

# ARM® Cortex®-A17 MPCore Processor

Revision: r1p1

## Technical Reference Manual



# ARM Cortex-A17 MPCore Processor

## Technical Reference Manual

Copyright © 2014 ARM. All rights reserved.

### Release Information

The following changes have been made to this book.

Change history			
Date	Issue	Confidentiality	Change
11 March 2014	A	Confidential	First release for r0p0
28 June 2014	B	Non-Confidential	First release for r1p0
18 September 2014	C	Non-Confidential	First release for r1p1

### 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-A17 MPCore Processor Technical Reference Manual

	<b>Preface</b>	
	About this book .....	vii
	Feedback .....	xi
<b>Chapter 1</b>	<b>Introduction</b>	
	1.1 About the Cortex-A17 MPCore processor .....	1-2
	1.2 Compliance .....	1-3
	1.3 Features .....	1-5
	1.4 Interfaces .....	1-6
	1.5 Implementation options .....	1-7
	1.6 Product documentation and design flow .....	1-8
	1.7 Product revisions .....	1-10
<b>Chapter 2</b>	<b>Functional Description</b>	
	2.1 About the Cortex-A17 MPCore processor functions .....	2-2
	2.2 Interfaces .....	2-6
	2.3 Clocking and resets .....	2-8
	2.4 Power management .....	2-17
<b>Chapter 3</b>	<b>Programmers Model</b>	
	3.1 About the programmers model .....	3-2
	3.2 ThumbEE instruction set .....	3-3
	3.3 Jazelle Architecture Extension .....	3-4
	3.4 Advanced SIMD and VFP Extensions .....	3-6
	3.5 Security Extensions architecture .....	3-7
	3.6 Virtualization Extensions architecture .....	3-8

	3.7	Large Physical Address Extension architecture .....	3-9
	3.8	Multiprocessing Extensions .....	3-10
	3.9	Operating states .....	3-11
	3.10	Memory model .....	3-12
<b>Chapter 4</b>		<b>System Control</b>	
	4.1	About system control .....	4-2
	4.2	Register summary .....	4-3
	4.3	Register descriptions .....	4-27
<b>Chapter 5</b>		<b>Memory Management Unit</b>	
	5.1	About the MMU .....	5-2
	5.2	TLB organization .....	5-3
	5.3	TLB match process .....	5-4
	5.4	Memory access sequence .....	5-5
	5.5	MMU enabling and disabling .....	5-7
	5.6	Intermediate table walk cache .....	5-8
	5.7	Memory types .....	5-10
	5.8	Memory region attributes .....	5-11
<b>Chapter 6</b>		<b>L1 Memory System</b>	
	6.1	About the L1 memory system .....	6-2
	6.2	Cache features .....	6-3
	6.3	L1 instruction memory system .....	6-4
	6.4	L1 data memory system .....	6-6
	6.5	Data prefetching .....	6-9
	6.6	Direct access to internal memory .....	6-10
<b>Chapter 7</b>		<b>L2 Memory System</b>	
	7.1	About the L2 Memory system .....	7-2
	7.2	L2 cache .....	7-3
	7.3	Cache coherency .....	7-6
	7.4	ACE master interface .....	7-7
	7.5	ACP .....	7-12
	7.6	Peripheral port .....	7-14
	7.7	External aborts and asynchronous errors .....	7-15
<b>Chapter 8</b>		<b>Generic Timer</b>	
	8.1	About the Generic Timer .....	8-2
	8.2	Generic Timer functional description .....	8-3
	8.3	Timer programmers model .....	8-4
<b>Chapter 9</b>		<b>Debug</b>	
	9.1	About debug .....	9-2
	9.2	Debug register interfaces .....	9-4
	9.3	Debug register summary .....	9-6
	9.4	Debug register descriptions .....	9-10
	9.5	Debug events .....	9-30
	9.6	External debug interface .....	9-31
<b>Chapter 10</b>		<b>Performance Monitoring Unit</b>	
	10.1	About the Performance Monitoring Unit .....	10-2
	10.2	PMU functional description .....	10-3
	10.3	PMU registers summary .....	10-4
	10.4	PMU register descriptions .....	10-7
	10.5	Events .....	10-10
	10.6	Interrupts .....	10-14
	10.7	Exporting PMU events .....	10-15

<b>Chapter 11</b>	<b>Program Trace Macrocell</b>	
11.1	About PTM .....	11-2
11.2	PTM options .....	11-3
11.3	PTM functional description .....	11-4
11.4	Reset .....	11-6
11.5	PTM programmers model .....	11-7
11.6	Register summary .....	11-11
11.7	Register descriptions .....	11-15
<b>Chapter 12</b>	<b>Cross Trigger</b>	
12.1	About the cross trigger .....	12-2
12.2	Trigger inputs and outputs .....	12-3
12.3	Cortex-A17 CTI .....	12-4
12.4	Cortex-A17 CTM .....	12-5
<b>Chapter 13</b>	<b>NEON and Floating-point Unit</b>	
13.1	About NEON and floating-point unit .....	13-2
13.2	Programmers model for NEON and floating-point unit .....	13-3
<b>Appendix A</b>	<b>Signal Descriptions</b>	
A.1	About the signal descriptions .....	A-2
A.2	Clock and reset signals .....	A-3
A.3	Configuration signals .....	A-4
A.4	Interrupt signals .....	A-5
A.5	Asynchronous error signals .....	A-6
A.6	Generic Timer signals .....	A-7
A.7	Power control signals .....	A-8
A.8	ACE master interface signals .....	A-9
A.9	Peripheral port AXI master interface signals .....	A-14
A.10	ACP slave interface signals .....	A-17
A.11	External debug interface .....	A-20
A.12	Cross trigger channel interface .....	A-24
A.13	DFT interface signals .....	A-25
A.14	MBIST interface signals .....	A-26
<b>Appendix B</b>	<b>Revisions</b>	

# Preface

This preface introduces the *ARM® Cortex®-A17 MPCore Processor Technical Reference Manual*. It contains the following sections:

- [About this book on page vii.](#)
- [Feedback on page xi.](#)

## About this book

This book is for the Cortex-A17 MPCore processor. This is a multiprocessor device that has between one to four processors.

## Product revision status

The *rn**pn* identifier indicates the revision status of the product described in this book, where:

- rn** Identifies the major revision of the product.
- pn** Identifies the minor revision or modification status of the product.

## Intended audience

This book is written for hardware and software engineers implementing the Cortex-A17 MPCore processor in system designs. It provides information that enables designers to integrate the Cortex-A17 MPCore processor into a target system.

## Using this book

This book is organized into the following chapters:

### Chapter 1 *Introduction*

Read this for an introduction to the Cortex-A17 MPCore processor and descriptions of the major features.

### Chapter 2 *Functional Description*

Read this for a description of the functionality of the Cortex-A17 MPCore 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, their structure, operation, and how to use them.

### Chapter 5 *Memory Management Unit*

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

### Chapter 6 *L1 Memory System*

Read this for a description of the *Level 1* (L1) memory system, including caches, *Translation Lookaside Buffers* (TLB), and store buffer.

### Chapter 7 *L2 Memory System*

Read this for a description of the *Level 2* (L2) memory system, including the *Snoop Control Unit* (SCU) and the ACE master interface.

### Chapter 8 *Generic Timer*

Read this for a description of the timers.

### Chapter 9 *Debug*

Read this for a description of the support for debug.

**Chapter 10 *Performance Monitoring Unit***

Read this for a description of the *Performance Monitoring Unit* (PMU) and associated events.

**Chapter 11 *Program Trace Macrocell***

Read this for a description of the *Program Trace Macrocell* (PTM).

**Chapter 12 *Cross Trigger***

Read this for a description of the *Cross Trigger* (CT) interfaces.

**Chapter 13 *NEON and Floating-point Unit***

Read this for a description of the NEON and floating-point unit.

**Appendix A *Signal Descriptions***

Read this for a description of the input and output signals.

**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:

Typographical conventions	
Style	Purpose
<i>italic</i>	Introduces special terminology, denotes cross-references, and citations.
<b>bold</b>	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.
<u>monospace</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
monospace <i>italic</i>	Denotes arguments to monospace text where the argument is to be replaced by a specific value.



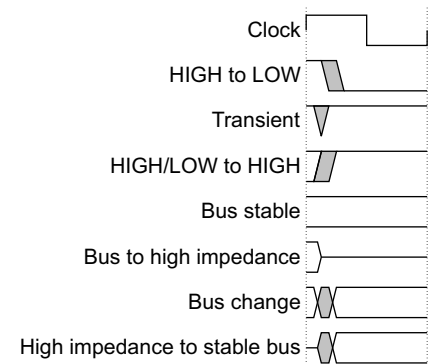
## Typographical conventions (continued)

Style	Purpose
<b>monospace bold</b>	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 <i>ARM Glossary</i> . For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

## Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

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



## 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:

<b>Signal level</b>	The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means: <ul style="list-style-type: none"> <li>HIGH for active-HIGH signals.</li> <li>LOW for active-LOW signals.</li> </ul>
<b>Lowercase n</b>	At the start or end of a signal name denotes an active-LOW signal.

## Additional reading

This section lists publications by ARM and 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® 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® 4 ATB Protocol Specification* (ARM IHI 0032).
- *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* (ARM DDI 0406).
- *ARM® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2* (ARM IHI 0048).
- *ARM® CoreSight™ Architecture Specification* (ARM IHI 0029).
- *ARM® CoreSight™ Program Flow Trace Architecture Specification* (ARM IHI 0035).
- *ARM® CoreSight™ SoC Technical Reference Manual* (ARM DDI 0480).
- *ARM® Cortex®-A Series Programmer's Guide* (ARM DEN0013).

The following confidential books are only available to licensees:

- *ARM® Cortex®-A17 MPCore Processor Configuration and Sign-off Guide* (ARM DII 0298).
- *ARM® Cortex®-A17 MPCore Processor Integration Manual* (ARM DIT 0060).

### Other publications

This section lists relevant documents published by third parties:

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

## Feedback

ARM welcomes feedback on this product and its documentation.

### Feedback on this product

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

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

### Feedback on content

If you have comments on content then send an e-mail to [errata@arm.com](mailto:errata@arm.com). Give:

- The title.
- The number, ARM DDI 0535C.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

———— **Note** —————

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

---

# Chapter 1

## Introduction

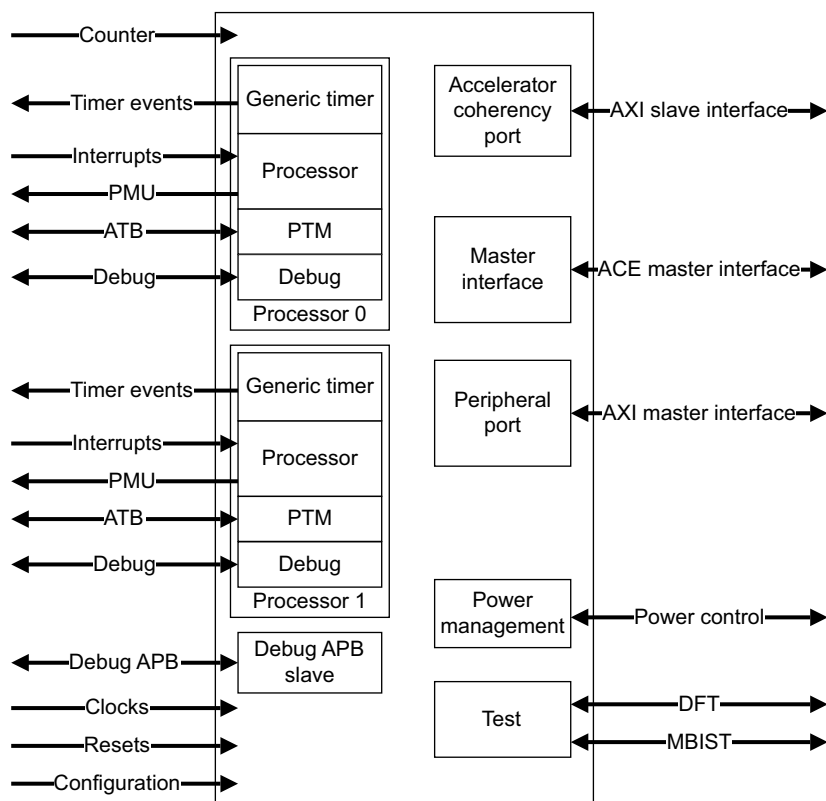
This chapter introduces the Cortex-A17 MPCore processor and its features. It contains the following sections:

- *About the Cortex-A17 MPCore processor on page 1-2.*
- *Compliance on page 1-3.*
- *Features on page 1-5.*
- *Interfaces on page 1-6.*
- *Implementation options on page 1-7.*
- *Product documentation and design flow on page 1-8.*
- *Product revisions on page 1-10.*

## 1.1 About the Cortex-A17 MPCore processor

The Cortex-A17 MPCore processor is a high-performance, low-power processor that implements the ARMv7 architecture. The Cortex-A17 MPCore processor has one to four processors in a single multiprocessor device with L1 and L2 cache subsystems.

Figure 1-1 shows an example of a Cortex-A17 MPCore processor implementation with two processors.



**Figure 1-1 Multiprocessor block diagram**

When a Cortex-A17 MPCore processor is implemented with a single processor, it still includes the *Snoop Control Unit* (SCU). See [Implementation options on page 1-7](#) for more information.

See [About the Cortex-A17 MPCore processor functions on page 2-2](#) for more information about the functional components.

## 1.2 Compliance

The Cortex-A17 MPCore processor complies with, or implements, the specifications described in:

- [ARM architecture](#).
- [Advanced Microcontroller Bus Architectures](#).
- [Debug architecture](#).
- [Generic Timer architecture on page 1-4](#).

This *Technical Reference Manual* (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-A17 MPCore processor implements the ARMv7-A profile 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 enable the switching of guest operating systems.
- *Large Physical Address Extension* (LPAE) for address translation of up to 40-bit physical addresses.
- Multiprocessing Extensions for multiprocessing functionality.

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

### 1.2.2 Advanced Microcontroller Bus Architectures

The Cortex-A17 MPCore processor ACE and debug interfaces comply with the:

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

### 1.2.3 Debug architecture

The Cortex-A17 MPCore processor implements the ARMv7.1 Debug architecture that includes support for CoreSight. For more information, see the:

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

## 1.2.4 Generic Timer architecture

The Cortex-A17 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*.

## 1.3 Features

The processor includes the following features:

- Full implementation of the ARMv7 architecture instruction set with the architecture extensions listed in [Compliance on page 1-3](#).
- Branch prediction:
  - Dynamic and static branch prediction.
  - Branch target address cache.
  - Nested return stack.
- Short loop instruction buffering.
- Two way superscalar instruction decode.
- Register renaming.
- Out-of-order instruction issue.
- Stride-based prefetcher.
- Harvard *Level 1* (L1) memory system.
- Non blocking MMU.
- A fully associative, instruction micro *Translation Lookaside Buffer* (TLB). The number of entries are an implementation option.
- A 32-entry, fully associative, data micro TLB.
- A 1024-entry, 4-way, unified main TLB with hit-under-miss capability.
- Automatic cache coherency between L1 data caches within the cluster.
- *Level 2* (L2) memory system.
- Single-bit correction and double-bit detection of errors in the L2 cache. This is an implementation option.
- NEON and floating-point unit, supporting the Advanced SIMDv2 and VFPv4-D32 extensions.
- Optionally implemented, *Memory Built-In Self Test* (MBIST) for production test of the processor RAMs.



## 1.4 Interfaces

The Cortex-A17 MPCore processor has the following external interfaces:

- ACE master interface.
- *Accelerator Coherency Port* (ACP) AXI slave interface.
- Peripheral port AXI master interface.
- Debug APB slave interface.
- *Design For Test* (DFT).
- Three MBIST interfaces.

Each processor in the multiprocessor device has the following external interfaces:

- Interrupt interface.
- Generic timer interface.
- *Performance Monitoring Unit* (PMU) interface.
- *Program Trace Macrocell* (PTM) *AMBA Trace Bus* (ATB) interface.
- Processor debug interface.

See [Interfaces on page 2-6](#) for more information on these interfaces.

## 1.5 Implementation options

Table 1-1 shows the Cortex-A17 MPCore processor RTL implementation options.

**Table 1-1 Implementation options for the Cortex-A17 MPCore processor RTL**

Feature	Range of options
<b>Global implementation options:</b>	
Number of processors	1, 2, 3, or 4
L2 cache sizes	256KB, 512KB, 1MB, 2MB, 4MB or 8MB
L2 ECC	Included or not
ACP interface	Included or not
Peripheral port	Included or not
MBIST interfaces	Included or not
Include reset repeaters	Included or not
L2 cache tag RAM setup, read, and write latencies	0 to 7
L2 cache data RAM setup, read, and write latencies	0 to 7
Include register slice for L2 cache read data	Included or not
Include SCU DDI register slice for each processor	Included or not
Include SCU DDI register slice for ACP	Included or not
Include SCU DDI register slice for ACE	Included or not
<b>Processor-level implementation options:<sup>a</sup></b>	
NEON and VFP support	Included or not
Instruction cache size	32KB or 64KB
BTAC	2K or 4K entries
Instruction micro TLB size	32, 48, or 64 entries

a. All processors in the Cortex-A17 MPCore processor are implemented with the same options.

## 1.6 Product documentation and design flow

This section describes the Cortex-A17 MPCore processor books, how they relate to the design flow, and the relevant architectural standards and protocols.

See [Additional reading on page x](#) for more information about the books described in this section.

### 1.6.1 Documentation

The Cortex-A17 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-A17 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-A17 MPCore processor then contact:

- The implementer to determine the build configuration of the implementation.
- The integrator to determine the pin configuration of the *System-on-Chip* (SoC) that you are using.

#### Configuration and Sign-off Guide

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

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

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

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

#### Integration Manual

The *Integration Manual* (IM) describes how to integrate the Cortex-A17 MPCore processor macrocell into a SoC. It includes a description of the pins that the integrator must tie off to configure the macrocell. Some of the integration tasks are affected by the configuration options used when implementing the Cortex-A17 MPCore processor.

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

### 1.6.2 Design flow

The Cortex-A17 MPCore processor is delivered as synthesizable Verilog RTL. Before it 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 includes integrating the RAMs into the design.

<b>Integration</b>	The integrator connects the macrocell into a SoC. This includes connecting it to a memory system and peripherals.
<b>Programming</b>	The system programmer develops the software required to configure and initialize the Cortex-A17 MPCore processor, and tests the required application software.

Each stage of the process:

- Can be performed by a different party.
- Can include implementation and integration choices that affect the behavior and features of the Cortex-A17 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. They usually include or exclude logic that can affect the area or maximum frequency of the resulting macrocell.

#### **Configuration inputs**

The integrator configures some features of the macrocell 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-A17 MPCore processor by programming particular values into software-visible registers. This affects the behavior of the Cortex-A17 MPCore processor.

---

#### **Note**

---

This manual refers to implementation-defined features that are applicable to build configuration options. References to a feature that is *included* means that the appropriate build and pin configuration options have been selected, while references to an *enabled* feature means one that has also been configured by software.

---

## 1.7 Product revisions

This section describes the differences in functionality between product revisions:

### 1.7.1 r0p0 - r1p0

The following changes have been made in this release:

- ID register value changed to reflect product revision status:

**Main ID Register**

0x411FC0E0

**Debug ID Register**

0x3515F010

- Added register bit ACTLR.ASSE.
- Added register bit L2CTLR.SFEN.
- Various engineering errata fixes.

### 1.7.2 r1p0 - r1p1

The following changes have been made in this release:

- ID register value changed to reflect product revision status:

**Main ID Register**

0x411FC0E1

**Debug ID Register**

0x3515F011

- Various engineering errata fixes.

# Chapter 2

## Functional Description

This chapter describes the functionality of the Cortex-A17 MPCore processor. It contains the following sections:

- *About the Cortex-A17 MPCore processor functions on page 2-2.*
- *Interfaces on page 2-6.*
- *Clocking and resets on page 2-8.*
- *Power management on page 2-17.*

## 2.1 About the Cortex-A17 MPCore processor functions

Figure 2-1 shows a top-level functional diagram of the Cortex-A17 MPCore processor.

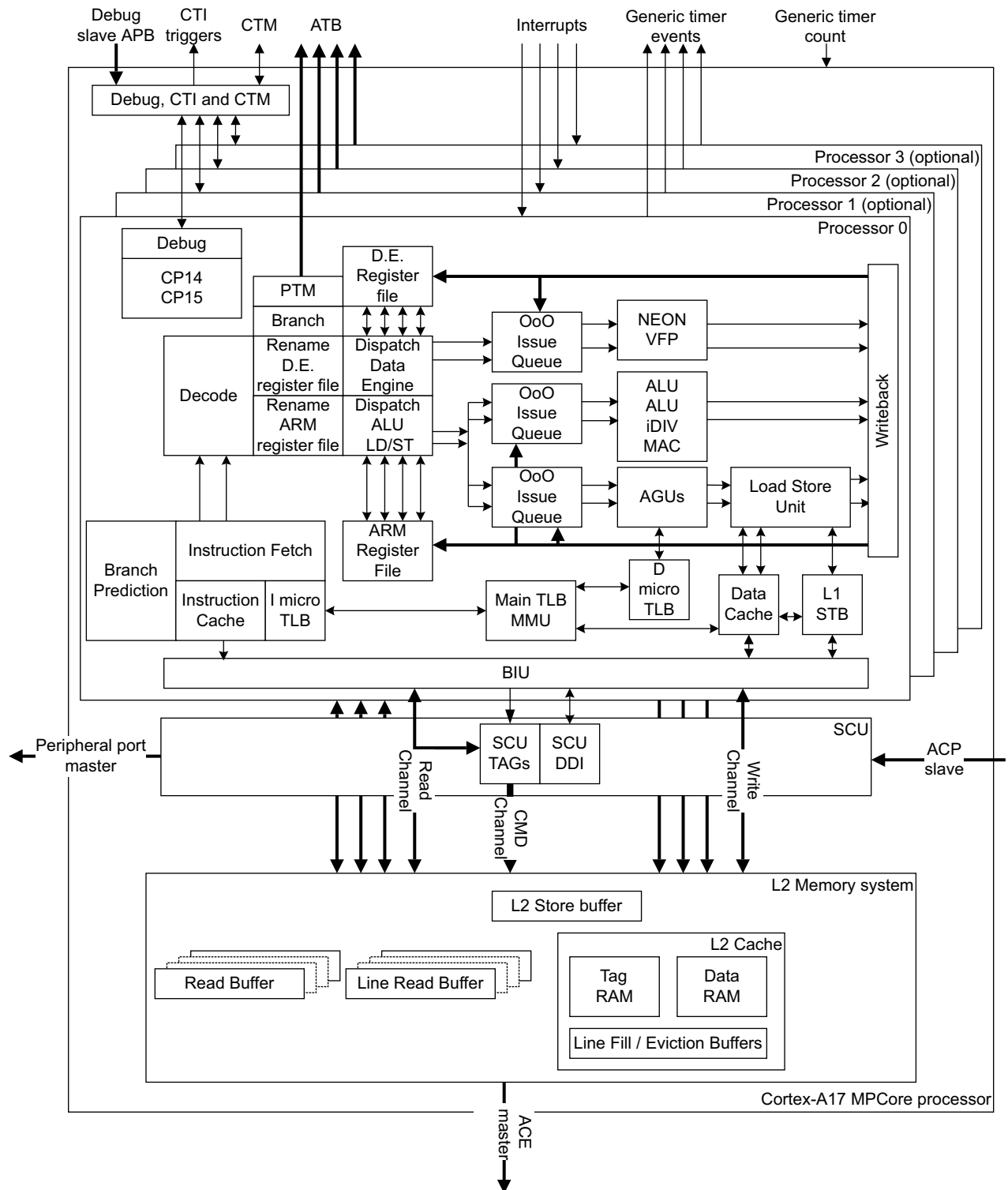


Figure 2-1 Cortex-A17 MPCore processor top-level diagram

### 2.1.1 Processor components

The following sections describe the main components and their functions:

- [MMU](#).
- [Instruction side memory system](#).
- [Data processing unit](#).
- [Data side memory system on page 2-5](#).
- [Debug on page 2-5](#).
- [Performance monitoring on page 2-5](#).

#### MMU

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

The results of a translation table walk are cached in *Translation Lookaside Buffers* (TLBs). The Cortex-A17 MPCore processor has three TLBs:

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

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

See [Chapter 5 Memory Management Unit](#).

#### Instruction side memory system

The instruction side (I-side) fetches multiple instructions from memory and provides an instruction stream with up to two instructions per cycle to the Instruction decoder. The I-side performs static and dynamic branch prediction. Dynamic prediction uses a two-level global history buffer and a branch target address cache. A nested return stack speeds up function returns.

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

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

The instruction cache is part of the L1 Memory system.

See [Chapter 6 L1 Memory System](#).

#### Data processing unit

The *Data Processing Unit* (DPU) holds most of the program-visible state of the processor, such as general-purpose registers, status registers and control registers. It decodes and executes instructions, operating on data held in the registers in accordance with the ARM Architecture. Instructions are fed to the DPU from the I-side. The DPU executes instructions that require data to be transferred to or from the memory system by interfacing to the D-side, which manages all load and store operations.



### **Instruction decode**

The instruction decoder processes the instructions presented by the I-side. ARM and Thumb instructions are decoded in one cycle, NEON and floating-point unit instructions are decoded in up to two cycles. The instruction decoder can provide up to two instructions per cycle to the renaming stage.

### **Register renaming**

Register renaming maps the register names, as specified in the instruction coding, to a larger set of registers. This allows the order of execution of instructions to be based on genuine data dependencies rather than on the names of registers.

Register renaming is performed independently on ARM core registers (R0-R12, SP, LR, and PC), and NEON and floating-point registers.

### **Dispatch stage**

There are two dispatch stages, one for the *Address Generation Unit* (AGU) and integer execution stages, and one for the NEON and floating-point unit. Each dispatch stage can dispatch up to two instructions that are dispatched to one of three issue queues.

### **Issue Queues**

There are three instruction issue queues:

- Integer execution queue (ALU, MAC and division).
- Load store queue.
- NEON and floating-point unit execution queue.

Each queue issues up to two instructions out of order to the execution stages.

### **Execution stages**

There are three execution stages:

- Integer execution.  
The integer execution stage is fed by the ALU, MAC and division queue. There are four integer stages:
  - Two ALU, shift and saturate.
  - One MAC.
  - One radix-4 divider.
- Address generation.  
Two AGUs perform address calculation for load and store operations. Addresses generated reference the data micro TLB to provide physical addresses for the L1 data cache.
- NEON and floating-point unit.  
The NEON and floating-point unit provides support for the ARMv7 Advanced SIMDv2 and VFPv4 instruction sets. See [Chapter 13 NEON and Floating-point Unit](#) for more information.

**Load/store unit**

The *Load/store Unit* (LSU) handles requests from the AGU. The physical addresses generated by the AGU are used for L1 data cache lookup. Data for loads and stores is transferred as 32-bit words between the LSU and integer execution units, and 64-bit double words to the NEON and floating-point unit. Cacheable and Non-cacheable writes are sent to an 8-entry, 64-bit wide store buffer.

**Data side memory system**

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

The D-side includes a prefetcher, which detects regular patterns of loads or stores and prefetches cache lines.

The D-side is part of the L1 memory system.

See [Chapter 6 L1 Memory System](#).

