instruction and data TLBs.
- L1 instruction and data caches.

- L2 unified TLB.

In addition to these processor memory arrays, during a powerup reset, the following shareable memory arrays are invalidated at reset:

- L2 duplicate snoop tag RAM.
- L2 prefetch stride queue RAM.
- L2 unified cache RAM, if **L2RSTDISABLE** is tied LOW.

The L1 instruction and data cache resets can take up to 128 **CLK** cycles after the deasserting edge of the reset signals, with each array being reset in parallel. Depending on the size of the L2 cache, the L2 cache reset can take 640 **CLK** cycles for a 512KB L2 cache and 5120 **CLK** cycles for a 4MB L2 cache. The L2 cache reset occurs in the background, in parallel with the L1 cache resets. The processor can begin execution in non-cacheable state, but any attempt to perform cacheable transactions stalls the processor until the appropriate cache reset is complete.

The branch prediction arrays require 512 **CLK** cycles to reset after the deasserting edge of reset. The processor begins execution with branch prediction disabled, any attempt to enable branch prediction with the SCTLR.Z bit, stalls the processor until the branch prediction cache resets are complete.

The Cortex-A15 MPCore processor has an input signal, **L2RSTDISABLE**, that controls the L2 cache hardware reset process. The usage models for the **L2RSTDISABLE** signal are as follows:

- When the Cortex-A15 MPCore processor powers up for the first time, **L2RSTDISABLE** must be held LOW to invalidate the L2 cache using the L2 cache hardware reset mechanism.
- For systems that do not retain the L2 cache RAM contents while the L2 memory system is powered down, **L2RSTDISABLE** must be held LOW to invalidate the L2 cache using the L2 cache hardware reset mechanism.
- For systems that retain the L2 cache RAM contents while the L2 memory system is powered down, **L2RSTDISABLE** must be held HIGH to disable the L2 cache hardware reset mechanism.

The **L2RSTDISABLE** signal is sampled during **nL2RESET** assertion and must be held a minimum of 32 **CLK** cycles after the deasserting edge of **nL2RESET**.

2.4 Power management

The Cortex-A15 MPCore processor provides mechanisms and support to control both dynamic and static power dissipation. The following sections describe:

- *Dynamic power management*.
- *Power domains* on page 2-28.
- *Power modes* on page 2-29.
- *Event communication using WFE and SEV instructions* on page 2-37.

2.4.1 Dynamic power management

This section describes the following dynamic power management features in the Cortex-A15 MPCore processor:

- *Processor Wait for Interrupt*.
- *Processor Wait for Event* on page 2-22.
- *L2 Wait for Interrupt* on page 2-23.
- *Processor retention in WFI and WFE low-power state* on page 2-24.
- *NEON and VFP clock gating* on page 2-27.
- *L2 control and tag banks clock gating* on page 2-27.
- *Regional clock gating* on page 2-28.

Processor Wait for Interrupt

Wait for Interrupt is a feature of the ARMv7-A architecture that puts the processor in a low power state by disabling most of the clocks in the processor while keeping the processor powered up. This reduces the power drawn to the static leakage current, leaving a small clock power overhead to enable the processor to wake up from WFI low-power state.

A processor enters into WFI low-power state by executing the `WFI` instruction.

When executing the `WFI` instruction, the processor waits for all instructions in the processor to retire before entering the idle or low power state. The `WFI` instruction ensures that all explicit memory accesses 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 exclusives instructions.

In addition, the `WFI` instruction ensures that store instructions have updated the cache or have been issued to the L2 memory system.

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

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

Exit from WFI low-power state occurs when the processor detects a reset or one of the WFI wake up events as described in the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

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

Figure 2-12 shows the upper bound for the **STANDBYWFI** deassertion timing after the assertion of **nIRQ** or **nFIQ** inputs.

Figure 2-12 STANBYW612

L2 Wait for Interrupt

When all the processors are in WFI low-power state, the shared L2 memory system logic that is common to all the processors can also enter a WFI low-power state. In L2 WFI low-power state, all internal clocks in the processor are disabled, with the exception of the asynchronous Debug **PCLKDBG** domain.

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

- All processors are in WFI low-power state and therefore, all the processors **STANDBYWFI** outputs are asserted. Assertion of all the processors **STANDBYWFI** outputs guarantee that all the processors are in idle and low power state. All clocks in the processor, with the exception of a small amount of clock wake up logic, are disabled.
- The SoC asserts the input pin **ACINACTM** after all responses are received and before it sends any new transactions on the AXI master interface. This prevents the L2 memory system from accepting any new requests from the AXI master interface and ensures that all outstanding transactions are complete. If the SoC asserts **ACVALIDM** while **ACINACTM** is asserted, then **ACINACTM** must be deasserted for the request to be accepted.
- The SoC asserts the input pin **AINACTS** after all responses are received and before it sends any new transactions on the ACP slave interface. This prevents the L2 memory system from accepting any new requests from the ACP slave interface and ensures that all outstanding transactions are complete. If the SoC asserts **ARVALIDS**, **AWVALIDS**, or **WVALIDS** while **AINACTS** is asserted, then **AINACTS** must be deasserted for the request to be accepted.
- When the L2 memory system completed the outstanding transactions for AXI interfaces, it can then enter the low power state, that is, L2 WFI low-power state. On entry into L2 WFI low-power state, **STANDBYWFIL2** is asserted. Assertion of **STANDBYWFIL2** guarantees that the L2 is in idle and does not accept any new transactions.
- The SoC can then deassert the **CLKEN** input to the Cortex-A15 MPCore processor to stop all remaining internal clocks within the Cortex-A15 MPCore processor that are derived from **CLK**. All clocks in the shared L2 memory system logic, Interrupt Controller and Timer, are disabled.

The SoC must assert the **CLKEN** input on a WFI wake up event to enable the L2 memory system and potentially the processor. There are two classes of wake up events:

- An event that requires only the L2 memory system to be enabled.
- An event that requires both the L2 memory and the processor to be enabled.

The following wake up events cause the L2 memory system and the processor to exit WFI low-power state:

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

Wake up events that only require the L2 memory system to exit WFI low-power state include:

- Deassertion of **ACINACTM** to service an external snoop request on the AC channel interface.
- Deassertion of **AINACTS** to service an ACP transaction on the slave interface.

When the processor exits from WFI low-power state, **STANDBYWFI** for that processor is deasserted. When the L2 memory system logic exits from WFI low-power state, **STANDBYWFIL2** is deasserted.

Figure 2-13 shows the L2 WFI timing for a 4-processor configuration.

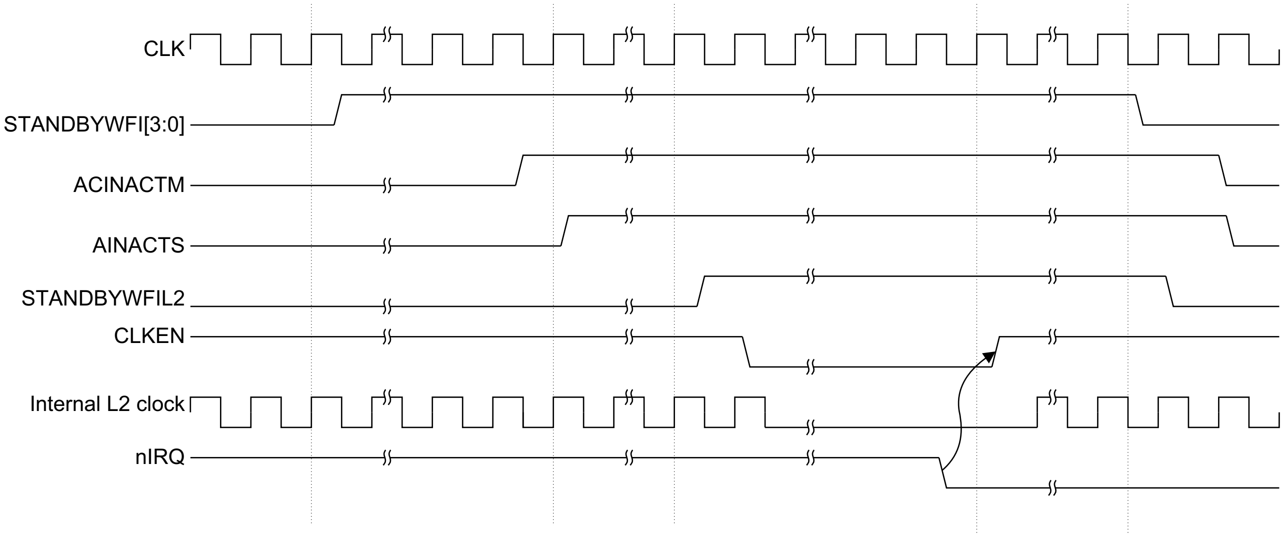


Figure 2-13 L2 Wait For Interrupt timing

Processor retention in WFI and WFE low-power state

Note

The processor retention in WFI and WFE low-power state is not available in revisions prior to r3p0.

When a processor is in WFI or WFE low-power state, the clocks to that processor are stopped. The clocks might start for short periods of time during the WFI or WFE low-power state to allow the processor to handle snoops or other short events without leaving WFI or WFE low-power state.

Whenever the clocks to a processor are stopped, it is possible for an external power controller to lower the voltage of that processor to a lower level where all state is retained while leakage is reduced. However, this is only possible if the external power controller can guarantee that the clocks do not restart without first allowing the voltage to be raised to the level required for normal operation. The Cortex-A15 MPCore processor has an interface that allows an external power controller to place one or more processors into a retention state. The external power controller ensures that the clocks do not restart without first allowing the power controller to exit the retention state.

When the clocks are stopped because the processor is in WFI or WFE low-power state, the **QACTIVE** signal is deasserted LOW for that processor as Figure 2-14 on page 2-25 shows. This indicates that retention is possible for that processor. The external power controller can place that processor in retention state by asserting **QREQn** LOW. The Cortex-A15 MPCore processor accepts the retention request by asserting **QACCEPTn** LOW. While **QREQn** and **QACCEPTn** are both asserted LOW, the processor is in quiescent state and the clocks to that processor remain stopped as long as **QREQn** remains asserted LOW. The external power controller can safely lower the voltage of that processor to a retention level.

If the individual processor cannot safely enter quiescent state when **QREQn** is asserted LOW, the Cortex-A15 MPCore processor asserts **QDENY** HIGH instead of asserting **QACCEPTn** LOW. When this occurs, the external power controller cannot lower the voltage of that processor. The external power controller must then deassert **QREQn** HIGH, after which the processor deasserts **QDENY** LOW.

If the clocks for a quiescent processor must be restarted, for example, to handle a snoop or to exit the WFI or WFE low-power state, the **QACTIVE** signal asserts HIGH. This indicates to the external power controller that it must allow that processor to exit quiescent state. To do this, the external power controller must first restore that processor power domain to a stable running voltage. The external power controller then deasserts **QREQn**. When this signal is deasserted, the Cortex-A15 MPCore processor can restart the clocks to that individual processor and then deasserts **QACCEPTn**.

Figure 2-14 shows a typical sequence where the external power controller places the processor in retention state. The processor executes a `WFI` instruction. **STANDBYWFI** is asserted and **QACTIVE** is deasserted. The external power controller asserts **QREQn** LOW to indicate that it wants to put that processor into retention. While the processor is still in WFI low-power state and the clocks are stopped, **QACCEPTn** is asserted LOW. At this point, the processor is in quiescent state and the external power controller can lower the voltage. During retention, a snoop must access the cache of the quiescent processor. The **QACTIVE** signal is asserted HIGH to request an exit from retention. The external power controller raises the voltage of that processor to running levels and deasserts **QREQn**. The clocks are started and the snoop can proceed. **QACCEPTn** is then deasserted HIGH. After the snoop is complete, **QACTIVE** is deasserted LOW. **QREQn** and **QACCEPTn** are then asserted LOW. The processor has re-entered quiescent state and the external power controller can lower the voltage once again. When the processor is ready to exit WFI low-power state, **QACTIVE** is asserted HIGH. **QREQn** is then deasserted HIGH, the processor exits WFI low-power state, and **QACCEPTn** is deasserted HIGH.

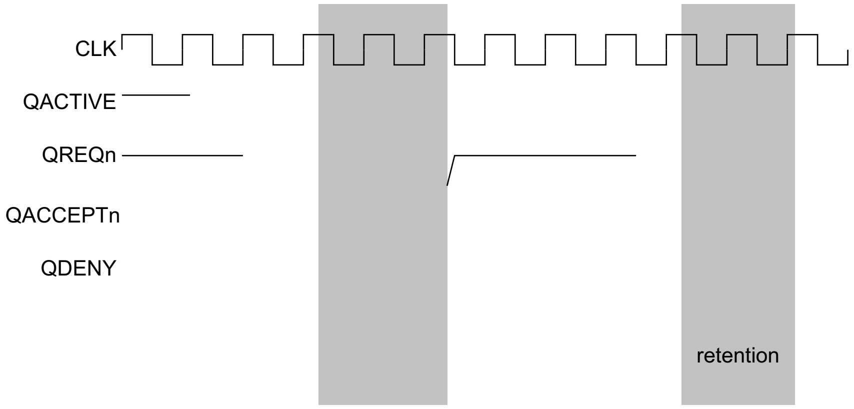


Figure 2-14 WFI successful retention timing

Figure 2-15 on page 2-26 shows a sequence where the external power controller attempts to but fails to take a processor to retention voltage levels. The processor enters WFI low-power state and deasserts **QACTIVE**. The external power controller asserts **QREQn** LOW and **QDENY** is asserted HIGH, followed by the deassertions of **QREQn** and **QDENY**. This sequence is then repeated.

Note

Figure 2-15 shows that in the second denial sequence, **QACTIVE** is still deasserted when the retention request is denied. **QACTIVE** is an indication of processor activity, but is only a hint. A retention request can be accepted when **QACTIVE** is asserted HIGH, or denied when **QACTIVE** is deasserted.

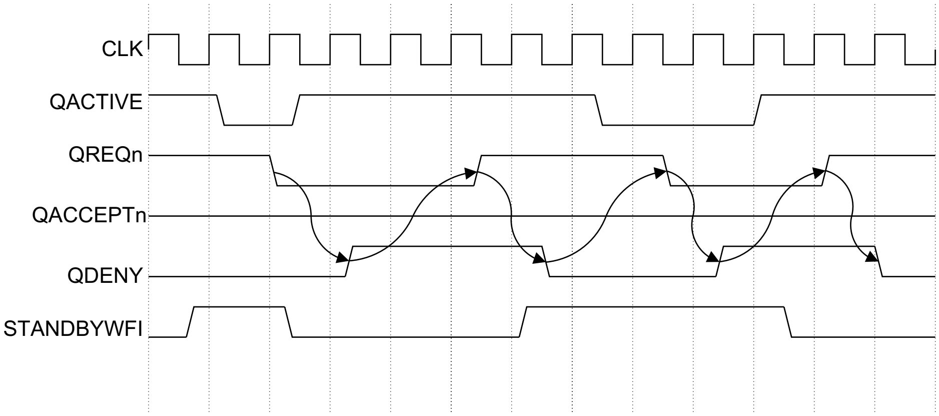


Figure 2-15 WFI denied retention timing

Each processor present has an independent set of pins **QREQn**, **QACCEPTn**, **QDENY**, and **QACTIVE**. **QREQn** is a fully asynchronous input and can only transition from deasserted HIGH to asserted LOW if **QACCEPTn** is deasserted HIGH and **QDENY** is deasserted LOW. After **QREQn** is asserted LOW, it must be held asserted LOW until either **QACCEPTn** is asserted LOW or **QDENY** is asserted HIGH. **QREQn** can then be deasserted HIGH, and must be held deasserted HIGH until both **QACCEPTn** is deasserted HIGH and **QDENY** is deasserted LOW.

The **QACTIVE** pin is an activity hint that is not part of the strict 4-phase handshake of the **QREQn**, **QACCEPTn**, and **QDENY** pins. If **QACTIVE** is asserted HIGH while a processor is in quiescent state, it indicates that the external power controller can, but is not required to, exit quiescent state. If **QACTIVE** is asserted HIGH while the processor is not in quiescent state, it indicates that the processor is not in WFI standby or WFE low-power state and there is not a likely processor retention opportunity. If **QACTIVE** is deasserted, it indicates that the processor is in WFI or WFE low-power state and there might be an opportunity to put the processor into quiescent state and lower its voltage.

Performance impact on the use of processor retention

The use of processor retention can impact performance. Any delay in deasserting **QREQn** when **QACTIVE** is asserted HIGH during quiescent state can delay the starting of the clocks to that processor. This directly impacts the performance of that processor if it exits WFI or WFE low-power state. This also impacts the performance of other processors or coherent memory masters if that processor is being restarted to handle a cache snoop. In general, only use this feature if the system can enter and exit the retention voltage level in a very short period of time.

Guidelines on the use of processor retention

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

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

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

- During processor reset, **QREQn** must be deasserted HIGH while **QACCEPTn** is asserted LOW.
- Each **QREQn** request is followed by the assertion of either **QACCEPTn** or **QDENY**, but not both. **QACCEPTn** cannot be asserted LOW at the same time as **QDENY** is asserted HIGH.
- **QACTIVE**, **QACCEPTn** and **QDENY** are synchronous outputs.
- **QREQn** is an asynchronous input.
- In a system that does not use the processor retention feature, **QREQn** must be tied HIGH.
- **QACTIVE** is only a hint. The external power controller can assert **QREQn** LOW at any time after reset. If **QACTIVE** is asserted HIGH, it is likely that the retention request can be denied.
- The Enable CPU WFI retention bit in the L2 Auxiliary Control Register, see *L2 Auxiliary Control Register* on page 4-100, must be set to enable this feature. If it is not set, all assertions of **QREQn** HIGH receive **QDENY** responses.
- If any processor sets the Force NEON/VFP clock enable active or Force main clock enable active bits in the Auxiliary Control Register, see *Auxiliary Control Register* on page 4-57, this feature is disabled and all assertions of **QREQn** receive **QDENY** responses.

NEON and VFP clock gating

The Cortex-A15 MPCore processor supports dynamic high-level clock gating of the NEON and VFP unit to reduce dynamic power dissipation.

With the NEON and VFP unit powered up, the clock to the unit is enabled when an Advanced SIMD or VFP instruction is detected in the pipeline, and is disabled otherwise.

You can set bit[29] of the Auxiliary Control Register, ACTLR, to 1 to disable dynamic clock gating of the NEON and VFP unit. See *Auxiliary Control Register* on page 4-57.

L2 control and tag banks clock gating

The Cortex-A15 MPCore processor supports dynamic high-level clock gating of the shared L2 control logic and the four L2 tag banks to reduce dynamic power dissipation.

The L2 tag bank clocks are only enabled when a corresponding access is detected in the pipeline.

The L2 control logic is disabled after 256 consecutive idle cycles. It is then enabled when an L2 access is detected, with an additional 4-cycle penalty for the wake up before the access is serviced.

You can set bit[28] of the L2 Auxiliary Control Register, L2ACTLR, to 1 to disable dynamic clock gating of the L2 tag banks. See *L2 Auxiliary Control Register* on page 4-100.

You can set bit[27] of the L2 Auxiliary Control Register, L2ACTLR, to 1 to disable dynamic clock gating of the L2 control logic. See *L2 Auxiliary Control Register* on page 4-100.

Regional clock gating

Note

This feature is not available in revisions prior to r3p0.

In addition to extensive local clock gating to register flops, you can configure the Cortex-A15 MPCore processor to include regional clock gates that can perform additional clock gating of logic blocks such as the register banks. This can potentially reduce dynamic power dissipation.

You can set bit[31] of the ACTLR2 to 1 to enable regional clock gating for each processor. See *Auxiliary Control Register 2* on page 4-105.

You can set bit[26] of the L2ACTLR to 1 to enable regional clock gating in the L2, Interrupt Controller, and Timer. See *L2 Auxiliary Control Register* on page 4-100.

At reset, both of these bits are clear and the regional clock gates enables are tied HIGH. You must set these bits to 1 to enable additional clock gating in the regional clock gates for potentially reduced dynamic power dissipation.

2.4.2 Power domains

The processor can support multiple power domains. Each processor supports the following power domains:

- The processor, including the Debug, PTM, and all of the L1 cache and branch prediction RAMs but excluding the NEON and VFP logic.
- The NEON and VFP logic.

For the remaining logic, the following power domains are supported:

- The L2 cache tag bank RAMs.
- The L2 control, the GIC and the Generic Timer logic.
- The Debug APB interface, CTI, and CTM logic.

Note

The design does not support a separate power domain for the L1 cache and branch prediction RAMs within the processor. It does not support L1 cache retention when the processor is powered down.

Figure 2-16 on page 2-29 shows the supported power domains and placeholders where you can insert clamps for a single processor in the multiprocessor.

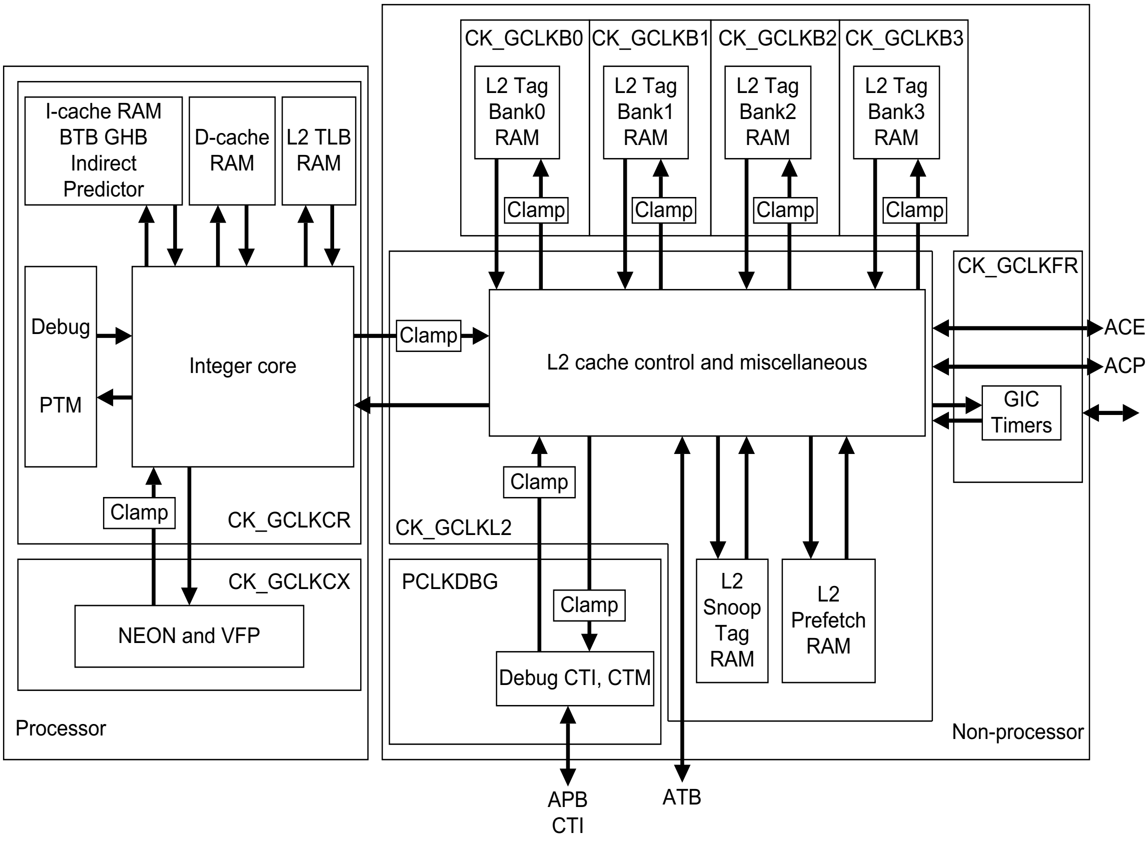


Figure 2-16 Power domains

2.4.3 Power modes

You can control the power domains independently to give different combinations of powerup and powerdown domains. However, only some powerup and powerdown domain combinations are valid and supported.

Table 2-3 shows the valid powerup and powerdown domain combinations for the different possible modes of operation.

Table 2-3 Valid power modes

Mode	Processor[a] (CLK)	NEON and VFP (CLK)	Debug-APB, CTI, and CTM (PCLKDBG)	L2 RAMs[b] (CLK)	L2 control, IC, Timer (CLK)
Full Run mode	Powered up	Powered up	Powered up	Powered up	Powered up
Run mode with Debug powered down	Powered up	Powered up	Powered down	Powered up	Powered up
Run mode with NEON and VFP powered down	Powered up	Powered down	Powered up	Powered up	Powered up
Run mode with NEON, VFP and Debug powered down	Powered up	Powered down	Powered down	Powered up	Powered up
L2 and Debug powered up	Powered down	Powered down	Powered up	Powered up	Powered up

Table 2-3 Valid power modes (continued)

Mode	Processor[a] (CLK)	NEON and VFP (CLK)	Debug-APB, CTI, and CTM (PCLKDBG)	L2 RAMs[b] (CLK)	L2 control, IC, Timer (CLK)
L2 RAMs retain with Debug powered up	Powered down	Powered down	Powered up	Retention state	Powered down
Debug powered up	Powered down	Powered down	Powered up	Powered down	Powered down
L2 powered up	Powered down	Powered down	Powered down	Powered up	Powered up
L2 RAMs retain or dormant mode	Powered down	Powered down	Powered down	Retention state	Powered down
Shutdown	Powered down	Powered down	Powered down	Powered down	Powered down

[a] Processor logic, including Debug, PTM, breakpoint and watchpoint logic, but excluding NEON and VFP.

[b] L2 cache tag bank RAMs that include tag, dirty, data, and data ECC RAMs if ECC support is present.

There are specific requirements that you must meet to power up and power down each power domain within the processor. Not adhering to these requirements can lead to UNPREDICTABLE results.

The powerup and powerdown sequences in the following sections are the only power sequences that ARM recommends. Any deviation from these sequences can lead to UNPREDICTABLE results.

The supported powerup and powerdown sequences are:

- *Processor power domain*.
- *NEON and VFP power domain* on page 2-32.
- *Debug power domain* on page 2-34.
- *External debug over power down* on page 2-34.
- *Dormant mode* on page 2-35.
- *Processor powerdown mode* on page 2-36.

Note

The powerup and powerdown sequences require that you isolate the powerup domain before power is removed from the powerdown domain. You must clamp the outputs of the powerdown domain to benign values to prevent data corruption or UNPREDICTABLE behavior in the powerup domain. The SoC controls the clamping using the **nISOLATExx** input pins.

Processor power domain

If a processor is not required, you can reduce leakage power by turning off the power to the processor. The processor refers to all processor logic, including Debug, PTM, breakpoint and watchpoint logic, but excluding the NEON and VFP unit. Powering down the processor requires that you also power down the NEON and VFP unit.

To enable the processor to be powered down, the implementation must place the processor and NEON and VFP unit on separately controlled power supplies. In addition, you must clamp the outputs of the processor and the NEON and VFP unit to benign values while the entire processor is powered down, to indicate that the processor is idle.

To power down the processor and the NEON and VFP power domains, apply the following sequence:

1. Clear the SCTLR.C bit, or HSCTLR.C bit if in Hyp mode, to prevent additional data cache allocation.
2. Clean and invalidate all data from the L1 data cache. The L2 duplicate snoop tag RAM for this processor is now empty. This prevents any new data cache snoops or data cache maintenance operations from other processors in the multiprocessor being issued to this processor.
3. Switch the processor from *Symmetric Multiprocessing* (SMP) mode to *Asymmetric Multiprocessing* (AMP) mode by clearing the ACTLR SMP bit. Clearing the SMP bit enables the processor to be taken out of coherency by preventing the processor from receiving cache, TLB, or BTB maintenance operations broadcast by other processors in the multiprocessor.
4. Ensure that the system does not send interrupts to the processor that is being powered down.
5. Execute an `ISB` instruction to ensure that all of the CP15 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 processor in the multiprocessor before the SMP bit was cleared have completed.
7. Execute a `WFI` instruction and wait until the **STANDBYWFI** output is asserted to indicate that the processor is in idle and low power state.
8. Activate the processor and the NEON and VFP output clamps by asserting the **nISOLATECPU** and **nISOLATECX** inputs LOW.
9. Remove power from the processor and the NEON and VFP power domains.

To power up the processor and the NEON and VFP power domains, apply the following sequence:

1. Assert **nCPUPORESET**.
2. Apply power to the processor and the NEON and VFP power domains while keeping **nCPUPORESET** asserted.
3. When power is restored, continue to hold **nCPUPORESET** for 16 **CLK** cycles to allow the reset to propagate.
4. Release the processor and the NEON and VFP output clamps by deasserting **nISOLATECPU** and **nISOLATECX**.
5. Continue a normal powerup reset sequence.

To power up the processor while keeping the NEON and VFP unit powered down, apply the following sequence:

1. Assert **nCPUPORESET**.
2. Apply power to the processor power domain while keeping **nCPUPORESET** asserted. Be sure to keep the NEON and VFP domain powered down.
3. When power is restored, continue to hold **nCPUPORESET** for 16 **CLK** cycles to allow the reset to propagate.

4. Release the processor output clamps by deasserting **nISOLATECPU**. Be sure to keep **nISOLATECX** asserted.
5. Continue a normal powerup reset sequence while **nISOLATECX** remain asserted.

NEON and VFP power domain

If the NEON and VFP unit is not required, you can reduce leakage power by turning off the power to the unit. While the unit is powered down, any Advanced SIMD or VFP instructions executed take the Undefined Instruction exception.

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

To power down the NEON and VFP power domain while the processor is in reset, apply the following sequence:

1. Assert **nCPUPORESET** for powerup reset or **nCORERESET** for soft reset.
2. Activate the NEON and VFP unit output clamps by asserting the **nISOLATECX** input LOW.
3. Remove power from the NEON and VFP power domain. If the processor is executing a powerup reset sequence or is first powering up, keep the NEON and VFP power domain off while applying power to the other power domains.
4. Complete and exit the reset sequence.

Note

If the NEON and VFP output clamps are released without following one of the specified NEON and VFP powerup sequences, the results are UNPREDICTABLE.

To power down the NEON and VFP power domain while the processor is not in reset, apply the following sequence:

1. You must disable access to the NEON and VFP unit by setting the CPACR and HCPTR. See *Coprocessor Access Control Register* on page 4-62 and *Hyp Coprocessor Trap Register* on page 4-71. All outstanding Advanced SIMD and VFP instructions retire and all subsequent Advanced SIMD and VFP instructions cause an Undefined Instruction exception.

```
MRC P15, 0, <Rd>, c1, c0, 2;    Read CPACR
BIC <Rd>, <Rd>, #0x00F00000;    Clear CP10 and CP11 bits
MCR p15, 0, <Rd>, c1, c0, 2;    Write CPACR
MRC p15, 4, <Rd>, c1, c1, 2;    Read HCPTR
ORR <Rd>, <Rd>, #0x00000C00;    Set TCP10 and TCP11 bits
MCR p15, 4, <Rd>, c1, c1, 2;    Write HCPTR
```

2. Execute an `ISB` instruction to ensure that all of the CP15 register changes in step 1 have been committed.
3. Software must signal to the external SoC that the NEON and VFP unit is disabled.
4. Activate the NEON and VFP output clamps by asserting the **nISOLATECX** input LOW.
5. Remove power from the NEON and VFP power domain.

Note

If the NEON and VFP output clamps are released without following one of the specified NEON and VFP powerup sequences, the results are UNPREDICTABLE.

To power up the NEON and VFP power domain while the processor is in reset, apply the following sequence:

1. Assert **nCPUPORESET** for powerup reset or **nCORERESET** for soft reset.
2. Apply power to the NEON and VFP power domain while keeping **nCPUPORESET** or **nCORERESET** asserted.
3. Release the NEON and VFP output clamps by deasserting **nISOLATECX**.
4. Complete and exit the reset sequence.
5. Software must poll the external SoC to determine that it is safe to enable the NEON and VFP unit.

After the completion of the reset sequence, you can enable the NEON and VFP unit by setting CPACR and HCPTR appropriately. See *Coprocessor Access Control Register* on page 4-62 and *Hyp Coprocessor Trap Register* on page 4-71.

To power up the NEON and VFP power domain while the processor is not in reset, apply the following sequence:

1. You must disable access to the NEON and VFP unit by setting the CPACR and HCPTR. This is a safety precaution in case software enabled access to the NEON and VFP unit while the unit was powered down.

   ```
   MRC p15, 0, <Rd>, c1, c0, 2;   Read CPACR
   BIC <Rd>, <Rd>, #0x00F00000;   Clear cp10 and cp11 bits
   MCR p15, 0, <Rd>, c1, c0, 2;   Write CPACR
   MRC p15, 4, <Rd>, c1, c1, 2;   Read HCPTR
   ORR <Rd>, <Rd>, #0x00000C00;   Set TCP10 and TCP11 bits
   MCR p15, 4, <Rd>, c1, c1, 2;   Write HCPTR
   ```

2. Execute an `ISB` instruction to ensure that all of the CP15 register changes in step 1 have been committed.
3. Software must signal to the external SoC that it is safe to power up the NEON and VFP unit.
4. Assert **nCXRESET**.
5. Apply power to the NEON and VFP power domain while keeping **nCXRESET** asserted.
6. Deassert **nCXRESET**. Keep **nCXRESET** asserted for at least 16 **CLK** cycles.
7. Release the NEON and VFP output clamps by deasserting **nISOLATECX**.
8. Software must poll the external SoC to determine that it is safe to enable the NEON and VFP unit.

After the completion of the reset sequence, you can enable the NEON and VFP unit by setting the CPACR and HCPTR appropriately.

Debug power domain

If the Cortex-A15 MPCore processor is running in an environment where debug facilities are not required for any of the processors in the multiprocessor, you can reduce leakage power by turning off the power to the debug unit in the **PCLKDBG** domain.

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

To power down the Debug APB **PCLKDBG**

To power up the processor power domain after external debug over powerdown support is no longer required, apply the following additional step to the processor and NEON and VFP powerup sequence, as described in *Processor power domain* on page 2-30, after releasing the processor and NEON VFP output clamps in step 4:

- Deassert **DBGPWRDWNREQ** to indicate that processor debug resources are available. There is no requirement for the SoC to wait for the deassertion of **DBGPWRDWNACK**.

Dormant mode

The Cortex-A15 MPCore processor supports Dormant mode, where all the processors, Debug **PCLKDBG**, 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 dirty RAMs.
- L2 data RAMs.
- L2 data ECC RAMs, if ECC support is present.

To support Dormant mode, the L2 cache RAMs must be implemented in separate power domains. In addition, you must clamp all inputs to the L2 cache RAMs to benign values, to avoid corrupting data when the processors and L2 control power domains enter and exit power down state.

Before entering Dormant mode, the architectural state of the Cortex-A15 MPCore processor, excluding the contents of the L2 cache RAMs that remain powered up, must be saved to external memory.

To exit from Dormant mode to Run mode, the SoC must perform a full powerup reset sequence. The SoC must assert the reset signals until power is restored. After power is restored, the Cortex-A15 MPCore processor exits the powerup reset sequence, then the architectural state must be restored.

To enter Dormant mode, apply the following sequence:

1. Clear the SCTLR C bit to prevent additional data cache allocation.
2. Clean and invalidate all data from the L1 data cache. The L2 duplicate snoop tag RAM for this processor is now empty. This prevents any new data cache snoops or data cache maintenance operations from other processors in the multiprocessor being issued to this processor.
3. Switch the processor from SMP mode to AMP mode by clearing the ACTLR SMP bit. Clearing the SMP bit enables the processor to be taken out of coherency by preventing the processor from receiving cache, TLB, or BTB maintenance operations broadcast by other processors in the multiprocessor.
4. Ensure that the system does not send interrupts to the processor that is being powered down.
5. Set the DBGOSDLR.DLK, OS Double Lock control bit, to force the debug interfaces to be quiescent.
6. 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.

Note

After a processor reset, cacheable memory is not accessible until the MMU is enabled. Any architectural state to be restored before the MMU is enabled must therefore be stored in non-cacheable memory.

7. Execute an `ISB` instruction to ensure that all of the CP15 register changes from the previous steps have been committed.

8. Execute a `DSB` instruction to ensure that all cache, TLB and branch predictor maintenance operations issued by any processor in the multiprocessor before the SMP bit was cleared have completed. In addition, this ensures that all state saving has completed.

9. Execute a `WFI` instruction and wait until the **STANDBYWFI** output is asserted, to indicate that the processor is in idle and low power state.

10. Repeat the previous steps for all processors, and wait for all **STANDBYWFI** outputs to be asserted.

11. The SoC asserts the input pin **ACINACTM** to idle the AXI master interface after all responses are received and before it sends any new transactions on the interface. The SoC asserts the input pin **AINACTS** to idle the ACP slave interface after all responses are received and before it sends any new transactions on the interface. 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.

    If the SoC asserts **ACVALIDM** while **ACINACTM** is asserted, then **ACINACTM** must be deasserted for the request to be accepted. If the SoC asserts **ARVALIDS**, **AWVALIDS**, or **WVALIDS** while **AINACTS** is asserted, then **AINACTS** must be deasserted for the request to be accepted.

12. When all processors **STANDBYWFI** and **STANDBYWFIL2** are asserted, the Cortex-A15 MPCore processor is ready to enter Dormant mode.

13. Activate the L2 cache RAM input clamps by asserting the **nISOLATEL2MISC** inputs LOW.

To power down the Cortex-A15 MPCore processor, apply the following sequence:

1. Ensure all non-lead processors are in shutdown mode, see *Processor power domain* on page 2-30.
2. For the remaining and lead processor, follow steps 1 and 2 in *Processor power domain* on page 2-30.
3. Ensure the ACP master does not send further requests to the individual processor. Assert **AINACTS** to idle the ACP slave interface after all responses are received.
4. Ensure any system snooping to the Cortex-A15 MPCore processor is disabled.
5. Disable L2 prefetches by writing `0x00000400` to the L2 Prefetch Control Register. See *L2 Prefetch Control Register* on page 4-103.
6. Execute an `ISB` instruction to ensure the L2 Prefetch Control Register write is complete. See *L2 Prefetch Control Register* on page 4-103.
7. Execute a `DSB` instruction to ensure completion of any prior prefetch requests.
8. Clean and invalidate all data from the L2 data cache.
9. Assert **ACINACTM** to idle the AXI master interface after all responses are received.
10. Ensure system interrupts to the Cortex-A15 MPCore processor are disabled.
11. Set the DBGOSDLR.DLK, OS Double Lock control bit, that forces the debug interfaces to be quiescent.
12. Follow steps 3 to 9 in *Processor power domain* on page 2-30.
13. Wait until the **STANDBYWFIL2** output is asserted to indicate that the L2 memory system is idle.
14. Activate the output clamps of the Cortex-A15 MPCore processor in the SoC.
15. Remove power from the L2 control and L2 RAM power domains.

To power up the Cortex-A15 MPCore processor, apply the following sequence:

1. For each processor in the multiprocessor, assert **nCPUPORESET** LOW.
2. For the lead processor in the multiprocessor, assert **nPRESETDBG** and **nL2RESET** LOW, and hold **L2RSTDISABLE** LOW.
3. Apply power to the processor, L2 control, L2 RAM and debug power domains while keeping the signals described in steps 1 and 2 LOW.
4. Release the output clamps of the Cortex-A15 MPCore processor in the SoC.
5. Continue a normal powerup reset sequence.

2.4.4 Event communication using WFE and SEV instructions

In the Cortex-A15 MPCore processor, an external agent can participate in the WFE and SEV event communication using the **EVENTI** pin. When this pin is asserted, it sends an event message to all the processors in the multiprocessor. This is similar to executing an `SEV` instruction on one processor in the multiprocessor. This enables the external agent to signal to the processor that it has released a semaphore and that the processor can leave the WFE standby power saving mode. The **EVENTI** input pin must remain HIGH for at least one **CLK** cycle to be visible by the processors.

The external agent can determine that at least one of the processors in the multiprocessor has executed an SEV instruction by checking the **EVENTO** pin. When any of the processors in the multiprocessor executes an SEV instruction, an event is signalled to all the processors in the multiprocessor, and the **EVENTO** pin is asserted. This pin is asserted HIGH for three **CLK** cycles when any of the processors executes an SEV instruction.

Chapter 3
Programmers Model

This chapter describes the processor registers and provides information for programming the processor. It contains the following sections:

3.1 About the programmers model

The Cortex-A15 MPCore processor implements the ARMv7-A architecture. This includes the:

- 32-bit ARM instruction set.
- Thumb instruction set that has both 16-bit and 32-bit instructions.
- ThumbEE instruction set.
- Jazelle Extension.
- Advanced SIMD and VFP Extensions.
- Security Extensions.
- Virtualization Extensions.
- Large Physical Address Extension.
- Multiprocessing Extensions.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

3.2 Execution environment support

For the Cortex-A15 MPCore processor, ARM deprecates any use of the ThumbEE instruction set and provides a trivial implementation of the Jazelle Extension.

This means that:

- Jazelle state is not supported, that is, the processor does not accelerate the execution of any Java bytecodes.
- The `BXJ` instruction behaves as a `BX` instruction.
- The processor supports ThumbEE state only to support legacy code that uses ThumbEE instructions.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

Table 3-1 shows the Jazelle register instruction summary and the response to the instructions.

Table 3-1 Jazelle register instruction summary

Register	Instruction	Response
Jazelle ID (JIDR)[a]	`MRC p14, 7, <Rd>, c0, c0, 0`	Reads as zero[b]
	`MCR p14, 7, <Rd>, c0, c0,0`	A write causes an undefined exception regardless of processor mode
Jazelle Main Configuration (JMCR)[c]	`MRC p14, 7, <Rd>, c2, c0, 0`	Reads as zero[d]
	`MCR p14, 7, <Rd>, c2, c0, 0`	Ignore writes
Jazelle OS Control (JOSCR)[e]	`MRC p14, 7, <Rd>, c1, c0, 0`	Reads as zero
	`MCR p14, 7, <Rd>, c1, c0, 0`	Ignore writes

[a] Accessible from all privilege levels.

[b] Can cause a trap to Hyp mode depending on the TIDO bit set in the HCR.

[c] Write-only accessible in unprivileged level, read-write accessible at PL1 or higher.

[d] Can cause a trap to Hyp mode depending on the TJDBX bit set in the HSTR.

[e] Accessible from PL1 or higher.

Note

Because no hardware acceleration is present in the processor when the `BXJ` instruction is used, the `BX` instruction is invoked.

3.3 Advanced SIMD and VFP Extensions

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

VFP extension performs single-precision and double-precision floating-point operations.

Note

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

All Advanced SIMD instructions and VFP instructions are available in both ARM and Thumb states.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

See Chapter 14 *NEON and VFP Unit* for implementation-specific information.

3.4 Security Extensions architecture

The Security Extensions architecture facilitate the development of secure applications. This section describes the following:

- *System boot sequence*.
- *Security Extensions configuration write access disable*.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

3.4.1 System boot sequence

Caution

The Security Extensions enable the development of a more secure software environment. The technology does not protect the processor from hardware attacks, and you must make sure that the hardware containing the boot code is appropriately secure.

The processor always boots in the privileged Supervisor mode in the Secure state, with SCR.NS set to 0. See *Secure Configuration Register* on page 4-63. This means that code that does not attempt to use the Security Extensions always runs in the Secure state. If the software uses both Secure and Non-secure states, the less trusted software, such as a complex operating system and application code running under that operating system, executes in Non-secure state, and the most trusted software executes in the Secure state.

The following sequence is expected to be typical use of the Security Extensions:

1. Exit from reset in Secure state.
2. Configure the security state of memory and peripherals. Some memory and peripherals are accessible only to the software running in Secure state.
3. Initialize the secure operating system. The required operations depend on the operating system, and include initialization of caches, MMU, exception vectors, and stacks.
4. Initialize Secure Monitor software to handle exceptions that switch execution between the Secure and Non-secure operating systems.
5. Optionally lock aspects of the secure state environment to additional configuration.
6. Pass control through the Secure Monitor software to the Non-secure OS with an `SMC` instruction.
7. Enable the Non-secure operating system to initialize. The required operations depend on the operating system, and typically include initialization of caches, MMU, exception vectors, and stacks.

The overall security of the software depends on the system design, and on the secure software itself.

3.4.2 Security Extensions configuration write access disable

The processor pin **CP15SDISABLE** disables write access to certain registers in the CP15 System Control Coprocessor. There is one **CP15SDISABLE** input for each processor. Attempts to write to these registers when **CP15SDISABLE** is HIGH result in an Undefined Instruction exception. Reads from the registers are still permitted.

You can use the **CP15SDISABLE** pin to disable subsequent write access to the system control processor registers after the Secure boot code runs. This protects the configuration set up by the Secure boot code.

A change to the **CP15SDISABLE** pin takes effect on the instructions decoded by the processor as quickly as possible. Software must perform an `ISB` instruction after a change to this pin on the boundary of the macrocell has occurred, to ensure that its effect is recognized for following instructions. It it is expected that:

- Control of the **CP15SDISABLE** pin remains within the SoC that embodies the macrocell.
- The **CP15SDISABLE** pin is driven LOW by the SoC hardware at reset.

See *Registers affected by CP15SDISABLE* on page 4-2 for a list of the system registers affected by CP15SDISABLE.

3.5 Virtualization Extensions architecture

The Virtualization Extensions are an extension to the ARMv7 architecture profile that provides a set of hardware features that support virtualizing the Non-secure state of an ARM VMSAv7 implementation. This supports system use of a virtual machine monitor, known as the hypervisor, that is responsible for switching Guest operating systems.

The Virtualization Extensions require implementation of the Security Extensions and the *Large Physical Address Extension* (LPAE).

The Virtualization Extensions also require implementation of:

- the v7.1 Debug architecture, see Chapter 10 *Debug*
- the PMUv2 Performance Monitors, see Chapter 11 *Performance Monitor Unit*.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

3.6 Large Physical Address Extension architecture

The *Large Physical Address Extension* (LPAE) is an extension to the ARMv7-A architecture profile that provides an address translation system supporting physical addresses of up to 40 bits at a fine grain of translation.

The LPAE requires implementation of the Multiprocessing Extensions.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

3.7 Multiprocessing Extensions

The Multiprocessing Extensions are an extension to the ARMv7-A profile, that provides a set of features that enhance multiprocessing functionality.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

3.8 Modes of operation and execution

This section describes the instruction set states and modes of the Cortex-A15 MPCore processor in:

- *Operating states*.

3.8.1 Operating states

The processor has the following instruction set states controlled by the T bit and J bit in the CPSR.

ARM state The processor executes 32-bit, word-aligned ARM instructions.

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

ThumbEE state The processor executes a variant of the Thumb instruction set designed as a target for dynamically generated code. This is code compiled on the device either shortly before or during execution from a portable bytecode or other intermediate or native representation.

The J bit and the T bit determine the instruction set used by the processor. Table 3-2 shows the encoding of these bits.

Table 3-2 CPSR J and T bit encoding

J	T	Instruction set state
0	0	ARM
0	1	Thumb
1	1	ThumbEE

Note

- The processor does not support Jazelle state. This means there is no processor state where the J bit is 1 and T bit is 0.
- Transition between ARM and Thumb states does not affect the processor mode or the register contents. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for information on entering and exiting ThumbEE state.

3.9 Memory model

The Cortex-A15 MPCore processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word.

The processor can store words in memory as either:

- big-endian format
- little-endian format.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information about big-endian and little-endian memory systems.

Note

Instructions are always treated as little-endian.

Chapter 4
System Control

This chapter describes the system control registers, their structure, operation, and how to use them. It contains the following sections:

4.1 About system control

The system control coprocessor, CP15, controls and provides status information for the functions implemented in the processor. The main functions of the system control coprocessor are:

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

4.1.1 Registers affected by CP15SDISABLE

The processor pin **CP15SDISABLE** disables write access to certain registers in the CP15 system control coprocessor. See *Security Extensions configuration write access disable* on page 3-5 for more information.

The Cortex-A15 MPCore processor does not have any implementation-defined registers that are affected by **CP15SDISABLE**.

For a list of registers affected by **CP15SDISABLE**, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

4.2 Register summary

This section gives a summary of the CP15 system control registers. For more information on using the CP15 system control registers, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

The system control coprocessor is a set of registers that you can write to and read from. Some of the registers permit more than one type of operation.

The following subsections describe the CP15 system control registers grouped by CRn order, and accessed by the MCR and MRC instructions in the order of CRn, Op1, CRm, Op2:

- *c0 registers* on page 4-4.
- *c1 registers* on page 4-5.
- *c2 registers* on page 4-6.
- *c3 registers* on page 4-6.
- *c5 registers* on page 4-6.
- *c6 registers* on page 4-7.
- *c7 registers* on page 4-7.
- *c8 registers* on page 4-9.
- *c9 registers* on page 4-10.
- *c10 registers* on page 4-11.
- *c12 registers* on page 4-12.
- *c13 registers* on page 4-12.
- *c14 registers* on page 4-12.
- *c15 registers* on page 4-13.

The Cortex-A15 MPCore processor supports the *Virtualization Extensions* (VE), the *Large Physical Address Extension* (LPAE), and the Generic Timer. See *Virtualization Extensions architecture* on page 3-7, *Large Physical Address Extension architecture* on page 3-8, and Chapter 9 *Generic Timer* for more information. The VE, LPAE, and Generic Timer contain a number of 64-bit registers. The following subsection describes these registers and provides cross references to individual register descriptions:

- *64-bit registers* on page 4-13.

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

- *Identification registers* on page 4-14.
- *Virtual memory control registers* on page 4-15.
- *PL1 Fault handling registers* on page 4-16.
- *Other system control registers* on page 4-17.
- *Cache maintenance operations* on page 4-17.
- *TLB maintenance operations* on page 4-18.
- *Address translation operations* on page 4-19.
- *Miscellaneous operations* on page 4-20.
- *Performance monitor registers* on page 4-21.
- *Security Extensions registers* on page 4-22.
- *Virtualization Extensions registers* on page 4-23.
- *Hyp mode TLB maintenance operations* on page 4-24.
- *Generic Timer registers* on page 4-25.
- *Implementation defined registers* on page 4-25.

Table 4-1 describes the column headings that the CP15 register summary tables use throughout this chapter.

Table 4-1 Column headings definition for CP15 register summary tables

Column name	Description
CRn	Register number within the system control coprocessor
Op1	Opcode_1 value for the register
CRm	Operational register number within CRn
Op2	Opcode_2 value for the register
Name	Short form architectural, operation, or code name for the register
Reset	Reset value of register
Description	Cross-reference to register description

4.2.1 c0 registers

Table 4-2 shows the 32-bit wide CP15 system control registers when CRn is c0.

Table 4-2 c0 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	MIDR	`0x414FC0F0`	*Main ID Register* on page 4-27
		1	CTR	`0x8444C004`[a]	*Cache Type Register* on page 4-28
		2	TCMTR	`0x00000000`	*TCM Type Register* on page 4-29
		3	TLBTR	`0x00000000`	*TLB Type Register* on page 4-29
		4, 7	MIDR	`0x414FC0F0`	Aliases of Main ID Register, *Main ID Register* on page 4-27
		5	MPIDR	-[b]	*Multiprocessor Affinity Register* on page 4-29
		6	REVIDR	`0x00000000`	*Revision ID Register* on page 4-30
	c1	0	ID_PFR0	`0x00001131`	*Processor Feature Register 0* on page 4-31
		1	ID_PFR1	`0x00011011`	*Processor Feature Register 1* on page 4-32
		2	ID_DFR0	`0x02010555`	*Debug Feature Register 0* on page 4-33
		3	ID_AFR0	`0x00000000`	*Auxiliary Feature Register 0* on page 4-34
		4	ID_MMFR0	`0x10201105`	*Memory Model Feature Register 0* on page 4-34
		5	ID_MMFR1	`0x20000000`	*Memory Model Feature Register 1* on page 4-36
		6	ID_MMFR2	`0x01240000`	*Memory Model Feature Register 2* on page 4-37
		7	ID_MMFR3	`0x02102211`	*Memory Model Feature Register 3* on page 4-38
	c2	0	ID_ISAR0	`0x02101110`	*Instruction Set Attribute Register 0* on page 4-40
		1	ID_ISAR1	`0x13112111`	*Instruction Set Attribute Register 1* on page 4-41
		2	ID_ISAR2	`0x21232041`	*Instruction Set Attribute Register 2* on page 4-43

Table 4-2 c0 register summary (continued)

Op1	CRm	Op2	Name	Reset	Description
		3	ID_ISAR3	`0x11112131`	*Instruction Set Attribute Register 3* on page 4-44
		4	ID_ISAR4	`0x10011142`	*Instruction Set Attribute Register 4* on page 4-46
		5	ID_ISAR5	`0x00000000`	*Instruction Set Attribute Register 5* on page 4-48
1	c0	0	CCSIDR	UNK	*Cache Size ID Register* on page 4-48
		1	CLIDR	`0x0A200023`	*Cache Level ID Register* on page 4-50
		7	AIDR	`0x00000000`	*Auxiliary ID Register* on page 4-51
2	c0	0	CSSELR	UNK	*Cache Size Selection Register* on page 4-51
4	c0	0	VPIDR	-[c]	*Virtualization Processor ID Register* on page 4-52
		5	VMPIDR	-[d]	*Virtualization Multiprocessor ID Register* on page 4-53

a. The reset value depends on the primary input, **IMINLN**. The value shown in Table 4-2 on page 4-4 assumes **IMINLN** is set to 1.

b. The reset value depends on the primary input, **CLUSTERID**, and the number of configured processors in the MPCore device.

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

d. The reset value is the value of the Multiprocessor Affinity Register.

4.2.2 c1 registers

Table 4-3 shows the 32-bit wide CP15 system control registers when CRn is c1.

Table 4-3 c1 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	SCTLR	`0x00C50078`[a]	*System Control Register* on page 4-54
		1	ACTLR	`0x00000000`	*Auxiliary Control Register* on page 4-57
		2	CPACR	`0x00000000`[b]	*Coprocessor Access Control Register* on page 4-62
	c1	0	SCR	`0x00000000`	*Secure Configuration Register* on page 4-63
		1	SDER	UNK	Secure Debug Enable Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	NSACR	`0x00000000`[c]	*Non-Secure Access Control Register* on page 4-65
4	c0	0	HSCTLR	UNK	*Hyp System Control Register* on page 4-67
		1	HACTLR	UNK	*Hyp Auxiliary Control Register* on page 4-69
	c1	0	HCR	`0x00000000`	Hyp Configuration Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	HDCR	`0x00000006`[d]	*Hyp Debug Configuration Register* on page 4-69
		2	HCPTR	`0x000033FF`[e]	*Hyp Coprocessor Trap Register* on page 4-71
		3	HSTR	`0x00000000`	Hyp System Trap Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		7	HACR	UNK	*Hyp Auxiliary Configuration Register* on page 4-74

a. The reset value depends on primary inputs, **CFGTE**, **CFGEND**, and **VINITHI**. The value shown in Table 4-3 on page 4-5 assumes these signals are set to zero.

b. The reset value depends on the VFP and NEON configuration. If VFP and NEON are implemented, the reset value is `0x00000000`. If VFP is implemented but NEON is not implemented, the reset value is `0x80000000`. If VFP and NEON are not implemented, the reset value is `0x00000000`.

c. The reset value depends on the VFP and NEON configuration. If VFP and NEON are implemented, the reset value is `0x00000000`. If VFP is implemented but NEON is not implemented, the reset value is `0x00008000`. If VFP and NEON are not implemented, the reset value is `0x00000000`.

d. The reset value for bit[7] is UNK.

e. The reset value depends on the VFP and NEON configuration. If VFP and NEON are implemented, the reset value is `0x000033FF`. If VFP is implemented but NEON is not implemented, the reset value is `0x0000B3FF`. If VFP and NEON are not implemented, the reset value is `0x0000BFFF`.

4.2.3 c2 registers

Table 4-4 shows the 32-bit wide CP15 system control registers when CRn is c2.

Table 4-4 c2 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	TTBR0	UNK	*Translation Table Base Register 0 and Register 1* on page 4-74
		1	TTBR1	UNK	*Translation Table Base Register 0 and Register 1* on page 4-74
		2	TTBCR	`0x00000000`[a]	*Translation Table Base Control Register* on page 4-74
4	c0	2	HTCR	UNK	*Hyp Translation Control Register* on page 4-75
	c1	2	VTCR	UNK	Virtualization Translation Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

a. 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.

4.2.4 c3 registers

Table 4-5 shows the 32-bit wide CP15 system control registers when CRn is c3.

Table 4-5 c3 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	DACR	UNK	Domain Access Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

4.2.5 c5 registers

Table 4-6 shows the 32-bit wide CP15 system control registers when CRn is c5.

Table 4-6 c5 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	DFSR	UNK	*Data Fault Status Register* on page 4-75
		1	IFSR	UNK	*Instruction Fault Status Register* on page 4-78
	c1	0	ADFSR	UNK	*Auxiliary Data Fault Status Register* on page 4-81
		1	AIFSR	UNK	*Auxiliary Instruction Fault Status Register* on page 4-82

Table 4-6 c5 register summary (continued)

Op1	CRm	Op2	Name	Reset	Description
4	c1	0	HADFSR	UNK	*Hyp Auxiliary Data Fault Syndrome Register* on page 4-82
		1	HAIFSR	UNK	*Hyp Auxiliary Instruction Fault Syndrome Register* on page 4-83
	c2	0	HSR	UNK	*Hyp Syndrome Register* on page 4-83

4.2.6 c6 registers

Table 4-7 shows the 32-bit wide CP15 system control registers when CRn is c6.

Table 4-7 c6 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	DFAR	UNK	Data Fault Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	IFAR	UNK	Instruction Fault Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
4	c0	0	HDFAR	UNK	Hyp Data Fault Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	HIFAR	UNK	Hyp Instruction Fault Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		4	HPFAR	UNK	Hyp IPA Fault Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

4.2.7 c7 registers

Table 4-8 shows the 32-bit wide CP15 system control registers when CRn is c7.

Table 4-8 c7 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	4	NOP	UNK	No Operation, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c1	0	ICIALLUIS	UNK	Invalidate all instruction caches to PoU[a] Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		6	BPIALLIS	UNK	Invalidate all branch predictors Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c4	0	PAR	UNK	*Physical Address Register* on page 4-85
	c5	0	ICIALLU	UNK	Invalidate all instruction caches to PoU, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	ICIMVAU	UNK	Invalidate instruction caches by MVA to PoU, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		4	CP15ISB	UNK	Instruction Synchronization Barrier operation, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-8 c7 register summary (continued)

Op1	CRm	Op2	Name	Reset	Description
		6	BPIALL	UNK	Invalidate all branch predictors, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		7	BPIMVA	UNK	Invalidate MVA from branch predictors, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c6	1	DCIMVAC	UNK	Invalidate data cache line by MVA to PoC[b], see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	DCISW	UNK	Invalidate data cache line by set/way, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c8	0	ATS1CPR	UNK	Stage 1 current state PL1 read, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	ATS1CPW	UNK	Stage 1 current state PL1 write, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	ATS1CUR	UNK	Stage 1 current state unprivileged read, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		3	ATS1CUW	UNK	Stage 1 current state unprivileged write, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		4	ATS12NSOPR	UNK	Stages 1 and 2 Non-secure PL1 read, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		5	ATS12NSOPW	UNK	Stages 1 and 2 Non-secure PL1 write, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		6	ATS12NSOUR	UNK	Stages 1 and 2 Non-secure unprivileged read, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		7	ATS12NSOUW	UNK	Stages 1 and 2 Non-secure unprivileged write, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c10	1	DCCMVAC	UNK	Clean data cache line by MVA to PoC, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	DCCSW	UNK	Clean data cache line by set/way, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		4	CP15DSB	UNK	Data Synchronization Barrier operation, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		5	CP15DMB	UNK	Data Memory Barrier operation, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c11	1	DCCMVAU	UNK	Clean data cache line by MVA to PoU, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c13	1	NOP	UNK	No Operation, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c14	1	DCCIMVAC	UNK	Clean and invalidate data cache line by MVA to PoC, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-8 c7 register summary (continued)

Op1	CRm	Op2	Name	Reset	Description
		2	DCCISW	UNK	Clean and invalidate data cache line by set/way, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
4	c8	0	ATS1HR	UNK	Stage 1 Hyp mode read, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	ATS1HW	UNK	Stage 1 Hyp mode write, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

a. PoU = Point of Unification. If **BROADCASTINNER** is LOW, the PoU is in the L1 data cache. If **BROADCASTINNER** is HIGH then the PoU is outside of the processor and is dependent on the external memory system.

b. PoC = Point of Coherence. The PoC is always outside of the processor and is dependent on the external memory system.

4.2.8 c8 registers

Table 4-9 shows the 32-bit wide CP15 system control registers when CRn is c8.

Table 4-9 c8 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c3	0	TLBIALLIS	UNK	Invalidate entire TLB Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	TLBIMVAIS	UNK	Invalidate unified TLB entry by MVA and ASID Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	TLBIASIDIS	UNK	Invalidate unified TLB by ASID match Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		3	TLBIMVAAIS	UNK	Invalidate unified TLB entry by MVA all ASID Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c5	0	ITLBIALL	UNK	Invalidate instruction TLB, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	ITLBIMVA	UNK	Invalidate instruction TLB entry by MVA and ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	ITLBIASID	UNK	Invalidate instruction TLB by ASID match, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c6	0	DTLBIALL	UNK	Invalidate data TLB, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	DTLBIMVA	UNK	Invalidate data TLB entry by MVA and ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	DTLBIASID	UNK	Invalidate data TLB by ASID match, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c7	0	TLBIALL	UNK	Invalidate unified TLB, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	TLBIMVA	UNK	Invalidate unified TLB by MVA and ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-9 c8 register summary (continued)

Op1	CRm	Op2	Name	Reset	Description
		2	TLBIASID	UNK	Invalidate unified TLB by ASID match, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		3	TLBIMVAA	UNK	Invalidate unified TLB entries by MVA all ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
4	c3	0	TLBIALLHIS	UNK	Invalidate entire Hyp unified TLB Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	TLBIMVAHIS	UNK	Invalidate Hyp unified TLB entry by MVA Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		4	TLBIALLNSNHIS	UNK	Invalidate entire Non-secure non-Hyp unified TLB Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c7	0	TLBIALLH	UNK	Invalidate entire Hyp unified TLB, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	TLBIMVAH	UNK	Invalidate Hyp unified TLB entry by MVA, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		4	TLBIALLNSNH	UNK	Invalidate entire Non-secure non-Hyp unified TLB, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

4.2.9 c9 registers

Table 4-10 shows the 32-bit wide CP15 system control registers when CRn is c9.

Table 4-10 c9 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c12	0	PMCR	`0x410F3000`	*Performance Monitor Control Register* on page 11-8
		1	PMNCNTENSET	UNK	Performance Monitor Count Enable Set Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	PMNCNTENCLR	UNK	Performance Monitor Count Enable Clear Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		3	PMOVSR	UNK	Performance Monitor Overflow Flag Status Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		4	PMSWINC	UNK	Performance Monitor Software Increment Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		5	PMSELR	UNK	Performance Monitor Event Counter Selection Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		6	PMCEID0	`0x3FFF0F3F`	*Performance Monitor Common Event Identification Register 0* on page 11-10
		7	PMCEID1	`0x00000000`	*Performance Monitor Common Event Identification Register 1* on page 11-12
	c13	0	PMCCNTR	UNK	Performance Monitor Cycle Count Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-10 c9 register summary (continued)

Op1	CRm	Op2	Name	Reset	Description
		1	PMXEVTYPER	UNK	Performance Monitor Event Type Select Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	PMXEVCNTR	UNK	Performance Monitor Event Count Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c14	0	PMUSERENR	`0x00000000`	Performance Monitor User Enable Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	PMINTENSET	UNK	Performance Monitor Interrupt Enable Set Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	PMINTENCLR	UNK	Performance Monitor Interrupt Enable Clear Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		3	PMOVSSET	UNK	Performance Monitor Overflow Flag Status Set Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
1	c0	2	L2CTLR	`0x00000000`[a]	*L2 Control Register* on page 4-85
		3	L2ECTLR	`0x00000000`	*L2 Extended Control Register* on page 4-87

a. The reset value depends on the processor configuration.

4.2.10 c10 registers

Table 4-11 shows the 32-bit wide CP15 system control registers when CRn is c10.

Table 4-11 c10 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c2	0	PRRR[a]	`0x00098AA4`	Primary Region Remap Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		0	MAIR0[b]	`0x00098AA4`	*Memory Attribute Indirection Register 0* on page 4-88
		1	NMRR[c]	`0x44E048E0`	Normal Memory Remap Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	MAIR1[d]	`0x44E048E0`	*Memory Attribute Indirection Register 1* on page 4-89
	c3	0	AMAIR0	UNK	*Auxiliary Memory Attribute Indirection Register 0* on page 4-89
		1	AMAIR1	UNK	*Auxiliary Memory Attribute Indirection Register 1* on page 4-89
4	c2	0	HMAIR0	UNK	Hyp Memory Attribute Indirection Register 0, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	HMAIR1	UNK	Hyp Memory Attribute Indirection Register 1, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c3	0	HAMAIR0	UNK	*Hyp Auxiliary Memory Attribute Indirection Register 0* on page 4-89
		1	HAMAIR1	UNK	*Hyp Auxiliary Memory Attribute Indirection Register 1* on page 4-89

a. The processor behavior for all implementation-defined encodings in the PRRR register is UNPREDICTABLE.

b. The processor behavior for all implementation-defined encodings in the MAIR0 register is UNPREDICTABLE.

c. The processor behavior for all implementation-defined encodings in the NMRR register is UNPREDICTABLE.

d. The processor behavior for all implementation-defined encodings in the MAIR1 register is UNPREDICTABLE.

4.2.11 c12 registers

Table 4-12 shows the 32-bit wide CP15 system control registers when CRn is c12.

Table 4-12 c12 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	VBAR	`0x00000000`[a]	Vector Base Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		1	MVBAR	UNK	Monitor Vector Base Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
	c1	0	ISR	UNK	Interrupt Status Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
4	c0	0	HVBAR	UNK	Hyp Vector Base Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

a. The reset value is `0x00000000` for the Secure copy of the register. You must program the Non-secure copy of the register with the required initial value, as part of the processor boot sequence.

4.2.12 c13 registers

Table 4-13 shows the 32-bit wide CP15 system control registers when CRn is c13.

Table 4-13 c13 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	FCSEIDR	`0x00000000`	*FCSE Process ID Register* on page 4-89
		1	CONTEXTIDR	UNK	Context ID Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		2	TPIDRURW	UNK	User Read/Write Thread ID Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		3	TPIDRURO	UNK	User Read-Only Thread ID Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
		4	TPIDRPRW	UNK	PL1 only Thread ID Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
4	c0	2	HTPIDR	UNK	Hyp Software Thread ID Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

4.2.13 c14 registers

See Chapter 9 *Generic Timer* for information on the Generic Timer registers.

4.2.14 c15 registers

Table 4-14 shows the 32-bit wide CP15 system control registers when CRn is c15.

Table 4-14 c15 register summary

Op1	CRm	Op2	Name	Reset	Description
0	c0	0	IL1Data0	UNK	*Instruction L1 Data n Register* on page 4-89
		1	IL1Data1	UNK	*Instruction L1 Data n Register* on page 4-89
		2	IL1Data2	UNK	*Instruction L1 Data n Register* on page 4-89
	c1	0	DL1Data0	UNK	*Data L1 Data n Register* on page 4-90
		1	DL1Data1	UNK	*Data L1 Data n Register* on page 4-90
		2	DL1Data2	UNK	*Data L1 Data n Register* on page 4-90
		3	DL1Data3	UNK	*Data L1 Data n Register* on page 4-90
	c4	0	RAMINDEX	UNK	*RAM Index Register* on page 4-91
1	c0	0	L2ACTLR	`0x00000000`	*L2 Auxiliary Control Register* on page 4-100
		3	L2PFR	`0x000009B0`	*L2 Prefetch Control Register* on page 4-103
		4	ACTLR2	`0x00000000`	*Auxiliary Control Register 2* on page 4-105
4	c0	0	CBAR	-[a]	*Configuration Base Address Register* on page 4-106

a. The reset value depends on the primary input, **PERIPHBASE[39:15]**.

4.2.15 64-bit registers

Table 4-15 gives a summary of the 64-bit wide CP15 system control registers, accessed by the `MCCR` and `MRRC` instructions.

Table 4-15 64-bit register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
-	0	c2	-	TTBR0	UNK	Translation Table Base Register 0, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	1	c2	-	TTBR1	UNK	Translation Table Base Register 1, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	4	c2	-	HTTBR	UNK	Hyp Translation Table Base Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	6	c2	-	VTTBR	UNK[a]	Virtualization Translation Table Base Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	0	c7	-	PAR	UNK	*Physical Address Register* on page 4-85
-	0	c14	-	CNTPCT	UNK	Physical Count Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-15 64-bit register summary (continued)

CRn	Op1	CRm	Op2	Name	Reset	Description
-	1	c14	-	CNTVCT	UNK	Virtual Count Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	2	c14	-	CNTP_CVAL	UNK	PL1 Physical Timer CompareValue Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	3	c14	-	CNTV_CVAL	UNK	Virtual Timer CompareValue Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	4	c14	-	CNTVOFF	UNK	Virtual Offset Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	6	c14	-	CNTHP_CVAL	UNK	PL2 Physical Timer CompareValue Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	0	c15	-	CPUMERRSR	-[b]	*CPU Memory Error Syndrome Register* on page 4-107
-	1	c15	-	L2MERRSR	-[b]	*L2 Memory Error Syndrome Register* on page 4-109

a. The reset value for bits[55:48] is b00000000.

b. The reset value for bits[63,47:40,39:32,31] are all b0.

4.2.16 Identification registers

Table 4-16 shows the 32-bit wide Identification registers.

Table 4-16 Identification registers

Name	CRn	Op1	CRm	Op2	Reset	Description
MIDR	c0	0	c0	0	`0x414FC0F0`	*Main ID Register* on page 4-27
CTR				1	`0x8444C004`[a]	*Cache Type Register* on page 4-28
TCMTR				2	`0x00000000`	*TCM Type Register* on page 4-29
TLBTR				3	`0x00000000`	*TLB Type Register* on page 4-29
MPIDR				5	-[b]	*Multiprocessor Affinity Register* on page 4-29
REVIDR				6	`0x00000000`	*Revision ID Register* on page 4-30
MIDR				4, 7	`0x414FC0F0`	Aliases of Main ID Register, *Main ID Register* on page 4-27
ID_PFR0			c1	0	`0x00001131`	*Processor Feature Register 0* on page 4-31
ID_PFR1				1	`0x00011011`	*Processor Feature Register 1* on page 4-32
ID_DFR0				2	`0x02010555`	*Debug Feature Register 0* on page 4-33
ID_AFR0				3	`0x00000000`	*Auxiliary Feature Register 0* on page 4-34
ID_MMFR0				4	`0x10201105`	*Memory Model Feature Register 0* on page 4-34
ID_MMFR1				5	`0x20000000`	*Memory Model Feature Register 1* on page 4-36

Table 4-16 Identification registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
ID_MMFR2				6	`0x01240000`	*Memory Model Feature Register 2* on page 4-37
ID_MMFR3				7	`0x02102211`	*Memory Model Feature Register 3* on page 4-38
ID_ISAR0			c2	0	`0x02101110`	*Instruction Set Attribute Register 0* on page 4-40
ID_ISAR1				1	`0x13112111`	*Instruction Set Attribute Register 1* on page 4-41
ID_ISAR2				2	`0x21232041`	*Instruction Set Attribute Register 2* on page 4-43
ID_ISAR3				3	`0x11112131`	*Instruction Set Attribute Register 3* on page 4-44
ID_ISAR4				4	`0x10011142`	*Instruction Set Attribute Register 4* on page 4-46
ID_ISAR5				5	`0x00000000`	*Instruction Set Attribute Register 5* on page 4-48
CCSIDR		1	c0	0	UNK	*Cache Size ID Register* on page 4-48
CLIDR				1	`0x0A200023`	*Cache Level ID Register* on page 4-50
AIDR				7	`0x00000000`	*Auxiliary ID Register* on page 4-51
CSSELR		2	c0	0	UNK	*Cache Size Selection Register* on page 4-51

a. The reset value depends on the primary input, **IMINLN**. The value shown in Table 4-16 on page 4-14 assumes **IMINLN** is set to 1.

b. The reset value depends on the primary input, **CLUSTERID**, and the number of configured processors in the MPCore device.

4.2.17 Virtual memory control registers

Table 4-17 shows the Virtual memory control registers.

Table 4-17 Virtual memory registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
SCTLR	c1	0	c0	0	`0x00C50078`[a]	32-bit	*System Control Register* on page 4-54
TTBR0	c2	0	c0	0	UNK	32-bit	*Translation Table Base Register 0 and Register 1* on page 4-74
	-	0	c2	-		64-bit	*Translation Table Base Register 0 and Register 1* on page 4-74
TTBR1	c2	0	c0	1	UNK	32-bit	*Translation Table Base Register 0 and Register 1* on page 4-74
	-	1	c2	-		64-bit	*Translation Table Base Register 0 and Register 1* on page 4-74
TTBCR	c2	0	c0	2	`0x00000000`[b]	32-bit	*Translation Table Base Control Register* on page 4-74
DACR	c3	0	c0	0	UNK	32-bit	Domain Access Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PRRR	c10	0	c2	0	`0x00098AA4`	32-bit	Primary Region Remap Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-17 Virtual memory registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
MAIR0				0	UNK	32-bit	Memory Attribute Indirection Register 0, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
NMRR				1	`0x44E048E0`	32-bit	Normal Memory Remap Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
MAIR1				1	UNK	32-bit	Memory Attribute Indirection Register 1, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
AMAIR0			c3	0	UNK	32-bit	*Auxiliary Memory Attribute Indirection Register 0* on page 4-89
AMAIR1				1	UNK	32-bit	*Auxiliary Memory Attribute Indirection Register 1* on page 4-89
CONTEXTIDR	c13	0	c0	1	UNK	32-bit	Process ID Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

a. The reset value depends on primary inputs, **CFGTE**, **CFGEND**, and **VINITHI**. The value shown in Table 4-17 on page 4-15 assumes these signals are set to zero.

b. 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.