See [Chapter 7 L2 Memory System](#).

**Store Buffer**

The processor has an 8-entry, 64-bit wide store buffer with merging capability.

The store buffer is written to L1 data cache under the following conditions:

- When an entry is full, having 64 bits of valid data.
- When more than 6 entries are in use.
- When the store buffer has not emptied for more than 128 cycles.
- When a DSB request or a cache maintenance operation is to be executed.
- On a DMB instruction.
- When a load partially hits in the store buffer.

**Debug**

The Cortex-A17 MPCore processor has a CoreSight compliant *Advanced Peripheral Bus version 3* (APBv3) debug interface. This permits system access to debug resources, for example, setting watchpoints and breakpoints. The processor provides extensive support for real-time debug and performance profiling.

See [Chapter 9 Debug](#) for more information.

**Performance monitoring**

The Cortex-A17 MPCore processor provides performance counters and event monitors that can be configured to gather statistics on the operation of the processor and the memory system. See [Chapter 10 Performance Monitoring Unit](#) for more information.

## 2.2 Interfaces

The Cortex-A17 MPCore processor has the following external interfaces:

- [ACE master interface](#).
- [Accelerator Coherency Port AXI slave interface](#).
- [Peripheral port AXI master interface](#).
- [Interrupt inputs](#).
- [Generic timer interface](#).
- [Performance Monitoring Unit \(PMU\) interface](#).
- [Debug APB interface](#).
- [Program Trace Macrocell interface on page 2-7](#).
- [DFT interface on page 2-7](#).
- [MBIST interface on page 2-7](#).

### 2.2.1 ACE master interface

The Cortex-A17 MPCore processor implements an ACE master interface. See [ACE master interface signals on page A-9](#) for more information on the signals.

See the *ARM® AMBA® AXI and ACE Protocol Specification* for more information.

### 2.2.2 Accelerator Coherency Port AXI slave interface

The Cortex-A17 MPCore processor implements an AMBA 3 *Accelerator Coherency Port* (ACP) slave interface that is monitored by the SCU. See [ACP slave interface signals on page A-17](#) for information on the signals.

### 2.2.3 Peripheral port AXI master interface

The Cortex-A17 MPCore processor implements an AXI4 master interface. See [Peripheral port AXI master interface signals on page A-14](#) for information on the signals.

### 2.2.4 Interrupt inputs

See [Interrupt signals on page A-5](#).

### 2.2.5 Generic timer interface

See [Generic timer interface](#).

### 2.2.6 Performance Monitoring Unit (PMU) interface

See [Performance Monitoring Unit \(PMU\) interface](#).

### 2.2.7 Debug APB interface

The Cortex-A17 MPCore processor implements an AMBA 3 APB slave interface that enables access to the debug registers. See the *ARM® CoreSight™ Architecture Specification* for more information. See [APB Interface signals on page A-20](#) for information on the signals.

### 2.2.8 Program Trace Macrocell interface

Each processor includes a PTM interface to send data to external debug and trace hardware. Trace data is written on to the *AMBA Trace Bus* (ATB). The ATB interface is compatible with the CoreSight architecture. See the *ARM® CoreSight™ Program Flow Trace Architecture Specification* for more information.

See [PTM interface signals on page A-22](#) for more information on the signals.

### 2.2.9 DFT interface

The Cortex-A17 MPCore 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 interface signals on page A-25](#) for information on these test signals.

### 2.2.10 MBIST interface

The *Memory Built In Self Test* (MBIST) controller interfaces provide support for manufacturing test of the memories embedded in the Cortex-A17 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 signals on page A-26](#) for information on these interfaces.

## 2.3 Clocking and resets

The following sections describe clocking and resets:

- [Clocks](#).
- [Clock enables](#).
- [Resets on page 2-12](#).

---

### Note

---

[N:0]. N denotes the number of the processor in the Cortex-A17 MPCore.

---

### 2.3.1 Clocks

The Cortex-A17 MPCore processor has the following clock inputs:

#### CLK

Clock to SCU logic, L2 logic, and other top-level logic not clocked by **CORECLK[N:0]**, **PCLKDBG**, or **L2RAMCLK**.

#### CORECLK[N:0]

Dedicated clock to each processor. They must be synchronous to **CLK** and be an integer multiple of the **CLK** period. **CORECLK** period must be no longer than eight **CLK** cycles.

#### PCLKDBG

Clock to the Debug APB slave logic, *Cross Trigger Interface* (CTI) and *Cross Trigger Matrix* (CTM). It must be synchronous to **CLK**.

Because **PCLKDBG** is common to all the processors in the cluster, it must only be enabled when all the **CORECLKs** are enabled. The **CLK** to **PCLKDBG** ratio can be calculated by taking the common multiple of all the **CLK** to **CORECLK** ratios in the cluster. The minimum ratio permitted is the *Lowest Common Multiple* (LCM).

For example, if the **CLK** to **CORECLK** ratios are 1:2, 1:4, and 1:5 then the minimum **CLK** to **PCLKDBG** ratio must be 1:20.

#### L2RAMCLK

Clock to the L2 data RAMs. It must be synchronous to **CLK** and be an integer multiple of **CLK**.

All external signals must be synchronous with reference to **CLK**.

### 2.3.2 Clock enables

Clock enables allow processors and other functional blocks to be clocked at different speeds. They control the frequency relationship between a high rate clock and a lower rate synchronous clock that can be applied to a slower functional block.

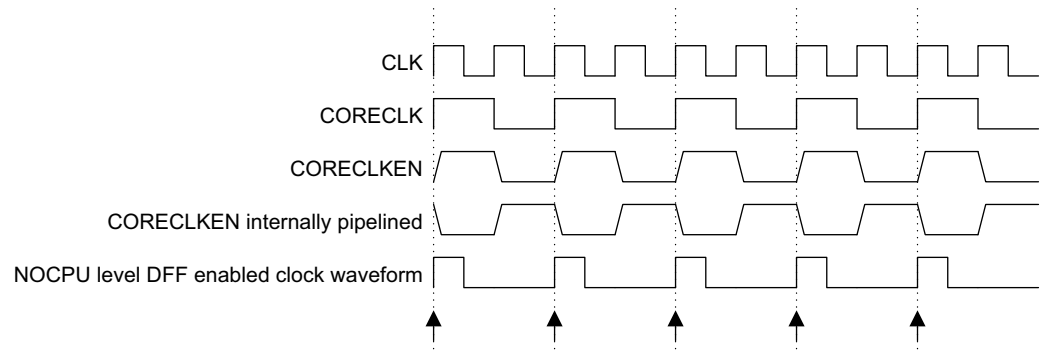
Clock enables must be used for generating continuous clock pulses. They must never be used for stopping a clock. Clocks to individual processors must be maintained during low-power or powerdown states.

All clock enable signals have a single-cycle pipeline delay.

**CORECLKEN[N:0]**

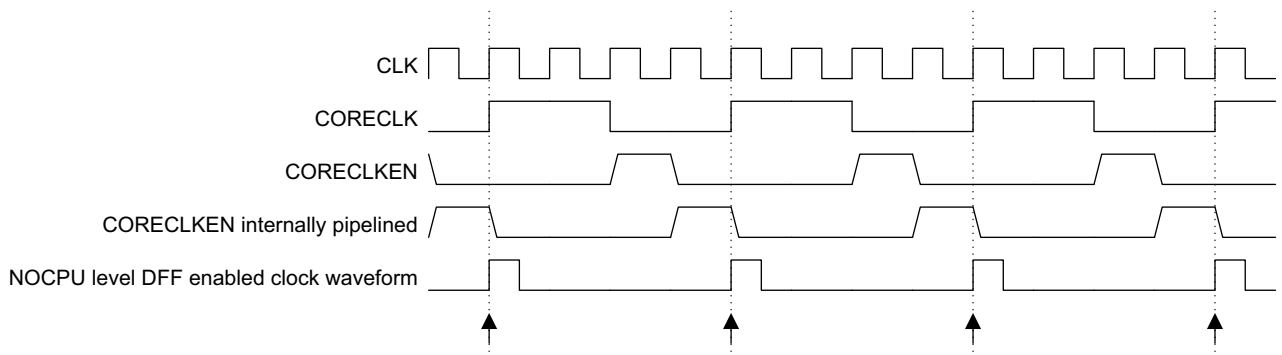
Clock enable to individual processors. Use it for gating **CLK** logic. Do not use it to stop **CORECLK**. To allow data to be correctly sampled using **CORECLK**, **CORECLKEN** must remain active irrespective of the state of **CORECLK** even if **CORECLK** has stopped.

Figure 2-2 shows a 1:2 **CLK** to **CORECLK** relationship.



**Figure 2-2 CLK and CORECLK 1:2 relationship**

Figure 2-3 shows a 1:4 **CLK** to **CORECLK** relationship.



**Figure 2-3 CLK to CORECLK 1:4 relationship**

Interface signals that occur between **CORECLK** and **CLK** must always be at-speed signals. Multicycle paths are not allowed at the interface.

**L2RAMCLKEN**

Clock enable for L2 data RAM.

Figure 2-4 on page 2-10 shows the relationship between **CLK** and **L2RAMCLKEN**.

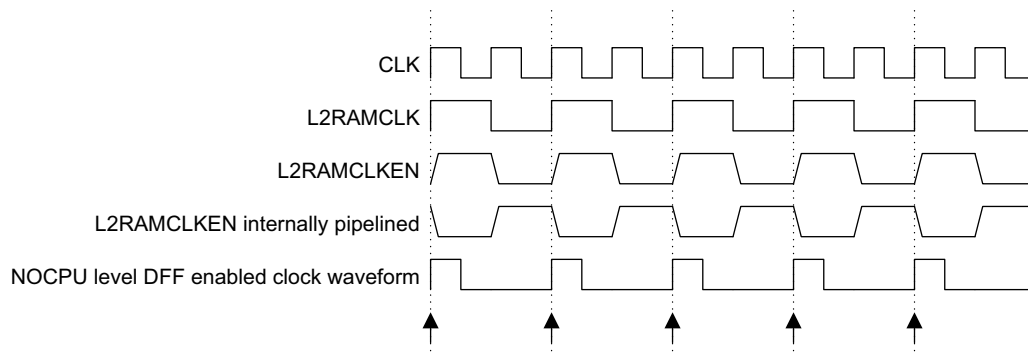


Figure 2-4 CLK and L2RAMCLKEN relationship

Figure 2-5 shows the relationship between **CLK**, **L2RAMCLK** and **L2RAMCLKEN**.

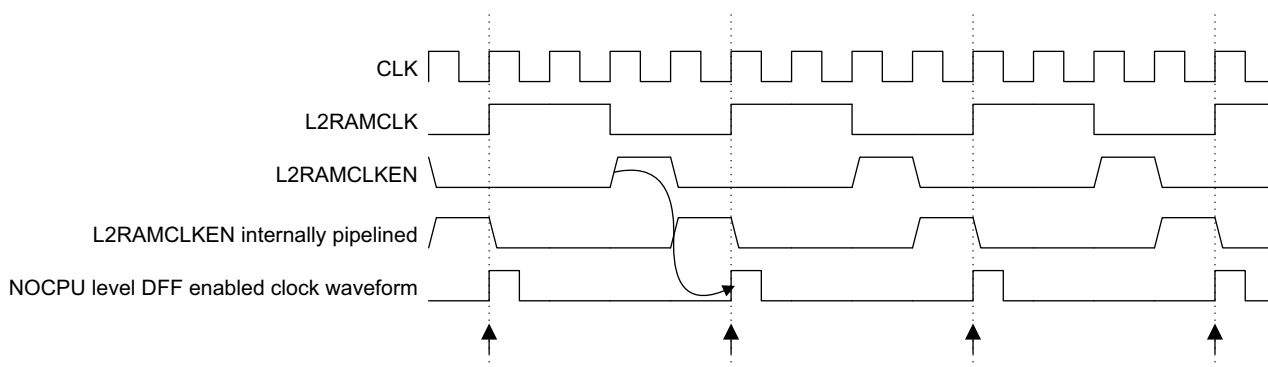


Figure 2-5 CLK to L2RAMCLK 1:4 relationship

### PCLKENDBG

Clock enable to **PCLKDBG**. It allows the external APB bus to be driven at a lower frequency. If the debug infrastructure in the system is required to be fully asynchronous to the processor clock, you must use a synchronizing component to connect the external AMBA APB to the processor.

———— **Note** ————

**PCLKDBG** must not be faster than the slowest **CORECLK**.

Figure 2-6 on page 2-11 shows a 1:2 relationship between **CLK** and **PCLKENDBG**.

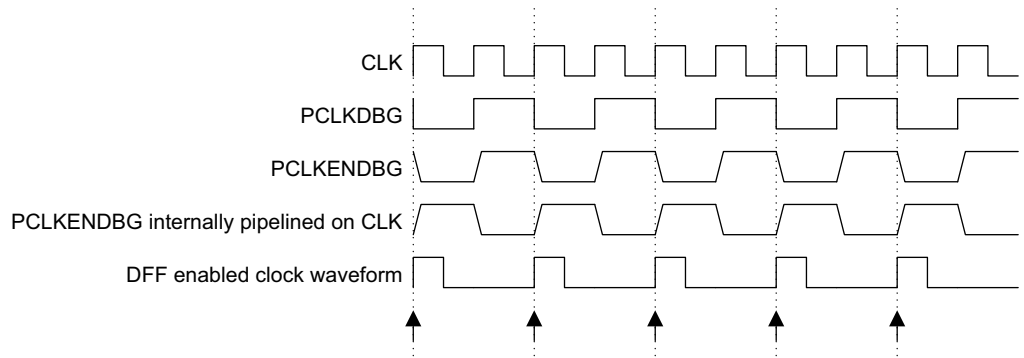


Figure 2-6 CLK and PCLKENDBG 1:2 relationship

Figure 2-7 shows a 1:4 relationship between **CLK** and **PCLKDBG**.

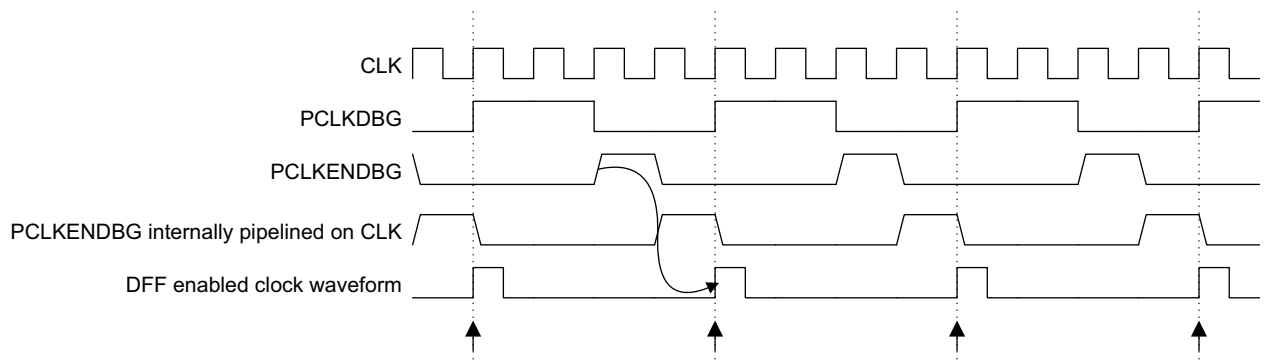


Figure 2-7 CLK and PCLKDBG 1:4 relationship

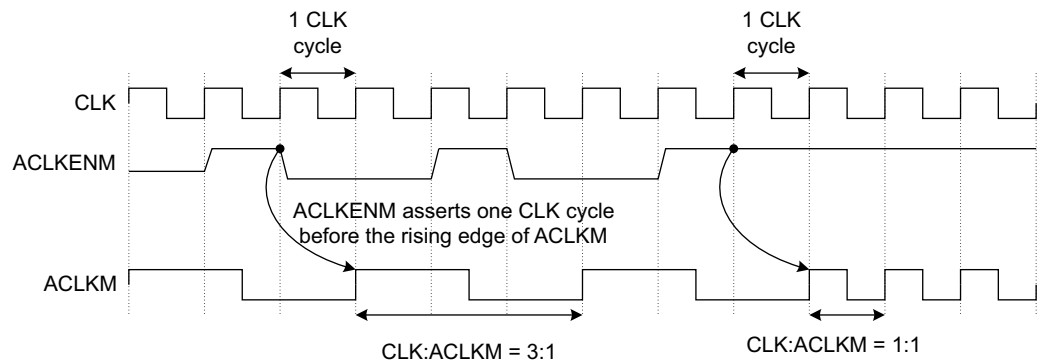
### ACLKENM

Clock enable for the ACE master interface. The SCU interface supports integer ratios of the **CLK** frequency, for example 1:1, 2:1, 3:1. These ratios are achieved through external clock enable signals. In all cases ACE transfers remain synchronous.

**ACLKENM** is registered once within the processor. It must be asserted one **CLK** cycle before the rising edge of the external ACE clock signal.

Figure 2-8 on page 2-12 shows a timing example of **ACLKENM** that changes the **CLK** to **ACLKM** frequency ratio from 3:1 to 1:1.





**Figure 2-8 ACLKENM with CLK:ACLKM ratio changing from 3:1 to 1:1**

**Note**

Figure 2-8 shows the timing relationship between the ACE master clock, **ACLKM** and **ACLKENM**, where **ACLKENM** asserts one clock cycle before the rising edge of **ACLKM**. It is important that the relationship between **ACLKM** and **ACLKENM** is maintained.

### ATCLKEN

Clock enable for the ATB interface of the PTM. It must be synchronous to **CLK**.

The **CLK** to **ATCLKEN** ratio can be calculated by taking the common multiple of all the **CLK** to **CORECLK** ratios in the cluster. The minimum ratio permitted is the LCM.

For example, if the **CLK** to **CORECLK** ratios are 1:2, 1:2, 1:4, and 1:5 then the minimum **CLK** to **ATCLKEN** ratio must be 1:20.

**ATCLKEN** must be asserted one **CLK** cycle prior to the rising edge of the external ATB transaction.

### CNTCLKEN

Clock enable for the generic timer count input, **CNTVALUEB**. It must be synchronous to **CLK** and be an integer multiple of **CLK**. **CNTCLKEN** must be asserted one **CLK** cycle prior to sampling **CNTVALUEB**.

## 2.3.3 Resets

The Cortex-A17 MPCore processor has multiple reset domains.

- Cortex-A17 MPCore multiprocessor resets can be asserted asynchronously but they must be deasserted synchronously for correct internal functionality. For deassertion, all sequential elements must receive the same first clock edge:
  - Reset repeaters, if present, handle this automatically, and you can deassert reset signals asynchronously.
  - If reset repeaters are not present, you must follow a specific reset sequence to ensure that reset signals are deasserted synchronously.
- Assert relevant clock enable signals during a reset sequence to ensure logic correctly sample reset values. **PCLKENDBG** and **CORECLKEN** especially must be active during core and debug resets.

- The resets are active-LOW signals that reset logic in the appropriate domain of the Cortex-A17 MPCore processor.
- Resets are closely associated with power domains. See [Power domains on page 2-24](#).

### Individual processor resets

[Table 2-1](#) shows the individual processor resets. These reset signals are up to 4-bit signals, where each bit represents one processor in the multiprocessor device.

**Table 2-1 Individual processor resets**

Reset	Description
<b>nCOREPORESET[N:0]</b>	Reset of all logic within the PDCPU power domain. Includes: <ul style="list-style-type: none"> <li>• Processor.</li> <li>• Debug.</li> <li>• PTM.</li> <li>• Breakpoint and watchpoint logic.</li> </ul>
<b>nCORERESSET[N:0]</b>	Reset of non-debug logic within the PDCPU power domain. Includes: <ul style="list-style-type: none"> <li>• Processor.</li> <li>• PTM.</li> </ul> Excludes: <ul style="list-style-type: none"> <li>• Debug logic in the PDCPU power domain.</li> </ul>
<b>nCOREPRESETDBG[N:0]</b>	Reset of debug logic within the PDCPU power domain. Includes: <ul style="list-style-type: none"> <li>• Debug.</li> <li>• Breakpoint and watchpoint logic.</li> </ul> Excludes: <ul style="list-style-type: none"> <li>• Processor.</li> <li>• PTM.</li> </ul>

### Multiprocessor resets

[Table 2-2](#) shows the resets that apply to common logic within the multiprocessor.

**Table 2-2 Multiprocessor resets**

Reset	Description
<b>nTOPPRESETDBG</b>	Initializes logic within the PDSOC power domain. Includes: <ul style="list-style-type: none"> <li>• Shared Debug-APB logic.</li> <li>• Top debug logic.</li> <li>• CTI and CTM.</li> </ul>
<b>nL2RESET</b>	Reset initializes the shared Level-2 memory system.
<b>nMBISTRESET</b>	Overrides all resets in MBIST mode.

## Reset combinations

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

**Table 2-3 Valid reset combinations**

Reset combination	Signals	Value	Description
All processor powerup reset	<b>nCOREPORESET [N:0]</b> <b>nCORERESET [N:0]</b> <b>nCOREPRESETDBG[N:0]</b> <b>nL2RESET</b> <b>nTOPPRESETDBG</b> <b>nMBISTRESET</b>	all = 0 all = 0 <sup>a</sup> all = 0 <sup>a</sup> 0 0 0	All logic is held in reset.
Individual processor powerup reset with Debug reset	<b>nCOREPORESET [N:0]</b> <b>nCORERESET [N:0]</b> <b>nCOREPRESETDBG[N:0]</b> <b>nL2RESET</b> <b>nTOPPRESETDBG</b> <b>nMBISTRESET</b>	[n] = 0 [n] = 0 <sup>a</sup> [n] = 0 <sup>a</sup> 1 1 1	Individual processor and Debug are held in reset. Debug logic in the PDSOC power domain is not reset.
All processor powerup and L2 resets with Top debug active	<b>nCOREPORESET [N:0]</b> <b>nCORERESET [N:0]</b> <b>nCOREPRESETDBG[N:0]</b> <b>nL2RESET</b> <b>nTOPPRESETDBG</b> <b>nMBISTRESET</b>	all = 0 all = 0 <sup>a</sup> all = 0 <sup>a</sup> 0 1 1	All processors and L2 are held in reset, so they can be powered up. This enables external debug over powerdown for all processors.
Individual processor powerup reset with Top debug active	<b>nCOREPORESET [N:0]</b> <b>nCORERESET [N:0]</b> <b>nCOREPRESETDBG[N:0]</b> <b>nL2RESET</b> <b>nTOPPRESETDBG</b> <b>nMBISTRESET</b>	[n] = 0 [n] = 0 <sup>a</sup> [n] = 0 <sup>a</sup> 1 1 1	Individual processor is held in reset, so that the processor can be powered up. This enables external debug over powerdown for the processor that is held in reset.
All processors software reset	<b>nCOREPORESET [N:0]</b> <b>nCORERESET [N:0]</b> <b>nCOREPRESETDBG[N:0]</b> <b>nL2RESET</b> <b>nTOPPRESETDBG</b> <b>nMBISTRESET</b>	all = 1 all = 0 all = 1 1 1 1	All logic excluding Debug and L2 memory system is held in reset. All breakpoints and watchpoints are retained. This enables debug over reset for all processors.
All processors software reset and L2 reset	<b>nCOREPORESET [N:0]</b> <b>nCORERESET [N:0]</b> <b>nCOREPRESETDBG[N:0]</b> <b>nL2RESET</b> <b>nTOPPRESETDBG</b> <b>nMBISTRESET</b>	all = 1 all = 0 all = 1 0 1 1	All logic excluding Debug is held in reset. All breakpoints and watchpoints are retained. This enables debug over reset for all processors.

Table 2-3 Valid reset combinations (continued)

Reset combination	Signals	Value	Description
Individual processor software reset	<b>nCOREPORESET [N:0]</b>	[n] = 1	Individual processor logic excluding Debug and PTM is held in reset. Breakpoints and watchpoints for that processor are retained. This enables debug over reset for processors held in reset.
	<b>nCORERESET [N:0]</b>	[n] = 0	
	<b>nCOREPRESETDBG[N:0]</b>	[n] = 1	
	<b>nL2RESET</b>	1	
	<b>nTOPPRESETDBG</b>	1	
	<b>nMBISTRESET</b>	1	
All processors debug reset	<b>nCOREPORESET [N:0]</b>	all = 1	Debug is held in reset. <sup>b</sup>
	<b>nCORERESET [N:0]</b>	all = 1	
	<b>nCOREPRESETDBG[N:0]</b>	all = 0	
	<b>nL2RESET</b>	1	
	<b>nTOPPRESETDBG</b>	0	
	<b>nMBISTRESET</b>	1	
Individual processor Debug reset	<b>nCOREPORESET [N:0]</b>	[n] = 1	Individual processor Debug is held in reset. <sup>c</sup>
	<b>nCORERESET [N:0]</b>	[n] = 1	
	<b>nCOREPRESETDBG[N:0]</b>	[n] = 0	
	<b>nL2RESET</b>	1	
	<b>nTOPPRESETDBG</b>	1	
	<b>nMBISTRESET</b>	1	
Run mode	<b>nCOREPORESET [N:0]</b>	all = 1	No logic is held in reset.
	<b>nCORERESET [N:0]</b>	all = 1	
	<b>nCOREPRESETDBG[N:0]</b>	all = 1	
	<b>nL2RESET</b>	1	
	<b>nTOPPRESETDBG</b>	1	
	<b>nMBISTRESET</b>	1	
MBIST mode	<b>nCOREPORESET [N:0]</b>	all = 0	MBIST mode active. MBIST mode is used for testing the macrocell RAMs. It must not be enabled in normal operation.
	<b>nCORERESET [N:0]</b>	all = 0	
	<b>nCOREPRESETDBG[N:0]</b>	all = 0	
	<b>nL2RESET</b>	0	
	<b>nTOPPRESETDBG</b>	0	
	<b>nMBISTRESET</b>	1	

a. For powerup reset or processor reset, **nCOREPORESET** must be asserted. The remaining processor resets, **nCORERESET** and **nCOREPRESETDBG[N:0]**, can be asserted, but it is not required.

b. A full debug reset during a debug transaction results in unpredictable behavior.

c. A core level debug reset during a debug transaction results in unpredictable behavior.

### Powerup reset

The sequence of operations required to reset Cortex-A17 MPCore processor depends on whether reset repeaters are implemented or not.

- When reset repeaters are implemented, you must:
  - Apply powerup reset to the multiprocessor when power is first applied to the SoC. Logic in all reset domains are placed in a benign state following the removal of powerup reset. This state must be held for 16 cycles of the slowest clock (**CLK**, **CORECLK**, or **PCLKDBG**).
- When reset repeaters are not implemented, you must:
  1. Apply powerup reset to the multiprocessor when power is first applied to the SoC.