4.2.18 PL1 Fault handling registers

Table 4-18 shows the 32-bit wide PL1 Fault handling registers.

Table 4-18 PL1 Fault handling registers

Name	CRn	Op1	CRm	Op2	Reset	Description
DFSR	c5	0	c0	0	UNK	*Data Fault Status Register* on page 4-75
IFSR				1	UNK	*Instruction Fault Status Register* on page 4-78
ADFSR			c1	0	UNK	*Auxiliary Data Fault Status Register* on page 4-81
AIFSR				1	UNK	*Auxiliary Instruction Fault Status Register* on page 4-82
DFAR	c6	0	c0	0	UNK	Data Fault Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
IFAR				2	UNK	*ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

The Virtualization Extensions include additional fault handling registers. For more information see *Virtualization Extensions registers* on page 4-23.

4.2.19 Other system control registers

Table 4-19 shows the other system control registers.

Table 4-19 Other system control registers

Name	CRn	Op1	CRm	Op2	Reset	Description
CPACR	c1	0	c0	2	`0x00000000`[a]	*Coprocessor Access Control Register* on page 4-62
FCSEIDR	c13	0	c0	0	`0x00000000`	*FCSE Process ID Register* on page 4-89

[a] The reset value depends on the VFP and NEON configuration. If VFP and NEON are implemented, the reset value is `0x00000000`. If VFP is implemented but NEON is not implemented, the reset value is `0x80000000`. If VFP and NEON are not implemented, the reset value is `0x00000000`.

4.2.20 Cache maintenance operations

Table 4-20 shows the 32-bit wide cache and branch predictor maintenance operations.

Table 4-20 Cache and branch predictor maintenance operations

Name	CRn	Op1	CRm	Op2	Reset	Description
ICIALLUIS	c7	0	c1	0	UNK	Instruction cache invalidate all to PoU[a] Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
BPIALLIS				6	UNK	Branch predictor invalidate all Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ICIALLU			c5	0	UNK	Instruction cache invalidate all to PoU, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ICIMVAU				1	UNK	Instruction cache invalidate by MVA to PoU, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
BPIALL				6	UNK	Branch predictor invalidate all, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
BPIMVA				7	UNK	Branch predictor invalidate by MVA, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
DCIMVAC			c6	1	UNK	Data cache invalidate by MVA to PoC[b], see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
DCISW				2	UNK	Data cache invalidate by set/way, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
DCCMVAC			c10	1	UNK	Data cache clean by MVA to PoC, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
DCCSW				2	UNK	Data cache clean by set/way, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-20 Cache and branch predictor maintenance operations (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
DCCMVAU			c11	1	UNK	Data cache clean by MVA to PoU, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
DCCIMVAC			c14	1	UNK	Data cache clean and invalidate by MVA to PoC, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
DCCISW				2	UNK	Data cache clean and invalidate by set/way, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

a. PoU = Point of Unification. If **BROADCASTINNER** is LOW, the PoU is in the L1 data cache. If **BROADCASTINNER** is HIGH, the PoU is outside of the processor and is dependent on the external memory system.

b. PoC = Point of Coherence. The PoC is always outside of the processor and is dependent on the external memory system.

4.2.21 TLB maintenance operations

Table 4-21 shows the 32-bit wide TLB maintenance operations.

Table 4-21 TLB maintenance operations

Name	CRn	Op1	CRm	Op2	Reset	Description
TLBIALLIS	c8	0	c3	0	UNK	Invalidate entire unified TLB Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TLBIMVAIS				1	UNK	Invalidate unified TLB by MVA and ASID Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TLBIASIDIS				2	UNK	Invalidate unified TLB by ASID Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TLBIMVAAIS				3	UNK	Invalidate unified TLB by MVA all ASID Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ITLBIALL			c5	0	UNK	Invalidate entire instruction TLB, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ITLBIMVA				1	UNK	Invalidate instruction TLB entry by MVA and ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ITLBIASID				2	UNK	Invalidate instruction TLB by ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
DTLBIALL			c6	0	UNK	Invalidate entire data TLB, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
DTLBIMVA				1	UNK	Invalidate data TLB entry by MVA and ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
DTLBIASID				2	UNK	Invalidate data TLB by ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TLBIALL			c7	0	UNK	Invalidate entire unified TLB, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-21 TLB maintenance operations (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
TLBIMVA				1	UNK	Invalidate unified TLB by MVA and ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TLBIASID				2	UNK	Invalidate unified TLB by ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TLBIMVAA				3	UNK	Invalidate unified TLB by MVA all ASID, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

The Virtualization Extensions include additional TLB operations for use in Hyp mode. For more information, see *Hyp mode TLB maintenance operations* on page 4-24.

4.2.22 Address translation operations

Table 4-22 shows the address translation register and operations.

Table 4-22 Address translation operations

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
PAR	c7	0	c4	0	UNK	32-bit	*Physical Address Register* on page 4-85
	-	0	c7	-		64-bit	
ATS1CPR	c7	0	c8	0	UNK	32-bit	Stage 1 current state PL1 read, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ATS1CPW				1	UNK	32-bit	Stage 1 current state PL1 write, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ATS1CUR				2	UNK	32-bit	Stage 1 current state unprivileged read, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ATS1CUW				3	UNK	32-bit	Stage 1 current state unprivileged write, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ATS12NSOPR				4	UNK	32-bit	Stages 1 and 2 Non-secure PL1 read, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ATS12NSOPW				5	UNK	32-bit	Stages 1 and 2 Non-secure PL1 write, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ATS12NSOUR				6	UNK	32-bit	Stages 1 and 2 Non-secure unprivileged read, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-22 Address translation operations (continued)

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
ATS12NSOUW				7	UNK	32-bit	Stages 1 and 2 Non-secure unprivileged write, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ATS1HR		4	c8	0	UNK	32-bit	Stage 1 Hyp mode read, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ATS1HW				1	UNK	32-bit	Stage 1 Hyp mode write, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

4.2.23 Miscellaneous operations

Table 4-23 shows the 32-bit wide miscellaneous operations.

Table 4-23 Miscellaneous system control operations

Name	CRn	Op1	CRm	Op2	Reset	Description
NOP	c7	0	c0	4	UNK	*System control No Operation* (NOP), see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CP15ISB			c5	4	UNK	Instruction Synchronization Barrier operation, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CP15DSB			c10	4	UNK	Data Synchronization Barrier operation, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CP15DMB				5	UNK	Data Memory Barrier operation, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
NOP			c13	1	UNK	*System control No Operation* (NOP), see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TPIDRURW	c13	0	c0	2	UNK	User Read/Write Thread ID Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TPIDRURO				3	UNK	User Read-Only Thread ID Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TPIDRPRW				4	UNK	PL1 only Thread ID Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
HTPIDR		4	c0	2	UNK	Hyp Software Thread ID Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

4.2.24 Performance monitor registers

Table 4-24 shows the 32-bit wide performance monitor registers.

Table 4-24 Performance monitor registers

Name	CRn	Op1	CRm	Op2	Reset	Description
PMCR	c9	0	c12	0	`0x410F3000`	*Performance Monitor Control Register* on page 11-8
PMNCNTENSET				1	UNK	Performance Monitor Count Enable Set Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PMNCNTENCLR				2	UNK	Performance Monitor Count Enable Clear Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PMOVSR				3	UNK	Performance Monitor Overflow Flag Status Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PMSWINC				4	UNK	Performance Monitor Software Increment Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PMSELR				5	UNK	Performance Monitor Event Counter Selection Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PMCEID0				6	`0x3FFF0F3F`	*Performance Monitor Common Event Identification Register 0* on page 11-10
PMCEID1				7	`0x00000000`	*Performance Monitor Common Event Identification Register 1* on page 11-12
PMCCNTR			c13	0	UNK	Performance Monitor Cycle Count Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PMXEVTYPER				1	UNK	Performance Monitor Event Type Select Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PMXEVCNTR				2	UNK	Performance Monitor Event Count Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PMUSERENR			c14	0	`0x00000000`	Performance Monitor User Enable Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-24 Performance monitor registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
PMINTENSET				1	UNK	Performance Monitor Interrupt Enable Set Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PMINTENCLR				2	UNK	Performance Monitor Interrupt Enable Clear Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
PMOVSSET				3	UNK	Performance Monitor Overflow Flag Status Set Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

4.2.25 Security Extensions registers

Table 4-25 shows the 32-bit wide Security Extensions registers.

Table 4-25 Security Extensions registers

Name	CRn	Op1	CRm	Op2	Reset	Description
SCR	c1	0	c1	0	`0x00000000`	*Secure Configuration Register* on page 4-63
SDER				1	UNK	Secure Debug Enable Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
NSACR				2	`0x00000000`[a]	*Non-Secure Access Control Register* on page 4-65
VBAR	c12	0	c0	0	`0x00000000`[b]	Vector Base Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
MVBAR				1	UNK	Monitor Vector Base Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
ISR			c1	0	UNK	Interrupt Status Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

[a] The reset value depends on the VFP and NEON configuration. If VFP and NEON are implemented, the reset value is `0x00000000`. If VFP is implemented but NEON is not implemented, the reset value is `0x00008000`. If VFP and NEON are not implemented, the reset value is `0x00000000`.

[b] The reset value is `0x00000000` for the Secure copy of the register. You must program the Non-secure copy of the register with the required initial value, as part of the processor boot sequence.

4.2.26 Virtualization Extensions registers

Table 4-26 shows the Virtualization Extensions registers.

Table 4-26 Virtualization Extensions registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
VPIDR	c0	4	c0	0	-[a]	32-bit	*Virtualization Processor ID Register* on page 4-52*l*
VMPIDR				5	-[b]	32-bit	*Virtualization Multiprocessor ID Register* on page 4-53
HSCTLR	c1	4	c0	0	UNK	32-bit	*Hyp System Control Register* on page 4-67
HACTLR				1	UNK	32-bit	*Hyp Auxiliary Control Register* on page 4-69
HCR			c1	0	`0x00000000`	32-bit	Hyp Configuration Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
HDCR				1	`0x00000006`[c]	32-bit	*Hyp Debug Configuration Register* on page 4-69
HCPTR				2	`0x000033FF`[d]	32-bit	*Hyp Coprocessor Trap Register* on page 4-71
HSTR				3	`0x00000000`	32-bit	Hyp System Trap Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
HACR				7	UNK	32-bit	*Hyp Auxiliary Control Register* on page 4-69
HTCR	c2	4	c0	2	UNK	32-bit	*Hyp Translation Control Register* on page 4-75
VTCR			c1	2	UNK	32-bit	Virtualization Translation Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
HTTBR	-	4	c2	-	UNK	64-bit	Hyp Translation Table Base Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
VTTBR	-	6	c2	-	UNK[e]	64-bit	Virtualization Translation Table Base Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
HADFSR	c5	4	c1	0	UNK	32-bit	*Hyp Auxiliary Data Fault Syndrome Register* on page 4-82
HAIFSR				1	`UNK`	32-bit	*Hyp Auxiliary Instruction Fault Syndrome Register* on page 4-83
HSR			c2	0	UNK	32-bit	*Hyp Syndrome Register* on page 4-83
HDFAR	c6	4	c0	0	UNK	32-bit	Hyp Data Fault Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
HIFAR				2	UNK	32-bit	Hyp Instruction Fault Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-26 Virtualization Extensions registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
HPFAR				4	UNK	32-bit	Hyp IPA Fault Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
HMAIR0	c10	4	c2	0	UNK	32-bit	Hyp Memory Attribute Indirection Register 0, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
HMAIR1				1	UNK	32-bit	Hyp Memory Attribute Indirection Register 1, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
HAMAIR0			c3	0	UNK	32-bit	*Hyp Auxiliary Memory Attribute Indirection Register 0* on page 4-89
HAMAIR1				1	UNK	32-bit	*Hyp Auxiliary Memory Attribute Indirection Register 1* on page 4-89
HVBAR	c12	4	c0	0	UNK	32-bit	Hyp Vector Base Address Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

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

b. The reset value is the value of the Multiprocessor Affinity Register.

c. The reset value for bit[7] is UNK.

d. The reset value depends on the VFP and NEON configuration. If VFP and NEON are implemented, the reset value is `0x000033FF`. If VFP is implemented but NEON is not implemented, the reset value is `0x0000B3FF`. If VFP and NEON are not implemented, the reset value is `0x0000BFFF`.

e. The reset value for bits[55:48] is b00000000.

4.2.27 Hyp mode TLB maintenance operations

Table 4-27 shows the 32-bit wide TLB maintenance operations added for Virtualization Extensions.

Table 4-27 Hyp mode TLB maintenance operations

Name	CRn	Op1	CRm	Op2	Reset	Description
TLBIALLHIS	c8	4	c3	0	UNK	Invalidate entire Hyp unified TLB Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TLBIMVAHIS				1	UNK	Invalidate Hyp unified TLB by MVA Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TLBIALLNSNHIS				4	UNK	Invalidate entire Non-secure Non-Hyp unified TLB Inner Shareable, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

Table 4-27 Hyp mode TLB maintenance operations (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
TLBIALLH			c7	0	UNK	Invalidate entire Hyp unified TLB, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TLBIMVAH				1	UNK	Invalidate Hyp unified TLB by MVA, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
TLBIALLNSNH				4	UNK	Invalidate entire Non-secure Non-Hyp unified TLB, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

4.2.28 Generic Timer registers

See Chapter 9 *Generic Timer* for information on the Generic Timer registers.

4.2.29 Implementation defined registers

Table 4-28 shows the IMPLEMENTATION DEFINED registers. These registers provide test features and any required configuration options specific to the Cortex-A15 MPCore processor.

Table 4-28 Implementation defined registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
ACTLR	c1	0	c0	1	`0x00000000`	32-bit	*Auxiliary Control Register* on page 4-57
L2CTLR	c9	1	c0	2	`0x00000000`[a]	32-bit	*L2 Control Register* on page 4-85
L2ECTLR				3	`0x00000000`	32-bit	*L2 Extended Control Register* on page 4-87
IL1DATA0	c15	0	c0	0	UNK	32-bit	*Instruction L1 Data n Register* on page 4-89
IL1DATA1				1	UNK	32-bit	*Instruction L1 Data n Register* on page 4-89
IL1DATA2				2	UNK	32-bit	*Instruction L1 Data n Register* on page 4-89
DL1DATA0			c1	0	UNK	32-bit	*Data L1 Data n Register* on page 4-90
DL1DATA1				1	UNK	32-bit	*Data L1 Data n Register* on page 4-90
DL1DATA2				2	UNK	32-bit	*Data L1 Data n Register* on page 4-90
DL1DATA3				3	UNK	32-bit	*Data L1 Data n Register* on page 4-90
RAMINDEX			c4	0	UNK	32-bit	*RAM Index Register* on page 4-91
L2ACTLR		1	c0	0	`0x00000000`	32-bit	*L2 Auxiliary Control Register* on page 4-100
L2PFR				3	`0x000009B0`	32-bit	*L2 Prefetch Control Register* on page 4-103
ACTLR2				4	`0x00000000`	32-bit	*Auxiliary Control Register 2* on page 4-105
CBAR		4	c0	0	-[b]	32-bit	*Configuration Base Address Register* on page 4-106
CPUMERRSR	-	0	c15	-	-[c]	64-bit	*CPU Memory Error Syndrome Register* on page 4-107
L2MERRSR	-	1	c15	-	-[c]	64-bit	*L2 Memory Error Syndrome Register* on page 4-109

a. The reset value depends on the processor configuration.

b. The reset value depends on the primary input, **PERIPHBASE[39:15]**.

c. The reset value for bits[63,47:40,39:32,31] are all zero.

4.3 Register descriptions

This section describes all the CP15 system control registers by coprocessor register number order. Table 4-2 on page 4-4 to Table 4-15 on page 4-13 provide cross references to individual registers.

4.3.1 Main ID Register

The MIDR characteristics are:

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

Usage constraints The MIDR is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-1 shows the MIDR bit assignments.

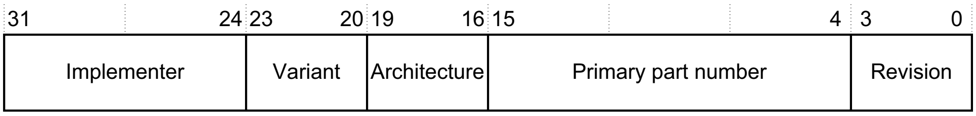


Figure 4-1 MIDR bit assignments

Table 4-29 shows the MIDR bit assignments.

Table 4-29 MIDR bit assignments

Bits	Name	Function
[31:24]	Implementer	Indicates the implementer code: `0x41` ARM Limited.
[23:20]	Variant	Indicates the variant number of the processor. This is the major revision number *n* in the *rn* part of the rnpn description of the product revision status: `0x4` Major revision number.
[19:16]	Architecture	Indicates the architecture code: `0xF` Defined by CPUID scheme.
[15:4]	Primary part number	Indicates the primary part number: `0xC0F` Cortex-A15.
[3:0]	Revision	Indicates the minor revision number of the processor. This is the minor revision number *n* in the *pn* part of the *rnpn* description of the product revision status: `0x0` Minor revision number.

To access the MIDR, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c0, 0; Read Main ID Register
```

4.3.2 Cache Type Register

The CTR characteristics are:

Purpose Provides information about the architecture of the caches.

Usage constraints The CTR is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-2 shows the CTR bit assignments.

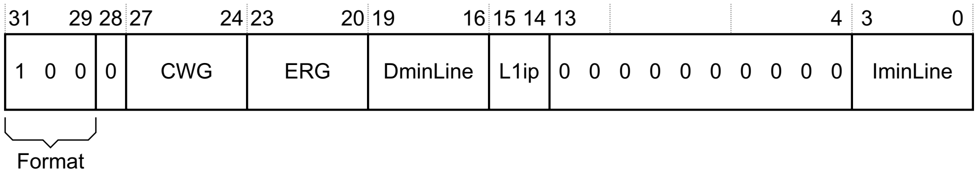


Figure 4-2 CTR bit assignments

Table 4-30 shows the CTR bit assignments.

Table 4-30 CTR bit assignments

Bits	Name	Function
[31:29]	Format	Indicates the implemented CTR format: `b100` ARMv7 format.
[28]	-	Reserved, RAZ.
[27:24]	CWG	Cache Writeback Granule. Log$_2$ of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified: `0x4` Cache writeback granule size is 16 words.
[23:20]	ERG	Exclusives Reservation Granule. Log$_2$ of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions: `0x4` Exclusive reservation granule size is 16 words.
[19:16]	DminLine	Log$_2$ of the number of words in the smallest cache line of all the data and unified caches that the processor controls: `0x4` Smallest data cache line size is 16 words.
[15:14]	L1ip	Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache: `b11` Physical index, physical tag.
[13:4]	-	Reserved, RAZ.
[3:0]	IminLine	Log$_2$ of the number of words in the smallest cache line of all the instruction caches that the processor controls. The primary input **IMINLN** defines the reset value: `0x3` Smallest instruction cache line size is 8 words, the **IMINLN** signal is LOW. `0x4` Smallest instruction cache line size is 16 words, the **IMINLN** signal is HIGH.

To access the CTR, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c0, 1; Read Cache Type Register
```

4.3.3 TCM Type Register

The processor does not implement instruction or data *Tightly Coupled Memory* (TCM), so this register is always RAZ/WI.

4.3.4 TLB Type Register

The TLBTR characteristics are:

Purpose Provides information about the TLB implementation.

Usage constraints The TLBTR is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-3 shows the TLBTR bit assignments.

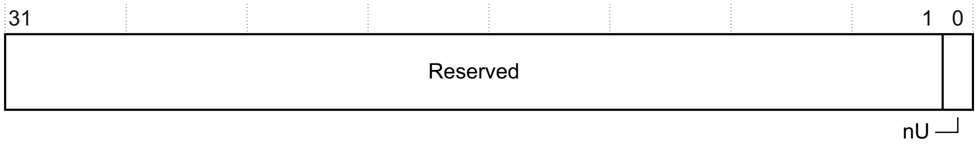


Figure 4-3 TLBTR bit assignments

Table 4-31 shows the TLBTR bit assignments.

Table 4-31 TLBTR bit assignments

Bits	Name	Function
[31:1]	-	Reserved, RAZ.
[0]	nU	Not Unified. Indicates whether the implementation has a unified TLB: `0x0` Processor has a unified TLB.

To access the TLBTR, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c0, 3; Read TLB Type Register
```

4.3.5 Multiprocessor Affinity Register

The MPIDR characteristics are:

Purpose Provides an additional processor identification mechanism for scheduling purposes in a multiprocessor system.

Usage constraints The MPIDR is:

- A read-only register.
- Common to the Secure and Non-secure states.

- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-4 shows the MPIDR bit assignments.

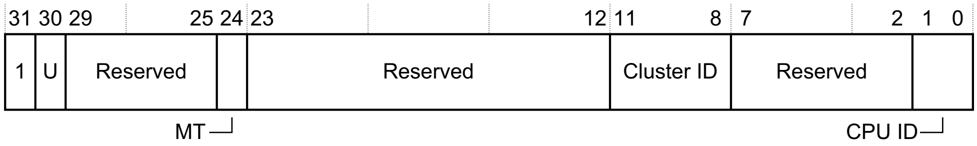


Figure 4-4 MPIDR bit assignments

Table 4-32 shows the MPIDR bit assignments.

Table 4-32 MPIDR bit assignments

Bits	Name	Function
[31]	-	RAO. Indicates that the processor implements the Multiprocessing Extensions register format.
[30]	U	Indicates a Uniprocessor system, as distinct from processor 0 in a multiprocessor system: **0** Processor is part of a multiprocessor system.
[29:25]	-	Reserved, RAZ.
[24]	MT	Indicates whether the lowest level of affinity consists of logical processors that are implemented using a hardware multi-threading type approach: **0** Performance of processors at the lowest affinity level is largely independent.
[23:12]	-	Reserved, RAZ.
[11:8]	Cluster ID	Indicates the value read in the **CLUSTERID** configuration pin. It identifies each processor in a multiprocessor configuration.
[7:2]	-	Reserved, RAZ.
[1:0]	CPU ID	Indicates the processor number in a Cortex-A15 MPCore device. For: • One processor, the CPU ID is `0x0`. • Two processors, the CPU IDs are `0x0` and `0x1`. • Three processors, the CPU IDs are `0x0`, `0x1`, and `0x2`. • Four processors, the CPU IDs are `0x0`, `0x1`, `0x2`, and `0x3`.

To access the MPIDR, read the CP15 registers with:

```
MRC p15, 0, <Rt>, c0, c0, 5; Read Multiprocessor Affinity Register
```

4.3.6 Revision ID Register

The REVIDR characteristics are:

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

Usage constraints The REVIDR is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-5 shows the REVIDR bit assignments.

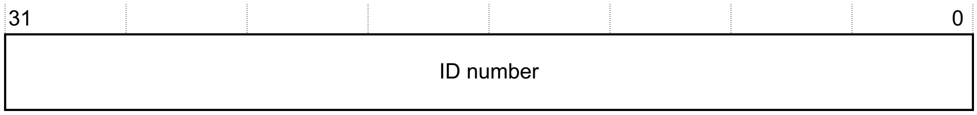


Figure 4-5 REVIDR bit assignments

Table 4-33 shows the REVIDR bit assignments.

Table 4-33 REVIDR bit assignments

Bits	Name	Function
[31:0]	ID number	Implementation-specific revision information. The reset value is determined by the specific Cortex-A15 MPCore implementation.

To access the REVIDR, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c0, 6; Read Revision ID Register
```

4.3.7 Processor Feature Register 0

The ID_PFR0 characteristics are:

Purpose Provides information about the programmers model and top-level information about the instruction sets supported by the processor.

Usage constraints The ID_PFR0 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-6 shows the ID_PFR0 bit assignments.

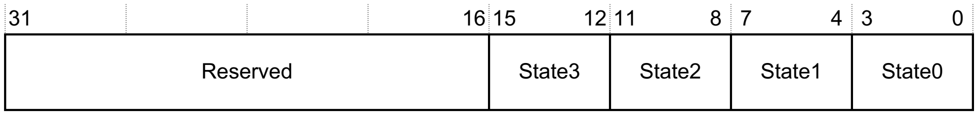


Figure 4-6 ID_PFR0 bit assignments

Table 4-34 shows the ID_PFR0 bit assignments.

Table 4-34 ID_PFR0 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RAZ.
[15:12]	State3	Indicates support for *Thumb Execution Environment* (ThumbEE) instruction set: `0x1` Processor supports ThumbEE instruction set.
[11:8]	State2	Indicates support for Jazelle extension: `0x1` Processor supports trivial implementation of Jazelle extension.
[7:4]	State1	Indicates support for Thumb instruction set: `0x3` Processor supports Thumb encoding after the introduction of Thumb-2 technology, and for all 16-bit and 32-bit Thumb basic instructions.
[3:0]	State0	Indicates support for ARM instruction set: `0x1` Processor supports ARM instruction set.

To access the ID_PFR0, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c1, 0; Read Processor Feature Register 0
```

4.3.8 Processor Feature Register 1

The ID_PFR1 characteristics are:

Purpose Provides information about the programmers model and Security Extensions support.

Usage constraints The ID_PFR1 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-7 shows the ID_PFR1 bit assignments.

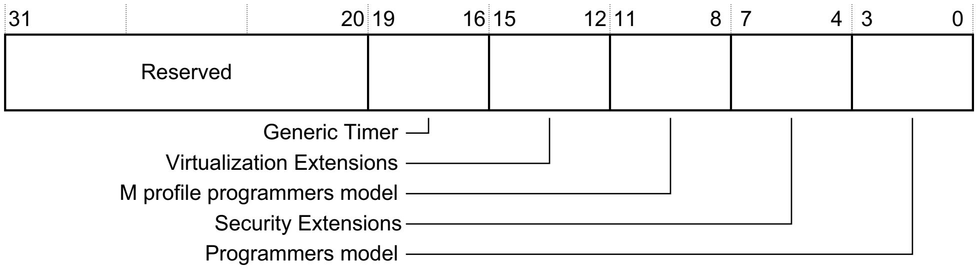


Figure 4-7 ID_PFR1 bit assignments

Table 4-35 shows the ID_PFR1 bit assignments.

Table 4-35 ID_PFR1 bit assignments

Bits	Name	Function
[31:20]	-	Reserved, RAZ.
[19:16]	Generic Timer	Indicates support for Generic Timer Extension: `0x1` Processor supports Generic Timer Extension.
[15:12]	Virtualization Extensions	Indicates support for Virtualization Extensions: `0x1` Processor supports Virtualization Extensions.
[11:8]	M profile programmers model	Indicates support for microcontroller programmers model: `0x0` Processor does not support microcontroller programmers model.
[7:4]	Security Extensions	Indicates support for Security Extensions. This includes support for Monitor mode and the `SMC` instruction: `0x1` Processor supports Security Extensions.

To access the ID_PFR1, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c1, 1 ; Read Processor Feature Register 1
```

4.3.9 Debug Feature Register 0

The ID_DFR0 characteristics are:

Purpose Provides top-level information about the debug system.

Usage constraints The ID_DFR0 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-8 shows the ID_DFR0 bit assignments.

Figure 4-8 ID_DFR0 bit assignments

Table 4-36 shows the ID_DFR0 bit assignments.

Table 4-36 ID_DFR0 bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RAZ.
[27:24]	Performance Monitors Extension	Indicates support for coprocessor-based ARM Performance Monitors Extension: `0x2` Processor supports Performance Monitors Extension, PMUv2 architecture.
[23:20]	Debug model, M profile	Indicates support for memory-mapped debug model for M profile processors: `0x0` Processor does not support M profile Debug architecture, with memory-mapped access.
[19:16]	Memory-mapped trace model	Indicates support for memory-mapped trace model: `0x1` Processor supports ARM trace architecture, with memory-mapped access.
[15:12]	Coprocessor trace model	Indicates support for coprocessor-based trace model: `0x0` Processor does not support ARM trace architecture, with CP14 access.
[11:8]	Memory-mapped debug model	Indicates support for memory-mapped debug model: `0x5` Processor supports v7.1 Debug architecture, with memory-mapped access.
[7:4]	Coprocessor Secure debug model	Indicates support for coprocessor-based Secure debug model: `0x5` Processor supports v7.1 Debug architecture, with CP14 access.
[3:0]	Coprocessor debug model	Indicates support for coprocessor-based debug model: `0x5` Processor supports v7.1 Debug architecture, with CP14 access.

To access the ID_DFR0, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c1, 2; Read Debug Feature Register 0
```

4.3.10 Auxiliary Feature Register 0

The processor does not implement ID_AFR0, so this register is always RAZ/WI.

4.3.11 Memory Model Feature Register 0

The ID_MMFR0 characteristics are:

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

Usage constraints The ID_MMFR0 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-9 on page 4-35 shows the ID_MMFR0 bit assignments.

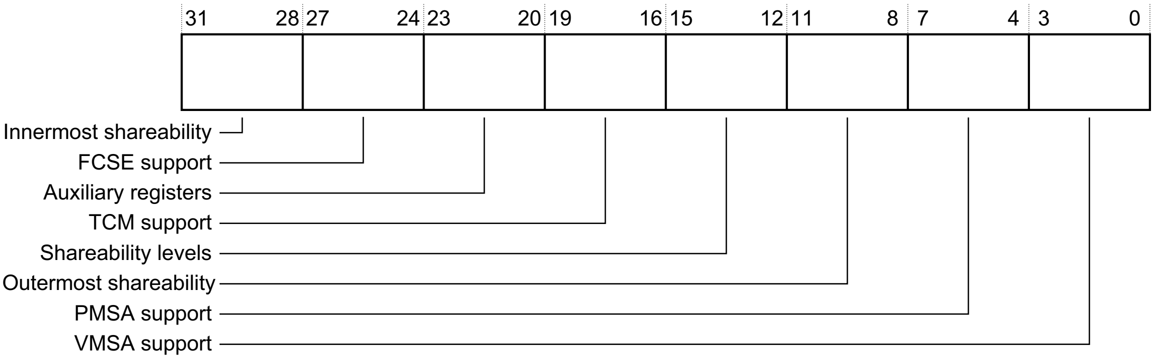


Figure 4-9 ID_MMFR0 bit assignments

Table 4-37 shows the ID_MMFR0 bit assignments.

Table 4-37 ID_MMFR0 bit assignments

Bits	Name	Function
[31:28]	Innermost shareability	Indicates the innermost shareability domain implemented: `0x1` Processor implements hardware coherency support.
[27:24]	FCSE	Indicates support for *Fast Context Switch Extension* (FCSE): `0x0` Processor does not support FCSE.
[23:20]	Auxiliary registers	Indicates support for Auxiliary registers: `0x2` Processor supports the ACTLR and ADFSR. See *Auxiliary Control Register* on page 4-57 and *Auxiliary Data Fault Status Register* on page 4-81.
[19:16]	TCM	Indicates support for TCMs and associated DMAs: `0x0` Processor does not support TCM.
[15:12]	Shareability levels	Indicates the number of shareability levels implemented: `0x1` Processor implements two levels of shareability.
[11:8]	Outermost shareability	Indicates the outermost shareability domain implemented: `0x1` Processor supports hardware coherency.
[7:4]	PMSA	Indicates support for a *Protected Memory System Architecture* (PMSA): `0x0` Processor does not support PMSA.
[3:0]	VMSA	Indicates support for a *Virtual Memory System Architecture* (VMSA). `0x5` Processor supports: • VMSAv7, with support for remapping and the Access flag. • *Privileged Execute Never* (PXN) bit in the Short-descriptor translation table format. • *Privileged Execute Never* (PXN) bit in the Long-descriptor translation table format.

To access the ID_MMFR0, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c1, 4; Read Memory Model Feature Register 0
```

4.3.12 Memory Model Feature Register 1

The ID_MMFR1 characteristics are:

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

Usage constraints The ID_MMFR1 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-10 shows the ID_MMFR1 bit assignments.

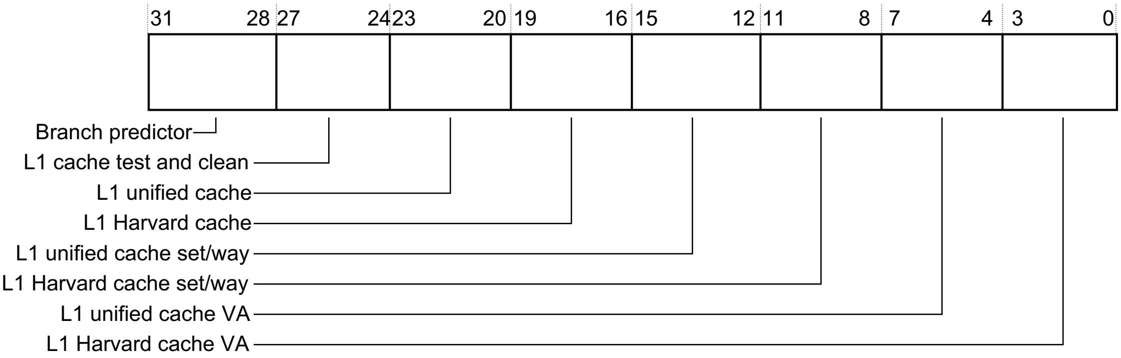


Figure 4-10 ID_MMFR1 bit assignments

Table 4-38 shows the ID_MMFR1 bit assignments.

Table 4-38 ID_MMFR1 bit assignments

Bits	Name	Function
[31:28]	Branch predictor	Indicates branch predictor management requirements. `0x2` Branch predictor requires flushing on: • Enabling or disabling the MMU. • Writing new data to instruction locations. • Writing new mappings to the translation tables. • Any change to the TTBR0, TTBR1, or TTBCR registers without a corresponding change to the FCSE ProcessID or ContextID.
[27:24]	L1 cache test and clean	Indicates the supported L1 data cache test and clean operations, for Harvard or unified cache implementation: `0x0` Not supported.
[23:20]	L1 unified cache	Indicates the supported entire L1 cache maintenance operations, for a unified cache implementation: `0x0` Not supported.
[19:16]	L1 Harvard cache	Indicates the supported entire L1 cache maintenance operations, for a Harvard cache implementation: `0x0` Not supported.

Table 4-38 ID_MMFR1 bit assignments (continued)

Bits	Name	Function
[15:12]	L1 unified cache set/way	Indicates the supported L1 cache line maintenance operations by set/way, for a unified cache implementation: `0x0` Not supported.
[11:8]	L1 Harvard cache set/way	Indicates the supported L1 cache line maintenance operations by set/way, for a Harvard cache implementation: `0x0` Not supported.
[7:4]	L1 unified cache VA	Indicates the supported L1 cache line maintenance operations by MVA, for a unified cache implementation: `0x0` Not supported.
[3:0]	L1 Harvard cache VA	Indicates the supported L1 cache line maintenance operations by MVA, for a Harvard cache implementation: `0x0` Not supported.

To access the ID_MMFR1, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c1, 5; Read Memory Model Feature Register 1
```

4.3.13 Memory Model Feature Register 2

The ID_MMFR2 characteristics are:

Purpose Provides information about the implemented memory model and memory management support of the processor.

Usage constraints The ID_MMFR2 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-11 shows the ID_MMFR2 bit assignments.

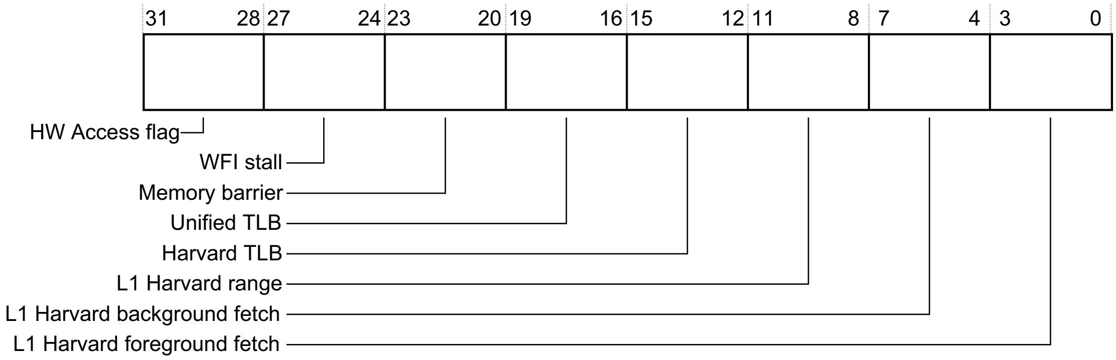


Figure 4-11 ID_MMFR2 bit assignments

Table 4-39 shows the ID_MMFR2 bit assignments.

Table 4-39 ID_MMFR2 bit assignments

Bits	Name	Function
[31:28]	HW Access flag	Indicates support for Hardware Access flag: `0x0` Not supported.
[27:24]	WFI stall	Indicates support for *Wait For Interrupt* (WFI) stalling: `0x1` Processor supports WFI stalling.
[23:20]	Memory barrier	Indicates the supported CP15 memory barrier operations. `0x2` Processor supports: • *Data Synchronization Barrier* (DSB). • *Instruction Synchronization Barrier* (ISB). • *Data Memory Barrier* (DMB).
[19:16]	Unified TLB	Indicates the supported TLB maintenance operations, for a unified TLB implementation. `0x4` Processor supports: • 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 PL1 and PL0 unified TLB. • Invalidate entire Hyp mode unified TLB.
[15:12]	Harvard TLB	Indicates the supported TLB maintenance operations, for a Harvard TLB implementation: `0x0` Not supported.
[11:8]	L1 Harvard range	Indicates the supported L1 cache maintenance range operations, for a Harvard cache implementation: `0x0` Not supported.
[7:4]	L1 Harvard background prefetch	Indicates the supported L1 cache background fetch operations, for a Harvard cache implementation: `0x0` Not supported.
[3:0]	L1 Harvard foreground prefetch	Indicates the supported L1 cache foreground fetch operations, for a Harvard cache implementation: `0x0` Not supported.

To access the ID_MMFR2, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c1, 6; Read Memory Model Feature Register 2
```

4.3.14 Memory Model Feature Register 3

The ID_MMFR3 characteristics are:

Purpose Provides information about the implemented memory model and memory management support of the processor.

Usage constraints The ID_MMFR3 is:

- A read-only register.

- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-12 shows the ID_MMFR3 bit assignments.

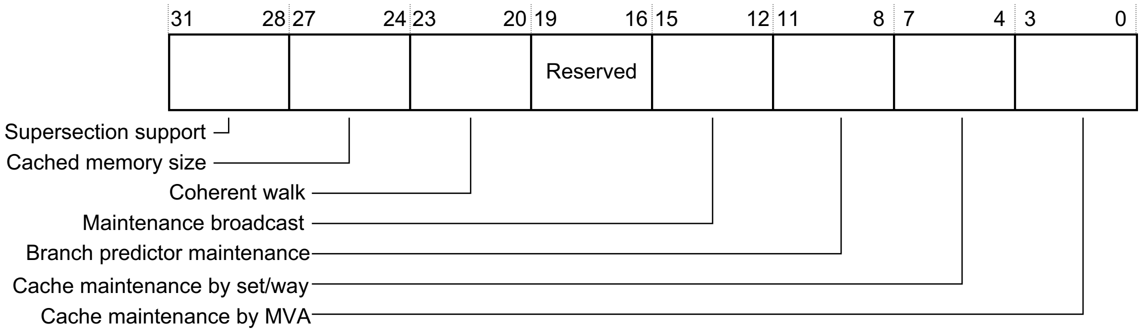


Figure 4-12 ID_MMFR3 bit assignments

Table 4-40 shows the ID_MMFR3 bit assignments.

Table 4-40 ID_MMFR3 bit assignments

Bits	Name	Function
[31:28]	Supersection support	Indicates support for supersections: `0x0` Processor supports supersections.
[27:24]	Cached memory size	Indicates the physical memory size supported by the processor caches: `0x2` Processor caches support 40-bit physical address range.
[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, RAZ.
[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.

Table 4-40 ID_MMFR3 bit assignments (continued)

Bits	Name	Function
[11:8]	Branch predictor maintenance	Indicates the supported branch predictor maintenance operations. `0x2` Processor supports: • Invalidate all branch predictors. • Invalidate branch predictors by MVA.
[7:4]	Cache maintenance by set/way	Indicates the supported cache maintenance operations by set/way. `0x1` Processor supports: • Invalidate data cache by set/way. • Clean data cache by set/way. • Clean and invalidate data cache by set/way.
[3:0]	Cache maintenance by MVA	Indicates the supported cache maintenance operations by MVA. `0x1` Processor supports: • Invalidate data cache by MVA. • 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, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c1, 7; Read Memory Model Feature Register 3
```

4.3.15 Instruction Set Attribute Register 0

The ID_ISAR0 characteristics are:

Purpose Provides information about the instruction set that the processor supports.

Usage constraints The ID_ISAR0 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-13 shows the ID_ISAR0 bit assignments.

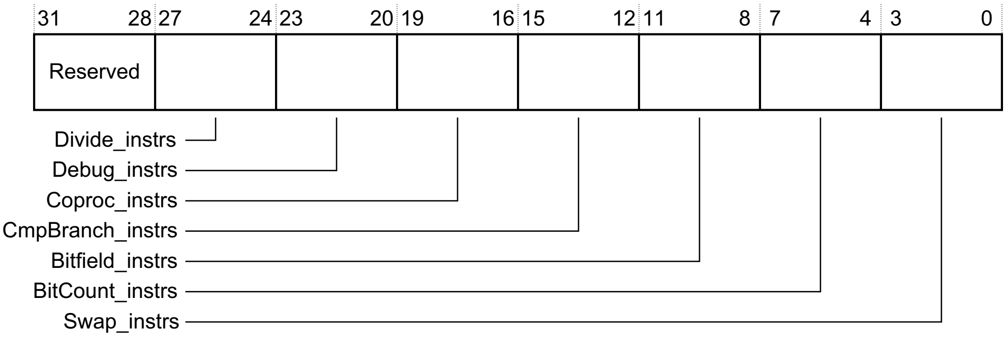


Figure 4-13 ID_ISAR0 bit assignments

Table 4-41 shows the ID_ISAR0 bit assignments.

Table 4-41 ID_ISAR0 bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RAZ.
[27:24]	Divide_instrs	Indicates support for Divide instructions. `0x2` Processor supports: • `SDIV` and `UDIV` in the Thumb instruction set. • `SDIV` and `UDIV` in the ARM instruction set.
[23:20]	Debug_instrs	Indicates the supported Debug instructions: `0x1` Processor supports `BKPT` instruction.
[19:16]	Coproc_instrs	Indicates the supported Coprocessor instructions: `0x0` None supported, except for separately attributed architectures including CP15, CP14, and Advanced SIMD and VFP.
[15:12]	CmpBranch_instrs	Indicates the supported combined Compare and Branch instructions in the Thumb instruction set: `0x1` Processor supports `CBNZ` and `CBZ` instructions.
[11:8]	Bitfield_instrs	Indicates the supported BifField instructions: `0x1` Processor supports `BFC`, `BFI`, `SBFX`, and `UBFX` instructions.
[7:4]	BitCount_instrs	Indicates the supported Bit Counting instructions: `0x1` Processor supports `CLZ` instruction.
[3:0]	Swap_instrs	Indicates the supported Swap instructions in the ARM instruction set: `0x0` Not supported.

To access the ID_ISAR0, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c2, 0 ; Read Instruction Set Attribute Register 0
```

4.3.16 Instruction Set Attribute Register 1

The ID_ISAR1 characteristics are:

Purpose Provides information about the instruction set that the processor supports.

Usage constraints The ID_ISAR1 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-14 on page 4-42 shows the ID_ISAR1 bit assignments.

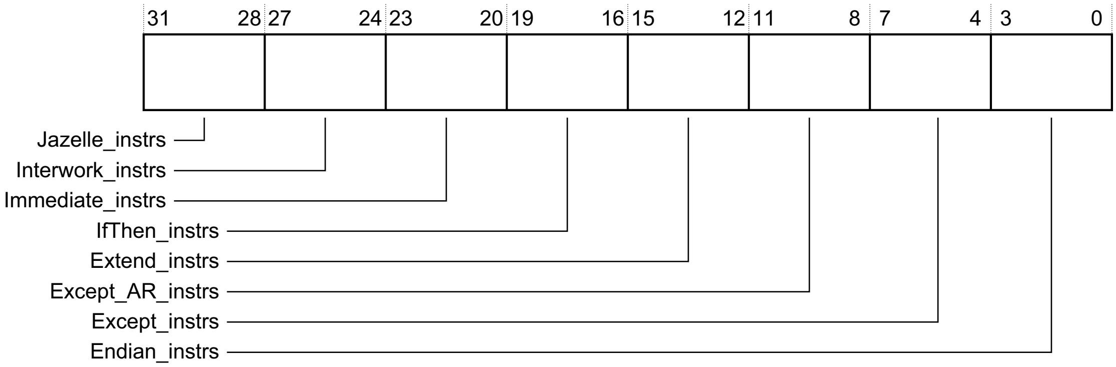


Figure 4-14 ID_ISAR1 bit assignments

Table 4-42 shows the ID_ISAR1 bit assignments.

Table 4-42 ID_ISAR1 bit assignments

Bits	Name	Function
[31:28]	Jazelle_instrs	Indicates the supported Jazelle extension instructions: `0x1` Processor supports `BXJ` instruction, and the J bit in the PSR.
[27:24]	Interwork_instrs	Indicates the supported Interworking instructions. `0x3` Processor supports: • `BX` instruction, and the T bit in the PSR. • `BLX` instruction, and PC loads have `BX`-like behavior. • Data-processing instructions in the ARM instruction set with the PC as the destination and the S bit clear have `BX`-like behavior.
[23:20]	Immediate_instrs	Indicates support for data-processing instructions with long immediates. `0x1` Processor supports: • `MOVT` instruction. • `MOV` instruction encodings with zero-extended 16-bit immediates. • Thumb `ADD` and `SUB` instruction encodings with zero-extended 12-bit immediates, and other `ADD`, `ADR`, and `SUB` encodings cross-referenced by the pseudocode for those encodings.
[19:16]	IfThen_instrs	Indicates the supported If-Then instructions in the Thumb instruction set: `0x1` Processor supports the `IT` instructions, and the IT bits in the PSRs.
[15:12]	Extend_instrs	Indicates the supported Extend instructions. `0x2` Processor supports: • `SXTB`, `SXTH`, `UXTB`, and `UXTH` instructions. • `SXTB16`, `SXTAB`, `SXTAB16`, `SXTAH`, `UXTB16`, `UXTAB`, `UXTAB16`, and `UXTAH` instructions. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[11:8]	Except_AR_instrs	Indicates the supported A and R profile exception-handling instructions: `0x1` Processor supports `SRS`, `RFE`, and `CPS` instructions.
[7:4]	Except_instrs	Indicates the supported exception-handling instructions in the ARM instruction set: `0x1` Processor supports `LDM` (exception return), `LDM` (user registers), and `STM` (user registers) instructions.
[3:0]	Endian_instrs	Indicates the supported Endian instructions: `0x1` Processor supports `SETEND` instruction, and the E bit in the PSRs.

To access the ID_ISAR1, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c2, 1 ; Read Instruction Set Attribute Register 1
```

4.3.17 Instruction Set Attribute Register 2

The ID_ISAR2 characteristics are:

Purpose Provides information about the instruction set that the processor supports.

Usage constraints The ID_ISAR2 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-15 shows the ID_ISAR2 bit assignments.

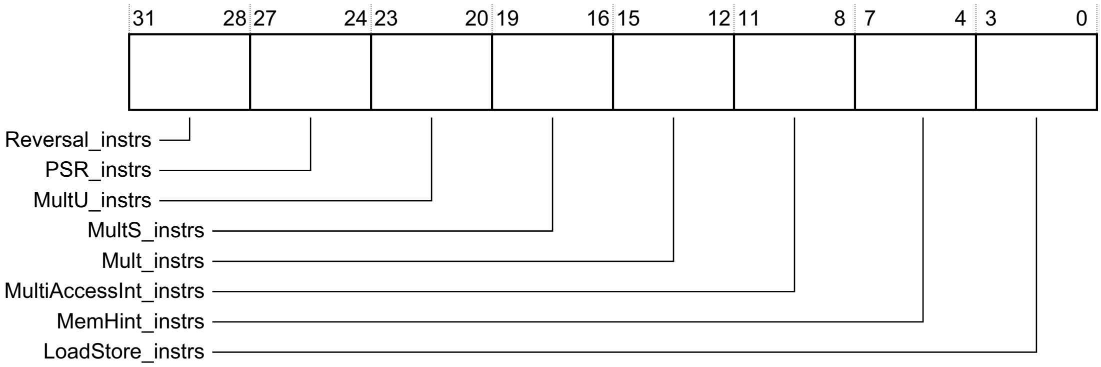


Figure 4-15 ID_ISAR2 bit assignments

Table 4-43 shows the ID_ISAR2 bit assignments.

Table 4-43 ID_ISAR2 bit assignments

Bits	Name	Function
[31:28]	Reversal_instrs	Indicates the supported Reversal instructions: `0x2` Processor supports `REV`, `REV16`, `REVSH`, and `RBIT` instructions.
[27:24]	PSR_instrs	Indicates the supported A profile instructions to manipulate the PSR: `0x1` Processor supports `MRS` and `MSR` instructions, and the exception return forms of data-processing instructions. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[23:20]	MultU_instrs	Indicates the supported advanced unsigned Multiply instructions: `0x2` Processor supports `UMULL`, `UMLAL`, and `UMAAL` instructions.

Table 4-43 ID_ISAR2 bit assignments (continued)

Bits	Name	Function
[19:16]	MultS_instrs	Indicates the supported advanced signed Multiply instructions. `0x3` Processor supports: • `SMULL` and `SMLAL` instructions. • `SMLABB`, `SMLABT`, `SMLALBB`,`SMLALBT`, `SMLALTB`, `SMLALTT`, `SMLATB`, `SMLATT`, `SMLAWB`, `SMLAWT`, `SMULBB`, `SMULBT`, `SMULTB`, `SMULTT`, `SMULWB`, `SMULWT` instructions, and the Q bit in the PSRs. • `SMLAD`, `SMLADX`, `SMLALD`, `SMLALDX`, `SMLSD`, `SMLSDX`, `SMLSLD`, `SMLSLDX`, `SMMLA`, `SMMLAR`, `SMMLS`, `SMMLSR`, `SMMUL`, `SMMULR`, `SMUAD`, `SMUADX`, `SMUSD`, and `SMUSDX` instructions.
[15:12]	Mult_instrs	Indicates the supported additional Multiply instructions: `0x2` Processor supports `MUL`, `MLA`, and `MLS` instructions.
[11:8]	MultiAccessInt_instrs	Indicates support for interruptible multi-access instructions: `0x0` None supported. This means that the `LDM` and `STM` instructions are not interruptible.
[7:4]	MemHint_instrs	Indicates the supported Memory Hint instructions: `0x4` Processor supports `PLD`, `PLI` (NOP), and `PLDW` instructions.
[3:0]	LoadStore_instrs	Indicates the supported additional load/store instructions: `0x1` Processor supports `LDRD` and `STRD` instructions.

To access the ID_ISAR2, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c2, 2 ; Read Instruction Set Attribute Register 2
```

4.3.18 Instruction Set Attribute Register 3

The ID_ISAR3 characteristics are:

Purpose Provides information about the instruction set that the processor supports.

Usage constraints The ID_ISAR3 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-16 on page 4-45 shows the ID_ISAR3 bit assignments.

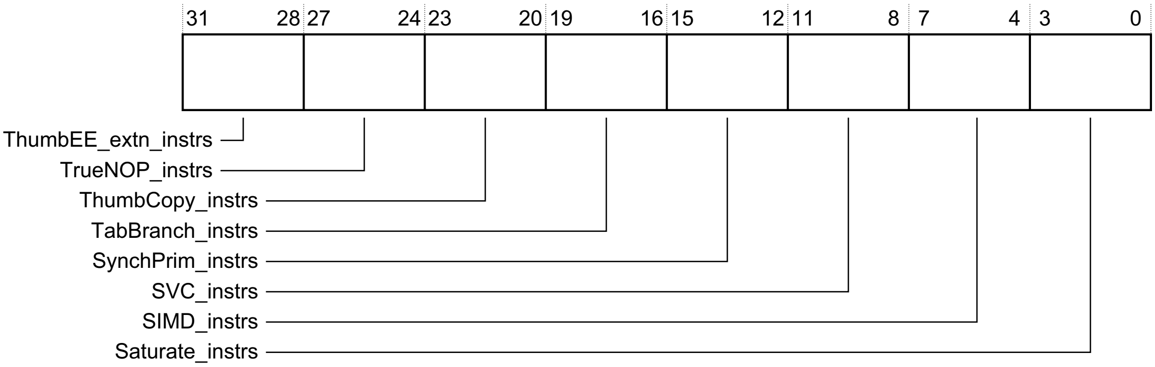


Figure 4-16 ID_ISAR3 bit assignments

Table 4-44 shows the ID_ISAR3 bit assignments.

Table 4-44 ID_ISAR3 bit assignments

Bits	Name	Function	
[31:28]	ThumbEE_extn_instrs	Indicates the supported *Thumb Execution Environment* (ThumbEE) extension instructions:	
		`0x1`	Processor supports `ENTERX` and `LEAVEX` instructions, and modifies the load behavior to include null checking.
[27:24]	TrueNOP_instrs	Indicates support for True `NOP` instructions:	
		`0x1`	Processor supports true `NOP` instructions in both the ARM and Thumb instruction sets, and the capability for additional NOP-compatible hints.
[23:20]	ThumbCopy_instrs	Indicates the supported Thumb non flag-setting `MOV` instructions:	
		`0x1`	Processor supports Thumb instruction set encoding T1 of the `MOV` (register) instruction, copying from a low register to a low register.
[19:16]	TabBranch_instrs	Indicates the supported Table Branch instructions in the Thumb instruction set:	
		`0x1`	Processor supports `TBB` and `TBH` instructions.
[15:12]	SynchPrim_instrs	This field is used with the SynchPrim_instrs_frac field of ID_ISAR4 to indicate the supported Synchronization Primitive instructions.	
		`0x2`	Processor supports: • `LDREX` and `STREX` instructions. • `CLREX`, `LDREXB`, `LDREXH`, `STREXB`, and `STREXH` instructions. • `LDREXD` and `STREXD` instructions.

Table 4-44 ID_ISAR3 bit assignments (continued)

Bits	Name	Function
[11:8]	SVC_instrs	Indicates the supported SVC instructions: 0x1 Processor supports SVC instruction. **Note** The SVC instruction was called the SWI instruction in previous versions of the ARM architecture.
[7:4]	SIMD_instrs	Indicates the supported *Single Instruction Multiple Data* (SIMD) instructions. 0x3 Processor supports: • SSAT and USAT instructions, and the Q bit in the PSRs. • 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. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[3:0]	Saturate_instrs	Indicates the supported Saturate instructions: 0x1 Processor supports QADD, QDADD, QDSUB, QSUB and the Q bit in the PSRs.

To access the ID_ISAR3, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c2, 3 ; Read Instruction Set Attribute Register 3
```

4.3.19 Instruction Set Attribute Register 4

The ID_ISAR4 characteristics are:

Purpose Provides information about the instruction set that the processor supports.

Usage constraints The ID_ISAR4 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-17 on page 4-47 shows the ID_ISAR4 bit assignments.

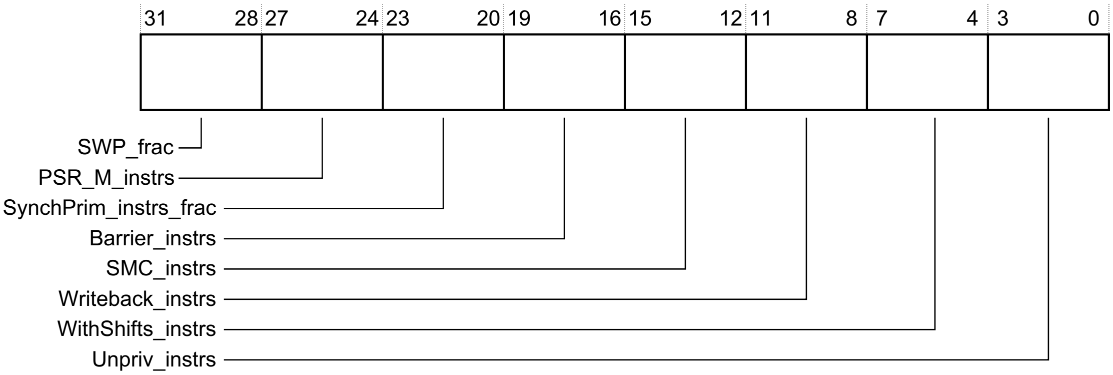


Figure 4-17 ID_ISAR4 bit assignments

Table 4-45 shows the ID_ISAR4 bit assignments.

Table 4-45 ID_ISAR4 bit assignments

Bits	Name	Function	
[31:28]	SWP_frac	Indicates support for the memory system locking the bus for `SWP` or `SWPB` instructions:	
		`0x1`	Processor supports `SWP` and `SWPB` instruction but only in a uniprocessor context. `SWP` and `SWPB` do not guarantee whether memory accesses from other masters can come between the load memory access and the store memory access of the `SWP` or `SWPB` instruction.
[27:24]	PSR_M_instrs	Indicates the supported M profile instructions to modify the PSRs:	
		`0x0`	None supported.
[23:20]	SynchPrim_instrs_frac	This field is used with the SynchPrim_instrs_frac field of ID_ISAR3 to indicate the supported Synchronization Primitive instructions.	
		`0x0`	Processor supports: • `LDREX` and `STREX` instructions. • `CLREX`, `LDREXB`, `LDREXH`, `STREXB`, and `STREXH` instructions. • `LDREXD` and `STREXD` instructions.
[19:16]	Barrier_instrs	Indicates the supported Barrier instructions in the ARM and Thumb instruction sets:	
		`0x1`	Processor supports `DMB`, `DSB`, and `ISB` barrier instructions.
[15:12]	SMC_instrs	Indicates the supported `SMC` instructions:	
		`0x1`	Processor supports `SMC` instruction.

Table 4-45 ID_ISAR4 bit assignments (continued)

Bits	Name	Function
[11:8]	Writeback_instrs	Indicates support for writeback addressing modes: `0x1` Processor supports all writeback addressing modes defined in ARMv7 architecture.
[7:4]	WithShifts_instrs	Indicates support for instructions with shifts. `0x4` Processor supports: • `LDRBT`, `LDRT`, `STRBT`, and `STRT` instructions. • `LDRHT`, `LDRSBT`, `LDRSHT`, and `STRHT` instructions. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[3:0]	Unpriv_instrs	Indicates the supported unprivileged instructions. `0x2` Processor supports: • Shifts of loads and stores over the range LSL 0-3. • Constant shift options, both on load/store and other instructions. • Register-controlled shift options.

To access the ID_ISAR4, read the CP15 register with:

```
MRC p15, 0, <Rt>, c0, c2, 4 ; Read Instruction Set Attribute Register 4
```

4.3.20 Instruction Set Attribute Register 5

ID_ISAR5 is reserved, so this register is always RAZ/WI.

4.3.21 Cache Size ID Register

The CCSIDR characteristics are:

Purpose Provides information about the architecture of the caches.

Usage constraints The CCSIDR is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

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

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-18 on page 4-49 shows the CCSIDR bit assignments.

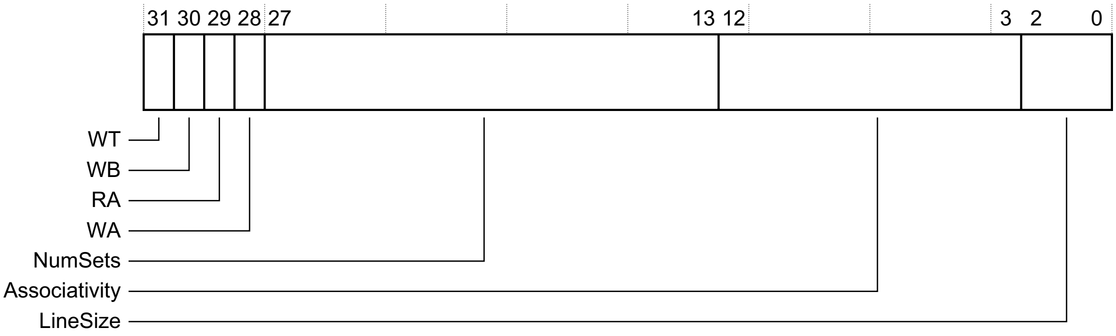


Figure 4-18 CCSIDR bit assignments

Table 4-46 shows the CCSIDR bit assignments.

Table 4-46 CCSIDR bit assignments

Bits	Name	Function
[31]	WT	Indicates support for Write-Through: **0** Cache level does not support Write-Through. **1** Cache level supports Write-Through.
[30]	WB	Indicates support for Write-Back: **0** Cache level does not support Write-Back. **1** Cache level supports Write-Back.
[29]	RA	Indicates support for Read-Allocation: **0** Cache level does not support Read-Allocation. **1** Cache level supports Read-Allocation.
[28]	WA	Indicates support for Write-Allocation: **0** Cache level does not support Write-Allocation. **1** Cache level supports Write-Allocation.
[27:13]	NumSets	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.
[12:3]	Associativity	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: `0x001` 2 ways. `0x00F` 16 ways.
[2:0]	LineSize	Indicates the ($\log_2$ (number of words in cache line)) - 2: `b010` 16 words per line.

Table 4-47 shows the individual bit field and complete register encodings for the CCSIDR. The CSSELR determines which CCSIDR to select.

Table 4-47 Encodings of the Cache Size ID Register

CSSELR	Size	Complete register encoding	Register bit field encoding						
			WT	WB	RA	WA	NumSets	Associativity	LineSize
`0x0`	32KB	`0x701FE00A`	0	1	1	1	`0xFF`	`0x1`	`0x2`
`0x1`	32KB	`0x201FE00A`	0	0	1	0	`0xFF`	`0x1`	`0x2`
`0x2`	512KB	`0x703FE07A`	0	1	1	1	`0x1FF`	`0xF`	`0x2`
	1024KB	`0x707FE07A`	0	1	1	1	`0x3FF`	`0xF`	`0x2`
	2048KB	`0x70FFE07A`	0	1	1	1	`0x7FF`	`0xF`	`0x2`
	4096KB	`0x71FFE07A`	0	1	1	1	`0xFFF`	`0xF`	`0x2`
`0x3-0xF`	-	-	Reserved						

To access the CCSIDR, read the CP15 register with:

```
MRC p15, 1, <Rt>, c0, c0, 0 ; Read Cache Size ID Register
```

4.3.22 Cache Level ID Register

The CLIDR characteristics are:

Purpose Identifies:

- The type of cache, or caches, implemented at each level, up to a maximum of seven levels.
- The Level of Coherency and Level of Unification for the cache hierarchy.

Usage constraints The CLIDR is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-19 shows the CLIDR bit assignments.

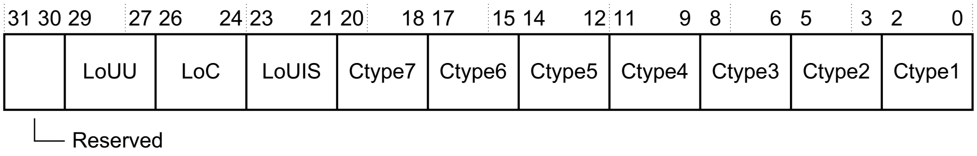


Figure 4-19 CLIDR bit assignments

Table 4-48 shows the CLIDR bit assignments.

Table 4-48 CLIDR bit assignments

Bits	Name	Function
[31:30]	-	Reserved, RAZ.
[29:27]]	LoUU	Indicates the Level of Unification Uniprocessor for the cache hierarchy: `b001` L2 cache.
[26:24]	LoC	Indicates the Level of Coherency for the cache hierarchy: `b010` L3 cache.
[23:21]	LoUIS	Indicates the Level of Unification Inner Shareable for the cache hierarchy: `b001` L2 cache.
[20:18]	Ctype7	Indicates the type of cache implemented at level 7: `b000` No cache.
[17:15]	Ctype6	Indicates the type of cache implemented at level 6: `b000` No cache.
[14:12]	Ctype5	Indicates the type of cache implemented at level 5: `b000` No cache.
[11:9]	Ctype4	Indicates the type of cache implemented at level 4: `b000` No cache.
[8:6]	Ctype3	Indicates the type of cache implemented at level 3: `b000` No cache.
[5:3]	Ctype2	Indicates the type of cache implemented at level 2: `b100` Unified cache.
[2:0]	Ctype1	Indicates the type of cache implemented at level 1: `b011` Separate instruction and data caches.

To access the CLIDR, read the CP15 register with:

```
MRC p15, 1, <Rt>, c0, c0, 1 ; Read Cache Level ID Register
```

4.3.23 Auxiliary ID Register

The processor does not implement AIDR, so this register is always RAZ/WI.

4.3.24 Cache Size Selection Register

The CSSELR characteristics are:

Purpose Selects the current CCSIDR, see *Cache Size ID Register* on page 4-48, by specifying:

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

Usage constraints The CSSELR is:

- A read/write register.
- Banked for the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-20 shows the CSSELR bit assignments.

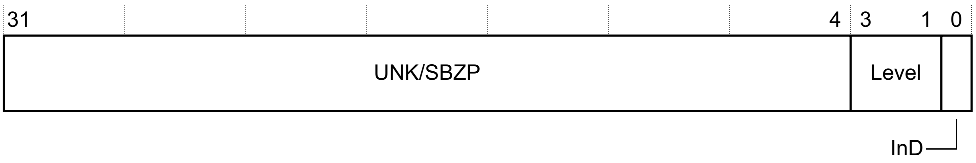


Figure 4-20 CSSELR bit assignments

Table 4-49 shows the CSSELR bit assignments.

Table 4-49 CSSELR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, UNK/SBZP
[3:1]	Level	Cache level of required cache: `b000` Level 1. `b001` Level 2. `b010-b111` Reserved.
[0]	InD	Instruction not Data bit: **0** Data or unified cache. **1** Instruction cache.

To access the CSSELR, read or write the CP15 register with:

```
MRC p15, 2, <Rt>, c0, c0, 0; Read Cache Size Selection Register
MCR p15, 2, <Rt>, c0, c0, 0; Write Cache Size Selection Register
```

4.3.25 Virtualization Processor ID Register

The VPIDR characteristics are:

Purpose Holds the value of the Virtualization Processor ID. A Non-secure read of the MIDR from PL1 returns the value of this register.

Usage constraints The VPIDR is:

- A read/write register.
- Only accessible from Hyp mode or from Monitor mode when SCR.NS is 1.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-21 on page 4-53 shows the VPIDR bit assignments.

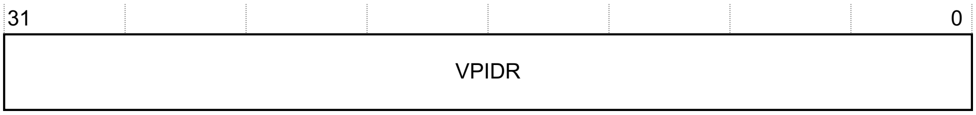


Figure 4-21 VPIDR bit assignments

Table 4-50 shows the VPIDR bit assignments.

Table 4-50 VPIDR bit assignments

Bits	Name	Function
[31:0]	VPIDR	MIDR value returned by Non-secure PL1 reads of the MIDR. For information on the subdivision of this value, see *Main ID Register* on page 4-27.

To access the VPIDR, read or write the CP15 register with:

```
MRC p15, 4, <Rt>, c0, c0, 0; Read Virtualization Processor ID Register
MCR p15, 4, <Rt>, c0, c0, 0; Write Virtualization Processor ID Register
```

4.3.26 Virtualization Multiprocessor ID Register

The VMPIDR characteristics are:

Purpose Holds the value of the Virtualization Multiprocessor ID. A Non-secure read of the MPIDR from PL1 returns the value of this register.

Usage constraints The VMPIDR is:

- A read/write register.
- Only accessible from Hyp mode or from Monitor mode when SCR.NS is 1.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-2 on page 4-4.

Figure 4-22 shows the VMPIDR bit assignments.

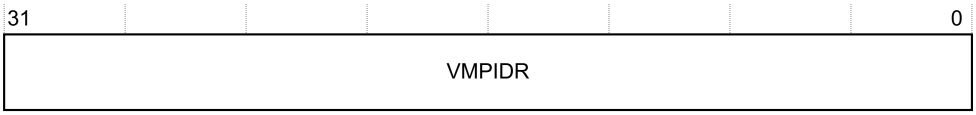


Figure 4-22 VMPIDR bit assignments

Table 4-51 shows the VMPIDR bit assignments.

Table 4-51 VMPIDR bit assignments

Bits	Name	Function
[31:0]	VMPIDR	MPIDR value returned by Non-secure PL1 reads of the MPIDR. For information on the subdivision of this value, see *Multiprocessor Affinity Register* on page 4-29.

To access the VMPIDR, read or write the CP15 register with:

```
MRC p15, 4, <Rt>, c0, c0, 5; Read Virtualization Multiprocessor ID Register
```

```
MCR p15, 4, <Rt>, c0, c0, 5; Write Virtualization Multiprocessor ID Register
```

4.3.27 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 a read/write register.
- Banked for Secure and Non-secure states for all implemented bits.
- Is only accessible from PL1 or higher.
- Has write access to the Secure copy of the register disabled when the **CP15SDISABLE** signal is asserted HIGH. Attempts to write to this register in Secure PL1 modes when **CP15SDISABLE** is HIGH result in an Undefined Instruction exception.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-3 on page 4-5.

Figure 4-23 shows the SCTLR bit assignments.

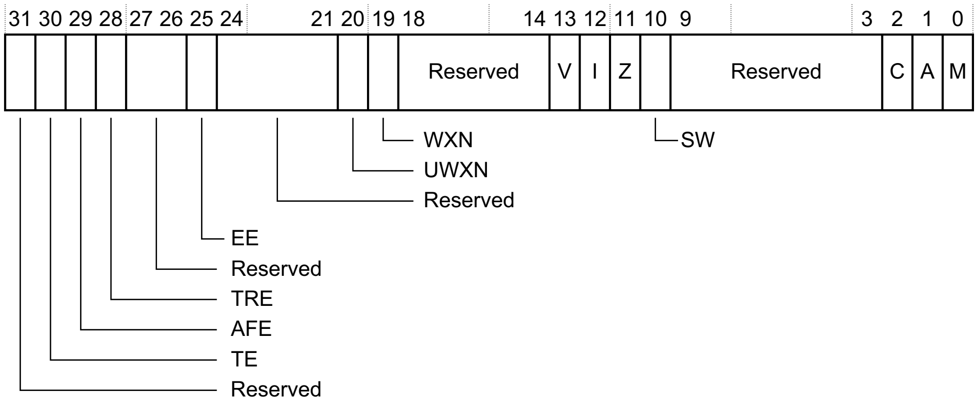


Figure 4-23 SCTLR bit assignments

Table 4-52 shows the SCTLR bit assignments.

Table 4-52 SCTLR bit assignments

Bits	Name	Access	Function
[31]	-	-	Reserved, UNK/SBZP.
[30]	TE	Banked	Thumb Exception enable. This bit controls whether exceptions are taken in ARM or Thumb state: **0** Exceptions, including reset, taken in ARM state. **1** Exceptions, including reset, taken in Thumb state. The primary input **CFGTE** defines the reset value of the TE bit.
[29]	AFE	Banked	Access flag enable. This bit enables use of the AP[0] bit in the translation table descriptors as the *Access flag*. It also restricts access permissions in the translation table descriptors to the simplified model as described in the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*: **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. When TTBCR.EAE is set to 1, to enable use of the Long-descriptor translation table format, this bit is UNK/SBOP.
[28]	TRE	Banked	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. When TTBCR.EAE is set to 1, to enable use of the Long-descriptor translation table format, this bit is UNK/SBOP. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[27]	-	-	Reserved, RAZ/WI.
[26]	-	-	Reserved, RAZ/SBZP.
[25]	EE	Banked	Exception Endianness. The value of this bit defines the value of the CPSR.E bit on entry to an exception vector, including reset. This value also indicates the endianness of the translation table data for translation table lookups: **0** Little endian. **1** Big endian. The primary input **CFGEND** defines the reset value of the EE bit.
[24]	-	-	Reserved, RAZ/WI.
[23:22]	-	-	Reserved, RAO/SBOP.
[21]	-	-	Reserved, RAZ/WI.

Table 4-52 SCTLR bit assignments (continued)

Bits	Name	Access	Function
[20]	UWXN	Banked	Unprivileged write permission implies PL1 *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 PL1: **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 PL1. This bit resets to 0 in both the Secure and the Non-secure copy of the register. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[19]	WXN	Banked	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. This bit resets to 0 in both the Secure and the Non-secure copy of the register. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[18]	-	-	Reserved, RAO/SBOP.
[17]	-	-	Reserved, RAZ/WI.
[16]	-	-	Reserved, RAO/SBOP.
[15]	-	-	Reserved, RAZ/SBZP.
[14]	-	-	Reserved, RAZ/WI.
[13]	V	Banked	Vectors bit. This bit selects the base address of the exception vectors: **0** Normal exception vectors, base address `0x00000000`. This base address can be remapped by updating the Vector Base Address Register. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*. **1** High exception vectors, base address `0xFFFF0000`. This base address is never remapped. The primary input **VINITHI** defines the reset value of the V bit.
[12]	I	Banked	Instruction cache enable. This is a global enable bit for instruction caches: **0** Instruction caches disabled. This is the reset value. **1** Instruction caches enabled.
[11]	Z	Banked	Branch prediction enable. This bit is used to enable branch prediction, also called program flow prediction: **0** Program flow prediction disabled. This is the reset value. **1** Program flow prediction enabled.
[10]	SW	Banked	SWP/SWPB enable bit. This bit enables the use of `SWP` and `SWPB` instructions: **0** `SWP` and `SWPB` are UNDEFINED. This is the reset value. **1** `SWP` and `SWPB` perform as described in the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.
[9:7]	-	-	Reserved, RAZ/SBZP.
[6:3]	-	-	Reserved, RAO/SBOP.

Table 4-52 SCTLR bit assignments (continued)

Bits	Name	Access	Function
[2]	C	Banked	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. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[1]	A	Banked	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. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[0]	M	Banked	MMU enable. This is a global enable bit for the PL1 and PL0 stage 1 MMU: **0** PL1 and PL0 stage 1 MMU disabled. This is the reset value. **1** PL1 and PL0 stage 1 MMU enabled. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

To access the SCTLR, read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c1, c0, 0 ; Read System Control Register
MCR p15, 0, <Rt>, c1, c0, 0 ; Write System Control Register
```

4.3.28 Auxiliary Control Register

The ACTLR characteristics are:

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

Usage constraints The ACTLR:

- Is a read/write register.
- Common to the Secure and Non-secure states.
- Is only accessible from PL1 or higher, with access rights that depend on the mode:
  - Read/write in Secure PL1 modes.
  - Read-only and write-ignored in Non-secure PL1 and PL2 modes if NSACR.NS_SMP is 0.
  - Read/write in Non-secure PL1 and PL2 modes if NSACR.NS_SMP is 1. In this case, all bits are write-ignored except for the SMP bit.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-3 on page 4-5.