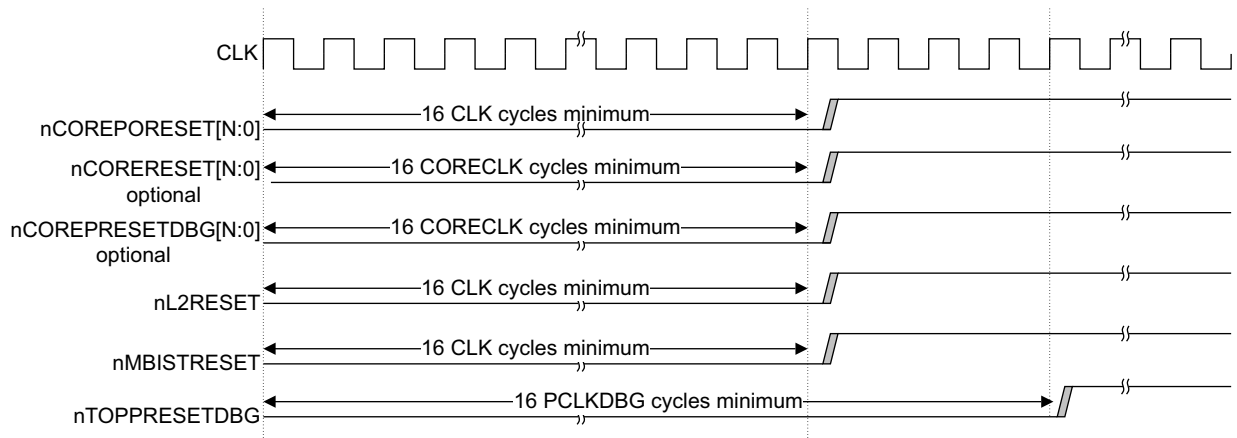
2. Apply a minimum of 16 clock cycles in each reset domain with powerup reset applied.
3. Stop the clocks in each reset domain.
4. Remove powerup reset, wait for 16 cycles in order to allow for signal propagation.
5. Restart the clocks, all sequential elements now see the same first clock edge.

Table 2-4 Shows the reset domains and their associated clocks.

**Table 2-4 Clocks and reset domains**

Clock	Reset domain
CLK	<ul style="list-style-type: none"> <li>• nCOREPORRESET[N:0].</li> <li>• nL2RESET.</li> <li>• nMBISTRESET.</li> </ul>
CORECLK	<ul style="list-style-type: none"> <li>• nCORERESSET[N:0].</li> <li>• nCOREPRESETDBG[N:0].</li> </ul>
PCLKDBG	nTOPPRESETDBG[N:0].

Figure 2-9 shows the powerup reset timing sequence when reset repeaters are implemented.



**Figure 2-9 Powerup reset timing**

## 2.4 Power management

Cortex-A17 MPCore processor provides features to reduce operating power. Each processor can enter a low-power state using the *Wait For Interrupt* (WFI) or *Wait For Event* (WFE) mechanisms that enable the processor to gate the majority of clocks, and remain in a low-power state until it receives an interrupt or event from another processor or external source.

See [Dynamic power management](#).

If state retention is implemented in your macrocell, further power saving is possible. Cortex-A17 MPCore processor provides two interfaces for your power management hardware to control processor state retention and L2 state retention.

See [State retention control on page 2-20](#).

Further power savings can be made by switching power to the macrocell power domains.

If power domain switching is implemented in your macrocell, additional signals are implemented to control power to the processor and L2 memory system power domains. These signals are not described in this book. Refer to your macrocell implementer for information about their names and functionality. The signals perform the following:

- Power domain switching signals, enable power up and power down of individual domains.
- Power domain isolation signals, are used for clamping the values of signals into and out of power switched domains.

See [Power domains on page 2-24](#).

### 2.4.1 Dynamic power management

The following sections describe the methods of entering standby mode:

- [Processor Wait for Interrupt](#).
- [Processor Wait for Event on page 2-18](#).
- [L2 Wait for Interrupt on page 2-19](#).

The processor exits standby mode on interrupts or external events.

See [Event communication using WFE or SEV on page 2-20](#).

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

#### Processor Wait for Interrupt

Wait for Interrupt is a feature of the ARMv7 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. Apart from a small dynamic power overhead, on the logic to enable the processor to wake up from WFI low-power state, the power drawn is reduced to static leakage.

Software indicates that the processor can enter the WFI low-power state by executing the WFI instruction.

The WFI instruction ensures all explicit memory accesses that occurred before the WFI instruction in program order, have retired. For example, the WFI instruction ensures that the following instructions received the required data or responses from the L2 memory system:

- Load instructions.
- Cache and TLB maintenance operations.

- Store exclusives instructions.

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 SCU snoop request that must be serviced by the processor L1 data cache.
- A cache or TLB maintenance operation that must be serviced by the processor L1 instruction cache, data cache, or TLB.
- 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 SCU snoop request, cache, or TLB maintenance operation or an APB access.

Figure 2-10 shows the upper bound for the **STANDBYWFI** deassertion timing after the assertion of **nIRQ** or **nFIQ** inputs.

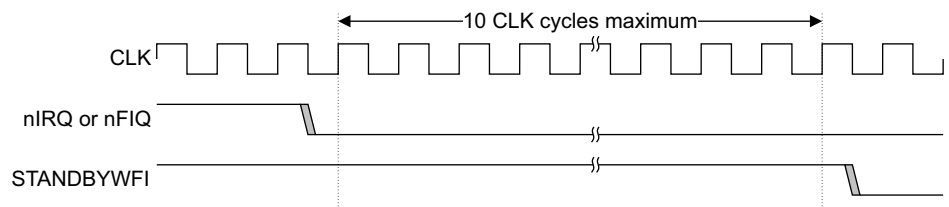


Figure 2-10 STANDBYWFI deassertion timing

### Processor Wait for Event

Wait for Event is a feature of the ARMv7 architecture that permits a processor in a multiprocessor system to request entry to a low-power state. Entry into the low-power state occurs when a WFE hint instruction is successfully executed. Exit occurs on a WFE wake-up event, unmasked interrupts, or asynchronous aborts.

Wake-up events include:

- Another processor in the multiprocessor system executes a Send Event hint instruction (SEV).
- Timer events.
- Input signal **EVENTI** is asserted.

Each processor includes a single-bit Event Register which, if set, prevents entry into low-power state. It is cleared by the WFE instruction and set by a wake-up event. This prevents the processor suspending when an event has been generated.

See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information about wake-up events and the Event Register.

In the low-power state most of the clocks in the processor are disabled 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.

External events might require the processor to temporarily enable the clocks to service the event. This does not cause the processor to exit WFE low-power state. The events are:

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

On entry into WFE low-power state, **STANDBYWFE** for that processor is asserted. Assertion of **STANDBYWFE** guarantees that the processor is in idle and low-power state. **STANDBYWFE** continues to assert even if the clocks in the processor are temporarily enabled.

The upper bound for the **STANDBYWFE** deassertion timing after the assertion of **nIRQ** or **nFIQ** inputs is identical to **STANDBYWFI** as shown in [Figure 2-10 on page 2-18](#).

## L2 Wait for Interrupt

When all the processors in the multiprocessor 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.

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

- All processors in the multiprocessor 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 signals, **ACINACTM** to idle the ACE master interface, and **AINACTS** to idle the ACP slave. This prevents the L2 memory system from accepting any new requests from the ACE master interface.
- When the L2 memory system completes any outstanding AXI and ACE transactions, it enters the L2 WFI low-power state and asserts **STANDBYWFIL2**. Assertion of **STANDBYWFIL2** guarantees that the L2 memory system is in idle and does not accept any new transactions.

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

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

If **ACINACTM** or **AINACTS** are deasserted **STANDBYWFIL2** is deasserted.

[Figure 2-11 on page 2-20](#) shows the L2 WFI timing for a 4-processor configuration.



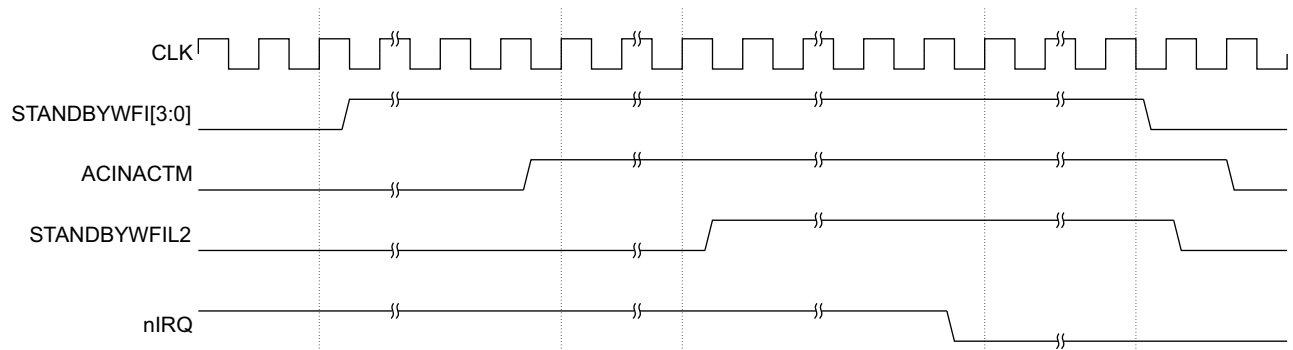


Figure 2-11 L2 Wait For Interrupt timing

### Event communication using WFE or SEV

An external agent can use the **EVENTI** pin to participate in the WFE or SEV event communications of the Cortex-A17 MPCore processor. When this pin is asserted, it sends an event message to all the processors in the cluster. This is similar to executing a SEV instruction on one processor in the cluster. This enables the external agent to signal that the processors can leave the WFE low-power state. The **EVENTI** input pin must remain HIGH for at least one **CLK** clock cycle to be visible by the processors.

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

## 2.4.2 State retention control

State retention is an implementation option. Refer to your implementer for additional information.

### Processor dynamic retention

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

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

Each processor in the multiprocessor has an interface that allows an external power controller to determine the operation state of the processor and to place the processor into a retention state. This interface consists of four pins:

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

The operational relationship of these signals are:

- **CPUQREQn** can only go LOW, if **CPUQACCEPTn** is HIGH and **CPUQDENY** is LOW.

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

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

1. The processor executes a WFI instruction. The clocks in the processor are stopped and **STANDBYWFI** is asserted.
2. The external power controller asserts **CPUQREQn** to indicate that it wants to put that processor into retention state.
3. While the processor is still in WFI low-power state and the clocks are stopped, the processor accepts the retention request by asserting **CPUQACCEPTn**.
4. While **CPUQREQn** and **CPUQACCEPTn** are both asserted, the processor is in quiescent state and the external power controller can safely put the processor into retention state.
5. During retention, if a snoop occurs to access the cache of the quiescent processor, the **CPUQACTIVE** signal is asserted to request exit from retention.
6. The external power controller brings the processor out of retention and deasserts **CPUQREQn**.
7. The processor deasserts **CPUQACCEPTn** to complete the handshake.
8. The clocks in the processor are restarted temporarily to allow the snoop request to the processor to proceed.
9. After the snoop access is complete, the processor deasserts **CPUQACTIVE**.
10. **CPUQREQn** and **CPUQACCEPTn** are then asserted. The processor has reentered quiescent state and the external power controller can put the processor into retention state again.
11. When the processor is ready to exit WFI low-power state, **CPUQACTIVE** is asserted.
12. **CPUQREQn** is then deasserted, the processor exits WFI low-power state, and **CPUQACCEPTn** is deasserted.

[Figure 2-12 on page 2-22](#) shows a typical sequence where the external power controller successfully places the processor in retention state.

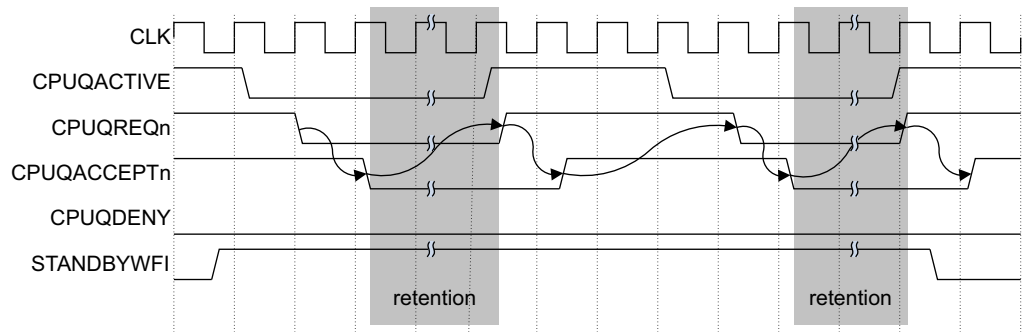


Figure 2-12 Successful retention timing

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

Figure 2-13 shows a sequence where the external power controller attempts to put a processor in retention state but the processor denies the request.

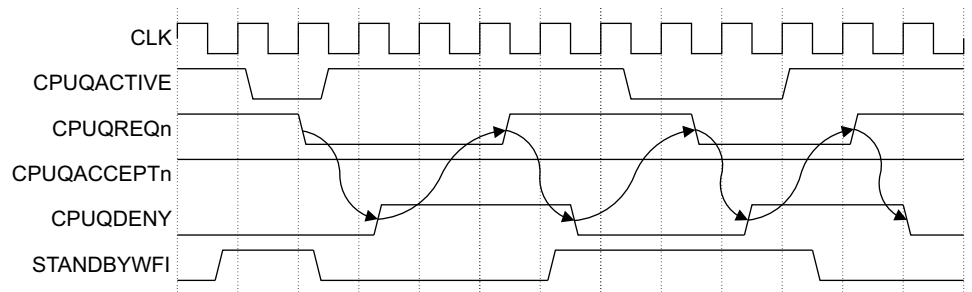


Figure 2-13 Denied retention timing

#### Guidelines on the use of processor dynamic retention

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:

- Processor retention mode, if implemented, is enabled at reset. To disable processor retention, you must set the processor retention mode bit in the SCU Control Register, SCUCTLR, for the intended processor. If retention mode for processor<n> is disabled, all assertions of **CPUQREQn** LOW are ignored and **CPUQACCEPTn** is held HIGH and **CPUQDENY** is held LOW.
- If the processor dynamic retention feature is not used, **CPUQREQn** must be tied HIGH and the SCUCTLR retention control bit set to disabled.

See [SCU Control Register](#) on page 4-82.

## L2 RAMs dynamic retention

The multiprocessor supports dynamic retention of the L2 data and tag RAMs. The multiprocessor has an interface that allows an external power controller to determine the operation state of the multiprocessor and to place the L2 RAMs into a retention state.

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

1. All processors are in WFI low-power state and therefore, all the processors **STANDBYWFI** outputs are asserted.
2. When all outstanding ACP requests are complete, the SoC asserts the input **AINACTS** to idle the ACP slave interface.
3. When all pending L2 activity is complete, the L2 deasserts **L2QACTIVE**.
4. The external power controller asserts **L2QREQn** to indicate that it wants to put the L2 RAMs into retention state.
5. If the L2 is still idle, it accepts the retention request by asserting **L2QACCEPTn**.
6. While **L2QREQn** and **L2QACCEPTn** are both asserted, the power controller can safely put the L2 RAMs into retention state.
7. If the L2 detects that one or more processors have exited WFI low-power state, the ACP becomes active or a snoop request must be serviced, the L2 asserts **L2QACTIVE** to request exit from retention.
8. The power controller brings the L2 RAMs out of retention and deasserts **L2QREQn**.
9. The L2 deasserts **L2QACCEPTn** to complete the handshake.

Figure 2-13 on page 2-22 shows the L2 dynamic retention timing.

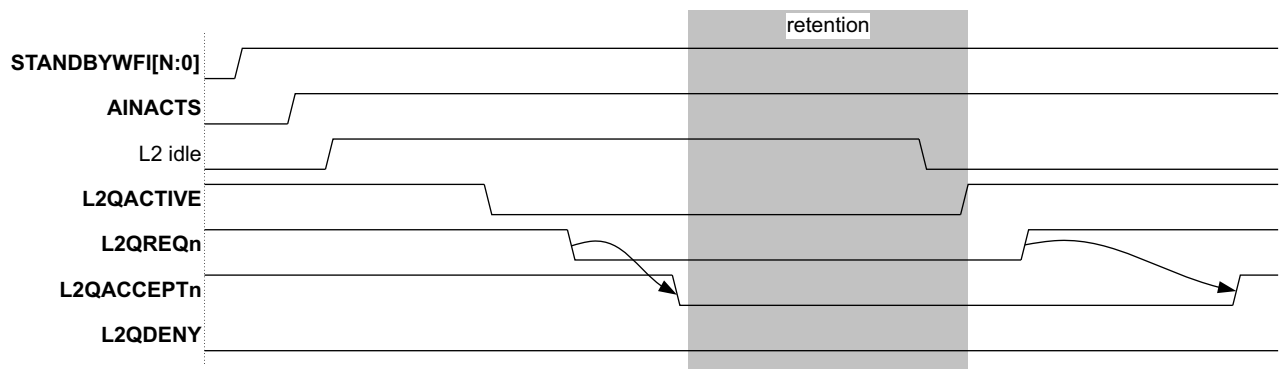


Figure 2-14 L2 dynamic retention timing

If the L2 exits idle in step 5, it asserts **L2QDENY** instead of **L2QACCEPTn**. In response, the power controller must deassert **L2QREQn**, causing the L2 to deassert **L2QDENY**.

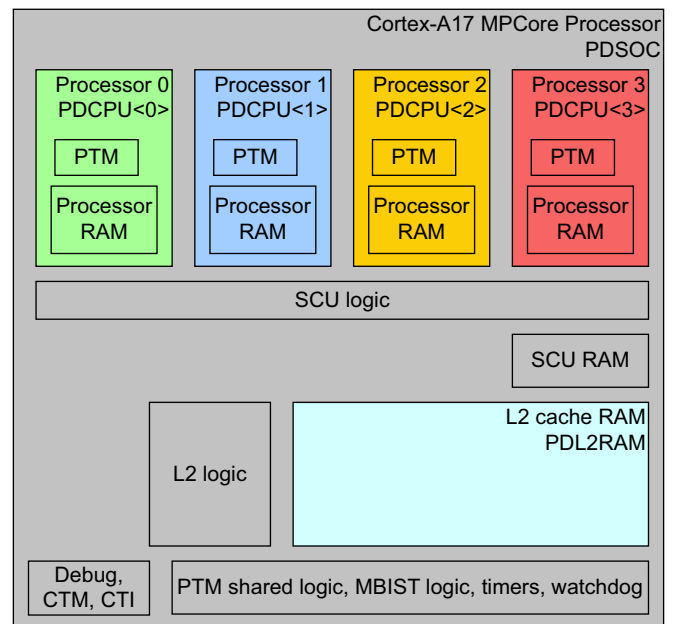
If the L2 dynamic retention feature is not used, **L2QREQn** must be tied HIGH and **L2ECTLR[0]** must be set.

### 2.4.3 Power domains

The Cortex-A17 MPCore processor power domains are:

- PDCPU<n> is the power domain for each processor and includes:
  - Processor<n>.
  - Advanced SIMD and VFP engine.
  - L1 Instruction and data TLBs and cache RAMs.
  - Branch prediction RAMs.
  - Processor-level debug logic.
  - PTM logic.
- PDSOC is the top-level power domain and includes:
  - SCU control logic.
  - SCU duplicate tag RAMs.
  - L2 cache control logic.
  - L2 linefill buffer, store buffer, and eviction buffer RAMs.
  - Store buffer RAMs.
  - Debug APB slave, CTM and CTI.
- PDL2RAM is the L2 cache RAM power domain and includes:
  - L2 data and tag RAMs.

Figure 2-15 shows an example of the domains embedded in a *System-on-Chip* (SoC) power domain.



**Figure 2-15 Power domains**

## 2.4.4 Power modes

The power domains can be controlled independently. However, only some combinations of domain powerup and powerdown states are permitted. Control for power domains is an implementation option.

Table 2-5 shows the valid power state combinations for the different possible modes of operation.

**Table 2-5 Supported power modes**

Mode	PDSOC	PDL2RAM	PDCPU<n>
Run mode	Powered up	Powered up	Powered up
Individual Processor n <sup>a</sup> shutdown mode	Powered up	Powered up	Powered down
Multiprocessor shutdown mode	Powered down	Powered down	Powered down
Multiprocessor dormant mode	Powered up	Powered up	Powered down

a. Where n represents 0-3. The multiprocessor can have several individual processors in shutdown mode.

The supported power modes are:

- [Run mode.](#)
- [Individual processor shutdown mode.](#)
- [Multiprocessor device shutdown mode on page 2-26.](#)
- [Dormant mode on page 2-28.](#)

### Note

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

## Run mode

This is the normal mode of operation where all of the processor functionality is available. The Cortex-A17 MPCore processor disables clocks and inputs to unused functional blocks. Only the logic in use to perform an operation consumes dynamic power.

## Individual processor shutdown mode

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

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

To power down the processor, apply the following sequence:

1. Clear the SCTL.R.C bit, or HSCTL.R.C bit if in Hyp mode, to prevent further data cache allocation.
2. Execute a CLREX instruction.

3. Clean and invalidate all data from the L1 data cache. The SCU 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 device being issued to this processor.
4. 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 or TLB maintenance operations broadcast by other processors in the multiprocessor device.

---

**Note**

---

You must disable all interrupts to the processor before, or at, this point.

---

5. Execute a DSB instruction to ensure that all cache, TLB and branch predictor maintenance operations issued by any processor in the multiprocessor device before the SMP bit was cleared have completed.
6. Execute an ISB instruction to ensure that all of the CP15 register changes from the previous steps have been committed.
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. Deassert **DBGPWRDUP** LOW. This ensures that external Debug APB accesses to the processor are disabled and return a slave error response.
8. Assert **nCOREPORESET** LOW.
9. Assert the processor output clamps.
10. Remove power from the PDCPU<n> power domain.

---

**Note**

---

**CORECLKEN** must not be held LOW while in shutdown mode. Clocks to **CORECLK** must be maintained. See [Clock enables on page 2-8](#).

---

To power up the processor, apply the following sequence:

1. Assert **nCOREPORESET** LOW. Ensure **DBGPWRDUP** is held LOW to prevent any external debug access to the processor.
2. Apply power to the PDCPU<n> power domain. Keep the state of the signals **nCOREPORESET** and **DBGPWRDUP** LOW.
3. Deassert the processor output clamps.
4. Deassert resets.
5. Assert **DBGPWRDUP** HIGH to allow external Debug APB access to the processor.
6. If required, use software to restore the state of the processor prior to powerdown.
7. Assert ACTLR.SMP bit HIGH for SMP mode. Continue a normal powerup reset sequence.

### Multiprocessor device shutdown mode

ARM recommends that one processor supervises the control of shutting down the multiprocessor. This is referred to as the lead processor.

In this mode the PDSOC and PDCPU power domains are shut down and all state is lost. To power down the multiprocessor device, apply the following sequence:

1. Ensure all non-lead processors are in shutdown mode, see [Individual processor shutdown mode on page 2-25](#).
2. Follow steps 1. to 3. in [Individual processor shutdown mode on page 2-25](#).
3. Clean and invalidate all data from L2 cache.
4. Follow steps 4 to 10. in [Individual processor shutdown mode on page 2-25](#).
5. Assert **ACINACTM** and **AINACTS** and wait until the **STANDBYWFI2** output is asserted to indicate that the L2 memory system is idle.
6. Assert the multiprocessor device output clamps.
7. Assert all resets LOW:
  - **nCOREPORESET[N:0]**.
  - **nCORERESET[N:0]**. Can be asserted, but it is not required.
  - **nCOREPRESETDBG[N:0]**. Can be asserted, but it is not required.
  - **nL2RESET**.
  - **nTOPPRESETDBG**.
  - **nMBISTRESET**.
8. Remove power from the PDSOC power domain.

---

**Note**

---

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

---

To power up the multiprocessor device, apply the following sequence:

1. For each processor in the multiprocessor device, assert **nCOREPORESET** LOW.
2. Assert **nL2RESET** LOW and hold **L2RSTDISABLE** LOW.
3. Apply power to the PDSOC and PDCPU domains while keeping the signals described in steps 1. and 2. LOW.
4. Deassert the multiprocessor device output clamps.
5. Continue a normal powerup reset sequence. Deassert the resets.
6. For each processor in the multiprocessor device, set the **ACTLR.SMP** bit to 1 for SMP mode.

---

**Note**

---

You must ensure the **ACTLR.SMP** bit is set to 1 before the caches and MMU are enabled, or any cache and TLB maintenance operations are performed. The only time this bit is set to 0 is during a processor powerdown sequence.

---



## Dormant mode

Optionally, the Dormant mode is supported in the multiprocessor device. In this mode all the processors and L2 control logic are powered down while the L2 cache RAMs are powered up and retain state. The RAM blocks that remain powered up during Dormant mode are:

- L2 tag RAMs.
- L2 data RAMs.

To support Dormant mode, you must ensure:

- That the L2 cache RAMs are in a separate power domain.
- To clamp all inputs to the L2 cache RAMs to benign values. This avoids 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 multiprocessor device, 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 processor exits the powerup reset sequence, and the architectural state must be restored.

To enter Dormant mode, apply the following sequence:

1. Clear the SCTLR.C bit to prevent further data cache allocation.
2. Execute a CLREX instruction.
3. Clean and invalidate all data from the L1 data cache. The SCU 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 device being issued to this processor.
4. 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 or TLB maintenance operations broadcast by other processors in the multiprocessor device.
5. 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.
6. Execute a DSB instruction to ensure that all cache, TLB and branch predictor maintenance operations issued by any processor in the multiprocessor device before the SMP bit was cleared have completed. In addition, this ensures that all state saving has completed.
7. Execute an ISB instruction to ensure that all of the CP15 register changes from the previous steps have been committed.
8. Execute a WFI instruction and wait until the **STANDBYWFI** output is asserted, to indicate that the processor is in idle and low-power state.
9. Repeat the previous steps for all processors, and wait for all **STANDBYWFI** outputs to be asserted.

10. The SoC asserts the input signals **ACINACTM** and **AINACTS** to idle the AXI master 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, **STANDBYWFI** is asserted to indicate that L2 memory system is idle.
11. When all processors **STANDBYWF** and **STANDBYWFIL2** are asserted, the multiprocessor device is ready to enter Dormant mode.
12. Assert the L2 cache RAM input clamps.
13. Remove power from the PDCPU power domains.

To exit Dormant mode, apply the following sequence:

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

# Chapter 3

## Programmers Model

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

- *About the programmers model on page 3-2.*
- *ThumbEE instruction set on page 3-3.*
- *Jazelle Architecture Extension on page 3-4.*
- *Advanced SIMD and VFP Extensions on page 3-6.*
- *Security Extensions architecture on page 3-7.*
- *Virtualization Extensions architecture on page 3-8.*
- *Large Physical Address Extension architecture on page 3-9.*
- *Multiprocessing Extensions on page 3-10.*
- *Operating states on page 3-11.*
- *Memory model on page 3-12.*

### 3.1 About the programmers model

The Cortex-A17 MPCore processor implements the ARMv7 architecture. This includes:

- The 32-bit ARM instruction set.
- The Thumb instruction set that has both 16-bit and 32-bit instructions.
- The *Thumb Execution Environment* (ThumbEE) instruction set.
- A trivial implementation of the Jazelle® architecture extension.
- The Advanced SIMD and VFP Extensions.
- The Security Extensions.
- The Virtualization Extensions.
- The Large Physical Address Extension.
- The Multiprocessing Extensions.

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

## 3.2 ThumbEE instruction set

The ThumbEE extension is a variant of the Thumb instruction set that is designed as a target for dynamically generated code.

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

### 3.3 Jazelle Architecture Extension

The Cortex-A17 MPCore processor provides a trivial implementation of the Jazelle Extension. This means that the processor does not accelerate the execution of any bytecodes.

In the trivial implementation of the Jazelle architecture extension:

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

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

#### 3.3.1 Register summary

[Table 3-1](#) gives a summary of the Cortex-A17 Jazelle Extension registers.

**Table 3-1 Summary of Jazelle Extension registers**

CRn	Op1	CRm	Op2	Name	Reset	Description
c0	7	c0	0	JIDR	0x00000000	<i>Jazelle Identity Register</i>
c1	7	c0	0	JOSCR	0x00000000	<i>Jazelle OS Control Register on page 3-5</i>
c2	7	c0	0	JMCR	0x00000000	<i>Jazelle Main Configuration Register on page 3-5</i>

#### 3.3.2 Register description

This section describes the Cortex-A17 Jazelle Extension registers. [Table 3-1](#) provides cross-references to individual register bits.

##### Jazelle Identity Register

The JIDR characteristics are:

<b>Purpose</b>	Enables software to determine the implementation of the Jazelle Extension provided by the processor.
<b>Usage constraints</b>	The JIDR is: <ul style="list-style-type: none"> <li>• A read-only register.</li> <li>• Accessible from all privilege levels.</li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 3-1</a> .

[Table 3-2](#) shows the JIDR bit assignments.

**Table 3-2 JIDR Register bit assignments**

Bits	Name	Function
[31:0]	-	Reserved, RAZ

To access the JIDR, read the CP14 register with:

MRC p14, 7, <Rd>, c0, c0, 0; Read Jazelle Identity Register

## Jazelle OS Control Register

The JOSCR characteristics are:

<b>Purpose</b>	Provides operating system control of the use of the Jazelle Extension.
<b>Usage constraints</b>	The JOSCR is: <ul style="list-style-type: none"> <li>• A read/write register.</li> <li>• Accessible only from PL1 or higher.</li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 3-1 on page 3-4</a> .

[Table 3-3](#) shows the JOSCR bit assignments.

**Table 3-3 JOSCR Register bit assignments**

Bits	Name	Function
[31:0]	-	Reserved, RAZ/WI

To access the JOSCR, read or write the CP14 register with:

MRC p14, 7, <Rd>, c1, c0, 0; Read Jazelle OS Control Register  
MCR p14, 7, <Rd>, c1, c0, 0; Write Jazelle OS Control Register

## Jazelle Main Configuration Register

The JMCR characteristics are:

<b>Purpose</b>	Provides control of the Jazelle Extension features.
<b>Usage constraints</b>	The JMCR is: <ul style="list-style-type: none"> <li>• A read/write register, with access rights that depend on the current privilege level: <ul style="list-style-type: none"> <li>— Write-only in unprivileged level.</li> <li>— Read-write at PL1 or higher.</li> </ul> </li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 3-1 on page 3-4</a> .

[Table 3-4](#) shows the JMCR bit assignments.

**Table 3-4 JMCR Register bit assignments**

Bits	Name	Function
[31:0]	-	Reserved, RAZ/WI

To access the JMCR, read or write the CP14 register with:

MRC p14, 7, <Rd>, c2, c0, 0; Read Jazelle Main Configuration Register  
MCR p14, 7, <Rd>, c2, c0, 0; Write Jazelle Main Configuration Register

## 3.4 Advanced SIMD and VFP Extensions

The 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.

The 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.



## 3.5 Security Extensions architecture

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

- [System boot sequence](#).
- [Security Extensions write access disable](#).

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

### 3.5.1 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 the NS bit set to 0. See [Secure Configuration Register on page 4-58](#). 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 against further 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.

### 3.5.2 Security Extensions write access disable

The Security Extensions write access disable, pin **CP15SDISABLE**, is not supported. This signal must be tied LOW.

## 3.6 Virtualization Extensions architecture

The Virtualization Extensions are a set of features that provide hardware support for virtualizing the Non-secure state of an ARM VMSAv7 implementation. This supports system use of a virtual machine monitor, known as the hypervisor, to switch 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 9 Debug](#).
- The PMUv2 Performance Monitors, see [Chapter 10 Performance Monitoring Unit](#).

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

## 3.7 Large Physical Address Extension architecture

The *Large Physical Address Extension* (LPAE) is an extension to the *Virtual Memory System Architecture* (VMSAv7) that provides an address translation system that supports physical addresses of up to 40 bits.

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

## 3.8 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.

## 3.9 Operating states

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

<b>ARM state</b>	The processor executes 32-bit, word-aligned ARM instructions.
<b>Thumb state</b>	The processor executes 16-bit and 32-bit, halfword-aligned Thumb instructions.
<b>ThumbEE state</b>	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 3-5](#) shows the encoding of these bits.

**Table 3-5 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.

### 3.10 Memory model

The Cortex-A17 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:

- [About system control on page 4-2.](#)
- [Register summary on page 4-3.](#)
- [Register descriptions on page 4-27.](#)

## 4.1 About system control

The CP15 system registers provide control and status information for the functions implemented in the processor. The main functions of the CP15 system registers are:

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



## 4.2 Register summary

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

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

- [c0 registers on page 4-5.](#)
- [c1 registers on page 4-6.](#)
- [c2 registers on page 4-7.](#)
- [c3 registers on page 4-7.](#)
- [c4 registers on page 4-7.](#)
- [c5 registers on page 4-7.](#)
- [c6 registers on page 4-8.](#)
- [c7 registers on page 4-8.](#)
- [c8 registers on page 4-10.](#)
- [c9 registers on page 4-11.](#)
- [c10 registers on page 4-12.](#)
- [c11 registers on page 4-12.](#)
- [c12 registers on page 4-13.](#)
- [c13 registers on page 4-13.](#)
- [c14 registers on page 4-13.](#)
- [c15 registers on page 4-14.](#)

The Cortex-A17 MPCore processor supports the *Virtualization Extensions* (VE) and the *Large Physical Address Extension* (LPAE). See [Virtualization Extensions architecture on page 3-8](#) and [Large Physical Address Extension architecture on page 3-9](#) and [Chapter 8 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 4-15.](#)

This section also summarizes the CP15 system control registers by functional groups:

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

[Table 4-1](#) describes the column headings in the CP15 register summary tables used throughout this section.

**Table 4-1 System control register field values**

Heading	Description
CRn	Primary 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 system control registers you can access when CRn is c0.

**Table 4-2 c0 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c0	0	c0	0	MIDR	0x411FC0E0	<a href="#">Main ID Register on page 4-27</a>
			1	CTR	..a	<a href="#">Cache Type Register on page 4-28</a>
			2	TCMTR	0x00000000	<a href="#">TCM Type Register on page 4-29</a>
			3	TLBTR	0x00000000	<a href="#">TLB Type Register on page 4-29</a>
			5	MPIDR	..b	<a href="#">Multiprocessor Affinity Register on page 4-29</a>
			6	REVIDR	0x00000000	<a href="#">Revision ID Register on page 4-30</a>
			4, 7	MIDR	0x411FC0E0	Aliases of Main ID Register, <a href="#">Main ID Register on page 4-27</a>
		c1	0	ID_PFR0	0x00001131	<a href="#">Processor Feature Register 0 on page 4-31</a>
			1	ID_PFR1	0x00011011	<a href="#">Processor Feature Register 1 on page 4-32</a>
			2	ID_DFR0	0x02010555	<a href="#">Debug Feature Register 0 on page 4-33</a>
			3	ID_AFR0	0x00000000	<a href="#">Auxiliary Feature Register 0 on page 4-34</a>
			4	ID_MMFR0	0x10101105	<a href="#">Memory Model Feature Register 0 on page 4-34</a>
			5	ID_MMFR1	0x40000000	<a href="#">Memory Model Feature Register 1 on page 4-35</a>
			6	ID_MMFR2	0x01240000	<a href="#">Memory Model Feature Register 2 on page 4-37</a>
			7	ID_MMFR3	0x02102211	<a href="#">Memory Model Feature Register 3 on page 4-38</a>
		c2	0	ID_ISAR0	0x02101110	<a href="#">Instruction Set Attribute Register 0 on page 4-40</a>
			1	ID_ISAR1	0x13112111	<a href="#">Instruction Set Attribute Register 1 on page 4-41</a>
			2	ID_ISAR2	0x21232041	<a href="#">Instruction Set Attribute Register 2 on page 4-43</a>
			3	ID_ISAR3	0x11112131	<a href="#">Instruction Set Attribute Register 3 on page 4-44</a>
			4	ID_ISAR4	0x00011142	<a href="#">Instruction Set Attribute Register 4 on page 4-46</a>
			5	ID_ISAR5	0x00000000	<a href="#">Instruction Set Attribute Register 5 on page 4-47</a>
	1	c0	0	CCSIDR	UNK	<a href="#">Cache Size ID Register on page 4-47</a>
			1	CLIDR	0x0A200023	<a href="#">Cache Level ID Register on page 4-49</a>
			7	AIDR	0x00000000	<a href="#">Auxiliary ID Register on page 4-50</a>
	2	c0	0	CSSELR	UNK	<a href="#">Cache Size Selection Register on page 4-50</a>
	4	c0	0	VPIDR	..c	<a href="#">Virtualization Processor ID Register on page 4-51</a>
			5	VMPIDR	..d	<a href="#">Virtualization Multiprocessor ID Register on page 4-52</a>

a. The reset value depends on the primary input **IMINLN**:

0x84448003 If **IMINLN** is LOW.

0x84448004 If **IMINLN** is HIGH.

b. The reset value depends on the primary input CLUSTER ID and the CPU ID value in the Multiprocessor Affinity Register.

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 system control registers you can access when CRn is c1.

**Table 4-3 c1 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c1	0	c0	0	SCTLR	0x00C50878 <sup>a</sup>	<i>System Control Register on page 4-52</i>
			1	ACTLR	0x00000006	<i>Auxiliary Control Register on page 4-55</i>
			2	CPACR	0x00000000	<i>Coprocessor Access Control Register on page 4-57</i>
		c1	0	SCR	0x00000000	<i>Secure Configuration Register on page 4-58</i>
			1	SDER	UNK	Secure Debug Enable Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	NSACR	0x00000000	<i>Non-secure Access Control Register on page 4-60</i>
	4	c0	0	HSCTLR	UNK	<i>Hyp System Control Register on page 4-62</i>
			1	HACTLR	UNK	<i>Hyp Auxiliary Control Register on page 4-64</i>
		c1	0	HCR	0x00000000	Hyp Configuration Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	HDCR	0x00000006 <sup>b</sup>	<i>Hyp Debug Control Register on page 4-64</i>
			2	HCPtr	- <sup>c</sup>	<i>Hyp Coprocessor Trap Register on page 4-66</i>
			3	HSTR	0x00000000	Hyp System Trap Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			7	HACR	UNK	<i>Hyp Auxiliary Configuration Register on page 4-68</i>

- a. The reset value depends on primary inputs, **TEINIT**, **CFGEND**, and **VINITHI**. The value shown in Table 4-2 on page 4-5 assumes these signals are set to zero.
- b. The reset value for bit [7] is UNK.
- c. The reset value depends on the NEON and floating-point unit configuration. If VFP and Advanced SIMD extensions are implemented, the reset value is 0x001033FF. If VFP and Advanced SIMD extensions are not implemented, the reset value is 0x0010BFFF.

### 4.2.3 c2 registers

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

**Table 4-4 c2 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c2	0	c0	0	TTBR0	UNK	Translation Table Base Register 0, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	TTBR1	UNK	Translation Table Base Register 1, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	TTBCR	0x00000000 <sup>a</sup>	Translation Table Base Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
4	c0	2	HTCR	UNK		<a href="#">Hyp Translation Control Register on page 4-68</a>
		c1	VTCT	UNK		Virtualization Translation Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

- 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 system control registers you can access when CRn is c3.

**Table 4-5 c3 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c3	0	c0	0	DACR	UNK	Domain Access Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

### 4.2.5 c4 registers

There are no system control registers to access when CRn is c4.

### 4.2.6 c5 registers

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

**Table 4-6 c5 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c5	0	c0	0	DFSR	UNK	<a href="#">Data Fault Status Register on page 4-68</a>
			1	IFSR	UNK	<a href="#">Instruction Fault Status Register on page 4-72</a>
		c1	0	ADFSR	0x00000000	<a href="#">Auxiliary Data Fault Status Register on page 4-75</a>
			1	AIFSR	0x00000000	<a href="#">Auxiliary Instruction Fault Status Register on page 4-75</a>
4	c1	0	HADFSR	UNK		<a href="#">Hyp Auxiliary Data Fault Status Syndrome Register on page 4-75</a>
			HAIFSR	UNK		<a href="#">Hyp Auxiliary Instruction Fault Status Syndrome Register on page 4-75</a>
	c2	0	HSR	UNK		<a href="#">Hyp Syndrome Register on page 4-75</a>

## 4.2.7 c6 registers

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

**Table 4-7 c6 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c6	0	c0	0	DFAR	UNK	Data Fault Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	IFAR	UNK	Instruction Fault Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	4	c0	0	HDFAR	UNK	Hyp Data Fault Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	HIFAR	UNK	Hyp Instruction Fault Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			4	HPFAR	UNK	Hyp IPA Fault Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

## 4.2.8 c7 registers

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

**Table 4-8 c7 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c7	0	c0	4	NOP	UNK	No Operation, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			0	ICIALLUIS	UNK	Instruction cache invalidate all to PoU <sup>a</sup> Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
				BPIALLIS	UNK	Branch predictor invalidate all Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c4	0	PAR	UNK	<a href="#">Physical Address Register on page 4-76</a>
		c5	0	ICIALLU	UNK	Instruction cache invalidate all to PoU, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	ICIMVAU	UNK	Instruction cache invalidate by MVA to PoU, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			4	CP15ISB	UNK	Instruction Synchronization Barrier operation, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			6	BPIALL	UNK	Branch predictor invalidate all, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			7	BPIMVA	UNK	Branch predictor invalidate by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c6	1	DCIMVAC	UNK	Data cache invalidate by MVA to PoC <sup>b</sup> , see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	DCISW	UNK	Data cache invalidate line by set/way, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

Table 4-8 c7 register summary (continued)

CRn	Op1	CRm	Op2	Name	Reset	Description
c7	0	c8	0	ATS1CPR	UNK	Stage 1 current state PL1 read, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	ATS1CPW	UNK	Stage 1 current state PL1 write, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	ATS1CUR	UNK	Stage 1 current state unprivileged (PL0) read, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			3	ATS1CUW	UNK	Stage 1 current state unprivileged (PL0) write, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			4	ATS12NSOPR	UNK	Stages 1 and 2 Non-secure PL1 read, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			5	ATS12NSOPW	UNK	Stages 1 and 2 Non-secure PL1 write, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			6	ATS12NSOUR	UNK	Stages 1 and 2 Non-secure unprivileged (PL0) read, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			7	ATS12NSOUW	UNK	Stages 1 and 2 Non-secure unprivileged (PL0) write, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c10	1	DCCMVAC	UNK	Data cache clean line by MVA to PoC, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	DCCSW	UNK	Data cache clean line by set/way, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			4	CP15DSB	UNK	Data Synchronization Barrier operation, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			5	CP15DMB	UNK	Data Memory Barrier operation, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c11	1	DCCMVAU	UNK	Clean data cache line by MVA to PoU, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c13	1	NOP	UNK	No Operation, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c14	1	DCCIMVAC	UNK	Data cache clean and invalidate line by MVA to PoC, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	DCCISW	UNK	Data cache clean and invalidate line by set/way, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	4	c8	0	ATS1HR	UNK	Add translation stage 1 Hyp mode read, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	ATS1HW	UNK	Add translation stage 1 Hyp mode write, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

a. PoU = Point of Unification.

b. PoC = Point of Coherence.

## 4.2.9 c8 registers

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

**Table 4-9 c8 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c8	0	c3	0	TLBIALLIS	UNK	Invalidate entire TLB Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	TLBIMVAIS	UNK	Invalidate unified TLB entry by MVA Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	TLBIASIDIS	UNK	Invalidate unified TLB by ASID match Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			3	TLBIMVAAIS	UNK	Invalidate unified TLB by MVA all ASID Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	c5	0	0	ITLBIALL	UNK	Invalidate instruction TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	ITLBIMVA	UNK	Invalidate instruction TLB entry by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	ITLBIASID	UNK	Invalidate instruction TLB by ASID match, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	c6	0	0	DTLBIALL	UNK	Invalidate data TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	DTLBIMVA	UNK	Invalidate data TLB entry by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	DTLBIASID	UNK	Invalidate data TLB by ASID match, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	c7	0	0	TLBIALL	UNK	Invalidate unified TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	TLBIMVA	UNK	Invalidate unified TLB entry by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	TLBIASID	UNK	Invalidate unified TLB by ASID match, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			3	TLBIMVAA	UNK	Invalidate unified TLB by MVA all ASID, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
4	c3	0	0	TLBIALLHIS	UNK	Invalidate entire Hyp Unified TLB Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	TLBIMVAHIS	UNK	Invalidate Unified Hyp TLB entry by MVA Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			4	TLBIALLNSNHIS	UNK	Invalidate entire NS Non-Hyp Unified TLB Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>



Table 4-9 c8 register summary (continued)

CRn	Op1	CRm	Op2	Name	Reset	Description
c8	4	c7	0	TLBIAL LH	UNK	Invalidate entire Hyp Unified TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	TLBIMVAH	UNK	Invalidate Unified Hyp TLB entry by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			4	TLBIAL LNSNH	UNK	Invalidate entire NS Non-Hyp Unified TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

#### 4.2.10 c9 registers

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

Table 4-10 c9 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c9	0	c12	0	PMCR	0x41D30000	<a href="#">Performance Monitor Control Register on page 10-7</a>
			1	PMNCNTENSET	UNK	Count Enable Set Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	PMNCNTENCLR	UNK	Count Enable Clear Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			3	PMOVS R	UNK	Overflow Flag Status Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			4	PMSWINC	UNK	Software Increment Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			5	PMSELR	UNK	Event Counter Selection Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			6	PMCEID0	0x3BFFF0F3F	Common Event Identification Register 0, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			7	PMCEID1	0x00000000	Common Event Identification Register 1, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c13	0	PMCCNTR	UNK	Cycle Count Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	PMXEVTYPER	UNK	Event Type Selection Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	PMXVCNTR	UNK	Event Count Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c14	0	PMUSERENR	0x00000000	User Enable Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

Table 4-10 c9 register summary (continued)

CRn	Op1	CRm	Op2	Name	Reset	Description
c9	0	c14	1	PMINTENSET	UNK	Interrupt Enable Set Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	PMINTENCLR	UNK	Interrupt Enable Clear Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			3	PMOVSSET	UNK	Performance Monitor Overflow Flag Status Set Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	1	c0	2	L2CTLR	0x00000000 <sup>a</sup>	<a href="#">L2 Control Register on page 4-77</a>
			3	L2ECTLR	0x00000000	<a href="#">L2 Extended Control Register on page 4-81</a>
			4	SCUCTLR	0x00000000 <sup>a</sup>	<a href="#">SCU Control Register on page 4-82</a>
			6	L2MRERRSR	0x00000000	<a href="#">L2 Memory Error Syndrome Register on page 4-87</a>

a. The reset value depends on the processor configuration.

#### 4.2.11 c10 registers

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

Table 4-11 c10 register summary

CRn	Op1	CRm	Op2	Name	Reset	Description
c10	0	c2	0	PRRR	UNK	Primary Region Remap Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
				MAIR0	UNK	Memory Attribute Indirection Register 0, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	NMRR	UNK	Normal Memory Remap Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
				MAIR1	UNK	Memory Attribute Indirection Register 1, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c3	0	AMAIRO	UNK	<a href="#">Auxiliary Memory Attribute Indirection Register 0 on page 4-89</a>
			1	AMAIR1	UNK	<a href="#">Auxiliary Memory Attribute Indirection Register 1 on page 4-89</a>
	4	c2	0	HMAIR0	UNK	Hyp Memory Attribute Indirection Register 0, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	HMAIR1	UNK	Hyp Memory Attribute Indirection Register 1, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c3	0	HAMAIRO	UNK	<a href="#">Hyp Auxiliary Memory Attribute Indirection Register 0 on page 4-89</a>
			1	HAMAIR1	UNK	<a href="#">Hyp Auxiliary Memory Attribute Indirection Register 1 on page 4-89</a>

#### 4.2.12 c11 registers

There are no system control registers to access when CRn is c11.

### 4.2.13 c12 registers

Table 4-12 shows the 32-bit wide system control registers you can access when CRn is c10.

**Table 4-12 c10 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c12	0	c0	0	VBAR	0x00000000 <sup>a</sup>	Vector Base Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	MVBAR	UNK	Monitor Vector Base Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c1	0	ISR	UNK	Interrupt Status Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	4	c0	0	HVBAR	UNK	Hyp Vector Base Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

- 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.

### 4.2.14 c13 registers

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

**Table 4-13 c13 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c13	0	c0	0	FCSEIDR	0x00000000	<a href="#">FCSE Process ID Register on page 4-89</a>
			1	CONTEXTIDR	UNK	FCSE Process ID Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			2	TPIDRURW	UNK	User Read/Write Thread ID Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			3	TPIDRURO	UNK	User Read Only Thread ID Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			4	TPIDRPRW	UNK	Privileged Only Thread ID Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	4	c0	2	HTPIDR	UNK	Hyp Software Thread ID Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

### 4.2.15 c14 registers

See [Chapter 8 Generic Timer](#) for information on the System Timer registers.

## 4.2.16 c15 registers

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

**Table 4-14 c15 register summary**

CRn	Op1	CRm	Op2	Name	Reset	Description
c15	0	c0	0	DGNCTLR0	UNK	<a href="#">Diagnostic control registers on page 4-84</a>
			1	DGNCTLR0	UNK	<a href="#">Diagnostic control registers on page 4-84</a>
			2	DGNCTLR0	UNK	<a href="#">Diagnostic control registers on page 4-84</a>
			4	DGNCCTLR	UNK	<a href="#">Diagnostic common control register on page 4-85</a>
	1	c14	0	DCCIALl	UNK	<a href="#">Data Cache Clean and Invalidate All on page 4-76</a>
	3	c0	0	CDBGDR0	UNK	Direct access to internal memory, Data Register 0. See <a href="#">Direct access to internal memory on page 6-10</a>
			1	CDBGDR1	UNK	Direct access to internal memory, Data Register 1. See <a href="#">Direct access to internal memory on page 6-10</a>
			2	CDBGDR2	UNK	Direct access to internal memory, Data Register 2. See <a href="#">Direct access to internal memory on page 6-10</a>
		c2	0	CDBGDCT	UNK	Data Cache Tag Read Operation Register. See <a href="#">Direct access to internal memory on page 6-10</a>
			1	CDBGICT	UNK	Instruction Cache Tag Read Operation Register. See <a href="#">Direct access to internal memory on page 6-10</a>
		c4	0	CDBGDCD	UNK	Data Cache Data Read Operation Register. See <a href="#">Direct access to internal memory on page 6-10</a>
			1	CDBGICD	UNK	Instruction Cache Data Read Operation Register. See <a href="#">Direct access to internal memory on page 6-10</a>
			2	CDBGTD	UNK	TLB Data Read Operation Register. See <a href="#">Direct access to internal memory on page 6-10</a>
	4	c0	1	FILASTARTR	..a	<a href="#">Peripheral port start address register on page 4-85</a>
			2	FILAENDR	..b	<a href="#">Peripheral port end address register on page 4-86</a>

- a. The reset value depends on the primary inputs CFGADDRFILSTARTNS[39:20], CFGADDRFILTNNS, CFGADDRFILSTARTS[39:20], CFGADDRFILTENS.
- b. The reset value depends on the primary inputs, CFGADDFILTENDNS[39:20], CFGADDFILTENDS[39:20].

### 4.2.17 64-bit registers

Table 4-15 gives a summary of the 64-bit wide CP15 system control registers, accessed by the MCRR and MRCC 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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
-	1	c2	-	TTBR1	UNK	Translation Table Base Register 1, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
-	4	c2	-	HTTBR	UNK	Hyp Translation Table Base Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
-	6	c2	-	VTTBR	UNK <sup>a</sup>	Virtualization Translation Table Base Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
-	0	c7	-	PAR	UNK	<a href="#">Physical Address Register on page 4-76</a>
-	0	c14	-	CNTPCT	UNK	Counter Physical Count Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
-	1	c14	-	CNTVCT	UNK	Counter Virtual Count Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
-	2	c14	-	CNTP_CVAL	UNK	Counter PL1 Physical Compare Value Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
-	3	c14	-	CNTV_CVAL	UNK	Counter PL1 Virtual Compare Value Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
-	4	c14	-	CNTVOFF	UNK	Counter Virtual Offset Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
-	6	c14	-	CNTHP_CVAL	UNK	Counter Non-secure PL2 Physical Compare Value Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

a. The reset value for bits [55:48] is 0b00000000.

### 4.2.18 Identification registers

Table 4-16 shows the identification registers.

**Table 4-16 Identification registers**

Name	CRn	Op1	CRm	Op2	Reset	Description
MIDR	c0	0	c0	0	0x410FC0E0	<a href="#">Main ID Register on page 4-27</a>
CTR				1	— <sup>a</sup>	<a href="#">Cache Type Register on page 4-28</a>
TCMTR				2	0x00000000	<a href="#">TCM Type Register on page 4-29</a>
TLBTR				3	0x00000000	<a href="#">TLB Type Register on page 4-29</a>
MPIDR				5	— <sup>b</sup>	<a href="#">Multiprocessor Affinity Register on page 4-29</a>
REVIDR				6	0x00000000	<a href="#">Revision ID Register on page 4-30</a>

Table 4-16 Identification registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
MIDR				4, 7	0x410FC0E0	Aliases of Main ID Register, <a href="#">Main ID Register</a> on page 4-27
ID_PFR0			c1	0	0x00001131	<a href="#">Processor Feature Register 0</a> on page 4-31
ID_PFR1				1	0x00011011	<a href="#">Processor Feature Register 1</a> on page 4-32
ID_DFR0				2	0x02010555	<a href="#">Debug Feature Register 0</a> on page 4-33
ID_AFR0				3	0x00000000	<a href="#">Auxiliary Feature Register 0</a> on page 4-34
ID_MMFR0				4	0x10101105	<a href="#">Memory Model Feature Register 0</a> on page 4-34
ID_MMFR1				5	0x40000000	<a href="#">Memory Model Feature Register 1</a> on page 4-35
ID_MMFR2				6	0x01240000	<a href="#">Memory Model Feature Register 2</a> on page 4-37
ID_MMFR3				7	0x02102211	<a href="#">Memory Model Feature Register 3</a> on page 4-38
ID_ISAR0			c2	0	0x02101110	<a href="#">Instruction Set Attribute Register 0</a> on page 4-40
ID_ISAR1				1	0x13112111	<a href="#">Instruction Set Attribute Register 1</a> on page 4-41
ID_ISAR2				2	0x21232041	<a href="#">Instruction Set Attribute Register 2</a> on page 4-43
ID_ISAR3				3	0x11112131	<a href="#">Instruction Set Attribute Register 3</a> on page 4-44
ID_ISAR4				4	0x10011142	<a href="#">Instruction Set Attribute Register 4</a> on page 4-46
ID_ISAR5	c0	0	c2	5	0x00000000	<a href="#">Instruction Set Attribute Register 5</a> on page 4-47
CCSIDR		1	c0	0	UNK	<a href="#">Cache Size ID Register</a> on page 4-47
CLIDR				1	0x0A200023	<a href="#">Cache Level ID Register</a> on page 4-49
AIDR				7	0x00000000	<a href="#">Auxiliary ID Register</a> on page 4-50
CSSELR		2	c0	0	UNK	<a href="#">Cache Size Selection Register</a> on page 4-50

- a. The reset value depends on the primary input, **IMINLN**:  
0x84448003 If **IMINLN** is LOW.  
0x84448004 If **IMINLN** is HIGH.
- b. The reset value depends on the primary input, **CLUSTERID**, and the number of configured processors in the Cortex-A17 MPCore processor.