Figure 4-24 on page 4-58 shows the ACTLR bit assignments.

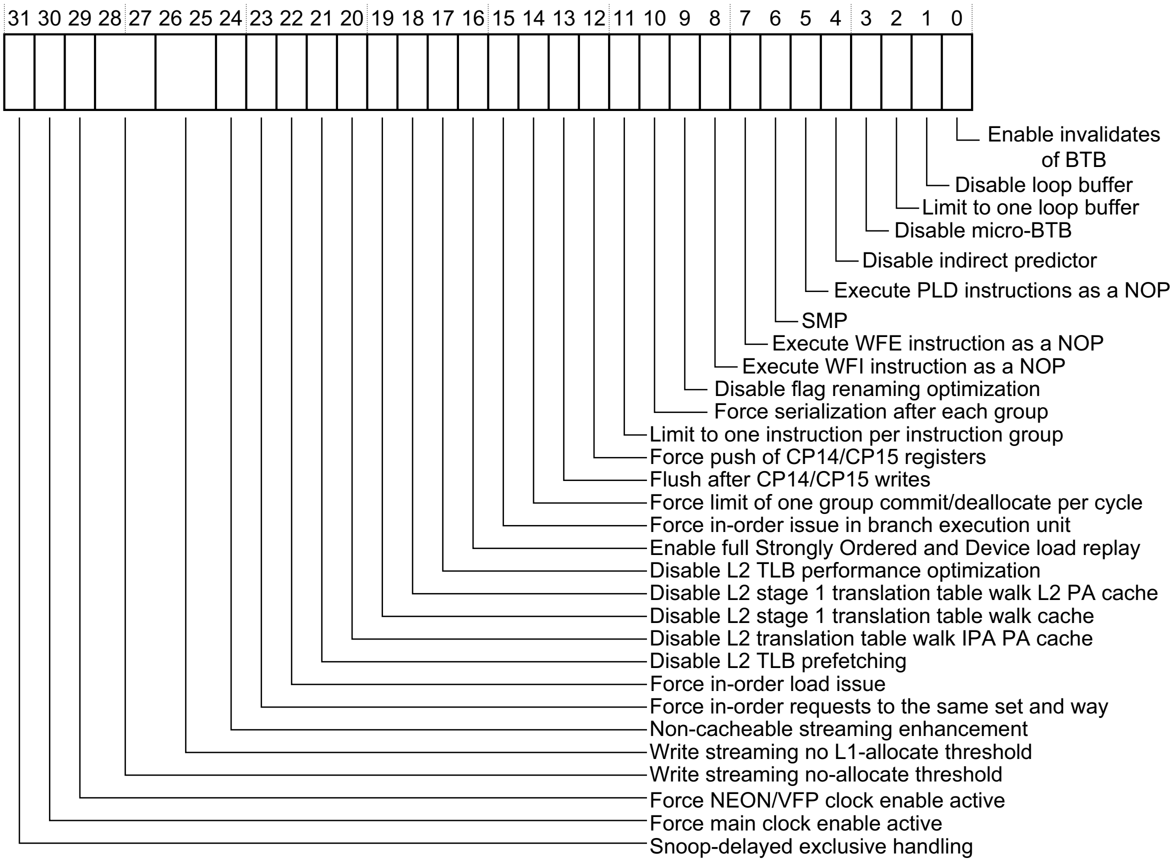

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Enable invalidates of BTB
Disable loop buffer
Limit to one loop buffer
Disable micro-BTB
Disable indirect predictor
Execute PLD instructions as a NOP
SMP
Execute WFE instruction as a NOP
Execute WFI instruction as a NOP
Disable flag renaming optimization
Force serialization after each group
Limit to one instruction per instruction group
Force push of CP14/CP15 registers
Flush after CP14/CP15 writes
Force limit of one group commit/deallocate per cycle
Force in-order issue in branch execution unit
Enable full Strongly Ordered and Device load replay
Disable L2 TLB performance optimization
Disable L2 stage 1 translation table walk L2 PA cache
Disable L2 stage 1 translation table walk cache
Disable L2 translation table walk IPA PA cache
Disable L2 TLB prefetching
Force in-order load issue
Force in-order requests to the same set and way
Non-cacheable streaming enhancement
Write streaming no L1-allocate threshold
Write streaming no-allocate threshold
Force NEON/VFP clock enable active
Force main clock enable active
Snoop-delayed exclusive handling

Figure 4-24 ACTLR bit assignments

Table 4-53 shows the ACTLR bit assignments.

Table 4-53 ACTLR bit assignments

Bits	Name	Function
[31]	Snoop-delayed exclusive handling	Snoop-delayed exclusive handling. You must set this bit to 1'b1 in any system with three or more processors that can execute exclusive accesses: **0** Normal exclusive handling behavior. This is the reset value. **1** Modifies exclusive handling behavior by delaying certain snoop requests.
[30][a]	Force main clock enable active	Forces main processor clock enable active: **0** Does not prevent the clock generator from stopping the processor clock. This is the reset value. **1** Prevents the clock generator from stopping the processor clock.
[29][a]	Force NEON/VFP clock enable active	Forces NEON and VFP unit clock enable active: **0** Does not prevent the clock generator from stopping the NEON and VFP unit clock. This is the reset value. **1** Prevents the clock generator from stopping the NEON and VFP unit clock.

Table 4-53 ACTLR bit assignments (continued)

Bits	Name	Function
[28:27]	Write streaming no-allocate threshold	Write streaming no-allocate threshold: `b00` 12th consecutive streaming cache line does not allocate in the L1 or L2 cache. This is the reset value. `b01` 128th consecutive streaming cache line does not allocate in the L1 or L2 cache. `b10` 512th consecutive streaming cache line does not allocate in the L1 or L2 cache. `b11` Disables streaming. All write-allocate lines allocate in the L1 or L2 cache.
[26:25]	Write streaming no L1-allocate threshold	Write streaming no L1-allocate threshold: `b00` 4th consecutive streaming cache line does not allocate in the L1 cache. This is the reset value. `b01` 64th consecutive streaming cache line does not allocate in the L1 cache. `b10` 128th consecutive streaming cache line does not allocate in the L1 cache. `b11` Disables streaming. All write-allocate lines allocate in the L1 cache.
[24]	Non-cacheable streaming enhancement	Non-cacheable streaming enhancement. You can set this bit only if your memory system meets the requirement that cache line fill requests from the Cortex-A15 MPCore processor are atomic. **0** Disables higher performance Non-Cacheable load forwarding. This is the reset value. **1** Enables higher performance Non-Cacheable load forwarding. See *Non-cacheable streaming enhancement* on page 6-9 for more information.
[23][a]	Force in-order requests to the same set and way	Forces in-order requests to the same set and way: **0** Does not force in-order requests to the same set and way. This is the reset value. **1** Forces in-order requests to the same set and way.
[22][a]	Force in-order load issue	Force in-order load issue: **0** Does not force in-order load issue. This is the reset value. **1** Forces in-order load issue.
[21][a]	Disable L2 TLB prefetching	Disables L2 TLB prefetching: **0** Enables L2 TLB prefetching. This is the reset value. **1** Disables L2 TLB prefetching.
[20][a]	Disable L2 translation table walk IPA PA cache	Disables L2 translation table walk *Intermediate Physical Address* (IPA) to *Physical Address* (PA) cache: **0** Enables L2 translation table walk IPA to PA cache. This is the reset value. **1** Disables L2 translation table walk IPA to PA cache.
[19][a]	Disable L2 stage 1 translation table walk cache	Disables L2 stage 1 translation table walk cache: **0** Enables L2 stage 1 translation table walk cache. This is the reset value. **1** Disables L2 stage 1 translation table walk cache.

[18][a]	Disable L2 stage 1 translation table walk L2 PA cache	Disables L2 stage 1 translation table walk L2 PA cache: **0** Enables L2 stage 1 translation table walk L2 PA cache. This is the reset value. **1** Disables L2 stage 1 translation table walk L2 PA cache.
[17][a]	Disable L2 TLB performance optimization	Disables L2 TLB performance optimization: **0** Enables L2 TLB optimization. This is the reset value. **1** Disables L2 TLB optimization.
[16][a]	Enable full Strongly-ordered and Device load replay	Enables full Strongly-ordered and Device load replay: **0** Disables full Strongly-ordered and Device load replay. This is the reset value. **1** Enables full Strongly-ordered and Device load replay.
[15][a]	Force in-order issue in branch execution unit	Forces in-order issue in branch execution unit: **0** Disables forced in-order issue. This is the reset value. **1** Forces in-order issue.
[14][a]	Force limit of one instruction group commit/de-allocate per cycle	Forces limit of one instruction group to commit and de-allocate per cycle: **0** Normal commit and de-allocate behavior. This is the reset value. **1** Limits commit and de-allocate to one instruction group per cycle.
[13][a]	Flush after CP14, CP15 writes	Flushes after certain CP14 and CP15 writes: **0** Normal behavior for CP14 and CP15 writes. This is the reset value. **1** Flushes after certain CP14 and CP15 writes.
[12][a]	Force push of CP14 and CP15 registers	Forces push of certain CP14 and CP15 registers from local dispatch copies to shadow copies: **0** Normal behavior for CP14 and CP15 instructions. This is the reset value. **1** Pushes certain CP14 and CP15 registers from local dispatch copies to shadow copies. **Noteto**

Table 4-53 ACTLR bit assignments (continued)

Bits	Name	Function
[9][a]	Disable flag renaming optimization	Disables flag renaming optimization: **0** Enables normal flag renaming optimization. This is the reset value. **1** Disables normal flag renaming optimization.
[8][a]	Execute `WFI` instruction as a `NOP` instruction	Executes `WFI` instruction as a `NOP` instruction: **0** Executes `WFI` instruction as defined in the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*. This is the reset value. **1** Executes `WFI` instruction as a `NOP` instruction, and does not put the processor in low-power state.
[7][a]	Execute `WFE` instruction as a `NOP` instruction	Executes `WFE` instruction as a `NOP` instruction: **0** Executes `WFE` instruction as defined in the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*. This is the reset value. **1** Executes `WFE` instruction as a `NOP` instruction, and does not put the processor in low-power state.
[6]	SMP	Enables the processor to receive instruction cache, BTB, and TLB maintenance operations from other processors. You must set this bit before enabling the caches and MMU, or performing any cache and TLB maintenance operations. You must clear this bit to 0 during a processor powerdown sequence. See *Power management* on page 2-21: **0** Disables receiving of instruction cache, BTB, and TLB maintenance operations. This is the reset value. **1** Enables receiving of instruction cache, BTB, and TLB maintenance operations. **Note** • Any processor instruction cache, BTB, and TLB maintenance operations can execute the request, regardless of the value of the SMP bit. • This bit has no impact on data cache maintenance operations. • In the Cortex-A15 MPCore processor, the L1 data cache and L2 cache are always coherent, for shared or non-shared data, regardless of the value of the SMP bit.
[5][a]	Execute PLD instructions as a `NOP`	Execute `PLD` and `PLDW` instructions as a `NOP` instruction: **0** Executes `PLD` and `PLDW` instructions as defined in the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*. This is the reset value. **1** Executes `PLD` and `PLDW` instructions as a `NOP` instruction.
[4][a]	Disable indirect predictor	Disables indirect predictor: **0** Enables indirect predictor. This is the reset value. **1** Disables indirect predictor.
[3][a]	Disable micro-BTB	Disables micro-*Branch Target Buffer* (BTB): **0** Enables micro-BTB. This is the reset value. **1** Disables micro-BTB.

Table 4-53 ACTLR bit assignments (continued)

Bits	Name	Function
[2][a]	Limit to one loop buffer detect per flush	Limits to one loop buffer detect per flush: **0** Normal behavior. This is the reset value. **1** Limits to one loop buffer detect per flush.
[1][a]	Disable loop buffer	Disables loop buffer: **0** Enables loop buffer. This is the reset value. **1** Disables loop buffer.
[0][a]	Enable invalidates of BTB	Enables invalidate of BTB: **0** The CP15 Invalidate Instruction Cache All and Invalidate Instruction Cache by MVA instructions only invalidates the instruction cache array. This is the reset value. **1** The CP15 Invalidate Instruction Cache All and Invalidate Instruction Cache by MVA instructions invalidates the instruction cache array and branch target buffer.

[a] This bit is provided for debugging and characterization purpose only. For normal operation, ARM recommends that you do not change the value of this bit from its reset value.

To access the ACTLR, read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c1, c0, 1 ; Read Auxiliary Control Register
MCR p15, 0, <Rt>, c1, c0, 1 ; Write Auxiliary Control Register
```

4.3.29 Coprocessor Access Control Register

The CPACR characteristics are:

Purpose Controls access to coprocessors CP10 and CP11.

Usage constraints The CPACR:

- Is a read/write register.
- Is Common to the Secure and Non-secure states.
- Is only accessible from PL1 or higher.
- Has no effect on instructions executed in Hyp mode.

Configurations Bits in the NSACR, see *Non-Secure Access Control Register* on page 4-65, control Non-secure access to the CPACR fields.

Attributes See the register summary in Table 4-3 on page 4-5.

Figure 4-25 shows the CPACR bit assignments.

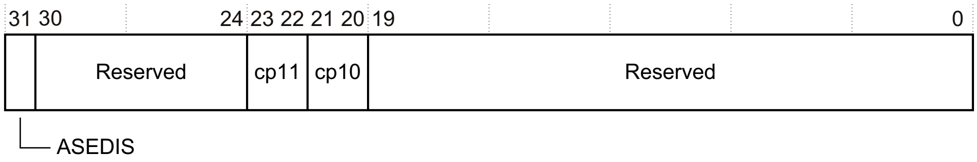


Figure 4-25 CPACR bit assignments

Table 4-54 shows the CPACR bit assignments.

Table 4-54 CPACR bit assignments

Bits	Name	Function
[31]	ASEDIS	Disable Advanced SIMD Extension functionality: **0** All Advanced SIMD and VFP instructions execute normally. **1** All Advanced SIMD instructions that are not VFP instructions are UNDEFINED when accessed from PL1 and Pl0 modes. If VFP is implemented and NEON is not implemented, this bit is RAO/WI. If VFP and NEON are not implemented, this bit is UNK/SBZP.
[30]	-	Reserved, RAZ/WI.
[29:28]	-	Reserved, UNK/SBZP.
[27:24]	-	Reserved, RAZ/WI.
[23:22]	cp11	Defines the access rights for coprocessor 11: `b00` Access denied. Any attempt to access the coprocessor generates an Undefined Instruction exception. This is the reset value. `b01` Access at PL1 or higher only. Any attempt to access the coprocessor from software executing at PL0 generates an Undefined Instruction exception. `b10` Reserved. The effect of this value is UNPREDICTABLE. `b11` Full access. The meaning of full access is defined by coprocessor 11. If VFP and NEON are not implemented, this field is RAZ/WI.
[21:20]	cp10	Defines the access rights for coprocessor 10: `b00` Access denied. Any attempt to access the coprocessor generates an Undefined Instruction exception. This is the reset value. `b01` Access at PL1 or higher only. Any attempt to access the coprocessor from software executing at PL0 generates an Undefined Instruction exception. `b10` Reserved. The effect of this value is UNPREDICTABLE. `b11` Full access. The meaning of full access is defined by coprocessor 10. If VFP and NEON are not implemented, this field is RAZ/WI.
[19:0]	-	Reserved, RAZ/WI.

Note

If the values of the cp11 and cp10 fields are not the same, the behavior is UNPREDICTABLE.

To access the CPACR, read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c1, c0, 2; Read Coprocessor Access Control Register
MCR p15, 0, <Rt>, c1, c0, 2; Write Coprocessor Access Control Register
```

4.3.30 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 mode 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 is 1.

Usage constraints The SCR is:

- A read/write register.
- A Restricted access register that exists only in the Secure state.
- Only accessible in Secure PL1 modes.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-3 on page 4-5.

Figure 4-26 shows the SCR bit assignments.

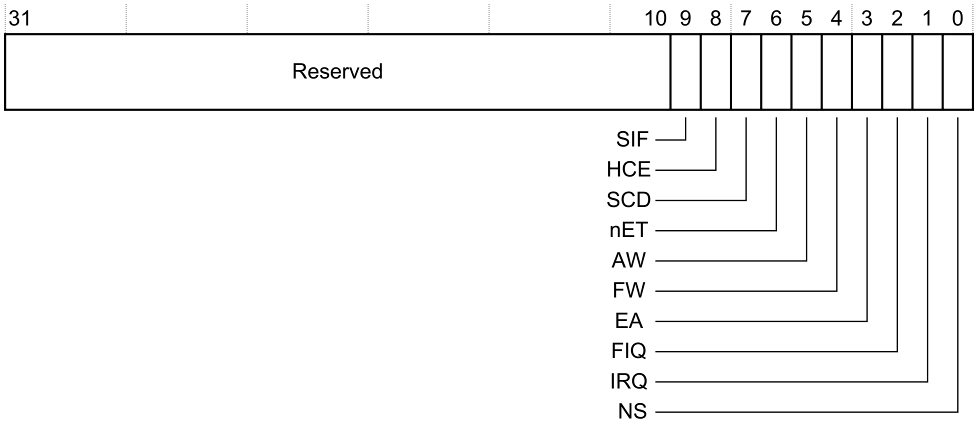


Figure 4-26 SCR bit assignments

Table 4-55 shows the SCR bit assignments.

Table 4-55 SCR bit assignments

Bits	Name	Function
[31:10]	-	Reserved, UNK/SBZP.
[9]	SIF	Secure Instruction Fetch. When the processor is in Secure state, this bit disables instruction fetches from Non-secure memory: **0** Secure state instruction fetches from Non-secure memory are permitted. This is the reset value. **1** Secure state instruction fetches from Non-secure memory are not permitted.
[8]	HCE	Hyp Call enable. This bit enables the use of `HVC` instruction from Non-secure PL1 modes: **0** The `HVC` instruction is UNDEFINED in Non-secure PL1 modes, and UNPREDICTABLE in Hyp mode. This is the reset value. **1** The `HVC` instruction is enabled in Non-secure PL1 modes, and performs a Hyp Call.
[7]	SCD	Secure Monitor Call disable. This bit causes the `SMC` instruction to be UNDEFINED in Non-secure state: **0** The `SMC` instruction executes normally in Non-secure state, and performs 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. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[6]	nET	Not Early Termination. This bit disables early termination of data operations. This bit is not implemented, UNK/SBZP.

Table 4-55 SCR bit assignments (continued)

Bits	Name	Function
[5]	AW	A bit writable. This bit controls whether CPSR.A can be modified in Non-secure state. For the Cortex-A15 MPCore processor: • This bit has no effect on whether CPSR.A can be modified in Non-secure state. The AW bit can be modified in either security state. • This bit, with the HCR.AMO bit, determines whether CPSR.A has any effect on exceptions that are routed to a Non-secure mode.
[4]	FW	F bit writable. This bit controls whether CPSR.F can be modified in Non-secure state. For the Cortex-A15 MPCore processor: • This bit has no effect on whether CPSR.F can be modified in Non-secure state. The FW bit can be modified in either security state. • This bit, with the HCR.FMO bit, determines whether CPSR.F has any effect on exceptions that are routed to a Non-secure mode.
[3]	EA	External Abort handler. This bit controls which mode takes external aborts: **0** External aborts taken in Abort mode. This is the reset value. **1** External aborts taken in Monitor mode.
[2]	FIQ	FIQ handler. This bit controls which mode takes FIQ exceptions: **0** FIQs taken in FIQ mode. This is the reset value. **1** FIQs taken in Monitor mode.
[1]	IRQ	IRQ handler. This bit controls which mode takes IRQ exceptions: **0** IRQs taken in IRQ mode. This is the reset value. **1** IRQs taken in Monitor mode.
[0]	NS	Non Secure bit. Except when the processor is in Monitor mode, this bit determines the security state of the processor: **0** Secure. This is the reset value. **1** Non-secure. **Note** When the processor is in Monitor mode, it is always in Secure state, regardless of the value of the NS bit. The value of the NS bit also affects the accessibility of the Banked CP15 registers in Monitor mode. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information on the NS bit.

To access the SCR, read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c1, c1, 0; Read Secure Configuration Register data
MCR p15, 0, <Rt>, c1, c1, 0; Write Secure Configuration Register data
```

4.3.31 Non-Secure Access Control Register

The NSACR characteristics are:

Purpose Defines the Non-secure access permission to coprocessors CP10 and CP11, and to the following bits:

- The SMP bit of the ACTLR, see *Auxiliary Control Register* on page 4-57.

- The L2 internal asynchronous error and AXI asynchronous error bits of the L2ECTLR, see *L2 Extended Control Register* on page 4-87.
- The ASEDIS bit of the CPACR, see *Coprocessor Access Control Register* on page 4-62.

Usage constraints The NSACR is:

- Only accessible from PL1 or higher, with access rights that depend on the mode and security state:
  - — Read/write in Secure PL1 modes.
  - — Read-only in Non-secure PL1 and PL2 modes.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-3 on page 4-5.

Figure 4-27 shows the NSACR bit assignments.

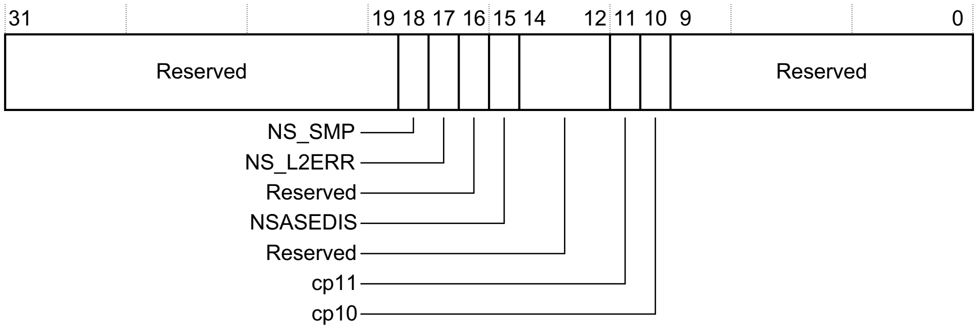


Figure 4-27 NSACR bit assignments

Table 4-56 shows the NSACR bit assignments.

Table 4-56 NSACR bit assignments

Bits	Name	Function
[31:20]	-	Reserved, UNK/SBZP.
[19]	-	Reserved, RAZ/WI.
[18]	NS_SMP	Determines if the SMP bit of the *Auxiliary Control Register*, see *Auxiliary Control Register* on page 4-57, is writable in Non-secure state: **0** A write to ACTLR in Non-secure state is ignored. This is the reset value. **1** A write to ACTLR in Non-secure state can modify the value of the SMP bit. Other bits in the ACTLR are write-ignored.
[17]	NS_L2ERR	Determines if the L2 internal asynchronous error and AXI asynchronous error bits of the *L2 Extended Control Register*, see *L2 Extended Control Register* on page 4-87, are writable in Non-secure state: **0** A write to L2ECTLR in Non-secure state is ignored. This is the reset value. **1** A write to L2ECTLR in Non-secure state can modify the value of the L2 internal asynchronous error and AXI asynchronous error bits. Other bits in the L2ECTLR are write-ignored.
[16]	-	Reserved.

Table 4-56 NSACR bit assignments (continued)

Bits	Name	Function
[15]	NSASEDIS	Disable Non-secure Advanced SIMD functionality: **0** This bit has no effect on the ability to write to CPACR.ASEDIS. This is the reset value. **1** When executing in Non-secure state, CPACR.ASEDIS is RAO/WI. If VFP is implemented and NEON is not implemented, this bit is RAO/WI. If VFP and NEON are not implemented, this bit is UNK/SBZP.
[14:12]	-	Reserved, RAZ/WI.
[11]	cp11	Non-secure access to coprocessor 11 enable: **0** Secure access only. Any attempt to access coprocessor 11 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 VFP and NEON are not implemented, this bit is RAZ/WI.
[10]	cp10	Non-secure access to coprocessor 10 enable: **0** Secure access only. Any attempt to access coprocessor 10 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 VFP and NEON are not implemented, this bit is RAZ/WI.
[9:0]	-	Reserved, RAZ/WI.

Note

If the values of the cp11 and cp10 fields are not the same, the behavior is UNPREDICTABLE.

To access the NSACR, read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c1, c1, 2 ; Read Non-Secure Access Control Register data
MCR p15, 0, <Rt>, c1, c1, 2 ; Write Non-Secure Access Control Register data
```

4.3.32 Hyp System Control Register

The HSCTLR characteristics are:

Purpose Provides the top level control of the system operation in Hyp mode. This register provides Hyp mode control of a subset of the features controlled by the SCTLR bits. See *System Control Register* on page 4-54.

Usage constraints The HSCTLR is:

- A read/write register.
- Only accessible from Hyp mode or from Monitor mode when SCR.NS is 1.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-3 on page 4-5.

Figure 4-28 on page 4-68 shows the HSCTLR bit assignments.

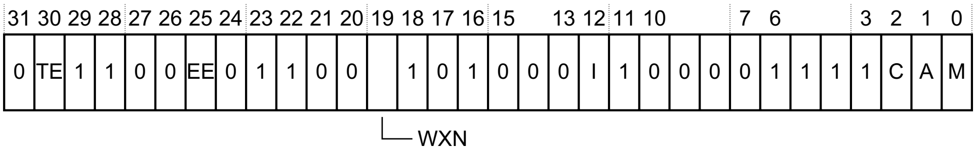


Figure 4-28 HSCTLR bit assignments

Table 4-57 shows the HSCTLR bit assignments.

Table 4-57 HSCTLR bit assignments

Bits	Name	Function
[31]	-	Reserved, RAZ/WI.
[30]	TE	Thumb Exception enable. This bit controls whether exceptions taken in Hyp mode are taken in ARM or Thumb state: **0** Exceptions taken in ARM state. **1** Exceptions taken in Thumb state.
[29:28]	-	Reserved, RAO/WI.
[27:26]	-	Reserved, RAZ/WI.
[25]	EE	Exception Endianness bit. The value of this bit defines the value of the CPSR.E bit on entry to an exception vector in Hyp mode. This value also indicates the endianness of the translation table data for translation table lookups, when executing in Hyp mode: **0** Little endian. **1** Big endian.
[24]	-	Reserved, RAZ/WI.
[23:22]	-	Reserved, RAO/WI.
[21]	FI	Fast Interrupts configuration enable bit. This bit can be used to reduce interrupt latency by disabling IMPLEMENTATION DEFINED performance features. This bit is not implemented, RAZ/WI.
[20]	-	Reserved, RAZ/WI.
[19]	WXN	Write permission implies *Execute Never* (XN): **0** Hyp translations that permit write are not forced to XN. **1** Hyp translations that permit write are forced to XN.
[18]	-	Reserved, RAO/WI.
[17]	-	Reserved, RAZ/WI.
[16]	-	Reserved, RAO/WI.
[15:13]	-	Reserved, RAZ/WI.
[12]	I	Instruction cache enable bit. This is a global enable bit for instruction caches, for memory accesses made in Hyp mode: **0** Instruction caches disabled. **1** Instruction caches enabled.
[11]	-	Reserved, RAO/WI.
[10:7]	-	Reserved, RAZ/WI.
[6:3]	-	Reserved, RAO/WI.

Table 4-57 HSCTLR bit assignments (continued)

Bits	Name	Function
[2]	C	Cache enable bit. This is a global enable bit for data and unified caches, for memory accesses made in Hyp mode: **0** Data and unified caches disabled. **1** Data and unified caches enabled.
[1]	A	Alignment bit. This is the enable bit for Alignment fault checking, for memory accesses made in Hyp mode: **0** Alignment fault checking disabled. **1** Alignment fault checking enabled.
[0]	M	MMU enable bit. This is a global enable bit for the PL2 stage 1 MMU: **0** PL2 stage 1 MMU disabled. **1** PL2 stage 1 MMU enabled.

To access the HSCTLR, read or write the CP15 register with:

```
MRC p15, 4, <Rt>, c1, c0, 0; Read Hyp System Control Register
MCR p15, 4, <Rt>, c1, c0, 0; Write Hyp System Control Register
```

4.3.33 Hyp Auxiliary Control Register

The processor does not implement HACTLR, so this register is UNK/SBZP in Hyp mode and in Monitor mode when SCR.NS is 1.

4.3.34 Hyp Debug Configuration Register

The HDCR characteristics are:

Purpose Controls the trapping to Hyp mode of Non-secure accesses, at PL1 or lower, to functions provided by the debug and trace architectures.

Usage constraints The HDCR is:

- A read/write register.
- Only accessible from Hyp mode or from Monitor mode when SCR.NS is 1.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-3 on page 4-5.

Figure 4-29 shows the HDCR bit assignments.

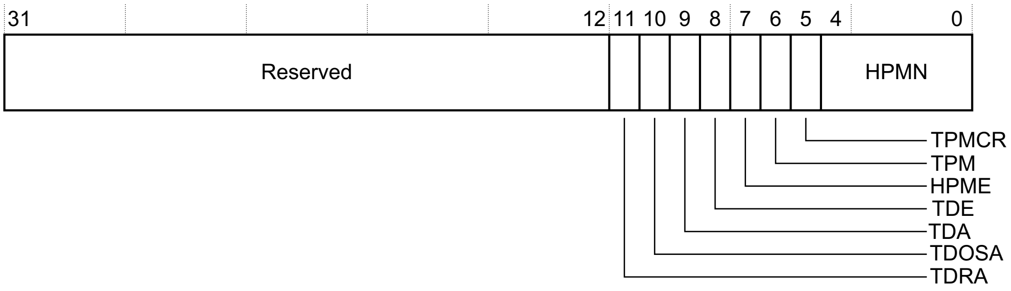


Figure 4-29 HDCR bit assignments

Table 4-58 shows the HDCR bit assignments.

Table 4-58 HDCR bit assignments

Bits	Name	Function
[31:12]	-	Reserved, UNK/SBZP.
[11]	TDRA	Trap Debug ROM Access: **0** Has no effect on Debug ROM accesses. **1** Trap valid Non-secure Debug ROM accesses to Hyp mode. When this bit is set to 1, any valid Non-secure access to the DBGDRAR or DBGDSAR register is trapped to Hyp mode. If this bit is set to 0 when TDA or TDE is set to 1, behavior is UNPREDICTABLE. This bit resets to 0.
[10]	TDOSA	Trap Debug OS-related register Access: **0** Has no effect on accesses to CP14 Debug registers. **1** Trap valid Non-secure accesses to CP14 OS-related Debug registers to Hyp mode. When this bit is set to 1, any valid Non-secure CP14 access to the following OS-related Debug registers is trapped to Hyp mode. • DBGOSLSR. • DBGOSLAR. • DBGOSDLR. • DBGPRCR. If this bit is set to 0 when TDA or TDE is set to 1, behavior is UNPREDICTABLE. This bit resets to 0.
[9]	TDA	Trap Debug Access: **0** Has no effect on accesses to CP14 Debug registers. **1** Trap valid Non-secure accesses to CP14 Debug registers to Hyp mode. If this bit is set to 0 when TDE is set to 1, behavior is UNPREDICTABLE. This bit resets to 0. If this bit is set to 1, then the TDRA and TDOSA bits must also be set to 1, otherwise the behavior is UNPREDICTABLE.
[8]	TDE	Trap Debug Exceptions: **0** Has no effect on Debug exceptions. **1** Trap valid 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. • The TDRA, TDOSA, and TDA bits must all be set to 1, otherwise behavior is UNPREDICTABLE. This bit resets to 0.
[7]	HPME	Hypervisor Performance Monitors Enable: **0** Hyp mode Performance Monitors counters disabled. **1** Hyp mode Performance Monitors counters enabled. When this bit is set to 1, access to the Performance Monitors counters 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.

To access the HDCR, read or write the CP15 register with:

```
MRC p15, 4, <Rt>, c1, c1, 1; Read Hyp Debug Configuration Register
MCR p15, 4, <Rt>, c1, c1, 1; Write Hyp Debug Configuration Register
```

4.3.35 Hyp Coprocessor Trap Register

The HCPTR characteristics are:

Purpose Controls the trapping to Hyp mode of Non-secure accesses, at PL1 or

Usage constraints The HCPTR is:

- A read/write register.
- Only accessible from Hyp mode or from Monitor mode when SCR.NS is 1.
- 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 descriptions for more information.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-3 on page 4-5.

Figure 4-30 shows the HCPTR bit assignments.

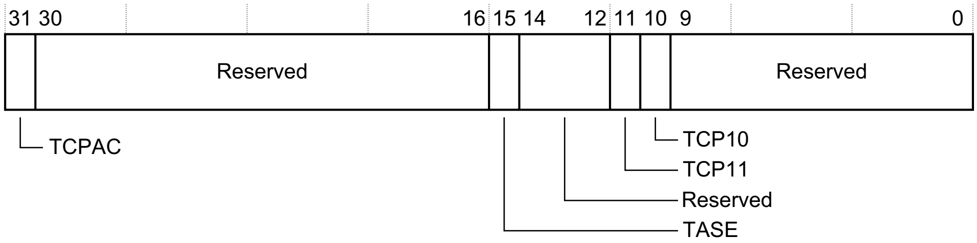


Figure 4-30 HCPTR bit assignments

Table 4-59 shows the HCPTR bit assignments.

Table 4-59 HCPTR bit assignments

Bits	Name	Function
[31]	TCPAC	Trap Coprocessor Access Control Register accesses: **0** Has no effect on CPACR accesses. **1** Trap valid Non-secure PL1 CPACR accesses to Hyp mode. When this bit is set to 1, any valid Non-secure PL1 or PL0 access to the CPACR is trapped to Hyp mode. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[30:16]	-	Reserved, UNK/SBZP.
[15]	TASE	Trap Advanced SIMD Extension 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. **Note** This bit value has no effect on possible use of the Advanced SIMD functionality from Non-secure PL1 and PL0 modes. **1** Trap valid Non-secure accesses to Advanced SIMD functionality to Hyp mode. When this bit is set to 1, any otherwise-valid access to Advanced SIMD functionality from: • A Non-secure PL1 or PL0 mode is trapped to Hyp mode. • Hyp mode generates an Undefined Instruction exception, taken in Hyp mode. **Note** If TCP10 and TCP11 are set to 1, then all Advanced SIMD use is trapped to Hyp mode, regardless of the value of this field. If VFP is implemented and NEON is not implemented, this bit is RAO/WI. If VFP and NEON 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, RAZ/WI.
[13:12]	-	Reserved, RAO/WI.

Table 4-59 HCPTR bit assignments (continued)

Bits	Name	Function
[11]	TCP11	Trap coprocessor 11: **0** If NSACR.CP11 is set to 1, then Hyp mode can access CP11, regardless of the value of CPACR.CP11. **Note** This bit value has no effect on possible use of CP11 from Non-secure PL1 and PL0 modes. **1** Trap valid Non-secure accesses to CP11 to Hyp mode. When TCP11 is set to 1, any otherwise-valid access to CP11 from: • A Non-secure PL1 or PL0 mode is trapped to Hyp mode. • Hyp mode generates an Undefined Instruction exception, taken in Hyp mode. If VFP and NEON are not implemented, this bit is RAO/WI. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[10]	TCP10	Trap coprocessor 10: **0** If NSACR.CP10 is set to 1, then Hyp mode can access CP10, regardless of the value of CPACR.CP10. **Note** This bit value has no effect on possible use of CP10 from Non-secure PL1 and PL0 modes. **1** Trap valid Non-secure accesses to CP10 to Hyp mode. When TCP10 is set to 1, any otherwise-valid access to CP10 from: • A Non-secure PL1 or PL0 mode is trapped to Hyp mode. • Hyp mode generates an Undefined Instruction exception, taken in Hyp mode. If VFP and NEON are not implemented, this bit is RAO/WI. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[9:0]	-	Reserved, RAO/WI.

To access the HCPTR, read or write the CP15 register with:

```
MRC p15, 4, <Rt>, c1, c1, 2; Read Hyp Coprocessor Trap Register
MCR p15, 4, <Rt>, c1, c1, 2; Write Hyp Coprocessor Trap Register
```

4.3.36 Hyp Auxiliary Configuration Register

The processor does not implement HACR, so this register is UNK/SBZP in Hyp mode and in Monitor mode when SCR.NS is 1.

4.3.37 Translation Table Base Register 0 and Register 1

The processor does not use any implementation-defined bits in the 32-bit TTBR0 and TTBR1 format, so these bits are UNK/SBZP.

4.3.38 Translation Table Base Control Register

The processor does not use any implementation-defined bits when using the Long-descriptor translation table format, so these bits are UNK/SBZP.

4.3.39 Hyp Translation Control Register

The processor does not use any implementation-defined bits in the HTCR, so these bits are UNK/SBZP.

4.3.40 Data Fault Status Register

The DFSR characteristics are:

Purpose Holds status information about the last data fault.

Usage constraints The DFSR is:

- A read/write register.
- Banked for Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-6 on page 4-6.

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

- *DFSR format when using the Short-descriptor translation table format*.
- *DFSR format when using the Long-descriptor translation table format* on page 4-77.

DFSR format when using the Short-descriptor translation table format

Figure 4-31 shows the DFSR bit assignments when using the Short-descriptor translation table format.

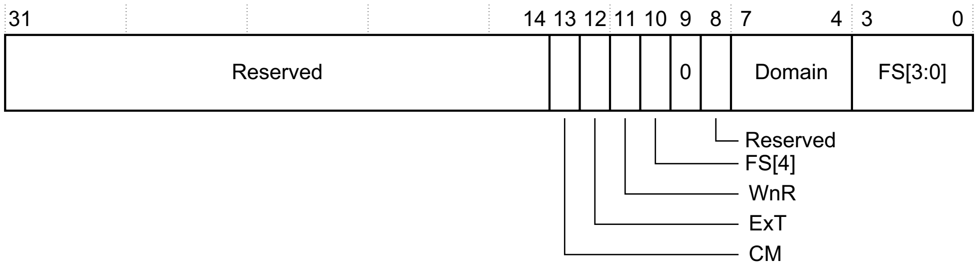


Figure 4-31 DFSR bit assignments for Short-descriptor translation table format

Table 4-60 shows the DFSR bit assignments when using the Short-descriptor translation table format.

Table 4-60 DFSR bit assignments for Short-descriptor translation table format

Bits	Name	Function
[31:14]	-	Reserved, UNK/SBZP.
[13]	CM	Cache maintenance fault. For synchronous faults, this bit indicates whether a cache maintenance operation generated the fault: **0** Abort not caused by a cache maintenance operation. **1** Abort caused by a cache maintenance operation. On an asynchronous fault, this bit is UNKNOWN.
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: **0** External abort marked as DECERR. **1** External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11]	WnR	Write not Read bit. This field indicates whether a write or a read access caused the abort: **0** Abort caused by a read access. **1** Abort caused by a write access. For faults on CP15 cache maintenance operations, including the VA to PA translation operations, this bit always returns a value of 1.
[10]	FS[4]	Part of the Fault Status field. See bits[3:0] in this table.
[9]	-	RAZ.
[8]	-	Reserved, UNK/SBZP.
[7:4]	Domain	The domain of the fault address. Specifies which of the 16 domains, D15-D0, was being accessed when a data fault occurred. ARMv7 deprecates any use of the domain field in the DFSR. For a Permission fault that generates a Data Abort exception, this field is UNKNOWN.
[3:0]	FS[3:0]	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved: `b00001` Alignment fault. `b00100` Instruction cache maintenance fault[a]. `b01100` Synchronous external abort on translation table walk, 1st level. `b01110` Synchronous external abort on translation table walk, 2nd level. `b11100` Synchronous parity error on translation table walk, 1st level. `b11110` Synchronous parity error on translation table walk, 2nd level. `b00101` Translation fault, 1st level. `b00111` Translation fault, 2nd level. `b00011` Access flag fault, 1st level. `b00110` Access flag fault, 2nd level. `b01001` Domain fault, 1st level. `b01011` Domain fault, 2nd level. `b01101` Permission fault, 1st level. `b01111` Permission fault, 2nd level. `b00010` Debug event. `b01000` Synchronous external abort, non-translation. `b11001` Synchronous parity error on memory access. `b10110` Asynchronous external abort. `b11000` Asynchronous parity error on memory access.

a. This fault is not generated by the Cortex-A15 MPCore processor.

DFSR format when using the Long-descriptor translation table format

Figure 4-32 shows the DFSR bit assignments when using the Long-descriptor translation table format.

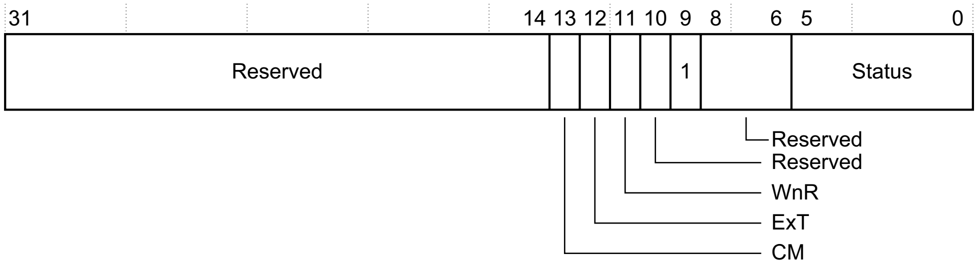


Figure 4-32 DFSR bit assignments for Long-descriptor translation table format

Table 4-61 shows the DFSR bit assignments when using the Long-descriptor translation table format.

Table 4-61 DFSR bit assignments for Long-descriptor translation table format

Bits	Name	Function
[31:14]	-	Reserved, UNK/SBZP.
[13]	CM	Cache maintenance fault. For synchronous faults, this bit indicates whether a cache maintenance operation generated the fault: **0** Abort not caused by a cache maintenance operation. **1** Abort caused by a cache maintenance operation. On an asynchronous fault, this bit is UNKNOWN.
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: **0** External abort marked as DECERR. **1** External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11]	WnR	Write not Read bit. This field indicates whether a write or a read access caused the abort: **0** Abort caused by a read access. **1** Abort caused by a write access. For faults on CP15 cache maintenance operations, including the VA to PA translation operations, this bit always returns a value of 1.
[10]	-	Reserved, UNK/SBZP.

Table 4-61 DFSR bit assignments for Long-descriptor translation table format (continued)

Bits	Name	Function
[9]	-	RAO.
[8:6]	-	Reserved, UNK/SBZP.
[5:0]	Status	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved: `b0001LL` Translation fault, LL bits indicate level. `b0010LL` Access flag fault, LL bits indicate level. `b0011LL` Permission fault, LL bits indicate level. `b010000` Synchronous external abort. `b011000` Synchronous parity error on memory access. `b010001` Asynchronous external abort. `b011001` Asynchronous parity error on memory access. `b0101LL` Synchronous external abort on translation table walk, LL bits indicate level. `b0111LL` Synchronous parity error on memory access on translation table walk, LL bits indicate level. `b100001` Alignment fault. `b100010` Debug event.

Table 4-62 shows how the LL bits in the Status field encode the lookup level associated with the MMU fault.

Table 4-62 Encodings of LL bits associated with the MMU fault

LL bits	Meaning
00	Reserved
01	First level
10	Second level
11	Third level

To access the DFSR, read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c5, c0, 0; Read Data Fault Status Register
MCR p15, 0, <Rt>, c5, c0, 0; Write Data Fault Status Register
```

4.3.41 Instruction Fault Status Register

The IFSR characteristics are:

Purpose Holds status information about the last instruction fault.

Usage constraints The IFSR is:

- A read/write register.
- Banked for Secure and Non-secure states.
- Accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-6 on page 4-6.

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

- *IFSR format when using the Short-descriptor translation table format*.
- *IFSR format when using the Long-descriptor translation table format* on page 4-80.

IFSR format when using the Short-descriptor translation table format

Figure 4-33 shows the IFSR bit assignments when using the Short-descriptor translation table format.

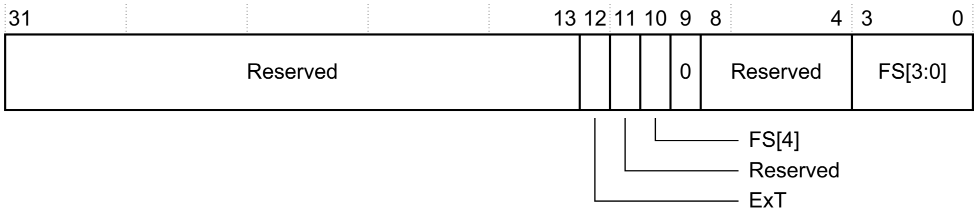


Figure 4-33 IFSR bit assignments for Short-descriptor translation table format

Table 4-63 shows the IFSR bit assignments when using the Short-descriptor translation table format.

Table 4-63 IFSR bit assignments for Short-descriptor translation table format

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

Table 4-63 IFSR bit assignments for Short-descriptor translation table format (continued)

Bits	Name	Function
[9]	-	RAZ.
[8:4]	-	Reserved, UNK/SBZP.
[3:0]	FS[3:0]	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved: `b01100` Synchronous external abort on translation table walk, 1st level. `b01110` Synchronous external abort on translation table walk, 2nd level. `b11100` Synchronous parity error on translation table walk, 1st level. `b11110` Synchronous parity error on translation table walk, 2nd level. `b00101` Translation fault, 1st level. `b00111` Translation fault, 2nd level. `b00011` Access flag fault, 1st level. `b00110` Access flag fault, 2nd level. `b01001` Domain fault, 1st level. `b01011` Domain fault, 2nd level. `b01101` Permission fault, 1st level. `b01111` Permission fault, 2nd level. `b00010` Debug event. `b01000` Synchronous external abort, non-translation. `b11001` Synchronous parity error on memory access.

IFSR format when using the Long-descriptor translation table format

Figure 4-34 shows the IFSR bit assignments when using the Long-descriptor translation table format.

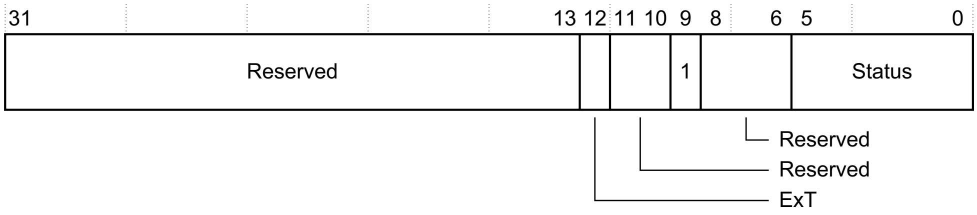


Figure 4-34 IFSR bit assignments for Long-descriptor translation table format

Table 4-64 shows the IFSR bit assignments when using the Long-descriptor translation table format.

Table 4-64 IFSR bit assignments for Long-descriptor translation table format

Bits	Name	Function
[31:13]	-	Reserved, UNK/SBZP.
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: **0** External abort marked as DECERR. **1** External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11:10]	-	Reserved, UNK/SBZP.

Table 4-64 IFSR bit assignments for Long-descriptor translation table format (continued)

Bits	Name	Function
[9]	-	RAO.
[8:6]	-	Reserved, UNK/SBZP.
[5:0]	Status	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved: `b0001LL` Translation fault, LL bits indicate level. `b0010LL` Access flag fault, LL bits indicate level. `b0011LL` Permission fault, LL bits indicate level. `b010000` Synchronous external abort. `b011000` Synchronous parity error on memory access. `b0101LL` Synchronous external abort on translation table walk, LL bits indicate level. `b0111LL` Synchronous parity error on memory access on translation table walk, LL bits indicate level. `b100010` Debug event.

Table 4-65 shows how the LL bits in the Status field encode the lookup level associated with the MMU fault.

Table 4-65 Encodings of LL bits associated with the MMU fault

LL Bits	Meaning
00	Reserved
01	First level
10	Second level
11	Third level

Note

If a Data Abort exception is generated by an instruction cache maintenance operation, 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, read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c5, c0, 1; Read Instruction Fault Status Register
MCR p15, 0, <Rt>, c5, c0, 1; Write Instruction Fault Status Register
```

4.3.42 Auxiliary Data Fault Status Register

The ADFSR characteristics are:

Purpose Holds the information about asynchronous L1 and L2 ECC double-bit errors.

Usage constraints The ADFSR is:

- A read/write register.
- Banked for Secure and Non-secure states.
- Accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-6 on page 4-6.

Figure 4-35 shows the ADFSR bit assignments.

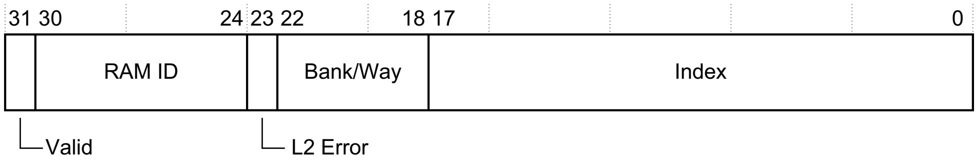


Figure 4-35 ADFSR bit assignments

Table 4-66 shows the ADFSR bit assignments.

Table 4-66 ADFSR bit assignments

Bits	Name	Function
[31]	Valid	Valid bit. This field indicates that an L1 or L2 ECC double-bit error has occurred: **0** No L1 or L2 ECC double-bit error occurred. **1** An L1 or L2 ECC double-bit error occurred.
[30:24]	RAM ID	RAM Identifier. This field indicates which RAM, the L1 ECC double-bit error occurred in: `0x8` L1 tag RAM. `0x9` L1 data RAM. If an L2 ECC double-bit error occurred, this field returns 0.
[23]	L2 Error	L2 Error bit. This field indicates an L2 ECC double-bit error occurred: **0** No L2 ECC double-bit error occurred. **1** An L2 ECC double-bit error occurred.
[22:18]	Bank/Way	Bank/Way bit. This field indicates which bank or way of the RAM, the L1 ECC double-bit error occurred in. If an L2 ECC double-bit error occurred, this field returns 0.
[17:0]	Index	Index. This field indicates the index address of the RAM, the L1 ECC double-bit error occurred in. If an L2 ECC double-bit error occurred, this field returns 0.

To access the ADFSR, read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c5, c1, 0; Read Auxiliary Data Fault Status Register
MCR p15, 0, <Rt>, c5, c1, 0; Write Auxiliary Data Fault Status Register
```

4.3.43 Auxiliary Instruction Fault Status Register

The processor does not implement the AIFSR, so this register is always RAZ/WI.

4.3.44 Hyp Auxiliary Data Fault Syndrome Register

The HADFSR characteristics is:

Purpose Holds syndrome information for the asynchronous L1 and L2 ECC double-bit errors that occurred in Hyp mode.

Usage constraints The HADFSR is:

- A read/write register.
- Only accessible from Hyp mode or from Monitor mode when SCR.NS is 1.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-6 on page 4-6.

Figure 4-36 shows the HADFSR bit assignments.

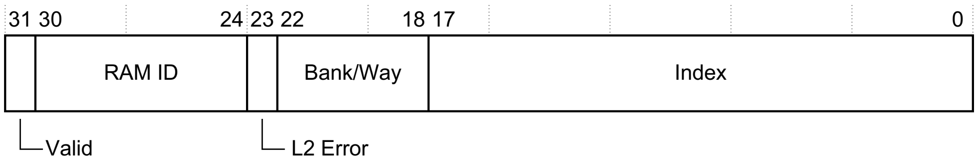


Figure 4-36 HADFSR bit assignments

Table 4-67 shows the HADFSR bit assignments.

Table 4-67 HADFSR bit assignments

Bits	Name	Function
[31]	Valid	Valid bit. This field indicates that an L1 or L2 ECC double-bit error has occurred: **0** No L1 or L2 ECC double-bit error occurred. **1** An L1 or L2 ECC double-bit error occurred.
[30:24]	RAM ID	RAM Identifier. This field indicates which RAM, the L1 ECC double-bit error occurred in: `0x8` L1 tag RAM. `0x9` L1 data RAM. If an L2 ECC double-bit error occurred, this field returns 0.
[23]	L2 Error	L2 Error bit. This field indicates an L2 ECC double-bit error occurred: **0** No L2 ECC double-bit error occurred. **1** An L2 ECC double-bit error occurred.
[22:18]	Bank/Way	Bank/Way bit. This field indicates which bank or way of the RAM, the L1 ECC double-bit error occurred in. If an L2 ECC double-bit error occurred, this field returns 0.
[17:0]	Index	Index. This field indicates the index address of the RAM, the L1 ECC double-bit error occurred in. If an L2 ECC double-bit error occurred, this field returns 0.

To access the HADFSR, read or write the CP15 register with:

```
MRC p15, 4, <Rt>, c5, c1, 0; Read Hyp Auxiliary Data Fault Status Syndrome Register
MCR p15, 4, <Rt>, c5, c1, 0; Write Hyp Auxiliary Data Fault Status Syndrome Register
```

4.3.45 Hyp Auxiliary Instruction Fault Syndrome Register

The processor does not implement HAIFSR, so this register is always RAZ/WI.

4.3.46 Hyp Syndrome Register

The HSR characteristics are:

Purpose Holds syndrome information for an exception taken to Hyp mode.

Usage constraints The HSR is:

- A read/write register.
- Only accessible from Hyp mode or from Monitor mode when SCR.NS is 1

- UNKNOWN when executing in Non-secure modes other than Hyp mode.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-26 on page 4-23.

Figure 4-37 shows the HSR bit assignments.

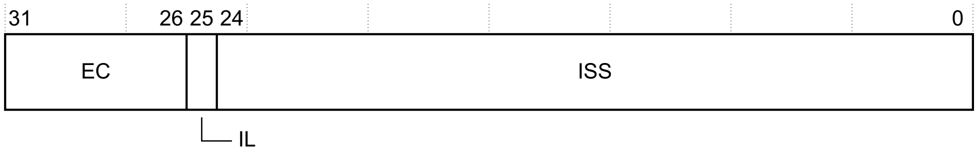


Figure 4-37 HSR bit assignments

Table 4-68 shows the HSR bit assignments.

Table 4-68 HSR bit assignments

Bits	Name	Function
[31:26]	EC	Exception class. The exception class for the exception that is taken to Hyp mode. When zero, this field indicates that the reason for the exception is not known. In this case, the other fields in this register are UNKNOWN. Otherwise, the field holds the exception class for the exception. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[25]	IL	Instruction length. Indicates the size of the instruction that has been trapped to Hyp mode: **0** 16-bit instruction. **1** 32-bit instruction. This field is not valid for: • Prefetch Aborts. • Data Aborts that do not have ISS information, or for which the ISS is not valid. In these cases the field is UNK/SBZP.
[24:0]	ISS	Instruction specific syndrome. The interpretation of this field depends on the value of the EC field. See *Encoding of ISS[24:20] when HSR[31:30] is 0b00*.

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 UNK/SBZP.

When an instruction is trapped, the COND field is `0xE`.

4.3.47 Physical Address Register

The processor does not use any implementation-defined bits in the 32-bit format or 64-bit format PAR, so these bits are UNK/SBZP.

4.3.48 L2 Control Register

The L2CTLR characteristics are:

Purpose Provides control options for the L2 memory system and ECC/parity support.

Usage constraints The L2CTLR is:

- A read/write register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher, with access rights that depend on the mode:
  - — Read/write in Secure PL1 modes with some bits that are read-only.
  - — Read-only and write-ignored in Non-secure PL1 and PL2 modes.
- This register can only be written 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 ACE 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 processor 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-A15 MPCore processor.
  3. Ensure that the system has no outstanding ACP requests to the Cortex-A15 MPCore processor.

When the L2 is idle, the processor can update the L2CTLR followed by an ISB. After the L2CTLR is updated, the MMUs can be enabled and normal ACE and ACP traffic can resume.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-10 on page 4-10.

Figure 4-38 on page 4-86 shows the L2CTLR bit assignments.

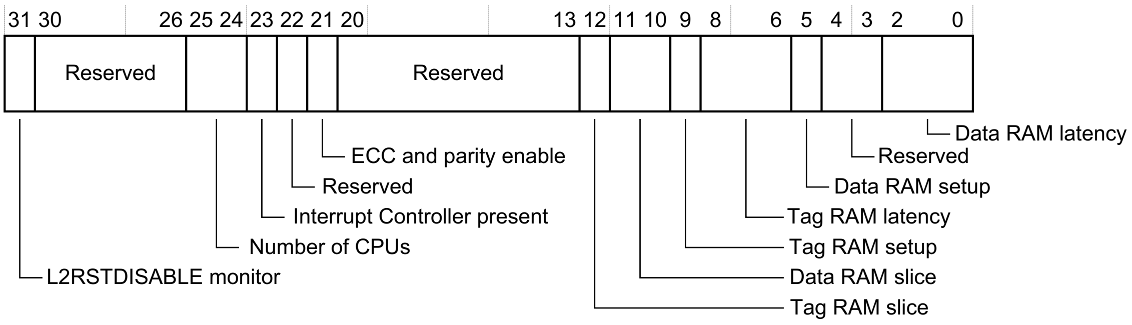


Figure 4-38 L2CTLR bit assignments

Table 4-69 shows the L2CTLR bit assignments.

Table 4-69 L2CTLR bit assignments

Bits	Name	Function
[31]	L2RSTDISABLE monitor	Monitors the L2 hardware reset disable pin, **L2RSTDISABLE**: **0** L2 valid RAM contents are reset by hardware. **1** L2 valid RAM contents are not reset by hardware. This bit is read-only and the reset value is determined by the primary input, **L2RSTDISABLE**.
[30:26]	-	Reserved, RAZ/WI.
[25:24]	Number of processors	Number of processors present: `b00` One processor, CPU0. `b01` Two processors, CPU0 and CPU1. `b10` Three processors, CPU0, CPU1, and CPU2. `b11` Four processors, CPU0, CPU1, CPU2, and CPU3. These bits are read-only and the reset value of this field is set to the number of processors present in the configuration.
[23]	Interrupt Controller	Interrupt Controller: **0** Interrupt Controller not present. **1** Interrupt Controller present. This is a read-only bit and the reset value depends on whether the Interrupt Controller is present.
[22]	-	Reserved, RAZ/WI.
[21]	ECC and parity enable	ECC and parity enable bit in L1 and L2 caches: **0** Disables ECC and parity. This is the reset value. **1** Enables ECC and parity. If ECC/parity is not implemented in L1 and L2 caches, this bit is RAZ/WI.
[20:13]	-	Reserved, RAZ/WI.
[12]	Tag RAM slice	L2 tag RAM slice: **0** 0 slice. **1** 1 slice. This is a read-only bit and the reset value of this field is set to the number of tag RAM slice present in the configuration. See *Register slice support for large cache sizes* on page 7-5 for more information.

Table 4-69 L2CTLR bit assignments (continued)

Bits	Name	Function
[11:10]	Data RAM slice	L2 data RAM slice: `b00` 0 slice. `b01` 1 slice. `b10` 2 slices. `b11` Invalid value. These are read-only bits and the reset value of this field is set to the number of data RAM slice present in the configuration. See *Register slice support for large cache sizes* on page 7-5 for more information.
[9]	Tag RAM setup	L2 tag RAM setup: **0** 0 cycle. This the reset value. **1** 1 cycle.
[8:6]	Tag RAM latency	L2 tag RAM latency: `b000` 2 cycles. This is the reset value. `b001` 2 cycles. `b010` 3 cycles. `b011` 4 cycles. `b1xx` 5 cycles, where *x* can be any value.
[5]	Data RAM setup	L2 data RAM setup: **0** 0 cycle. This the reset value. **1** 1 cycle.
[4:3]	-	Reserved, RAZ/WI.
[2:0]	Data RAM latency	L2 data RAM latency: `b000` 2 cycles. This is the reset value. `b001` 2 cycles. `b010` 3 cycles. `b011` 4 cycles. `b100` 5 cycles. `b101` 6 cycles. `b110` 7 cycles. `b111` 8 cycles.

To access the L2CTLR, read or write the CP15 register with:

```
MRC p15, 1, <Rt>, c9, c0, 2; Read L2 Control Register
MCR p15, 1, <Rt>, c9, c0, 2; Write L2 Control Register
```

4.3.49 L2 Extended Control Register

The L2ECTLR characteristics are:

Purpose Provides additional control options for the L2 memory system.

Usage constraints The L2ECTLR is:

- A read/write register.
- Common to the Secure and Non-secure states.

- Only accessible from PL1 or higher, with access rights that depend on the mode:
  - — Read/write in Secure PL1 modes.
  - — Read-only and write-ignored in Non-secure PL1 and PL2 modes if NSACR.NS_L2ERR is 0.
  - — Read/write in Non-secure PL1 and PL2 modes if NSACR.NS_L2ERR is 1. In this case, all bits are write-ignored except for bits[30:29].

Configurations Available in all configurations.

Attributes See the register summary in Table 4-10 on page 4-10.

Figure 4-39 shows the L2ECTLR bit assignments.

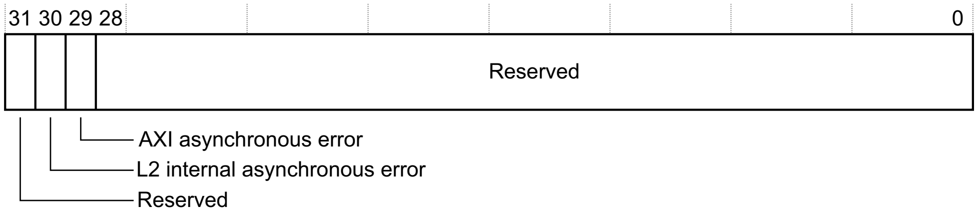


Figure 4-39 L2ECTLR bit assignments

Table 4-70 shows the L2ECTLR bit assignments.

Table 4-70 L2ECTLR bit assignments

Bits	Name	Function
[31]	-	Reserved, RAZ/WI.
[30]	L2 internal asynchronous error	L2 internal asynchronous error caused by L2 RAM double-bit ECC error or illegal writes to the Interrupt Controller memory-map region: **0** No pending L2 internal asynchronous error. This is the reset value. **1** L2 internal asynchronous error has occurred. A write of 0 clears this bit. A write of 1 is ignored.
[29]	AXI asynchronous error	AXI asynchronous error indication: **0** No pending AXI asynchronous error. This is the reset value. **1** AXI asynchronous error has occurred. A write of 0 clears this bit. A write of 1 is ignored.
[28:0]	-	Reserved, RAZ/WI.

To access the L2ECTLR, read or write the CP15 register with:

```
MRC p15, 1, <Rt>, c9, c0, 3; Read L2 Extended Control Register
MCR p15, 1, <Rt>, c9, c0, 3; Write L2 Extended Control Register
```

4.3.50 Memory Attribute Indirection Register 0

The processor behavior for all implementation-defined encodings in the MAIR0 register is UNPREDICTABLE.

4.3.51 Memory Attribute Indirection Register 1

The processor behavior for all implementation-defined encodings in the MAIR1 register is UNPREDICTABLE.

4.3.52 Auxiliary Memory Attribute Indirection Register 0

The processor behavior for all implementation-defined encodings in the AMAIR0 register is UNPREDICTABLE.

4.3.53 Auxiliary Memory Attribute Indirection Register 1

The processor behavior for all implementation-defined encodings in the AMAIR1 register is UNPREDICTABLE.

4.3.54 Hyp Auxiliary Memory Attribute Indirection Register 0

The processor behavior for all implementation-defined encodings in the HAMAIR0 register is UNPREDICTABLE.

4.3.55 Hyp Auxiliary Memory Attribute Indirection Register 1

The processor behavior for all implementation-defined encodings in the HAMAIR1 register is UNPREDICTABLE.

4.3.56 FCSE Process ID Register

The processor does not implement *Fast Context Switch Extension* (FCSE), so this register is always RAZ/WI.

4.3.57 Instruction L1 Data n Register

The IL1Data*n*, where *n* is from 0 to 2, characteristics are:

Purpose Holds the instruction side L1 array information returned by the RAMINDEX write operation. See *RAM Index Register* on page 4-91 for more information.

Note

Because the data, BTB, GHB, and TLB arrays are greater than 32-bits wide, the processor contains multiple IL1Data registers, to hold the array information.

Usage constraints The IL1Data*n* is:

- A read/write register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-14 on page 4-13.

Figure 4-40 on page 4-90 shows the IL1Data*n* bit assignments.

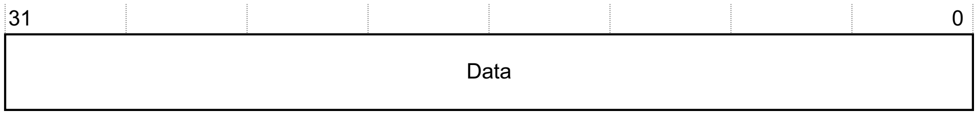


Figure 4-40 IL1Data*n* bit assignments

Table 4-71 shows the IL1Data*n* bit assignments.

Table 4-71 IL1Data*n* bit assignments

Bits	Name	Function
[31:0]	Data	Holds the instruction side L1 array information

To access the IL1Data*n*, read or write the CP15 registers with:

```
MRC p15, 0, <Rt>, c15, c0, n; Read Instruction L1 Data n Register
MCR p15, 0, <Rt>, c15, c0, n; Write Instruction L1 Data n Register
```

where *n* is 0, 1, or 2 for the Opcode_2 value of IL1Data0, IL1Data1, or IL1Data2 Register.

4.3.58 Data L1 Data n Register

The DL1Data*n*, where *n* is from 0 to 3, characteristics are:

Purpose Holds the data side L1 or L2 array information returned by the RAMINDEX write operation. See *RAM Index Register* on page 4-91 for more information.

Note

Because the data, tag and TLB arrays are greater than 32-bits wide, the processor contains multiple DL1Data registers, to hold the array information.

Usage constraints The DL1Data*n* is:

- A read/write register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-14 on page 4-13.

Figure 4-41 shows the DL1Data*n* bit assignments.

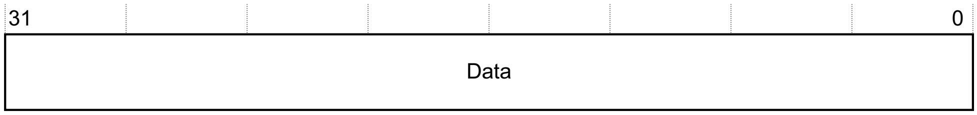


Figure 4-41 DL1Data*n* bit assignments

Table 4-72 shows the DL1Data*n* bit assignments.

Table 4-72 DL1Data*n* bit assignments

Bits	Name	Function
[31:0]	Data	Holds the data side L1 or L2 array information

To access the DL1Data*n*, read or write the CP15 registers with:

```
MRC p15, 0, <Rt>, c15, c1, n; Read Data L1 Data n Register
MCR p15, 0, <Rt>, c15, c1, n; Write Data L1 Data n Register
```

where *n* is 0, 1, 2, or 3 for the Opcode_2 value of DL1Data0, DL1Data1, DL1Data2, or DL1Data3 Register.

4.3.59 RAM Index Register

The RAMINDEX characteristics are:

Purpose Read the instruction side L1 array contents into the IL1Data*n* register or read the data side L1 or L2 array contents into the DL1Data*n* register.

Usage constraints The RAMINDEX is:

- A write-only operation.
- Any write to the RAMINDEX register must be followed by a DSB.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-14 on page 4-13.

Figure 4-42 shows the RAMINDEX bit assignments.

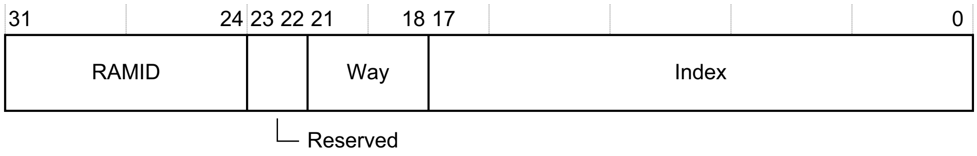


Figure 4-42 RAMINDEX bit assignments

Table 4-73 shows the RAMINDEX bit assignments.

Table 4-73 RAMINDEX bit assignments

Bits	Name	Function
[31:24]	RAMID	RAM Identifier. This field indicates which RAM is being accessed: `0x00` L1-I tag RAM. `0x01` L1-I data RAM. `0x02` L1-I BTB RAM. `0x03` L1-I GHB RAM. `0x04` L1-I TLB array. `0x05` L1-I indirect predictor RAM. `0x08` L1-D tag RAM. `0x09` L1-D data RAM. `0x0A` L1-D load-TLB array. `0x0B` L1-D store-TLB array. `0x10` L2 tag RAM. `0x11` L2 data RAM. `0x12` L2 snoop tag RAM. `0x13` L2 data ECC RAM. `0x14` L2 dirty RAM. `0x18` L2 TLB RAM.
[23:22]	-	Reserved, RAZ/WI.
[21:18]	Way	Indicates the way of the RAM that is being accessed.
[17:0]	Index	Indicates the index address of the RAM that is being accessed.

Note

- In Non-secure PL1 and PL2 modes, the RAMINDEX operation returns the contents of the RAM only if the entry is marked valid and Non-secure. Entries that are marked invalid or Secure update the IL1Data*n* or DL1Data*n* registers with `0x0` values.
- In Secure PL1 modes, the RAMINDEX operation returns the contents of the RAM, regardless of whether the entry is marked valid or invalid, and Secure or Non-secure.

Note

- The L1-I, L1-D, L2 TLB, and L2 snoop tag RAMs can only be accessed by the processor where the RAM resides or that owns the RAM.
- The L2 tag, data, data ECC, and dirty RAMs can be accessed by any processor.

Figure 4-43 shows the RAMINDEX bit assignments for accessing L1-I tag RAM.

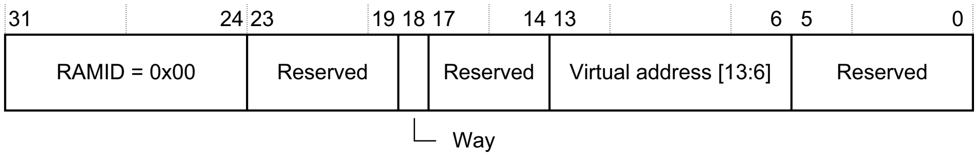


Figure 4-43 RAMINDEX bit assignments for L1-I tag RAM

The RAMINDEX address bits for accessing L1-I tag RAM are:

Way Way select.

VA[13:7] Row select.

VA[6] Bank select.

The data returned from accessing L1-I tag RAM are:

IL1DATA2 32'b0.

IL1DATA1 32'b0.

IL1DATA0 Tag data[31:0].

Figure 4-44 shows the RAMINDEX bit assignments for accessing L1-I data RAM.

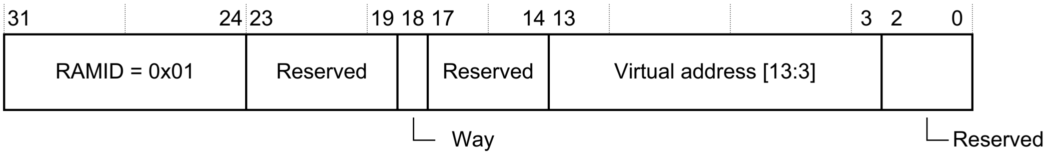


Figure 4-44 RAMINDEX bit assignments for L1-I data RAM

The RAMINDEX address bits for accessing L1-I data RAM are:

Way Way select.

VA[13:6] Set select.

VA[5:4] Bank select.

VA[3] Upper or lower doubleword within the quadword.

The data returned from accessing L1-I data RAM are:

IL1DATA2 Parity and auxiliary information.

IL1DATA1 Word1[31:0].

IL1DATA0 Word0[31:0].

Figure 4-45 shows the RAMINDEX bit assignments for accessing L1-I BTB RAM.

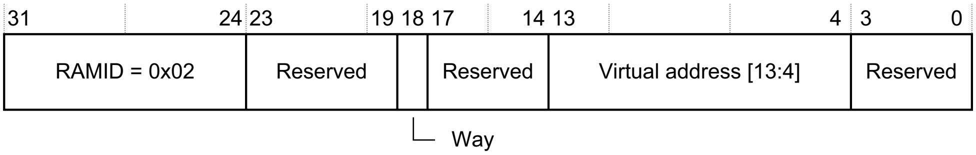


Figure 4-45 RAMINDEX bit assignments for L1-I BTB RAM

The RAMINDEX address bits for accessing L1-I BTB RAM are:

Way Way select.

VA[13:5] Row select.

VA[4] Bank select.

The data returned from accessing L1-I BTB RAM are:

IL1DATA2 BTB data[77:64].

IL1DATA1 BTB data[63:32].

IL1DATA0 BTB data[31:0].

Figure 4-46 shows the RAMINDEX bit assignments for accessing L1-I GHB RAM.

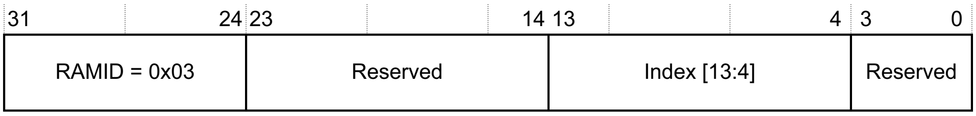


Figure 4-46 RAMINDEX bit assignments for L1-I GHB RAM

The RAMINDEX address bits for accessing L1-I GHB RAM are:

Index[13:5] Row select.

Index[4] Bank select.

The data returned from accessing L1-I GHB RAM are:

IL1DATA2 32'b0.

IL1DATA1 GHB data[47:32].

IL1DATA0 GHB data[31:0].

Figure 4-47 shows the RAMINDEX bit assignments for accessing L1-I TLB array.

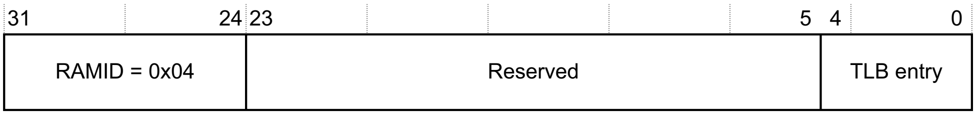


Figure 4-47 RAMINDEX bit assignments for L1-I TLB array

The RAMINDEX address bits for accessing L1-I TLB array are:

TLB entry Selects one of the 32 entries.

The data returned from accessing L1-I TLB array are:

IL1DATA2 TLB entry data[95:64].

IL1DATA1 TLB entry data[63:32].

IL1DATA0 TLB entry data[31:0].

Figure 4-48 shows the RAMINDEX bit assignments for accessing L1-I indirect predictor RAM.

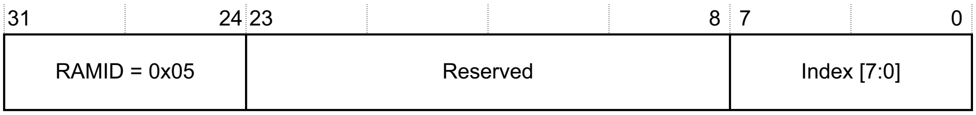


Figure 4-48 RAMINDEX bit assignments for L1-I indirect predictor RAM

The RAMINDEX address bits for accessing L1-I indirect predictor RAM are:

Index[7:0] Indirect predictor entry.

The data returned from accessing L1-I indirect predictor RAM are:

IL1DATA2 32'b0.

IL1DATA1 32'b0.

IL1DATA0 Indirect predictor data[31:0].

Figure 4-49 shows the RAMINDEX bit assignments for accessing L1-D tag RAM.

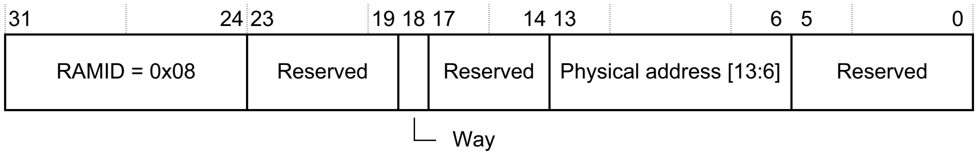


Figure 4-49 RAMINDEX bit assignments for L1-D tag RAM

The RAMINDEX address bits for accessing L1-D tag RAM are:

Way Way select.

PA[13:8] Row select.

PA[7:6] Bank select.

The data returned from accessing L1-D tag RAM are:

DL1DATA3 32'b0.

DL1DATA2 32'b0.

DL1DATA1 Tag ECC[6:0].

DL1DATA0 Tag data[28:0].

Figure 4-50 shows the RAMINDEX bit assignments for accessing L1-D data RAM.

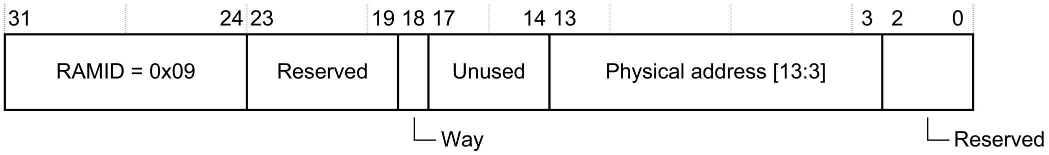


Figure 4-50 RAMINDEX bit assignments for L1-D data RAM

The RAMINDEX address bits for accessing L1-D data RAM are:

Way Way select.

PA[13:6] Set select.

PA[5:4] Bank select.

PA[3] Upper or lower doubleword within the quadword.

The data returned from accessing L1-D data RAM are:

DL1DATA3 Word1 ECC[6:0].

DL1DATA2 Word0 ECC[6:0].

DL1DATA1 Word1 data[31:0].

DL1DATA0 Word0 data[31:0].

Figure 4-51 shows the RAMINDEX bit assignments for accessing L1-D load TLB array.

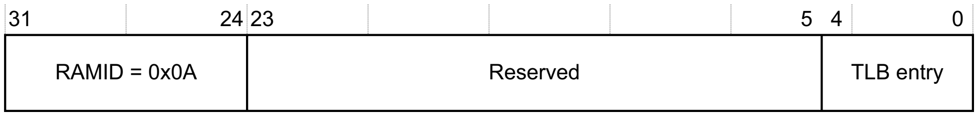


Figure 4-51 RAMINDEX bit assignments for L1-D load TLB array

The RAMINDEX address bits for accessing L1-D load TLB array are:

TLB entry Selects one of the 32 entries.

The data returned from accessing L1-D load TLB array are:

DL1DATA3 TLB entry data[101:96].

DL1DATA2 TLB entry data[95:64].

DL1DATA1 TLB entry data[63:32].

DL1DATA0 TLB entry data[31:0].

Figure 4-52 shows the RAMINDEX bit assignments for accessing L1-D store TLB array.

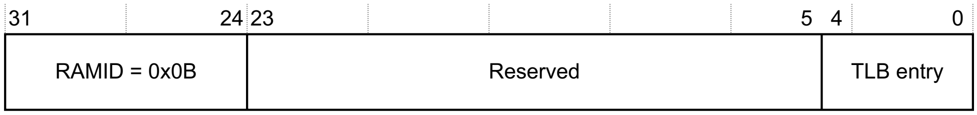


Figure 4-52 RAMINDEX bit assignments for L1-D store TLB array

The RAMINDEX address bits for accessing L1-D store TLB array are:

TLB entry Selects one of the 32 entries.

The data returned from accessing L1-D store TLB array are:

DL1DATA3 TLB entry data[101:96].

DL1DATA2 TLB entry data[95:64].

DL1DATA1 TLB entry data[63:32].

DL1DATA0 TLB entry data[31:0].

Figure 4-53 on page 4-97 shows the RAMINDEX bit assignments for accessing L2 tag RAM.

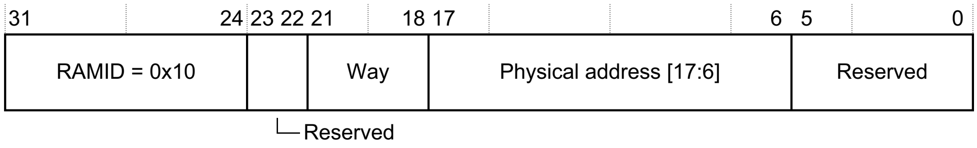


Figure 4-53 RAMINDEX bit assignments for L2 tag RAM

The RAMINDEX address bits for accessing L2 tag RAM are:

Way[3:0] Way select.

PA[17:8] Row select.

PA[7:6] Tag bank select.

The data returned from accessing L2 tag RAM are:

DL1DATA3 32'b0.

DL1DATA2 32'b0.

DL1DATA1 Tag ECC[6:0].

DL1DATA0 Tag data[29:0].

Figure 4-54 shows the RAMINDEX bit assignments for accessing L2 data RAM.

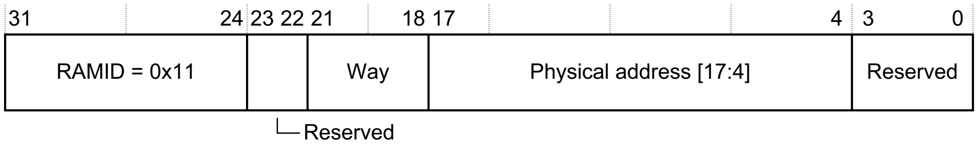


Figure 4-54 RAMINDEX bit assignments for L2 data RAM

The RAMINDEX address bits for accessing L2 data RAM are:

Way[3:0] Way select.

PA[17:8] Row select.

PA[7:6] Tag bank select.

PA[5:4] Data bank select.

The data returned from accessing L2 data RAM are:

DL1DATA3 Data[127:96].

DL1DATA2 Data[95:64].

DL1DATA1 Data[63:32].

DL1DATA0 Data[31:0].

Figure 4-55 on page 4-98 shows the RAMINDEX bit assignments for accessing L2 snoop tag RAM.

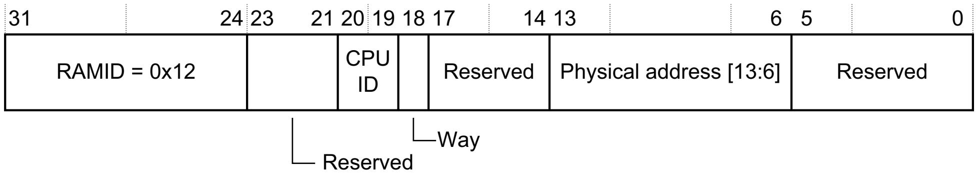


Figure 4-55 RAMINDEX bit assignments for L2 snoop tag RAM

The RAMINDEX address bits for accessing L2 snoop tag RAM are:

CPUID[1:0] Processor ID of the executing processor that has access to the L2 snoop tag RAM.

Way Way select.

PA[13:8] Row select.

PA[7:6] Bank select.

The data returned from accessing L2 snoop tag RAM are:

DL1DATA3 32'b0.

DL1DATA2 32'b0.

DL1DATA1 Tag ECC[6:0].

DL1DATA0 Tag data[28:0].

Figure 4-56 shows the RAMINDEX bit assignments for accessing L2 data ECC RAM.

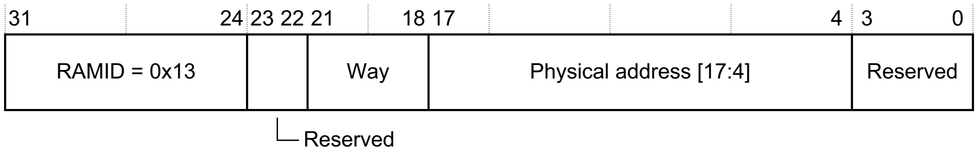


Figure 4-56 RAMINDEX bit assignments for L2 data ECC RAM

The RAMINDEX address bits for accessing L2 data ECC RAM are:

Way[3:0] Way select.

PA[17:8] Row select.

PA[7:6] Tag bank select.

PA[5:4] Data bank select.

The data returned from accessing L2 data ECC RAM are:

DL1DATA3 32'b0.

DL1DATA2 32'b0.

DL1DATA1 32'b0.

DL1DATA0 Data ECC[15:0].

Figure 4-57 on page 4-99 shows the RAMINDEX bit assignments for accessing L2 dirty RAM.

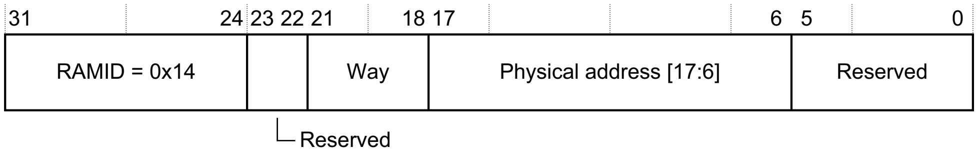


Figure 4-57 RAMINDEX bit assignments for L2 dirty RAM

The RAMINDEX address bits for accessing L2 dirty RAM are:

Way[3:0] Way select.

PA[17:8] Row select.

PA[7:6] Tag bank select.

The data returned from accessing L2 dirty RAM are:

DL1DATA3 32'b0.

DL1DATA2 32'b0.

DL1DATA1 32'b0.

DL1DATA0 Dirty data[17:0].

Figure 4-58 shows the RAMINDEX bit assignments for accessing L2 TLB RAM.

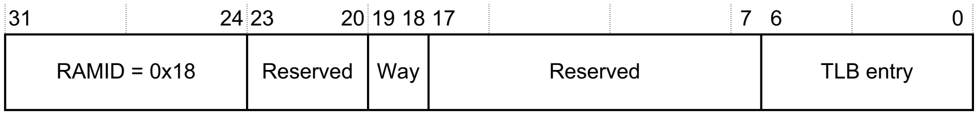


Figure 4-58 RAMINDEX bit assignments for L2 TLB RAM

The RAMINDEX address bits for accessing L2 TLB RAM are:

Way Way select.

TLB entry Selects one of the 128 entries in each way.

The data returned from accessing L2 TLB RAM are:

DL1DATA3 TLB entry data[99:96].

DL1DATA2 TLB entry data[95:64].

DL1DATA1 TLB entry data[63:32].

DL1DATA0 TLB entry data[31:0].

For example, to read one entry in the instruction side L1 data array:

```
LDR R0, =0x01000D80;
MCR p15, 0, R0, c15, c4, 0; Read I-L1 TLB data into IL1Data0-2
DSB
ISB
MRC p15, 0, R1, c15, c0, 0; Move IL1Data0 Register to R1
MRC p15, 0, R2, c15, c0, 1; Move IL1Data1 Register to R2
MRC p15, 0, R3, c15, c0, 2; Move IL1Data2 Register to R3
```

To access the RAMINDEX, write the CP15 register with:

```
MCR p15, 0, <Rt>, c15, c4, 0; Write RAM Index Register
```

4.3.60 L2 Auxiliary Control Register

The L2ACTLR characteristics are:

Purpose Provides configuration and control options for the L2 memory system.

Usage constraints The L2ACTLR:

- Is a read/write register.
- Is Common to the Secure and Non-secure states.
- Is only accessible from PL1 or higher, with access rights that depend on the mode:
  - Read/write in Secure PL1 modes.
  - Read-only and write-ignored in Non-secure PL1 and PL2 modes.
- This register can only be written 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 ACE 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 processor 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-A15 MPCore processor.
  3. Ensure that the system has no outstanding ACP requests to the Cortex-A15 MPCore 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 Available in all configurations.

Attributes See the register summary in Table 4-14 on page 4-13.

Figure 4-59 on page 4-101 shows the L2ACTLR bit assignments.

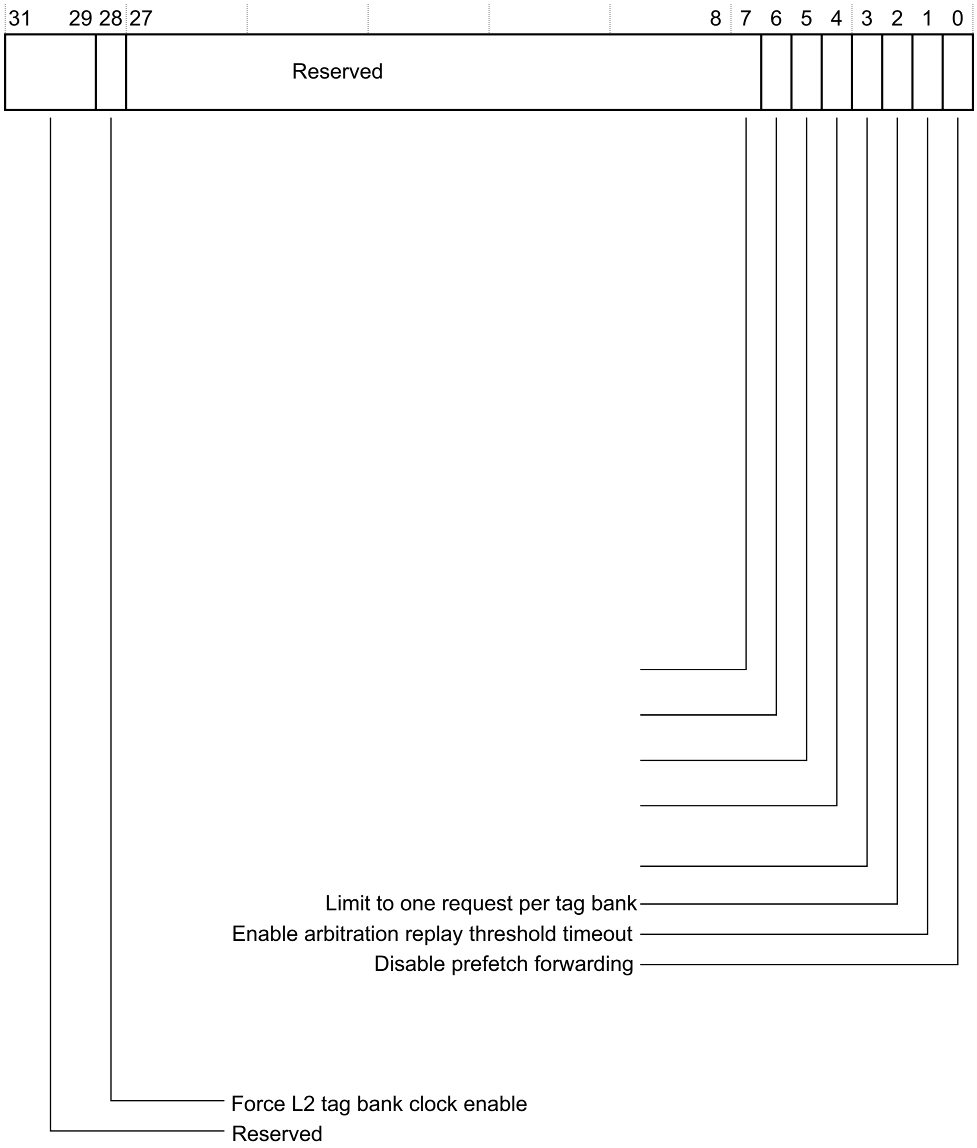

31
29 28 27
8 7 6 5 4 3 2 1 0
Reserved
Limit to one request per tag bank
Enable arbitration replay threshold timeout
Disable prefetch forwarding
Force L2 tag bank clock enable
Reserved

Figure 4-59 L2ACTLR bit assignments

Table 4-74 shows the L2ACTLR bit assignments.

Table 4-74 L2ACTLR bit assignments (continued)

Bits	Name	Function
[26][a]	Enable L2, GIC, and Timer regional clock gates	Enables L2, GIC, and Timer regional clock gates: **0** Disables the L2, GIC, and Timer regional clock gates. This is the reset value. **1** Enables the L2, GIC, and Timer regional clock gates for additional clock gating. When this bit is set, the regional clock gates can gate-off the clock and potentially reduce dynamic power dissipation.
[25]	Enable replay threshold single issue - all tag banks	Enables replay threshold single issue for all tag banks when L2 arbitration replay threshold is reached: **0** Disables single issue across tag banks when L2 arbitration replay threshold is reached. This is the reset value. **1** Enables single issue across tag banks when L2 arbitration replay threshold is reached.
[24:17]	-	Reserved, RAZ/WI.
[16][a]	Enable replay threshold single issue - current tag bank	Enables replay threshold single issue for current tag bank: **0** Disables replay threshold single issue. This is the reset value. **1** Enables replay threshold single issue. If 32 consecutive transactions on a tag bank replay, then single issue is forced until a transaction successfully passes hazard checking.
[15]	Enable CPU WFI retention mode[a]	Enables CPU WFI retention mode: **0** Disables CPU WFI retention mode. This is the reset value. **1** Enables CPU WFI retention mode.
[14]	Enable UniqueClean evictions with data[a]	Enables UniqueClean evictions with data: **0** Disables UniqueClean evictions with data. This is the reset value. **1** Enables UniqueClean evictions with data.
[13]	Disable SharedClean data transfers[a]	Disables SharedClean data transfers: **0** Enables SharedClean data transfers. This is the reset value. **1** Disables SharedClean data transfers.
[12]	Disable multiple outstanding WriteClean/WriteBack/Evicts using the same AWID[a]	Disables multiple outstanding WriteClean/WriteBack/Evicts using the same AWID: **0** Enables multiple outstanding WriteClean/WriteBack/Evicts using the same AWID. This is the reset value. **1** Disables multiple outstanding WriteClean/WriteBack/Evicts using the same AWID.
[11]	Disable *Data Synchronization Barrier* (DSB) with no *Distributed Virtual Memory* (DVM) synchronization[a]	Disables DSB with no DVM synchronization: **0** Enables DSB with no DVM synchronization. This is the reset value. **1** Disables DSB with no DVM synchronization.
[10]	Disable Non-secure debug array read	Disables Non-secure debug array read: **0** Enables Non-secure debug array read access to Non-secure memory. This is the reset value. **1** Disables Non-secure debug array read access.
[9]	-	Reserved, RAZ/WI.

Table 4-74 L2ACTLR bit assignments (continued)

Bits	Name	Function
[8]	Disable DVM/CMO message broadcast	Disables *Distributed Virtual Memory* (DVM) transactions and cache maintenance operation message broadcast: **0** Enables DVM and cache maintenance operation message broadcast. This is the reset value. **1** Disables DVM and cache maintenance operation message broadcast.
[7]	Enable hazard detect timeout	Enables hazard detect timeout: **0** Disables hazard detect timeout. This is the reset value. **1** Enables hazard detect timeout.
[6]	Disable shareable transactions from master	Disables shareable transactions from master: **0** Enables shareable transactions from master. This is the reset value. **1** Disables shareable transactions from master.
[5]	-	Reserved, RAZ/WI.
[4]	Disable WriteUnique and WriteLineUnique transactions from master	Disables WriteUnique and WriteLineUnique transactions from master: **0** Enables WriteUnique and WriteLineUnique transactions from master. This is the reset value. **1** Disables WriteUnique and WriteLineUnique transactions from master.
[3]	Disable clean/evict push to external	Disables clean/evict push to external: **0** Enables clean/evict to be pushed out to external. This is the reset value. **1** Disables clean/evict from being pushed to external.
[2]	Limit to one request per tag bank	Limit to one request per tag bank: **0** Normal behavior of permitting parallel requests to the tag banks. This is the reset value. **1** Limits to one request per tag bank.
[1]	Enable arbitration replay threshold timeout	Enable arbitration replay threshold timeout: **0** Disables arbitration replay threshold timeout. This is the reset value. **1** Enables arbitration replay threshold timeout.
[0]	Disable prefetch forwarding	Disables prefetch forwarding: **0** Enables prefetch forwarding. This is the reset value. **1** Disables prefetch forwarding.

a. This bit is not available in revisions prior to r3p0.

To access the L2ACTLR, read or write the CP15 register with:

```
MRC p15, 1, <Rt>, c15, c0, 0; Read L2 Auxiliary Control Register
MCR p15, 1, <Rt>, c15, c0, 0; Write L2 Auxiliary Control Register
```

4.3.61 L2 Prefetch Control Register

The L2PFR characteristics are:

Purpose Provides control options for the L2 automatic hardware prefetcher.

Usage constraints The L2PFR:

- Is a read/write register.
- Is Common to the Secure and Non-secure states.

- Is only accessible from PL1 or higher, with access rights that depend on the mode:
  - — Read/write in Secure PL1 modes.
  - — Read-only and write-ignored in Non-secure PL1 and PL2 modes.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-14 on page 4-13.

Figure 4-60 shows the L2PFR bit assignments.

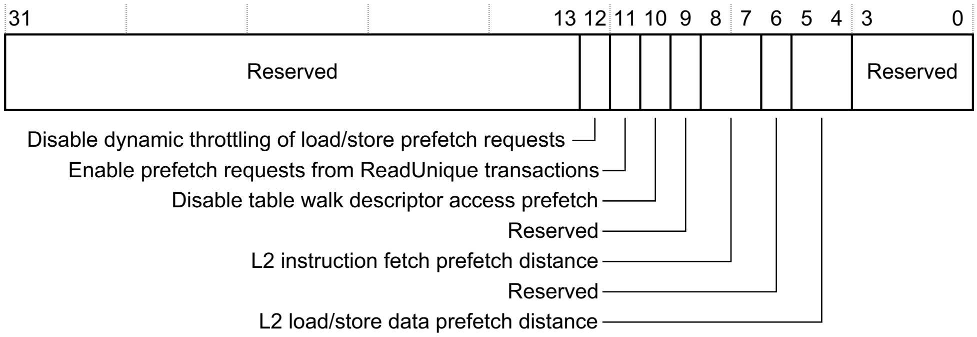

31
13 12 11 10 9 8 7 6 5 4 3 0
Reserved
Reserved
Disable dynamic throttling of load/store prefetch requests
Enable prefetch requests from ReadUnique transactions
Disable table walk descriptor access prefetch
Reserved
L2 instruction fetch prefetch distance
Reserved
L2 load/store data prefetch distance

Figure 4-60 L2PFR bit assignments

Table 4-75 shows the L2PFR bit assignments.

Table 4-75 L2 Prefetch Control Register bit assignments

Bits	Name	Function
[31:13]	-	Reserved, RAZ/WI.
[12]	Disable dynamic throttling of load/store prefetch requests	Disable dynamic throttling of load/store prefetch requests: **0** Enables dynamic throttling of load/store prefetch requests This is the reset value. **1** Disables dynamic throttling of load/store prefetch requests.
[11]	Enable prefetch requests from ReadUnique transactions	Enable prefetch requests from ReadUnique transactions: **0** Disables prefetch requests from being generated by ReadUnique transactions. **1** Enables prefetch requests to be generated by ReadUnique transactions. This is the reset value.
[10]	Disable table walk descriptor access prefetch	Disables table walk descriptor access prefetch: **0** Enables table walk descriptor access prefetch. This is the reset value. **1** Disables table walk descriptor access prefetch.
[9]	-	Reserved, RAZ/WI.
[8:7]	L2 instruction fetch prefetch distance	Indicates the L2 instruction fetch prefetch distance: `b00` 0 cache line, disables instruction prefetch. `b01` 1 cache line. `b10` 2 cache lines. `b11` 3 cache lines. This is the reset value.

Table 4-75 L2 Prefetch Control Register bit assignments (continued)

Bits	Name	Function
[6]	-	Reserved, RAZ/WI.
[5:4]	L2 load/store data prefetch distance	L2 load/store data prefetch distance: `b00` 0 cache line. `b01` 2 cache lines. `b10` 4 cache lines. `b11` 8 cache lines. This is the reset value.
[3:0]	-	Reserved, RAZ/WI.

To access the L2PFR, read or write the CP15 register with:

```
MRC p15, 1, <Rt>, c15, c0, 3; Read L2 Prefetch Control Register
MCR p15, 1, <Rt>, c15, c0, 3; Write L2 Prefetch Control Register
```

Note

Setting the value of this register to `0x400` disables all hardware prefetch.

4.3.62 Auxiliary Control Register 2

Note

This register is not available in revisions prior to r3p0.

The ACTLR2 characteristics are:

Purpose Provides configuration and control options for the processor.

Usage constraints The ACTLR2:

- Is a read/write register.
- Is Common to the Secure and Non-secure states.
- Is only accessible from PL1 or higher, with access rights that depend on the mode:
  - Read/write in Secure PL1 modes.
  - Read-only and write-ignored in Non-secure PL1 and PL2 modes.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-14 on page 4-13.

Figure 4-61 shows the ACTLR2 bit assignments.

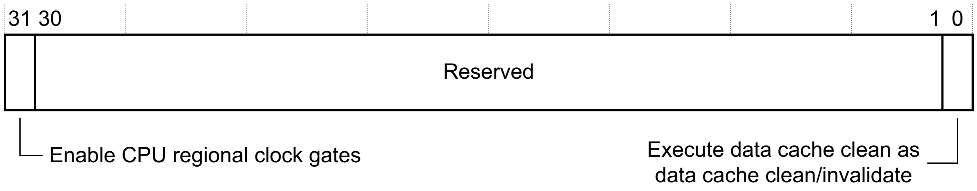

31 30
1 0
Reserved
Enable CPU regional clock gates
Execute data cache clean as data cache clean/invalidate

Figure 4-61 ACTLR2 bit assignments

Table 4-76 shows the ACTLR2 bit assignments.

Table 4-76 Auxiliary Control Register 2 bit assignments

Bits	Name	Function
[31]	Enable CPU regional clock gates	Enable CPU regional clock gates: **0** Disables the CPU regional clock gates. When this bit is cleared, the regional clock gate enable is tied HIGH. This is the reset value. **1** Enables the CPU regional clock gates for additional clock gating. When this bit is set, the regional clock gates can gate-off the clock and potentially reduce dynamic power dissipation.
[30:1]	-	Reserved.
[0]	Execute data cache clean as data cache clean/invalidate	Execute data cache clean as data cache clean and invalidate: **0** Normal behavior, executes data cache clean as data cache clean. This is the reset value. **1** Executes data cache clean as data cache clean and invalidate.

To access the ACTLR2, read or write the CP15 register with:

```
MRC p15, 1, <Rt>, c15, c0, 4; Read Auxiliary Control Register 2
MCR p15, 1, <Rt>, c15, c0, 4; Write Auxiliary Control Register 2
```

4.3.63 Configuration Base Address Register

The CBAR characteristics are:

Purpose Holds the physical base address of the memory-mapped Interrupt Controller registers.

Usage constraints The CBAR is:
- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-14 on page 4-13.

Figure 4-62 shows the CBAR bit assignments.

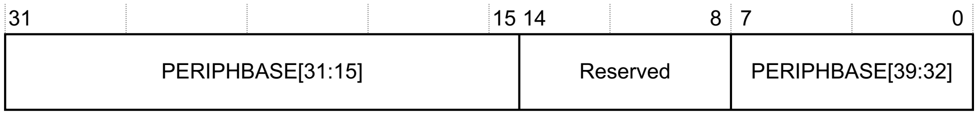


Figure 4-62 CBAR bit assignments

Table 4-77 shows the CBAR bit assignments.

Table 4-77 CBAR bit assignments

Bits	Name	Function
[31:15]	PERIPHBASE[31:15]	The primary input **PERIPHBASE[31:15**] determines the reset value.
[11:8]	-	Reserved, UNK/SBZP.
[7:0]	PERIPHBASE[39:32]	The primary input **PERIPHBASE[39:32]** determines the reset value.

To access the CBAR, read the CP15 register with:

```
MRC p15, 4, <Rt>, c15, c0, 0; Read Configuration Base Address Register
```

4.3.64 CPU Memory Error Syndrome Register

The CPUMERRSR characteristics are:

Purpose Holds the number of memory errors that have occurred in the following L1 and L2 RAMs:

- L1-I tag RAM.
- L1-I data RAM.
- L1-I BTB RAM.
- L1-D tag RAM.
- L1-D data RAM.
- L2 TLB RAM.

Usage constraints The CPUMERRSR:

- Is a 64-bit read/write register.
- Is Common to the Secure and Non-secure states.
- Is only accessible from PL1 or higher.
- A write of any value to the register updates the register to `0x0`.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-15 on page 4-13.

Figure 4-63 shows the CPUMERRSR bit assignments.

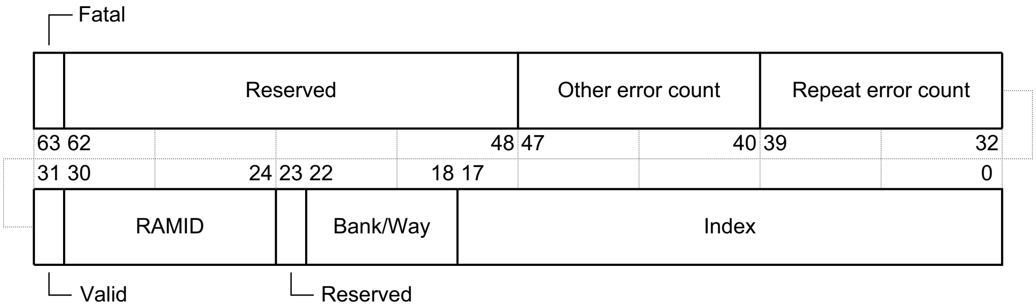


Figure 4-63 CPUMERRSR bit assignments

Table 4-78 shows the CPUMERRSR bit assignments.

Table 4-78 CPUMERRSR bit assignments

Bits	Name	Function
[63]	Fatal	Fatal bit. This bit is set to 1 on the first memory error that caused a Data Abort. It is a sticky bit so that after it is set, it remains set until the register is written. The reset value is 0.
[62:48]	-	Reserved, RAZ/WI.
[47:40]	Other error count	This field is set to 0 on the first memory error and is incremented on any memory error that does not match the RAMID, bank, way, or index information in this register while the sticky Valid bit is set. The reset value is 0.
[39:32]	Repeat error count	This field is set to 0 on the first memory error and is incremented on any memory error that exactly matches the RAMID, bank, way or index information in this register while the sticky Valid bit is set. The reset value is 0.
[31]	Valid	Valid bit. This bit is set to 1 on the first memory error. It is a sticky bit so that after it is set, it remains set until the register is written. The reset value is 0.
[30:24]	RAMID	RAM Identifier. Indicates the RAM, the first memory error occurred in: `0x00` L1-I tag RAM. `0x01` L1-I data RAM. `0x02` L1-I BTB RAM. `0x08` L1-D tag RAM. `0x09` L1-D data RAM. `0x18` L2 TLB RAM.
[23]	-	Reserved, RAZ/WI.
[22:18]	Bank/Way	Indicates the bank or way of the RAM where the first memory error occurred.
[17:0]	Index	Indicates the index address of the first memory error.

Note

- If two or more memory errors in the same RAM occur in the same cycle, only one error is reported.
- If two or more first memory error events from different RAMs occur in the same cycle, one of the errors is selected arbitrarily, while the Other error count field is only incremented 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 only incremented by one.

To access the CPUMERRSR, read or write the CP15 register with:

```
MRRC p15, 0, <Rt>, <Rt2>, c15; Read CPU Memory Error Syndrome Register
MCCR p15, 0, <Rt>, <Rt2>, c15; Write CPU Memory Error Syndrome Register
```

4.3.65 L2 Memory Error Syndrome Register

The L2MERRSR characteristics are:

Purpose Holds the number of memory errors that have occurred in the following L2 RAMs:

- L2 tag RAM.
- L2 data RAM.
- L2 snoop tag RAM.
- L2 dirty RAM.

Usage constraints The L2MERRSR:

- Is a 64-bit read/write register.
- Is Common to the Secure and Non-secure states.
- Is only accessible from PL1 or higher.
- A write of any value to the register updates the register to 0.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-15 on page 4-13.

Figure 4-64 shows the L2MERRSR bit assignments.

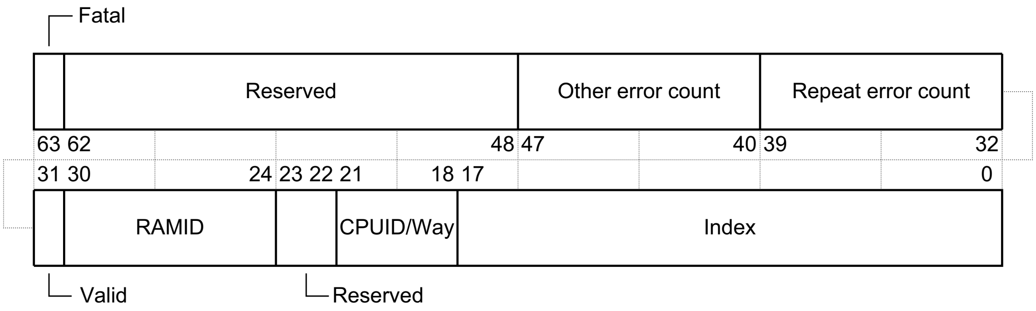


Figure 4-64 L2MERRSR bit assignments

Table 4-79 shows the L2MERRSR bit assignments.

Table 4-79 L2MERRSR bit assignments

Bits	Name	Function
[63]	Fatal	Fatal bit. This bit is set to 1 on the first memory error that caused a Data Abort. It is a sticky bit so that after it is set, it remains set until the register is written. The reset value is 0.
[62:48]	-	Reserved, RAZ/WI.
[47:40]	Other error count	This field is set to 0 on the first memory error and is incremented on any memory error that does not match the RAMID, bank, way, or index information in this register while the sticky Valid bit is set. The reset value is 0.
[39:32]	Repeat error count	This field is set to 0 on the first memory error and is incremented on any memory error that exactly matches the RAMID, bank, way or index information in this register while the sticky Valid bit is set. The reset value is 0.

Table 4-79 L2MERRSR bit assignments (continued)

Bits	Name	Function
[31]	Valid	Valid bit. This bit is set to 1 on the first memory error. It is a sticky bit so that after it is set, it remains set until the register is written. The reset value is 0.
[30:24]	RAMID	RAM Identifier. Indicates the RAM where the first memory error occurred: `0x10` L2 tag RAM. `0x11` L2 data RAM. `0x12` L2 snoop tag RAM. `0x14` L2 dirty RAM.
[23:22]	-	Reserved, RAZ/WI.
[21:18]	CPUID/Way	Indicates which processor and way of the RAM where the first memory error occurred. For L2 tag, data, and dirty RAMs, bits[21:18] indicate one of 16 ways, from way 0 to way 15. `b0000` CPU0 tag, way 0. `b0001` CPU0 tag, way 1. `b0010` CPU1 tag, way 0. `b0011` CPU1 tag, way 1. `b0100` CPU2 tag, way 0. `b0101` CPU2 tag, way 1. `b0110` CPU3 tag, way 0. `b0111` CPU3 tag, way 1. For L2 snoop tag RAM: • Bits[20:19] indicate which processor of the L1 tag RAM. • Bit[18] indicates which way of the tag RAM.
[17:0]	Index	Indicates the index address of the first memory error.

Note

- If two or more memory errors in the same RAM occur in the same cycle, only one error is reported.
- If two or more first memory error events from different RAMs occur in the same cycle, one of the errors is selected arbitrarily, while the Other error count field is only incremented 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 only incremented by one.

To access the L2MERRSR, read or write the CP15 register with:

```
MRRC p15, 1, <Rt>, <Rt2>, c15; Read L2 Memory Error Syndrome Register
MCCR p15, 1, <Rt>, <Rt2>, c15; Write L2 Memory Error Syndrome Register
```

Chapter 5
Memory Management Unit

This chapter describes the *Memory Management Unit* (MMU). It contains the following sections:

5.1 About the MMU

The Cortex-A15 MPCore processor implements the Extended VMSAv7 MMU. The MMU supports:

- ARMv7-A *Virtual Memory System Architecture* (VMSA).
- Security Extensions.
- *Large Physical Address Extensions* (LPAE).
- Virtualization Extensions.

The Extended VMSAv7 MMU controls address translation, access permissions, and memory attributes determination and checking, for memory accesses.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for a full architectural description of the Extended VMSAv7.

Note

The Cortex-A15 MPCore processor does not support the Transient attribute in the LPAE. Use of this attribute results in UNPREDICTABLE behavior.

The MMU controls table walk hardware that accesses translation tables in memory. The MMU works with the L1 and L2 memory system to translate virtual addresses to physical addresses. The MMU enables fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes held in the L1 and L2 *Translation Look-aside Buffers* (TLBs).

The Cortex-A15 MMU features include the following:

- 32-entry fully-associative L1 instruction TLB.
- Two separate 32-entry fully associative L1 TLBs for data load and store pipelines.
- 4-way set-associative 512-entry L2 TLB in each processor.
- Intermediate table walk caches.
- The TLB entries contain a global indicator or an *Address Space Identifier* (ASID) to permit context switches without TLB flushes.
- The TLB entries contain a *Virtual Machine Identifier* (VMID) to permit virtual machine switches without TLB flushes.

5.2 TLB organization

The Cortex-A15 MPCore processor implements a 2-level TLB structure. The TLBs, at either the L1 or the L2 level, do not require to be flushed on a context or virtual machine switch. The Cortex-A15 MMU does not support the locking of TLB entries at either Level 1 or Level 2.

This section describes the TLB organization in:

- *L1 instruction TLB*.
- *L1 data TLB*.
- *L2 TLB*.

5.2.1 L1 instruction TLB

The L1 instruction TLB is a 32-entry fully-associative structure. This TLB caches entries at the 4KB granularity of *Virtual Address* (VA) to *Physical Address* (PA) mapping only. If the page tables map the memory region to a larger granularity than 4K, it only allocates one mapping for the particular 4K region to which the current access corresponds.

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

5.2.2 L1 data TLB

There are two separate 32-entry fully-associative TLBs that are used for data loads and stores, respectively. Similar to the L1 instruction TLB, both of these cache entries at the 4KB granularity of VA to PA mappings only.

At implementation time, the Cortex-A15 MPCore processor can be configured with the `-l1tlb_1m` option, to have the L1 data TLB cache entries at both the 4KB and 1MB granularity. With this configuration, any translation that results in a 1MB or larger page is cached in the L1 data TLB as a 1MB entry. Any translation that results in a page smaller than 1MB is cached in the L1 data TLB as a 4KB entry. By default, all translations are cached in the L1 data TLB as a 4KB entry.

A hit in the data load or store TLBs provides a single **CLK** cycle access to the translation, and returns the physical address to the instruction cache for comparison. It also checks the access permissions to signal a Data Abort.

5.2.3 L2 TLB

Misses from the L1 instruction and data TLBs are handled by a unified L2 TLB. This is a 512-entry 4-way set-associative structure. The L2 TLB supports all the VMSAv7 page sizes of 4K, 64K, 1MB and 16MB in addition to the LPAE page sizes of 2MB and 1GB.

Accesses to the L2 TLB take a variable number of cycles, based on the competing requests from each of the L1 TLBs, TLB maintenance operations in flight, and the different page size mappings in use.

5.3 TLB match process

The Virtualization Extensions and the Security Extensions provide for multiple virtual address spaces that are translated differently. The TLB entries store all the required context information to facilitate a match and avoid the requirement for a TLB flush on a context or virtual machine switch. Each TLB entry contains a virtual address, page size, physical address, and a set of memory properties that include the memory type and access permissions. Each entry is marked as being associated with a particular ASID, or as global for all application spaces. The TLB entry also contains a field to store the VMID that brought in the entry, applicable to accesses made from the Non-secure state, as defined by the Virtualization Extensions. There is also a bit that records whether that TLB entry is allocated on a Hyp mode request. A TLB entry match occurs when the following conditions are met:

- Its virtual address, moderated by the page size such as the virtual address bits[31:N], where N is log2 of the page size for that translation stored in the TLB entry, matches that of the requested address.
- The Non-secure TLB ID, NSTID, matches the Secure or Non-secure state of the requests.
- The Hyp mode bit matches whether the request was made from Hyp mode.
- The ASID matches the current ASID held in the CONTEXTIDR, TTBR0, or TTBR1 register or the entry is marked global.
- The VMID matches the current VMID held in the VTTBR register.

Note

- For a request originating from Hyp mode, the ASID and VMID match are ignored.
- For a request originating from Secure state, the VMID match is ignored.

5.4 Memory access sequence

When the processor generates a memory access, the MMU:

1. Performs a lookup for the requested virtual address, current ASID, current VMID, and security state in the relevant L1 instruction or data load/store TLB.
2. Performs a lookup for the requested virtual address, current ASID, current VMID, and security state in the unified L2 TLB if there is a miss in the relevant L1 TLB.
3. Performs a hardware translation table walk if there is a miss in the L2 TLB.

You can configure the MMU to perform hardware translation table walks using either the classic VMSAv7 Short-descriptor translation table format, or the Long-descriptor translation table format specified by the LPAE. This is controlled by programming the *Extended Address Enable* (EAE) bit in the appropriate Secure or Non-secure Translation Table Base Control Register (TTBCR). See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for information on translation table formats.

Note

Translations in Hyp mode and Stage2 translations are always performed with the Long-descriptor translation table format as specified by the LPAE.

You can configure the MMU to perform translation table walks in cacheable regions using:

- The Short-descriptor translation table format by setting the IRGN bits in the Translation Table Base Register 0 (TTBR0) and Translation Table Base Register 1 (TTBR1).
- The Long-descriptor translation table format by setting the IRGN bits in the relevant Translation Table Base Control Register (TTBCR).

If the encoding of the IRGN bits is Write-Back, an L1 data cache lookup is performed and data is read from the data cache. If the encoding of the IRGN bits is Write-Through or Non-Cacheable, an access to external memory is performed.

In the case of an L2 TLB miss, the hardware does a translation table walk if the translation table walk is enabled by:

- The Short-descriptor translation table format by clearing the PD0 or PD1 bit in the TTBCR.
- The Long-descriptor translation table format by clearing the EPD0 or EPD1 bit in the TTBCR.

If translation table walks are disabled, for example, PD0 or EPD0 is set to 1 for TTBR0, or PD1 or EPD1 is set to 1 for TTBR1, the processor returns a Translation fault. If the TLB finds a matching entry, it uses the information in the entry as follows:

- The access permission bits and the domain when using the Short-descriptor translation table format, determine if the access is permitted. If the matching entry does not pass the permission checks, the MMU signals a memory abort. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for a description of access permission bits, abort types and priorities, and for a description of the Instruction Fault Status Register (IFSR) and Data Fault Status Register (DFSR).
- The memory region attributes specified in the TLB entry determine if the access is:
  - Secure or Non-secure.
  - Inner, Outer shareable or not.
  - Normal Memory, Device, or Strongly-ordered.

- The TLB translates the virtual address to a physical address for the memory access.

5.5 MMU enabling and disabling

You can enable or disable the MMU. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

5.6 Intermediate table walk caches

The Cortex-A15 MPCore processor implements dedicated caches that store intermediate levels of translation table entries as part of a table walk. Cached entries are associated with an ASID. The TLB and the intermediate caches must be invalidated using the standard architectural rule that states, if you change a translation table entry at any level of the walk that is associated with a particular ASID, you must invalidate entries that correspond to that ASID and its associated VA range.

Care is required when using the reserved ASID method for context switch. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

Example 5-1 shows how to synchronize ASID and TTBR changes using a reserved ASID.

Example 5-1 Using a reserved ASID to synchronize ASID and TTBR changes

In this example, the operating system uses a particular reserved ASID value for the synchronization of the ASID and the Translation Table Base Register. You can use this approach only when the size of the mapping for any given virtual address is the same in the old and new translation tables. The example uses the value of 0.

The software uses the following sequences that must be executed from memory marked as global:

```
Change ASID to 0
ISB
Change Translation Table Base Register
ISB
Change ASID to new value
ISB
```

If the code relies on only leaf translation table entries that are cached, it can incorrectly assume that entries tagged with the reserved ASID are not required to be flushed. For example:

- Global leaf entries that remain valid or must be flushed for all ASIDs when modified.
- Non-global leaf entries that are not used because the reserved ASID is not set outside the context switch code.

The incorrect assumption leads to the following failure:

- The context switch code sets the ASID to the reserved value.
- Speculative fetching reads and caches the first level page table entry, using the current TTBR, and tagging the entry with the reserved ASID. This is a pointer to a second level table.
- Context switch completes.
- Processing continues, and the process with the page tables terminates. The OS frees and reallocates the page table memory.
- A later context switch sets the ASID to the reserved value
- Speculative fetching makes use of the cached first level page table entry, because it is tagged with the reserved ASID, and uses it to fetch a second level page table entry. Because the memory is reallocated and reused, the entry contains random data that can appear to be a valid, global entry. This second level page table entry is cached.

- Context switch completes, and application execution continues.
- The application references the address range covered by the cached second level page table entry. Because the entry is marked as global, a match occurs and so data is fetched from a random address.

Note

When you use a reserved ASID, you must invalidate the TLB to deallocate the translation table memory.

5.7 External aborts

External memory errors are defined as those that occur in the memory system rather than those that are detected by the MMU. External memory errors are extremely rare. External errors are caused by errors flagged by the AXI interfaces or generated because of an uncorrected ECC error in the L1 data cache or L2 cache arrays when the request is external to the Cortex-A15 MPCore processor. You can configure external aborts to trap to the monitor mode by setting the EA bit in the Secure Configuration Register to 1. See *Secure Configuration Register* on page 4-63 for more information.

This section describes external aborts in:

- *External aborts on data read or write*.
- *Synchronous and asynchronous aborts*.

See *Asynchronous errors* on page 7-11 for more information on external memory errors.

5.7.1 External aborts on data read or write

Externally generated errors during a data read or write can be asynchronous. This means that the r14_abt entry into the abort handler on such an abort might not hold the address of the instruction that caused the abort.

The DFAR is UNPREDICTABLE when an asynchronous abort occurs.

For a load multiple or store multiple operation, the address captured in the DFAR is that of the address that generated the synchronous external abort.

5.7.2 Synchronous and asynchronous aborts

To determine a fault type, read the DFSR for a Data Abort or the IFSR for a Prefetch Abort.

The processor supports an Auxiliary Data Fault Status Register for software compatibility. This register provides additional information about the faults generated on data accesses because of an uncorrected ECC errors. See *Auxiliary Data Fault Status Register* on page 4-81 for more information. The processor defines an Auxiliary Instruction Fault Status register, but this register is not updated.

Chapter 6
Level 1 Memory System

This chapter describes the *Level 1* (L1) memory system. It contains the following sections:

- *About the L1 memory system* on page 6-2.
- *Cache organization* on page 6-3.
- *L1 instruction memory system* on page 6-4.
- *L1 data memory system* on page 6-6.
- *Program flow prediction* on page 6-12.
- *L1 RAM memories* on page 6-15.

6.1 About the L1 memory system

The L1 memory system consists of separate instruction and data caches.

The L1 instruction memory system has the following features:

- 32KB 2-way set-associative instruction cache.
- Fixed line length of 64 bytes.
- Parity protection per 16 bits.
- Instruction cache that is *Physically-Indexed and Physically-Tagged* (PIPT).
- *Least Recently Used* (LRU) cache replacement policy.
- MBIST support.

The L1 data memory system has the following features:

- 32KB 2-way set-associative data cache.
- Fixed line length of 64 bytes.
- ECC protection per 32 bits.
- Data cache that is PIPT.
- Out-of-order, speculative, nonblocking load requests to Normal memory, Strongly ordered, and Device memory.
- LRU cache replacement policy.
- MBIST support.

Note

The Cortex-A15 MPCore processor does not support TLB or cache lockdown.

6.2 Cache organization

You can disable each cache independently. See *System Control Register* on page 4-54. On a cache miss, critical word-first filling of the cache is performed.

If the cache reports a hit on a memory location that is marked as Non-Cacheable, Device, or Strongly-ordered, this is called an unexpected cache hit. In this architecturally unpredictable case, the cache might return incorrect data. Because the caches are physically addressed, improper page table configuration is the only way to create this scenario.

6.3 L1 instruction memory system

The instruction cache can source up to 128 bits per fetch depending on alignment. A single fetch can span a 128-bit aligned region or cache line, but cannot span a page boundary.

Sequential cache read operations reduce the number of full cache reads. This has the benefit of reducing power consumption. If a cache read is sequential to the previous cache read, and the read is within the same cache line, only the data RAM way that was previously read is accessed.

The L1 instruction cache appears to software as a physically tagged, physically indexed array. Therefore, the instruction cache is only required to be flushed when writing new data to an instruction address.

This section describes the L1 instruction memory system in:

- *Instruction cache disabled behavior*.
- *Instruction cache speculative memory accesses*.
- *Fill buffers* on page 6-5.
- *Non-cacheable fetching* on page 6-5.
- *Parity error handling* on page 6-5.
- *Cache line length and heterogeneous systems* on page 6-5.

6.3.1 Instruction cache disabled behavior

The SCTLR.I bit, see *System Control Register* on page 4-54, enables or disables the L1 instruction cache. If the I bit is disabled, fetches cannot access any of the instruction cache arrays. An exception to this rule is the CP15 instruction cache operations. If the instruction cache is disabled, the instruction cache maintenance operations can still execute normally.

6.3.2 Instruction cache speculative memory accesses

An instruction remains in the pipeline between the fetch and the execute stages. Because there can be several unresolved branches in the pipeline, instruction fetches are speculative, meaning there is no guarantee that they are executed. A branch or exceptional instruction in the code stream can cause a pipeline flush, discarding the currently fetched instructions.

Because of the aggressive prefetching behavior, you must not place read-sensitive devices in the same page as code. Pages with Device or Strongly-ordered memory type attributes are treated as Non-Cacheable Normal Memory. You must mark pages that contain read-sg4□.9/TT27(mu)3.9tructfu

6.3.3 Fill buffers

The instruction cache is fed by dual fill buffers that hold instructions returned from the L2 cache on a linefill operation, or instructions from non-cacheable regions. The fill buffers are non-blocking. An instruction cache hit can bypass an in-progress cache miss, even before the critical word is returned.

6.3.4 Non-cacheable fetching

Fetches that occur when the instruction cache is disabled, or from a page with attributes not indicating inner cacheable, do not result in the line entering the instruction cache. Incoming instructions from the L2 cache are stored in the fill buffers until the fetch reaches the end of the cache line or a non-sequential fetch occurs, whichever occurs first. Therefore, multiple repeated fetches from the same address can occur in sequence when a region is not given cacheable attributes.

6.3.5 Parity error handling

The instruction cache implements one parity bit per 16-bits of instruction data. The instruction cache tag array is also protected by one parity bit per tag entry. Parity errors invalidate the offending cache line, and force a refetch from the L2 cache on the next access. No aborts are generated on parity errors that occur within the instruction cache. The location of a parity error is reported in the CPU Memory Error Syndrome Register, see *CPU Memory Error Syndrome Register* on page 4-107. Because the data cache shares this register, there is no guarantee that this register contains the location of the last instruction side parity error.

6.3.6 Cache line length and heterogeneous systems

Systems that are comprised of both the Cortex-A15 MPCore processor and other processors operating under a shareable memory system must consider differences in the cache line length. The Cortex-A15 MPCore processor L1 caches contain 64-byte lines. Other processors, however, can feature caches that support cache line lengths different than those of the Cortex-A15 MPCore processor. System software often requires invalidation of a range of addresses that might be present in multiple processors. This is accomplished with a loop of invalidate cache by MVA CP15 operations that step through the address space in cache line-sized strides. For code to be portable across all ARMv7-A architecture-compliant devices, system software queries the CP15 Cache Type Register to obtain the stride size, see *Cache Type Register* on page 4-28 for more information.

Systems that contain a combination of processors with 64-byte and 32-byte lines must handle data cache side operations within the interconnect, but it is not a requirement that the interconnect handles instruction side operations. The Cortex-A15 MPCore processor contains the **IMINLN** signal, that on reset, sets the value of the IminLine field within the Cache Type Register. You must set this signal 0 or 1, depending on the minimum instruction cache line size in the system. When **IMINLN** is set to 0, the minimum system-wide instruction cache line size is 32 bytes. When **IMINLN** is set to 1, the minimum system-wide instruction cache line size is 64 bytes. This signal is only sampled at reset.

6.4 L1 data memory system

The L1 data memory system executes all memory operations in the Cortex-A15 MPCore processor. In addition, it handles cache maintenance operations, TLB maintenance operations, and exclusive operations using the Load-Exclusive, Store-Exclusive and Clear-Exclusive instructions.

The L1 memory system supports out-of-order execution of instructions. Loads can be executed and return their data while they are still speculative and might be flushed. Stores can be executed, but not committed to memory, while they are still speculative. Speculative loads can forward data from older speculative stores.

The L1 memory system is non-blocking and supports hit-under-miss. For Normal memory, up to six 64-byte cache line requests can be outstanding at a time. While those requests are waiting for memory, loads to different cache lines can hit the cache and return their data, and stores to different cache lines can hit and update the cache.

The L1 data memory system includes the following:

- L1 data cache.
- Address generation logic.
- The L1 TLBs.
- Buffering for stores that have not been written to the cache or memory.
- Fill buffers for processing cache line fills and non-cacheable reads.
- Coherence logic for handling snoop requests.

This section describes L1 data memory system in:

- *Behavior for different memory types*.
- *Coherence* on page 6-8.
- *Cache disabled behavior* on page 6-9.
- *Non-cacheable streaming enhancement* on page 6-9.
- *Synchronization primitives* on page 6-9.
- *LDRT and STRT instructions* on page 6-10.
- *Preload instruction behavior* on page 6-10.
- *Error Correction Code* on page 6-11.

6.4.1 Behavior for different memory types

The L1 data memory system uses memory attributes from the MMU to determine the behavior for memory transactions to regions of memory. See Chapter 5 *Memory Management Unit* for more information.

The L1 data memory system uses the following memory types:

- *Write-Back Read-Write-Allocate* on page 6-7.
- *Write-Back No-Allocate* on page 6-7.
- *Write-Through* on page 6-8.
- *Non-Cacheable* on page 6-8.
- *Strongly-ordered and Device* on page 6-8.

Note

Some attribute combinations are only available if the LPAE page table format is used.

Table 6-1 shows the memory attribute combinations available.

Table 6-1 Memory attribute combinations

Memory type	Cacheability	Allocation policy	Cortex-A15 MPCore processor behavior
Strongly-ordered	-	-	Strongly-ordered
Device	-	-	Device
Normal	Non-Cacheable	-	Normal Non-Cacheable
Normal	Write-Through	Read-Allocate	Write-Through No-Allocate
Normal	Write-Through	Write-Allocate	Write-Through No-Allocate
Normal	Write-Through	Read-Write-Allocate	Write-Through No-ALlocate
Normal	Write-Through	No-Allocate	Write-Through Allocate
Normal	Write-Back	Read-Allocate	Write-Back Read-Write-Allocate
Normal	Write-Back	Write-Allocate	Write-Back Read-Write-Allocate
Normal	Write-Back	Read-Write-Allocate	Write-Back Read-Write-Allocate
Normal	Write-Back	No-Allocate	Write-Back No-Allocate

The L1 and L2 data memory system uses the inner memory attributes from the MMU to determine its behavior.

If any memory instruction crosses a 4KB page boundary between two pages with different memory types such as Normal or Strongly-ordered, the result is UNPREDICTABLE and an abort might be triggered or incorrect data delivered.

If any given physical address is mapped to virtual addresses with different memory types or different cacheability such as Non-Cacheable, Write-Through, or Write-Back, the result is UNPREDICTABLE. This can occur if two virtual addresses are mapped to the same physical address at the same time with different memory type or cacheability, or if the same virtual address has its memory type or cacheability changed over time without the appropriate cache cleaning or barriers.

Write-Back Read-Write-Allocate

This is expected to be the most common and highest performance memory type. Any read or write to this memory type searches the cache to determine if the line is resident. If it is, the line is read or updated. A store that hits a Write-Back cache line does not update main memory.

If the required cache line is not in the cache, one or more cache lines is requested from the L2 cache. The L2 cache can obtain the lines from its cache, from another coherent L1 cache, or from memory. The line is then placed in the L1 cache, and the operation completes from the L1 cache.

Write-Back No-Allocate

Use Write-Back No-Allocate memory to access data that might be in the cache because other virtual pages that are mapped to the same physical address are Write-Back Read-Write-Allocate. Write-Back No-Allocate memory is used to avoid polluting the caches when accessing large memory structures that are used only one time. The cache is searched and the correct data is delivered or updated if the data resides in one of the caches. However, if the request misses the

L1 or L2 cache, the line is not allocated into that cache. For a read that misses all caches, the required data is read to satisfy the memory request, but the line is not added to the cache. For a write that misses in all caches, the modified bytes are updated in memory.

Note

The No-Allocate allocation hint is only a performance hint. The Cortex-A15 MPCore processor might in some cases, allocate Write-Back No-Allocate lines into the L1 data cache or the L2.

Write-Through

The Cortex-A15 MPCore processor memory system treats all Write-Through pages as Write-Through No-Allocate. This means that no cache line from any Write-Through page allocates in any L1 data or L2 cache. Because it is not legal to map a physical page with multiple cacheability attributes, no Write-Through page can be brought into the cache from a different virtual address mapping. Therefore, memory requests for Write-Through cache lines are not looked-up in the cache. They are sent directly to the AXI master interface.

Memory that is inner or outer shared and inner Write-Through are treated as inner Non-Cacheable and outer Non-Cacheable. Memory requests to memory of this type generate ReadNoSnoop and WriteNoSnoop system-shareable requests on the ACE interconnect.

Non-Cacheable

Normal Non-Cacheable memory is not looked-up in any cache. The requests are sent directly to memory. Read requests might over-read in memory, for example, reading 64 bytes of memory for a 4-byte access, and might satisfy multiple memory requests with a single external memory access. Write requests might be merged with other write requests to the same bytes or nearby bytes.

Memory that is inner or outer shared and inner Non-Cacheable are treated as inner Non-Cacheable and outer Non-Cacheable. Memory requests to memory of this type generate ReadNoSnoop and WriteNoSnoop system-shareable requests on the ACE interconnect.

Strongly-ordered and Device

Strongly-ordered and Device memory types are used for communicating with input and output devices and memory-mapped peripherals. They are not looked-up in any cache.

For the Cortex-A15 MPCore processor, there is no distinction between shareable and nonshareable devices. Both Device and Strongly-ordered memory types are ordered together.

All the memory operations for a single instruction can be sent to the interconnect as a burst request. No Strongly-ordered or Device read request on the interconnect can cross an aligned 64-byte boundary. In addition, no Strongly-ordered or Device write request on the interconnect can cross an aligned 16-byte boundary.

6.4.2 Coherence

All memory requests for pages that are marked as Inner Shareable in the page tables and are Write-Back Cacheable, regardless of allocation policy, are coherent in all the caches that comprise the inner domain. At a minimum, this includes the L1 data cache of the executing core, the L2 cache, and all other L1 data caches in the Cortex-A15 MPCore processor. The inner domain might contain additional caches outside the Cortex-A15 MPCore processor depending on how the system is configured.

External global monitor

If synchronization primitives are used for memory pages that are Strongly-ordered, Device, or Inner-Shareable Normal Non-Cacheable, a global monitor must be provided in the interconnect. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information. The memory requests are sent on the AXI interface as Read-Exclusive or Write-Exclusive. See the *AMBA AXI and ACE Protocol Specification* for more information.

Note

- Use of synchronization primitives on addresses in regions marked as Strongly-ordered or Device is UNPREDICTABLE in the ARMv7-A Architecture. Code that makes such accesses is not portable.
- Load-Exclusive and Store-Exclusive requests are not supported to shareable Normal Write-Through memory.
- Load-Exclusive and Store-Exclusive requests are not supported to shareable Normal Write-Back memory if the cache is disabled, SCTLR.C is 0.

6.4.6 LDRT and STRT instructions

The `LDRT`, `LDRHT`, `LDRBT`, `STRT`, `STRHT`, and `STRBT` instructions are used in privileged modes to emulate user mode instructions and to enforce user mode permissions. These instructions are for all memory types when enforcing permission checking against the permissions that the page table specifies. The User mode permissions from the page table are used instead of the privileged mode permissions.

These instructions are also used to modify the privileged and user information on the **ARPROT** and **AWPROT** signals on the AXI. This is required if external permission checking hardware exists in the fabric memory.

The `LDRT` and `STRT` instructions for Strongly-ordered and Device pages appear on the AXI with an **AxPROT** value that indicates user mode access. However, the same instructions for Normal Memory might not always result in AXI transactions with an **AxPROT** value that indicates user mode access. This is because any Normal Memory page permits speculative prefetching at any time. Those prefetch requests, either caused by hardware prefetching or speculative prefetching triggered by flushed memory instructions, can have a value of the **AxPROT** field that indicates privileged mode access. This reflects the mode of the processor during the prefetch.

For Normal Write-Through Cacheable or Non-Cacheable memory, the processor can still access the memory speculatively, and can merge multiple stores together before issuing them to the AXI. Because of this, you must use the `LDRT` and `STRT` instructions to present user mode on **AxPROT** if the `LDRT` and `STRT` instructions are preceded and followed by `DMB` instructions:

- `DMB`.
- `LDRT` or `STRT`.
- `DMB`.

The `DMB` instructions prevent the `LDRT` or `STRT` instruction from hitting any previously requested read data, or from merging with any other requests. The `DMB` instructions can be `DMBSY`, `DMBISH`, `DMBISH`, and `DMBOSH`.

6.4.7 Preload instruction behavior

The Cortex-A15 MPCore processor supports both the `PLD` and `PLDW` prefetch hint instructions. For Normal Write-Back Cacheable memory page, the preload instructions cause the line to be allocated to the L1 data cache of the executing core. The `PLD` instruction brings the line into the cache in Exclusive or Shared state and the PLDW instruction brings the line into the cache in Exclusive state. The preload instruction cache, PLDI, is treated as a NOP. PLD and PLDW instructions are performance hints instructions only and might be dropped in some cases.

6.4.8 Error Correction Code

The L1 data cache supports optional single bit correct and double bit detect error correction logic in both the tag and data arrays. The ECC granularity for the tag array is the tag for a single cache line and the ECC granularity for the data array is a 32-bit word.

Because of the ECC granularity in the data array, a write to the array cannot update a portion of a 4-byte aligned memory location because there is not enough information to calculate the new ECC value. This is the case for any store instruction that does not write one or more aligned 4-byte regions of memory. In this case, the L1 data memory system reads the existing data in the cache, merges in the modified bytes, and calculates the ECC from the merged value. The L1 memory system attempts to merge multiple stores together to meet the aligned 4-byte ECC granularity and to avoid the read-modify-write requirement.

Single bit ECC errors in the tag or cache are corrected in the background. Because the line is removed from the L1 cache as part of the correction process, no software intervention is required. No exception or interrupt is generated. The CPU Memory Error Syndrome Register, see *CPU Memory Error Syndrome Register* on page 4-107, is updated to indicate a non-fatal error.

Double bit ECC errors in the tag or cache are detected and an imprecise Data Abort is triggered. The line that contains the error is evicted from the cache. When a double bit error is reported, you must assume that data corruption has occurred and handle this appropriately.

For any detected ECC error in the L1 memory system, the CPU Memory Error Syndrome Register is updated. For the first error reported, the register is updated with information for the RAM, bank, way, and index that contain the error. If that same location reports multiple errors, the repeat error count is incremented. If any other RAM locations report errors, the other error count is incremented. Double-bit ECC errors set the fatal bit. When the register is written with zeros, the register clears all counts and starts to monitor for a new first error again.

If a double-bit ECC error triggers an abort in the L1 data memory system, the Auxiliary Data Fault Status Register is updated with information for the RAM, bank, set, way, and index that triggered the abort. See *Auxiliary Data Fault Status Register* on page 4-81.

6.5 Program flow prediction

The Cortex-A15 MPCore processor contains program flow prediction hardware, also known as *branch prediction*. With program flow prediction disabled, all taken branches incur a penalty associated with flushing the pipeline. To avoid this penalty, the branch prediction hardware operates at the front of the instruction pipeline. The branch prediction hardware consists of:

- A *Branch Target Buffer* (BTB) to identify branches and provide targets for direct branches.
- 2-level global history-based direction predictor.
- Indirect predictor to provide targets for indirect branches.
- Return stack.
- Static predictor.

The combination of global history-based direction predictor and BTB are called *dynamic predictor*.

This section describes program flow prediction in:

- *Predicted and non-predicted instructions*.
- *Return stack predictions* on page 6-13.
- *Indirect predictor* on page 6-13.
- *Static predictor* on page 6-13.
- *Enabling program flow prediction* on page 6-13.

6.5.1 Predicted and non-predicted instructions

This section describes the instructions that the processor predicts. Unless otherwise specified, the list applies to ARM, Thumb, and ThumbEE instructions. As a general rule, the branch prediction hardware predicts all branch instructions regardless of the addressing mode, including:

- Conditional branches.
- Unconditional branches.
- Indirect branches.
- Branches that switch between ARM and Thumb states.
- PC destination data processing operations.
- Handler branches, `HB` and `HBL`, in ThumbEE state.
- `BXJ`, because of the inclusion of the trivial Jazelle Extension, this degenerates to a `BX` instruction.

However, the following branch instructions are not predicted:

- Instructions with the *S* suffix are not predicted because they are typically used to return from exceptions and have side effects that can change privilege mode and security state.
- All mode changing instructions.

In Thumb state, you can make a branch that is normally encoded as unconditional conditional by including an *If-Then* (IT) block. It is then treated as a normal conditional branch.

6.5.2 Return stack predictions

The return stack stores the address and the ARM or Thumb state of the instruction after a function-call type branch instruction. This address is the same as the link register value stored in r14. The following instructions cause a return stack push if predicted:

- `BL immediate`.
- `BLX(1) immediate`.
- `BLX(2) register`.
- `HBL`, ThumbEE state.
- `HBLP`, ThumbEE state.

The following instructions cause a return stack pop if predicted:

- `BX r14`.
- `MOV pc, r14`.
- `LDM r13, {…pc}`.
- `LDR pc, [r13]`.

The `LDR` instruction can use any of the addressing modes, as long as r13 is the base register. Additionally, in ThumbEE state, you can also use r9 as a stack pointer so the `LDR` and `LDM` instructions with pc as a destination and r9 as a base register are also treated as a return stack pop.

Because return-from-exception instructions can change the processor privilege mode and security state, they are not predicted. This includes the `LDM(3)` instruction, and the `MOVS pc, r14` instruction.

6.5.3 Indirect predictor

The indirect predictor can predict indirect branches that are not return-type instructions. This predictor augments the branch address with an additional state that predicts the target address of an indirect branch. The conditional branch predictor still predicts the direction of conditional indirect branches. The indirect predictor only provides the address on a predicted taken conditional indirect branch.

6.5.4 Static predictor

Branches must be resolved one time to be predicted by the dynamic predictor. To accelerate cold startup of code, the Cortex-A15 MPCore processor includes a static predictor that detects branches in the code stream as follows:

- Direct unconditional branches, `B immediate`, are predicted taken.
- Direct unconditional call-type branches, `BL immediate` and `BLX immediate`, are predicted taken, and the r14 value is pushed on the return stack.
- Direct conditional backwards branches, `BCC immediate`, are predicted taken.
- Unconditional return-type branches, see *Return stack predictions*, are predicted taken and the target is popped from the return stack.

To avoid potential illegal speculation, the static predictor is disabled when the MMU is disabled. The static predictor does not predict return stack pops with r9 base register in ThumbEE state.

6.5.5 Enabling program flow prediction

The program flow prediction is enabled by setting the Z bit in the CP15 System Control Register to 1. See *System Control Register* on page 4-54 for more information.

Reset:

- Disables program flow prediction.
- Invalidates the BTB.
- Resets the GHB and indirect predictor to a known state.

No software intervention is required to prepare the prediction logic before enabling program flow prediction.

6.6 L1 RAM memories

The L1 memory system contains several RAM memories that can be configured to use ECC or parity error detection mechanisms. Any RAM memory that uses ECC support can perform single bit error correction and double bit error detection. Contents of the RAM memories with parity support can invalidate entries if a parity error is detected because this data is associated with read-only structures.

Table 6-1 on page 6-7 shows all RAM memories contained in the L1 memory system.

Table 6-2 L1 RAM memories

RAM memory	ECC or Parity
L1 instruction tag RAM	Parity
L1 instruction data RAM	Parity
L1 instruction BTB RAM	Parity
L1 instruction GHB RAM	None
L1 instruction indirect predictor RAM	None
L1 data tag RAM	ECC
L1 data data RAM	ECC
L2 TLB RAM[a]	Parity

[a] The L2 TLB RAM is a unified TLB structure that supports L1 instruction and L1 data TLB misses.

Chapter 7
Level 2 Memory System

This chapter describes the *Level 2* (L2) memory system. It contains the following sections:

7.1 About the L2 memory system

The L2 memory system consists of a tightly-coupled L2 cache and an integrated *Snoop Control Unit* (SCU), connecting up to four processors within a Cortex-A15 MPCore device. The L2 memory system also interfaces with an AMBA 4 (ACE) interconnect and an *Accelerator Coherency Port* (ACP) that is implemented as an AXI3 slave interface.

The features of the L2 memory system include:

- Configurable L2 cache size of 512KB, 1MB, 2MB and 4MB.
- Fixed line length of 64 bytes.
- Physically indexed and tagged cache.
- 16-way set-associative cache structure.
- Banked pipeline structures.
- Strictly enforced inclusion property with L1 data caches.
- Random cache-replacement policy.
- Configurable 64-bit or 128-bit wide ACE with support for multiple outstanding requests.
- Configurable 64-bit or 128-bit wide ACP with support for multiple incoming requests.
- Duplicate copies of the L1 data cache directories for coherency support.
- Optional *Error Correction Code* (ECC) support.
- Optional hardware prefetch support.
- Software-programmable variable latency RAMs.
- Register slice support for large L2 cache sizes to minimize impact on routing delays.
- MBIST support.

Note

The Cortex-A15 MPCore processor does not support TLB or cache lockdown.

7.2 Cache organization

The L2 cache is 16-way set-associative of configurable size. The cache is physically-addressed. The cache sizes are configurable with sizes of 512KB, 1MB, 2MB, and 4MB.

You can configure the L2 memory system pipeline to insert wait states to take into account the latencies of the compiled memories for the implementation of the RAMs.

The L2 cache incorporates a single dirty bit per cache line. A write to a cache line results in the line being written back to memory after the line is evicted from the L2 cache.

This section describes cache organization in:

- *L2 cache bank structure*.
- *Strictly-enforced inclusion property with L1 data caches* on page 7-4.
- *Enabling and disabling the L2 cache* on page 7-4.
- *Error Correction Code* on page 7-4.
- *Register slice support for large cache sizes* on page 7-5.

7.2.1 L2 cache bank structure

The L2 cache is partitioned into multiple banks to enable parallel operations. The following levels of banking exist:

- The tag array is partitioned into multiple banks to enable up to four requests to access different tag banks of the L2 cache simultaneously.
- Each tag bank is partitioned into multiple data banks to enable streaming accesses to the data banks. Each tag bank consists of four data banks.

Figure 7-1 shows the logical representation of an L2 cache bank structure with a configuration of all possible tag and data bank combinations.

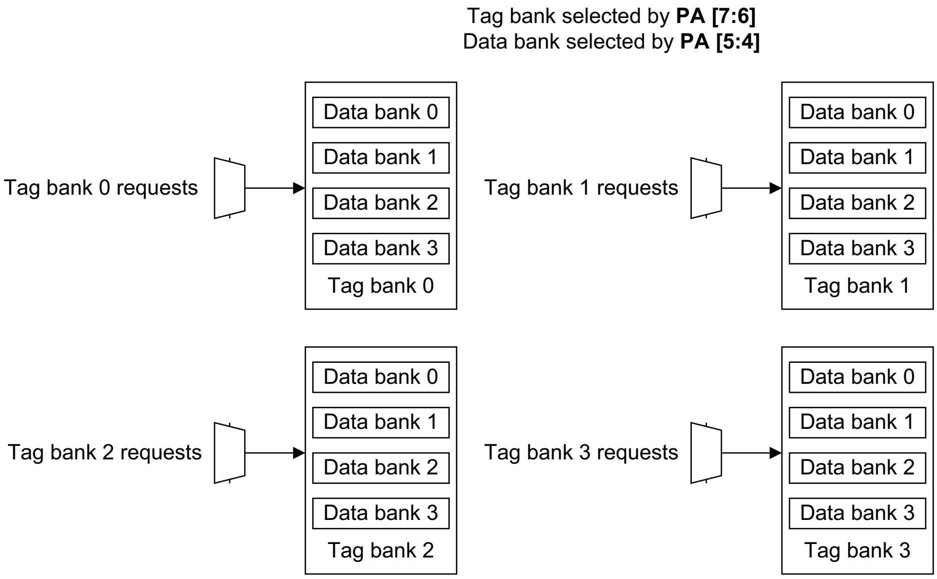


Figure 7-1 L2 cache bank structure

7.2.2 Strictly-enforced inclusion property with L1 data caches

The L2 memory system requires support for inclusion between the L1 data caches and the L2 cache. A line that resides in any of the L1 data caches must also reside in the L2 cache. This approach has the following benefits:

- Any AXI ReadClean operation that results in a line being in shared state in the L1 data caches can be returned from the L2 cache. This yields the highest performance for delivering data to a core.
- When powering down the processor, the time to clean and invalidate the entire L1 data cache is more efficient.

7.2.3 Enabling and disabling the L2 cache

For processor requests, the L2 cache is enabled when the C bit of the SCTLR register is enabled, see *System Control Register* on page 4-54. The cache attributes are provided with each request, taking into account the page attributes that the MMU page tables provided and overriding these attributes if the corresponding cache enable bit in the SCTLR is disabled.

To enable the L2 cache to cache both instructions and data following the reset sequence, you must:

1. Complete the processor reset sequence.
2. Program the I bit and C bit of the SCTLR.

Note

If the L2 memory system is configured to support ECC, you must enable this feature by programming bit[21] of the L2 Control Register before you can program the C bit. See *L2 Control Register* on page 4-85.

To disable the L2 cache, you must use the following sequence:

1. Disable the C bit.
2. Clean and invalidate the L1 and L2 caches.

For ACP requests, the L2 cache is enabled if the request uses Normal Write-Back memory attributes. The processor searches the L2 cache to determine if the request is valid before allocating the line for Normal Write-Back Read-Write Allocate memory.

7.2.4 Error Correction Code

The L2 cache can be configured to support ECC. Some memories within the L2 memory system support ECC when configured. For core instruction and data accesses resulting in an L2 cache hit, where a single-bit error is detected on the data array, the L2 memory system supports in-line ECC correction. Uncorrected data is forwarded to the requesting unit, and in parallel, the ECC circuitry checks for accuracy. If a single-bit error is detected, any uncorrected data returned within two cycles before the error indicator must be discarded. The L2 memory system begins to stream corrected data to the requestor.

After it is determined that no data is being transferred, the L2 memory system shifts back to return uncorrected data until the next single-bit error is detected.

For all other single-bit ECC errors detected, the request is flushed from the L2 pipeline and is forced to reissue. The tag bank where the single-bit error occurred, performs a read-modify-write sequence to correct the single-bit error in the array. The request is then reissued.

7.2.5 Register slice support for large cache sizes

As the L2 cache size is increased, the area of the implementation increases. This increase adds significant route delays to and from the RAM memories. This increase can impact the maximum frequency of the implementation. To counter this, you can insert register slices before and after the RAM memories to offset the longer route delays. This enables the frequency target of the implementation to remain high. Additional slices can have an impact on overall L2 hit latency, but can enable requests to be streamed in a more efficient manner. You can increase the programmed latency values of the RAMs to cover the additional route delays without adding the slices. However, this method has an impact on performance because requests cannot be streamed as efficiently.

The L2 data and data ECC RAMs support up to two inserted register slices, whereas all other L2 RAMs can only support one inserted register slice. Each register slice introduces a pair of registers, one before the RAM and one after the RAM.

Bits[12:10] of the CP15 L2 Control Register, L2CTLR, indicate the number of RAM register slices in the design. In addition, the L2CTLR contains bits to program the setup and latency for the L2 tag and data RAMs. See *L2 Control Register* on page 4-85 for more information.

Overall RAM latency calculation

The RAM latency is a function of the following:

- Programmed latency in the L2 Control Register, L2CTLR, see *L2 Control Register* on page 4-85.
- Additional strobe clock setup required value in the L2CTLR.
- Number of slices added.

RAM latency = programmed value + strobe setup + 2 (N), where *N* is the number of register slices to insert.

Table 7-1 shows the adjusted L2 tag RAM latency with the register slice and setup factored in.

Table 7-1 L2 tag RAM latency with slice and setup factored in

	Total adjusted tag RAM latency			
L2CTLR[8:6]	**Tag slice = 0** **Tag setup = 0**	**Tag slice = 0** **Tag setup = 1**	**Tag slice = 1** **Tag setup = 0**	**Tag slice = 1** **Tag setup = 1**
000[a]	2	3	4	5
001	2	3	4	5
010	3	4	5	5
011	4	5	5	5
100	5	5	5	5
1xx, >= 4	5	5	5	5

[a] This is the reset value.

Note

- The L2 tag RAM total latency is set to a maximum of 5 cycles.

- Each tag slice adds 2 cycles and affects the L2 tag, snoop tag, dirty, and prefetch stride queue RAMs.
- Setting tag setup to 1 adds 1 cycle.
- Slice and setup have priority over programmed latency in determining the total adjusted RAM latency.

Example 7-1 shows a tag RAM access with 3 cycles total RAM latency.

Example 7-1 Tag RAM access with 3 cycles total latency

When tag slice = 0, L2CTLR[9] = 0, L2CTLR[8:6] = 3'b010, the following applies:

- No slice cycle.
- No setup cycle.
- 3 cycles tag RAM access.
- 3 cycles total tag RAM latency.

Example 7-2 shows a tag RAM access with 5 cycles total RAM latency.

Example 7-2 Tag RAM access with 5 cycles total latency

When tag slice = 1, L2CTLR[9] = 1, L2CTLR[8:6] = 3'b010, the following applies:

- 2 slice cycles.
- 1 setup cycle.
- 2 cycles tag RAM access adjusted because of slice and setup values.
- 5 cycles total tag RAM latency.

Table 7-2 shows the adjusted L2 data RAM latency with the register slice and setup factored in.

Table 7-2 L2 data RAM latency with slice and setup factored in

	Total adjusted data RAM latency					
L2CTLR [2:0]	**Data slice = 0 Data setup = 0**	**Data slice = 0 Data setup = 1**	**Data slice = 1 Data setup = 0**	**Data slice = 1 Data setup = 1**	**Data slice = 2 Data setup = 0**	**Data slice = 2 Data setup = 1**
000[a]	2	3	4	5	6	7
001	2	3	4	5	6	7
010	3	4	5	6	7	8
011	4	5	6	7	8	8
100	5	6	7	8	8	8
101	6	7	8	8	8	8
110	7	8	8	8	8	8
111	8	8	8	8	8	8

[a] This is the reset value.

Note

- The L2 data RAM total latency is set to a maximum of 8 cycles.
- Each data slice adds 2 cycles and affects the L2 data and data ECC RAMs.
- Setting data setup to 1 adds 1 cycle.
- Slice and setup have priority over programmed latency in determining the total adjusted RAM latency.

Example 7-3 shows a data RAM access with 4 cycles total RAM latency.

Example 7-3 Data RAM access with 4 cycles total latency

When data slice = 0, L2CTLR[5] = 0, L2CTLR[2:0] = 3'b011, the following applies:

- no slice cycle.
- no setup cycle.
- 4 cycles data RAM access.
- 4 cycles total data RAM latency.

Example 7-4 shows a data RAM access with 8 cycles total RAM latency.

Example 7-4 Data RAM access with 8 cycles total latency

When data slice = 2, L2CTLR[5] = 1, L2CTLR[2:0] = 3'b011, the following applies:

- 4 slice cycles.
- 1 setup cycle.
- 3 cycles data RAM access adjusted because of slice and setup values.
- 8 cycles total data RAM latency.

7.3 L2 RAM memories

The L2 memory system contains several RAM memories that can be configured to use ECC or parity error detection mechanisms. Any RAM memory that uses ECC support can perform single bit error correction and double bit error detection. Contents of the RAM memories with parity support can invalidate entries if a parity error is detected because this data is associated with read-only structures.

Table 7-3 shows all RAM memories contained in the L2 memory system.

Table 7-3 L2 RAM memories

RAM memory	ECC or Parity
L2 tag RAM	ECC or parity
L2 snoop tag RAM	ECC
L2 data RAM	ECC
L2 data ECC RAM	ECC associated with L2 data RAM
L2 dirty RAM	ECC
L2 prefetch RAM	Parity

7.4 L2 cache prefetcher

The Cortex-A15 MPCore processor includes a hardware L2 auto-prefetcher. Some of the key features are:

- Software-programmable prefetches on any L2 miss of 0, 2, 4, or 8 for load-store misses and 0, 1, 2, or 3 for instruction fetch misses. All prefetches are allocated into the L2 cache.
- Separate mechanisms to detect and prefetch:
  - Load-store streams, to stride detection within a 4K page.
  - Instruction fetch streams, to fetch consecutive cache lines on an L2 instruction fetch miss or an L2 cache prefetch hit.
  - Table walk descriptor, to fetch the consecutive cache line on an L2 table walk descriptor miss.

  Note

  The prefetcher is limited to prefetch within the 4KB page of the current request.

- A 16-entry prefetch request queue per processor that holds the prefetch requests generated by either the load-store, instruction fetch, or table walk prefetchers.
- A throttle mechanism to limit a maximum of six outstanding prefetch requests from consuming all of the shared resources that handle the data transfer to and from memory.
- Support for forwarding from prefetched requests. If a read request was sent over AXI because of a prefetch request, and a demand access for the same line was received, the read data can be forwarded from the internal data buffers to the demand request, before waiting for the line to be allocated to the cache.

You can program the L2 Prefetch Control Register, see *L2 Prefetch Control Register* on page 4-103, to indicate the maximum number of prefetches to be allocated in the PRQ on the following:

- An instruction fetch miss in the L2 cache.
- A load-store miss with a stride match in the L2 cache.

The programmed distance is also used as the skip distance for any load-store or instruction fetch read with a stride match that hits in the L2 cache. In these cases, a single prefetch request is allocate in the PRQ as:

prefetch address = current address + (stride x programmed distance)

Note

The stride for an instruction fetch access is always one cache line.

7.5 Cache coherency

The SCU uses hybrid *Modified Exclusive Shared Invalid* (MESI) and *Modified Owned Exclusive Shared Invalid* (MOESI) protocols to maintain coherency between the individual L1 data caches and the L2 cache. The L1 data caches support the MESI protocol. The L2 memory system contains a snoop tag array that is a duplicate copy of each of the L1 data cache directories. The snoop tag array reduces the amount of snoop traffic between the L2 memory system and the L1 memory system. Any line that resides in the snoop tag array in the Modified/Exclusive state belongs to the L1 memory system. Any access that hits against a line in this state must be serviced by the L1 memory system and passed to the L2 memory system. If the line is invalid or in the shared state in the snoop tag array, then the L2 cache can supply the data.

The SCU contains buffers that can handle direct cache-to-cache transfers between cores without reading or writing any data on the ACE. Lines can migrate back and forth without any change to the MOESI state of the line in the L2 cache.

Shareable transactions on the ACP are also coherent, so the snoop tag arrays are queried as a result of ACP transactions. For reads where the shareable line resides in one of the L1 data caches in the Modified/Exclusive state, the line is transferred from the L1 memory system to the L2 memory system and passed back on the ACP.

7.6 Asynchronous errors

The L2 memory system has two outputs that indicate asynchronous error conditions. An asynchronous external error condition exists when either:

- The **nAXIERRIRQ** output is asserted LOW.
- The **nINTERRIRQ** output is asserted LOW.

If an asynchronous error condition is detected, the corresponding bit in the L2 Extended Control Register is asserted. The asynchronous error condition can be cleared by writing 1'b0 to the corresponding bit of the L2ECTLR. Software can only clear the L2ECTLR. Any attempt to assert the error by writing the L2ECTLR is ignored. See *L2 Extended Control Register* on page 4-87 for more information.

External errors on the ACE write response channel or on the ACE read data channel, associated with a write-allocate memory transaction, cause the **nAXIERRIRQ** signal to be asserted LOW.

Double-bit ECC errors or invalid accesses to the internal GIC interrupt controller cause the **nINTERRIRQ** signal to be asserted LOW.

Any external error associated with a load instruction of type ReadNoSnoop, ReadShared, ReadClean or ReadUnique is reported back to the requestor along with an error response and this might trigger an abort.

External errors associated with cache maintenance operations of type CleanShared or CleanInvalid are silently dropped.

.

7.7 AXI Coherency Extensions

*AXI Coherency Extension*s (ACE) is an extension to the AXI protocol and provides the following enhancements:

- Third-level caches.
- On-chip RAM.
- Peripherals.
- External memory.

The width of the AXI read and write channels can be configured for a 64-bit or 128-bit interface.

ACE supports 1:1 clock ratios with respect to the processor clock. It can also run at any integer multiple of the processor clock N:1.

Note

- The Cortex-A15 MPCore processor does not support a 64-bit ACE with a 128-bit ACP.
- No read or write request from a Cortex-A15 MPCore ACE master port can cross a 64-byte aligned boundary. The request, including non-cacheable, is always broken up.

This section describes ACE in:

- *ACE L2 memory interface attributes*.
- *ARID and AWID*.
- *ACE transfers* on page 7-13.
- *Distributed virtual memory transactions* on page 7-14.
- *Cache maintenance transactions* on page 7-14.
- *Snoop filter support* on page 7-15.
- *ACE configurations* on page 7-15.
- *Configuration signals* on page 7-16.

7.7.1 ACE L2 memory interface attributes

Table 7-4 shows the ACE L2 memory interface attributes for the Cortex-A15 MPCore processor. The table lists the maximum possible values for the read and write issuing capabilities.

Table 7-4 ACE L2 memory interface attributes

Attribute	Value	Description
Write issuing capability	16	16 outstanding writes supported that can be evictions, single writes, or write bursts of any memory type.
Read issuing capability	11	11 outstanding reads supported that can be linefills, single reads, or read bursts of any memory type.
Combined issuing capability	27	-
Snoop acceptance capability	20	12 requests buffered and 8 requests actively being processed.

7.7.2 ARID and AWID

When the system issues multiple requests on the AR channel with the same **ARID**, or on the AW channel with the same **AWID**, it must follow the appropriate ordering rules as described in the *ARM® AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™, ACE and ACE-Lite™*.

For certain transactions, the system must be able to identify which processor generated the request. This applies to requests affecting the global exclusive monitor in addition to Strongly ordered or Device memory type accesses to peripherals.

For non-barrier transactions, **ARCACHEM[3:0]** and **AWCACHEM[3:0]** identify whether the memory types are Strongly ordered, Device, or Normal Non-cacheable. See the *AMBA AXI Protocol Specification*. For these memory types, if **ARIDM[5]** or **AWIDM[5]** is 1'b1, then the request is generated from one of the processors. If **ARIDM[5]** or **AWIDM[5]** is 1'b0, the request is originated from the master connected to the ACP slave port. **ARIDM[1:0]** or **AWIDM[1:0]** indicates which processor generated the request.

For an exclusive read transaction such as when **ARLOCK** is asserted, **ARID[1:0]** indicates which processor generated the request. Only processors can generate exclusive read requests, and not the ACP or any other source.

For an exclusive write transaction such as **AWLOCK** asserted, **AWID[1:0]** indicates which processor generated the request. Only processors can generate exclusive write requests, and not the ACP or any other source.

The system does not rely on specific values of **ARID** or **AWID** that correspond with specific transaction sources or transaction types other than the information described in this section.

7.7.3 ACE transfers

For Normal Inner-Cacheable memory transfers initiated from one of the processors, the following transfers are supported on the ACE:

- WRAP 8x64-bit read transfers (64-bit ACE only).
- WRAP 4x128-bit read transfers (128-bit ACE only).
- WRAP 8x64-bit write transfers (64-bit ACE only).
- WRAP 4x128-bit write transfers (128-bit ACE only).

For Non-Cacheable, Cacheable but not allocated, Strongly-ordered, or Device transactions initiated from one of the processors, the following transfers are supported on the ACE:

- INCR N (N:1-16) 8-bit read transfers.
- INCR N (N:1-16) 16-bit read transfers.
- INCR N (N:1-16) 32-bit read transfers.
- INCR N (N:1-8) 64-bit read transfers.
- INCR N (N:1-4) 128-bit read transfers.
- INCR N (N:1-16) 8-bit write transfers.
- INCR N (N:1-16) 16-bit write transfers.
- INCR N (N:1-16) 32-bit write transfers.
- INCR N (N:1-8) 64-bit write transfers.
- INCR N (N:1-4) 128-bit write transfers.

If there are requests on the Cortex-A15 MPCore ACP port, the following transfers can be generated on the ACE if comparable requests are received on the ACP:

- WRAP N (N:1-16) 8-bit read transfers.
- WRAP N (N:1-16) 16-bit read transfers.
- WRAP N (N:1-16) 32-bit read transfers.
- WRAP N (N:1-8) 64-bit read transfers.
- WRAP N (N:1-4) 128-bit read transfers.
- WRAP N (N:1-16) 8-bit write transfers.
- WRAP N (N:1-16) 16-bit write transfers.
- WRAP N (N:1-16) 32-bit write transfers.

- WRAP N (N:1-8) 64-bit write transfers.
- WRAP N (N:1-4) 128-bit write transfers.
- INCR N (N:1-16) 8-bit read transfers.
- INCR N (N:1-16) 16-bit read transfers.
- INCR N (N:1-16) 32-bit read transfers.
- INCR N (N:1-8) 64-bit read transfers.
- INCR N (N:1-4) 128-bit read transfers.
- INCR N (N:1-16) 8-bit write transfers.
- INCR N (N:1-16) 16-bit write transfers.
- INCR N (N:1-16) 32-bit write transfers.
- INCR N (N:1-8) 64-bit write transfers.
- INCR N (N:1-4) 128-bit write transfers.
- FIXED N (N:1-16) 8-bit read transfers.
- FIXED N (N:1-16) 16-bit read transfers.
- FIXED N (N:1-16) 32-bit read transfers.
- FIXED N (N:1-8) 64-bit read transfers.
- FIXED N (N:1-4) 128-bit read transfers.
- FIXED N (N:1-16) 8-bit write transfers.
- FIXED N (N:1-16) 16-bit write transfers.
- FIXED N (N:1-16) 32-bit write transfers.
- FIXED N (N:1-8) 64-bit write transfers.
- FIXED N (N:1-4) 128-bit write transfers.

Note

- ACP requests with burst mode INCR that crosses a 64-byte aligned boundary are broken into multiple ACE requests of burst type INCR and of size 64 bytes or less.
- ACP requests with burst mode WRAP that crosses a 64-byte aligned boundary are broken into multiple ACE requests of burst type INCR of size 64 bytes or less. The received data is returned on the ACP port in the order required by the WRAP request.
- ACP requests with burst mode FIXED that requests more than 64 bytes generate multiple FIXED requests on the ACE of size 64 bytes or less.

7.7.4 Distributed virtual memory transactions

In a system where the Cortex-A15 MPCore processor can receive a *Distributed Virtual Memory* (DVM) synchronization message over the AXI master snoop address channel, **BRESP** for any write transaction must not be asserted to the processor until all AXI masters that might have initiated the DVM synchronization request observe the transaction.

Note

The Cortex-A15 MPCore processor does not support a multi-part DVM hint message.

7.7.5 Cache maintenance transactions

The **BROADCASTCACHEMAINT** signal controls the broadcasting of cache maintenance operations on the AR-channel. For more information on cache maintenance operations, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* and the *ARM® AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™, ACE and ACE-Lite™*.

7.7.6 Snoop filter support

In general, the Cortex-A15 MPCore processor can issue either a write-back or an evict transaction for any cache line that is removed from the L2 cache. You can use these messages to manage an external snoop filter. However, the snoop filter logic must not depend on an evict message for every clean line dropped from the Cortex-A15 MPCore processor caches. In some circumstances, the Cortex-A15 MPCore processor might not signal an eviction. For example, clean evictions are not guaranteed to occur in all cases for write-through memory types.

7.7.7 ACE configurations

The Cortex-A15 MPCore processor supports the following ACE configurations:

- *AXI3 mode*.
- *ACE non-coherent, no L3 cache*.
- *ACE non-coherent, with L3 cache*.
- *ACE outer coherent*.
- *ACE inner coherent* on page 7-16.

AXI3 mode

The AXI3 mode configuration has the following key features:

- AXI3-compliant.
- ReadNoSnoop is the only AR channel command issued.
- WriteNoSnoop is the only AW channel command issued.
- No cache maintenance, DVM operations, or barriers issued on the AR channel
- No request sent to the processor on the snoop AC channel.

ACE non-coherent, no L3 cache

The ACE non-coherent, no L3 cache configuration has the following key features:

- ACE-compliant with no coherent masters on the ACE.
- No L3 cache external to the processor.
- ReadNoSnoop and barriers are the only AR channel commands issued.
- WriteNoSnoop and barriers are the only AW channel commands issued.
- No cache maintenance or DVM operations issued on the AR channe.l
- No request sent to the processor on the snoop AC channel.

ACE non-coherent, with L3 cache

The ACE non-coherent, with L3 cache configuration has the following key features:

- ACE-compliant with no coherent masters on the ACE.
- L3 cache external to the processor.
- ReadNoSnoop, barriers, and cache maintenance are the only AR channel commands issued.
- WriteNoSnoop and barriers are the only AW channel commands issued.
- No DVM operations issued on the AR channel.
- No request sent to the processor on the snoop AC channel.

ACE outer coherent

The ACE outer coherent configuration has the following key features:

- ACE-compliant with coherent masters on the ACE.

- The coherent masters can share memory in the outer shareable domain only.
- L3 cache external to the processor.
- The AR channel can issue any command, except for DVM operations.
- The AW channel can issue any command.
- The system can send snoop commands for outer shareable memory regions on the snoop AC channel.

ACE inner coherent

The ACE inner coherent configuration has the following key features:

- ACE-compliant with coherent masters on the ACE.
- The coherent masters can share memory in the inner or outer shareable domains.
- L3 cache external to the processor.
- The AR channel can issue any command, including DVM operations.
- The AW channel can issue any command.
- The system can send DVM or snoop commands for any shareable memory region on the snoop AC channel.

7.7.8 Configuration signals

The Cortex-A15 MPCore processor implements the following ACE configuration signals:

- **BROADCASTINNER**.
- **BROADCASTOUTER**.
- **BROADCASTCACHEMAINT**.
- **SYSBARDISABLE**.

Table 7-5 shows the permitted combinations of these signals and the supported configurations in the Cortex-A15 MPCore processor.

Table 7-5 Supported AXI master bus configurations

Signal	Feature				
	AXI3 mode	ACE non-coherent, no L3 cache	ACE non-coherent, with L3 cache	ACE outer coherent	ACE inner coherent
BROADCASTINNER	0	0	0	0	1
BROADCASTOUTER	0	0	0	1	1
BROADCASTCACHEMAINTENANCE	0	0	1	1	1
SYSBARDISABLE	1	0	0	0	0

Note

- If the **BROADCASTINNER** and **BROADCASTOUTER** signals are deasserted, this implies that the Cortex-A15 MPCore processor is in a non-coherent system with no coherent masters on the ACE interconnect and no transactions can appear on the snoop AC channel.

- If any valid requests are issued to the Cortex-A15 MPCore processor on the AC channel when **BROADCASTINNER** and **BROADCASTOUTER** are both deasserted, the results are UNPREDICTABLE.

Table 7-6 shows the key features in each of the supported ACE configurations.

Table 7-6 Supported features in the ACE configurations

Features	Configuration				
	AXI3 mode	ACE non-coherent, no L3 cache	ACE non-coherent, with L3 cache	ACE outer coherent	ACE inner coherent
AXI3 compliance	Y	N	N	N	N
ACE compliance	N	Y	Y	Y	Y
Barriers on AR channel	N	Y	Y	Y	Y
Caches on AR channel	N	N	Y	Y	Y
Snoops on AC channel	N	N	N	Y	Y
Coherent requests on AR or AW channel	N	N	N	Y	Y
DVM requests on AR channel	N	N	N	N	Y

7.8 Accelerator Coherency Port

Accelerator Coherency Port (ACP) is implemented as an AXI3 slave interface that supports the following features:

- Configurable 64-bit or 128-bit read and write interfaces.
- All burst types such as INCR, WRAP, or FIXED.

Note

- ACP does not support fixed addressing mode to Normal Memory.
- The Cortex-A15 MPCore processor does not support a 64-bit ACE with a 128-bit ACP.

This section describes ACP in:

- *ACP L2 memory interface attributes*.
- *Burst support*.
- *ACP user bits*.

7.8.1 ACP L2 memory interface attributes

Table 7-7 shows the ACP L2 memory interface attributes for the Cortex-A15 MPCore processor. The table lists the maximum possible values for the read and write issuing capabilities.

Table 7-7 ACP L2 memory interface attributes

Attribute	Value	Description
ACP read acceptance capability	8	4 requests buffered and 4 requests actively being processed.
ACP write acceptance capability	8	4 requests buffered and 4 requests actively being processed.

7.8.2 Burst support

ACP supports a maximum burst lengths of 16 over AXI3. The L2 memory system breaks up transactions on a cache line boundary, creating additional requests to the L2 pipeline, and possibly the ACE for L2 cache misses. Ordering is maintained for the requests, and the responses are generated in the appropriate order on the ACP. For reads, data must be returned in the order of the original burst received. For writes to Strongly-ordered memory, the BRESP is generated from the final destination. All other BRESPs are generated on the last write data packet received on the ACP interface.

The Cortex-A15 MPCore processor supports fixed burst type transactions only to Device and Strongly-ordered memory.

7.8.3 ACP user bits

Because ACP transactions can be cached in the L2 cache, and the L2 observes inner memory page attributes, these must be supplied on the ACP.

To pass the outer page attributes, use the **ARCACHE[3:0]** and **AWCACHE[3:0]** fields.

To pass the inner page attributes, use the **ARUSER[5:2]** and **AWUSER[5:2]** fields. To pass the inner shareable attribute, use **ARUSER[1]** and **AWUSER[1]**. To pass the outer shareable attribute, use **ARUSER[0]** and **AWUSER[0]**. See Appendix A *Signal Descriptions* for more information.

Chapter 8
Generic Interrupt Controller

This chapter describes the *Generic Interrupt Controller* (GIC) for the Cortex-A15 MPCore processor. It contains the following sections:

- *About the Generic Interrupt Controller* on page 8-2.
- *GIC functional description* on page 8-3.
- *GIC programmers model* on page 8-7.

8.1 About the Generic Interrupt Controller

The GIC collates and arbitrates from a large number of interrupt sources. It provides:

- Masking of interrupts.
- Prioritization of interrupts.
- Distribution of the interrupts to the target processors.
- Tracking the status of interrupts.
- Generation of interrupts by software.
- Support for Security Extensions.
- Support for Virtualization Extensions.

The Cortex-A15 MPCore GIC is compliant with version 2.0 of the *ARM Generic Interrupt Controller Architecture Specification*.

This chapter only describes features that are specific to the Cortex-A15 MPCore implementation.

8.2 GIC functional description

This section provides a functional description of the Cortex-A15 MPCore GIC in:

- *GIC clock frequency*.
- *GIC memory-map*.
- *Interrupt sources*

Table 8-1 lists the address offsets for the GIC blocks relative to the **PERIPHBASE** base address. Read access to reserved regions results in a Data Abort exception to the requesting processor. Write access to reserved regions results in the assertion of **nINTERRIRQ**.

Table 8-1 Cortex-A15 MPCore GIC memory map

Base offset from PERIPHBASE[39:15]	Offset range from PERIPHBASE[39:15]	GIC block
`0x0000`	`0x0000` – `0x0FFF`	Reserved
`0x1000`	`0x1000` – `0x1FFF`	Distributor
`0x2000`	`0x2000` – `0x3FFF`	CPU interface
`0x4000`	`0x4000` – `0x4FFF`	Virtual interface control (common base address)
`0x5000`	`0x5000` – `0x5FFF`	Virtual interface control (processor-specific base address)
`0x6000`	`0x6000` – `0x7FFF`	Virtual CPU interface

Note

The GIC provides two ways to access the GIC virtual interface control registers:

- A single common base address for the GIC virtual interface control registers. Each processor can access its own GIC virtual interface control registers through this base address. This base address is at offset `0x4000` relative to the **PERIPHBASE** address.
- A different base address for each processor to access the GIC virtual interface control registers. Any processor can use these addresses to access its own GIC virtual interface control registers, or to access the GIC virtual interface control registers of any other processor in the MPCore device. The starting base address is at offset `0x5000` relative to the **PERIPHBASE** address, with address bits[10:9] as the CPU ID decode.

8.2.3 Interrupt sources

The Cortex-A15 MPCore processor can support up to 256 interrupts. Each interrupt source is identified by a unique ID.

The Cortex-A15 MPCore processor has the following interrupt sources:

Software Generated Interrupts

SGIs are generated by writing to the Software Generated Interrupt Register (GICD_SGIR). A maximum of 16 SGIs, ID0-ID15, can be generated for each CPU interface. An SGI has edge-triggered properties. The software triggering of the interrupt is equivalent to the edge transition of the interrupt signal on a peripheral input.

Private Peripheral Interrupts

A PPI is an interrupt generated by a peripheral that is specific to a single processor. There are seven PPIs for each CPU interface:

Legacy nFIQ signal (PPI0)

In legacy FIQ mode, the external **nFIQ** signal bypasses the interrupt distributor logic and directly drives the interrupt request to the corresponding processor.

When a processor uses the GIC rather than the external **nFIQ** signal in legacy mode, by enabling its own CPU interface, the **nFIQ** signal is treated like other interrupt lines and uses ID28. The interrupt is active-LOW level-sensitive.

Secure Physical Timer event (PPI1)

This is the event generated from the Secure physical timer and uses ID29. The interrupt is active-LOW level-sensitive.

Non-secure Physical Timer event (PPI2)

This is the event generated from the Non-secure physical timer and uses ID30. The interrupt is active-LOW level-sensitive.

Legacy nIRQ signal (PPI3)

In legacy IRQ mode, the external **nIRQ** signal bypasses the interrupt distributor logic and directly drives the interrupt request to the corresponding processor.

When a processor uses the GIC rather than the external **nIRQ** signal in legacy mode, by enabling its own CPU interface, the **nIRQ** signal is treated like other interrupt lines and uses ID31. The interrupt is active-LOW level-sensitive.

Virtual Timer event (PPI4)

This is the event generated from the virtual timer and uses ID27. The interrupt is active-LOW level-sensitive.

Hypervisor Timer event (PPI5)

This is the event generated from the physical timer in Hypervisor mode and uses ID26. The interrupt is active-LOW level-sensitive.

Virtual Maintenance Interrupt (PPI6)

The Virtualization Extensions support in the *ARM Generic Interrupt Controller Architecture Specification* permits for a maintenance interrupt to be generated under several conditions. The virtual maintenance interrupt uses ID25. The interrupt is active-HIGH level-sensitive.

Shared Peripheral Interrupts

SPIs are triggered by events generated on associated interrupt input lines. The GIC can support up to 224 SPIs corresponding to the external **IRQS[223:0]** signal. The number of SPIs available depends on the implemented configuration of the Cortex-A15 MPCore processor. The permitted values are 0, 32, 64, 96, 128, 160, 192, or 224. SPIs start at ID32. The SPIs can be configured to be edge-triggered or active-HIGH level-sensitive.

8.2.4 Interrupt priority levels

The Cortex-A15 MPCore processor implements a 5-bit version of the interrupt priority field for 32 interrupt priority levels in Secure state.

8.2.5 GIC configuration

Table 8-2 lists the configurable options for the GIC.

Table 8-2 GIC configurable options

Feature	Range of options
Generic Interrupt Controller	Included or Not
Shared Peripheral Interrupts	0 to 224, in steps of 32

Bit[23] of the L2 Control Register indicates if the GIC is present or not, in the configuration. See *L2 Control Register* on page 4-85 for more information.

If you configure the design to exclude the GIC, SPIs and the remaining GIC signals are not available, except **PERIPHBASE[39:15]**. **PERIPHBASE[39:15]** are retained, and the value can be read in the Configuration Base Address Register, to permit software to read the location of the GIC if it exists in the system external to the Cortex-A15 MPCore processor. See *Configuration Base Address Register* on page 4-106.

The Cortex-A15 MPCore processor always includes the virtual interrupt signals, **nVIRQ** and **nVFIQ**, regardless of whether the GIC is present or not. There is one **nVIRQ** and one **nVFIQ** for each processor. If you configure the processor to exclude the GIC, the input pins **nVIRQ** and **nVFIQ** can be tied off to HIGH if unused, or can be driven by an external GIC in the SoC.

If you configure the processor to include the GIC, and the GIC is used, the input pins **nVIRQ** and **nVFIQ** must be tied off to HIGH. This is because the internal GIC generates the virtual interrupt signals to the processors. If you configure the processor to include the GIC, and the GIC is not used, the input pins **nVIRQ** and **nVFIQ** can be driven by an external GIC in the SoC.

8.3 GIC programmers model

This section describes the programmers model for the Cortex-A15 MPCore GIC in:

- *Distributor register summary*.
- *Distributor register descriptions* on page 8-9.
- *CPU interface register summary* on page 8-15.
- *CPU interface register descriptions* on page 8-16.
- *Virtual interface control register summary* on page 8-18.
- *Virtual interface control register descriptions* on page 8-19.
- *Virtual CPU interface register summary* on page 8-20.
- *Virtual CPU interface register descriptions* on page 8-21.

8.3.1 Distributor register summary

The Distributor centralizes all interrupt sources, determines the priority of each interrupt, and for each CPU interface dispatches the interrupt with the highest priority to the interface for priority masking and preemption handling.

The Distributor provides a programming interface for:

- Globally enabling the forwarding of interrupts to the CPU interfaces.
- Enabling or disabling each interrupt.
- Setting the priority level of each interrupt.
- Routing each interrupt to its target processor.
- Setting each shared peripheral interrupt to be level-sensitive or edge-triggered.
- Setting each interrupt as either Group 0 or Group 1, see the *ARM Generic Interrupt Controller (GIC) Architecture Specification* for information on interrupt grouping
- Forwarding an SGI to one or more target processors.
- Visibility of the state of each interrupt.
- A mechanism for software to set or clear the pending state of a peripheral interrupt.

In addition, the Distributor provides:

- Visibility of the state of each interrupt.
- A mechanism for software to set or clear the pending state of a peripheral interrupt.

The Distributor provides the ability to set interrupts as either:

- Secure Group 0.
- Non-secure Group 1.
- Secure Group 1.

See the *ARM Generic Interrupt Controller (GIC) Architecture Specification* for information on interrupt grouping.

Table 8-3 on page 8-8 shows the register map for the Distributor. The offsets in this table are relative to the Distributor block base address as shown in Table 8-1 on page 8-4.

The GICD_IPRIORITYRn, GICD_ITARGETSRn, GICD_CPENDSGIR and GICD_SPENDSGIR registers are byte-accessible and word-accessible. All other registers in Table 8-3 are word-accessible. Registers not described in Table 8-3 are RAZ/WI.

Table 8-3 Distributor register summary

Offset	Name	Type	Reset	Description
`0x000`	GICD_CTLR	RW[a]	`0x00000000`	Distributor Control Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x004`	GICD_TYPER	RO	Configuration dependent	*Interrupt Controller Type Register* on page 8-10
`0x008`	GICD_IIDR	RO	`0x0000043B`	*Distributor Implementer Identification Register* on page 8-11
`0x080 - 0x09C`	GICD_IGROUPRn	RW[b]	`0x00000000`	Interrupt Group Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x100`	GICD_ISENABLERn	RW	`0x0000FFFF`[c]	Interrupt Set-Enable Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x104 - 0x11C`			`0x00000000`	
`0x180`	GICD_ICENABLERn	RW	`0x0000FFFF`[c]	Interrupt Clear-Enable Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x184 - 0x19C`			`0x00000000`	
`0x200 - 0x21C`	GICD_ISPENDRn	RW	`0x00000000`	Interrupt Set-Pending Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x280 - 0x29C`	GICD_ICPENDRn	RW	`0x00000000`	Interrupt Clear-Pending Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x300 - 0x31C`	GICD_ISACTIVERn	RW	`0x00000000`	Interrupt Set-Active Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x380 - 0x39C`	GICD_ICACTIVERn	RW	`0x00000000`	Interrupt Clear-Active Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x400 - 0x4FC`	GICD_IPRIORITYRn	RW[d]	`0x00000000`	Interrupt Priority Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x800 - 0x81C`	GICD_ITARGETSRn[e]	RO	Configuration dependent	Interrupt Processor Targets Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x820 - 0x8FC`		RW[f]	`0x00000000`	
`0xC00`	GICD_ICFGRn	RO	`0xAAAAAAAA`[g]	*Interrupt Configuration Register* on page 8-12
`0xC04`		RO	`0x55540000`[g]	
`0xC08 - 0xC3C`		RW	`0x55555555`[g]	
`0xD00`	GICD_PPISR	RO	`0x00000000`	*Private Peripheral Interrupt Status Register* on page 8-13
`0xD04 -0xD1C`	GICD_SPISRn	RO	`0x00000000`	*Shared Peripheral Interrupt Status Registers* on page 8-13
`0xF00`	GICD_SGIR[h]	WO	-	Software Generated Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*

`0xF10 - 0xF1C`	GICD_CPENDSGIRn	RW	`0x00000000`	SGI Clear-Pending Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0xF20 - 0xF2C`	GICD_SPENDSGIRn	RW	`0x00000000`	SGI Set-Pending Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0xFD0`	GICD_PIDR4	RO	`0x04`	Peripheral ID4 Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0xFD4`	GICD_PIDR5	RO	`0x00`	Peripheral ID5 Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0xFD8`	GICD_PIDR6	RO	`0x00`	Peripheral ID6 Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0xFDC`	GICD_PIDR7	RO	`0x00`	Peripheral ID7 Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0xFE0`	GICD_PIDR0	RO		

8.3.2 Distributor register descriptions

This section only describes registers whose implementation is specific to the Cortex-A15 MPCore processor. All other registers are described in the *ARM Generic Interrupt Controller Architecture Specification*. Table 8-3 on page 8-8 provides cross references to individual registers.

Interrupt Controller Type Register

The GICD_TYPER characteristics are:

Purpose Provides information about the configuration of the GIC. It indicates:

- Whether the GIC implements the Security Extensions.
- The maximum number of interrupt IDs that the GIC supports.
- The maximum number of CPU interfaces implemented.
- The number of implemented *Lockable Shared Peripheral Interrupts* (LSPIs).

Usage constraints There are no usage constraints.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-3 on page 8-8.

Figure 8-1 shows the GICD_TYPER bit assignments.

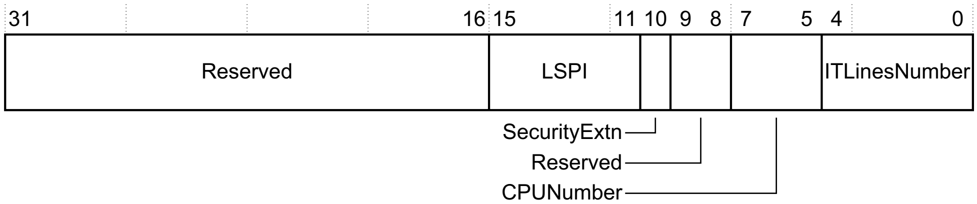


Figure 8-1 GICD_TYPER bit assignments

Table 8-4 shows the GICD_TYPER bit assignments.

Table 8-4 GICD_TYPER bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RAZ.
[15:11]	LSPI	Returns the number of *Lockable Shared Peripheral Interrupts* (LSPIs) that the GIC contains: `b11111` 31 LSPIs. These are the interrupts of IDs 32-62. When **CFGSDISABLE** is asserted, the GIC prevents writes to any register locations that control the operating state of an LSPI.
[10]	SecurityExtn	Indicates whether the GIC implements the Security Extensions. This bit always returns a value of 1, indicating that the Security Extensions are implemented.

Table 8-4 GICD_TYPER bit assignments (continued)

Bits	Name	Function
[9:8]	-	Reserved, RAZ.
[7:5]	CPUNumber	Indicates the number of implemented processors: `b000` The Cortex-A15 MPCore configuration contains one processor. `b001` The Cortex-A15 MPCore configuration contains two processors. `b010` The Cortex-A15 MPCore configuration contains three processors. `b011` The Cortex-A15 MPCore configuration contains four processors. All other values are reserved for future expansions.
[4:0]	ITLinesNumber	Indicates the number of interrupts that the GIC supports: `b00000` Up to 32 interrupts[a], no external interrupt lines. `b00001` Up to 64 interrupts, 32 external interrupt lines. `b00010` Up to 96 interrupts, 64 external interrupt lines. `b00011` Up to 128 interrupts, 96 external interrupt lines. `b00100` Up to 160 interrupts, 128 external interrupt lines. `b00101` Up to 192 interrupts, 160 external interrupt lines. `b00110` Up to 224 interrupts, 192 external interrupt lines. `b00111` Up to 256 interrupts, 224 external interrupt lines. All other values are reserved for future expansions.

[a] The Distributor always uses interrupts of IDs 0 to 31 to control any SGIs and PPIs that the GIC might contain.

Distributor Implementer Identification Register

The GICD_IIDR characteristics are:

Purpose Provides information about the implementer and revision of the Distributor.

Usage constraints There are no usage constraints.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-3 on page 8-8.

Figure 8-2 shows the GICD_IIDR bit assignments.

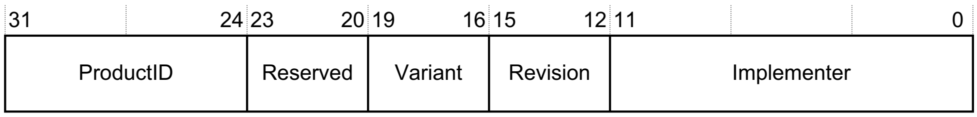


Figure 8-2 GICD_IIDR bit assignments

Table 8-5 shows the GICD_IIDR bit assignments.

Table 8-5 GICD_IIDR bit assignments

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID of the GIC: `0x00` Product ID.
[23:20]	-	Reserved, RAZ
[19:16]	Variant	Indicates the major revision number of the GIC: `0x0` Variant number.
[15:12]	Revision	Indicates the minor revision number of the GIC: `0x0` Revision number.
[11:0]	Implementer	Indicates the implementer: `0x43B` ARM implementation.

Interrupt Configuration Register

The GICD_ICFGR provides a 2-bit field that describes the configuration for each interrupt that the GIC supports.

The options for each bit-pair depend on the interrupt type as follows:

SGI The bits are read-only and a bit-pair always reads as b10 because SGIs are edge-triggered.

PPI The bits are read-only and a bit-pair always reads as b01. Table 8-6 shows that the PPIs are implemented as level-sensitive.

Table 8-6 PPI implementation

Interrupt	Name	Level-sensitive
PPI6	Virtual Maintenance interrupt	active-HIGH
PPI5	Hypervisor timer event	active-LOW
PPI4	Virtual timer event	active-LOW
PPI3	Legacy **nIRQ** pin	active-LOW
PPI2	Non-secure physical timer event	active-LOW
PPI1	Secure physical timer event	active-LOW
PPI0	Legacy **nFIQ** pin	active-LOW

SPI The *Least Significant Bit* (LSB) of the bit-pair is read-only and is always 1. You can program the *Most Significant Bit* (MSB) of the bit-pair to alter the triggering sensitivity as follows:

`b01` Interrupt is active-HIGH level-sensitive.

`b11` Interrupt is rising edge-sensitive.

Private Peripheral Interrupt Status Register

The GICD_PPISR characteristics are:

Purpose Enables a Cortex-A15 MPCore processor to access the status of the PPI inputs on the Distributor.

Usage constraints A processor can only read the status of its own PPI and therefore cannot read the status of the PPI for other processors.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-3 on page 8-8.

Figure 8-3 shows the GICD_PPISR bit assignments.

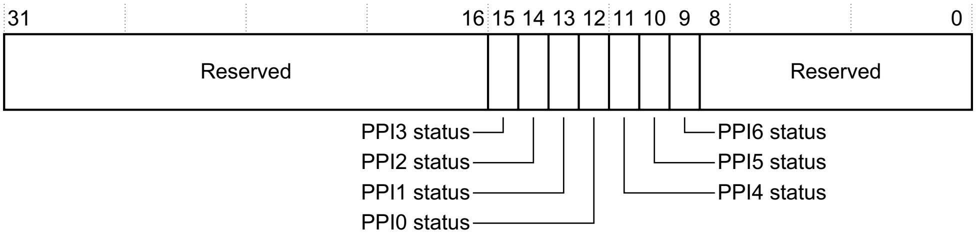


Figure 8-3 GICD_PPISR bit assignments

Table 8-7 shows the GICD_PPISR bit assignments.

Table 8-7 GICD_PPISR bit assignments

Bits	Name	Description
[31:16]	-	Reserved, RAZ
[15:9]	PPI status	Returns the status of the **PPI[6:0]** inputs on the Distributor: **PPI6** Virtual Maintenance Interrupt. **PPI5** Hypervisor timer event. **PPI4** Virtual timer event. **PPI3** nIRQ. **PPI2** Non-secure physical timer event. **PPI1** Secure physical timer event. **PPI0** nFIQ. **PPI0-5** Active-LOW level-sensitive. **PPI6** Active-HIGH level-sensitive. **Note** These bits return the actual status of the **PPI[6:0]** signals. The GICD_ISPENDRn and GICD_ICPENDRn can also provide the **PPI[6:0]** status but because you can write to these registers, they might not contain the true status of the **PPI[6:0]** signals.
[8:0]	-	Reserved, RAZ

Shared Peripheral Interrupt Status Registers

The GICD_SPISR characteristics are:

Purpose Enables a Cortex-A15 MPCore processor to access the status of the **IRQS[223:0]** inputs on the Distributor.

Usage constraints There are no usage constraints.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-3 on page 8-8.

Figure 8-4 shows the GICD_SPISR bit assignments.

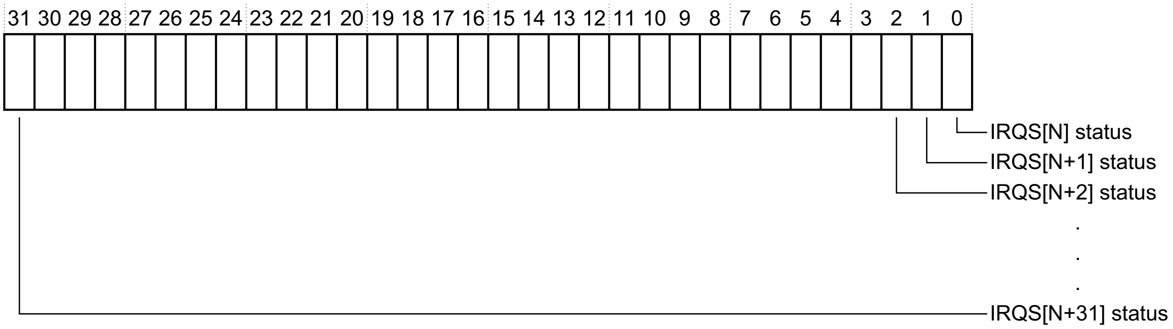


Figure 8-4 GICD_SPISR bit assignments

Table 8-8 shows the GICD_SPISR bit assignments.

Table 8-8 GICD_SPISR bit assignments

Bits	Name	Function
[31:0]	IRQS[N+31:N]	Returns the status of the **IRQS[223:0]** inputs on the Distributor. For each bit: **0** **IRQS** is LOW. **1** **IRQS** is HIGH. ——— **Note** ——— • The **IRQS** that a bit refers to depends on its bit position and the base address offset of the GICD_SPISR. • These bits return the actual status of the **IRQS** signals. The GICD_ISPENDRn and GICD_ICPENDRn can also provide the **IRQS** status but because you can write to these registers, they might not contain the actual status of the **IRQS** signals.

Figure 8-5 on page 8-15 shows the address map that the Distributor provides for the SPIs.

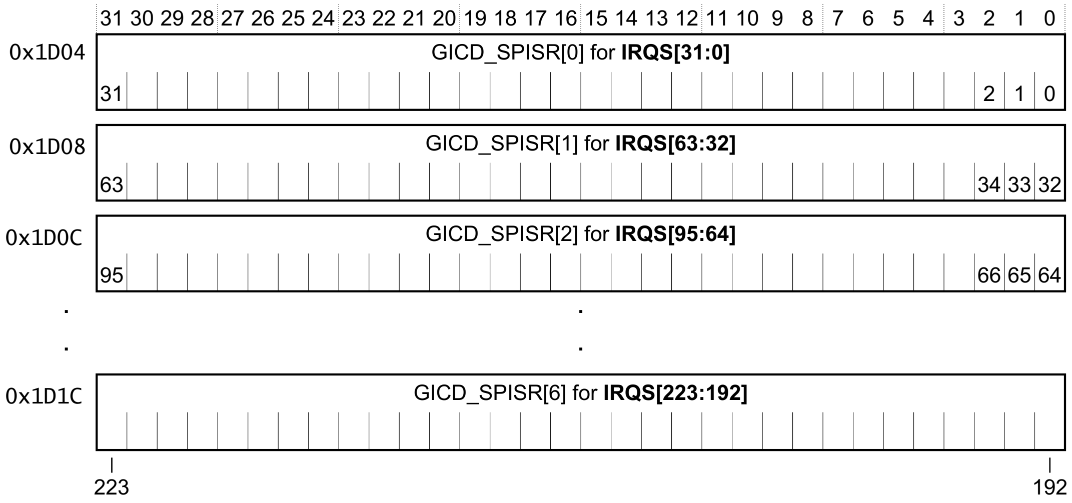


Figure 8-5 GICD_SPISR address map

The Distributor provides up to seven registers to support 224 SPIs. If the GIC is configured to use fewer than 224 SPIs, it reduces the number of registers accordingly. For locations where interrupts are not implemented, the bit is RAZ/WI.

8.3.3 CPU interface register summary

Each CPU interface block provides the interface for a Cortex-A15 MPCore processor that operates with the GIC. 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.

For more information on CPU interfaces, see the *ARM Generic Interrupt Controller Architecture Specification*.

Table 8-9 shows the register map for the CPU interface. The offsets in this table are relative to the CPU interface block base address as shown in Table 8-1 on page 8-4.

All the registers in Table 8-9 are word-accessible. Registers not described in this table are RAZ/WI.

Table 8-9 CPU interface register summary

Offset	Name	Type	Reset	Description
`0x0000`	GICC_CTLR	RW	`0x00000000`	CPU Interface Control Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0004`	GICC_PMR	RW	`0x00000000`	Interrupt Priority Mask Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0008`	GICC_BPR	RW	`0x00000002` (S)[a] `0x00000003` (NS)[b]	Binary Point Register, see *ARM Generic Interrupt Controller Architecture Specification*

Table 8-9 CPU interface register summary (continued)

Offset	Name	Type	Reset	Description
`0x000C`	GICC_IAR	RO	`0x000003FF`	Interrupt Acknowledge Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0010`	GICC_EOIR	WO	-	End Of Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0014`	GICC_RPR	RO	`0x000000FF`	Running Priority Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0018`	GICC_HPPIR	RO	`0x000003FF`	Highest Priority Pending Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x001C`	GICC_ABPR	RW[c]	`0x00000003`	Aliased Binary Point Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0020`	GICC_AIAR	RO[c]	`0x000003FF`	Aliased Interrupt Acknowledge Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0024`	GICC_AEOIR	WO[c]	-	Aliased End of Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0028`	GICC_AHPPIR	RO[c]	`0x000003FF`	Aliased Highest Priority Pending Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x00D0`	GICC_APR0	RW	`0x00000000`	*Active Priority Register*
`0x00E0`	GICC_NSAPR0	RW[c]	`0x00000000`	*Non-secure Active Priority Register* on page 8-17
`0x00FC`	GICC_IIDR	RO	`0x0002043B`	*CPU Interface Identification Register* on page 8-17
`0x1000`	GICC_DIR	WO	-	Deactivate Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*

a. S = Secure.

b. NS = Non-secure.

c. This register is only accessible from a Secure access.

8.3.4 CPU interface register descriptions

This section only describes registers whose implementation is specific to the Cortex-A15 MPCore processor. All other registers are described in the *ARM Generic Interrupt Controller Architecture Specification*. Table 8-9 on page 8-15 provides cross references to individual registers.

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 if the GIC is implemented.

Attributes See the register summary in Table 8-9 on page 8-15.

The Cortex-A15 MPCore processor implements the GICC_APR0 according to the recommendations described in the *ARM Generic Interrupt Controller Architecture Specification*.

Table 8-10 shows the Cortex-A15 MPCore GICC_APR0 implementation.

Table 8-10 Active Priority Register implementation

Number of group priority bits	Preemption levels	Minimum legal value of Secure GICC_BPR	Minimum legal value of Non-secure GICC_BPR	Active Priority Registers implemented	View of Active Priority Registers for Non-secure accesses
5	32	2	3	GICC_APR0 [31:0]	GICC_NSAPR0 [31:16] appears as GICC_APR0 [15:0]

Non-secure Active Priority Register

The GICC_NSAPR0 characteristics are:

Purpose Provides support for preserving and restoring state in power-management applications.

Usage constraints. This register is only accessible from a Secure access.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-9 on page 8-15.

The Cortex-A15 MPCore processor implements the GICC_NSAPR0 according to the recommendations described in the *ARM® Generic Interrupt Controller Architecture Specification*. It is consistent with the GICC_APR0 Register.

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 if the GIC is implemented.

Attributes See the register summary in Table 8-9 on page 8-15.

Figure 8-6 shows the GICC_IIDR bit assignments.

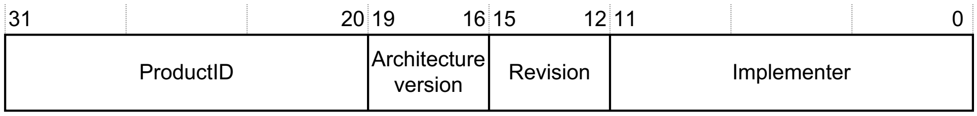


Figure 8-6 GICC_IIDR bit assignments

Table 8-11 shows the GICC_IIDR bit assignments.

Table 8-11 GICC_IIDR bit assignments

Bit	Name	Function
[31:20]	ProductID	Identifies the product: `0x000` Product ID.
[19:16]	Architecture version	Identifies the architecture version of the GIC: `0x2` Version 2.0.
[15:12]	Revision	Identifies the revision number for the CPU interface: `0x0` Revision 0.
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the CPU interface. For an ARM implementation, these values are: **Bits[11:8]** = `0x4` The JEP106 continuation code of the implementer. **Bit[7]** Always 0. **Bits[6:0]** = `0x3B` The JEP106 identity code of the implementer.

8.3.5 Virtual interface control register summary

The virtual interface control registers are management registers. Configuration software on the Cortex-A15 MPCore processor must ensure they are accessible only by a hypervisor, or similar software.

Table 8-12 shows the register map for the virtual interface control registers. The offsets in this table are relative to the virtual interface control registers block base address as shown in Table 8-1 on page 8-4.

All the registers in Table 8-12 are word-accessible. Registers not described in this table are RAZ/WI.

Table 8-12 Virtual interface control register summary

Offset	Name	Type	Reset	Description
`0x000`	GICH_HCR	RW	`0x00000000`	Hypervisor Control Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x004`	GICH_VTR	RO	`0x90000003`	*VGIC Type Register* on page 8-19
`0x008`	GICH_VMCR	RW	`0x004C0000`	Virtual Machine Control Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x010`	GICH_MISR	RO	`0x00000000`	Maintenance Interrupt Status Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x020`	GICH_EISR0	RO	`0x00000000`	End of Interrupt Status Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x030`	GICH_ELSR0	RO	`0x0000000F`	Empty List Register Status Registers, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0F0`	GICH_APR	RW	`0x00000000`	Active Priorities Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x100`	GICH_LR0	RW	`0x00000000`	List Register 0, see *ARM Generic Interrupt Controller Architecture Specification*

Table 8-12 Virtual interface control register summary (continued)

Offset	Name	Type	Reset	Description
`0x104`	GICH_LR1	RW	`0x00000000`	List Register 1, see *ARM Generic Interrupt Controller Architecture Specification*
`0x108`	GICH_LR2	RW	`0x00000000`	List Register 2, see *ARM Generic Interrupt Controller Architecture Specification*
`0x10C`	GICH_LR3	RW	`0x00000000`	List Register 3, see *ARM Generic Interrupt Controller Architecture Specification*

8.3.6 Virtual interface control register descriptions

This section only describes registers whose implementation is specific to the Cortex-A15 MPCore processor. All other registers are described in the *ARM® Generic Interrupt Controller Architecture Specification*. Table 8-12 on page 8-18 provides cross references to individual registers.

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 if the GIC is implemented.

Attributes See the register summary in Table 8-12 on page 8-18.

Figure 8-7 shows the GICH_VTR bit assignments.

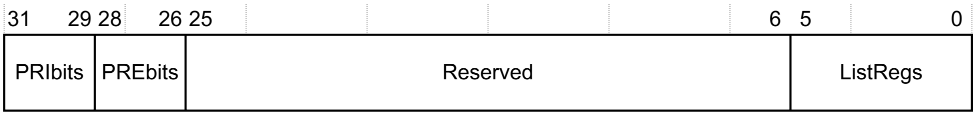


Figure 8-7 GICH_VTR bit assignments

Table 8-13 shows the GICH_VTR bit assignments.

Table 8-13 GICH_VTR bit assignments

Bit	Name	Description
[31:29]	PRIbits	Indicates the number of priority bits implemented, minus one: `0x4` Five bits of priority and 32 priority levels.
[28:26]	PREbits	Indicates the number of preemption bits implemented, minus one: `0x4` Five bits of preemption and 32 preemption levels.
[25:6]	-	Reserved, RAZ.
[5:0]	ListRegs	Indicates the number of implemented List Registers, minus one: `0x3` Four List Registers.

8.3.7 Virtual CPU interface register summary

The virtual CPU interface forwards virtual interrupts to a connected Cortex-A15 MPCore processor, subject to the normal GIC handling and prioritization rules. The virtual interface control registers control virtual CPU interface operation, and in particular, the virtual CPU interface uses the contents of the List registers to determine when to signal virtual interrupts. When a processor accesses the virtual CPU interface, the List registers are updated. For more information on virtual CPU interface, see the *ARM Generic Interrupt Controller Architecture Specification*.

Table 8-14 shows the register map for the virtual CPU interface. The offsets in this table are relative to the virtual CPU interface block base address as shown in Table 8-1 on page 8-4.

All the registers in Table 8-14 are word-accessible. Registers not described in this table are RAZ/WI.

Table 8-14 Virtual CPU interface register summary

Offset	Name	Type	Reset	Description
`0x0000`	GICV_CTLR	RW	`0x00000000`	VM Control Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0004`	GICV_PMR	RW	`0x00000000`	VM Priority Mask Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0008`	GICV_BPR	RW	`0x00000002`	VM Binary Point Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x000C`	GICV_IAR	RO	`0x000003FF`	VM Interrupt Acknowledge Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0010`	GICV_EOIR	WO	-	VM End Of Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0014`	GICV_RPR	RO	`0x000000FF`	VM Running Priority Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0018`	GICV_HPPIR	RO	`0x000003FF`	VM Highest Priority Pending Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x001C`	GICV_ABPR	RW	`0x00000003`	VM Aliased Binary Point Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0020`	GICV_AIAR	RO	`0x000003FF`	VM Aliased Interrupt Acknowledge Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0024`	GICV_AEOIR	WO	-	VM Aliased End of Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x0028`	GICV_AHPPIR	RO	`0x000003FF`	VM Aliased Highest Priority Pending Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*
`0x00D0`	GICV_APR0	RW	`0x00000000`	*VM Active Priority Register* on page 8-21
`0x00FC`	GICV_IIDR	RO	`0x0002043B`	*VM CPU Interface Identification Register* on page 8-21
`0x1000`	GICV_DIR	WO	-	VM Deactivate Interrupt Register, see *ARM Generic Interrupt Controller Architecture Specification*

8.3.8 Virtual CPU interface register descriptions

This section only describes registers whose implementation is specific to the Cortex-A15 MPCore processor. All other registers are described in the *ARM® Generic Interrupt Controller Architecture Specification*. Table 8-14 on page 8-20 provides cross references to individual registers.

VM Active Priority Register

The GICV_APR0 characteristics are:

Purpose For software compatibility, this register is present in the virtual CPU interface. However, in virtualized system, it is not used in the 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 if the GIC is implemented.

Attributes See the register summary in Table 8-14 on page 8-20.

The Cortex-A15 MPCore processorprocessor implements the GICV_APR0 as an alias of GICH_APR.

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 if the GIC is implemented.

Attributes See the register summary in Table 8-14 on page 8-20.

The bit assignments for the VM CPU Interface Identification Register are identical to the corresponding register in the CPU interface, see *CPU Interface Identification Register* on page 8-17.

Chapter 9
Generic Timer

This chapter describes the Generic Timer for the Cortex-A15 MPCore processor. It contains the following sections:

9.1 About the Generic Timer

The Generic Timer can trigger events after a period of time has elapsed. It provides:

- A physical counter that contains the count value of the system counter.
- A virtual counter that indicates virtual time.
- Generation of timer events as interrupt outputs.
- Generation of event streams.
- Support for Virtualization Extensions.

The Cortex-A15 MPCore Generic Timer is compliant with the ARMv7-A architecture.

This chapter only describes features that are specific to the Cortex-A15 MPCore implementation.

9.2 Generic Timer functional description

The Cortex-A15 MPCore Generic Timer includes:

- A physical counter that contains the count value of the system counter.
- A virtual counter that indicates virtual time. The virtual counter contains the value of the physical counter minus a 64-bit virtual offset.
- A set of four timers, one for each processor in the MPCore device.

The four timers are of the following:

- A PL1 physical timer. The registers for the PL1 physical timer are banked to provide Secure and Non-secure copies.
- A PL2 physical timer.
- A virtual timer.

The Cortex-A15 MPCore processor does not include the system counter which resides in the SoC. The system counter value is distributed to the Cortex-A15 MPCore processor with a synchronous binary encoded 64-bit bus, **CNTVALUEB[63:0]**.

Each timer provides an active-LOW interrupt output that is an external pin to the SoC and is sent to the GIC as a *Private Peripheral Interrupt* (PPI). See *Interrupt sources* on page 8-4 for the ID and PPI allocation of the Timer interrupts.

Table 9-1 shows the signals that are the external interrupt output pins.

Table 9-1 Generic Timer signals

Signal[a]	Description
nCNTNSIRQ[n:0]	Non-secure PL1 physical timer event
nCNTSIRQ[n:0]	Secure PL1 physical timer event
nCNTHPIRQ[n:0]	Non-secure PL2 physical timer event
nCNTVIRQ[n:0]	Virtual timer event

[a] **n** is the number of processors present in the MPCore device, minus one.

9.3 Generic Timer programmers model

Within each processor, a set of Generic Timer registers are allocated to the CP15 coprocessor space. The Generic Timer registers are either 32-bits wide or 64-bits wide.

Table 9-2 shows the Generic Timer registers.

Table 9-2 Generic Timer registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
CNTFRQ	c14	0	c0	0	UNK	32-bit	Counter Frequency Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTPCT	-	0	c14	-	UNK	64-bit	Physical Count Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTKCTL	c14	0	c1	0	-[a]	32-bit	Timer PL1 Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTP_TVAL			c2	0	UNK	32-bit	PL1 Physical TimerValue Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTP_CTL				1	-[b]	32-bit	PL1 Physical Timer Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTV_TVAL			c3	0	UNK	32-bit	Virtual TimerValue Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTV_CTL				1	[b]	32-bit	Virtual Timer Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTVCT	-	1	c14	-	UNK	64-bit	Virtual Count Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTP_CVAL		2			UNK	64-bit	PL1 Physical Timer CompareValue Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTV_CVAL		3			UNK	64-bit	Virtual Timer CompareValue Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTVOFF		4			UNK	64-bit	Virtual Offset Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTHCTL	c14	4	c1	0	-[c]	32-bit	Timer PL2 Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTHP_TVAL			c2	0	UNK	32-bit	PL2 Physical TimerValue Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTHP_CTL				1	[b]	32-bit	PL2 Physical Timer Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
CNTHP_CVAL	-	6	c14	-	UNK	64-bit	PL2 Physical Timer CompareValue Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*

[a] The reset value for bits[9:8, 2:0] is `b00000`.

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

c. The reset value for bit[2] is 0 and for bits[1:0] is b11.

10.1 About debug

This section gives an overview of debug and describes the debug components. The processor forms one component of a debug system. Figure 10-1 shows a typical system.

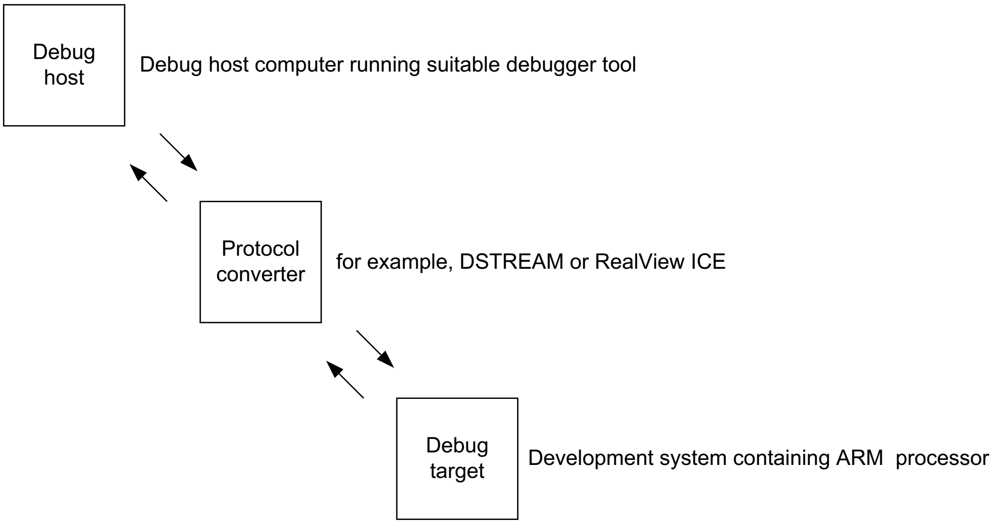

Debug host
Debug host computer running suitable debugger tool
Protocol converter
for example, DSTREAM or RealView ICE
Debug target
Development system containing ARM processor

Figure 10-1 Typical debug system

This typical system has several parts:

- *Debug host*.
- *Protocol converter*.
- *Debug target*.
- *The debug unit*.

10.1.1 Debug host

The debug host is a computer, for example a personal computer, running a software debugger such as RealView Debugger. The debug host enables you to issue high-level commands such as setting breakpoint at a certain location, or examining the contents of a memory address.

10.1.2 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 RealView ICE is required to convert between the two protocols.

10.1.3 Debug target

The debug target is the lowest level of the system. An example of a debug target is a development system with a test chip or a silicon part with a processor.

The debug target implements system support for the protocol converter to access the debug unit using the *Advanced Peripheral Bus* (APB) slave interface.

10.1.4 The debug unit

The processor debug unit assists in debugging software running on the processor. You can use the processor debug unit, in combination with a software debugger program, to debug:

- Application software.

- Operating systems.
- Hardware systems based on an ARM processor.

The debug unit enables you to:

- Stop program execution.
- Examine and alter process and coprocessor state.
- Examine and alter memory and input/output peripheral state.
- Restart the processor.

10.2 Debug register interfaces

The processor implements the ARMv7.1 Debug architecture and debug events as described in the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

The Debug architecture defines a set of debug registers. The debug register interfaces provide access to these registers from:

- Software running on the processor, see *Processor interfaces*.
- An external debugger, see *External debug interface* on page 10-35.

This section describes:

- *Processor interfaces*.
- *Breakpoints and watchpoints*.
- *Effects of resets on debug registers*.

10.2.1 Processor interfaces

The processor has the following interfaces to the debug, performance monitor, and trace registers:

Debug registers

This interface is Baseline CP14, Extended CP14, and memory-mapped. You can access the debug register map using the APB slave port. See *External debug interface* on page 10-35.

Performance monitor

This interface is CP15 based and memory-mapped. You can access the performance monitor registers using the APB slave port. See *External debug interface* on page 10-35.

Trace registers

This interface is memory-mapped. See *External debug interface* on page 10-35.

10.2.2 Breakpoints and watchpoints

The processor supports six hardware breakpoints, four watchpoints, and a standard *Debug Communications Channel* (DCC). Four of the breakpoints match only to virtual address and the other two match against either virtual address or context ID, or *Virtual Machine Identifier* (VMID). All the watchpoints can be linked to two breakpoints to enable a memory request to be trapped in a given process context.

10.2.3 Effects of resets on debug registers

The processor has the following reset signals that affect the debug registers:

nCPUPORESET

This signal initializes the processor logic, including the debug, *Program Trace Macrocell* (PTM), breakpoint, watchpoint logic, and performance monitors logic.

nDBGRESET

This signal resets the debug and PTM logic in the processor **CLK** domain, including the breakpoint and watchpoint logic. Performance monitors logic is not affected.

nPRESETDBG

This signal initializes the shared Debug APB, *Cross Trigger Interface* (CTI), and *Cross Trigger Matrix* (CTM) logic.

10.3 Debug register summary

Table 10-1 shows the 32-bit or 64-bit wide CP14 interface registers, accessed by the `MCR`, `MRC`, `MCCR`, or `MRRC` instructions in the order of CRn, Op1, CRm, Op2.

Table 10-1 CP14 debug register summary

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
0	`0x000`	c0	0	c0	0	DBGDIDR	RO	*Debug ID Register* on page 10-10
1-5	`0x004-0x014`	-	-	-	-	-	-	Reserved
6	`0x018`	c0	0	c6	0	DBGWFAR	RW	Watchpoint Fault Address Register, UNK/SBZ
7	`0x01C`	c0	0	c7	0	DBGVCR	RW	Vector Catch Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
8	`0x020`	-	-	-	-	-	-	Reserved
9	`0x024`	-	-	-	-	DBGECR	RW	Event Catch Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
10	`0x028`	c0	0	c10	0	-	-	Not implemented
11	`0x02C`	c0	0	c11	0	-	-	Not implemented
12-31	`0x030-0x07C`	-	-	-	-	-	-	Reserved
32	`0x080`	c0	0	c0	2	DBGDTRRX external view	RW	Host to Target Data Transfer, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
33	`0x084`	-	-	-	-	DBGITR	WO	Instruction Transfer Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
						DBGPCSR	RO	*Program Counter Sampling Register* on page 10-11
34	`0x088`	c0	0	c2	2	DBGDSCR external view	RW	Debug Status and Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
35	`0x08C`	c0	0	c3	2	DBGDTRTX external view	RW	Target to Host Transfer, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
36	`0x090`	-	-	-	-	DBGDRCR	WO	*Debug Run Control Register* on page 10-12

Table 10-1 CP14 debug register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
37	`0x094`	-	-	-	-	DBGEACR	RW	*Debug External Auxiliary Control Register* on page 10-13
38-39	`0x098-0x09C`	-	-	-	-	-	-	Reserved
40	`0x0A0`	-	-	-	-	DBGPCSR	RO	*Program Counter Sampling Register* on page 10-11
41	`0x0A4`	-	-	-	-	DBGCIDSR	RO	Context ID Sampling Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
42	`0x0A8`	-	-	-	-	DBGVIDSR	RO	Virtualization ID Sampling Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
43-63	`0x0AC-0x0FC`	-	-	-	-	-	-	Reserved
64-69	`0x100-0x114`	c0	0	c0-c5	4	DBGBVRn	RW	*Breakpoint Value Registers* on page 10-14
70-79	`0x118-0x13C`	-	-	-	-	-	-	Reserved
80-85	`0x140-0x154`	c0	0	c0-c5	5	DBGBCRn	RW	*Breakpoint Control Registers* on page 10-15
86-95	`0x158-0x17C`	-	-	-	-	-	-	Reserved
96-99	`0x180-0x18C`	c0	0	c0-c3	6	DBGWVRn	RW	*Watchpoint Value Registers* on page 10-17
100-111	`0x190-0x1BC`	-	-	-	-	-	-	Reserved
112-115	`0x1C0-0x1CC`	c0	0	c0-c3	7	DBGWCRn	RW	*Watchpoint Control Registers* on page 10-18
116-147	`0x1D0-0x24C`	-	-	-	-	-	-	Reserved
148-149	`0x250-0x254`	c1	0	c4-c5	1	DBGBXVRn	RW	*Breakpoint Extended Value Registers* on page 10-21
150-191	`0x258-0x2FC`	-	-	-	-	-	-	Reserved
192	`0x300`	c1	0	c0	4	DBGOSLAR	WO	*OS Lock Access Register* on page 10-22
193	`0x304`	c1	0	c1	4	DBGOSLSR	RO	*OS Lock Status Register* on page 10-23
194	`0x308`	-	-	-	-	-	-	Not implemented
195	`0x30C`	-	-	-	-	-	-	Reserved
196	`0x310`	c1	0	c4	4	DBGPRCR	RW	*Device Powerdown and Reset Control Register* on page 10-24

Table 10-1 CP14 debug register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
197	`0x314`	-	-	-	-	DBGPRSR	RO	Device Powerdown and Reset Status Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
198-831	`0x318-0xCFC`	-	-	-	-	-	-	Reserved
832-895	`0xD00-0xDFC`	-	-	-	-	Processor ID registers	RO	Processor ID registers, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
896-957	`0xE00-0xEF4`	-	-	-	-	-	-	Reserved
958	`0xEF8`	-	-	-	-	DBGITOCTRL	WO	*Integration Output Control Register* on page 10-27
959	`0xEFC`	-	-	-	-	DBGITISR	RO	*Integration Input Status Register* on page 10-28
960	`0xF00`	-	-	-	-	DBGITCTRL	RW	*Integration Mode Control Register* on page 10-28
961-999	`0xF04-0xF9C`	-	-	-	-	-	-	Reserved
1000	`0xFA0`	c7	0	c8	6	DBGCLAIMSET	RW	*Claim Tag Set Register* on page 10-29
1001	`0xFA4`	c7	0	c9	6	DBGCLAIMCLR	RW	*Claim Tag Clear Register* on page 10-30
1002-1003	`0xFA8-0xFAC`	-	-	-	-	-	-	Reserved
1004	`0xFB0`	-	-	-	-	DBGLAR	WO	Lock Access Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
1005	`0xFB4`	-	-	-	-	DBGLSR	RO	Lock Status Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
1006	`0xFB8`	c7	0	c14	6	DBGAUTHSTATUS	RO	Authentication Status Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
1007	`0xFBC`	-	-	-	-	-	-	Reserved
1008	`0xFC0`	c7	0	c0	7	DBGDEVID2	RO	UNK
1009	`0xFC4`	c7	0	c1	7	DBGDEVID1	RO	*Debug Device ID Register 1* on page 10-30
1010	`0xFC8`	c7	0	c2	7	DBGDEVID	RO	*Debug Device ID Register* on page 10-31

Table 10-1 CP14 debug register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
1011	`0xFCC`	-	-	-	-	DBGDEVTYPE	RO	Device Type Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
1012	`0xFD0`	-	-	-	-	DBGPID4	RO	*Peripheral Identification Registers* on page 10-32
1013-1015	`0xFD4-0xFDC`	-	-	-	-	DBGPID5-7	-	Reserved
1016	`0xFE0`	-	-	-	-	DBGPID0	RO	*Peripheral Identification Registers* on page 10-32
1017	`0xFE4`	-	-	-	-	DBGPID1	RO	
1018	`0xFE8`	-	-	-	-	DBGPID2	RO	
1019	`0xFEC`	-	-	-	-	DBGPID3	RO	
1020	`0xFF0`	-	-	-	-	DBGCID0	RO	*Component Identification Registers* on page 10-33
1021	`0xFF4`	-	-	-	-	DBGCID1	RO	
1022	`0xFF8`	-	-	-	-	DBGCID2	RO	
1023	`0xFFC`	-	-	-	-	DBGCID3	RO	
-	-	c0	0	c1	0	DBGDSCR internal view	RO	Debug Status and Control Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	-	c0	0	c5	0	DBGDTRRX internal view	WO	Host to Target Data Transfer, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
						DBGDTRTX internal view	RO	Target to Host Transfer, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	-	c1	0	c0	0	DBGDRAR (MRC)	RO	*Debug ROM Address Register* on page 10-20
-	-	c1	0	-	-	DBGDRAR (MRRC)	RO	*Debug ROM Address Register* on page 10-20
-	-	c1	0	c3	4	DBGOSDLR	RW	OS Double Lock Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
-	-	c2	0	c0	0	DBGDSAR (MRC)	RO	*Debug Self Address Offset Register* on page 10-26
-	-	c2	0	-	-	DBGDSAR (MRRC)	RO	*Debug Self Address Offset Register* on page 10-26

10.4 Debug register descriptions

This section describes the debug registers. Table 10-1 on page 10-6 provides cross references to individual registers.

10.4.1 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 There are no usage constraints. See the *Debug Device ID Register* on page 10-31 for more information about the debug implementation.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-2 shows the DBGDIDR bit assignments.

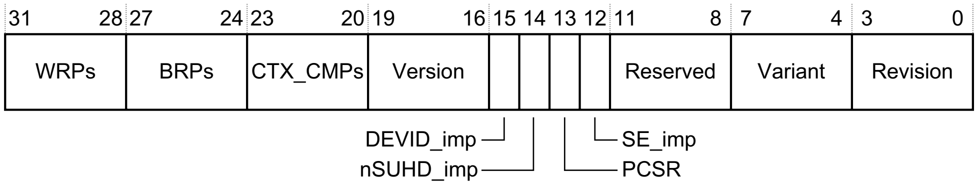


Figure 10-2 DBGDIDR bit assignments

Table 10-2 shows the DBGDIDR bit assignments.

Table 10-2 DBGDIDR bit assignments

Bits	Name	Function
[31:28]	WRPs	The number of *Watchpoint Register Pairs* (WRPs) implemented. The number of implemented WRPs is one more than the value of this field. **0x3** The processor implements 4 WRPs.
[27:24]	BRPs	The number of *Breakpoint Register Pairs* (BRPs) implemented. The number of implemented BRPs is one more than the value of this field. **0x5** The processor implements 6 BRPs.
[23:20]	CTX_CMPs	The number of BRPs that can be used for Context matching. This is one more than the value of this field. **0x1** The processor implements two Context matching breakpoints, that is, breakpoints 4 and 5.
[19:16]	Version	The Debug architecture version. **0x5** The processor implements ARMv7, v7.1 Debug architecture.
[15]	DEVID_imp	Debug Device ID Register bit. **1** DBGDEVID is implemented.
[14]	nSUHD_imp	Secure User Halting Debug not implemented bit. **1** The processor does not implement Secure User Halting Debug.
[13]	PCSR_imp	Program Counter Sampling Register (DBGPCSR) implemented as register 33 bit. **1** DBGPCSR is implemented as register 33.

Table 10-2 DBGDIDR bit assignments (continued)

Bits	Name	Function
[12]	SE_imp	Security Extensions implemented bit. **1** The processor implements Security Extensions.
[11:8]	-	Reserved.
[7:4]	Variant	This field indicates the variant number of the processor. This number is incremented on functional changes. The value matches bits[23:20] of the CP15 Main ID Register. See *Main ID Register* on page 4-27 for more information.
[3:0]	Revision	This field indicates the revision number of the processor. This number is incremented on bug fixes. The value matches bits[3:0] of the CP15 Main ID Register. See *Main ID Register* on page 4-27 for more information.

10.4.2 Program Counter Sampling Register

The DBGPCSR characteristics are:

Purpose Enables a debugger to sample the *Program Counter* (PC).

Usage constraints ARM deprecates reading a PC sample through register 33 when the DBGPCSR is also implemented as register 40. DBGPCSR is not visible in the CP14 interface.

Configurations DBGPCSR is implemented as both debug register 33 and 40.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-3 shows the DBGPCSR bit assignments.

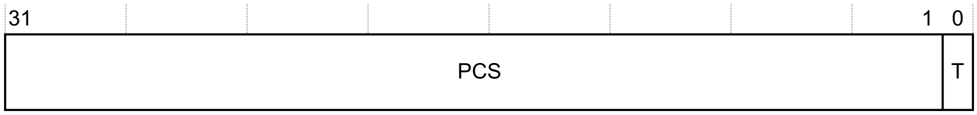


Figure 10-3 DBGPCSR bit assignments

Table 10-3 shows the DBGPCSR bit assignments.

Table 10-3 DBGPCSR bit assignments

Bits	Name	Function
[31:1]	PCS	Program Counter sample value. The sampled value of bits[31:1] of the PC. The sampled value is either the virtual address of an instruction, or the virtual address of an instruction address plus an offset that depends on the processor instruction set state. DBGDEVID1.PCSROffset indicates whether an offset is applied to the sampled addresses. For the Cortex-A15 MPCore processor, an offset is applied.
[0]	T	This bit indicates whether the sampled address is an ARM instruction, or a Thumb or ThumbEE instruction: **0** If DBGPCSR[1] is 0, the sampled address is an ARM instruction. **1** The sampled address is a Thumb or ThumbEE instruction. If T is 0 then DBGPCSR[1] is 0, ((DBGPCSR[31:2] << 2) - 8) is the address of the sampled ARM instruction. If T is 1, ((DBGPCSR[31:1] << 1) - 4) is the address of the sampled Thumb or ThumbEE instruction. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

10.4.3 Debug Run Control Register

The DBGDRCR characteristics are:

Purpose Software uses this register to:

- Request the processor to enter or exit Debug state.
- Clear to 0 the sticky exception bits in the DBGDSCR, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for information on the DBGDSCR.
- Clear to 0 DBGDSCR.PipeAdv, the Sticky Pipeline Advance bit.

Usage constraints DBGDRCR is not accessible in the CP14 interface.

Configurations Required in all implementations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-4 shows the DBGDRCR bit assignments.

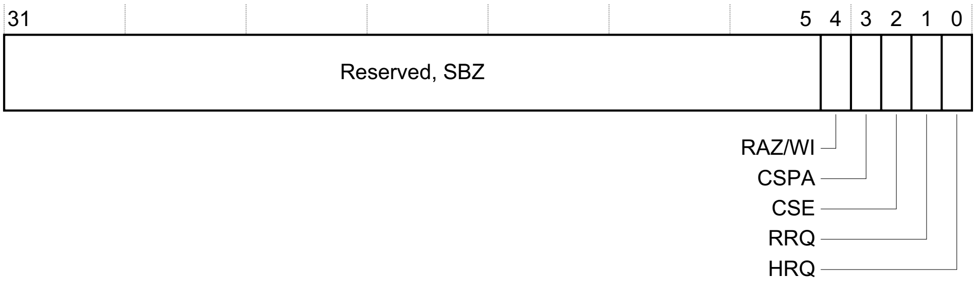


Figure 10-4 DBGDRCR bit assignments

Table 10-4 shows the DBGDRCR bit assignments.

Table 10-4 DBGDRCR bit assignments

Bits	Name	Function
[31:5]	-	SBZ.
[4]	-	RAZ/WI.
[3]	CSPA	Clear Sticky Pipeline Advanced. This bit clears the DBGDSCR.PipeAdv bit to 0. The actions on writing to this bit are: **0** No action. **1** Clears the DBGDSCR.PipeAdv bit to 0. When the processor is powered down, it is UNPREDICTABLE whether a write of 1 to this bit clears DBGDSCR.PipeAdv to 0.
[2]	CSE	Clear Sticky Exceptions. This bit clears the DBGDSCR sticky exceptions bits to 0. The actions on writing to this bit are: **0** No action. **1** Clears DBGDSCR[8:6] to 0b000. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information on the DBGDSCR.
[1]	RRQ	Restart request. The actions on writing to this bit are: **0** No action. **1** Request exit from Debug state. Writing 1 to this bit requests that the processor exits Debug state. This request is held until the processor exits Debug state. After the request has been made, the debugger can poll the DBGDSCR.RESTARTED bit until it reads as 1. The processor ignores writes to this bit if it is in Non-debug state.
[0]	HRQ	Halt request. The actions on writing to this bit are: **0** No action. **1** Request entry from Debug state. Writing 1 to this bit requests that the processor enters Debug state. This request is held until the processor enters Debug state. After the request has been made, the debugger can poll the DBGDSCR.HALTED bit until it reads 1. The processor ignores writes to this bit if it is already in Debug state.

10.4.4 Debug External Auxiliary Control Register

The DBGEACR characteristics are:

Purpose Provides implementation-defined configuration and control options.

Usage constraints The DBGEACR is not accessible from the CP14 interface.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-5 on page 10-14 shows the DBGEACR bit assignments.

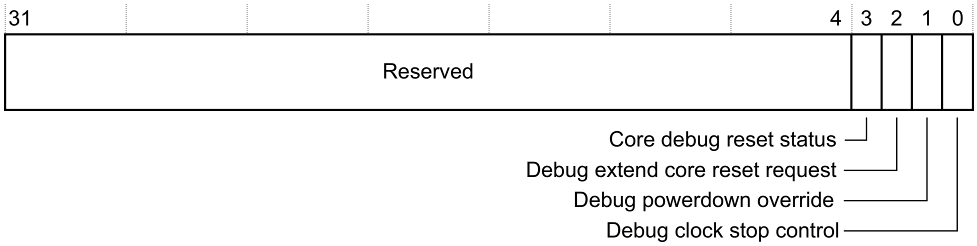


Figure 10-5 DBGEACR bit assignments

Table 10-5 shows the DBGEACR bit assignments.

Table 10-5 DBGEACR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RAZ/WI.
[3]	Core debug reset status	Read-only status bit that reflects the current reset state of the debug logic in the processor power domain: **0** Debug logic in processor power domain is not in reset state. **1** Debug logic in processor power domain is currently in reset state.
[2]	Debug extend core reset request	Debug core reset request extend bit. If debug is enabled and this bit is: **0** **DBGRSTREQ** is asserted for 16 cycles when 1'b1 is written to bit[1] of the Device Powerdown and Reset Control Register. See *Device Powerdown and Reset Control Register* on page 10-24. This is the reset value. **1** **DBGRSTREQ** is asserted for 64 cycles when 1'b1 is written to bit[1] of the Device Powerdown and Reset Control Register.
[1]	Debug powerdown override	Debug powerdown control bit. If debug is enabled and this bit is: **0** Error response is generated for APB accesses to the core domain debug registerswhen the core is powered down or OS Double Lock is set. This is the reset value. **1** APB accesses to the core domain debug registers proceed normally when the core is powered down or OS Double Lock is set.
[0]	Debug clock stop control	Debug clock control bit. If debug is enabled and this bit is: **0** Does not prevent the clock generator from stopping the processor clock. This is the reset value. **1** Prevents the clock generator from stopping the processor clock.

10.4.5 Breakpoint Value Registers

The DBGBVR characteristics are:

Purpose Holds a value for use in breakpoint matching, either the virtual address of an instruction or a Context ID. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information about the address used.

Usage constraints Used in conjunction with a DBGBCR, see *Breakpoint Control Registers* on page 10-15. Each DBGBVR is associated with a DBGBCR to form a breakpoint. DBGBVRn is associated with DBGBCRn to form breakpoint n.

Configurations The processor implements 6 BRPs, and is specified by the DBGDIDR.BRPs field, see *Debug ID Register* on page 10-10.

Attributes See the register summary in Table 10-1 on page 10-6. The debug logic reset value of a DBGBVR is UNK.

Figure 10-6 shows the DBGBVR bit assignments.

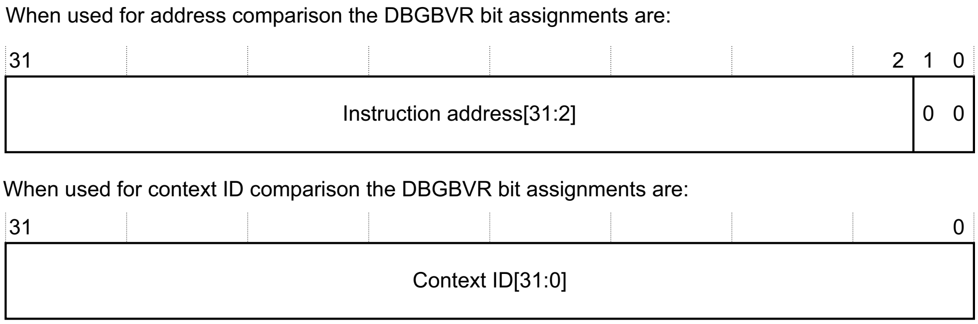

When used for address comparison the DBGBVR bit assignments are:
31
2 1 0
Instruction address[31:2]
0 0
When used for context ID comparison the DBGBVR bit assignments are:
31
0
Context ID[31:0]

Figure 10-6 DBGBVR bit assignments

Table 10-6 shows the DBGBVR bit assignments when the register is used for address comparison.

Table 10-6 DBGBVR bit assignments when register is used for address comparison

Bits	Name	Function
[31:2]	Instruction address[31:2]	This field indicates bits[31:2] of the virtual address value for comparison
[1:0]	-	This field must be written as `b00`, otherwise the generation of breakpoint debug events by this breakpoint is UNPREDICTABLE

Table 10-7 shows the DBGBVR bit assignments when the register is used for Context ID comparison.

Table 10-7 Breakpoint Value Register bit assignments when register is used for Context ID comparison

Bits	Name	Function
[31:0]	Context ID[31:0]	This field indicates bits[31:0] of the Context ID value for comparison

10.4.6 Breakpoint Control Registers

The DBGBCR characteristics are:

Purpose Holds control information for a breakpoint.

Usage constraints Used in conjunction with a DBGBVR, see *Breakpoint Value Registers* on page 10-14. Each DBGBVR is associated with a DBGBCR to form a *Breakpoint Register Pair* (BRP). DBGBVRn is associated with DBGBCRn to form breakpoint n.

Configurations The processor implements 6 BRPs, and is specified by the DBGDIDR.BRPs field, see *Debug ID Register* on page 10-10.

Attributes See the register summary in Table 10-1 on page 10-6. The debug logic reset value of a DBGBCR is UNKNOWN.

Figure 10-7 shows the DBGBCR bit assignments.

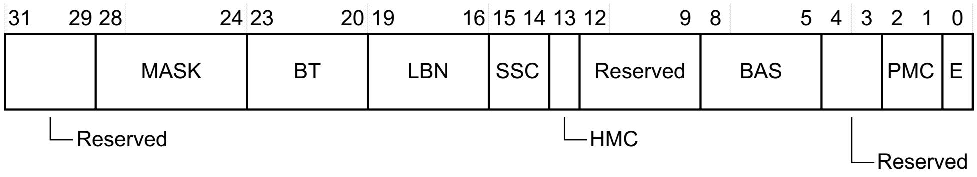


Figure 10-7 DBGBCR bit assignments

Table 10-8 shows the DBGBCR bit assignments.

Table 10-8 DBGBCR bit assignments

Bits	Name	Function
[31:29]	-	Reserved.
[28:24]	MASK	Address range mask. The processor does not support address range masking.
[23:20]	BT	Breakpoint Type. This field controls the behavior of debug event generation. This includes the meaning of the value held in the associated DBGBVR, indicating whether it is an instruction address match or mismatch or a Context match. It also controls whether the breakpoint is linked to another breakpoint. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for the breakpoint type values and meanings.
[19:16]	LBN	Linked Breakpoint Number. If this breakpoint is programmed for Linked instruction address match or mismatch then this field must be programmed with the number of the breakpoint that holds the Context match to be used in the combined instruction address and Context comparison. Otherwise, this field must be programmed to `b0000`. Reading this register returns an UNKNOWN value for this field, and the generation of debug events is UNPREDICTABLE, if either: • This breakpoint is not programmed for Linked instruction address match or mismatch and this field is not programmed to `b0000`. • This breakpoint is programmed for Linked instruction address match or mismatch and the breakpoint indicated by this field does not support Context matching or is not programmed for Linked Context matching. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[15:14]	SSC	Security State Control. This field enables the breakpoint to be conditional on the security state of the processor. This field is used with the *Hyp Mode Control* (HMC), and *Privileged Mode Control* (PMC), fields. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for possible values of the fields, and the mode and security states that can be tested. This field must be programmed to `b00` if DBGBCR.BT is programmed for Linked Context match. If this is not done, the generation of debug events by this breakpoint is UNPREDICTABLE. **Note** When this field is set to a value other than `b00`, the SSC field controls the processor security state in which the access matches, not the required security attribute of the access.
[13]	HMC	Hyp Mode Control bit. This field is used with the SSC and PMC fields. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for possible values of the fields, and the mode and security states that can be tested. This field must be programmed to 0 if DBGBCR.BT is programmed for Linked Context match. If this is not done, the generation of debug events by this breakpoint is UNPREDICTABLE.

Table 10-8 DBGBCR bit assignments (continued)

Bits	Name	Function
[12:9]	-	Reserved.
[8:5]	BAS	Byte Address Select. This field enables match or mismatch comparisons on only certain bytes of the word address held in the DBGBVR. The operation of this field depends also on: • The Breakpoint Type field being programmed for instruction address match or mismatch. • The MASK field being programmed to `b00000`, no mask. • The instruction set state of the processor, indicated by the CPSR.J and CPSR.T bits. This field must be programmed to `b1111` if either: • DBGBCR.BT is programmed for Linked or Unlinked Context ID match • DBGBCR.MASK is programmed to a value other than `b00000`. If this is not done, the generation of debug events by this breakpoint is UNPREDICTABLE.
[4:3]	-	Reserved.
[2:1]	PMC	Privileged Mode Control. This field enables breakpoint matching conditional on the mode of the processor. This field is used with the SSC and HMC fields. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for possible values of the fields, and the mode and security states that can be tested. This field must be programmed to `b11` if DBGBCR.BT is programmed for Linked Context ID match. If this is not done, the generation of debug events by this breakpoint is UNPREDICTABLE.
[0]	E	Breakpoint enable. This bit enables the breakpoint: **0** Breakpoint disabled. **1** Breakpoint enabled. A breakpoint never generates debug events when it is disabled.

10.4.7 Watchpoint Value Registers

The DBGWVR characteristics are:

Purpose Holds a data address value for use in watchpoint matching. The address used must be the virtual address of the data.

Usage constraints Used in conjunction with a DBGWCR to form a watchpoint. See *Watchpoint Control Registers* on page 10-18. Each DBGWVR is associated with a DBGWCR to form a *Watchpoint Register Pair* (WRP). DBGWVRn is associated with DBGWCRn to form watchpoint n.

Configurations The processor implements 4 WRPs, and is specified by the DBGDIDR.WRPs field, see *Debug ID Register* on page 10-10.

Attributes See the register summary in Table 10-1 on page 10-6. The debug logic reset value of a DBGWVR is UNKNOWN.

Figure 10-8 shows the DBGWVR bit assignments.

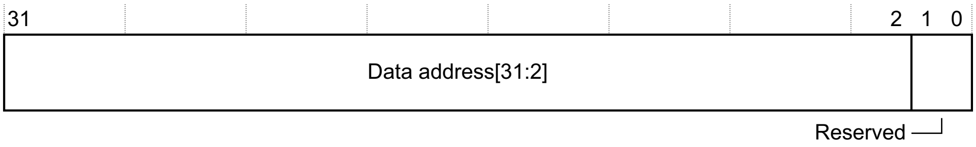


Figure 10-8 DBGWVR bit assignments

Table 10-9 shows the DBGWVR bit assignments.

Table 10-9 DBGWVR bit assignments

Bits	Name	Function
[31:2]	Data address[31:2]	This field indicates bits[31:2] of the value for comparison, address[31:2]
[1:0]	-	RAZ on reads and SBZP for writes

10.4.8 Watchpoint Control Registers

The DBGWCR characteristics are:

Purpose Holds control information for a watchpoint.

Usage constraints Used in conjunction with a DBGWVR, see *Watchpoint Value Registers* on page 10-17. Each DBGWCR is associated with a DBGWVR to form a *Watchpoint Register Pair* (WRP). DBGWCRn is associated with DBGWVRn to form watchpoint n.

Configurations The processor implements 4 WRPs, and is specified by the DBGDIDR.WRPs field, see *Debug ID Register* on page 10-10.

Attributes See the register summary in Table 10-1 on page 10-6. The debug logic reset value of a DBGWCR is UNKNOWN.

Figure 10-9 shows the DBGWCR bit assignments.

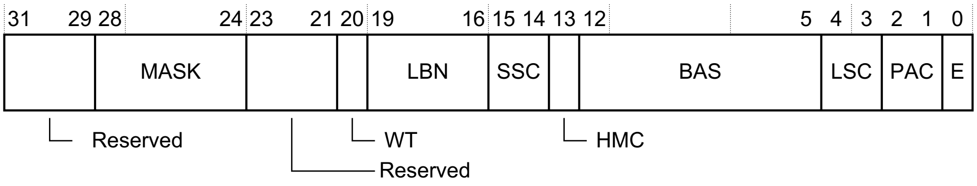


Figure 10-9 DBGWCR bit assignments

Table 10-10 shows the DBGWCR bit assignments.

Table 10-10 DBGWCR bit assignments

Bits	Name	Function
[31:29]	-	Reserved.
[28:24]	MASK	Address range mask. The processor supports watchpoint address range masking. This field can set a watchpoint on a range of addresses by masking lower order address bits out of the watchpoint comparison. The value of this field is the number of low order bits of the address that are masked off, except that values of 1 and 2 are reserved. Therefore, the meaning of Watchpoint Address range mask values are: `b00000` No mask. `b00001` Reserved. `b00010` Reserved. `b00011` `0x00000007` mask for data address, three bits masked. `b00100` `0x0000000F` mask for data address, four bits masked. `b00101` `0x0000001F` mask for data address, five bits masked. . . . `b11111` `0x7FFFFFFF` mask for data address, 31 bits masked.
[23:21]	-	Reserved.
[20]	WT	Watchpoint Type. This bit is set to 1 to link the watchpoint to a breakpoint to create a linked watchpoint that requires both data address matching and Context matching: **0** Unlinked data address match. **1** Linked data address match. When this bit is set to 1 the linked breakpoint number field indicates the breakpoint to which this watchpoint is linked. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[19:16]	LBN	Linked Breakpoint Number. If this watchpoint is programmed with the watchpoint type set to linked, then this field must be programmed with the number of the breakpoint that defines the Context match to be combined with data address comparison. Otherwise, this field must be programmed to `b0000`. Reading this register returns an UNKNOWN value for this field, and the generation of Watchpoint debug events is UNPREDICTABLE, if either: • This watchpoint does not have linking enabled and this field is not programmed to `b0000`. • This watchpoint has linking enabled and the breakpoint indicated by this field does not support Context matching, is not programmed for Context matching, or does not exist. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[15:14]	SSC	Security State Control. This field enables the watchpoint to be conditional on the security state of the processor. This field is used with the *Hyp Mode Control* (HMC) and *Privileged Access Control* (PAC) fields. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for possible values of the fields, and the access modes and security states that can be tested.
[13]	HMC	Hyp Mode Control. This field is used with the *Security State Control* (SSC) and PAC fields. The value of DBGWCR.PAC has no effect for accesses made in Hyp mode. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for possible values of the fields, and the access modes and security states that can be tested.

Table 10-10 DBGWCR bit assignments (continued)

Bits	Name	Function
[12:5]	BAS	Byte Address Select. The processor implements an 8-bit Byte address select field, DBGWCR[12:5]. A DBGWVR is programmed with a word-aligned address. This field enables the watchpoint to hit only if certain bytes of the addressed word are accessed. The watchpoint hits if an access hits any byte being watched, even if: • The access size is larger than the size of the region being watched. • The access is unaligned, and the base address of the access is not in the same word of memory as the address in the DBGWVR. • The access size is smaller than the size of region being watched. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[4:3]	LSC	Load/store access control. This field enables watchpoint matching on the type of access being made: `b00` Reserved. `b01` Match on any load, Load-Exclusive, or swap. `b10` Match on any store, Store-Exclusive or swap. `b11` Match on all type of access.
[2:1]	PAC	Privileged Access Control. This field enables watchpoint matching conditional on the mode of the processor. This field is used with the SSC and PAC fields. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for possible values of the fields, and the access modes and security states that can be tested.
[0]	E	Watchpoint enable. This bit enables the watchpoint: **0** Watchpoint disabled. **1** Watchpoint enabled. A watchpoint never generates a Watchpoint debug event when it is disabled. For more information about possible watchpoint values, see *Watchpoint Value Registers* on page 10-17.

10.4.9 Debug ROM Address Register

The DBGDRAR characteristics are:

Purpose Defines the base physical address of a memory-mapped debug component, usually a ROM Table that locates and describes the memory-mapped debug components in the system.

Usage constraints This register is only visible in the CP14 interface, and therefore does not have a memory offset.

Configurations The DBGDRAR is:

- a 64-bit register when accessed by the `MRRC` instruction
- a 32-bit register when accessed by the `MRC` instruction.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-10 on page 10-21 shows the DBGDRAR bit assignments as a 32-bit register.

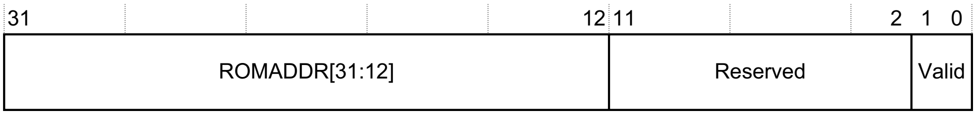


Figure 10-10 DBGDRAR 32-bit register bit assignments

Figure 10-11 shows the DBGDRAR bit assignments as a 64-bit register.

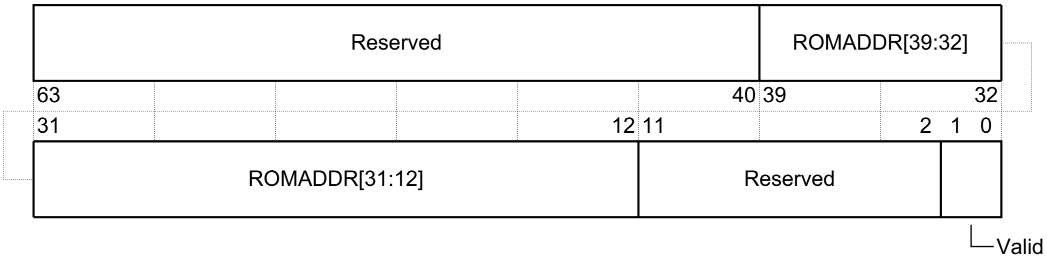


Figure 10-11 DBGDRAR 64-bit register bit assignments

Table 10-11 shows the DBGDRAR bit assignments.

Table 10-11 DBGDRAR bit assignments

Bits	Name	Function
[63:40]	-	Reserved.
[39:32]	ROMADDR[39:32]	Bits[39:32] of the debug component physical address. Bits[11:0] of the address are zero. If DBGDRAR.Valid is zero, the value of this field is UNKNOWN.
[31:12]	ROMADDR[31:12]	Bits[31:12] of the debug component physical address. Bits[11:0] of the address are zero. The ROMADDR[39:32] signal sets the values of the ROMADDR fields. If DBGDRAR.Valid is zero, the value of this field is UNKNOWN.
[11:2]	-	Reserved.
[1:0]	Valid	Valid bits. This field indicates whether the address is valid: `b00` Address is not valid. `b11` Address is valid. ——— **Note** ——— **ROMADDRV** must be set HIGH if **ROMADDR[39:12]** is set to a valid value.

10.4.10 Breakpoint Extended Value Registers

The DBGBXVR characteristics are:

Purpose Holds a value for use in breakpoint matching, to support VMID matching.

Usage constraints Used in conjunction with a DBGBCR and a DBGBVR. See *Breakpoint Control Registers* on page 10-15, and *Breakpoint Value Registers* on page 10-14. Each DBGBXVR is associated with a DBGBCR and DBGBVR to form a *Breakpoint Register Pair* (BRP). DBGBXVRn is associated with DBGBCRn and DBGBVRn to form breakpoint n.

Configurations The processor implements 2 BRPs that can be used for VMID matching, and is specified by the DBGDIDR.CTX_CMPs field, see *Debug ID Register* on page 10-10.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-12 shows the DBGBXVR bit assignments.

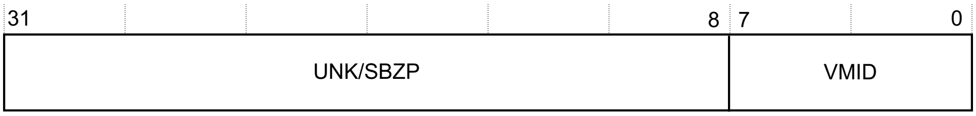


Figure 10-12 DBGBXVR bit assignments

Table 10-12 shows the DBGBXVR bit assignments.

Table 10-12 DBGBXVR bit assignments

Bits	Name	Function
[31:8]	-	UNK/SBZP.
[7:0]	VMID	VMID value. Compared VTTBR.VMID, the *Virtual Machine ID* (VMID) field. The debug logic generates a debug event when an instruction that matches the breakpoint is committed for execution. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

10.4.11 OS Lock Access Register

The DBGOSLAR characteristics are:

Purpose Provides a lock for the debug registers. The OS Lock also disables Software debug events.

Usage constraints Use DBGOSLSR to check the current status of the lock, see *OS Lock Status Register* on page 10-23.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-13 shows the DBGOSLAR bit assignments.

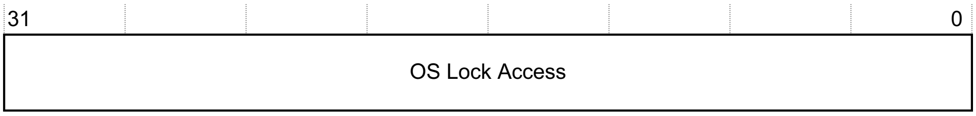


Figure 10-13 DBGOSLAR bit assignments

Table 10-13 shows the DBGOSLAR bit assignments.

Table 10-13 DBGOSLAR bit assignments

Bits	Name	Function
[31:0]	OS Lock Access	Writing the key value `0xC5ACCE55` to this field locks the debug registers. Writing any other value to this register unlocks the debug registers if they are locked. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

10.4.12 OS Lock Status Register

The DBGOSLSR characteristics are:

Purpose Provides status information for the OS Lock. Software can read this register to detect whether the OS Save and Restore mechanism is implemented. If it is not implemented, the read of DBGOSLSR.OSLM returns zero.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-14 shows the DBGOSLSR bit assignments.

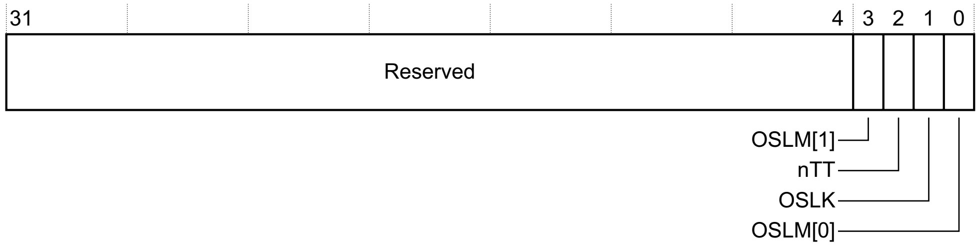


Figure 10-14 DBGOSLSR bit assignments

Table 10-14 shows the DBGOSLSR bit assignments.

Table 10-14 DBGOSLSR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, UNK.
[3]	OSLM[1]	OS Lock Model implemented bit. This field identifies the form of OS Save and Restore mechanism implemented: `b10` The processor implements the OS Lock Model but does not implement DBGOSSRR. **Note** This field splits across the two non-contiguous bits in the register.

Table 10-14 DBGOSLSR bit assignments (continued)

Bits	Name	Function
[2]	nTT	Not 32-bit access. This bit is always RAZ. It indicates that a 32-bit access is required to write the key to the OS Lock Access Register.
[1]	OSLK	This bit indicates the status of the OS Lock: **0** Lock not set. **1** Lock set. The OS Lock is set or cleared by writing to the DBGOSLAR, see *OS Lock Access Register* on page 10-22. The OS Lock is set to 1 on a core powerup reset. Setting the OS Lock restricts access to debug registers. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[0]	OSLM[0]	OS Lock Model implemented bit. This field identifies the form of OS Save and Restore mechanism implemented: `b10` The processor implements the OS Lock Model but does not implement DBGOSSRR. **Note** This field splits across the two non-contiguous bits in the register.

10.4.13 Device Powerdown and Reset Control Register

The DBGPRCR characteristics are:

Purpose Controls processor functionality related to reset and powerdown.

Usage constraints There are no usage constraints.

Configurations Required in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-15 shows the DBGPRCR bit assignments.

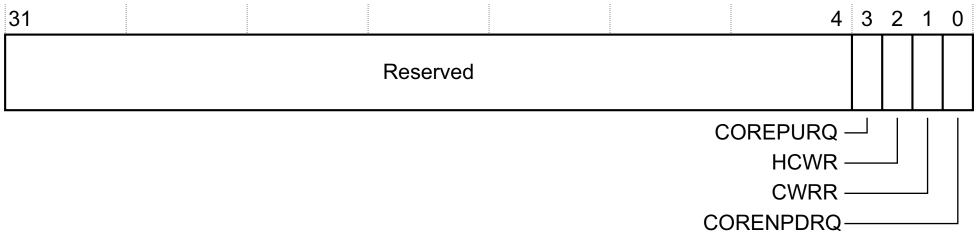


Figure 10-15 DBGPRCR bit assignments

Table 10-15 shows the DBGPRCR bit assignments.

Table 10-15 DBGPRCR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, UNK/SBZP.
[3]	COREPURQ	Core Powerup Request bit. This bit enables a debugger to request that the power controller powers up the core, enabling access to the debug registers in the core power domain: **0** **DBGPWRUPREQ** is LOW, this is the reset value. **1** **DBGPWRUPREQ** is HIGH. This bit is only defined for the memory-mapped and external debug interfaces. For accesses to DBGPRCR from CP14, this bit is UNK/SBZP. This bit can be read and written when the core power domain is powered down, and when DBGPRSR.DLK is set to 1. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[2]	HCWR	Hold Core Warm Reset bit. Writing 1 to this bit means the non-debug logic of the processor is held in reset after a core is powered up or a warm reset: **0** Does not hold the non-debug logic in reset on a power up or warm reset. **1** Holds the non-debug logic of the processor in reset on a power up or a warm reset. The processor is held in this state until this bit is cleared to 0. For accesses to DBGPRCR from CP14, this bit is UNK/SBZP. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[1]	CWRR	Core Warm Reset Request bit. Writing 1 to this bit issues a request for a warm reset: **0** No action. **1** Request internal reset by asserting the **DBGRSTREQ** output HIGH for 16 or 64 cycles. For more information, see bit[2] Debug extend core reset request of the DBGEACR, *Debug External Auxiliary Control Register* on page 10-13. Reads from this bit are UNKNOWN, and writes to this bit from the memory-mapped or external debug interface are ignored when any of the following apply: • The core power domain is powered down. • DBGPRSR.DLK, OS Double Lock status bit, is set to 1. • For the external debug interface, the OS lock is set. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[0]	CORENPDRQ	Core No Powerdown Request bit. When set to 1, the **DBGNOPWRDWN** output signal is HIGH. This output is connected to the system power controller and is interpreted as a request to operate in emulate mode. In this mode, the processor that includes PTM are not actually powered down when requested by software or hardware handshakes: **0** **DBGNOPWRDWN** is LOW. This is the reset value. **1** **DBGNOPWRDWN** is HIGH. This bit is UNKNOWN on reads and ignores writes when any of the following apply: • The core power domain is powered down. If the CORENPDRQ bit is 1, it loses this value through the powerdown. • DBGPRSR.DLK, OS Double Lock status bit is set to 1. • For the external debug interface, the OS Lock is set.

10.4.14 Debug Self Address Offset Register

The DBGDSAR characteristics are:

Purpose Defines the offset from the base address defined by DBGDRAR of the physical base address of the debug registers for the processor. See *Debug ROM Address Register* on page 10-20.

Usage constraints This register is only visible in the CP14 interface, and therefore does not have a memory offset.

Configurations The DBGDSAR is:

- a 64-bit register when accessed by the `MRRC` instruction
- a 32-bit register when accessed by the `MRC` instruction.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-16 shows the DBGDSAR bit assignments as a 32-bit register.

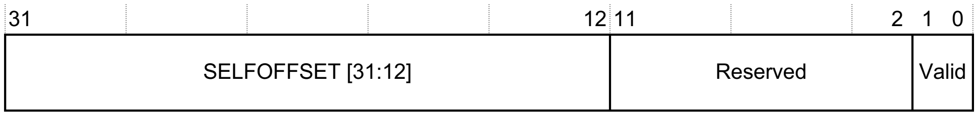


Figure 10-16 DBGDSAR 32-bit register bit assignments

Figure 10-17 shows the DBGDSAR bit assignments as a 64-bit register.

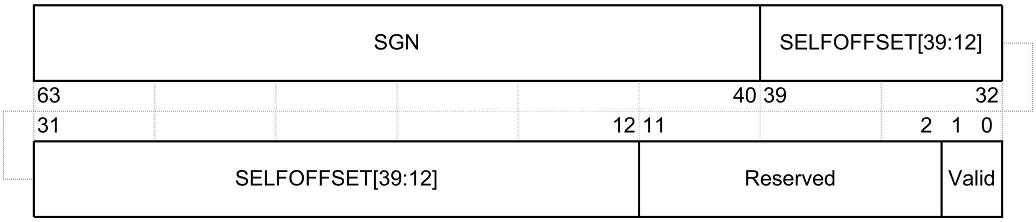


Figure 10-17 DBGDSAR 64-bit register bit assignments

Table 10-16 shows the DBGDSAR bit assignments.

Table 10-16 DBGDSAR bit assignments

Bits	Name	Function
[63:40]	SGN	Sign extension. Each bit must be the same as DBGDSAR[39].
[39:32]	SELFOFFSET[39:32]	Bits[39:32] of the two's complement offset from the base address defined by DBGDRAR to the physical address where the debug registers are mapped. Bits[11:0] of the address are zero. See *Debug ROM Address Register* on page 10-20. If DBGDSAR.Valid is zero, the value of this field is UNKNOWN.

Table 10-16 DBGDSAR bit assignments (continued)

Bits	Name	Function
[31:12]	SELFOFFSET[31:12]	Bits[31:12] of the two's complement offset from the base address defined by DBGDRAR to the physical address where the debug registers are mapped. Bits[11:0] of the address are zero. See *Debug ROM Address Register* on page 10-20. If DBGDSAR.Valid is zero, the value of this field is UNKNOWN.
[11:2]	-	Reserved.
[1:0]	Valid	Valid bit. This field indicates whether the debug self address offset is valid: `b00` Offset is not valid. `b11` Offset is valid.

10.4.15 Integration Output Control Register

The DBGITOCTRL characteristics are:

Purpose Controls signal outputs when bit[0] of the DBGITCTRL is set. See *Integration Mode Control Register* on page 10-28.

Usage constraints DBGITOCTRL is not accessible on the CP14 interface.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-18 shows the DBGITOCTRL bit assignments.

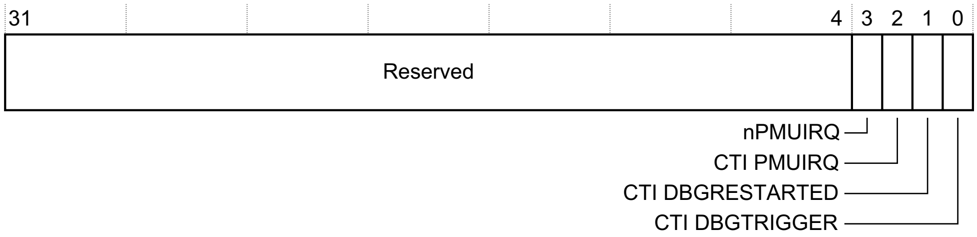


Figure 10-18 DBGITOCTRL bit assignments

Table 10-17 shows the DBGITOCTRL bit assignments.

Table 10-17 DBGITOCTRL bit assignments

Bits	Name	Function
[31:4]	-	Reserved, SBZP.
[3]	nPMUIRQ	This bit drives the **nPMUIRQ** output. When this bit is set to 1, the corresponding **nPMUIRQ** signal is cleared to 0. The reset value is 0.
[2]	CTI PMUIRQ	This bit drives the internal signal equivalent to **PMUIRQ** that goes from the *Performance Monitor Unit* (PMU) to the *Cross Trigger Interface* (CTI). The reset value is 0.
[1]	CTI DBGRESTARTED	This bit drives the internal signal that goes from the Debug unit to the CTI to acknowledge success of a debug restart command. The reset value is 0.
[0]	CTI DBGTRIGGER	This bit drives the internal signal equivalent to **DBGTRIGGER** that goes from the Debug unit to the CTI. The reset value is 0.

10.4.16 Integration Input Status Register

The DBGITISR characteristics are:

Purpose Enables the values of signal inputs to be read when bit[0] of the DBGITCTRL is set. See *Integration Mode Control Register*.

Usage constraints DBGITISR is not accessible on the extended CP14 interface.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-19 shows the DBGITISR bit assignments.

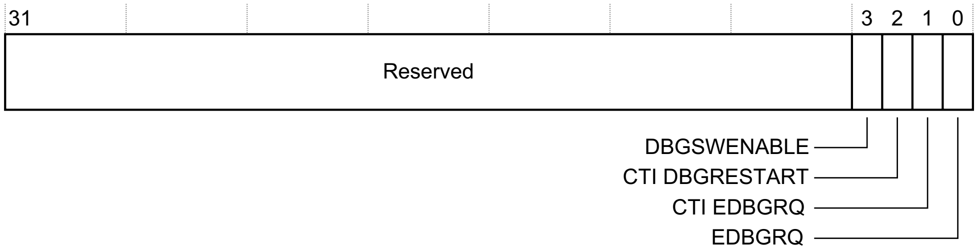


Figure 10-19 DBGITISR bit assignments

Table 10-18 shows the DBGITISR bit assignments.

Table 10-18 DBGITISR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RAZ.
[3]	DBGSWENABLE	This field reads the state of the **DBGSWENABLE** input.
[2]	CTI DBGRESTART	CTI debug restart bit. This field reads the state of the debug restart input coming from the CTI into the debug unit.
[1]	CTI EDBGRQ	CTI debug request bit. This field reads the state of the debug request input coming from the CTI into the debug unit.
[0]	EDBGRQ	This field reads the state of the **EDBGRQ** input.

10.4.17 Integration Mode Control Register

The DBGITCTRL characteristics are:

Purpose Switches the processor from its default functional mode into integration mode, where test software can control directly the inputs and outputs of the processor, for integration testing or topology detection. When the processor is in integration mode, the test software uses the integration registers to drive output values and to read inputs.

Usage constraints DBGITCTRL is not accessible on the extended CP14 interface.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-20 on page 10-29 shows the DBGITCTRL bit assignments.

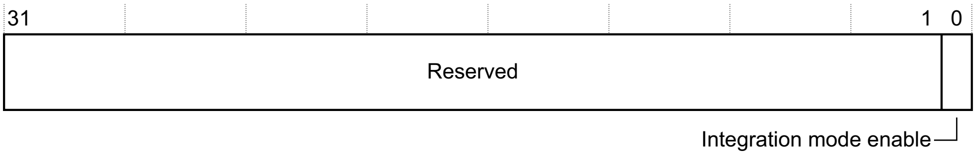


Figure 10-20 DBGITCTRL bit assignments

Table 10-19 shows the DBGITCTRL bit assignments.

Table 10-19 DBGITCTRL bit assignments

Bits	Name	Function
[31:1]	-	Reserved.
[0]	Integration mode enable	When this bit is set to 1, the device reverts to an integration mode to enable integration testing or topology detection: **0** Normal operation. **1** Integration mode enabled.

10.4.18 Claim Tag Set Register

The DBGCLAIMSET characteristics are:

Purpose Used by software to set CLAIM bits to 1.

Usage constraints Use in conjunction with the DBGCLAIMCLR register. See *DBGCLAIMCLR bit assignments* on page 10-30.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-21 shows the DBGCLAIMSET bit assignments.

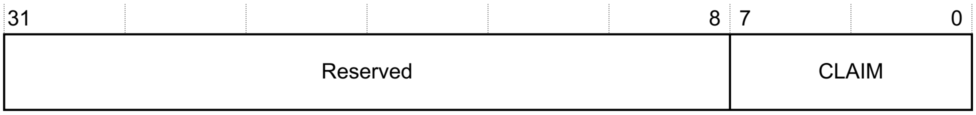


Figure 10-21 DBGCLAIMSET bit assignments

Table 10-20 shows the DBGCLAIMSET bit assignments.

Table 10-20 DBGCLAIMSET bit assignments

Bits	Name	Function
[31:8]	-	Reserved.
[7:0]	CLAIM	CLAIM bits. Writing a 1 to one of these bits sets the corresponding CLAIM bit to 1. A single write operation can set multiple bits to 1. Writing 0 to one of these bits has no effect. The CLAIM bits do not have any specific functionality. ARM expects the usage model to be that an external debugger and a debug monitor can set specific bits to 1 to claim the corresponding debug resources. This field is RAO. See *Claim Tag Clear Register* on page 10-30 for information on how to: • Clear CLAIM bits to 0. • Read the current values of the CLAIM bits.

10.4.19 Claim Tag Clear Register

The DBGCLAIMCLR characteristics are:

Purpose Used by software to read the values of the CLAIM bits, and to clear these bits to zero.

Usage constraints Use in conjunction with the DBGCLAIMSET register. See *DBGCLAIMSET bit assignments* on page 10-29.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-22 shows the DBGCLAIMCLR bit assignments.

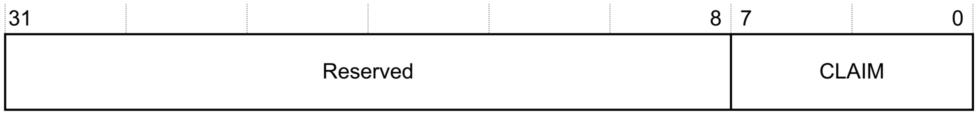


Figure 10-22 DBGCLAIMCLR bit assignments

Table 10-21 shows the DBGCLAIMCLR bit assignments.

Table 10-21 DBGCLAIMCLR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RAZ/SBZP.
[7:0]	CLAIM	CLAIM bits. Writing a 1 to one of these bits clears the corresponding CLAIM bit to 0. A single write operation can clear multiple bits to 0. Writing 0 to one of these bits has no effect. Reading the register returns the current values of these bits. The debug logic reset value of these bits is 0. For more information about the CLAIM bits, see *Claim Tag Set Register* on page 10-29.

10.4.20 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 There are no usage constraints.

Configurations This register is required in all implementations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-23 shows the DBGDEVID1 bit assignments.

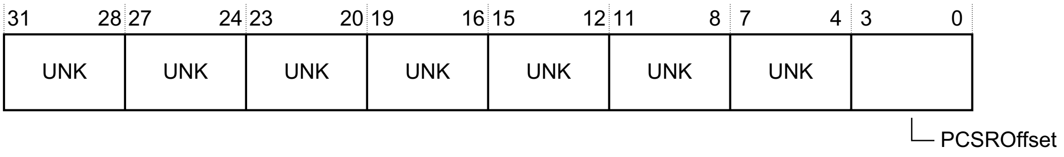


Figure 10-23 DBGDEVID1 bit assignments

Table 10-22 shows the DBGDEVID1 bit assignments.

Table 10-22 DBGDEVID1 bit assignments

Bits	Name	Function
[31:4]	-	Reserved, UNK.
[3:0]	PCSROffset	This field defines the offset applied to DBGPCSR samples: `b0000` DBGPCSR samples are offset by a value that depends on the instruction set state. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

10.4.21 Debug Device ID Register

The DBGDEVID characteristics are:

Purpose Extends the DBGDIDR by describing other features of the debug implementation.

Usage constraints Use in conjunction with DBGDIDR to find the features of the debug implementation. See *DBGDIDR bit assignments* on page 10-10.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-6.

Figure 10-24 shows the DBGDEVID bit assignments.

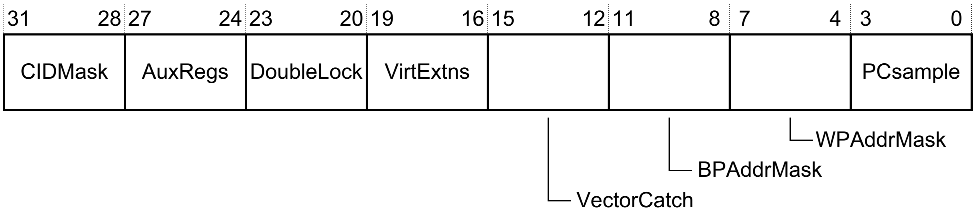


Figure 10-24 DBGDEVID bit assignments

Table 10-23 shows the DBGDEVID bit assignments.

Table 10-23 DBGDEVID bit assignments

Bits	Name	Function
[31:28]	CIDMask	CIDMask. This field indicates the level of support for the Context ID matching breakpoint masking capability. `b0000` Context ID masking not implemented.
[27:24]	AuxRegs	Specifies support for the Debug External Auxiliary Control Register. See *Debug External Auxiliary Control Register* on page 10-13: `b0001` The processor supports Debug External Auxiliary Control Register.
[23:20]	DoubleLock	Specifies support for the OS Double Lock Register: `b0001` The processor supports OS Double Lock Register.
[19:16]	VirExtns	Specifies the implementation of the Virtualization Extensions to the Debug architecture: `b0001` The processor implements the Virtualization Extensions to the Debug architecture.

Table 10-23 DBGDEVID bit assignments (continued)

Bits	Name	Function
[15:12]	VectorCatch	Defines the form of the vector catch event implemented: `b0000` The processor implements address matching form of vector catch.
[11:8]	BPAddrMask	Indicates the level of support for the *Immediate Virtual Address* (IVA) matching breakpoint masking capability: `b1111` Breakpoint address masking not implemented. DBGBCRn[28:24] are UNK/SBZP.
[7:4]	WPAddrMask	Indicates the level of support for the DVA matching watchpoint masking capability: `b0001` Watchpoint address mask implemented.
[3:0]	PCsample	Indicates the level of support for Program Counter sampling using debug registers 40, 41 and 42: `b0011` DBGPCSR, DBGCIDSR and DBGVIDSR are implemented as debug registers 40, 41, and 42.

10.4.22 Peripheral Identification Registers

The Peripheral Identification Registers provide standard information required for all components that conform to the ARM Debug Interface v5 specification. They are a set of eight registers, listed in register number order in Table 10-24.

Table 10-24 Summary of the Peripheral Identification Registers

Register	Value	Offset
Peripheral ID4	`0x04`	`0xFD0`
Peripheral ID5	`0x00`	`0xFD4`
Peripheral ID6	`0x00`	`0xFD8`
Peripheral ID7	`0x00`	`0xFDC`
Peripheral ID0	`0x0F`	`0xFE0`
Peripheral ID1	`0xBC`	`0xFE4`
Peripheral ID2	`0x3B`	`0xFE8`
Peripheral ID3	`0x00`	`0xFEC`

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

The *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* describes these registers.

10.4.23 Component Identification Registers

There are four read-only Component Identification Registers, Component ID0 through Component ID3. Table 10-25 shows these registers.

Table 10-25 Summary of the Component Identification Registers

Register	Value	Offset
Component ID0	`0x0D`	`0xFF0`
Component ID1	`0x90`	`0xFF4`
Component ID2	`0x05`	`0xFF8`
Component ID3	`0xB1`	`0xFFC`

The Component Identification Registers identify Debug as an ARM Debug Interface v5 component. The *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* describes these registers.

10.5 Debug events

A debug event can be either:

- A software debug event.
- A halting debug event.

A processor responds to a debug event in one of the following ways:

- Ignores the debug event.
- Takes a debug exception.
- Enters debug state.

This section describes debug events in:

- *Watchpoint debug events*.
- *Asynchronous aborts*.
- *Debug OS Lock*.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information on debug events.

10.5.1 Watchpoint debug events

In the Cortex-A15 MPCore processor, watchpoint debug events are always synchronous. Memory hint instructions and cache clean operations 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, the value reported in DFAR is guaranteed to be no lower than the address of the watchpointed location rounded down to a multiple of 16 bytes.

10.5.2 Asynchronous aborts

The processor ensures that all possible outstanding asynchronous Data Aborts are recognized before entry to the debug state. The debug asynchronous aborts discarded bit, DBGDSCR.ADAdiscard, is set to 1 on entry to debug state.

While in debug state with DBGDSCR.ADAdiscard set, asynchronous Data Aborts generated as a result of double bit ECC errors are not discarded. This is because the processor cannot determine whether the abort was generated from the debugger or from external system activity.

While in debug state with DBGDSCR.ADAdiscard set, the Cortex-A15 MPCore processor reports AXI bus errors by asserting **nAXIERRIRQ** or **nINTERRIRQ** pin.

10.5.3 Debug OS Lock

Debug OS Lock is set by the powerup reset, **nCPUPORESET**, see *Resets* on page 2-11. For normal behavior of debug events and debug register accesses, Debug OS Lock must be cleared. For more information, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

10.6 External debug interface

The system can access memory-mapped debug registers through the APB interface. The APB interface is compliant with the AMBA 3 APB interface.

Figure 10-25 shows the debug interface implemented in the Cortex-A15 MPCore processor. For more information on these signals, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

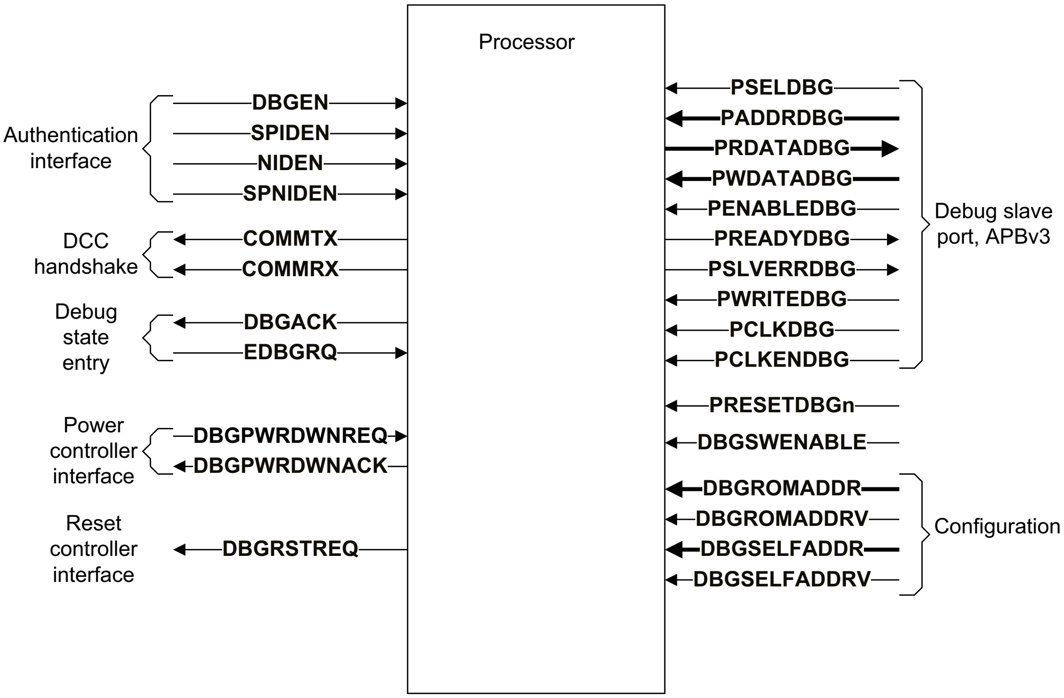


Figure 10-25 External debug interface, including APBv3 slave port

This section describes external debug interface in:

- *Memory map*.
- *Miscellaneous debug signals* on page 10-36.
- *Changing the authentication signals* on page 10-36.

10.6.1 Memory map

The basic memory map supports up to four processors in an MPCore device. Table 10-26 shows the address mapping for the debug trace components.

Table 10-26 Address mapping for debug trace components

Address range	Component[a]
`0x00000 - 0x00FFF`	ROM table
`0x01000 - 0x0FFFF`	Reserved
`0x10000 - 0x10FFF`	CPU 0 Debug
`0x11000 - 0x11FFF`	CPU 0 PMU
`0x12000 - 0x12FFF`	CPU 1 Debug

Table 10-26 Address mapping for debug trace components (continued)

Address range	Component[a]
`0x13000 - 0x13FFF`	CPU 1 PMU
`0x14000 - 0x14FFF`	CPU 2 Debug
`0x15000 - 0x15FFF`	CPU 2 PMU
`0x16000 - 0x16FFF`	CPU 3 debug
`0x17000 - 0x17FFF`	CPU 3 PMU
`0x18000 - 0x18FFF`	CPU 0 CTI
`0x19000 - 0x19FFF`	CPU 1 CTI
`0x1A000 - 0x1AFFF`	CPU 2 CTI
`0x1B000 - 0x1BFFF`	CPU 3 CTI
`0x1C000 - 0x1CFFF`	CPU 0 Trace
`0x1D000 - 0x1DFFF`	CPU 1 Trace
`0x1E000 - 0x1EFFF`	CPU 2 Trace
`0x1F000 - 0x1FFFF`	CPU 3 Trace

[a] Indicates the mapped component if present, otherwise reserved.

10.6.2 Miscellaneous debug signals

This section describes miscellaneous debug input and output signals in:

- *DBGPWRDWNREQ*.
- *DBGPWRDWNACK*.

DBGPWRDWNREQ

You must set the **DBGPWRDWNREQ** signal HIGH before removing power from the core domain. Bit[0] of the Device Powerdown and Reset Status Register reflects the value of this **DBGPWRDWNREQ** signal.

Note

DBGPWRDWNREQ must be tied LOW if the particular implementation does not support separate core and debug power domains.

DBGPWRDWNACK

This signal indicates to the system that it is safe to bring the core voltage down.

10.6.3 Changing the authentication signals

The **NIDEN**, **DBGEN**, **SPIDEN**, and **SPNIDEN** input signals are either tied off to some fixed value or controlled by some external device.

Chapter 11
Performance Monitor Unit

This chapter describes the *Performance Monitor Unit* (PMU) and the registers that it uses. It contains the following sections:

11.1 About the PMU

The processor includes logic to gather various statistics on the operation of the processor and memory system during runtime, based on PMUv2 architecture. These events provide useful information about the behavior of the processor that you can use when debugging or profiling code.

The processor PMU provides six counters. Each counter can count any of the events available in the processor. The absolute counts recorded might vary because of pipeline effects. This has negligible effect except in cases where the counters are enabled for a very short time.

11.2 PMU functional description

This section describes the functionality of the PMU in:

- *Event interface*.
- *CP15 and APB interface*.
- *Counters*.

Figure 11-1 shows the various major blocks inside the PMU.

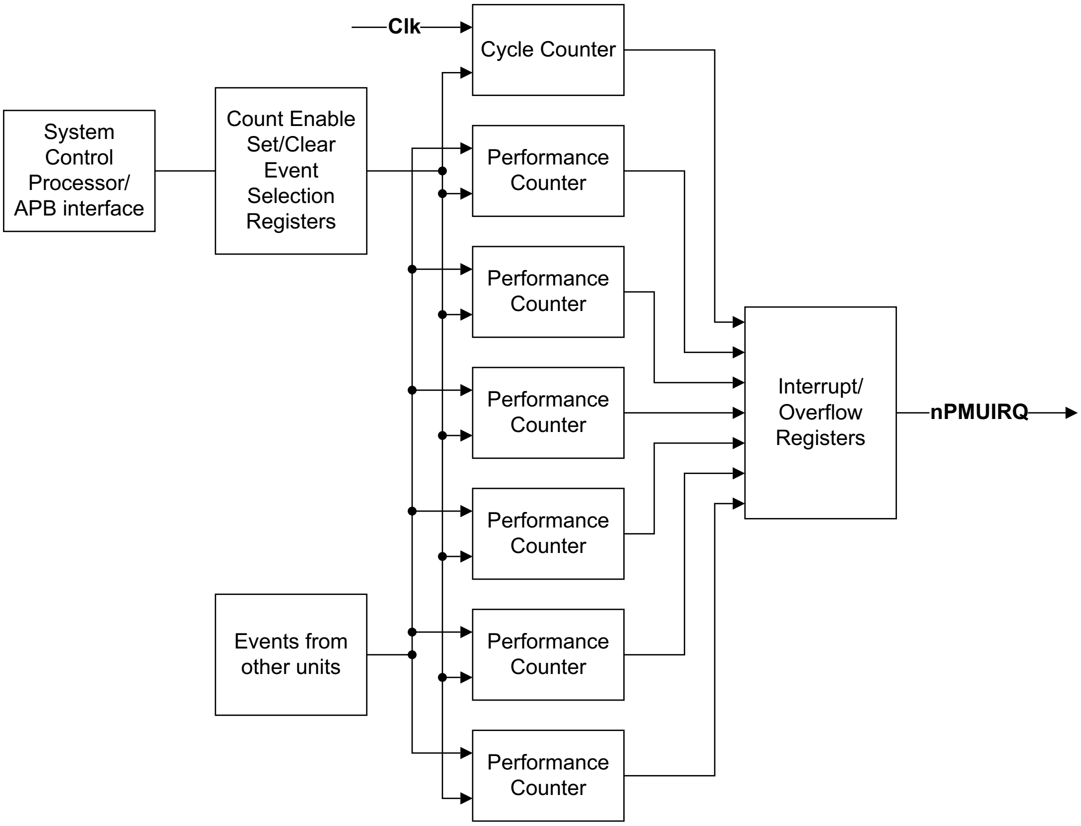


Figure 11-1 PMU block diagram

11.2.1 Event interface

Events from all other units from across the design are provided to the PMU.

11.2.2 CP15 and APB interface

The PMU registers can be programmed using the CP15 system control coprocessor or external APB interface.

11.2.3 Counters

The PMU has a 32-bit counter that increments when they are enabled based on events.

11.3 PMU register summary

The PMU counters and their associated control registers are accessible from the internal CP15 interface and from the Debug APB interface.

Table 11-1 gives a summary of the Cortex-A15 PMU registers.

Table 11-1 PMU register summary

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
0	`0x000`	c9	0	c13	2	PMXEVCNTR0	RW	Event Count Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
1	`0x004`	c9	0	c13	2	PMXEVCNTR1	RW	
2	`0x008`	c9	0	c13	2	PMXEVCNTR2	RW	
3	`0x00C`	c9	0	c13	2	PMXEVCNTR3	RW	
4	`0x010`	c9	0	c13	2	PMXEVCNTR4	RW	
5	`0x014`	c9	0	c13	2	PMXEVCNTR5	RW	
6-30	`0x018-0x78`	-	-	-	-	-	-	Reserved
31	`0x07C`	c9	0	c13	0	PMCCNTR	RW	Cycle Count Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
32-255	`0x080-0x3FC`	-	-	-	-	-		Reserved
256	`0x400`	c9	0	c13	1	PMXEVTYPER0	RW	Event Type Select Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
257	`0x404`	c9	0	c13	1	PMXEVTYPER1	RW	
258	`0x408`	c9	0	c13	1	PMXEVTYPER2	RW	
259	`0x40C`	c9	0	c13	1	PMXEVTYPER3	RW	
260	`0x410`	c9	0	c13	1	PMXEVTYPER4	RW	
261	`0x414`	c9	0	c13	1	PMXEVTYPER5	RW	
262-286	`0x418-0x478`	-	-	-	-	-	-	Reserved
287	`0x47C`	c9	0	c13	1	PMXEVTYPER31	RW	Performance Monitors Event Type Select Register 31, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
288-767	`0x480-0xBFC`	-	-	-	-	-	-	Reserved
768	`0xC00`	c9	0	c12	1	PMCNTENSET	RW	Count Enable Set Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
769-775	`0xC04-0xC1C`	-	-	-	-	-	-	Reserved
776	`0xC20`	c9	0	c12	2	PMCNTENCLR	RW	Count Enable Clear Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
777-783	`0xC24-0xC3C`	-	-	-	-	-	-	Reserved

Table 11-1 PMU register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
784	`0xC40`	c9	0	c14	1	PMINTENSET	RW	Interrupt Enable Set Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
785-791	`0xC44-0xC5C`	-	-	-	-	-	-	Reserved
792	`0xC60`	c9	0	c14	2	PMINTENCLR	RW	Interrupt Enable Clear Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
793-799	`0xC64-0xC7C`	-	-	-	-	-	-	Reserved
800	`0xC80`	c9	0	c12	3	PMOVSR	RW	Overflow Flag Status Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
801-807	`0xC84-0xC9C`	-	-	-	-	-	-	Reserved
808	`0xCA0`	c9	0	c12	4	PMSWINC	WO	Software Increment Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
809-815	`0xCA4-0xCBC`	-	-	-	-	-	-	Reserved
816	`0xCC0`	c9	0	c14	3	PMOVSSET	RW	Performance Monitor Overflow Status Set Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
817-895	`0xCC4-0xDFC`	-	-	-	-	-	-	Reserved
896	`0xE00`	-	-	-	-	PMCFGR	RO	*Performance Monitor Configuration Register* on page 11-7
897	`0xE04`	c9	0	c12	0	PMCR	RW	*Performance Monitor Control Register* on page 11-8
898	`0xE08`	c9	0	c14	0	PMUSERENR	RW	User Enable Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
899-903	`0xE0C-0xE1C`	-	-	-	-	-	-	Reserved
904	`0xE20`	c9	0	c12	6	PMCEID0	RO	*Common Event Identification Register 0 bit assignments* on page 11-11
905	`0xE24`	c9	0	c12	7	PMCEID1	RO	*Performance Monitor Common Event Identification Register 1* on page 11-12
906-1003	`0xE28-0xFAC`	-	-	-	-	-	-	Reserved

1004	`0xFB0`	-	-	-	-	PMLAR	WO	Lock Access Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
1005	`0xFB4`	-	-	-	-	PMLSR	RO	Lock Status Register, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*
1006	`0xFB8`					PMAUTHSTATUS	RO	Authentication Status Register, see the *ARM®*

11.4 PMU register descriptions

This section describes the Cortex-A15 MPCore PMU registers. Table 11-1 on page 11-4 provides cross references to individual registers.

11.4.1 Performance Monitor Configuration Register

The PMCFGR characteristics are:

Purpose Contains PMU-specific configuration data.

This register is visible only in the memory-mapped views of the Performance Monitors registers.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes See the register summary in Table 11-1 on page 11-4.

Figure 11-2 shows the PMCFGR bit assignments.

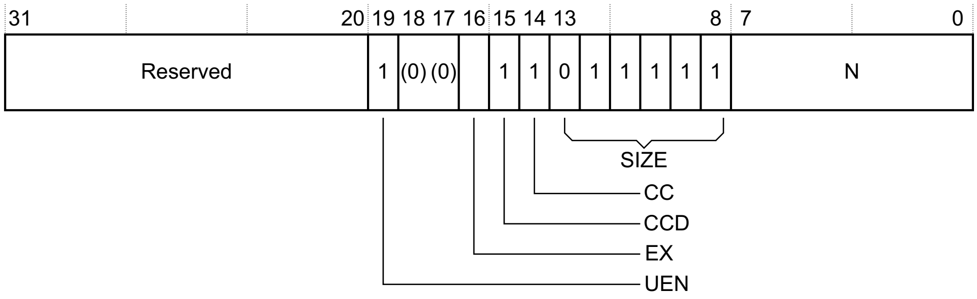


Figure 11-2 PMCFGR bit assignments

Table 11-2 shows the PMCFGR bit assignments.

Table 11-2 PMCFGR bit assignments

Bits	Name	Function
[31:20]	-	Reserved.
[19]	UEN	User mode enable register supported bit. This bit is RAO: **1** User mode enable register implemented.
[18:17]	-	Reserved.
[16]	EX	Export supported: **1** Export is supported. PMCR.X is writable.
[15]	CCD	Cycle counter clock divider implemented. This bit is RAO: **1** The cycle count divider implemented. PMCR.D is writable.

Table 11-2 PMCFGR bit assignments (continued)

Bits	Name	Function
[14]	CC	Cycle counter implemented. This bit is RO: **1** Cycle counter is implemented. PMCR.C is writable.
[13:8]	Size	Counter size. This field is RO and reads as `b011111`: `b011111` 32-bit counters.
[7:0]	N	Number of event counters. This field is RO with a value that indicates the number of implemented event counters: `b00000110` Six counters. **Note** The cycle counter is not included in the value indicated by the N field. The value of HDCR.HPMN has no effect on the value returned by this field.

11.4.2 Performance Monitor Control Register

The PMCR characteristics are:

Purpose Provides information on the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Usage constraints The PMCR is:

- A read/write register.
- Common to the Secure and Non-secure states.
- Accessible in Hyp mode, and all modes executing at PL1 when HDCR.TPM and HDCR.TPMCR are set to 0.
- Accessible in User mode only when PMUSERENR.EN is set to 1, HDCR.TPM and HDCR.TPMCR are set to 0.

Configurations Available in all configurations.

Attributes See the register summary in Table 11-1 on page 11-4.

Figure 11-3 shows the PMCR bit assignments.

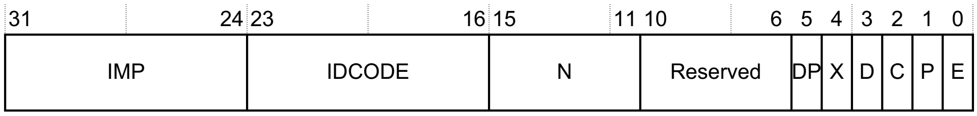


Figure 11-3 PMCR bit assignments

Table 11-3 shows the PMCR bit assignments.

Table 11-3 PMCR bit assignments

Bits	Name	Function
[31:24]	IMP	Implementer code: `0x41` ARM. This is a read-only field.
[23:16]	IDCODE	Identification code: `0x0F` Cortex-A15. This is a read-only field.
[15:11]	N	Number of event counters. In Non-secure modes other than Hyp mode, this field reads the value of HDCR.HPMN. See *Hyp Debug Configuration Register* on page 4-69. In Secure state and Hyp mode, this field returns `0x6` that indicates the number of counters implemented. This is a read-only field.
[10:6]	-	Reserved, UNK/SBZP.
[5]	DP	Disable 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 the non-invasive debug is not permitted and enabled. This bit is read/write. For more information on prohibited regions, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.
[4]	X	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, that can be implemented as a signal exported from the processor to an interrupt controller.
[3]	D	Clock divider: **0** When enabled, PMCCNTR counts every clock cycle. This is the reset value. **1** When enabled, PMCCNTR counts every 64 clock cycles. This bit is read/write.

Table 11-3 PMCR bit assignments (continued)

Bits	Name	Function
[2]	C	Clock counter reset: **0** No action. **1** Reset PMCCNTR to 0. ——— **Note** ——— Resetting PMCCNTR does not clear the PMCCNTR overflow bit to 0. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information. ———————— This bit is write-only, and always RAZ.
[1]	P	Event counter reset: **0** No action. **1** Reset all event counters, not including PMCCNTR, to 0. In Non-secure modes other than Hyp mode, a write of 1 to this bit does not reset event counters that the HDCR.HPMN field reserves for Hyp mode use. See *Hyp Debug Configuration Register* on page 4-69. In Secure state and Hyp mode, a write of 1 to this bit resets all the event counters.
[0]	E	Enable bit: **0** All counters, including PMCCNTR, are disabled. This is the reset value. **1** All counters are enabled. This bit does not disable or enable, counting by event counters reserved for Hyp mode by HDCR.HPMN. It also does not suppress the generation of performance monitor overflow interrupt requests by those counters. This bit is read/write.

To access the PMCR, read or write the CP15 registers with:

```
MRC p15, 0, <Rt>, c9, c12, 0; Read Performance Monitor Control Register
MCR p15, 0, <Rt>, c9, c12, 0; Write Performance Monitor Control Register
```

11.4.3 Performance Monitor Common Event Identification Register 0

The PMCEID0 characteristics are:

Purpose Defines which common architectural and common micro-architectural feature events are implemented.

Usage constraints The PMCEID0 is:

- A read-only register.
- Common to the Secure and Non-secure states.
- Accessible in Hyp mode and all modes executing at PL1 when HDCR.TPM is set to 0.
- Accessible in User mode only when PMUSERENR.EN is set to 1 and HDCR.TPM is set to 0.

Configurations Available in all configurations.

Attributes See the register summary in Table 11-1 on page 11-4.

Table 11-4 on page 11-11 shows the PMCEID0 bit assignments with event implemented or not implemented when the associated bit is set to 1 or 0.

PMCEID1[31:0] is reserved.

Table 11-4 Common Event Identification Register 0 bit assignments

Bit	Event number	Event implemented if set to 1 or not implemented if set to 0
[31]	0x1F	Reserved.
[30]	0x1E	
[29]	0x1D	Bus cycle. This event is implemented.
[28]	0x1C	Instruction architecturally executed, condition check pass, write to translation table base. This event is implemented.
[27]	0x1B	Instruction speculatively executed. This event is implemented.
[26]	0x1A	Local memory error. This event is implemented.
[25]	0x19	Bus access. This event is implemented.
[24]	0x18	Level 2 data cache write-back. This event is implemented.
[23]	0x17	Level 2 data cache refill. This event is implemented.
[22]	0x16	Level 2 data cache access. This event is implemented.
[21]	0x15	Level 1 data cache write-back. This event is implemented.
[20]	0x14	Level 1 instruction cache access. This event is implemented.
[19]	0x13	Data memory access. This event is implemented.
[18]	0x12	Predictable branch speculatively executed. This bit is RAO.
[17]	0x11	Cycle, this bit is RAO.
[16]	0x10	Mispredicted or not predicted branch speculatively executed. This bit is RAO.
[15]	0x0F	Instruction architecturally executed, condition check pass, unaligned load or store. This event is not implemented.
[14]	0x0E	Instruction architecturally executed, condition check pass, procedure return. This event is not implemented.
[13]	0x0D	Instruction architecturally executed, immediate branch. This event is not implemented.
[12]	0x0C	Instruction architecturally executed, condition check pass, software change of the PC. This event is not implemented.
[11]	0x0B	Instruction architecturally executed, condition check pass, write to CONTEXTIDR. This event is implemented.
[10]	0x0A	Instruction architecturally executed, condition check pass, exception return. This event is implemented.
[9]	0x09	Exception taken. This event is implemented.
[8]	0x08	Instruction architecturally executed, this bit is RAO.
[7]	0x07	Instruction architecturally executed, condition check pass, store. This event is not implemented.
[6]	0x06	Instruction architecturally executed, condition check pass, load. This event is not implemented.
[5]	0x05	Level 1 data TLB refill. This event is implemented.
[4]	0x04	Level 1 data cache access. This bit is RAO.
[3]	0x03	Level 1 data cache refill. This bit is RAO.

Table 11-4 Common Event Identification Register 0 bit assignments (continued)

Bit	Event number	Event implemented if set to 1 or not implemented if set to 0
[2]	`0x02`	Level 1 instruction TLB refill. This event is implemented.
[1]	`0x01`	Level 1 instruction cache refill. This event is implemented.
[0]	`0x00`	Instruction architecturally executed, condition check pass, software increment. This bit is RAO.

11.4.4 Performance Monitor Common Event Identification Register 1

PMCEID1 is reserved, so this register is always RAZ.

11.4.5 Peripheral Identification Registers

The Peripheral Identification Registers provide standard information required for all components that conform to the ARM PMUv2 architecture. They are a set of eight registers, listed in register number order in Table 11-5. These registers are Performance Monitors registers that are visible only in the memory-mapped views of the Performance Monitors registers.

Table 11-5 Summary of the Peripheral Identification Registers

Register	Value	Offset
Peripheral ID4	`0x04`	`0xFD0`
Peripheral ID5	`0x00`	`0xFD4`
Peripheral ID6	`0x00`	`0xFD8`
Peripheral ID7	`0x00`	`0xFDC`
Peripheral ID0	`0xAF`	`0xFE0`
Peripheral ID1	`0xB9`	`0xFE4`
Peripheral ID2	`0x3B`	`0xFE8`
Peripheral ID3	`0x00`	`0xFEC`

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

The *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* describes these registers.

11.5 Effect of OS Double Lock on PMU register access

All PMU register accesses through the memory-mapped and external debug interfaces behave as if the core power domain is off when OS Double Lock is set. For more information, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

11.6 Events

Table 11-7 shows the events that are generated and the numbers that the PMU uses to reference the events. The table also shows the event mnemonic and the bit position of each event on the event bus. Event reference numbers that are not listed are reserved.

Table 11-7 PMU events

Event number	Event mnemonic	PMU event bus (to external)[a]	PMU event bus (to trace)[a]	Event name
0x00	SW_INCR	-	[0]	Instruction architecturally executed, condition code check pass, software increment
0x01	L1I_CACHE_REFILL	[0]	[1]	Level 1 instruction cache refill
0x02	L1I_TLB_REFILL	[1]	[2]	Level 1 instruction TLB refill
0x03	L1D_CACHE_REFILL	[2]	[3]	Level 1 data cache refill
0x04	L1D_CACHE	-	[5:4]	Level 1 data cache access
0x05	L1D_TLB_REFILL	-	[7:6]	Level 1 data TLB refill
0x08	INST_RETIRED	[6:3]	[11:8]	Instruction architecturally executed
0x09	EXC_TAKEN	[7]	[12]	Exception taken
0x0A	EXC_RETURN	[8]	[13]	Instruction architecturally executed, condition code check pass, exception return
0x0B	CID_WRITE_RETIRED	-	[14]	Instruction architecturally executed, condition code check pass, write to CONTEXTIDR
0x10	BR_MIS_PRED	[9]	[15]	Mispredicted or not predicted branch speculatively executed
0x11	CPU_CYCLES	-	[16]	Cycle
0x12	BR_PRED	[10]	[17]	Predictable branch speculatively executed
0x13	MEM_ACCESS	-	[19:18]	Data memory access
0x14	L1I_CACHE	[11]	[20]	Level 1 instruction cache access
0x15	L1D_CACHE_WB	[12]	[21]	Level 1 data cache write-back
0x16	L2D_CACHE	-	[23:22]	Level 2 data cache access
0x17	L2D_CACHE_REFILL	[13]	[24]	Level 2 data cache refill
0x18	L2D_CACHE_WB	[14]	[25]	Level 2 data cache write-back
0x19	BUS_ACCESS	-	[27:26]	Bus access
0x1A	MEMORY_ERROR	-	[28]	Local memory error
0x1B	INST_SPEC	-	[30:29]	Instruction speculatively executed
0x1C	TTBR_WRITE_RETIRED	-	[31]	Instruction architecturally executed, condition code check pass, write to TTBR

Table 11-7 PMU events (continued)

Event number	Event mnemonic	PMU event bus (to external)[a]	PMU event bus (to trace)[a]	Event name
`0x1D`	BUS_CYCLES	-	[32]	Bus cycle
`0x40`	L1D_CACHE_LD	[15]	[33]	Level 1 data cache access, read
`0x41`	L1D_CACHE_ST	[16]	[34]	Level 1 data cache access, write
`0x42`	L1D_CACHE_REFILL_LD	-	[35]	Level 1 data cache refill, read
`0x43`	L1D_CACHE_REFILL_ST	-	[36]	Level 1 data cache refill, write
`0x46`	L1D_CACHE_WB_VICTIM	-	[37]	Level 1 data cache write-back, victim
`0x47`	L1D_CACHE_WB_CLEAN	-	[38]	Level 1 data cache write-back, cleaning and coherency
`0x48`	L1D_CACHE_INVAL	-	[39]	Level 1 data cache invalidate
`0x4C`	L1D_TLB_REFILL_LD	[17]	[40]	Level 1 data TLB refill, read
`0x4D`	L1D_TLB_REFILL_ST	[18]	[41]	Level 1 data TLB refill, write
`0x50`	L2D_CACHE_LD	[19]	[42]	Level 2 data cache access, read
`0x51`	L2D_CACHE_ST	[20]	[43]	Level 2 data cache access, write
`0x52`	L2D_CACHE_REFILL_LD	-	[44]	Level 2 data cache refill, read
`0x53`	L2D_CACHE_REFILL_ST	-	[45]	Level 2 data cache refill, write
`0x56`	L2D_CACHE_WB_VICTIM	-	[46]	Level 2 data cache write-back, victim
`0x57`	L2D_CACHE_WB_CLEAN	-	[47]	Level 2 data cache write-back, cleaning and coherency
`0x58`	L2D_CACHE_INVAL	-	[48]	Level 2 data cache invalidate
`0x60`	BUS_ACCESS_LD	-	[49]	Bus access, read
`0x61`	BUS_ACCESS_ST	-	[50]	Bus access, write
`0x62`	BUS_ACCESS_SHARED	-	[52:51]	Bus access, Normal, Cacheable, Shareable
`0x63`	BUS_ACCESS_NOT_SHARED	-	[54:53]	Bus access, not Normal, Cacheable, Shareable
`0x64`	BUS_ACCESS_NORMAL	-	[56:55]	Bus access, normal
`0x65`	BUS_ACCESS_PERIPH	-	[58:57]	Bus access, peripheral
`0x66`	MEM_ACCESS_LD	-	[59]	Data memory access, read
`0x67`	MEM_ACCESS_ST	-	[60]	Data memory access, write
`0x68`	UNALIGNED_LD_SPEC	-	[61]	Unaligned access, read
`0x69`	UNALIGNED_ST_SPEC	-	[62]	Unaligned access, write
`0x6A`	UNALIGNED_LDST_SPEC	-	[64:63]	Unaligned access
`0x6C`	LDREX_SPEC	[21]	[65]	Exclusive instruction speculatively executed, `LDREX`

Table 11-7 PMU events (continued)

Event number	Event mnemonic	PMU event bus (to external)[a]	PMU event bus (to trace)[a]	Event name
`0x6D`	STREX_PASS_SPEC	[22]	[66]	Exclusive instruction speculatively executed, `STREX` pass
`0x6E`	STREX_FAIL_SPEC	[23]	[67]	Exclusive instruction speculatively executed, `STREX` fail
`0x70`	LD_SPEC	-	[69:68]	Instruction speculatively executed, load
`0x71`	ST_SPEC	-	[71:70]	Instruction speculatively executed, store
`0x72`	LDST_SPEC	-	[73:72]	Instruction speculatively executed, load or store
`0x73`	DP_SPEC	-	[75:74]	Instruction speculatively executed, integer data processing
`0x74`	ASE_SPEC	-	[77:76]	Instruction speculatively executed, Advanced SIMD Extension
`0x75`	VFP_SPEC	-	[79:78]	Instruction speculatively executed, Floating-point Extension
`0x76`	PC_WRITE_SPEC	-	[81:80]	Instruction speculatively executed, software change of the PC
`0x78`	BR_IMMED_SPEC	-	[82]	Branch speculatively executed, immediate branch
`0x79`	BR_RETURN_SPEC	-	[83]	Branch speculatively executed, procedure return
`0x7A`	BR_INDIRECT_SPEC	-	[84]	Branch speculatively executed, indirect branch
`0x7C`	ISB_SPEC	-	[85]	Barrier speculatively executed, `ISB`
`0x7D`	DSB_SPEC	[24]	[86]	Barrier speculatively executed, `DSB`
`0x7E`	DMB_SPEC	[24]	[87]	Barrier speculatively executed, `DMB`

[a] Event count is encoded as a plain binary number to accomodate count values of more than one.

11.7 Interrupts

The Cortex-A15 MPCore processor asserts the **nPMUIRQ** signal when interrupt is generated by the PMU. You can route this signal to an external interrupt controller for prioritization and masking. This is the only mechanism that signals this interrupt to the processor.

Interrupt is also driven as a trigger input to the CTI. See Chapter 13 *Cross Trigger* for more information.

11.8 Exporting PMU events

This section describes exporting of PMU events in:

- *External hardware*.
- *Debug trace hardware*.

11.8.1 External hardware

In addition to the counters in the processor, some of the events that table describes are exported on the **PMUEVENT** bus and can be connected to external hardware.

11.8.2 Debug trace hardware

Some of the events that tables describes are exported to the PTM unit, other external debug, or trace hardware, to enable the events to be monitored. See Chapter 12 *Program Trace Macrocell*l and Chapter 13 *Cross Trigger* for more information.

Chapter 12
Program Trace Macrocell

This chapter describes the *Program Trace Macrocell* (PTM) for the Cortex-A15 MPCore processor. It contains the following sections:

12.1 About PTM

The PTM is a module that performs real-time instruction flow tracing based on version 1.1 of the *Program Flow Trace* (PFTv1.1) architecture. The PTM is a CoreSight component, and is an integral part of the ARM Real-time Debug solution, RealView. See the CoreSight documentation in *Additional reading* on page ix.

12.2 PTM options

Table 12-1 shows the options implemented in the Cortex-A15 PTM.

Table 12-1 Cortex-A15 MPCore PTM implementation options

Resource	Implemented, or number of instances
Number of address comparators pairs	4
Context ID comparators	1
VMID comparator	1
Embedded ICE watchpoint inputs	0
Counters	2
Sequencers	1
External inputs	4
External outputs	2
Extended external inputs, PMUEVENT	88
Extended external input selectors	2
Instrumentation resources	0
FIFOFULL supported	No
Software access to registers	Yes
FIFO depth	84 bytes
Trace output	Synchronous ATB interface
Timestamp size	64 bits
Timestamp encoding	Natural binary

12.3 PTM functional description

This section describes the PTM in:

- *Processor interface*.
- *Trace generation*.
- *Filtering and triggering resources* on page 12-5.
- *FIFO* on page 12-5.
- *Trace out* on page 12-5.

Figure 12-1 shows the main functional blocks of the PTM.

Figure 12-1 PTM functional blocks

12.3.1 Processor interface

This block monitors the behavior of the processor and generates waypoint information.

12.3.2 Trace generation

The PFT architecture assumes that the trace tools can access a copy of the code being traced. For this reason, the PTM generates trace only at certain points in program execution, called *waypoints*. This reduces the amount of trace data generated by the PTM compared to the ETM protocol. Waypoints are changes in the program flow or events, such as an exception. The trace tools use waypoints to follow the flow of program execution. For full reconstruction of the program flow, the PTM traces:

- Indirect branches, with target address and condition code.
- Direct branches with only the condition code.
- Instruction Synchronization Barrier.
- Exceptions, with an indication of where the exception occurred.
- Exceptions, with an indication of where the exception returned.
- Changes in processor instruction set state.
- Changes in processor security state.
- Changes in Context ID
- Changes in VMID.
- Entry to and return from Debug state when Halting Debug-mode is enabled.

You can also configure the PTM to trace:

- Cycle count between traced waypoints.
- Global system timestamps.
- Target addresses for taken direct branches.

12.3.3 Filtering and triggering resources

You can filter the PTM trace such as configuring it to trace only in certain address ranges. More complicated logic analyzer style filtering options are also available.

The PTM can also generate a trigger that is a signal to the trace capture device to stop capturing trace.

12.3.4 FIFO

The trace generated by the PTM is in a highly-compressed form. The trace compression causes the FIFO enable bursts to be flattened out. When the FIFO becomes full, the FIFO signals an overflow. The trace generation logic does not generate any new trace until the FIFO has emptied. This causes a gap in the trace when viewed in the debugger.

12.3.5 Trace out

Trace from FIFO is output on the synchronous AMBA ATB interface.

12.4 Reset

The resets for the processor and the PTM are usually separate to enable tracing through a processor reset. If the PTM is reset, tracing stops until the PTM is reprogrammed and re-enabled. However, if the processor is reset, the last waypoint that the processor provides before the reset might not be traced.

Programming registers

You program and read the PTM registers using the Debug APB interface. A reset of the PTM initializes the following registers:

- *Main Control Register* on page 12-14.
- *Synchronization Frequency Register* on page 12-19.
- CoreSight Trace ID Register, see the *CoreSight Program Flow Trace Architecture Specification*.
- The CoreSight registers at offsets `0xF00` to `0xFFC`.
- *Peripheral Identification Registers* on page 12-30.
- *Component Identification Registers* on page 12-30.

To start tracing, you must program the following registers to avoid UNPREDICTABLE behavior:

- *Main Control Register* on page 12-14.
- Trigger Event Register, TTER, see the *CoreSight Program Flow Trace Architecture Specification*.
- *TraceEnable Start/Stop Control Register* on page 12-18.
- *TraceEnable Control Register 1* on page 12-18.
- TraceEnable Event Register, see the *CoreSight Program Flow Trace Architecture Specification*.
- CoreSight Trace ID Register, see the *CoreSight Program Flow Trace Architecture Specification*.

You might also require to program the following:

- Address Comparator Registers if the respective address comparators are used.
- Counter Registers if the respective counters are used.
- Sequencer Registers if the sequencer is used.
- External Output Event Registers if the external outputs are used.
- Context ID Comparator Registers if the context ID comparator is used.
- VMID Comparator Register if the VMID comparator is used.
- Timestamp Event Register is timestamping is used.
- *Extended External Input Selection Register* on page 12-22 if the extended external inputs are used.

12.5.2 Register short names

All of the PTM registers have short names. Most of these are mnemonics for the full name of the register, except that the short name starts with the letters ETM, indicating that the register is defined by an ARM trace architecture. The ETM architecture is the original ARM trace architecture, and because register assignments are consistent across the trace architectures the register short names always take the ETM prefix. Table 12-2 gives some examples of the register short names.

Table 12-2 Examples of register short names

PTM register name	Register short name	Explanation of short name
Main Control Register	ETMCR	Trace Control Register
Trigger Event Register	ETMTRIGGER	Trace Trigger Register
Address Comparator Value Register 3	ETMACVR3	Trace Address Comparator Value Register 3

The use of the ETM prefix for the register short names means that the short names are distinct from the short names used for other registers, such as the processor control coprocessor registers and the debug registers.

12.5.3 Event definitions

As described in the *CoreSight Program Trace Flow Architecture Specification*, there are several event registers that you can program to select specific inputs as control events. Table 12-3 shows the event resources defined for the PTM.

Table 12-3 Event resource definitions

Resource type	Index values	Description
`b000`	0-7	Single address comparator 1-8
`b001`	0-3	Address range comparator 1-4
`b100`	0-1	Counter 1-2 at zero
`b101`	0-2	Sequencer in states 1-3
	8	Context ID comparator
	11	VMID comparator
	15	Trace start/stop resource
`b110`	0-3	External inputs 1-4
	8-9	Extended external input selectors 1-2
	13	Processor is in Non-secure state
	14	Trace prohibited by processor
	15	Hard-wired resource (always true)

12.5.4 Turning off the PTM

During normal operation, the PTM buffers individual bytes of trace, and only generates output when at least 4 bytes are available. To enable the clocks to be gated, the PTM provides a mechanism to flush all trace out of the FIFOs in certain conditions. This differs from the AFVALID mechanism, which only ensures that trace up to a certain point has been output.

When the PTM requires to enter an idle state, all trace in the FIFO is output. After the final data on the ATB interface has been accepted, the PTM is in an idle state.

If the Programming bit is set, trace generation is stopped. The PTM then enters the idle state. This ensures that no trace packets remain. After the PTM completes the idle state transition with the Programming bit set, reading the Status Register reports the Programming bit as set. The Programming bit must not be cleared until the Status Register reports the Programming bit as set.

12.5.5 Interaction with the performance monitoring unit

The processor includes a *Performance Monitoring Unit* (PMU) that enables events, such as cache misses and instructions executed, to be counted over a period of time. This section describes how the PMU and PTM function together.

Use of PMU events by the PTM

All PMU architectural events are available to the PTM through the extended input facility. See the *ARM Architectural Reference Manual* for more information on PMU events.

The PTM uses two extended external input selectors to access the PMU events. Each selector can independently select one of the PMU events, which are then active for the cycles where the relevant events occur. These selected events can then be accessed by any of the event registers within the PTM.

12.5.6 Effect of OS Double Lock on trace register access

All trace register accesses through the memory-mapped and external debug interfaces behave as if the processor power domain is off when OS Double Lock is set. For more information on OS Double Lock, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

12.6 Register summary

This section summarizes the PTM registers. For full descriptions of the PTM registers, see:

- *Register descriptions* on page 12-14, for the implementation-defined registers.
- The *CoreSight Program Flow Trace Architecture Specification*, for the other registers.

Note

- Registers not listed here are not implemented. Reading a non-implemented register address returns 0. Writing to a non-implemented register address has no effect.
- In Table 12-4, access type is described as follows:

  RW Read and write.

  RO Read only.

  WO Write only.

All PTM registers are 32 bits wide. The PTM registers are defined in the *CoreSight Program Flow Trace Architecture Specification*. Table 12-4 lists all of the registers that are implemented in the PTM with their offsets from a base address. This base address is defined by the system integrator when placing the PTM in the Debug APB memory map.

Table 12-4 PTM register summary

Base offset	Function	Type	Description
PTM configuration			
`0x000`	Main Control	RW	*Main Control Register* on page 12-14
`0x004`	Configuration Code	RO	*Configuration Code Register* on page 12-16
`0x008`	Trigger Event	RW	*CoreSight Program Flow Trace Architecture Specification*
`0x010`	Status	RW	*CoreSight Program Flow Trace Architecture Specification*
`0x014`	System Configuration	RO	*System Configuration Register* on page 12-17
TraceEnable control			
`0x018`	TraceEnable Start/Stop Control	RW	*TraceEnable Start/Stop Control Register* on page 12-18
`0x020`	TraceEnable Event	RW	*CoreSight Program Flow Trace Architecture Specification*
`0x024`	TraceEnable Control	RW	*TraceEnable Control Register 1* on page 12-18
Address comparators			
`0x040-0x05C`	Address Comparator Value 1- 8	RW	*CoreSight Program Flow Trace Architecture Specification*
`0x080-0x09C`	Address Comparator Access Type 1- 8	RW	
Counters			
`0x140-0x144`	Counter Reload Value 1-2	RW	*CoreSight Program Flow Trace Architecture Specification*
`0x150-0x154`	Counter Enable 1-2	RW	
`0x160-0x164`	Counter Reload Event 1-2	RW	
`0x170-0x174`	Counter Value 1-2	RW	
Sequencer registers			

Table 12-4 PTM register summary (continued)

Base offset	Function	Type	Description
`0x180-0x194`	Sequencer State Transition Event 1-6	RW	*CoreSight Program Flow Trace Architecture Specification*
`0x19C`	Current Sequencer State	RW	*CoreSight Program Flow Trace Architecture Specification*
External output event			
`0x1A0-0x1A4`	External Output Event 1-2	RW	*CoreSight Program Flow Trace Architecture Specification*
Context ID comparators			
`0x1B0`	Context ID Comparator Value 1	RW	*CoreSight Program Flow Trace Architecture Specification*
`0x1BC`	Context ID Comparator Mask	RW	*CoreSight Program Flow Trace Architecture Specification*
General control			
`0x1E0`	Synchronization Frequency	RW	*Synchronization Frequency Register* on page 12-19
`0x1E4`	ID	RO	See *ETM ID Register* on page 12-20
`0x1E8`	Configuration Code Extension	RO	*Configuration Code Extension Register* on page 12-21
`0x1EC`	Extended External Input Selection	RW	*Extended External Input Selection Register* on page 12-22
`0x1F8`	Timestamp Event	RW	*CoreSight Program Flow Trace Architecture Specification*
`0x1FC`	Auxiliary Control Register	RW	*Auxiliary Control Register* on page 12-23
`0x200`	CoreSight Trace ID	RW	*CoreSight Program Flow Trace Architecture Specification*
`0x204`	VMID Comparator value	RW	*CoreSight Program Flow Trace Architecture Specification*
`0x300`	OS Lock Access Specification	WO	*CoreSight Program Flow Trace Architecture Specification*
`0x304`	OS Lock Status	RO	*CoreSight Program Flow Trace Architecture Specification*
`0x310`	Power Down Control	RW	*Power Down Control Register* on page 12-24
`0x314`	Power Down Status	RO	*CoreSight Program Flow Trace Architecture Specification*
Integration registers			
`0xEDC`	Miscellaneous Outputs	WO	*Miscellaneous Output Register* on page 12-25
`0xEE0`	Miscellaneous Inputs	RO	*Miscellaneous Input Register* on page 12-26
`0xEE8`	Trigger	WO	*Trigger Register* on page 12-26
`0xEEC`	ATB Data 0	WO	*ITATBDATA0 bit assignments* on page 12-27
`0xEF0`	ATB Control 2	RO	*ATB Control Register 2* on page 12-28
`0xEF4`	ATB Identification	WO	*ATB Identification Register* on page 12-28
`0xEF8`	ATB Control 0	WO	*ATB Control Register 0* on page 12-29
`0xF00`	Integration Mode Control	RW	*Integration Mode Control Register* on page 12-30
`0xFA0`	Claim Tag Set	RW	*CoreSight Program Flow Trace Architecture Specification*
`0xFA4`	Claim Tag Clear	RW	*CoreSight Program Flow Trace Architecture Specification*
`0xFB0`	Lock Access	WO	*CoreSight Program Flow Trace Architecture Specification*
`0xFB4`	Lock Status	RO	*CoreSight Program Flow Trace Architecture Specification*

Table 12-4 PTM register summary (continued)

Base offset	Function	Type	Description
`0xFB8`	Authentication Status	RO	*CoreSight Program Flow Trace Architecture Specification*
`0xFC8`	Device Configuration	RO	*CoreSight Program Flow Trace Architecture Specification*
`0xFCC`	Device Type	RO	*CoreSight Program Flow Trace Architecture Specification*
Peripheral and Component ID registers			
`0xFD0`	Peripheral ID4	RO	*Peripheral Identification Registers* on page 12-30
`0xFD4`	Peripheral ID5	RO	
`0xFD8`	Peripheral ID6	RO	
`0xFDC`	Peripheral ID7	RO	
`0xFE0`	Peripheral ID0	RO	
`0xFE4`	Peripheral ID1	RO	
`0xFE8`	Peripheral ID2	RO	
`0xFEC`	Peripheral ID3	RO	
`0xFF0`	Component ID0	RO	*Component Identification Registers* on page 12-30
`0xFF4`	Component ID1	RO	
`0xFF8`	Component ID2	RO	
`0xFFC`	Component ID3	RO	

For more information about these registers and the packets implemented by the PTM, see the *CoreSight Program Flow Trace Architecture Specification*.

12.7 Register descriptions

This section describes the implementation-specific PTM registers in the Cortex-A15 MPCore processor. Table 12-4 on page 12-11 provides cross references to individual registers.

The *ARM® CoreSight™ Program Flow Trace Architecture Specification* describes the other PTM registers.

12.7.1 Main Control Register

The ETMCR characteristics are:

Purpose Controls general operation of the PTM, such as whether tracing is enabled or is cycle-accurate.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-2 shows the ETMCR bit assignments.

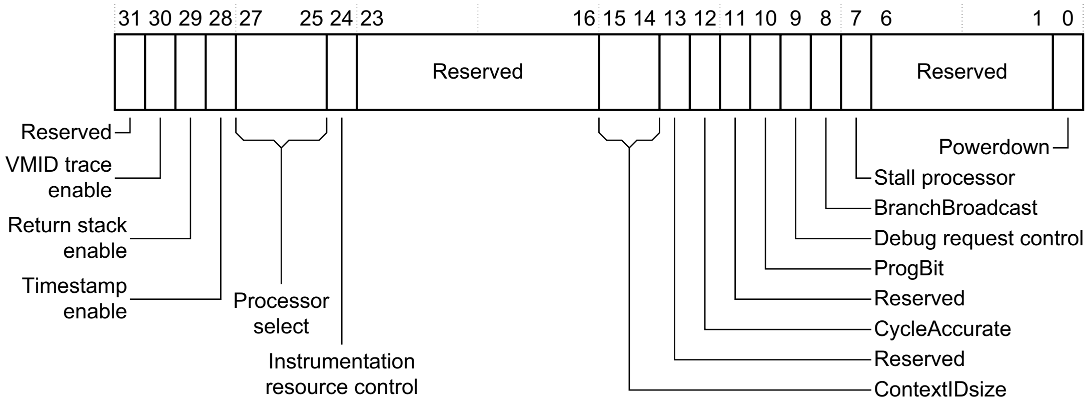


Figure 12-2 ETMCR bit assignments

Table 12-5 shows the ETMCR bit assignments.

Table 12-5 ETMCR bit assignments

Bits	Name	Function
[31]	-	SBZP.
[30]	VMID trace enable	This bit controls VMID tracing. Set this bit to 1 to enable VMID tracing. The reset value is 0.
[29]	Return stack enable	Set this bit to 1 to enable use of the return stack. The reset value is 0.
[28]	Timestamp enable	Set this bit to 1 to enable timestamping. The reset value is 0.
[27:25]	Processor select	RAZ. This bit is not implemented.
[24]	Instrumentation resources access control	RAZ. This bit is not implemented.
[23:16]	-	SBZP.

Table 12-5 ETMCR bit assignments (continued)

Bits	Name	Function
[15:14]	ContextIDsize	The possible value of this field are: `b00` No Context ID tracing. `b01` One byte traced, Context ID bits[7:0]. `b10` Two bytes traced, Context ID bits[15:0]. `b11` Four bytes traced, Context ID bits[31:0]. The reset value is 0. **Note** The PTM traces only the number of bytes specified, even if the new Context ID value is larger than this.
[13]	-	SBZP.
[12]	CycleAccurate	Set this bit to 1 to enable cycle-accurate tracing. The reset value is 0.
[11]	-	SBZP.
[10]	ProgBit	Programming bit. You must set this bit to 1 to program the PTM, and clear it to 0 when programming is complete. The reset value is 1.
[9]	Debug request control	When this bit is set to 1 and the trigger event occurs, the **DBGRQ** output is asserted until **DBGACK** is observed. This enables a debugger to force the processor into Debug state. The reset value is 0.
[8]	BranchBroadcast	Set this bit to 1 to enable branch broadcasting. Branch broadcasting traces the addresses of direct branch instructions. You must not set this bit to 1 if bit[29] of this register is set to 1 to enable use of the return stack. Behavior is UNPREDICTABLE if you enable both use of the return stack and branch broadcasting. The reset value is 0.
[7]	Stall processor	RAZ. This bit is not implemented.
[6:1]	-	SBZP.
[0]	Powerdown	A pin controlled by this bit enables the PTM power to be controlled externally. The external pin is **PTMPWRDOWN**, or inverted as **PTMPWRUP**. This bit must be cleared by the trace software tools at the beginning of a debug session. When this bit is set to 1, the PTM must be powered down and disabled, and then operated in a low power mode with all clocks stopped. When this bit is set to 1, writes to some registers and fields might be ignored. You can always write to the following registers and fields: • ETMCR, bit[0] and bits[27:25]. • ETMLAR. • ETMCLAIMSET. • ETMCLAIMCLR. • ETMOSLAR. When ETMCR is written with this bit set to 1, writes to bits other than bits[27:25, 0] might be ignored. The reset value is 1.

12.7.2 Configuration Code Register

The ETMCCR characteristics are:

Purpose Enables software to read the implementation-defined configuration of the PTM, giving the number of each type of resource. Where a value indicates the number of instances of a particular resource, zero indicates that there are no implemented resources of that resource type.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-3 shows the ETMCCR bit assignments.

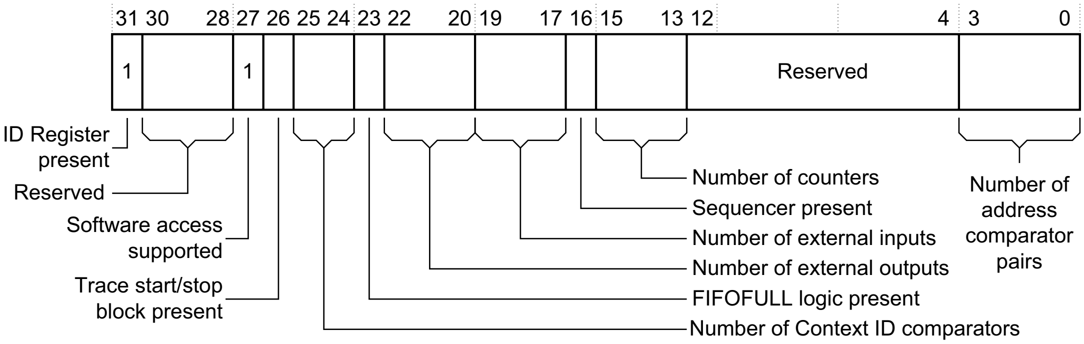

31
30
28
27
26
25
24
23
22
20
19
17
16
15
13
12
4
3
0
1
1
Reserved
ID Register present
Reserved
Software access supported
Trace start/stop block present
Number of counters
Sequencer present
Number of external inputs
Number of external outputs
FIFOFULL logic present
Number of Context ID comparators
Number of address comparator pairs

Figure 12-3 ETMCCR bit assignments

Table 12-6 shows the ETMCCR bit assignments.

Table 12-6 ETMCCR bit assignments

Bits	Name	Function
[31]	ID Register present	ID Register: **1** ID Register is present. See *ETM ID Register* on page 12-20 for more information.
[30:28]	-	RAZ.
[27]	Software access supported	Software access support: **1** The Cortex-A15 MPCore processor supports software access.
[26]	Trace start/stop block present	Trace start/stop block: **1** The trace start/stop block is present.
[25:24]	Number of Context ID comparators	Number of Context ID comparators. For the Cortex-A15 MPCore processor, this value is 1.
[23]	FIFOFULL logic present	Indicates that it is not possible to stall the processor to prevent FIFO overflow. For the Cortex-A15 MPCore processor, this value is 0.
[22:20]	Number of external outputs	Specifies the number of external outputs. For the Cortex-A15 MPCore processor, this value is 2.
[19:17]	Number of external inputs	Specifies the number of external inputs. For the Cortex-A15 MPCore processor, this value is 4.

Table 12-6 ETMCCR bit assignments (continued)

Bits	Name	Function
[16]	Sequencer present	Sequencer: **1** The sequencer is present.
[15:13]	Number of counters	Specifies the number of counters. For the Cortex-A15 MPCore processor, this value is 2.
[12:4]	-	SBZP.
[3:0]	Number of address comparator pairs	Specifies the number of address comparator pairs. For the Cortex-A15 MPCore processor, this value is 4.

12.7.3 System Configuration Register

The ETMSCR characteristics are:

Purpose Shows the PTM features supported by the PTM macrocell. The contents of this register are based on inputs provided by the ASIC.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-4 shows the ETMSCR bit assignments.

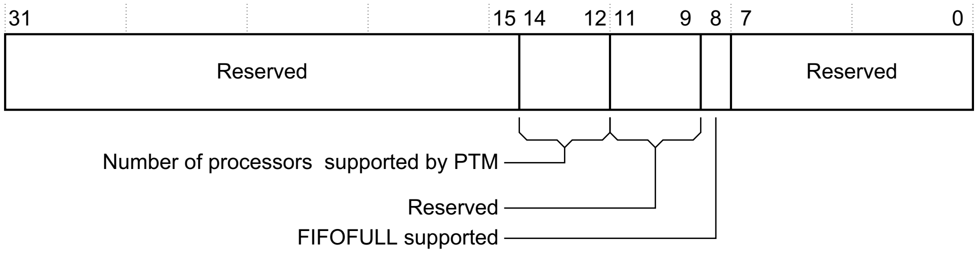


Figure 12-4 ETMSCR bit assignments

Table 12-7 shows the ETMSCR bit assignments.

Table 12-7 ETMSCR bit assignments

Bits	Name	Function
[31:15]	-	SBZP.
[14:12]	Number of processors supported by PTM	Indicates the number of processor supported minus one. For the Cortex-A15 MPCore processor, this value is 0.
[11:9]	-	SBZP.
[8]	FIFOFULL supported	Indicates that FIFOFULL is not supported. For the Cortex-A15 MPCore processor, this value is 0.
[7:0]	-	SBZP.

12.7.4 TraceEnable Start/Stop Control Register

The ETMSSCR characteristics are:

Purpose Specifies the single address comparators that hold the trace start and stop addresses.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-5 shows the ETMTSSCR bit assignments.

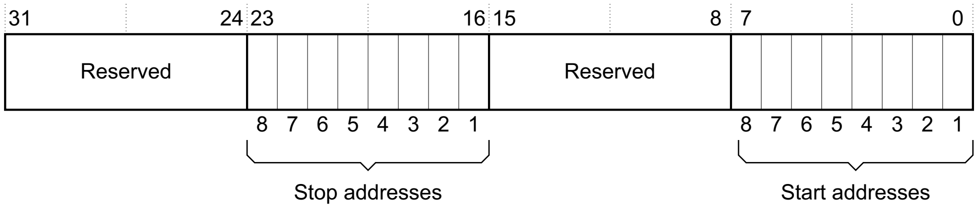


Figure 12-5 ETMSSCR bit assignments

Table 12-8 shows the ETMTSSCR bit assignments.

Table 12-8 ETMSSCR bit assignments

Bits	Name	Function
[31:24]	-	Reserved
[23:16]	Stop addresses	When a bit is set to 1, it selects a single address comparator (8-1) as a stop address for the TraceEnable Start/Stop block. For example, if you set bit[16] to 1 it selects single address comparator 1 as a stop address.
[15:8]	-	Reserved.
[7:0]	Start addresses	When a bit is set to 1, it selects a single address comparator (8-1) as a start address for the TraceEnable Start/Stop block. For example, if you set bit[0] to 1 it selects single address comparator 1 as a start address.

12.7.5 TraceEnable Control Register 1

The ETMECR1 characteristics are:

Purpose

- Enables the start/stop logic.
- Specifies the address range comparators used for include or exclude control.
- Defines whether the specified address range comparators are used for include or exclude control.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-6 on page 12-19 shows the ETMECR1 bit assignments.

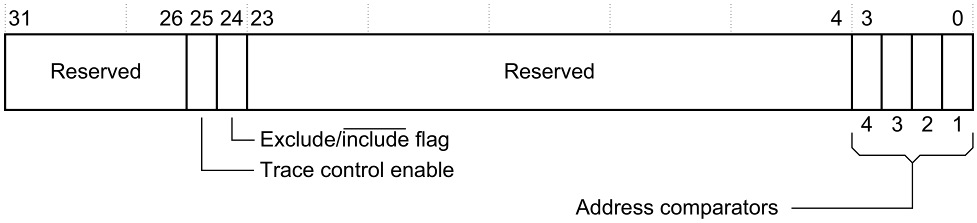


Figure 12-6 ETMECR1 bit assignments

Table 12-9 shows the ETMECR1 bit assignments.

Table 12-9 ETMECR1 bit assignments

Bits	Name	Function
[31:26]	-	SBZP.
[25]	Trace control enable	Trace start/stop control enable. The possible values of this bit are: **0** Tracing is unaffected by the trace start/stop logic. **1** Tracing is controlled by the trace on and off addresses configured for the trace start/stop logic. The trace start/stop event resource is not affected by the value of this bit. See
[24]	Exclude/include flag	Exclude/include flag. The possible values of this bit are: **0** Include. The specified address range comparators indicate the regions where tracing can occur. No tracing occurs outside this region. **1** Exclude. The specified address range comparators indicate regions to be excluded from the trace. When outside an exclude region, tracing can occur.
[23:4]	-	SBZP.
[3:0]	Address comparators	When this bit is set to 1, it selects an address range comparator, from 4 to 1, for exclude/include control. For example, bit[0] set to 1 selects address range comparator 1.

Tracing all instructions

To trace all processor execution:

- Set bit[24], the exclude/include flag, in the ETMECR1 to 1.
- Set all other bits in the ETMECR1 to 0.
- Set the ETMTEEVER to `0x0000006F` (TRUE).

This has the effect of excluding nothing, that is, tracing everything, and setting the trace enable event to always true, with the start/stop logic ignored.

12.7.6 Synchronization Frequency Register

The ETMSYNCFR characteristics are:

Purpose Holds the trace synchronization frequency value.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Bits[2:0] of this register are not implemented and read as zero. In all other respects, the *CoreSight Program Trace Flow Architecture Specification* describes the operation of this register in the PTM.

12.7.7 ETM ID Register

The ETMIDR characteristics are:

Purpose
- Holds the PTM architecture variant.
- Defines the programmers model for the PTM.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-7 shows the ETMIDR bit assignments.

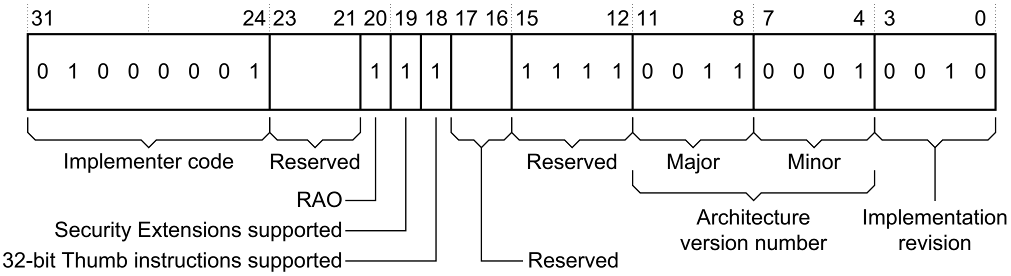


Figure 12-7 ETMIDR bit assignments

Table 12-10 shows the ETMIDR bit assignments.

Table 12-10 ID Register bit assignments

Bits	Name	Function
[31:24]	Implementer code	Indicates the implementer: `0x41` ARM.
[23:21]	-	RAZ.
[20]	-	RAO.
[19]	Security Extensions supported	Support for Security Extensions. The value of this bit is 1, indicating that the processor implements the ARM architecture Security Extensions.
[18]	32-bit Thumb instructions supported	Support for 32-bit Thumb instructions. The value of this bit is 1, indicating that a 32-bit Thumb instruction is traced as a single instruction.
[17:16]	-	RAZ.
[15:12]	-	This field reads as b1111.
[11:8]	Major architecture version	Indicates the major architecture version number. This field reads as b0011.
[7:4]	Minor architecture version	Indicates the minor architecture version number. This field reads as b0001.
[3:0]	Implementation revision	Indicates the implementation revision. This field reads as b0011.

12.7.8 Configuration Code Extension Register

The ETMCCER characteristics are:

Purpose Holds the PTM configuration information additional to that in the ETMCCR. See *Configuration Code Register* on page 12-16.

Usage constraints Software uses this register with the ETMCCR.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-8 shows the ETMCCER bit assignments.

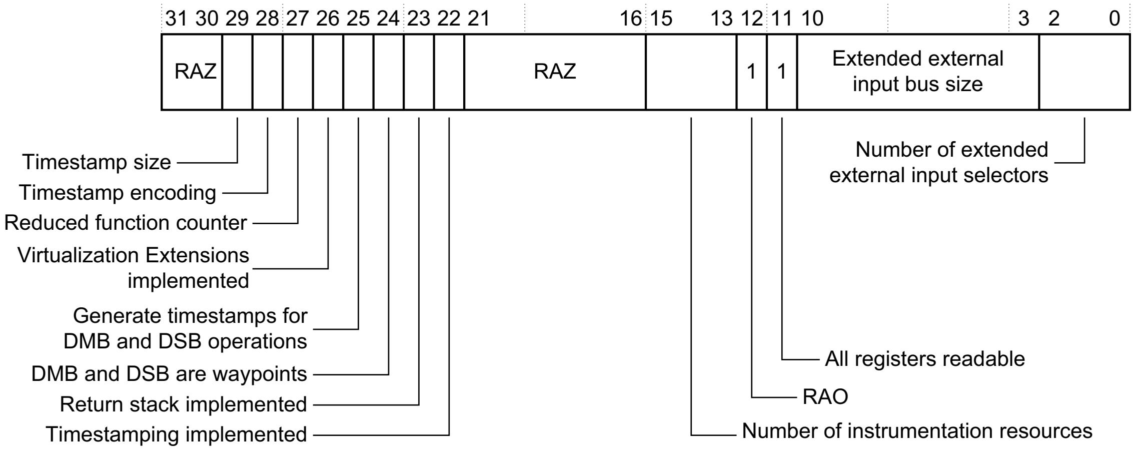

31 30 29 28 27 26 25 24 23 22 21
16 15
13 12 11 10
3 2
0
RAZ
RAZ
1
1
Extended external input bus size
Timestamp size
Timestamp encoding
Reduced function counter
Virtualization Extensions implemented
Generate timestamps for DMB and DSB operations
DMB and DSB are waypoints
Return stack implemented
Timestamping implemented
Number of extended external input selectors
All registers readable
RAO
Number of instrumentation resources

Figure 12-8 ETMCCER bit assignments

Table 12-11 shows the ETMCCER bit assignments.

Table 12-11 ETMCCER bit assignments

Bits	Name	Function
[31:30]	-	SBZP.
[29]	Timestamp size	The timestamp size is 64 bits. For the Cortex-A15 MPCore processor, this value is 1.
[28]	Timestamp encoding	Specifies that timestamp is encoded as a natural binary number. For the Cortex-A15 MPCore processor, this value is 1.
[27]	Reduced function counter	Specifies that all counters are implemented as full function counters. For the Cortex-A15 MPCore processor, this value is 0.
[26]	Virtualization Extensions implemented	Specifies that Virtualization Extensions are implemented. For the Cortex-A15 MPCore processor, this value is 1.
[25]	Generate timestamp for DMB and DSB operations	Timestamps are not generated for DMB and DSB operations. For the Cortex-A15 MPCore processor, this value is 0.
[24]	DMB and DSB are waypoints	`DMB` and `DSB` instructions are not treated as waypoints. For the Cortex-A15 MPCore processor, this value is 0.
[23]	Return stack implemented	Specifies that return stack is implemented. For the Cortex-A15 MPCore processor, this value is 1.

Table 12-11 ETMCCER bit assignments (continued)

Bits	Name	Function
[22]	Timestamping implemented	Specifies that timestamping is implemented. For the Cortex-A15 MPCore processor, this value is 1.
[21:16]	-	RAZ.
[15:13]	Number of instrumentation resources	Specifies the number of instrumentation resources. For the Cortex-A15 MPCore processor, this value is 0.
[12]	-	RAO.
[11]	All registers readable	Indicates that all registers, except some Integration Test Registers, are readable. Registers with names that start with IT are the Integration Test Registers, for example ITATBCTR1.
[10:3]	Extended External input bus size	Specifies the size of the extended external input bus of 88.
[2:0]	Number of extended external input selectors	Specifies the number of extended external input selectors: `b010` Two selectors.

12.7.9 Extended External Input Selection Register

The ETMEXTINSELR characteristics are:

Purpose Selects the extended external inputs.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-9 shows the ETMEXTINSELR bit assignments.

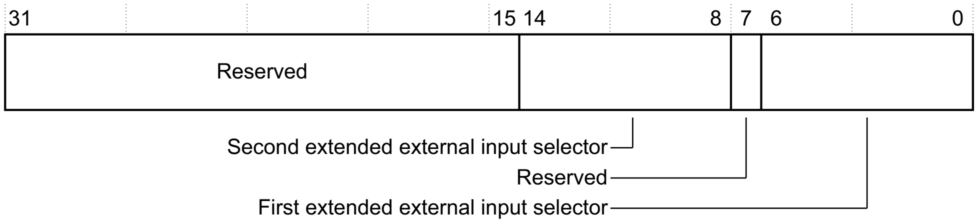

31
15 14
8 7 6
0
Reserved
Second extended external input selector
Reserved
First extended external input selector

Figure 12-9 ETMEXTINSELR bit assignments

Table 12-12 shows the ETMEXTINSELR bit assignments.

Table 12-12 ETMEXTINSELR bit assignments

Bits	Name	Function
[31:15]	-	Reserved
[14:8]	Second extended external input selector	Selects the second extended external input
[7]	-	Reserved
[6:0]	First extended external input selector	Selects the first extended external input

12.7.10 Auxiliary Control Register

The ETMAUXCR characteristics are:

Purpose Provides additional PTM controls.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-10 shows the ETMAUXCR bit assignments.

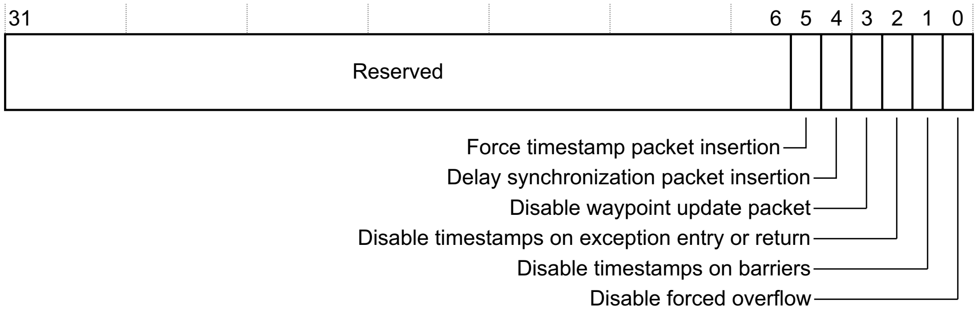

31
6
5
4
3
2
1
0
Reserved
Force timestamp packet insertion
Delay synchronization packet insertion
Disable waypoint update packet
Disable timestamps on exception entry or return
Disable timestamps on barriers
Disable forced overflow

Figure 12-10 ETMAUXCR bit assignments

Table 12-13 shows the ETMAUXCR bit assignments.

Table 12-13 ETMAUXCR bit assignments

Bits	Name	Function
[31:6]	-	Reserved.
[5]	Force timestamp packet insertion	Force insertion of timestamp packets, regardless of current trace activity: **0** Timestamp packets delayed when trace activity is high. This is the reset value. **1** Timestamp packets inserted regardless of trace activity. This bit might be set if timestamp packets occur too far apart. Setting this bit might cause the trace FIFO to overflow more frequently when trace activity is high.
[4]	Delay synchronization packet insertion	Delays insertion of synchronization packets: **0** Synchronization packets inserted normally regardless of trace activity. This is the reset value. **1** Synchronization packets delayed when trace activity is high. This bit might be set if synchronization packets occur too close. Setting this bit might cause the trace FIFO to overflow less frequently when trace activity is high.
[3]	Disable waypoint update packet	Specifies whether the PTM issues waypoint update packets if there are more than 4096 bytes between waypoints: **0** PTM always issues update packets if there are more than 4096 bytes between waypoints. This is the reset value. **1** PTM does not issue waypoint update packets unless required to do so as the result of an exception or debug entry.

Table 12-13 ETMAUXCR bit assignments (continued)

Bits	Name	Function
[2]	Disable timestamps on exception entry or return	Specifies whether the PTM issues a timestamp on an exception entry or return: **0** PTM issues timestamps on exception entry or return. This is the reset value. **1** PTM does not issue timestamps on exception entry or return.
[1]	Disable timestamps on barriers	Specifies whether the PTM issues a timestamp on a barrier instruction: **0** PTM issues timestamps on barrier instructions. This is the reset value. **1** PTM does not issue timestamps on barriers.
[0]	Disable forced overflow	Specifies whether the PTM enters overflow state when synchronization is requested, and the previous synchronization sequence has not yet completed. This does not affect entry to overflow state when the FIFO becomes full: **0** Forced overflow enabled. This is the reset value. **1** Forced overflow disabled.

When setting any of bits[3:0] of this register the PTM behavior might contradict the *CoreSight Program Flow Trace Architecture Specification*. Tools must be aware of the implications of setting any of these bits:

- Bit[0] might be set if the FIFO overflows because of the forced overflow condition. See the *CoreSight Program Flow Trace Architecture Specification* for information on this condition. If this bit is set, tools must be aware that synchronization might not occur within the desired synchronization period.
- Bit[1] might be set if timestamp packets are not required on barrier instructions. Typically, this might be set when using timestamping as a low-bandwidth measure of time, but might make correlation of multiple trace sources impossible.
- Bit[2] might be set if timestamp packets are not required on exception entry or return. Typically, this might be set when using timestamping as a low-bandwidth measure of time, but might make correlation of multiple trace sources impossible.
- Bit[3] might be set if tools do not require the waypoint update packet that is output if there are more than 4096 bytes between waypoints.

12.7.11 Power Down Control Register

The ETMPDCR characteristics are:

Purpose Controls powerdown of the PTM.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations that have full support for trace over powerdown.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-11 on page 12-25 shows the ETMPDCR bit assignments.

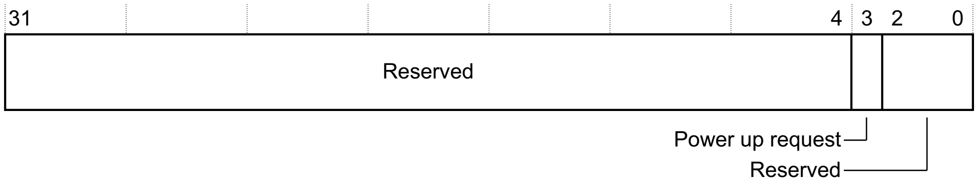


Figure 12-11 ETMPDCR bit assignments

Table 12-14 shows the ETMPDCR bit assignments.

Table 12-14 ETMPDCR bit assignments

Bits	Name	Function
[31:4]	-	Reserved.
[3]	Power up request	Core power up request bit. This bit enables a request to the power controller to power up the core, enabling access to the trace registers in the core power domain: 0 = **DBGPWRUPREQ** is LOW. This is the reset value. 1 = **DBGPWRUPREQ** is HIGH.
[2:0]	-	Reserved.

12.7.12 Miscellaneous Output Register

The ITMISCOUT characteristics are:

Purpose Controls signal outputs when bit[0] of the ETMITCTRL is set. See *Integration Mode Control Register* on page 12-30.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-12 shows the ITMISCOUT bit assignments.

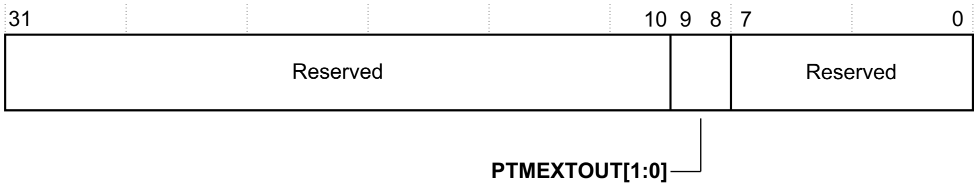


Figure 12-12 ITMISCOUT bit assignments

Table 12-15 shows the ITMISCOUT bit assignments.

Table 12-15 ITMISCOUT bit assignments

Bits	Name	Function
[31:10]	-	Reserved
[9:8]	**PTMEXTOUT[1:0]**	Drives the **PTMEXTOUT[1:0]** outputs
[7:0]	-	Reserved

12.7.13 Miscellaneous Input Register

The ITMISCIN characteristics are:

Purpose Enables the values of signal inputs to be read when bit[0] of the ETMITCTRL is set. See *Integration Mode Control Register* on page 12-30 is set.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-13 shows the ITMISCIN bit assignments.

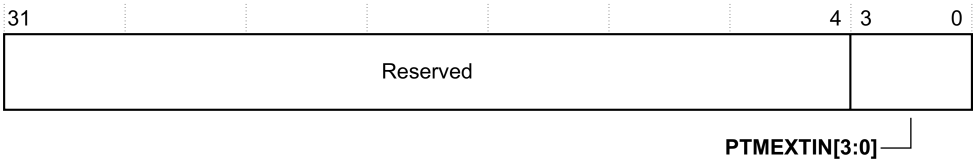


Figure 12-13 ITMISCIN bit assignments

Table 12-16 shows the ITMISCIN bit assignments.

Table 12-16 ITMISCIN bit assignments

Bits	Name	Function
[31:4]	-	Reserved
[3:0]	PTMEXTIN[3:0]	Returns the value of the **EXTIN[3:0]** inputs

12.7.14 Trigger Register

The ITTRIGGER characteristics are:

Purpose Controls signal outputs when bit[0] of the ETMITCTRL is set. See *Integration Mode Control Register* on page 12-30.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-14 shows the ITTRIGGER bit assignments.

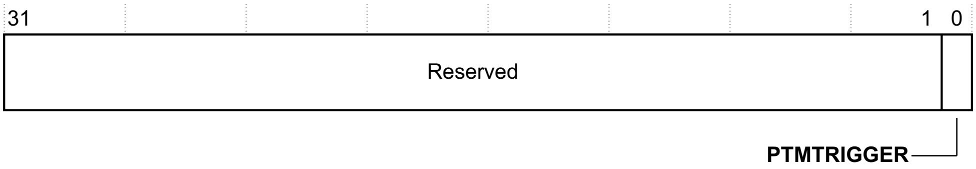


Figure 12-14 ITTRIGGER bit assignments

Table 12-17 shows the ITTRIGGER bit assignments.

Table 12-17 ITTRIGGER bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	PTMTRIGGER	Drives the **PTMTRIGGER** output

12.7.15 ATB Data Register 0

The ITATBDATA0 characteristics are:

Purpose Controls signal outputs when bit[0] of the ETMITCTRL is set. See *Integration Mode Control Register* on page 12-30.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-15 shows the ITATBDATA0 bit assignments.

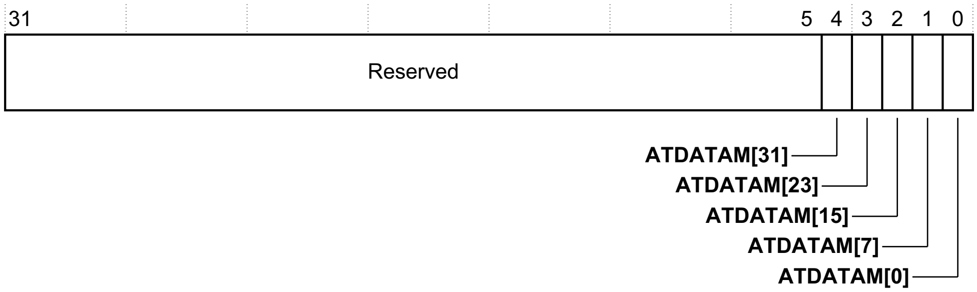


Figure 12-15 ITATBDATA0 bit assignments

Table 12-18 shows the ITATBDATA0 bit assignments.

Table 12-18 ITATBDATA0 bit assignments

Bits	Name	Function
[31:5]	-	Reserved
[4]	ATDATAM[31]	Drives the **ATDATAM[31]** output
[3]	ATDATAM[23]	Drives the **ATDATAM[23]** output
[2]	ATDATAM[15]	Drives the **ATDATAM[15]** output
[1]	ATDATAM[7]	Drives the **ATDATAM[7]** output
[0]	ATDATAM[0]	Drives the **ATDATAM[0]** output

12.7.16 ATB Control Register 2

The ITATBCTR2 characteristics are:

Purpose Enables the values of signal inputs to be read when bit[0] of the ETMITCTRL is set. See *Integration Mode Control Register* on page 12-30.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-16 shows the ITATBCTR2 bit assignments.

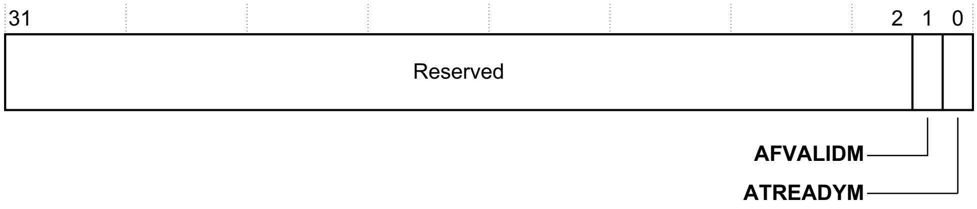


Figure 12-16 ITATBCTR2 bit assignments

Table 12-19 shows the ITATBCTR2 bit assignments.

Table 12-19 ITATBCTR2 bit assignments

Bits	Name	Function
[31:2]	-	Reserved
[1]	AFVALIDM	Returns the value of **AFVALIDM** input
[0]	ATREADYM	Returns the value of **ATREADYM** input[a]

[a] To sample **ATREADYM** correctly from the Cortex-A15 MPCore processor signals, **ATVALIDM** must be asserted.

12.7.17 ATB Identification Register

The ITATBID characteristics are:

Purpose Controls signal outputs when bit[0] of the ETMITCTRL is set. See *Integration Mode Control Register* on page 12-30.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-17 on page 12-29 shows the ITATBID bit assignments.

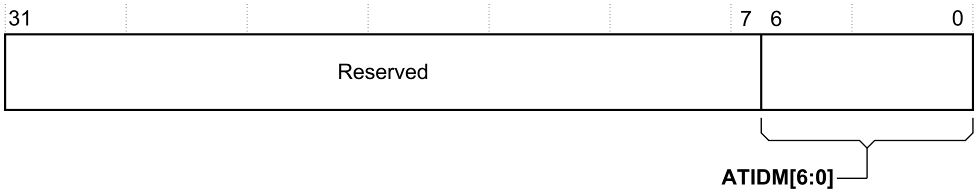


Figure 12-17 ITATBID bit assignments

Table 12-20 shows the ITATBID bit assignments.

Table 12-20 ITATBID bit assignments

Bits	Name	Function
[31:7]	-	Reserved
[6:0]	ATIDM[6:0]	Drives the **ATIDM[6:0]** outputs

12.7.18 ATB Control Register 0

The ITATBCTR0 characteristics are:

Purpose Controls signal outputs when bit[0] of the ETMITCTRL is set. See *Integration Mode Control Register* on page 12-30.

Usage constraints There are no usage constraints.

Configurations Available in all PTM configurations.

Attributes See the register summary in Table 12-4 on page 12-11.

Figure 12-18 shows the ITATBCTR0 bit assignments.

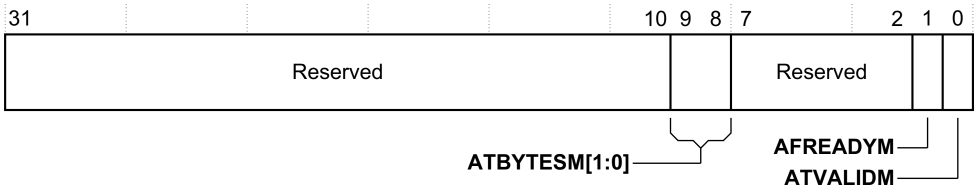


Figure 12-18 ITATBCTR0 bit assignments

Table 12-21 shows the ITATBCTR0 bit assignments.

Table 12-21 ITATBCTR0 bit assignments

Bits	Name	Function
[31:10]	-	Reserved
[9:8]	**ATBYTESM[1:0]**	Drives the **ATBYTESM** outputs
[7:2]	-	Reserved
[1]	**AFREADYM**	Drives the **AFREADYM** output
[0]	**ATVALIDM**	Drives the **ATVALIDM** output

12.7.19 Integration Mode Control Register

The ETMITCTRL Register enables topology detection and integration testing.

When bit[0] is set to 1, the PTM enters an integration mode. On reset this bit is cleared to 0.

Before entering integration mode, the PTM must be powered up and in programming mode. This means bit[0] of the Main Control Register is set to 0, and bit[10] of the Main Control Register is set to 1. See *Main Control Register* on page 12-14.

After leaving integration mode, the PTM must be reset before attempting to perform tracing.

The *CoreSight Program Trace Flow Architecture Specificatio*n describes the operation of this register in the PTM.

12.7.20 Peripheral Identification Registers

The Peripheral Identification Registers provide standard information required for all CoreSight components. They are a set of eight registers, listed in register number order in Table 12-22.

Table 12-22 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	`0x5F`	`0xFE0`
Peripheral ID1	`0xB9`	`0xFE4`
Peripheral ID2	`0x3B`	`0xFE8`
Peripheral ID3	`0x00`	`0xFEC`

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

The *CoreSight Program Trace Flow Architecture Specificatio*n describes these registers.

12.7.21 Component Identification Registers

There are four read-only Component Identification Registers, Component ID0 to Component ID3. Table 12-23 shows these registers.

Table 12-23 Summary of the Component Identification Registers

Register	Value	Offset
Component ID0	`0x0D`	`0xFF0`
Component ID1	`0x90`	`0xFF4`
Component ID2	`0x05`	`0xFF8`
Component ID3	`0xB1`	`0xFFC`

The Component Identification Registers identify PTM as a CoreSight component. The *CoreSight Program Trace Flow Architecture Specificatio*n describes these registers.

Chapter 13
Cross Trigger

This chapter describes the cross trigger interfaces for the Cortex-A15 MPCore processor. It contains the following sections:

13.1 About the cross trigger

The Cortex-A15 MPCore processor has a single external cross trigger channel interface. This external interface is connected to the CoreSight CTI interface corresponding to each processor through a simplified *Cross Trigger Matrix* (CTM). A number of trigger inputs and trigger outputs are connected between debug components in the Cortex-A15 MPCore processor and CoreSight CTI blocks.

The CoreSight *Cross Trigger Interface* (CTI) enables the debug logic, PTM, and PMU, to interact with each other and with other CoreSight components. This is called cross triggering. For example, you configure the CTI to generate an interrupt when the PTM trigger event occurs.

Figure 13-1 shows the debug system components and the available trigger inputs and trigger outputs.

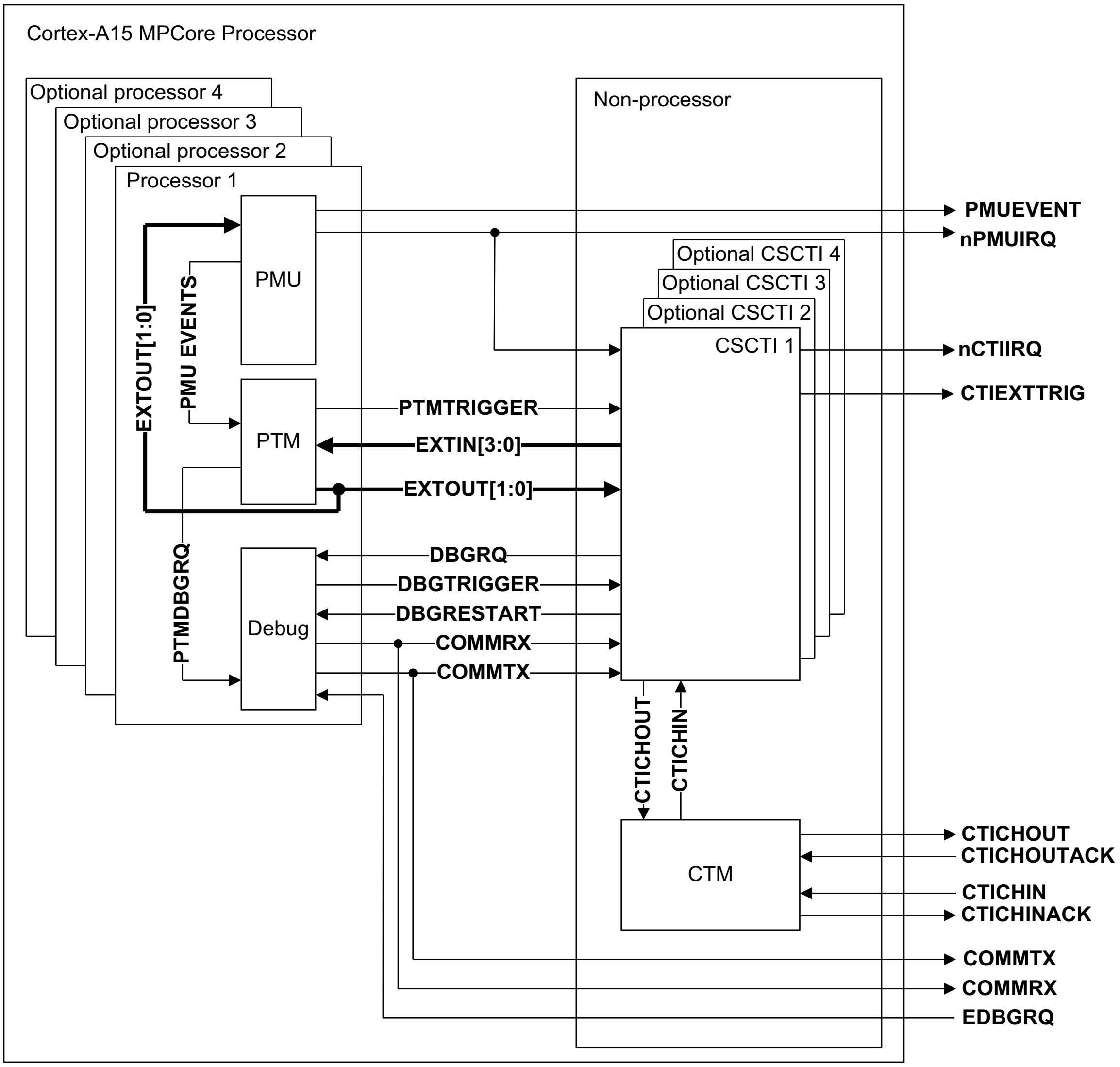


Figure 13-1 Debug system components

13.2 Trigger inputs and outputs

This section describes the trigger inputs and outputs that are available to the CTI.

Table 13-1 shows the CTI inputs.

Table 13-1 Trigger inputs

CTI input	Name	Description
0	**DBGTRIGGER**, pulsed	Pulsed on entry to debug state
1	**PMUIRQ**[a]	PMU generated interrupt
2	**EXTOUT[0]**	PTM external output
3	**EXTOUT[1]**	PTM external output
4	**COMMTX**	Debug communication transmit channel is empty
5	**COMMRX**	Debug communication receive channel is full
6	**PTMTRIGGER**	PTM trigger
7	-	-

a. This signal is the same as **nPMUIRQ** with inverted polarity.

Table 13-2 shows the CTI outputs.

Table 13-2 Trigger outputs

CTI output	Name	Description
0	**EDBGRQ**	Causes the processor to enter debug state
1	**EXTIN[0]**	PTM external input
2	**EXTIN[1]**	PTM external input
3	**EXTIN[2]**	PTM external input
4	**EXTIN[3]**	PTM external input
5	**CTIEXTTRIG**	CTI external trigger
6	**nCTIIRQ**	CTI interrupt
7	**DBGRESTART**	Causes the processor to exit debug state

13.3 Cortex-A15 CTI

In the Cortex-A15 MPCore processor, CTI operates in the **PCLKDBG** domain and synchronizes the trigger inputs and outputs to **PCLKDBG** where required. Handshaking is required for all trigger outputs. Because the simplified CTM is implemented in the same clock domain, synchronization and handshaking is not required for channel interface. In addition, APB synchronization is not required. All the trigger inputs are masked by internal **NIDEN**. Only the trigger output signals **EDBGRQ**, **CTIIRQ** and **DBGRESTART** are masked by internal **DBGEN**.

13.4 Cortex-A15 CTM

The CoreSight CTI channel signals from all the processors are combined using a simplified *Cross Trigger Matrix* (CTM) block so that a single cross trigger channel interface is presented in the Cortex-A15 processor. This module can combine up to four internal channel interfaces corresponding to each processor along with one external channel interface.

In the simplified CTM, external channel output is driven by the OR output of all internal channel outputs. Each internal channel input is driven by the OR output of internal channel outputs of all other CTIs in addition to the external channel input. The internal channel acknowledgement signals from the CTIs are not used because all the CTIs and the CTM are in the same **PCLKDBG** domain.

Chapter 14
NEON and VFP Unit

This chapter describes the Cortex-A15 NEON and VFP unit, their features, and the registers that they use. This chapter contains the following sections:

14.1 About NEON and VFP unit

NEON technology is the implementation of the Advanced *Single Instruction Multiple Data* (SIMD) extension to the ARMv7-A architecture. It provides support for integer and floating-point vector operations. This technology extends the processor functionality to provide support for the ARMv7 Advanced SIMDv2 instruction set.

VFP is the vector floating-point coprocessor extension to the ARMv7-A architecture. It provides low-cost high performance floating-point computation. VFP extends the processor functionality to provide support for the ARMv7 VFPv4 instruction set.

You can configure the Cortex-A15 MPCore processor to include different combinations of support for Advanced SIMD and VFP extensions. Table 14-1 shows the possible combinations.

Table 14-1 Combinations of Advanced SIMD and VFP extensions

Advanced SIMDv2	VFPv4
Supported	Supported
Not supported	Supported
Not supported	Not supported

This section describes the following:

- *Advanced SIMDv2 support*.
- *VFPv4 support*.

14.1.1 Advanced SIMDv2 support

The processor supports all addressing modes, data types, and operations in the Advanced SIMDv2 extension. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for information on the Advanced SIMDv2 instruction set.

14.1.2 VFPv4 support

The processor supports all addressing modes, data types, and operations in the VFPv4 extension with version 3 of the Common VFP subarchitecture. The processor implements VFPv4-D32. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for information on the VFPv4 instruction set.

In the Cortex-A15 VFP implementation:

- All scalar operations are implemented entirely in hardware, with support for all combinations of rounding modes, flush-to-zero, and default NaN modes.
- Vector operations are not supported. Any attempt to execute a vector operation results in an Undefined Instruction exception. If an application requires VFP vector operation, then it must use VFP support code. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for information on VFP vector operation support.
- The Cortex-A15 VFP implementation does not generate asynchronous VFP exceptions.

Note

An attempt to execute a vector operation that results in an Undefined Instruction exception does not set the FPEXC.DEX bit.

14.2 Programmers model for NEON and VFP unit

This section describes the programmers model for the Cortex-A15 NEON and VFP unit in:

- *Accessing the Advanced SIMD and VFP feature identification registers*.
- *Enabling Advanced SIMD and VFP extensions*.
- *Register summary* on page 14-6.
- *Register descriptions* on page 14-6.

14.2.1 Accessing the Advanced SIMD and VFP feature identification registers

Software can identify the versions of the ARMv7 Advanced SIMD and VFP extensions, and the features they provide, using the feature identification registers. These registers reside in the coprocessor space for coprocessors CP10 and CP11.

You can access the feature identification registers using the VMRS and VMSR instructions, for example:

```
VMRS <Rd>, FPSID ; Read Floating-Point System ID Register
VMRS <Rd>, MVFR0 ; Read Media and VFP Feature Register 0
VMRS <Rd>, MVFR1 ; Read Media and VFP Feature Register 1
```

Table 14-2 lists the feature identification registers for the Advanced SIMD and VFP extensions.

Table 14-2 Advanced SIMD and VFP feature identification registers

Name	Description
FPSID	See *Floating-Point System ID Register* on page 14-6
MVFR0	See *Media and VFP Feature Register 0* on page 14-11
MVFR1	See *Media and VFP Feature Register 1* on page 14-10

14.2.2 Enabling Advanced SIMD and VFP extensions

From reset, both the Advanced SIMD and VFP extensions are disabled. Any attempt to execute either an Advanced SIMD or VFP instruction results in an Undefined Instruction exception being taken. To enable software access to the Advanced SIMD and VFP features, ensure that:

- Access to CP10 and CP11 is enabled for the appropriate privilege level. See *Coprocessor Access Control Register* on page 4-62.
- If Non-secure access to the Advanced SIMD or VFP features is required, the access bits for CP10 and CP11 in the NSACR are set to 1. See *Non-Secure Access Control Register* on page 4-65.
- If Hyp mode access to the Advanced SIMD or VFP features is required, the trap bits for CP10 and CP11 in the HCPTR are set to 0. See *Hyp Coprocessor Trap Register* on page 4-71.

To enable Advanced SIMD and VFP operations, software must set the FPEXC.EN bit to 1. See *Floating-Point Exception Register* on page 14-12.

When Advanced SIMD and VFP operation is disabled because FPEXC.EN is 0, all Advanced SIMD and VFP instructions are treated as UNDEFINED except for execution of the following in privileged modes:

- A VMSR to the FPEXC or FPSID register.
- A VMRS from the FPEXC, FPSID, MVFR0 or MVFR1 register.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information on enabling Advanced SIMD and VFP support.

Using the Advanced SIMD and VFP in Secure state only

To use the Advanced SIMD and VFP in Secure state only, you must first program the CPACR and FPEXC registers. See *Coprocessor Access Control Register* on page 4-62 and *Floating-Point Exception Register* on page 14-12.

1. Enable access to CP10 and CP11 and clear the ASEDIS bit in the CPACR:

```
MOV r0, 0x00F00000
MCR p15, 0, r0, c1, c0, 2
ISB
```

2. Set the FPEXC.EN bit to enable Advanced SIMD and VFP:

```
MOV r3, #0x40000000
VMSR FPEXC, r3
```

Using the Advanced SIMD and VFP in Secure state and Non-secure state other than Hyp mode

To use the Advanced SIMD and VFP in Secure state and Non-secure state other than Hyp mode, you must first define the NSACR, then define the CPACR and FPEXC registers. See *Non-Secure Access Control Register* on page 4-65, *Coprocessor Access Control Register* on page 4-62, and *Floating-Point Exception Register* on page 14-12.

1. Enable Non-secure access to CP10 and CP11 and clear the NSASEDIS bit in the NSACR:

```
MRC p15, 0, r0, c1, c1, 2
ORR r0, r0, #(3<<10)    ; Enable Non-secure access to CP10 and CP11
BIC r0, r0, #(3<<14)    ; Clear NSASEDIS bit
MCR p15, 0, r0, c1, c1, 2
ISB
```

2. Enable access to CP10 and CP11 and clear the ASEDIS bit in the CPACR:

```
MOV r0, 0x00F00000
MCR p15, 0, r0, c1, c0, 2
ISB
```

3. Set the FPEXC.EN bit to enable Advanced SIMD and VFP:

```
MOV r3, #0x40000000
VMSR FPEXC, r3
```

Using the Advanced SIMD and VFP in Hyp mode

To use the Advanced SIMD and VFP in Hyp mode, you must first define the NSACR, then define the HCPTR and FPEXC registers.

1. Enable Non-secure access to CP10 and CP11 and clear the NSASEDIS bit in the NSACR:

```
MRC p15, 0, r0, c1, c1, 2
ORR r0, r0, #(3<<10);    Enable Non-secure access to CP10 and CP11
BIC r0, r0, #(3<<14);    Clear the NSASEDIS bit
MCR p15, 0, r0, c1, c1, 2
ISB
```

2. Clear the TCP10, TCP11, and TASE bits in the HCPTR:

```
MRC p15, 4, r0, c1, c1, 2
BIC r0, r0, #(3<<10);    Clear the TCP10 and TCP11 bits
BIC r0, r0, #(3<<14);    Clear the TASE bit
```

```
MCR p15, 4, r0, c1, c1, 2
ISB
```

3. Set the FPEXC.EN bit to enable Advanced SIMD and VFP:

```
MOV r3, #0x40000000
VMSR FPEXC, r3
```

At this point the processor can execute Advanced SIMD and VFP instructions.

Note

Operation is UNPREDICTABLE if you configure the *Coprocessor Access Control Register* (CPACR) such that CP10 and CP11 do not have identical access permissions.

14.2.3 Register summary

Table 14-3 gives a summary of the Cortex-A15 Advanced SIMD and VFP system registers.

Table 14-3 Advanced SIMD and VFP system registers

Name	Type	Reset	Description
FPSID	RO	`0x410430F0`	See *Floating-Point System ID Register*
FPSCR	RW	`0x00000000`	See *Floating-Point Status and Control Register* on page 14-7
MVFR1	RO	`0x11111111`	See *Media and VFP Feature Register 1* on page 14-10
MVFR0	RO	`0x10110222`	See *Media and VFP Feature Register 0* on page 14-11
FPEXC	RW	`0x00000000`	See *Floating-Point Exception Register* on page 14-12

Note

The *Floating-Point Instruction Registers*, FPINST and FPINST2 are not implemented, and any attempt to access them is UNPREDICTABLE.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for information on permitted accesses to the Advanced SIMD and VFP system registers.

14.2.4 Register descriptions

This section describes the Cortex-A15 Advanced SIMD and VFP system registers. Table 14-3 provides cross references to individual registers.

Floating-Point System ID Register

The FPSID characteristics are:

Purpose Provides top-level information about the floating-point implementation.

Usage constraints Only accessible from PL1 or higher.

Configurations Available if VFP is implemented. The FPSID register is a Configurable access register which is a system control register that secure software can configure the access to. When the settings in the CPACR, see *Coprocessor Access Control Register* on page 4-62, permit access to the FPSID register:

- It is accessible in Non-secure state only if the NSACR.{CP11, CP10} bits are both set to 1. See *Non-Secure Access Control Register* on page 4-65.
- Bits in the HCPTR, see *Hyp Coprocessor Trap Register* on page 4-71, also control Non-secure access to the register.

Attributes See the register summary in Table 14-3 on page 14-6.

Figure 14-1 shows the FPSID bit assignments.

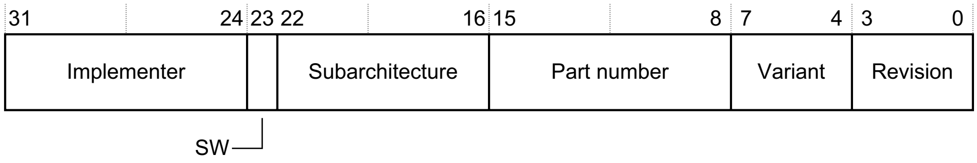


Figure 14-1 FPSID bit assignments

Table 14-4 shows the FPSID bit assignments.

Table 14-4 FPSID bit assignments

Bits	Name	Function
[31:24]	Implementer	Indicates the implementer: `0x41` ARM Limited.
[23]	SW	Software bit. This bit indicates that a system provides only software emulation of the VFP floating-point instructions: `0x0` The system includes hardware support for VFP floating-point operations.
[22:16]	Subarchitecture	Subarchitecture version number: `0x04` VFP architecture v4 with Common VFP subarchitecture v3. The VFP architecture version is indicated by the MVFR0 and MVFR1 registers.
[15:8]	Part number	Indicates the part number for the floating-point implementation: `0x30` VFP.
[7:4]	Variant	Indicates the variant number: `0xF` Cortex-A15.
[3:0]	Revision	Indicates the revision number for the floating-point implementation: `0x0` Revision 0.

Floating-Point Status and Control Register

The FPSCR characteristics are:

Purpose Provides status information and control of unprivileged execution for the floating-point system.

Usage constraints There are no usage constraints.

Configurations Available if VFP is implemented. The FPSCR register is a Configurable access register which is a system control register that secure software can configure the access to. When the settings in the CPACR, see *Coprocessor Access Control Register* on page 4-62, permit access to the FPSCR register:

- It is accessible in Non-secure state only if the NSACR.{CP11, CP10} bits are both set to 1. See *Non-Secure Access Control Register* on page 4-65.
- Bits in the HCPTR, see *Hyp Coprocessor Trap Register* on page 4-71, also control Non-secure access to the register.

Attributes See the register summary in Table 14-3 on page 14-6.

Figure 14-2 shows the FPSCR bit assignments.

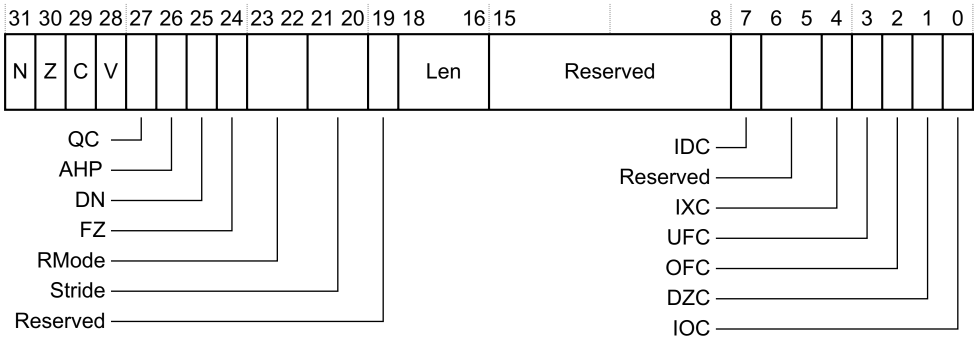


Figure 14-2 FPSCR bit assignments

Table 14-5 shows the FPSCR bit assignments.

Table 14-5 FPSCR bit assignments

Bits	Field	Function
[31]	N	VFP Negative condition code flag. Set to 1 if a VFP comparison operation produces a less than result.
[30]	Z	VFP Zero condition code flag. Set to 1 if a VFP comparison operation produces an equal result.
[29]	C	VFP Carry condition code flag. Set to 1 if a VFP comparison operation produces an equal, greater than, or unordered result.
[28]	V	VFP Overflow condition code flag. Set to 1 if a VFP comparison operation produces an unordered result.
[27]	QC	Cumulative saturation bit. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated after 0 was last written to this bit. If Advanced SIMD is not implemented, this bit is UNK/SBZP.
[26]	AHP	Alternative Half-Precision control bit: **0** IEEE half-precision format selected. **1** Alternative half-precision format selected.

Table 14-5 FPSCR bit assignments (continued)

Bits	Field	Function
[25]	DN	Default NaN mode control bit: **0** NaN operands propagate through to the output of a floating-point operation. **1** Any operation involving one or more NaNs returns the Default NaN. The value of this bit only controls VFP arithmetic. Advanced SIMD arithmetic always uses the Default NaN setting, regardless of the value of the DN bit.
[24]	FZ	Flush-to-zero mode control bit: **0** Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. **1** Flush-to-zero mode enable. The value of this bit only controls VFP arithmetic. Advanced SIMD arithmetic always uses the Flush-to-zero setting, regardless of the value of the FZ bit.
[23:22]	RMode	Rounding Mode control field: `b00` *Round to Nearest* (RN) mode. `b01` *Round towards Plus Infinity* (RP) mode. `b10` *Round towards Minus Infinity* (RM) mode. `b11` *Round towards Zero* (RZ) mode. The specified rounding mode is used by almost all VFP floating-point instructions. Advanced SIMD arithmetic always uses the Round to Nearest setting, regardless of the value of the RMode bits.
[21:20]	Stride	Use of non-zero value in this field for VFP short vector operation is deprecated in ARMv7. If this field is set to a non-zero value, the VFP data processing operations, except Vector Compare and Vector Convert instructions, generate an Undefined Instruction exception. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[19]	-	UNK/SBZP.
[18:16]	Len	Use of non-zero value in this field for VFP short vector operation is deprecated in ARMv7. If this field is set to a non-zero value, the VFP data processing operations, except Vector Compare and Vector Convert instructions, generate an Undefined Instruction exception. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*for more information.
[15]	-	RAZ/SBZP.
[14:13]	-	UNK/SBZP.
[12:8]	-	RAZ/SBZP.
[7]	IDC	Input Denormal cumulative exception bit.
[6:5]	-	UNK/SBZP.
[4]	IXC	Inexact cumulative exception bit.
[3]	UFC	Underflow cumulative exception bit.
[2]	OFC	Overflow cumulative exception bit.
[1]	DZC	Division by Zero cumulative exception bit.
[0]	IOC	Invalid Operation cumulative exception bit.

Media and VFP Feature Register 1

The MVFR1 characteristics are:

Purpose Together with MVFR0, describes the features provided by the Advanced SIMD and VFP extensions.

Usage constraints Only accessible from PL1 or higher.

Configurations Available if VFP is implemented. The MVFR1 register is a Configurable access register which is a system control register that secure software can configure the access to. When the settings in the CPACR, see *Coprocessor Access Control Register* on page 4-62, permit access to the MVFR1 register:

- It is accessible in Non-secure state only if the NSACR.{CP11, CP10} bits are both set to 1. See *Non-Secure Access Control Register* on page 4-65.
- Bits in the HCPTR, see *Hyp Coprocessor Trap Register* on page 4-71, also control Non-secure access to the register.

Attributes See the register summary in Table 14-3 on page 14-6.

Figure 14-3 shows the MVFR1 bit assignments.

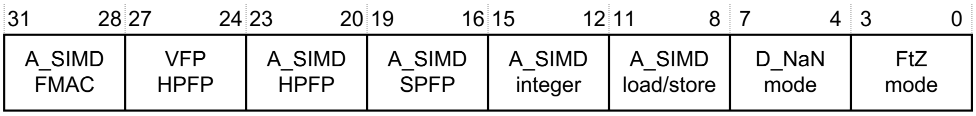


Figure 14-3 MVFR1 bit assignments

Table 14-6 shows the MVFR1 bit assignments.

Table 14-6 MVFR1 bit assignments

Bits	Name	Function
[31:28]	A_SIMD FMAC	Indicates whether the Advanced SIMD or VFP supports fused multiply accumulate operations: `0x1` Supported.
[27:24]	VFP HPFP	Indicates whether the VFP supports half-precision floating-point conversion operations: `0x1` Supported.
[23:20]	A_SIMD HPFP	Indicates whether the Advanced SIMD extension supports half-precision floating-point conversion operations: `0x1` Supported. If Advanced SIMD is implemented, the reset value is `0x1`. If Advanced SIMD is not implemented, the reset value is `0x0`.
[19:16]	A_SIMD SPFP	Indicates whether the Advanced SIMD extension supports single-precision floating-point operations: `0x1` Supported. If Advanced SIMD is implemented, the reset value is `0x1`. If Advanced SIMD is not implemented, the reset value is `0x0`.
[15:12]	A_SIMD integer	Indicates whether the Advanced SIMD extension supports integer operations: `0x1` Supported. If Advanced SIMD is implemented, the reset value is `0x1`. If Advanced SIMD is not implemented, the reset value is `0x0`.

Table 14-6 MVFR1 bit assignments (continued)

Bits	Name	Function
[11:8]	A_SIMD load/store	Indicates whether the Advanced SIMD extension supports load/store instructions: `0x1` Supported. If Advanced SIMD is implemented, the reset value is `0x1`. If Advanced SIMD is not implemented, the reset value is `0x0`.
[7:4]	D_NaN mode	Indicates whether the VFP hardware implementation supports only the Default NaN mode: `0x1` Hardware supports propagation of NaN values.
[3:0]	FtZ mode	Indicates whether the VFP hardware implementation supports only the Flush-to-Zero mode of operation: `0x1` Hardware supports full denormalized number arithmetic.

Media and VFP Feature Register 0

The MVFR0 characteristics are:

Purpose Together with MVFR1, describes the features provided by the Advanced SIMD and VFP extensions.

Usage constraints Only accessible from PL1 or higher.

Configurations Available if VFP is implemented. The MVFR0 register is a Configurable access register which is a system control register that secure software can configure the access to. When the settings in the CPACR, see *Coprocessor Access Control Register* on page 4-62, permit access to the MVFR0 register:

- It is accessible in Non-secure state only if the NSACR.{CP11, CP10} bits are both set to 1. See *Non-Secure Access Control Register* on page 4-65.
- Bits in the HCPTR, see *Hyp Coprocessor Trap Register* on page 4-71, also control Non-secure access to the register.

Attributes See the register summary in Table 14-3 on page 14-6.

Figure 14-4 shows the MVFR0 bit assignments.

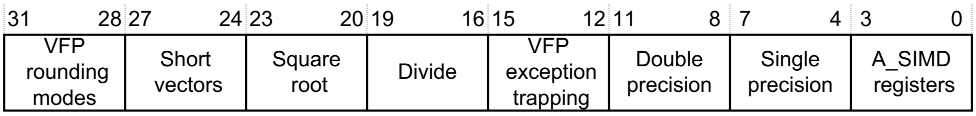


Figure 14-4 MVFR0 bit assignments

Table 14-7 shows the MVFR0 bit assignments.

Table 14-7 MVFR0 bit assignments

Bits	Name	Function
[31:28]	VFP rounding modes	Indicates the rounding modes supported by the VFP floating-point hardware: `0x1` Supported.
[27:24]	Short vectors	Indicates the hardware support for VFP short vectors: `0x0` Not supported.
[23:20]	Square root	Indicates the hardware support for VFP square root operations: `0x1` Supported.
[19:16]	Divide	Indicates the hardware support for VFP divide operations: `0x1` Supported.
[15:12]	VFP exception trapping	Indicates whether the VFP hardware implementation supports exception trapping: `0x0` Not supported.
[11:8]	Double precision	Indicates the hardware support for VFP double-precision operations: `0x2` VFPv4 double-precision supported. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[7:4]	Single precision	Indicates the hardware support for VFP single-precision operations: `0x2` VFPv4 single-precision supported. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.
[3:0]	A_SIMD registers	Indicates support for the Advanced SIMD register bank: `0x2` 32 x 64-bit registers supported. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

Floating-Point Exception Register

The FPEXC characteristics are:

Purpose Provides a global enable for the Advanced SIMD and VFP extensions, and indicates how the state of these extensions is recorded.

Usage constraints Only accessible from PL1 or higher.

Configurations Available if VFP is implemented. The FPEXC register is a Configurable access register which is a system control register that secure software can configure the access to. When the settings in the CPACR, see *Coprocessor Access Control Register* on page 4-62, permit access to the FPEXC register:

- It is accessible in Non-secure state only if the NSACR.{CP11, CP10} bits are both set to 1. See *Non-Secure Access Control Register* on page 4-65.
- Bits in the HCPTR, see *Hyp Coprocessor Trap Register* on page 4-71, also control Non-secure access to the register.

Attributes See the register summary in Table 14-3 on page 14-6.

Figure 14-5 on page 14-13 shows the FPEXC bit assignments.

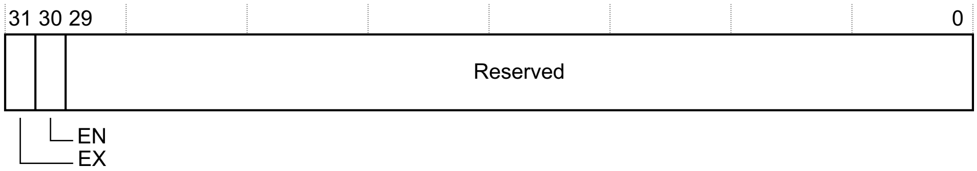


Figure 14-5 FPEXC bit assignments

Table 14-8 shows the FPEXC Register bit assignments.

Table 14-8 FPEXC bit assignments

Bits	Name	Function
[31]	EX	Exception bit. The Cortex-A15 implementation does not generate asynchronous VFP exceptions, therefore this bit is RAZ/WI.
[30]	EN	Enable bit. A global enable for the Advanced SIMD and VFP extensions: **0** The Advanced SIMD and VFP extensions are disabled. **1** The Advanced SIMD and VFP extensions are enabled and operate normally. The EN bit is cleared at reset.
[29:26]	-	Reserved, RAZ/WI.
[25:0]	-	Reserved, UNK/SBZP.

Note

The Cortex-A15 MPCore processor implementation does not support deprecated VFP short vector feature. Attempts to execute VFP data-processing instructions, except VFP Compare and VFP Convert instructions, when the FPSCR.LEN field is non-zero result in an Undefined Instruction exception. You can use software to emulate the short vector feature, if required.

Appendix A
Signal Descriptions

This appendix describes the Cortex-A15 MPCore processor signals. It contains the following sections:

A.1 About the signal descriptions

The tables in this appendix list the Cortex-A15 MPCore processor signals, along with their direction, input or output, and a high-level description.

A.2 Clock signals

Table A-1 shows the clock and clock enable signals.

Table A-1 Clock and clock enable signals

Signal	Type	Description
CLK	Input	Global clock.
CLKEN	Input	Global clock enable. This signal can only be deasserted with all the processors in the MPCore device and L2 are in WFI low-power state, and both the ACE and ACP are idle.
CPUCLKOFF[N:0][a]	Input	Individual processor clock disable. **0** Processor clock is enabled. **1** Processor clock is stopped. This signal is only present if the Cortex-A15 MPCore processor is configured with the **CPUCLKOFF** pins. The default configuration does not include the **CPUCLKOFF** pins. This signal can only be asserted when the processor is already powered down, or when the processor is powered up. To enable the powerup reset sequence to complete, this signal must be deasserted after power has been completely restored. See *Clocks* on page 2-8 for more information.

[a] These signals are not available in revisions prior to r3p0.

See *Clocking and resets* on page 2-8 for more information.

A.3 Reset signals

Table A-2 shows the reset and reset control signals. The value of **N** is one less than the number of processors in your design.

Table A-2 Reset signals

Signal	Type	Description
nCPUPORESET[N:0]	Input	Individual processor resets: **0** Apply reset to processor that includes NEON and VFP, Debug, PTM, breakpoint and watchpoint logic. **1** Do not apply reset to processor that includes NEON and VFP, Debug, PTM, breakpoint and watchpoint logic.
nCORERESET[N:0]	Input	Individual processor reset excluding Debug and PTM: **0** Apply reset to processor that includes NEON and VFP, but excludes Debug, PTM, breakpoint and watchpoint logic. **1** Do not apply reset to processor that includes NEON and VFP, but excludes Debug, PTM, breakpoint and watchpoint logic.
nCXRESET[N:0]	Input	Individual processor NEON and VFP resets: **0** Apply reset to NEON and VFP. **1** Do not apply reset to NEON and VFP.
nDBGRESET[N:0]	Input	Individual processor Debug and PTM resets: **0** Apply reset to Debug, PTM, breakpoint and watchpoint logic. **1** Do not apply reset to Debug, PTM, breakpoint and watchpoint logic.
nL2RESET	Input	L2 reset: **0** Apply reset to shared L2 memory system controller. **1** Do not apply reset to shared L2 memory system controller.
L2RSTDISABLE	Input	L2 cache hardware reset disable: **0** L2 cache is reset by hardware. **1** L2 cache is not reset by hardware.

See *Clocking and resets* on page 2-8 for more information.

A.4 Configuration signals

Table A-3 shows the configuration signals. The value of **N** is one less than the number of processors in your design.

Table A-3 Configuration signals

Signal	Type	Description
CFGEND[N:0]	Input	Individual processor control of the endianness configuration at reset. It sets the initial value of the EE bit in the CP15 System Control Register (SCTLR): **0** EE bit is LOW. **1** EE bit is HIGH. This signal is only sampled during reset of the processor.
CFGTE[N:0]	Input	Individual processor control of the default exception handling state. It sets the initial value of the TE bit in the CP15 System Control Register (SCTLR): **0** TE bit is LOW. **1** TE bit is HIGH. This signal is only sampled during reset of the processor.
CLUSTERID[3:0]	Input	Value read in the Cluster ID field, bits[11:8], of the CP15 processor Affinity Register (MPDIR). This signal is only sampled during reset of the processor.
IMINLN	Input	Individual processor control of the instruction cache minimum line size at reset. It sets the initial value of the IminLine field in the CP15 Cache Type Register (CTR): **0** 32-bytes. **1** 64-bytes. This signal is only sampled during reset of the processor.
VINITHI[N:0]	Input	Individual processor control of the location of the exception vectors at reset. It sets the initial value of the V bit in the CP15 System Control Register (SCTLR): **0** Exception vectors start at address `0x00000000`. **1** Exception vectors start at address `0xFFFF0000`. This signal is only sampled during reset of the processor.
CP15SDISABLE[N:0]	Input	Disable write access to some secure CP15 registers.

A.5 Generic Interrupt Controller signals

Table A-4 shows the GIC signals. The value of **N** is one less than the number of processors in your design.

Table A-4 GIC signals

Signal	Type	Description
CFGSDISABLE[a]	Input	Disable write access to some secure GIC registers.
IRQS[n:0][b]	Input	Interrupt request input lines for the GIC where **n** can be 31, 63, up to 223 by increments of 32.
nIRQ[N:0]	Input	Individual processor IRQ request input lines. Active-LOW, interrupt request: **0** Activate interrupt. **1** Do not activate interrupt. The processor treats the **nIRQ** input as level-sensitive. To guarantee that an interrupt is taken, the **nIRQ** input must be asserted until the processor acknowledges the interrupt.
nFIQ[N:0]	Input	Individual processor FIQ request input line. Active-LOW, FIQ request: **0** Activate FIQ request. **1** Do not activate FIQ request. The processor treats the **nFIQ** input as level-sensitive. To guarantee that an interrupt is taken, the **nFIQ** input must be asserted until the processor acknowledges the interrupt.
nVIRQ[N:0]	Input	Individual processor virtual IRQ request input lines. Active-LOW, interrupt request: **0** Activate virtual IRQ request. **1** Do not activate virtual IRQ request. The processor treats the **nVIRQ** input as level-sensitive. To guarantee that an interrupt is taken, the **nVIRQ** input must be asserted until the processor acknowledges the interrupt. If the Cortex-A15 MPCore processor is configured to include the GIC, and the GIC is used, the input pins **nVIRQ** and **nVFIQ** must be tied off to HIGH. If the processor is configured to include the GIC, and the GIC is not used, the input pins **nVIRQ** and **nVFIQ** can be driven by an external GIC in the SoC. See *GIC configuration* on page 8-6 for more information.
nVFIQ[N:0]	Input	Individual processor virtual FIQ request input lines. Active-LOW, virtual FIQ request: **0** Activate virtual FIQ request. **1** Do not activate virtual FIQ request. The processor treats the **nVFIQ** input as level-sensitive. To guarantee that an interrupt is taken, the **nVFIQ** input must be asserted until the processor acknowledges the interrupt. If the Cortex-A15 MPCore processor is configured to include the GIC, and the GIC is used, the input pins **nVIRQ** and **nVFIQ** must be tied off to HIGH. If the processor is configured to include the GIC, and the GIC is not used, the input pins **nVIRQ** and **nVFIQ** can be driven by an external GIC in the SoC. See *GIC configuration* on page 8-6 for more information.
nIRQOUT[N:0][a]	Output	Active-LOW output of individual processor **nIRQ** from the GIC. For use when processors are powered down and interrupts from the GIC are routed to an external power controller.

Table A-4 GIC signals (continued)

Signal	Type	Description
nFIQOUT[N:0][a]	Output	Active-LOW output of individual processor **nFIQ** from the GIC. For use when processors are powered down and interrupts from the GIC are routed to an external power controller.
PERIPHBASE[39:15]	Input	Specifies the base address for the GIC registers. This value is sampled into the CP15 Configuration Base Address Register (CBAR) at reset. See *Configuration Base Address Register* on page 4-106.
PERIPHCLKEN[a]	Input	GIC clock enable.

[a] This signal is not present if the Cortex-A15 MPCore processor is configured without the GIC.

[b] This signal is not present if the Cortex-A15 MPCore processor is configured with zero *Shared Peripheral Interrupt* (SPI) inputs or without the GIC.

A.6 Generic Timer signals

Table A-5 shows the Generic Timer signals. The value of **N** is one less than the number of processors in your design.

Table A-5 Generic Timer signals

Signal	Type	Description
nCNTHPIRQ[N:0]	Output	Hypervisor physical timer event
nCNTPNSIRQ[N:0]	Output	Non-secure physical timer event
nCNTPSIRQ[N:0]	Output	Secure physical timer event
nCNTVIRQ[N:0]	Output	Virtual timer event
CNTVALUEB[63:0]	Input	Global system counter value in binary format

A.7 WFE and WFI standby signals

Table A-6 shows the WFE and WFI standby signals. The value of N is one less than the number of processors in your design.

Table A-6 WFE and WFI standby signals

Signal	Type	Description
EVENTI	Input	Event input for processor wake-up from WFE standby mode. When this signal is asserted, it acts as a WFE wake-up event to all the processors in the MPCore device. This signal must be asserted for at least one **CLK** cycle. See *Event communication using WFE and SEV instructions* on page 2-37 for more information.
EVENTO	Output	Event output. This signal is asserted HIGH for three **CLK** cycles when any of the processors in the MPCore device executes an SEV instruction. See *Event communication using WFE and SEV instructions* on page 2-37 for more information.
STANDBYWFE[N:0]	Output	Indicates if a processor is in WFE standby mode: **0** Processor not in WFE standby mode. **1** Processor in WFE standby mode.
STANDBYWFI[N:0]	Output	Indicates if a processor is in WFI standby mode: **0** Processor not in WFI standby mode. **1** Processor in WFI standby mode.
STANDBYWFIL2	Output	Indicates if L2 is in WFI standby mode. This signal is active when the following are true: • All processors are in standby WFI. • **ACINACTM** and **AINACTS** are asserted HIGH. • L2 is idle.

A.8 Power management signals

Table A-7 shows the power management signals. The value of **N** is one less than the number of processors in your design.

Table A-7 Power management signals

Signal	Type	Description
nISOLATECPU[N:0][a]	Input	Individual processor clamp control: **0** Processor clamp active. **1** Processor clamp not active.
nISOLATECX[N:0][a]	Input	Individual NEON and VFP clamp control: **0** NEON and VFP clamp active. **1** NEON and VFP clamp not active.
nISOLATEL2MISC[a]	Input	L2 control, GIC, and Timer clamp control: **0** L2 control, GIC, and Timer clamp active. **1** L2 control, GIC, and Timer clamp not active.
nISOLATEPDBG[a]	Input	Debug, CTI, and CTM in the **PCLKDBG** domain clamp control: **0** Debug, CTI, and CTM in the **PCLKDBG** domain clamp active. **1** Debug, CTI, and CTM in the **PCLKDBG** domain clamp not active.
nPWRUPCPU[N:0][a]	Input	Individual processor power switch enable: **0** Processor power switch enabled. **1** Processor power switch not enabled.
nPWRUPCX[N:0][a]	Input	Individual NEON and VFP power switch enable: **0** NEON and VFP power switch enabled. **1** NEON and VFP power switch not enabled.
nPWRUPL2MISC[a]	Input	L2 control, GIC, and Timer power switch enable: **0** L2 control, GIC, and Timer power switch enabled. **1** L2 control, GIC, and Timer power switch not enabled.
nPWRUPL2RAM[a]	Input	L2 tag bank RAMs power switch enable: **0** L2 tag bank RAMs power switch enabled. **1** L2 tag bank RAMs power switch not enabled.
nPWRUPPDBG[a]	Input	Debug, CTI, and CTM in the **PCLKDBG** domain power switch enable: **0** Debug, CTI, and CTM in the **PCLKDBG** domain power switch enabled. **1** Debug, CTI, and CTM in the **PCLKDBG** domain power switch not enabled.
QACTIVE[N:0][b]	Output	Processor activity hint: **0** Processor not active, possible retention. **1** Processor active.

Table A-7 Power management signals (continued)

Signal	Type	Description
QREQn[N:0][b]	Input	Processor retention mode request: **0** External power controller requesting retention mode. **1** External power controller not requesting retention mode.
QACCEPTn[N:0][b]	Output	Processor retention mode acceptance: **0** Retention mode request accepted. **1** Retention mode request not yet accepted.
QDENY[N:0][b]	Output	Processor retention mode denial: **0** Retention mode request not yet denied. **1** Retention mode request denied.

[a] This signal is not present if the Cortex-A15 MPCore processor is configured without the power switch and clamp control signals.

[b] This signal is not available in revisions prior to r3p0.

A.9 AXI interfaces

For descriptions of AXI interface signals, see the *AMBA AXI Protocol Specification*. In this section, the suffix M in the signal name signifies an AXI master interface signal and the suffix S signifies an AXI slave interface signal.

This section describes the AXI interfaces in:

- *AXI master interface signals*.
- *ACP signals* on page A-17.

A.9.1 AXI master interface signals

The following sections describe the AXI master interface signals:

- *Clock and configuration signals*.
- *Asynchronous error signals* on page A-13.
- *Write address channel signals* on page A-14.
- *Write data channel signals* on page A-14.
- *Write response channel signals* on page A-15.
- *Read address channel signals* on page A-15.
- *Read data channel signals* on page A-15.
- *Snoop address channel signals* on page A-16.
- *Snoop data channel signals* on page A-16.
- *Snoop response channel signals* on page A-16.
- *Read/write acknowledge signals* on page A-17.

Clock and configuration signals

Table A-8 shows the clock and configuration signals for the AXI master interface.

Table A-8 Clock and configuration signals

Signal	Type	Description
ACLKENM	Input	AXI master bus clock enable. See *Clocking and resets* on page 2-8 for more information.
ACINACTM	Input	Snoop interface is inactive and no longer accepting requests.
A64n128M	Input	Selects 64-bit or 128-bit AXI bus width: **0** 128-bit bus width. **1** 64-bit bus width.
BROADCASTCACHEMAINT	Input	Enable broadcasting of cache maintenance operations to downstream caches: **0** Cache maintenance operations are not broadcasted to downstream caches. **1** Cache maintenance operations are broadcasted to downstream caches. This signal is only sampled during reset of the processor. See *ACE configurations* on page 7-15 for more information.

Table A-8 Clock and configuration signals (continued)

Signal	Type	Description
BROADCASTINNER	Input	Enable broadcasting of inner shareable transactions: **0** Inner shareable transactions are not broadcasted externally. **1** Inner shareable transactions are broadcasted externally. If **BROADCASTINNER** is tied HIGH, you must also tie **BROADCASTOUTER** HIGH. This signal is only sampled during reset of the processor. See *ACE configurations* on page 7-15 for more information.
BROADCASTOUTER	Input	Enable broadcasting of outer shareable transactions: **0** Outer shareable transactions are not broadcasted externally. **1** Outer shareable transactions are broadcasted externally. This signal is only sampled during reset of the processor. See *ACE configurations* on page 7-15 for more information.
SYSBARDISABLE	Input	Disable broadcasting of barriers onto system bus: **0** Barriers are broadcasted onto system bus, this requires an AMBA 4 interconnect. **1** Barriers are not broadcasted onto the system bus. This is compatible with an AXI3 interconnect. If **SYSBARDISABLE** is tied HIGH, you must tie the following signals LOW for full AXI3 compatibility: • **BROADCASTCACHEMAINT**. • **BROADCASTINNER**. • **BROADCASTOUTER**. This signal is only sampled during reset of the processor. See *ACE configurations* on page 7-15 for more information.

Asynchronous error signals

Table A-9 shows the asynchronous error signals.

Table A-9 Asynchronous error signals

Signal	Type	Description
nAXIERRIRQ	Output	Error indicator for AXI write transactions with a BRESP error condition. Writing 0 to bit[29] of the L2ECTLR clears the error indicator, see *L2 Extended Control Register* on page 4-87 for more information.
nINTERRIRQ	Output	Error indicator for: • L2 RAM double-bit ECC error. • Illegal writes to the GIC memory-map region, see *GIC configuration* on page 8-6. Writing 0 to bit[30] of the L2ECTLR clears the error indicator, see *L2 Extended Control Register* on page 4-87 for more information.

Write address channel signals

Table A-10 shows the write address channel signals for the AXI master interface.

Table A-10 Write address channel signals

Signal	Type	Description
AWADDRM[39:0]	Output	Address
AWBARM[1:0]	Output	Barrier type
AWBURSTM[1:0]	Output	Burst type
AWCACHEM[3:0]	Output	Cache type
AWDOMAINM[1:0]	Output	Domain type
AWIDM[5:0]	Output	Request ID
AWLENM[7:0]	Output	Burst length
AWLOCKM	Output	Lock type
AWPROTM[2:0]	Output	Protection type
AWREADYM	Input	Address ready
AWSIZEM[2:0]	Output	Burst size
AWSNOOPM[2:0]	Output	Snoop request type
AWUSERM[1:0]	Output	User signals: **b00** SharedClean **b01** UniqueClean **b10** SharedDirty **b11** UniqueDirty
AWVALIDM	Output	Address valid

Write data channel signals

Table A-11 shows the write data signals for the AXI master interface.

Table A-11 Write data channel signals

Signal	Type	Description
WDATAM[127:0]	Output	Write data
WIDM[5:0]	Output	Write ID
WLASTM	Output	Write last
WREADYM	Input	Write ready
WSTRBM[15:0]	Output	Write strobes
WVALIDM	Output	Write valid

Write response channel signals

Table A-12 shows the write response channel signals for the AXI master interface.

Table A-12 Write response channel signals

Signal	Type	Description
BIDM[5:0]	Input	Response ID
BREADYM	Output	Response ready
BRESPM[1:0]	Input	Write response
BVALIDM	Input	Response valid

Read address channel signals

Table A-13 shows the read address channel signals for the AXI master interface.

Table A-13 Read address channel signals

Signal	Type	Description
ARADDRM[39:0]	Output	Address
ARBARM[1:0]	Output	Barrier type
ARBURSTM[1:0]	Output	Burst type
ARCACHEM[3:0]	Output	Cache type
ARDOMAINM[1:0]	Output	Domain type
ARIDM[5:0]	Output	Request ID
ARLENM[7:0]	Output	Burst length
ARLOCKM	Output	Lock type
ARPROTM[2:0]	Output	Protection type
ARREADYM	Input	Address ready
ARSIZEM[2:0]	Output	Burst size
ARSNOOPM[3:0]	Output	Snoop request type
ARVALIDM	Output	Address valid

Read data channel signals

Table A-14 shows the read data channel signals for the AXI master interface.

Table A-14 Read data channel signals

Signal	Type	Description
RDATAM[127:0]	Input	Read data
RIDM[5:0]	Input	Read ID
RLASTM	Input	Read last

Table A-14 Read data channel signals (continued)

Signal	Type	Description
RREADYM	Output	Read ready
RRESPM[3:0]	Input	Read response
RVALIDM	Input	Read valid

Snoop address channel signals

Table A-15 shows the snoop address channel signals for the AXI master interface.

Table A-15 Snoop address channel signals

Signal	Type	Description
ACADDRM[39:0]	Input	Address
ACPROTM[2:0]	Input	Protection type
ACREADYM	Output	Address ready
ACSNOOPM[3:0]	Input	Transaction type
ACVALIDM	Input	Address valid

Snoop data channel signals

Table A-16 shows the snoop data channel signals for the AXI master interface.

Table A-16 Snoop data channel signals

Signal	Type	Description
CDDATAM[127:0]	Output	Snoop data
CDLASTM	Output	Snoop last
CDREADYM	Input	Snoop ready
CDVALIDM	Output	Snoop valid

Snoop response channel signals

Table A-17 shows the snoop response channel signals for the AXI master interface.

Table A-17 Snoop response channel signals

Signal	Type	Description
CRREADYM	Input	Response ready
CRRESPM[4:0]	Output	Snoop response
CRVALIDM	Output	Response valid

Read/write acknowledge signals

Table A-18 shows the read/write acknowledge signals for the AXI master interface.

Table A-18 Read/write acknowledge signals

Signal	Type	Description
RACKM	Output	Read acknowledge
WACKM	Output	Write acknowledge

A.9.2 ACP signals

The following sections describe the ACP signals:

- *Clock and configuration signals*.
- *Write address channel signals*.
- *Write data channel signals* on page A-18.
- *Write response channel signals* on page A-18.
- *Read address channel signals* on page A-19.
- *Read data channel signals* on page A-19.

Clock and configuration signals

Table A-19 shows the clock and configuration signals for the ACP.

Table A-19 Clock and configuration signals

Signal	Type	Description
A64n128S	Input	Selects 64-bit or 128-bit AXI slave bus width: **0** 128-bit bus width. **1** 64-bit bus width.
ACLKENS	Input	AXI slave bus clock enable.
AINACTS	Input	AXI slave inactive and no longer accepting requests.

Write address channel signals

Table A-20 shows the write address channel signals for the ACP.

Table A-20 Write address channel signals

Signal	Type	Description
AWADDRS[39:0]	Input	Address
AWBURSTS[1:0]	Input	Burst type
AWIDS[2:0]	Input	Request ID
AWCACHES[3:0]	Input	Cache type
AWLENS[3:0]	Input	Burst length
AWPROTS[2:0]	Input	Protection type
AWREADYS	Output	Address ready

Table A-20 Write address channel signals (continued)

Signal	Type	Description
AWSIZES[2:0]	Input	Burst size
AWUSERS[5:0]	Input	User signals: • [5:2] Inner attributes: `b0000` Strongly-ordered. `b0001` Device. `b0011` Normal Memory Non-Cacheable. `b0110` Write-Through. `b0111` Write-Back No Write-Allocate. `b1111` Write-Back Write-Allocate. • [1] Inner shareable • [0] Outer shareable.
AWVALIDS	Input	Address valid

Write data channel signals

Table A-21 shows the write data channel signals for the ACP.

Table A-21 Write data channel signals

Signal	Type	Description
WDATAS[127:0]	Input	Write data
WLASTS	Input	Write last
WREADYS	Output	Write ready
WSTRBS[15:0]	Input	Write strobes
WVALIDS	Input	Write valid

Write response channel signals

Table A-22 shows the write response channel signals for the ACP.

Table A-22 Write response channel signals

Signal	Type	Description
BIDS[2:0]	Output	Response ID
BREADYS	Input	Response ready
BRESPS[1:0]	Output	Write response
BVALIDS	Output	Response valid

Read address channel signals

Table A-23 shows the read address channel signals for the ACP.

Table A-23 Read address channel signals

Signal	Type	Description
ARADDRS[39:0]	Input	Address
ARBURSTS[1:0]	Input	Burst type
ARCACHES[3:0]	Input	Cache type
ARIDS[2:0]	Input	Request ID
ARLENS[3:0]	Input	Burst length
ARPROTS[2:0]	Input	Protection type
ARREADYS	Output	Address ready
ARSIZES[2:0]	Input	Burst size
ARUSERS[5:0]	Input	User signals: • [5:2] Inner attributes: `b0000` Strongly-ordered. `b0001` Device. `b0011` Normal Memory Non-Cacheable. `b0110` Write-Through. `b1011` Write-Back No Read-Allocate. `b1111` Write-Back Read-Write-Allocate. • [1] Inner shareable. • [0] Outer shareable.
ARVALIDS	Input	Address valid

Read data channel signals

Table A-24 shows the read data channel signals for the ACP.

Table A-24 Read data channel signals

Signal	Type	Description
RDATAS[127:0]	Output	Read data
RIDS[2:0]	Output	Read ID
RLASTS	Output	Read last
RREADYS	Input	Read ready
RRESPS[1:0]	Output	Read response
RVALIDS	Output	Read valid

A.10 External debug interface

The following sections describe the external debug interface signals:

- *APB interface signals*.
- *Authentication interface signals* on page A-21.
- *Miscellaneous debug signals* on page A-21.

A.10.1 APB interface signals

Table A-25 shows the APB interface signals.

Table A-25 APB interface signals

Signal	Type	Description
nPRESETDBG	Input	Active-LOW APB reset input: **0** Reset APB. **1** Do not reset APB.
PCLKDBG	Input	APB clock.
PCLKENDBG	Input	APB clock enable.
PADDRDBG[16:2]	Input	APB address bus bits[16:2].
PADDRDBG31	Input	APB address bus bit[31]: **0** Not an external debugger access. **1** External debugger access.
PENABLEDBG	Input	Indicates the second and subsequent cycles of an APB transfer.
PRDATADBG[31:0]	Output	APB read data bus.
PREADYDBG	Output	APB slave ready. An APB slave can assert **PREADYDBG** to extend a transfer by inserting wait states.
PSELDBG	Input	Debug registers select: **0** Debug registers not selected. **1** Debug registers selected.
PSLVERRDBG	Output	APB slave transfer error: **0** No transfer error. **1** Transfer error.
PWDATADBG[31:0]	Input	APB write data bus.
PWRITEDBG	Input	APB read or write signal: **0** Reads from APB. **1** Writes to APB.

A.10.2 Authentication interface signals

Table A-26 shows the authentication interface signals. The value of **N** is one less than the number of processors in your design.

Table A-26 Authentication interface signals

Signal	Type	Description
DBGEN[N:0]	Input	Invasive debug enable: **0** Not enabled. **1** Enabled.
NIDEN[N:0]	Input	Non-invasive debug enable: **0** Not enabled. **1** Enabled.
SPIDEN[N:0]	Input	Secure privileged invasive debug enable: **0** Not enabled. **1** Enabled.
SPNIDEN[N:0]	Input	Secure privileged non-invasive debug enable: **0** Not enabled. **1** Enabled.

A.10.3 Miscellaneous debug signals

Table A-27 shows the miscellaneous debug signals. The value of **N** is one less than the number of processors in your design.

Table A-27 Miscellaneous debug signals

Signal	Type	Description
COMMRX[N:0]	Output	Communications channel receive. Receive portion of Data Transfer Register full flag: **0** Empty. **1** Full.
COMMTX[N:0]	Output	Communication channel transmit. Transmit portion of Data Transfer Register empty flag: **0** Full. **1** Empty.
DBGACK[N:0]	Output	Debug acknowledge: **0** External debug request not acknowledged. **1** External debug request acknowledged.
DBGSWENABLE[N:0]	Input	Debug software access enable: **0** Not enabled. **1** Enabled, access by the software through the Extended CP14 interface is permitted.
EDBGRQ[N: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.**

Table A-27 Miscellaneous debug signals (continued)

Signal	Type	Description
DBGROMADDR[39:12]	Input	Specifies bits[39:12] of the ROM Table physical address. If the address cannot be determined, tie this signal off to 0. This signal is only sampled during reset of the processor.
DBGROMADDRV	Input	Valid signal for **DBGROMADDR**. If the address cannot be determined, tie this signal LOW. This signal is only sampled during reset of the processor.
DBGSELFADDR[39:17]	Input	Specifies bits[39:17] of the two's complement signed offset from the ROM Table physical address to the physical address where the debug registers are memory-mapped. If the offset cannot be determined, tie this signal off to 0. This signal is only sampled during reset of the processor.
DBGSELFADDRV	Input	Valid signal for **DBGSELFADDR**. If the offset cannot be determined, tie this signal LOW. This signal is only sampled during reset of the processor.
DBGNOPWRDWN[N:0]	Output	No powerdown request: **0** On a powerdown request, the SoC power controller powers down the processor. **1** On a powerdown request, the SoC power controller does not power down the processor.
DBGPWRDWNACK[N:0]	Output	Processor powerdown acknowledge: **0** No acknowledge for processor powerdown request. **1** Acknowledge for processor powerdown request.
DBGPWRDWNREQ[N:0]	Input	Processor powerdown request: **0** No request for processor power down. **1** Request for processor power down.
DBGPWRUPREQ[N:0]	Output	Processor powerup request: **0** No request for processor power up. **1** Request for processor power up.
DBGRSTREQ[N:0]	Output	Warm reset request: **0** No request for warm reset. **1** Request for warm reset.

A.11 PTM interface

This section describes the PTM interface in:

- *ATB interface*.
- *Miscellaneous PTM interface*.

A.11.1 ATB interface

Table A-28 shows the signals of the ATB interface. In this table, the value **x** represents processor 0, 1, 2, or 3 in your design.

Table A-28 ATB interface signals

Signal	Type	Description
AFREADYMx	Output	FIFO flush acknowledge: **0** FIFO flush not complete. **1** FIFO flush complete.
AFVALIDMx	Input	FIFO flush request.
ATBYTESMx[1:0]	Output	CoreSight ATB device data size: `b00` 1 byte. `b01` 2 byte. `b10` 3 byte. `b11` 4 byte.
ATDATAMx[31:0]	Output	ATB data bus.
ATIDMx[6:0]	Output	ATB trace source identification.
ATREADYMx	Input	ATB device ready: **0** Not ready. **1** Ready.
ATVALIDMx	Output	ATB valid data: **0** No valid data. **1** Valid data.
ATCLKEN	Input	ATB clock enable.
SYNCREQx	Input	Synchronization request. The input must be driven HIGH for one **ATCLK** cycle.

A.11.2 Miscellaneous PTM interface

Table A-29 shows the miscellaneous Program Trace Macrocell signals.

Table A-29 Miscellaneous PTM interface signals

Signal	Type	Description
TSCLKCHANGE	Input	**CLK** frequency change indicator. The input must be driven HIGH for a single **CLK** cycle if the frequency of **CLK** is changed. If not used, this input must be tied LOW.
TSVALUEB[63:0]	Input	Global system timestamp value in binary format.

A.12 Cross trigger channel interface

Table A-30 shows the cross trigger channel interface signals. The value of **N** is one less than the number of processors in your design.

Table A-30 Cross trigger channel interface signals

Signal	Type	Description
CIHSBYPASS[3:0]	Input	Cross trigger channel interface handshake bypass.
CISBYPASS	Input	Cross trigger channel interface sync bypass.
CTICHIN[3:0]	Input	Cross trigger channel input. Each bit represents a valid channel input: **0** Channel input inactive. **1** Channel input active.
CTICHINACK[3:0]	Output	Cross trigger channel input acknowledge.
CTICHOUT[3:0]	Output	Cross trigger channel output. Each bit represents a valid channel output: **0** Channel output inactive. **1** Channel output active.
CTICHOUTACK[3:0]	Input	Cross trigger channel output acknowledge.
CTIEXTTRIG[N:0]	Output	Cross trigger external trigger output.
CTIEXTTRIGACK[N:0]	Input	Cross trigger external trigger output acknowledge.
nCTIIRQ[N:0]	Output	Active-LOW cross trigger interrupt output: **0** Interrupt active. **1** Interrupt not active.

A.13 PMU signals

Table A-31 shows the performance monitoring signals. In this table, the value of **N** is one less than the number of processors and the value **x** represents processor 0, 1, 2, or 3 in your design.

Table A-31 Performance monitoring signals

Signal	Type	Description
nPMUIRQ[N:0]	Output	PMU interrupt signal
PMUEVENTx[24:0]	Output	PMU event bus

A.14 DFT and MBIST interfaces

This section describes:

- *DFT interface*.
- *MBIST interface*.

A.14.1 DFT interface

Table A-32 shows the DFT interface signals.

Table A-32 DFT interface signals

Signal	Type	Description
DFTCLKBYPASS	Input	Bypasses the strobe clock register to the L2 RAMs, forcing the L2 RAMs to be tested using **CLK** as the source clock
DFTCRCLKDISABLE	Input	Disables processor clock grid
DFTCXCLKDISABLE	Input	Disables NEON and VFP clock grid
DFTL2CLKDISABLE	Input	Disables L2 clock grid
DFTRAMHOLD	Input	Disables the RAM chip selects during scan shift
DFTRSTDISABLE	Input	Disables internal synchronized reset during scan shift
DFTSE	Input	Scan shift enable, forces on the clock grids during scan shift

A.14.2 MBIST interface

Table A-33 shows the L1 *Memory Built-In Self Test* (MBIST) interface signals. The value of N is one less than the number of processors in your design.

Table A-33 L1 MBIST interface signals

Signal	Type	Description
nMBISTRESET1	Input	L1 MBIST reset
MBISTACK1[N:0]	Output	L1 MBIST test acknowledge
MBISTADDR1[12:0]	Input	L1 MBIST logical address
MBISTARRAY1[4:0]	Input	L1 MBIST array selector
MBISTBE1[7:0]	Input	L1 MBIST bit write enable
MBISTCFG1	Input	L1 MBIST configuration
MBISTINDATA1[99:0]	Input	L1 MBIST data in
MBISTOUTDATA1[99:0]	Output	L1 MBIST data out
MBISTREADEN1	Input	L1 MBIST read enable
MBISTREQ1[N:0]	Input	L1 MBIST test request
MBISTWRITEEN1	Input	L1 MBIST write enable

Table A-34 shows the L2 MBIST interface signals.

Table A-34 L2 MBIST interface signals

Signals	Type	Description
nMBISTRESET2	Input	L2 MBIST reset
MBISTACK2	Output	L2 MBIST test acknowledge
MBISTADDR2[15:0]	Input	L2 MBIST logical address
MBISTARRAY2[4:0]	Input	L2 MBIST array selector
MBISTBE2[15:0]	Input	L2 MBIST bit write enable
MBISTCFG2[6:0]	Input	L2 MBIST configuration
MBISTINDATA2[127:0]	Input	L2 MBIST data in
MBISTOUTDATA2[127:0]	Output	L2 MBIST data out
MBISTREADEN2	Input	L2 MBIST read enable
MBISTREQ2	Input	L2 MBIST test request
MBISTWRITEEN2	Input	L2 MBIST write enable

Appendix B
Revisions

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

Table B-1 Issue A

Change	Location	Affects
First release	-	-

Table B-2 Differences between issue A and issue B

Change	Location	Affects
Added L2 arbitration register slice as another Cortex-A15 configurable option	Table 1-1 on page 1-7	r0p0
Added a note to indicate that if L2 arbitration register slice is included, an additional pipeline stage is added to the L2 arbitration logic	*Implementation options* on page 1-7	r0p0
Updated the table for valid combinations of L2 tag and data RAM register slice	Table 1-2 on page 1-8	r0p0
Added a new section for event communication using `WFE` and `SEV` instructions	*Event communication using WFE and SEV instructions* on page 2-37	r0p0
Updated the reset value of the Main ID Register	• Table 4-2 on page 4-4 • Table 4-16 on page 4-14	r1p0
Updated bits[23:20] of the Main ID Register	*Main ID Register* on page 4-27	r1p0

Table B-2 Differences between issue A and issue B (continued)

Change	Location	Affects
Added bit[10] Disable Non-secure debug array read to the L2 Auxiliary Control Register	*L2 Auxiliary Control Register* on page 4-100	r0p0
Updated description of Non-cacheable streaming enhancement	*Non-cacheable streaming enhancement* on page 6-9	r0p0
Updated ID number of *Software Generated Interrupt* (SGI)	*Interrupt sources* on page 8-4	r0p0
Added event number `0x62` and `0x63` to the PMU event table	Table 11-7 on page 11-15	r1p0
Updated description for bits[3:0] of the ETMIDR	*ETM ID Register* on page 12-20	r1p0
Updated the value for Peripheral ID2 Register	• Table 10-24 on page 10-32 • Table 11-5 on page 11-12 • Table 12-22 on page 12-30.	r1p0
Updated trigger input name of PMU generated interrupt	Table 13-1 on page 13-3	r0p0
Clarified the step instructions for using the Advanced SIMD and VFP in Secure state	*Using the Advanced SIMD and VFP in Secure state only* on page 14-5	r0p0
Clarified the step instructions for using the Advanced SIMD and VFP in Secure state and Non-secure state	*Using the Advanced SIMD and VFP in Secure state and Non-secure state other than Hyp mode* on page 14-5	r0p0
Clarified the step instructions for using the Advanced SIMD and VFP in Hyp mode	*Using the Advanced SIMD and VFP in Hyp mode* on page 14-5	r0p0
Clarified description of the GIC signals	Table A-4 on page A-6	r0p0
Clarified description of the **EVENTI** and **EVENTO** signals	Table A-6 on page A-9	r0p0
Updated and moved **SYNCREQx** from Miscellaneous PTM interface signals table to ATB interface signals table	Table A-28 on page A-23	r0p0

Table B-3 Differences between issue B and issue C

Change	Location	Affects
Updated description for L2 wait for interrupt	*L2 Wait for Interrupt* on page 2-23	r2p0
Updated the reset value of the Main ID Register	• Table 4-2 on page 4-4 • Table 4-16 on page 4-14	r2p0
Corrected the reset value of the ID_PFR0 Register	• Table 4-2 on page 4-4 • Table 4-16 on page 4-14	All revisions
Updated bits[23:20] of the Main ID Register	*Main ID Register* on page 4-27	r2p0
Clarified description of bits[11:8] of the ID_PFR0 Register	*Processor Feature Register 0* on page 4-31	All revisions
Updated L2ACTLR bit[5] of the L2 Auxiliary Control Register	*L2 Auxiliary Control Register* on page 4-100	r2p0
Updated description of GIC memory-map	*GIC memory-map* on page 8-3	r2p0
Updated the value for Peripheral ID2 Register	• Table 10-24 on page 10-32 • Table 11-5 on page 11-12 • Table 12-22 on page 12-30	r2p0

Table B-3 Differences between issue B and issue C (continued)

Change	Location	Affects
Renamed PMCCFILTR to PMXEVTYPER31 in the PMU register summary table	Table 11-1 on page 11-4	r2p0
Updated description for bits[3:0] of the ETMIDR	*ETM ID Register* on page 12-20	r2p0
Updated the GIC configuration section	*GIC configuration* on page 8-6	r2p0
Updated description of the **nVIRQ** and **nVFIQ** input pins	Table A-4 on page A-6	r2p0

Table B-4 Differences between issue C and issue D

Change	Location	Affects
Updated the reset value of the Main ID Register	• Table 4-2 on page 4-4 • Table 4-16 on page 4-14	r2p1
Added the reset values for MAIR0 and MAIR1	Table 4-11 on page 4-11	r2p1
Updated bits[23:20] of the Main ID Register	*Main ID Register* on page 4-27	r2p1
Clarified description for bits[11:10] and bit[12] of the L2 Control Register	*L2 Control Register* on page 4-85	r2p1
Added a new section for cache maintenance transactions in the Level 2 Memory System chapter	*Cache maintenance transactions* on page 7-14	r2p1
Clarified that some memories within the L2 memory system support ECC when configured	*Error Correction Code* on page 7-4	r2p1
Added a section in the L2 memory system for asynchronous errors	*Asynchronous errors* on page 7-11	r2p1
Updated the DBGDIDR bit assignments	Table 10-2 on page 10-10	r2p1
Updated the field name for bits[63:40] of the Debug Self Address Offset Register	Table 10-16 on page 10-26	r2p1
Clarified description for debug registers	Chapter 10 *Debug*	r2p1
Updated **ARUSERS[5:0]** for the read address channel	Table A-23 on page A-19	r2p1

Table B-5 Differences between issue D and issue E

Change	Location	Affects
Added regional clock gates and processor clock stop pins as additional configuration options	• Table 1-1 on page 1-7 • *Implementation options* on page 1-7	r3p0
Added description for **CPUCLKOFF** inputs	*Clocks* on page 2-8	r3p0
Added description for regional clock gating	*Regional clock gating* on page 2-28	r3p0
Updated the L2 Wait For Interrupt timing diagram to show that **CLKEN** is asserted before the Internal L2 clock signal	Figure 2-13 on page 2-24	All revisions
Added new section for processor retention in WFI and WFE low-power state	*Processor retention in WFI and WFE low-power state* on page 2-24	r3p0

Table B-5 Differences between issue D and issue E (continued)

Change	Location	Affects
Updated the sequence for powering down the processor and the NEON and VFP power domains	*Processor power domain* on page 2-30	r3p0
Added the processor powerdown mode as one of the power sequences that ARM recommends	*Processor powerdown mode* on page 2-36	r3p0
Adding the deassertion of **ACINACTM** or **AINACTS** as conditions for exiting L2 WFI low-power state	*L2 Wait for Interrupt* on page 2-23	All revisions
Updated the reset value of the Main ID Register	• Table 4-2 on page 4-4 • Table 4-16 on page 4-14	r3p0
Updated bits[23:20] of the Main ID Register	*Main ID Register* on page 4-27	r3p0
Added Auxiliary Control Register 2	• Table 4-1 on page 4-13 • Table 4-28 on page 4-25 • *Auxiliary Control Register 2* on page 4-105	r3p0
Modified the L2 Auxiliary Control Register	*L2 Auxiliary Control Register* on page 4-100	r3p0
Removed the footnote and updated the value for Peripheral ID2 Register	• Table 10-24 on page 10-32 • Table 11-5 on page 11-12 • Table 12-22 on page 12-30	All revisions
Updated the Performance Monitor Common Event Register 0 bit assignment table to indicate which event is implemented	Table 11-4 on page 11-11	All revisions
Corrected the reset value for bit[10] of the ETMCR	Table 12-5 on page 12-14	All revisions
Updated bit[0] of the ETMCR	Table 12-5 on page 12-14	All revisions
Updated bits[3:0] of the ETMIDR	Table 12-10 on page 12-20	All revisions
Updated the clock and clock enable signals table	Table A-1 on page A-3	r3p0
Updated the write address channel signals table	Table A-10 on page A-14	r3p0
Updated the power management signals table	Table A-7 on page A-10	r3p0

Table B-6 Differences between issue E and issue F

Change	Location	Affects
Updated the reset value of the Main ID Register	• Table 4-2 on page 4-4 • Table 4-16 on page 4-14	r3p1
Updated bits[23:20] of the Main ID Register	*Main ID Register* on page 4-27	r3p1
Updated information about external errors associated with a load instruction and with cache maintenance operations.	*Asynchronous errors* on page 7-11	All revisions

Table B-7 Differences between issue F and issue G

Change	Location	Affects
Clarified description for **CLKEN**	*Clocks* on page 2-8	r3p2
Updated the reset sequence for a full soft reset on the processor	*Soft reset* on page 2-17	r3p2
Updated information on L2 Wait for Interrupt	*L2 Wait for Interrupt* on page 2-23	r3p2
Updated the reset value of the Main ID Register	• Table 4-2 on page 4-4 • Table 4-16 on page 4-14	r3p2
Updated bits[23:20] of the Main ID Register	*Main ID Register* on page 4-27	r3p2
Corrected description for bits[26:25] of the Auxiliary Control Register	*Auxiliary Control Register* on page 4-57	All revisions
Updated text for the MAIR0, MAIR1, AMAIR0, AMAIR1, HAMAIR0, and HAMAIR1 registers	• *Memory Attribute Indirection Register 0* on page 4-88 • *Memory Attribute Indirection Register 1* on page 4-89 • *Auxiliary Memory Attribute Indirection Register 0* on page 4-89 • *Auxiliary Memory Attribute Indirection Register 1* on page 4-89 • *Hyp Auxiliary Memory Attribute Indirection Register 0* on page 4-89 • *Hyp Auxiliary Memory Attribute Indirection Register 1* on page 4-89	All revisions
Clarified information about intermediate table walk caches	*Intermediate table walk caches* on page 5-8	All revisions
Updated description for instruction cache speculative memory accesses	*Instruction cache speculative memory accesses* on page 6-4	r3p2
Clarified information about external memory errors	*Asynchronous errors* on page 7-11	All revisions
Added Debug Device ID Register 1	*Debug Device ID Register 1* on page 10-30	r3p2
Added event mnemonic column to the PMU events table	Table 11-7 on page 11-15	All revisions

Table B-8 Differences between issue G and issue H

Change	Location	Affects
Updated description for the **CPUCLKOFF** pins	*Clocks* on page 2-8	All revisions
Updated description in the L2 Wait for Interrupt section	*L2 Wait for Interrupt* on page 2-23	All revisions
Updated the NEON and VFP power domain section	*NEON and VFP power domain* on page 2-32	All revisions
Updated the sequence when the Cortex-A15 MPCore processor enters Dormant mode	*Dormant mode* on page 2-35	All revisions
Combined the ThumbEE architecture section with the Jazelle Extension section from Chapter 3 *Programmers Model* into one section called Execution environment support	*Execution environment support* on page 3-3	All revisions
Updated the reset value of the Main ID Register	• Table 4-2 on page 4-4 • Table 4-16 on page 4-14	r3p3

Table B-8 Differences between issue G and issue H (continued)

Change	Location	Affects
Added clarification to the description of the PoU and PoC footnotes	Table 4-20 on page 4-17	All revisions
Updated bits[23:20] of the Main ID Register	*Main ID Register* on page 4-27	r3p3
Updated description for the Auxiliary Control Register	*Auxiliary Control Register* on page 4-57	All revisions
Updated description for the Coprocessor Access Control Register	*Coprocessor Access Control Register* on page 4-62	All revisions
Clarified description for the Non-Secure Access Control Register	*Non-Secure Access Control Register* on page 4-65	All revisions
Updated description for bit[11] TDRA of the Hyp Debug Configuration Register	*Hyp Debug Configuration Register* on page 4-69	All revisions
Updated description for bit[9] TDA of the Hyp Debug Configuration Register	*Hyp Debug Configuration Register* on page 4-69	All revisions
Added a note in the MMU chapter to state that the Cortex-A15 MPCore processor does not support the Transient attribute in the *Large Physical Address Extensions* (LPAE)	*About the MMU* on page 5-2	All revisions
The processor no longer supports the use of the Prefetch bit in the L2 cache replacement algorithm. Bit[9] of the L2ACTLR is removed.	*L2 Auxiliary Control Register* on page 4-100	r2p2, r3p0, r3p1, and r3p2
Added a note to the L2 Prefetch Control Register description to indicate that setting a value of `0x400` to this register disables all the hardware prefetch	*L2 Prefetch Control Register* on page 4-103	All revisions
Update description for the Write-Through memory type	*Write-Through* on page 6-8	All revisions
Update description for the Non-Cacheable memory type	*Non-Cacheable* on page 6-8	All revisions
Added a note to indicate that any valid requests issued to the Cortex-A15 MPCore processor on the AC channel when **BROADCASTINNER** and **BROADCASTOUTER** are both deasserted, cause UNPREDICTABLE result	Table 7-5 on page 7-16	All revisions
Updated description for the GICD_IPRIORITYRn register to indicate that changing the value of a priority field in the GICD_IPRIORITYRn register changes the priority of an active interrupt.	Table 8-3 on page 8-8	All revisions
Added a footnote to the description of the GICD_ITARGETSR registers to state that no SPIs are statically configured in hardware.	Table 8-3 on page 8-8	All revisions
Added a footnote to the description of the GICD_SGIR register to indicate that this register has no effect when the GICD_CTLR bit settings disable the Distributor.	Table 8-3 on page 8-8	All revisions
Clarified definition of the OS Lock Access Register	*OS Lock Access Register* on page 10-22	All revisions
Updated description for bit[1] Core Warm Reset Request bit of the DBGPRCR	*Device Powerdown and Reset Control Register* on page 10-24	All revisions

Table B-8 Differences between issue G and issue H (continued)

Change	Location	Affects
Corrected description for bits[3:0] PCSROffset of the Debug Device ID Register 1	*Debug Device ID Register 1* on page 10-30	All revisions
Added a footnote to the PMU events table to indicate that event count is encoded as plain binary number to accomodate count values of more than one.	Table 11-7 on page 11-15	All revisions
Updated the configurations information for all the Advanced SIMD and VFP system registers	Chapter 14 *NEON and VFP Unit*	All revisions

Table B-9 Differences between issue H and issue I

Change	Location	Affects
Updated the reset value of the Main ID Register	• Table 4-2 on page 4-4 • Table 4-16 on page 4-14	r4p0
Updated bits[23:20] of the Main ID Register	*Main ID Register* on page 4-27	r4p0
Added bit[25] to the L2 Auxiliary Control Register	*L2 Auxiliary Control Register* on page 4-100	r4p0
Added a new L2 memory interface attribute for snoop acceptance capability	Table 7-4 on page 7-12	r4p0
Added a new table for the ACP L2 memory interface attribute	Table 7-7 on page 7-18	r4p0
Moved the step for asserting **ACINACTM** to after the step for cleaning and invalidating all data from the L2 data cache	*Processor powerdown mode* on page 2-36	r4p0
Added a step for setting the DBGOSDLR.DLK to force the debug interfaces to be quiescent	*Dormant mode* on page 2-35	r4p0
Added L1 data TLB support for 1MB page table entries	• Table 1-1 on page 1-7 • *L1 data TLB* on page 5-3	r4p0
Added a sentence to indicate that the GIC does not support a 1:1 clock frequency ratio	• *Clocks* on page 2-8 • *GIC clock frequency* on page 8-3	All revisions