#### 4.2.19 Virtual memory control registers

Table 4-17 shows the virtual memory control registers.

Table 4-17 Virtual memory control registers

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
SCTLR	c1	0	c0	0	0x00C50878 <sup>a</sup>	32-bit	<a href="#">System Control Register</a> on page 4-52
TTBR0	c2	0	c0	0	UNK	32-bit	Translation Table Base Register 0, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	-	0	c2	-		64-bit	
TTBR1		0	c0	1	UNK	32-bit	Translation Table Base Register 1, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	-	1	c2	-		64-bit	

Table 4-17 Virtual memory control registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
TTBCR		0	c0	2	0x00000000 <sup>b</sup>	32-bit	Translation Table Base Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DACR	c3	0	c0	0	UNK	32-bit	Domain Access Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PRRR	c10	0	c2	0	UNK	32-bit	Primary Region Remap Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
MAIR0				0	UNK	32-bit	Memory Attribute Indirection Register 0, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
NMRR				1	UNK	32-bit	Normal Region Remap Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
MAIR1				1	UNK	32-bit	Memory Attribute Indirection Register 1, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
CONTEXTIDR	c13	0	c0	1	UNK	32-bit	Process ID Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

- a. The reset value depends on primary inputs, **TEINIT**, **CFGEND**, and **VINITI**. The value shown in [Table 4-17 on page 4-16](#) 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.

#### 4.2.20 PL1 Fault handling registers

[Table 4-18](#) shows the PL1 Fault handling registers.

Table 4-18 PL1 Fault handling registers

Name	CRn	Op1	CRm	Op2	Reset	Description
DFSR	c5	0	c0	0	UNK	<a href="#">Data Fault Status Register on page 4-68</a>
IFSR				1	UNK	<a href="#">Instruction Fault Status Register on page 4-72</a>
ADFSR			c1	0	0x00000000	<a href="#">Auxiliary Data Fault Status Register on page 4-75</a>
AIFSR				1	0x00000000	<a href="#">Auxiliary Instruction Fault Status Register on page 4-75</a>
DFAR	c6	0	c0	0	UNK	Data Fault Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
IFAR				2	UNK	Instruction Fault Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

The Virtualization Extensions include additional fault handling registers. For more information see [Virtualization Extensions registers on page 4-23](#).

#### 4.2.21 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
ACTLR	c1	0	c0	1	0x00000006	<a href="#">Auxiliary Control Register on page 4-55</a>
CPACR				2	0x00000000	<a href="#">Coprocesor Access Control Register on page 4-57</a>
FCSEIDR	c13	0	c0	0	0x00000000	<a href="#">FCSE Process ID Register on page 4-89</a>

#### 4.2.22 Cache maintenance operations

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

**Table 4-20 Cache and branch predictor maintenance operations**

Name	CRn	Op1	CRm	Op2	Reset	Description
ICIALUIS	c7	0	c1	0	UNK	Instruction cache invalidate all to PoU <sup>a</sup> Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
BPIALLIS				6	UNK	Branch predictor invalidate all Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ICIALLU			c5	0	UNK	Instruction cache invalidate all to PoU, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ICIMVAU				1	UNK	Instruction cache invalidate by MVA to PoU, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
BPIALL	c7	0	c5	6	UNK	Branch predictor invalidate all, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
BPIMVA				7	UNK	Branch predictor invalidate by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DCIMVAC			c6	1	UNK	Data cache invalidate by MVA to PoC <sup>b</sup> , see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DCISW				2	UNK	Data cache invalidate by set/way, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DCCMVAC			c10	1	UNK	Data cache clean by MVA to PoC, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DCCSW				2	UNK	Data cache clean by set/way, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DCCMVAU			c11	1	UNK	Data cache clean by MVA to PoU, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DCCIMVAC			c14	1	UNK	Data cache clean and invalidate by MVA to PoC, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DCCISW				2	UNK	Data cache clean and invalidate by set/way, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DCCIALL	c15	1	c14	0	UNK	<a href="#">Data Cache Clean and Invalidate All on page 4-76</a>

a. PoU = Point of Unification.



b. PoC = Point of Coherence.

#### 4.2.23 TLB maintenance operations

Table 4-21 shows the 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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIMVAIS				1	UNK	Invalidate unified TLB by MVA Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIASIDIS				2	UNK	Invalidate unified TLB by ASID Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIMVAAIS				3	UNK	Invalidate unified TLB by MVA all ASID Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ITLBIALL			c5	0	UNK	Invalidate entire instruction TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ITLBIMVA				1	UNK	Invalidate instruction TLB entry by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ITLBIASID				2	UNK	Invalidate instruction TLB by ASID, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DTLBIALL	c8	0	c6	0	UNK	Invalidate entire data TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DTLBIMVA				1	UNK	Invalidate data TLB entry by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
DTLBIASID				2	UNK	Invalidate data TLB by ASID, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIALL			c7	0	UNK	Invalidate entire unified TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIMVA				1	UNK	Invalidate unified TLB by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIASID				2	UNK	Invalidate unified TLB by ASID, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIMVAA				3	UNK	Invalidate unified TLB by MVA all ASID, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIALLHIS		4	c3	0	UNK	Invalidate entire Hyp unified TLB Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

Table 4-21 TLB maintenance operations (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
TLBIMVAHIS				1	UNK	Invalidate Hyp unified TLB entry by MVA Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIALLSNHIS				4	UNK	Invalidate entire Non-secure Non-Hyp unified TLB Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIALLH			c7	0	UNK	Invalidate entire Hyp unified TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIMVAH				1	UNK	Invalidate Hyp unified TLB entry by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIALLSNH				4	UNK	Invalidate entire Non-secure Non-Hyp unified TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

The Virtualization Extensions include additional TLB operations for use in Hyp mode. For more information, see [Virtualization Extensions registers on page 4-23](#).

#### 4.2.24 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	<a href="#">Physical Address Register on page 4-76</a>
	-		c7	-		64-bit	
ATS1CPR			c8	0	UNK	32-bit	Stage 1 current state PL1 read, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ATS1CPW				1	UNK	32-bit	Stage 1 current state PL1 write, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ATS1CUR	c7	0	c8	2	UNK	32-bit	Stage 1 current state unprivileged (PL0) read, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ATS1CUW				3	UNK	32-bit	Stage 1 current state unprivileged (PL0) write, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ATS12NSOPR				4	UNK	32-bit	Stages 1 and 2 Non-secure PL1 read, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ATS12NSOPW				5	UNK	32-bit	Stages 1 and 2 Non-secure PL1 write, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ATS12NSOUR				6	UNK	32-bit	Stages 1 and 2 Non-secure unprivileged (PL0) read, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

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 (PL0) write, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ATS1HR		4	c8	0	UNK	32-bit	Stage 1 Hyp mode read, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ATS1HW				1	UNK	32-bit	Stage 1 Hyp mode write, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

#### 4.2.25 Miscellaneous operations

Table 4-23 shows the 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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
CP15ISB			c5	4	UNK	Instruction Synchronization Barrier operation, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
CP15DSB			c10	4	UNK	Data Synchronization Barrier operation, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
CP15DMB				5	UNK	Data Memory Barrier operation, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
NOP			c13	1	UNK	System control No Operation (NOP), see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TPIDRURW	c13	0	c0	2	UNK	User Read/Write Thread ID Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TPIDRURO				3	UNK	User Read-Only Thread ID Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TPIDRPRW				4	UNK	PL1 only Thread ID Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
HTPIDR		4	c0	2	UNK	Hyp Software Thread ID Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

## 4.2.26 Performance monitor registers

Table 4-24 shows the performance monitor registers.

**Table 4-24 Performance monitor registers**

Name	CRn	Op1	CRm	Op2	Reset	Description
PMCR	c9	0	c12	0	0x410D3000	<i>Performance Monitor Control Register on page 10-7</i>
PMNCNTENSET				1	UNK	Count Enable Set Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMNCNTENCLR				2	UNK	Count Enable Clear Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMOVSr				3	UNK	Overflow Flag Status Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMSWINC				4	UNK	Software Increment Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMSELR				5	UNK	Event Counter Selection Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMCEID0				6	0x3BFFF0F3F	Common Event Identification Register 0, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMCEID1				7	0x00000000	Common Event Identification Register 1, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMCCNTR			c13	0	UNK	Cycle Count Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMXEVTYPER				1	UNK	Event Type Select Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMXEVCNTR				2	UNK	Event Count Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMUSERENR			c14	0	0x00000000	User Enable Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMINTENSET				1	UNK	Interrupt Enable Set Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMINTENCLR				2	UNK	Interrupt Enable Clear Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
PMOVSSET				3	UNK	Performance Monitor Overflow Flag Status Set Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

## 4.2.27 Security Extensions registers

Table 4-25 shows the Security Extensions registers.

**Table 4-25 Security Extensions registers**

Name	CRn	Op1	CRm	Op2	Reset	Description
SCR	c1	0	c1	0	0x00000000	<a href="#">Secure Configuration Register on page 4-58</a>
SDER				1	UNK	Secure Debug Enable Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
NSACR				2	0x00000000	<a href="#">Non-secure Access Control Register on page 4-60</a>
VBAR	c12	0	c0	0	0x00000000 <sup>a</sup>	Vector Base Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
MVBAR				1	UNK	Monitor Vector Base Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
ISR			c1	0	UNK	Interrupt Status Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

- 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.

## 4.2.28 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	– <sup>a</sup>	32-bit	<a href="#">Virtualization Processor ID Register on page 4-51</a>
VMPIDR				5	– <sup>b</sup>	32-bit	<a href="#">Virtualization Multiprocessor ID Register on page 4-52</a>
HSCTLR	c1	4	c0	0	UNK	32-bit	<a href="#">Hyp System Control Register on page 4-62</a>
HACTLR				1	UNK		<a href="#">Hyp Auxiliary Control Register on page 4-64</a>
HCR			c1	0	0x00000000	32-bit	Hyp Configuration Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
HDCR				1	0x00000006 <sup>c</sup>	32-bit	<a href="#">Hyp Debug Control Register on page 4-64</a>
HCPTR				2	0x000033FF <sup>d</sup>	32-bit	<a href="#">Hyp Coprocessor Trap Register on page 4-66</a>
HSTR				3	0x00000000	32-bit	Hyp System Trap Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
HTCR	c2	4	c0	2	UNK	32-bit	<a href="#">Hyp Translation Control Register on page 4-68</a>
VTCTCR			c1	2	UNK	32-bit	Virtualization Translation Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

Table 4-26 Virtualization Extensions registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
HTTBR	-	4	c2	-	UNK	64-bit	Hyp Translation Table Base Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
VTTBR	-	6	c2	-	UNK <sup>e</sup>	64-bit	Virtualization Translation Table Base Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
HADFSR	c5	4	c1	0	UNK	32-bit	<a href="#">Hyp Auxiliary Data Fault Status Syndrome Register on page 4-75</a>
HAIFSR				1	UNK	32-bit	<a href="#">Hyp Auxiliary Instruction Fault Status Syndrome Register on page 4-75</a>
HSR			c2	0	UNK	32-bit	<a href="#">Hyp Syndrome Register on page 4-75</a>
HDFAR	c6	4	c0	0	UNK	32-bit	Hyp Data Fault Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
HIFAR				2	UNK	32-bit	Hyp Instruction Fault Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
HPFAR				4	UNK	32-bit	Hyp IPA Fault Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
HMAIR0	c10	4	c2	0	UNK	32-bit	Hyp Memory Attribute Indirection Register 0, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
HMAIR1				1	UNK	32-bit	Hyp Memory Attribute Indirection Register 1, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
HAMAIRO			c3	0	UNK	32-bit	<a href="#">Hyp Auxiliary Memory Attribute Indirection Register 0 on page 4-89</a>
HMAIR1				1	UNK	32-bit	<a href="#">Hyp Auxiliary Memory Attribute Indirection Register 1 on page 4-89</a>
HVBAR	c12	4	c0	0	UNK	32-bit	Hyp Vector Base Address Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

- 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 NEON and floating-point unit configuration. If VFP and Advanced SIMD extensions are implemented, the reset value is 0x000033FF. If VFP and Advanced SIMD extensions are not implemented, the reset value is 0x0000BFFF.
- e. The reset value for bits [54:48] is 0b00000000.

## 4.2.29 Hyp mode TLB maintenance operations

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

**Table 4-27 Hyp mode TLB maintenance operations**

Name	CRn	Op1	CRm	Op2	Reset	Description
TLBIALLLHIS	c8	4	c3	0	UNK	Invalidate entire Hyp unified TLB Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIMVAHIS				1	UNK	Invalidate Hyp unified TLB by MVA Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIALLLNSNHIS				4	UNK	Invalidate entire Non-secure Non-Hyp unified TLB Inner Shareable, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIALLLH			c7	0	UNK	Invalidate entire Hyp unified TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIMVAH				1	UNK	Invalidate Hyp unified TLB by MVA, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
TLBIALLLNSNH				4	UNK	Invalidate entire Non-secure Non-Hyp unified TLB, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

## 4.2.30 Generic Timer registers

See [Chapter 8 Generic Timer](#) for information on the timer registers.

## 4.2.31 Implementation defined registers

Table 4-28 shows the 32-bit wide implementation defined registers. These registers provide test features and any required configuration options specific to the Cortex-A17 MPCore processor.

**Table 4-28 Implementation defined registers**

Name	CRn	Op1	CRm	Op2	Reset	Description
L2CTLR	c9	1	c0	2	0x00000000 <sup>a</sup>	<a href="#">L2 Control Register on page 4-77</a>
L2ECTLR				3	0x00000000	<a href="#">L2 Extended Control Register on page 4-81</a>
SCUCTLR				4	0x00000000 <sup>b</sup>	<a href="#">SCU Control Register on page 4-82.</a>
L2MERRSR				6	0x00000000	<a href="#">L2 Memory Error Syndrome Register on page 4-87</a>
DGNCTLR0	c15	0	c0	0	UNK	<a href="#">Diagnostic control registers on page 4-84</a>
DGNCTLR1				1	UNK	<a href="#">Diagnostic control registers on page 4-84</a>
DGNCTLR2				2	UNK	<a href="#">Diagnostic control registers on page 4-84</a>
DGNCCTLR				4	UNK	<a href="#">Diagnostic common control register on page 4-85</a>

Table 4-28 Implementation defined registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
CDBGDR0	c15	3	c0	0	UNK	Data Register 0. See <a href="#">Direct access to internal memory on page 6-10</a>
CDBGDR1				1	UNK	Data Register 1. See <a href="#">Direct access to internal memory on page 6-10</a>
CDBGDR2				2	UNK	Data Register 2. See <a href="#">Direct access to internal memory on page 6-10</a>
CDBGDCT			c2	0	UNK	Data Cache Tag Read Operation Register. See <a href="#">Direct access to internal memory on page 6-10</a>
CDBGICT				1	UNK	Instruction Cache Tag Read Operation Register. See <a href="#">Direct access to internal memory on page 6-10</a>
CDBGDCD			c4	0	UNK	Data Cache Data Read Operation Register. See <a href="#">Direct access to internal memory on page 6-10</a>
CDBGICD				1	UNK	Instruction Cache Data Read Operation Register. See <a href="#">Direct access to internal memory on page 6-10</a>
CDBGTD				2	UNK	TLB Data Read Operation Register. See <a href="#">Direct access to internal memory on page 6-10</a>
FILASTARTR	4	c0	c0	1	..c	<a href="#">Peripheral port start address register on page 4-85</a>
FILAENDR				2	..d	<a href="#">Peripheral port end address register on page 4-86</a>

- a. The reset value depends on the processor configuration.
- b. The reset value depends on the number of processors implemented.
- c. The reset value depends on the primary inputs, **CFGADDRFILSTARTNS[39:20]**, **CFGADDRFILTNNS** and **CFGADDRFILSTARTS[39:20]**, **CFGADDRFILTENS**.
- d. The reset value depends on the primary inputs, **CFGADDFILTENDNS[39:20]** and **CFGADDFILTENDS[39:20]**.



## 4.3 Register descriptions

This section describes all the CP15 system control registers by coprocessor register number order. [Table 4-2 on page 4-5](#) to [Table 4-15 on page 4-15](#) 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 device ID number.

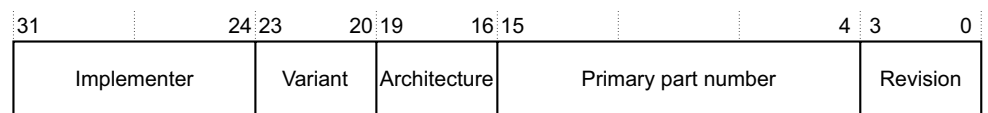
**Usage constraints** The MIDR is:

- 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 4-5](#).

[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.
[23:20]	Variant	Indicates the variant number of the processor. This is the major revision number <i>n</i> in the <i>rn</i> part of the <i>rn</i> pn description of the product revision status: 0x1 Major revision number.
[19:16]	Architecture	Indicates the architecture code: 0xF ARMv7.
[15:4]	Primary part number	Indicates the primary part number: 0xC0E Cortex-A17 MPCore processor part number.
[3:0]	Revision	Indicates the revision number of the processor. This is the minor revision number <i>n</i> in the <i>pn</i> part of the <i>rn</i> pn description of the product revision status: 0x1 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.

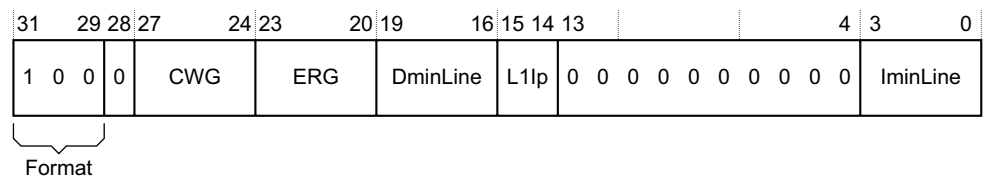
**Usage constraints** The CTR is:

- 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 4-5](#).

[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 CTR format: 0x4 ARMv7 format.
[28]	-	Reserved, RAZ.
[27:24]	CWG	Cache Write-Back granule. Log <sub>2</sub> of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified: 0x4 Cache Write-Back granule size is 16 words.
[23:20]	ERG	Exclusives Reservation Granule. Log <sub>2</sub> 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 <sub>2</sub> 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]	L1lp	L1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache: 0b10 <i>Virtually Indexed Physically Tagged (VIPT)</i> .
[13:4]	-	Reserved, RAZ.
[3:0]	IminLine	Log <sub>2</sub> of the number of words in the smallest cache line of all the instruction caches that the processor controls. The primary input <b>IMINLN</b> defines the reset value: 0x3 Smallest instruction cache line size is 8 words. <b>IMINLN</b> signal is LOW. 0x4 Smallest instruction cache line size is 16 words. <b>IMINLN</b> 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 the TCMTR, so this register is always RAZ/WI.

### 4.3.4 TLB Type Register

The TLBTR characteristics are:

**Purpose** Provides information about the TLB implementation.

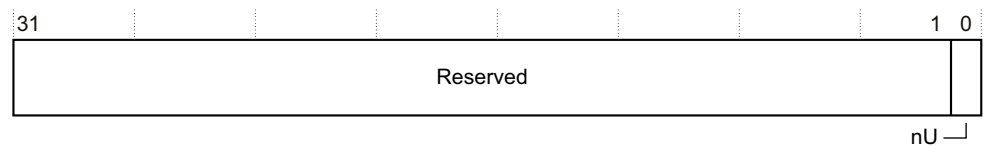
**Usage constraints** The TLBTR is:

- 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 4-3 on page 4-6](#).

[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	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.

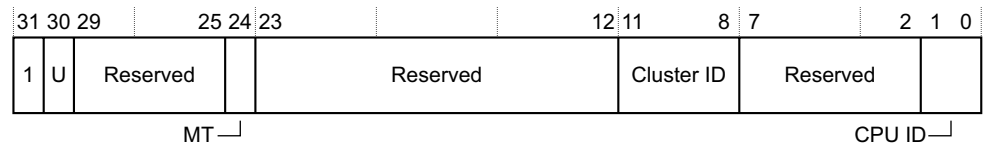
**Usage constraints** The MPIDR is:

- 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 4-5](#).

[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]	-	Indicates that the processor implements the Multiprocessing Extensions register format: 0x1                  ARMv7 multi-processor format.
[30]	U	Indicates a Uniprocessor system, as distinct from processor 0 in a multiprocessor system: 0x0                  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 multi-threading type approach. 0x0                  Processors are not implemented using a multi-threading approach.
[23:12]	-	Reserved, RAZ.
[11:8]	Cluster ID	Indicates the value read in the <b>CLUSTERID</b> configuration bus. It identifies each processor in a multiprocessor configuration.
[7:2]	-	Reserved, RAZ.
[1:0]	CPU ID	Indicates the processor number in the Cortex-A17 MPCore processor: 0x0                  Processor is CPU0. 0x1                  Processor is CPU1. 0x2                  Processor is CPU2. 0x3                  Processor is CPU3.

To access the MPIDR, read the CP15 registers with:

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

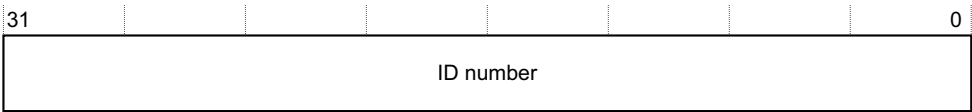
### 4.3.6 Revision ID Register

The REVIDR characteristics are:

<b>Purpose</b>	Provides implementation-specific revision information that can only be interpreted in conjunction with the MIDR.
<b>Usage constraints</b>	The REVIDR is: <ul style="list-style-type: none"> <li>• A read-only register.</li> <li>• Common to the Secure and Non-secure states.</li> <li>• Only accessible from PL1 or higher.</li> </ul>
<b>Configurations</b>	Available in all configurations.

**Attributes** See the register summary in [Table 4-3 on page 4-6](#).

[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-A17 MPCore processor implementation.

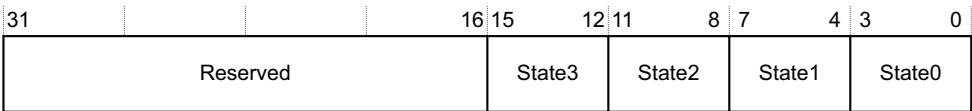
To access the REVIDR, read the CP15 register with:  
MRC p15, 0, <Rt>, c0, c0, 6; Read Revision ID Register

**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 4-5](#).

[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 <i>Thumb Execution Environment</i> (ThumbEE) instruction set: 0x1 ThumbEE instruction set implemented.
[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 architecture extensions supported by the processor.

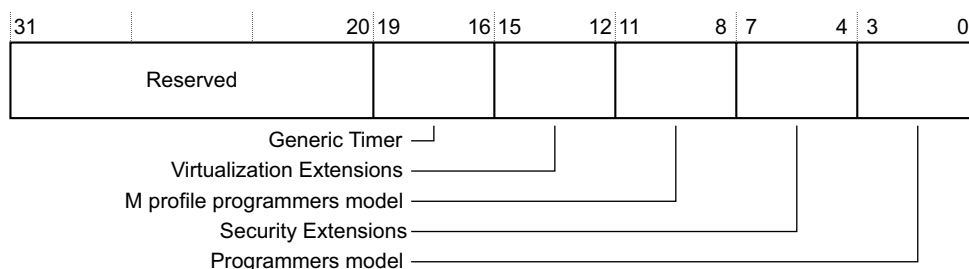
**Usage constraints** The ID\_PFR1 is:

- 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 4-5](#).

[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: 0x1 Processor supports Generic Timer.
[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.
[3:0]	Programmers model	Indicates support for the standard programmers model for ARMv4 and later. Model must support User, FIQ, IRQ, Supervisor, Abort, Undefined and System modes: 0x1 Processor supports the standard programmers model for ARMv4 and later.

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 for the processor.

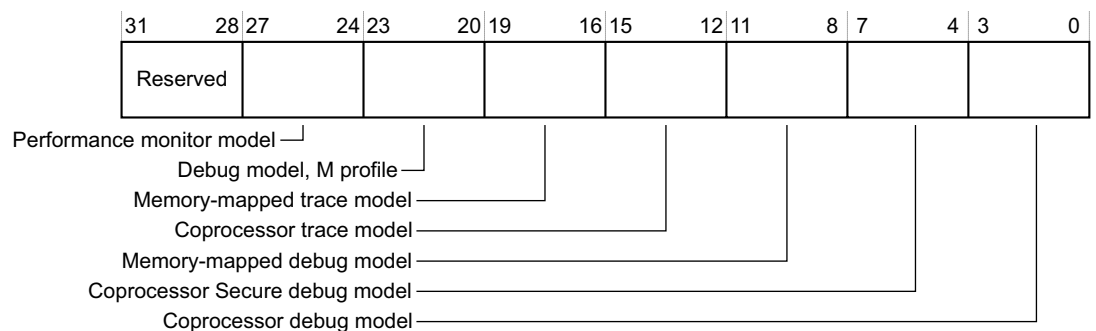
**Usage constraints** The ID\_DFR0 is:

- 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 4-5](#).

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 monitor model	Indicates support for performance monitor model: 0x2 Processor supports <i>Performance Monitor Unit version 2</i> (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.
[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:

<b>Purpose</b>	Provides information about the memory model and memory management support of the processor.
<b>Usage constraints</b>	The ID_MMFR0 is: <ul style="list-style-type: none"> <li>• A read-only register.</li> <li>• Common to the Secure and Non-secure states.</li> <li>• Only accessible from PL1 or higher.</li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 4-2 on page 4-5</a> .

[Figure 4-9 on page 4-35](#) shows the ID\_MMFR0 bit assignments.



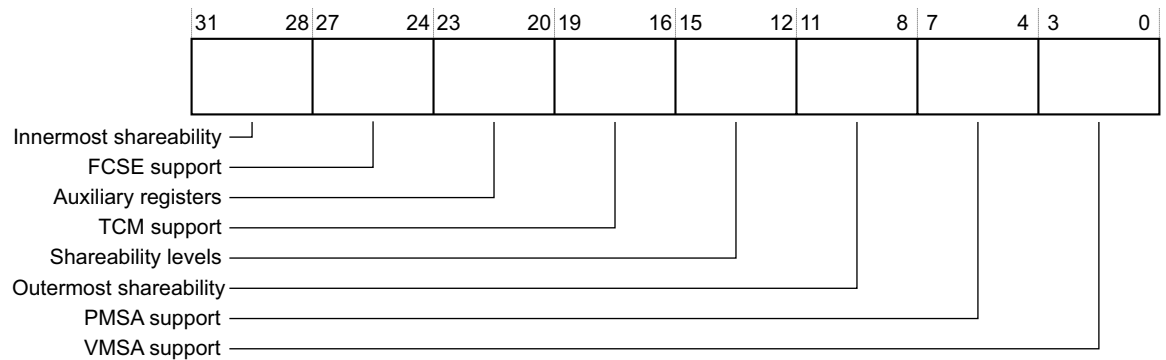


Figure 4-9 ID\_MMFR0 bit assignments

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 <i>Fast Context Switch Extension</i> (FCSE): 0x0 Processor does not support FCSE.
[23:20]	Auxiliary registers	Indicates support for Auxiliary registers: 0x1 Processor supports the Auxiliary Control Register only.
[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 support	Indicates support for a <i>Protected Memory System Architecture</i> (PMSA): 0x0 Processor does not support PMSA.
[3:0]	VMSA support	Indicates support for a <i>Virtual Memory System Architecture</i> (VMSA). 0x5 Processor supports: <ul style="list-style-type: none"> <li>VMSAv7, with support for remapping and the Access flag.</li> <li><i>Privileged Execute Never</i> (PXN) within the first level descriptors.</li> <li>64-bit address translation descriptors.</li> </ul>

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:

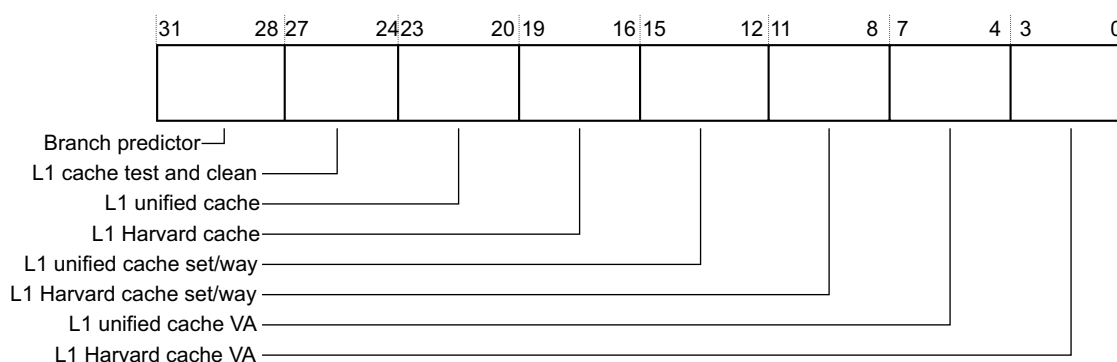
**Purpose** Provides information about the memory model and memory management support of the processor.

- 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 4-5](#).

[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: 0x4 Branch predictor requires no flushing at any time.
[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.
[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.

Table 4-38 ID\_MMFR1 bit assignments (continued)

Bits	Name	Function
[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 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 4-5](#).

[Figure 4-11](#) shows the ID\_MMFR2 bit assignments.

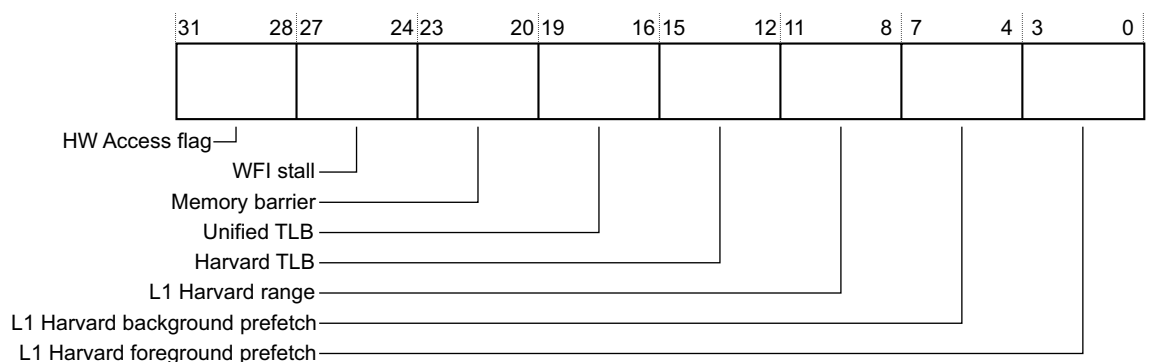


Figure 4-11 ID\_MMFR2 bit assignments

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 <i>Wait For Interrupt</i> (WFI) stalling: 0x1 Processor supports WFI stalling.
[23:20]	Memory barrier	Indicates the supported CP15 memory barrier operations. 0x2 Processor supports: <ul style="list-style-type: none"> <li>• <i>Data Synchronization Barrier</i> (DSB).</li> <li>• <i>Instruction Synchronization Barrier</i> (ISB).</li> <li>• <i>Data Memory Barrier</i> (DMB).</li> </ul>
[19:16]	Unified TLB	Indicates the supported TLB maintenance operations, for a unified TLB implementation. 0x4 Processor supports: <ul style="list-style-type: none"> <li>• Invalidate all entries in the TLB.</li> <li>• Invalidate TLB entry by MVA.</li> <li>• Invalidate TLB entries by ASID match.</li> <li>• Invalidate TLB entries by MVA All ASID.</li> <li>• Invalidate unified Hyp TLB entry by MVA.</li> <li>• Invalidate entire Non-secure Non-Hyp unified TLB.</li> <li>• Invalidate entire Hyp unified TLB.</li> </ul>
[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 prefetch operations, for a Harvard cache implementation: 0x0 Not supported.
[3:0]	L1 Harvard foreground prefetch	Indicates the supported L1 cache foreground prefetch 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 memory model and memory management support of the processor.

**Usage constraints** The ID\_MMFR3 is:

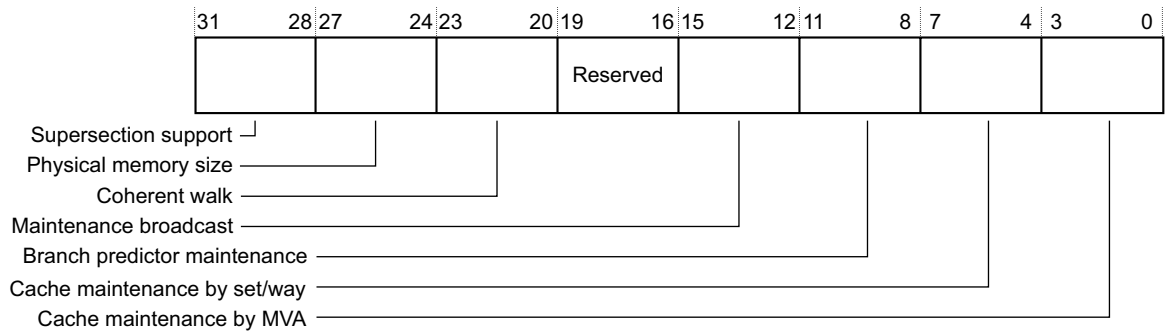
- A read-only register.

- 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 4-5](#).

[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]	Physical memory size	Indicates the size of physical memory supported by the processor caches: 0x2 Processor caches support 40-bit memory address.
[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: <ul style="list-style-type: none"> <li>Invalidate entire branch predictor array.</li> <li>Invalidate branch predictor by MVA.</li> </ul>
[7:4]	Cache maintenance by set/way	Indicates the supported cache maintenance operations by set/way. 0x1 Processor supports: <ul style="list-style-type: none"> <li>Invalidate data cache by set/way.</li> <li>Clean data cache by set/way.</li> <li>Clean and invalidate data cache by set/way.</li> </ul>
[3:0]	Cache maintenance by MVA	Indicates the supported cache maintenance operations by MVA. 0x1 Processor supports: <ul style="list-style-type: none"> <li>Invalidate data cache by MVA.</li> <li>Clean data cache by MVA.</li> <li>Clean and invalidate data cache by MVA.</li> <li>Invalidate instruction cache by MVA.</li> <li>Invalidate all instruction cache entries.</li> </ul>

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 beyond the basic set.

**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 4-5](#).

[Figure 4-13](#) shows the ID\_ISAR0 bit assignments.

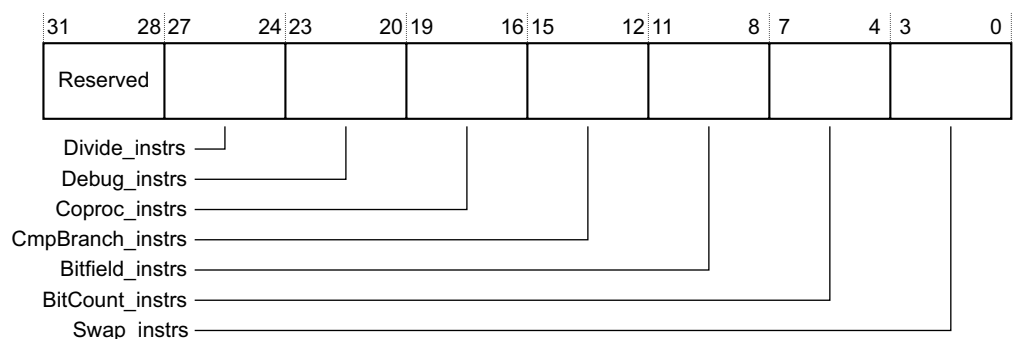
**Figure 4-13 ID\_ISAR0 bit assignments**

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: <ul style="list-style-type: none"> <li>SDIV and UDIV in the Thumb instruction set.</li> <li>SDIV and UDIV in the ARM instruction set.</li> </ul>
[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 bit field 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 SWP and SWPB instructions are 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:

<b>Purpose</b>	Provides information about the instruction set that the processor supports beyond the basic set.
<b>Usage constraints</b>	The ID_ISAR1 is: <ul style="list-style-type: none"> <li>A read-only register.</li> <li>Common to the Secure and Non-secure states.</li> <li>Only accessible from PL1 or higher.</li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 4-2 on page 4-5</a> .

[Figure 4-14 on page 4-42](#) shows the ID\_ISAR1 bit assignments.

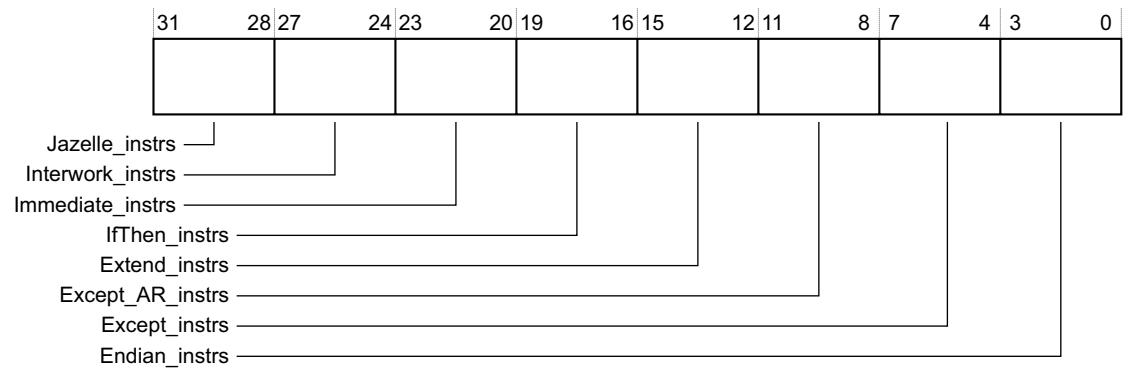


Figure 4-14 ID\_ISAR1 bit assignments

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: <ul style="list-style-type: none"> <li>• BX instruction, and the T bit in the PSR.</li> <li>• BLX instruction, and PC loads have BX-like behavior.</li> <li>• Data-processing instructions in the ARM instruction set with the PC as the destination and the S bit cleared to 0, have BX-like behavior.</li> </ul>
[23:20]	Immediate_instrs	Indicates support for data-processing instructions with long immediates. 0x1 Processor supports: <ul style="list-style-type: none"> <li>• MOV<sub>T</sub> instruction.</li> <li>• MOV instruction encodings with zero-extended 16-bit immediates.</li> <li>• 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.</li> </ul>
[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: <ul style="list-style-type: none"> <li>• SXTB, SXT<sub>H</sub>, UXTB, and UXT<sub>H</sub> instructions.</li> <li>• SXTB16, SXTAB, SXTAB16, SXTA<sub>H</sub>, UXTB16, UXTAB, UXTAB16, and UXTA<sub>H</sub> instructions.</li> </ul>
[11:8]	Except_AR_instrs	Indicates the supported A and R profile exception-handling instructions: 0x1 Processor supports SR5, 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:



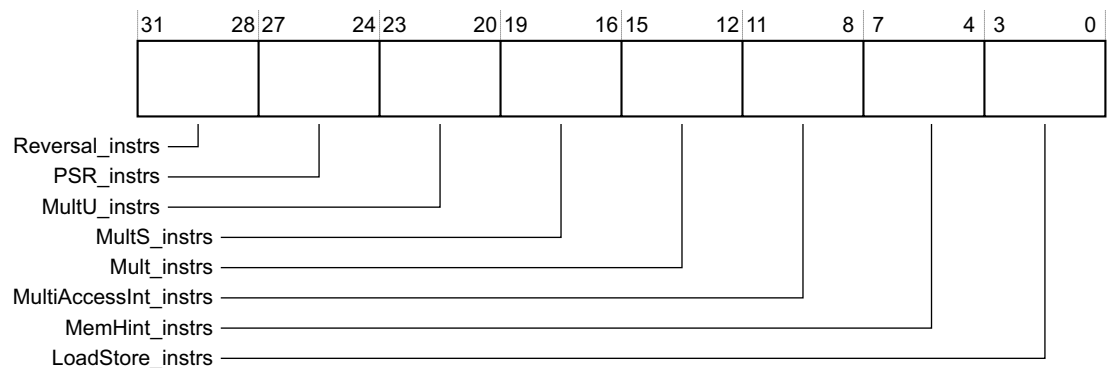
MRC p15, 0, <Rt>, c0, c2, 1 ; Read Instruction Set Attribute Register 1

### 4.3.17 Instruction Set Attribute Register 2

The ID\_ISAR2 characteristics are:

- Purpose** Provides information about the instruction set that the processor supports beyond the basic set.
- 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 4-5](#).

[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 PSR instructions: 0x1 Processor supports MRS and MSR instructions, and the exception return forms of data-processing instructions.  <div> <div> <b>Note</b> </div> <div> The exception return forms of the data-processing instructions are: <ul style="list-style-type: none"> <li>• In the ARM instruction set, data-processing instructions with the PC as the destination and the S bit set. These instructions might be affected by the ISAR4.WithShifts attribute.</li> <li>• In the Thumb instruction set, the SUBS PC, LR, #N instruction.</li> </ul> </div> </div>
[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: <ul style="list-style-type: none"> <li>• SMULL and SMLAL instructions.</li> <li>• SMLABB, SMLABT, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLATB, SMLATT, SMLAWB, SMLAWT, SMULBB, SMULBT, SMULTB, SMULTT, SMULWB, SMULWT instructions, and the Q bit in the PSRs.</li> <li>• SMLAD, SMLADX, SMLALD, SMLALDX, SMLSD, SMLSDX, SMLSLD, SMLSLDX, SMMLA, SMMLAR, SMMLS, SMMLSR, SMMUL, SMMULR, SMUAD, SMUADX, SMUSD, and SMUSDX instructions.</li> </ul>
[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 processor does not support interruptible LDM and STM instructions.
[7:4]	MemHint_instrs	Indicates the supported memory hint instructions: 0x4 Processor supports PLD, PLI, 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:

<b>Purpose</b>	Provides information about the instruction set that the processor supports beyond the basic set.
<b>Usage constraints</b>	The ID_ISAR3 is: <ul style="list-style-type: none"> <li>• A read-only register.</li> <li>• Common to the Secure and Non-secure states.</li> <li>• Only accessible from PL1 or higher.</li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 4-2 on page 4-5</a> .

[Figure 4-16 on page 4-45](#) shows the ID\_ISAR3 bit assignments.

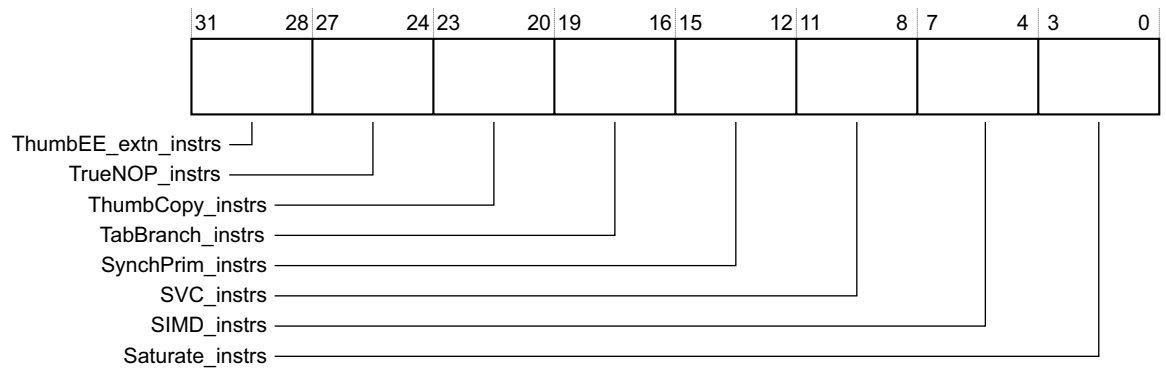


Figure 4-16 ID\_ISAR3 bit assignments

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 <i>Thumb Execution Environment</i> (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	Indicates the supported Synchronization Primitive instructions. 0x2 Processor supports: <ul style="list-style-type: none"> <li>LDREX and STREX instructions.</li> <li>CLREX, LDREXB, LDREXH, STREXB, and STREXH instructions.</li> <li>LDREXD and STREXD instructions.</li> </ul>
[11:8]	SVC_instrs	Indicates the supported SVC instructions: 0x1 Processor supports SVC instruction.
[7:4]	SIMD_instrs	Indicates the supported <i>Single Instruction Multiple Data</i> (SIMD) instructions. 0x3 Processor supports: <ul style="list-style-type: none"> <li>SSAT and USAT instructions, and the Q bit in the PSRs.</li> <li>PKHBT, PKHTB, QADD16, QADD8, QASX, QSUB16, QSUB8, QSAX, SADD16, SADD8, SASX, 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.</li> </ul>
[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:

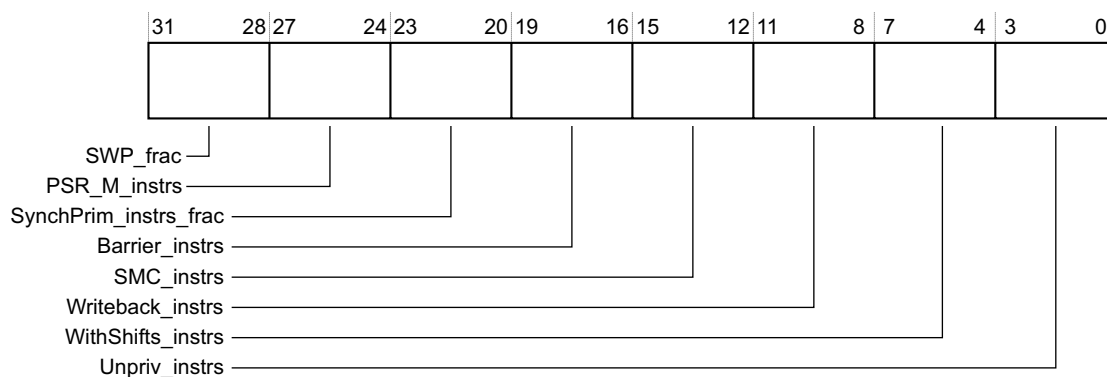
MRC p15, 0, <Rt>, c0, c2, 3 ; Read Instruction Set Attribute Register 3

### 4.3.19 Instruction Set Attribute Register 4

The ID\_ISAR4 characteristics are:

- Purpose** Provides information about the instruction set that the processor supports beyond the basic set.
- 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 4-5](#).

[Figure 4-17](#) 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: 0x0 SWP and SWPB instructions are not supported.
[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 field of ID_ISAR3 to indicate the supported Synchronization Primitive instructions: 0x0 Processor supports: <ul style="list-style-type: none"> <li>• LDREX and STREX instructions.</li> <li>• CLREX, LDREXB, LDREXH, STREXB, and STREXH instructions.</li> <li>• LDREXD and STREXD instructions.</li> </ul>
[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 Write-Back addressing modes: 0x1 Processor supports all Write-Back addressing modes defined in ARMv7 architecture.
[7:4]	WithShifts_instrs	Indicates support for instructions with shifts. 0x4 Processor supports: <ul style="list-style-type: none"> <li>Shifts of loads and stores over the range LSL 0-3.</li> <li>Constant shift options, both on load/store and other instructions.</li> <li>Register-controlled shift options.</li> </ul>
[3:0]	Unpriv_instrs	Indicates the supported unprivileged instructions. 0x2 Processor supports: <ul style="list-style-type: none"> <li>LDRBT, LDRT, STRBT, and STRT instructions.</li> <li>LDRHT, LDRSBT, LDRSHT, and STRHT instructions.</li> </ul>

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:

<b>Purpose</b>	Provides information about the architecture of the caches.
<b>Usage constraints</b>	The CCSIDR is: <ul style="list-style-type: none"> <li>A read-only register.</li> <li>Common to the Secure and Non-secure states.</li> <li>Only accessible from PL1 or higher.</li> </ul>
<b>Configurations</b>	There is one CCSIDR for each cache size and level of cache. The CSSELR determines which CCSIDR is accessible.
<b>Attributes</b>	See the register summary in <a href="#">Table 4-2 on page 4-5</a> .

[Figure 4-18](#) shows the CCSIDR bit assignments.

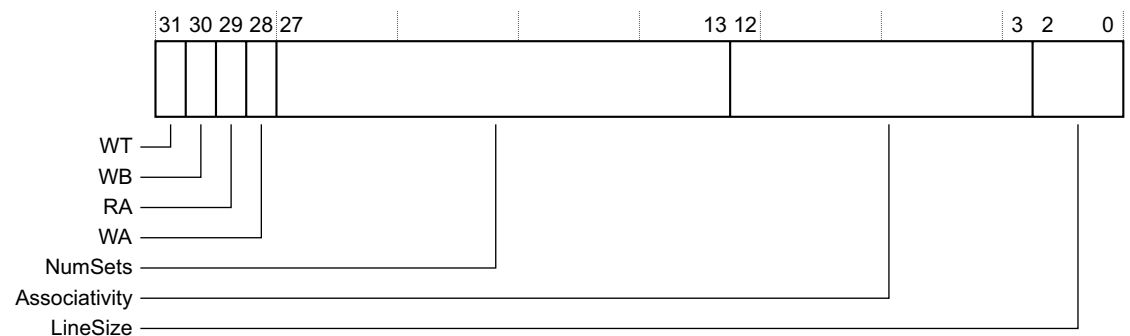
**Figure 4-18 CCSIDR bit assignments**

Table 4-46 shows the CCSIDR bit assignments.

**Table 4-46 CCSIDR bit assignments**

Bits	Name	Function
[31]	WT	Indicates support for Write-Through: <b>0</b> Cache level does not support Write-Through. <b>1</b> Cache level supports Write-Through.
[30]	WB	Indicates support for Write-Back: <b>0</b> Cache level does not support Write-Back. <b>1</b> Cache level supports Write-Back.
[29]	RA	Indicates support for Read-Allocation: <b>0</b> Cache level does not support Read-Allocation. <b>1</b> Cache level supports Read-Allocation.
[28]	WA	Indicates support for Write-Allocation: <b>0</b> Cache level does not support. <b>1</b> Cache level supports Write-Allocation.
[27:13]	NumSets <sup>a</sup>	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 <sup>a</sup>	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: 0b000000001 2-ways. 0b000000011 4-ways. 0b000000111 8-ways. 0b000001111 16-ways.
[2:0]	LineSize <sup>a</sup>	Indicates the $(\log_2(\text{number of words in cache line})) - 2$ : 0b001 8 words per line. 0b010 16 words per line.

a. For more information about encoding, see Table 4-47.

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 CCSIDR encodings**

Cache	CSSELR	Size	Complete register encoding	Register bit field encoding						
				WT	WB	RA	WA	NumSets	Associativity	LineSize
L1 data cache	0x0	32KB	0x700FE01A	0	1	1	1	0x7F	0x3	0x2
L1 instruction cache	0x1	32KB	0x200FE01A	0	0	1	0	0xEF	0x3	0x2
		64KB	0x207FE01A					0xFF	0x3	0x2

Table 4-47 CCSIDR encodings (continued)

Cache	CSSELR	Size	Complete register encoding	Register bit field encoding						
				WT	WB	RA	WA	NumSets	Associativity	LineSize
L2 cache	0x2	256KB	0x701FE07A	0	1	1	1	0xFF	0xF	0x2
		512KB	0x703FE07A					0x1FF		
		1024KB	0x707FE07A					0x3FF		
		2048KB	0x70FFE07A					0x7FF		
		4096KB	0x71FFE07A					0xFFF		
		8192KB	0x73FFE07A					0x1FFF		
Reserved	0x3-0xF	-	-	-	-	-	-	-	-	-

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:

<b>Purpose</b>	Identifies: <ul style="list-style-type: none"> <li>The type of cache, or caches, implemented at each level.</li> <li>The Level of Coherency and Level of Unification for the cache hierarchy.</li> </ul>
<b>Usage constraints</b>	The CLIDR is: <ul style="list-style-type: none"> <li>A read-only register.</li> <li>Common to the Secure and Non-secure states.</li> <li>Only accessible from PL1 or higher.</li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 4-2 on page 4-5</a> .

[Figure 4-19](#) shows the CLIDR bit assignments.

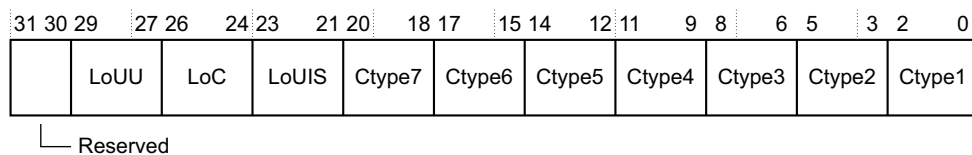


Figure 4-19 CLIDR bit assignments

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: 0x1 L2 cache.
[26:24]	LoC	Indicates the Level of Coherency for the cache hierarchy: 0b010 L2 cache coherency.
[23:21]	LoUIS	Indicates the Level of Unification Inner Shareable for the cache hierarchy: 0b001 L2 cache.
[20:18]	Ctype7	Indicates the type of cache implemented at level 7: 0b000 No cache.
[17:15]	Ctype6	Indicates the type of cache implemented at level 6: 0b000 No cache.
[14:12]	Ctype5	Indicates the type of cache implemented at level 5: 0b000 No cache.
[11:9]	Ctype4	Indicates the type of cache implemented at level 4: 0b000 No cache.
[8:6]	Ctype3	Indicates the type of cache implemented at level 3: 0b000 No cache.
[5:3]	Ctype2	Indicates the type of cache implemented at level 2: 0b100 Unified instruction and data caches.
[2:0]	Ctype1	Indicates the type of cache implemented at level 1: 0b011 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:

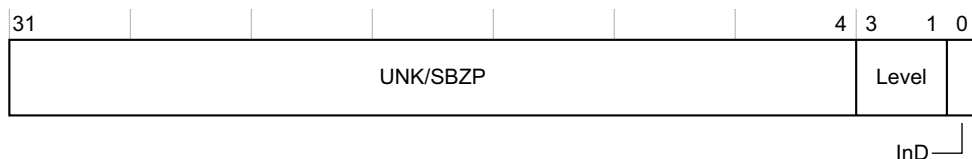
<b>Purpose</b>	Selects the current CCSIDR, see <a href="#">Cache Size ID Register on page 4-47</a> , by specifying: <ul style="list-style-type: none"> <li>The required cache level.</li> <li>The cache type, either instruction or data cache.</li> </ul>
<b>Usage constraints</b>	The CSSELR is: <ul style="list-style-type: none"> <li>A read/write register.</li> <li>Banked for the Secure and Non-secure states.</li> <li>Only accessible from PL1 or higher.</li> </ul>



**Configurations** Available in all configurations.

**Attributes** See the register summary in [Table 4-2 on page 4-5](#).

[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: 0b000 Level 1. 0b001 Level 2. 0b010-0b111 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** Provides the value of the Virtualization Processor ID.

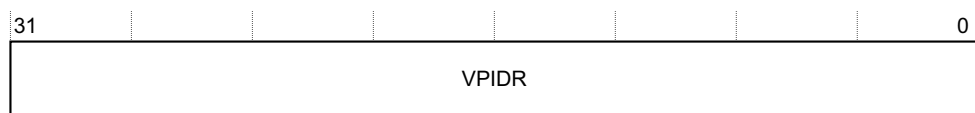
**Usage constraints** The VPIDR is:

- 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 4-5](#).

[Figure 4-21](#) 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 reads of the MIDR from PL1

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

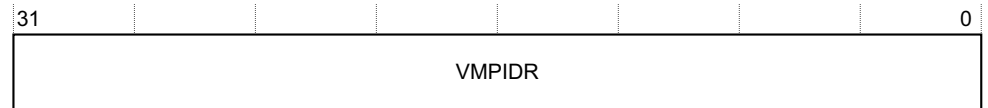
MRC p15, 4, <Rt>, c0, c0, 0; Read Virtualization Processor ID Register  
MCR p15, 4, <Rt>, c0, c0, 0; Write Virtualization Processor ID Register

#### 4.3.26 Virtualization Multiprocessor ID Register

The VMPIDR characteristics are:

- Purpose** Provides the value of the Virtualization Multiprocessor ID.
- 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 4-5](#).

[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 reads of the MPIDR from PL1

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:

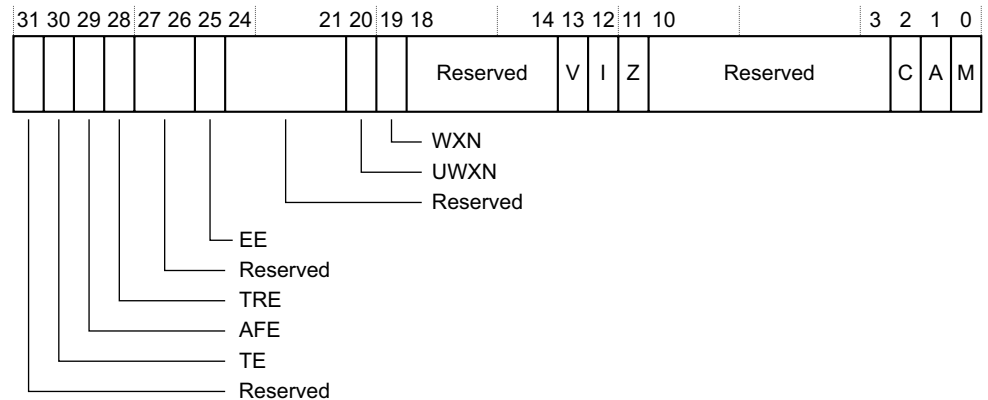
- Purpose** Provides the top-level control of the system, including its memory system.
- Usage constraints** The SCTLR is:
- A read/write register.
  - Banked for Secure and Non-secure states for all implemented bits.

- Is only accessible from PL1 or higher.

**Configurations** Available in all configurations.

**Attributes** See the register summary in [Table 4-3 on page 4-6](#).

[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	Function
[31]	-	Reserved, UNK/SBZP.
[30]	TE	Thumb Exception enable. This bit controls whether exceptions are taken in ARM or Thumb state: <b>0</b> Exceptions, including reset, taken in ARM state. <b>1</b> Exceptions, including reset, taken in Thumb state. The primary input <b>TEINIT[N:0]</b> defines the reset value of the TE bit.
[29]	AFE	Access flag enable bit. This bit enables use of the AP[0] bit in the translation table descriptors as the Access flag. <b>0</b> 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. <b>1</b> In the translation table descriptors, AP[0] is the Access flag. Only the simplified model for access permissions is supported.
[28]	TRE	TEX remap enable bit. 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: <b>0</b> 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. <b>1</b> TEX remap enabled. TEX[2:1] are reassigned for use as bits managed by the operating system. The TEX[0], C and B bits are used to describe the memory region attributes, with the MMU remap registers.
[27:26]	-	Reserved, RAZ/WI.

Table 4-52 SCTLR bit assignments (continued)

Bits	Name	Function
[25]	EE	Exception Endianness bit. The value of this bit defines the value of the CPSR.E bit on entry to an exception vector, including reset. This value also indicates the endianness of the translation table data for translation table lookups: <b>0</b> Little endian. <b>1</b> Big endian. The primary input <b>CFGEND[N:0]</b> defines the reset value of the EE bit.
[24]	-	Reserved, RAZ/WI.
[23:22]	-	Reserved, RAO/SBOP.
[21]	-	Reserved, RAZ/WI.
[20]	UWXN	Unprivileged write permission implies PL1 <i>Execute Never</i> (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. <b>0</b> Regions with unprivileged write permission are not forced to be XN, this is the reset value. <b>1</b> Regions with unprivileged write permission are forced to be XN for accesses from software executing at PL1
[19]	WXN	Write permission implies <i>Execute Never</i> (XN). This bit can be used to require all memory regions with write permissions to be treated as XN. <b>0</b> Regions with write permission are not forced to be XN, this is the reset value. <b>1</b> Regions with write permissions are forced to be XN.
[18]	-	Reserved, RAO/SBOP.
[17]	-	Reserved, RAZ/WI.
[16]	-	Reserved, RAO/SBOP.
[15]	-	Reserved, RAZ/SBZP.
[14]	-	Reserved, RAZ/WI.
[13]	V	Vectors bit. This bit selects the base address of the exception vectors: <b>0</b> Normal exception vectors, base address 0x00000000. Software can remap this base address using the VBAR. <b>1</b> High exception vectors, base address 0xFFFF0000. This base address is never remapped. The primary input <b>VINITHI[N:0]</b> defines the reset value of the V bit.
[12]	I	Instruction cache enable bit. This is a global enable bit for instruction caches: <b>0</b> Instruction caches disabled, this is the reset value. <b>1</b> Instruction caches enabled.
[11]	Z	RAO/WI. Branch prediction enable bit. Branch prediction, also called program flow prediction, is always enabled when the MMU is enabled.
[10]	-	Reserved, RAZ/WI.
[9:7]	-	Reserved, RAZ/SBZP.
[6:3]	-	Reserved, RAO/SBOP.

**Table 4-52 SCTLR bit assignments (continued)**

Bits	Name	Function
[2]	C	Cache enable bit. This is a global enable bit for data and unified caches: <b>0</b> Data and unified caches disabled, this is the reset value. <b>1</b> Data and unified caches enabled.
[1]	A	Alignment bit. This is the enable bit for Alignment fault checking: <b>0</b> Alignment fault checking disabled, this is the reset value. <b>1</b> Alignment fault checking enabled.
[0]	M	Address translation enable bit. This is a global enable bit for the MMU stage 1 address translation: <b>0</b> Address translation disabled, this is the reset value. <b>1</b> Address translation enabled.

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.
- Only accessible from PL1 or higher:
  - Read/write in Secure PL1 mode.
  - In Non-secure, PL1 or PL2 mode:  
 NSACR.NS\_SMP = 0, ACTLR.SMP read-only write ignored.  
 NSACR.NS\_SMP = 1, ACTLR.SMP read/write.  
 NSACR.NS\_ACTLR\_PF\_WRITE = 0, ACTLR.L1PF and ACTLR.L2PF read-only write ignored.  
 NSACR.NS\_ACTLR\_PF\_WRITE = 1, ACTLR.L1PF and ACTLR.L2PF read/write.

**Configurations** Available in all configurations.

**Attributes** See the register summary in [Table 4-3 on page 4-6](#).

[Figure 4-24 on page 4-56](#) shows the ACTLR bit assignments.

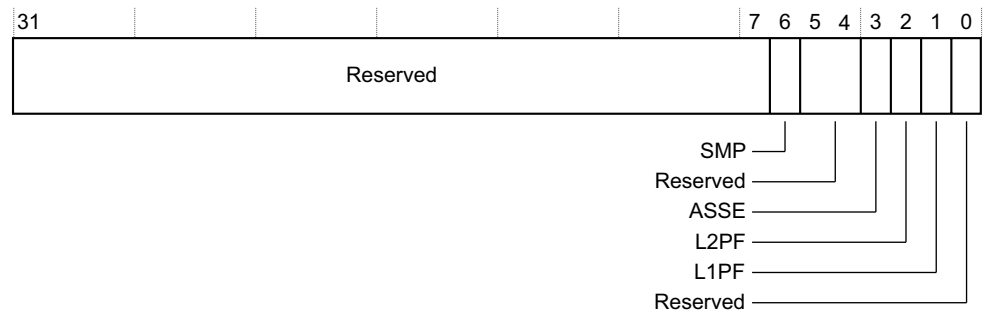


Figure 4-24 ACTLR bit assignments

Table 4-53 shows the ACTLR bit assignments.

Table 4-53 ACTLR bit assignments

Bits	Name	Function
[31:7]	-	Reserved, RAZ/WI.
[6]	SMP	<p>Enables coherent requests to the processor:</p> <p><b>0</b> Disables:</p> <ul style="list-style-type: none"> <li>Broadcast of I\$ and TLB maintenance operations.</li> <li>Synchronisation requests.</li> </ul> <p>Incoming I\$, TLB and Synchronisation requests from other processors are ignored. This is the reset value.</p> <p><b>1</b> Enables:</p> <ul style="list-style-type: none"> <li>Broadcast of I\$ and TLB maintenance operations.</li> <li>Synchronisation requests.</li> </ul> <p>Incoming I\$, TLB and Synchronisation requests from other processors are processed.</p> <p><b>Note</b></p> <p>You must ensure this bit is set to 1 before the caches and MMU are enabled, or any cache and TLB maintenance operations are performed. The only time this bit is set to 0 is during a processor powerdown sequence. See <a href="#">Power management on page 2-17</a>.</p>
[5:4]	-	Reserved, RAZ/WI.
[3]	ASSE	<p>ACE STREX Signaling Enable. When set, ACE CleanUnique transactions originated by a STREX instruction are performed with <b>ARLOCK</b> asserted.</p> <p>This bit is:</p> <ul style="list-style-type: none"> <li>RW in Secure state.</li> <li>RWI in Non-secure state.</li> <li>The reset value is 0.</li> </ul> <p><b>Note</b></p> <p>Do not allow this bit to change during execution of a STREX instruction.</p>

Bits	Name	Function
[2]	L2PF	Enable L2 prefetch: <b>0</b> Disables L2 prefetch. <b>1</b> Enables L2 prefetch. This is the reset value.
[1]	L1PF	Enable L1 prefetch: <b>0</b> Disables L1 prefetch. <b>1</b> Enables L1 prefetch. This is the reset value.
[0]	-	Reserved, RAZ/WI.

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

Figure 4-25 shows the 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 Functionality: <b>0</b> All Advanced SIMD and VFP instructions execute normally. <b>1</b> All Advanced SIMD instructions executed take an Undefined instruction exception. If VFP and Advanced SIMD extensions are not implemented, this bit is UNK/SBZP.
[30:24]	-	Reserved, RAZ/WI.
[23:22]	CP11	Defines the access rights for coprocessor CP11: <b>0b00</b> Access denied. Attempted accesses generate an Undefined Instruction exception. This is the reset value. <b>0b01</b> Access at PL1 or higher only. Attempted accesses in User mode generate an Undefined Instruction exception. <b>0b10</b> Reserved. <b>0b11</b> Full access. If VFP and Advanced SIMD extensions are not implemented, this bit is RAZ/WI.
[21:20]	CP10	Defines the access rights for coprocessor CP10: <b>0b00</b> Access denied. Attempted accesses generate an Undefined Instruction exception. This is the reset value. <b>0b01</b> Access at PL1 or higher only. Attempted accesses generate an Undefined Instruction exception. <b>0b10</b> Reserved. <b>0b11</b> Full access. If VFP and Advanced SIMD extensions are not implemented, this bit 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:

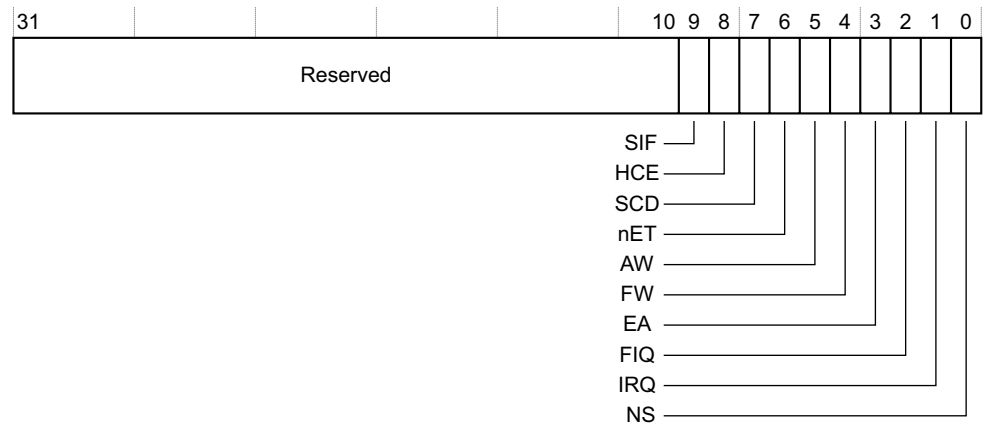
<b>Purpose</b>	Defines the configuration of the current security state. It specifies: <ul style="list-style-type: none"> <li>• The security state of the processor, Secure or Non-secure.</li> <li>• What mode the processor branches to, if an IRQ, FIQ or external abort occurs.</li> <li>• Whether the CPSR.F and CPSR.A bits can be modified when SCR.NS = 1.</li> </ul>
<b>Usage constraints</b>	The SCR is: <ul style="list-style-type: none"> <li>• A read/write register.</li> <li>• A restricted access register that exists only in the Secure state.</li> <li>• Only accessible in Secure PL1 modes.</li> </ul>



**Configurations** Available in all configurations.

**Attributes** See the register summary in [Table 4-3 on page 4-6](#).

[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: <b>0</b> Secure state instruction fetches from Non-secure memory are permitted, this is the reset value. <b>1</b> 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: <b>0</b> The HVC instruction is UNDEFINED in Non-secure PL1 modes, and UNPREDICTABLE in Hyp mode. This is the reset value. <b>1</b> The HVC instruction is enabled in Non-secure PL1 mode, and performs a Hyp Call.
[7]	SCD	Secure Monitor Call disable. This bit causes the SMC instruction to be UNDEFINED in Non-secure state: <b>0</b> The SMC instruction executes normally in Non-secure state, and performs a Secure Monitor Call, this is the reset value. <b>1</b> 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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.
[6]	nET	Not Early Termination. This bit disables early termination of data operations. This bit is not implemented, UNK/SBZP.
[5]	AW	A bit writable. This bit controls whether CPSR.A can be modified in Non-secure state: <ul style="list-style-type: none"> <li>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.</li> <li>This bit, with the HCR.AMO bit, determines whether CPSR.A has any effect on exceptions that are routed to a Non-secure mode.</li> </ul>

Table 4-55 SCR bit assignments (continued)

Bits	Name	Function
[4]	FW	<p>F bit writable. This bit controls whether CPSR.F can be modified in Non-secure state:</p> <ul style="list-style-type: none"> <li>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.</li> <li>This bit, with the HCR.FMO bit, determines whether CPSR.F has any effect on exceptions that are routed to a Non-secure mode.</li> </ul>
[3]	EA	<p>External Abort handler. This bit controls which mode takes external aborts:</p> <p><b>0</b> External aborts taken in Abort mode, this is the reset value.</p> <p><b>1</b> External aborts taken in Monitor mode.</p>
[2]	FIQ	<p>FIQ handler. This bit controls which mode takes FIQ exceptions:</p> <p><b>0</b> FIQs taken in FIQ mode, this is the reset value.</p> <p><b>1</b> FIQs taken in Monitor mode.</p>
[1]	IRQ	<p>IRQ handler. This bit controls which mode takes IRQ exceptions:</p> <p><b>0</b> IRQs taken in IRQ mode, this is the reset value.</p> <p><b>1</b> IRQs taken in Monitor mode.</p>
[0]	NS	<p>Non-secure bit. Except when the processor is in Monitor mode, this bit determines the security state of the processor.</p> <p><b>0</b> Secure, this is the reset value.</p> <p><b>1</b> Non-secure.</p> <p>———— <b>Note</b> ————</p> <p>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.</p> <p>See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information on the NS bit.</p>

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:

<b>Purpose</b>	Defines the Non-secure access permission to coprocessors CP0 to CP13.
<b>Usage constraints</b>	<p>The NSACR is:</p> <ul style="list-style-type: none"> <li>A restricted access register that exists only in the Secure state but can be read from the Non-secure state.</li> <li>Only accessible from PL1 or higher, with access rights that depend on the mode and security state: <ul style="list-style-type: none"> <li>Read/write in Secure PL1 modes.</li> <li>Read-only in Non-secure PL1 and PL2 modes.</li> </ul> </li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 4-3 on page 4-6</a> .

[Figure 4-27 on page 4-61](#) shows the NSACR bit assignments.

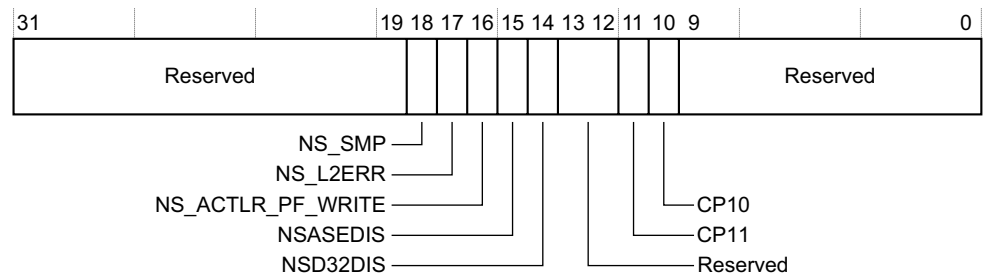


Figure 4-27 NSACR bit assignments

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	Controls write access to the SMP bit of the <i>Auxiliary Control Register</i> (ACTLR) in Non-secure state: <b>0</b> A write to ACTLR in Non-secure state is write-ignored. This is the reset value. <b>1</b> 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	Controls write access to the <i>L2 Extended Control Register</i> (L2ECTLR) in Non-secure state: <b>0</b> A write to L2ECTLR in Non-secure state is ignored. This is the reset value. <b>1</b> A write to L2ECTLR in Non-secure state is accepted.
[16]	NS_ACTLR_PF_WRITE	Controls write access to the ACTLR.L1PF and ACTLR.L2PF registers in Non-secure state: <b>0</b> A write to ACTLR.PF_ENABLE in Non-secure state is ignored. This is the reset value. <b>1</b> A write to ACTLR.PF_ENABLE in Non-secure state is accepted.
[15]	NSASEDIS	Disable Non-secure Advanced SIMD functionality: <b>0</b> This bit has no effect on the ability to write CPACR.ASEDIS, this is the reset value. <b>1</b> When executing in Non-secure state, the CPACR.ASEDIS bit has a fixed value of 1 and writes to it are ignored. If VFP and Advanced SIMD extensions are not implemented, this bit is UNK/SBZP.
[14]	NSD32DIS	Disable the Non-secure use of D16-D31 of the floating-point register file: <b>0</b> This bit has no effect on the ability to write CPACR.D32DIS. This is the reset value.
[13:12]	-	Reserved, RAZ/WI.

**Table 4-56 NSACR bit assignments (continued)**

Bits	Name	Function
[11]	CP11	Non-secure access to coprocessor CP11 enable: <b>0</b> Secure access only. Any attempt to access coprocessor CP11 in Non-secure state results in an Undefined Instruction exception. If the processor is in Non-secure state, the corresponding bits in the CPACR ignore writes and read as 0b00, access denied. This is the reset value. <b>1</b> Secure or Non-secure access. If VFP and Advanced SIMD extensions are not implemented, this bit is RAZ/WI.
[10]	CP10	Non-secure access to coprocessor CP10 enable: <b>0</b> Secure access only. Any attempt to access coprocessor CP10 in Non-secure state results in an Undefined Instruction exception. If the processor is in Non-secure state, the corresponding bits in the CPACR ignore writes and read as 0b00, access denied. This is the reset value. <b>1</b> Secure or Non-secure access. If VFP and Advanced SIMD extensions 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. In Hyp mode this register has access to a subset of SCTLR bits.

**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 4-6](#).

[Figure 4-28](#) shows the HSCTLR bit assignments.

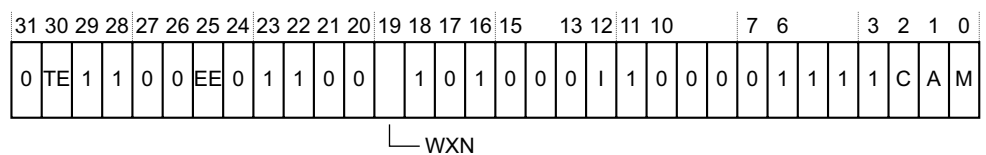
**Figure 4-28 HSCTLR bit assignments**

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. Controls whether exceptions taken in Hyp mode are taken in ARM or Thumb state: <b>0</b> Exceptions taken in ARM state. <b>1</b> Exceptions taken in Thumb state.
[29:28]	-	Reserved, RAO/WI.
[27:26]	-	Reserved, RAZ/WI.
[25]	EE	Exception Endianness 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: <b>0</b> Little endian. <b>1</b> Big endian.
[24]	-	Reserved, RAZ/WI.
[23:22]	-	Reserved, RAO/WI.
[21]	-	Reserved, RAZ/WI.
[20]	-	Reserved, RAZ/WI.
[19]	WXN	Write permission implies <i>Execute Never</i> (XN): <b>0</b> Hyp translations that permit write are not forced to be XN. <b>1</b> Hyp translations that permit write are forced to be XN.
[18]	-	Reserved, RAO/WI.
[17]	-	Reserved, RAZ/WI.
[16]	-	Reserved, RAO/WI.
[15:13]	-	Reserved, RAZ/WI.
[12]	I	Instruction cache enable bit for memory accesses made in Hyp mode: <b>0</b> Instruction caches disabled. <b>1</b> 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	Data and unified cache enable bit for memory accesses made in Hyp mode: <b>0</b> Data and unified caches disabled. <b>1</b> Data and unified caches enabled.
[1]	A	Alignment fault checking enable bit for memory accesses made in Hyp mode: <b>0</b> Alignment fault checking disabled. <b>1</b> Alignment fault checking enabled.
[0]	M	MMU stage 1 address translation enable bit for memory accesses made in Hyp mode: <b>0</b> Address translation disabled. <b>1</b> Address translation 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 always UNK/SBZP in Hyp mode and in Monitor mode when SCR.NS is 1.

### 4.3.34 Hyp Debug Control 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 4-6](#).

[Figure 4-29](#) shows the HDCR bit assignments.

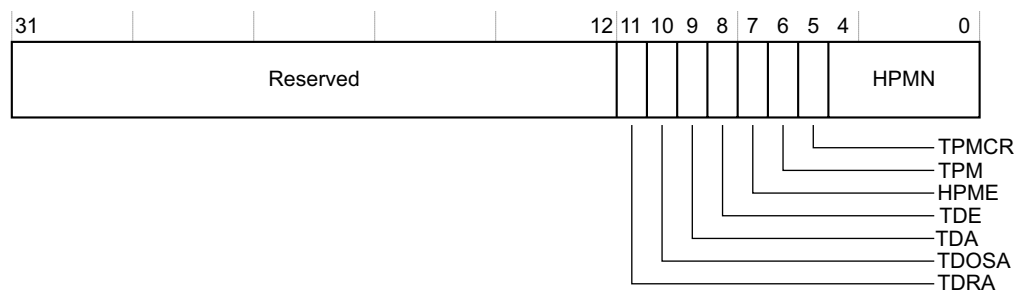
**Figure 4-29 HDCR bit assignments**

Table 4-58 shows the HDCR bit assignments.

**Table 4-58 HDCR bit assignments**

Bits	Name	Function
[31:12]	-	Reserved, UNK/SBZP.
[11]	TDRA	<p>Trap Debug ROM Access:</p> <p><b>0</b> Has no effect on Debug ROM accesses.</p> <p><b>1</b> Trap valid Non-secure Debug ROM accesses to Hyp mode.</p> <p>When this bit is set to 1, any valid Non-secure access to the following registers is trapped to Hyp mode:</p> <ul style="list-style-type: none"> <li>• DBGDRAR.</li> <li>• DBGDSAR.</li> <li>• DBGOSDLR.</li> <li>• DBGPRCR.</li> </ul> <p>If this bit is set to 0 when TDE is set to 1, behavior is UNPREDICTABLE. This bit resets to 0.</p>
[10]	TDOSA	<p>Trap Debug OS-related register Access:</p> <p><b>0</b> Has no effect on accesses to CP14 Debug registers.</p> <p><b>1</b> Trap valid Non-secure accesses to CP14 OS-related Debug registers to Hyp mode.</p> <p>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:</p> <ul style="list-style-type: none"> <li>• DBGOSLSR.</li> <li>• DBGOSLAR.</li> <li>• DBGOSDLR.</li> <li>• DBGPRCR.</li> </ul> <p>If this bit is set to 0 when TDE is set to 1, behavior is UNPREDICTABLE. This bit resets to 0.</p>
[9]	TDA	<p>Trap Debug Access:</p> <p><b>0</b> Has no effect on accesses to CP14 Debug registers.</p> <p><b>1</b> Trap valid Non-secure accesses to CP14 Debug registers to Hyp mode</p> <p>When this bit is set to 1, any valid Non-secure access to the CP14 Debug registers, other than the registers trapped by the TDRA and TDOSA bits, is trapped to Hyp mode.</p> <p>If this bit is set to 0 when TDE is set to 1, behavior is UNPREDICTABLE. This bit resets to 0.</p>
[8]	TDE	<p>Trap Debug Exceptions:</p> <p><b>0</b> Has no effect on Debug exceptions.</p> <p><b>1</b> Trap valid Non-secure Debug exceptions to Hyp mode.</p> <p>When this bit is set to 1:</p> <ul style="list-style-type: none"> <li>• Any Debug exception taken in Non-secure state is trapped to Hyp mode.</li> <li>• The TDRA, TDOSA, and TDA bits must all be set to 1, otherwise behavior is UNPREDICTABLE.</li> </ul> <p>This bit resets to 0.</p>
[7]	HPME	<p>Hypervisor Performance Monitor Enable:</p> <p><b>0</b> Hyp mode performance monitor counters disabled.</p> <p><b>1</b> Hyp mode performance monitor counters enabled.</p> <p>When this bit is set to 1, access to the performance monitors that are reserved for use from Hyp mode is enabled. For more information, see the description of the HPMN field.</p> <p>The reset value of this bit is UNKNOWN.</p>

Table 4-58 HDCR bit assignments (continued)

Bits	Name	Function
[6]	TPM	<p>Trap Performance Monitor accesses:</p> <p><b>0</b> Has no effect on performance monitor accesses.</p> <p><b>1</b> Trap valid Non-secure performance monitor accesses to Hyp mode.</p> <p>When this bit is set to 1, any valid Non-secure access to the Performance Monitor registers is trapped to Hyp mode. This bit resets to 0. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.</p>
[5]	TPMCR	<p>Trap Performance Monitor Control Register accesses:</p> <p><b>0</b> Has no effect on PMCR accesses.</p> <p><b>1</b> Trap valid Non-secure PMCR accesses to Hyp mode.</p> <p>When this bit is set to 1, any valid Non-secure access to the PMCR is trapped to Hyp mode. This bit resets to 0. See the for more information.</p>
[4:0]	HPMN	<p>Hyp Performance Monitor count. Specifies the number of performance monitor counters that are accessible from Non-secure PL1 modes.</p> <p>In Non-secure state, HPMN divides the performance monitor counters.</p> <p>For example, If PMnEVCNTR is performance monitor counter <math>n</math> then, in Non-secure state:</p> <ul style="list-style-type: none"> <li>If <math>n</math> is in the range <math>0 \leq n &lt; \text{HPMN}</math>, the counter is accessible from PL1 and PL2, and from PL0 if unprivileged access to the counters is enabled</li> <li>If <math>n</math> is in the range <math>\text{HPMN} \leq n &lt; \text{PMCR.N}</math>, the counter is accessible only from PL2. The HPME bit enables access to the counters in this range.</li> </ul> <p>Behavior of the Performance Monitors counters is UNPREDICTABLE if this field is set to a value greater than PMCR.N.</p> <p>This field resets to 0x6, which is the value of PMCR.N.</p>

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

MRC p15, 4, <Rt>, c1, c1, 1; Read Hyp Debug Control Register  
MCR p15, 4, <Rt>, c1, c1, 1; Write Hyp Debug Control 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 lower, to functions provided coprocessors other than CP14 and CP15. The HCPTR also controls the access to floating-point and Advanced SIMD functionality from Hyp mode.

#### ————— Note —————

Accesses to floating-point and Advanced SIMD functionality from Hyp mode:

- Are not affected by settings in the CPACR. See [Coprocessor Access Control Register on page 4-57](#).
- Are affected by settings in the NSACR. See [Non-secure Access Control Register on page 4-60](#). The NSACR settings take precedence over the HCPTR settings. See the Usage constraints for more information.

**Usage constraints** The HCPTR is:

- A read/write register.

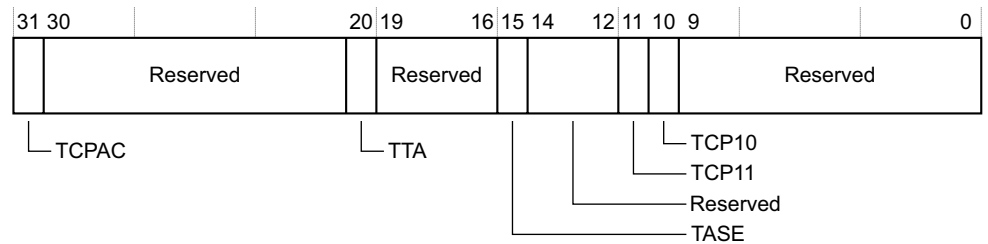


- Only accessible from Hyp mode or from Monitor mode when SCR.NS is 1.
- If a bit in the NSACR prohibits a Non-secure access, then the corresponding bit in the HCPTR behaves as RAO/WI for Non-secure accesses. See the bit descriptions for more information.

**Configurations** Available in all configurations.

**Attributes** See the register summary in [Table 4-3 on page 4-6](#).

[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: <b>0</b> Has no effect on CPACR accesses. <b>1</b> 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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.
[30:21]	-	Reserved, UNK/SBZP.
[20]	TTA	Trap Trace Access, RAZ/WI.
[19:16]	-	Reserved, UNK/SBZP.
[15]	TASE	Trap Advanced SIMD Extension: <b>0</b> 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 <hr/> <b>Note</b> This bit value has no effect on possible use of the Advanced SIMD functionality from Non-secure PL1 and PL0 modes. <hr/> <b>1</b> 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: <ul style="list-style-type: none"> <li>• A Non-secure PL1 or PL0 mode is trapped to Hyp mode.</li> <li>• Hyp mode generates an Undefined Instruction exception, taken in Hyp mode.</li> </ul> If VFP and Advanced SIMD extensions 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	<p>Trap coprocessor CP11:</p> <p><b>0</b> If NSACR.CP11 is set to 1, then Hyp mode can access CP11, regardless of the value of CPACR.CP11.</p> <p style="text-align: center;"><b>Note</b></p> <p>This bit value has no effect on possible use of CP11 from Non-secure PL1 and PL0 modes.</p> <p><b>1</b> Trap valid Non-secure accesses to CP11 to Hyp mode. When TCP11 is set to 1, any otherwise-valid access to CP11 from:</p> <ul style="list-style-type: none"> <li>• A Non-secure PL1 or PL0 mode is trapped to Hyp mode.</li> <li>• Hyp mode generates an Undefined Instruction exception, taken in Hyp mode.</li> </ul> <p>If VFP and Advanced SIMD extensions are not implemented, this bit is RAO/WI. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.</p>
[10]	TCP10	<p>Trap coprocessor CP10:</p> <p><b>0</b> If NSACR.CP10 is set to 1, then Hyp mode can access CP10, regardless of the value of CPACR.CP10.</p> <p style="text-align: center;"><b>Note</b></p> <p>This bit value has no effect on possible use of CP10 from Non-secure PL1 and PL0 modes.</p> <p><b>1</b> Trap valid Non-secure accesses to CP10 to Hyp mode. When TCP10 is set to 1, any otherwise-valid access to CP10 from:</p> <ul style="list-style-type: none"> <li>• a Non-secure PL1 or PL0 mode is trapped to Hyp mode</li> <li>• Hyp mode generates an Undefined Instruction exception, taken in Hyp mode.</li> </ul> <p>If VFP and Advanced SIMD extensions are not implemented, this bit is RAO/WI. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.</p>
[9:0]	-	Reserved, RAO/WI.

**Note**

If the values of the TCP11 and TCP10 fields are not the same, the behavior is UNPREDICTABLE.

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 always UNK/SBZP in Hyp mode and in Monitor mode when SCR.NS is 1.

### 4.3.37 Hyp Translation Control Register

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

### 4.3.38 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.
  - Accessible from PL1 or higher.

**Configurations** Available in all configurations.

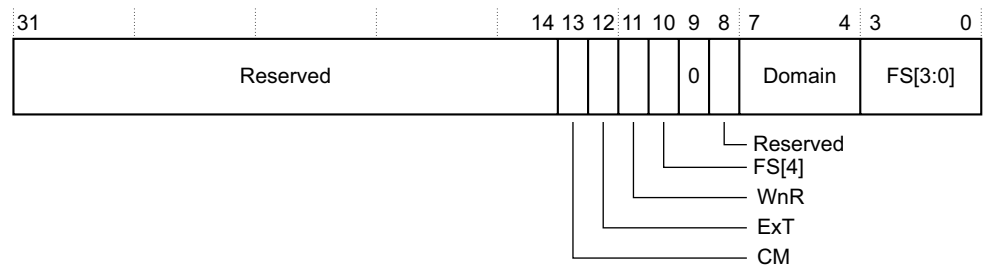
**Attributes** See the register summary in [Table 4-6 on page 4-7](#).

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

- [DFSR when using the Short-descriptor translation table format](#)
- [DFSR when using the Long-descriptor translation table format on page 4-70](#).

### DFSR 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, indicates whether a cache maintenance operation generated the fault: <b>0</b> Abort not caused by a cache maintenance operation. <b>1</b> Abort caused by a cache maintenance operation.
[12]	ExT	External abort type. Indicates whether an AXI Decode or Slave error caused an abort: <b>0</b> External abort marked as DECERR. <b>1</b> External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11]	WnR	Write not Read bit. Indicates whether the abort was caused by a write or a read access: <b>0</b> Abort caused by a read access. <b>1</b> 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.

**Table 4-60 DFSR bit assignments for Short-descriptor translation table format (continued)**

Bits	Name	Function
[9]	LPAE	On taking a Data Abort exception, this bit is set to 0 to indicate use of the Short-descriptor translation table formats. Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation. Unless the register has been updated to report a fault, a subsequent read of the register returns the value written to it.
[8]	-	Reserved, UNK/SBZP.
[7:4]	Domain	Specifies which of the 16 domains, D15-D0, was being accessed when a data fault occurred. For permission faults that generate Data Abort exception, this field is UNKNOWN. ARMv7 deprecates any use of the domain field in the DFSR.
[3:0]	FS[3:0]	Fault Status bits. This field indicates the type of exception generated, any encoding not listed is reserved: <div> <div>0b00001</div> <div>Alignment fault.</div> </div> <div> <div>0b00010</div> <div>Debug event.</div> </div> <div> <div>0b00011</div> <div>Access flag fault, section.</div> </div> <div> <div>0b00101</div> <div>Translation fault, section.</div> </div> <div> <div>0b00110</div> <div>Access flag fault, page.</div> </div> <div> <div>0b00111</div> <div>Translation fault, page.</div> </div> <div> <div>0b01001</div> <div>Domain fault, section.</div> </div> <div> <div>0b01011</div> <div>Domain fault, page.</div> </div> <div> <div>0b01100</div> <div>Synchronous external abort on translation table walk, 1st level.</div> </div> <div> <div>0b01101</div> <div>Permission fault, section.</div> </div> <div> <div>0b01110</div> <div>Synchronous external abort on translation table walk, 2nd level.</div> </div> <div> <div>0b01111</div> <div>Permission fault, page.</div> </div> <div> <div>0b10000</div> <div>TLB conflict abort.</div> </div> <div> <div>0b10110</div> <div>Asynchronous external abort.</div> </div>
<p><b>Note</b></p> <p>See DFSR.10 for FS[4].</p>		

### DFSR when using the Long-descriptor translation table format

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

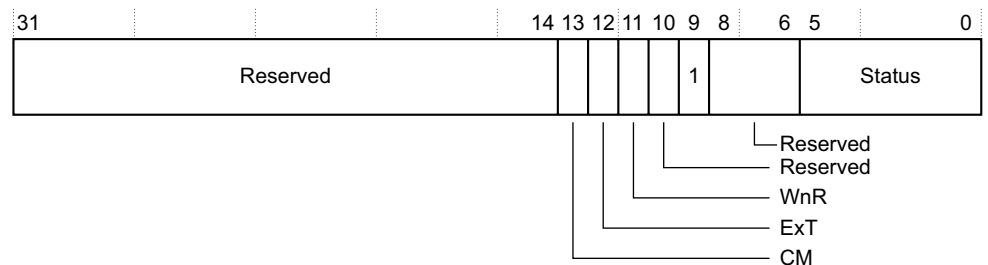
**Figure 4-32 DFSR bit assignments for Long-descriptor translation table format**

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, indicates whether a cache maintenance operation generated the fault: <b>0</b> Abort not caused by a cache maintenance operation. <b>1</b> Abort caused by a cache maintenance operation.
[12]	ExT	External abort type. Indicates whether an AXI Decode or Slave error caused an abort: <b>0</b> External abort marked as DECERR. <b>1</b> External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11]	WnR	Write not Read bit. Indicates whether the abort was caused by a write or a read access: <b>0</b> Abort caused by a read access. <b>1</b> 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.
[9]	LPAE	On taking a Data Abort exception, this bit is set to 1 to indicate use of the Long-descriptor translation table formats. Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation. Unless the register has been updated to report a fault, a subsequent read of the register returns the value written to it.
[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. 0b0001LL Translation fault, LL bits indicate level. 0b0010LL Access fault flag, LL bits indicate level. 0b0011LL Permission fault, LL bits indicate level. 0b011000 TLB conflict abort. 0b010001 Asynchronous external abort. 0b0101LL Synchronous external abort on translation table walk, LL bits indicate level. 0b100001 Alignment fault. 0b100010 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**

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

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

### 4.3.39 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.

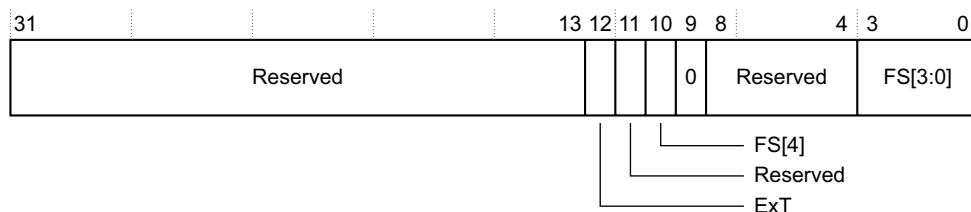
**Attributes** See the register summary in [Table 4-6 on page 4-7](#).

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

- [IFSR when using the Short-descriptor translation table format](#)
- [IFSR when using the Long-descriptor translation table format on page 4-73](#).

#### IFSR 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 Long-descriptor translation table format**

Bits	Name	Function
[31:13]	-	Reserved, UNK/SBZP.
[12]	ExT	External abort type. Indicates whether an AXI Decode or Slave error caused an abort: <b>0</b> External abort marked as DECERR. <b>1</b> 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 Long-descriptor translation table format (continued)**

Bits	Name	Function
[9]	LPAE	On taking a Data Abort exception, this bit is set to 1 to indicate use of the Long-descriptor translation table formats.  Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation. Unless the register has been updated to report a fault, a subsequent read of the register returns the value written to it.
[8:4]	-	Reserved, UNK/SBZP.
[3:0]	FS[3:0]	Fault Status bits. Indicates the type of exception generated. Any encoding not listed is reserved. <div> <div>0b00010</div> <div>Debug event</div> </div> <div> <div>0b00011</div> <div>Access flag fault, section.</div> </div> <div> <div>0b00101</div> <div>Translation fault, section.</div> </div> <div> <div>0b00110</div> <div>Access flag fault, page.</div> </div> <div> <div>0b00111</div> <div>Translation fault, page.</div> </div> <div> <div>0b01000</div> <div>Synchronous external abort, non-translation.</div> </div> <div> <div>0b01001</div> <div>Domain fault, section.</div> </div> <div> <div>0b01011</div> <div>Domain fault, page.</div> </div> <div> <div>0b01100</div> <div>Synchronous external abort on translation table walk, 1st level.</div> </div> <div> <div>0b01101</div> <div>Permission Fault, Section.</div> </div> <div> <div>0b01110</div> <div>Synchronous external abort on translation table walk, 2nd level.</div> </div> <div> <div>0b01111</div> <div>Permission fault, page.</div> </div> <div> <div>0b10000</div> <div>TLB conflict abort.</div> </div>
<p style="text-align: center;"><b>Note</b></p> <p>See IFSR.10 for FS[4].</p>		

### IFSR when using the Long-descriptor translation table format

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

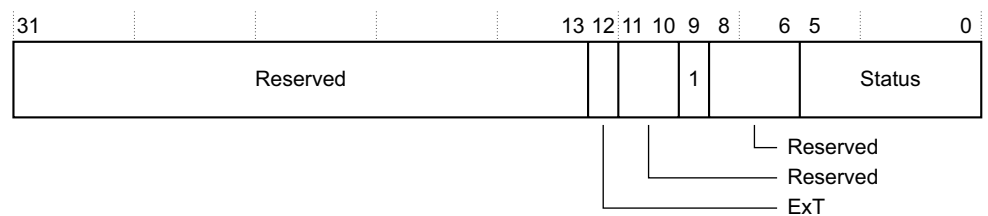
**Figure 4-34 IFSR bit assignments for Long-descriptor translation table format**

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. Indicates whether an AXI Decode or Slave error caused an abort: <b>0</b> External abort marked as DECERR <b>1</b> External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11:10]	-	Reserved, UNK/SBZP.
[9]	LPAE	On taking a Data Abort exception, this bit is set to 1 to indicate use of the Long-descriptor translation table formats. Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation. Unless the register has been updated to report a fault, a subsequent read of the register returns the value written to it.
[8:6]	-	Reserved, UNK/SBZP.
[5:0]	Status	Fault Status bits. Indicates the type of exception generated. Any encoding not listed is reserved. 0b0001LL Translation fault, LL bits indicate level. 0b0010LL Access fault flag, LL bits indicate level. 0b0011LL Permission fault, LL bits indicate level. 0b0101LL External fault on Translation table walk, LL bits indicate level. 0b010000 Synchronous external abort. 0b100010 Debug event. 0b110000 TLB conflict abort.

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**

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

#### Note

If a Data Abort exception is generated by an instruction cache maintenance operation when the Long-descriptor translation table format is selected, the fault is reported as a Cache Maintenance fault in the DFSR or HSR with the appropriate Fault Status code. For such exceptions reported in the DFSR, the corresponding IFSR is UNKNOWN.

To access the IFSR, 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



#### 4.3.40 Auxiliary Data Fault Status Register

The processor does not implement ADFSR, so this register is always UNK/SBZP.

#### 4.3.41 Auxiliary Instruction Fault Status Register

The processor does not implement AIFSR, so this register is always UNK/SBZP.

#### 4.3.42 Hyp Auxiliary Data Fault Status Syndrome Register

The processor does not implement HADFSR, so this register is always UNK/SBZP.

#### 4.3.43 Hyp Auxiliary Instruction Fault Status Syndrome Register

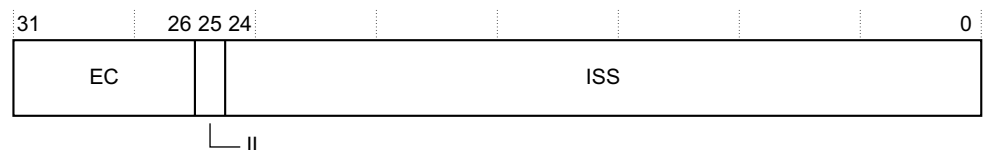
The processor does not implement HAIFSR, so this register is always UNK/SBZP.

#### 4.3.44 Hyp Syndrome Register

The HSR characteristics are:

<b>Purpose</b>	Holds syndrome information for an exception taken in Hyp mode.
<b>Usage constraints</b>	<p>The HSR is:</p> <ul style="list-style-type: none"> <li>• A read/write register.</li> <li>• Only accessible from Hyp mode or from Monitor mode when SCR.NS is 1.</li> <li>• UNKNOWN when executing in Non-secure modes other than Hyp mode.</li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 4-26 on page 4-23</a> .

[Figure 4-35](#) shows the HSR bit assignments.



**Figure 4-35 HSR bit assignments**

[Table 4-66](#) shows the HSR bit assignments.

**Table 4-66 HSR bit assignments**

Bits	Name	Function
[31:26]	EC	Exception class. The exception class for the exception that is taken in Hyp mode. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.
[25]	IL	Instruction length. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.
[24:0]	ISS	Instruction specific syndrome. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information. The interpretation of this field depends on the value of the EC field. See <a href="#">Encoding of ISS[24:20] when HSR[31:30] is 0b00 on page 4-76</a> .

### 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.

#### 4.3.45 Physical Address Register

The processor does not use any implementation-defined bits in the 32-bit format or 64-bit format PAR. Bit[8] is UNK/SBZP. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

#### 4.3.46 Data Cache Clean and Invalidate All

<b>Purpose</b>	Clean and invalidate the selected level of data or unified cache.
----------------	---

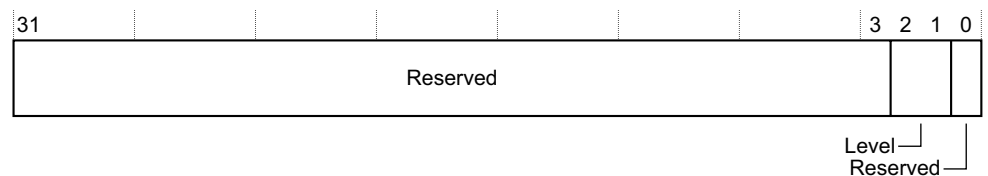
**Usage constraints** The DCCIALL is:

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

<b>Configurations</b>	Available in all configurations.
-----------------------	----------------------------------

**Attributes** See the register summary in [Table 4-14 on page 4-14](#).

Figure 4-36 shows the DCCIALL bit assignments.



**Figure 4-36 DCCIALL bit assignments**

Table 4-67 shows the DCCIALL bit assignments.

**Table 4-67 DCCIALL bit assignments**

Bits	Name	Function
[31:3]	-	Reserved, SBZ.
[2:1]	Level	Cache level: 0b00 Level 1 cache. 0b01 Level 2 cache. 0b10 Reserved. 0b11 Reserved.
[0]	-	Reserved, SBZ.

To perform the DCCIALL operation write to the CP15 register:

MCR p15, 1, <Rt>, c15, c14, 0; DCCIALL Data or unified cache clean and invalidate all

#### 4.3.47 L2 Control Register

The L2CTLR characteristics are:

**Purpose** Provides information about the implemented options of the multiprocessor device.

**Usage constraints** The L2CTLR is:

- A read/write register.
- Common to all processors in the multiprocessor device.
- 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-undefined 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 ACE or ACP requests.

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 4-6 on page 4-7](#).

Figure 4-37 shows the L2CTLR bit assignments.

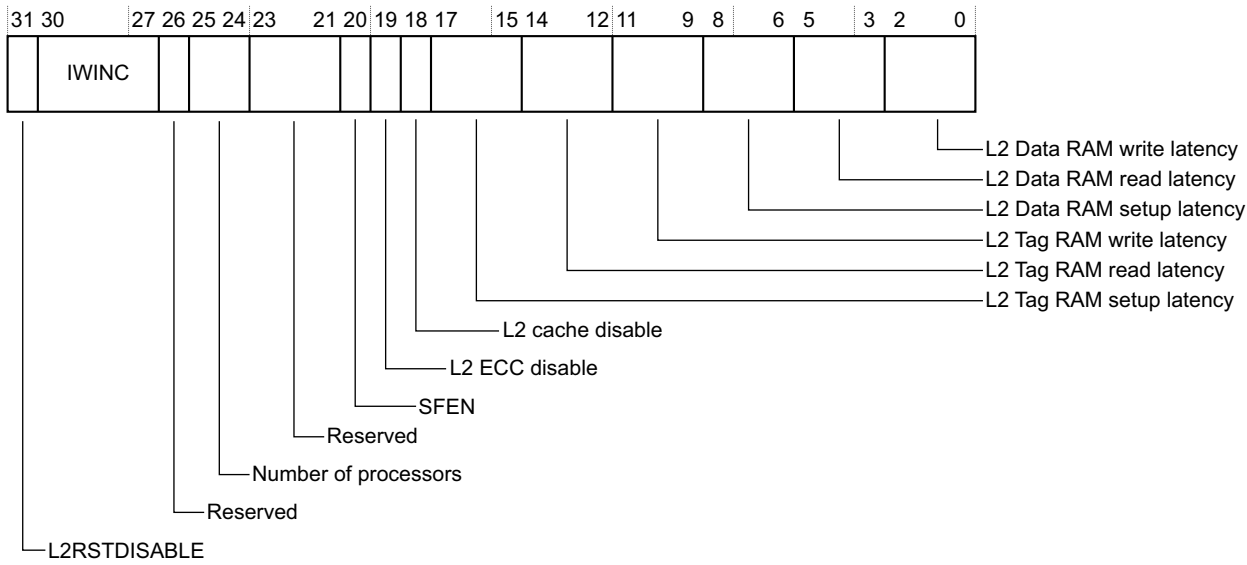


Figure 4-37 L2CTLR bit assignments

Table 4-68 shows the L2CTLR bit assignments.

**Table 4-68 L2CTLR bit assignments**

Bits	Name	Function																								
[31]	L2RSTDISABLE	Monitors the L2 hardware reset disable pin, <b>L2RSTDISABLE</b> :  <div><div>0</div><div>L2 valid RAM contents are reset by hardware.</div><div>1</div><div>L2 valid RAM contents are not reset by hardware.</div></div> This bit is read-only and the reset value is determined by the primary input, <b>L2RSTDISABLE</b> .																								
[30:27]	IWINC	<p>L2 cache clean and clean and invalidate all operations loop on all ways and on all sets. Each set is referenced by an index. These operations evict cache lines marked as dirty. This corresponds to the cache cleaning algorithm specified in the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>.</p> <p>The IWINC register controls how the index increments. The reset value 0b0000 gives an increment of 1, this evicts dirty lines on index 0,1,2,3,4,5, ... (modulo the total number of sets). Setting IWINC to 0b0100 gives an increment value 15, which evicts dirty lines on index 0,15,30,45, ... (modulo the total number of sets).</p> <table><tr><td>0b0000</td><td>1. This is the reset value.</td></tr><tr><td>0b0001</td><td>1.</td></tr><tr><td>0b0010</td><td>3.</td></tr><tr><td>0b0011</td><td>7.</td></tr><tr><td>0b0100</td><td>15.</td></tr><tr><td>0b0101</td><td>31.</td></tr><tr><td>0b0110</td><td>63.</td></tr><tr><td>... n</td><td>(2^n)-1.</td></tr><tr><td>0b1100</td><td>4095.</td></tr><tr><td>0b1101</td><td>8191.</td></tr><tr><td>0b1110</td><td>8191.</td></tr><tr><td>0b1111</td><td>8191.</td></tr></table> <p>You can program the IWINC field to optimize cache flush speed for system configuration of address decode supporting a complex multi-channel multi-rank memory system.</p>	0b0000	1. This is the reset value.	0b0001	1.	0b0010	3.	0b0011	7.	0b0100	15.	0b0101	31.	0b0110	63.	... n	(2^n)-1.	0b1100	4095.	0b1101	8191.	0b1110	8191.	0b1111	8191.
0b0000	1. This is the reset value.																									
0b0001	1.																									
0b0010	3.																									
0b0011	7.																									
0b0100	15.																									
0b0101	31.																									
0b0110	63.																									
... n	(2^n)-1.																									
0b1100	4095.																									
0b1101	8191.																									
0b1110	8191.																									
0b1111	8191.																									
[26]	-	Reserved, RAZ/WI.																								
[25:24]	Number of processors	<p>Number of processors present:</p> <table><tr><td>0b00</td><td>One processor, Processor 0.</td></tr><tr><td>0b01</td><td>Two processors, Processor 0 and Processor 1.</td></tr><tr><td>0b10</td><td>Three processors, Processor 0, Processor 1, and Processor 2.</td></tr><tr><td>0b11</td><td>Four processors, Processor 0, Processor 1, Processor 2, and Processor 3.</td></tr></table> <p>These bits are read-only and the reset value of this field is set to the number of processors present in the implementation.</p>	0b00	One processor, Processor 0.	0b01	Two processors, Processor 0 and Processor 1.	0b10	Three processors, Processor 0, Processor 1, and Processor 2.	0b11	Four processors, Processor 0, Processor 1, Processor 2, and Processor 3.																
0b00	One processor, Processor 0.																									
0b01	Two processors, Processor 0 and Processor 1.																									
0b10	Three processors, Processor 0, Processor 1, and Processor 2.																									
0b11	Four processors, Processor 0, Processor 1, Processor 2, and Processor 3.																									
[23:21]	-	Reserved, RAZ/WI.																								

Table 4-68 L2CTLR bit assignments (continued)

Bits	Name	Function
[20]	SFEN	<p>Snoop Filter Enable. When set, the processor generates ACE messages to support a snoop filter within the ACE interconnect. ARM recommends that this bit is not set when a snoop filter is not present.</p> <p>This bit is:</p> <ul style="list-style-type: none"> <li>• RW in Secure state.</li> <li>• RWI in Non-secure state.</li> <li>• The reset value is 0.</li> </ul> <p>———— <b>Note</b> ————</p> <p>Set this bit before caches and MMU are enabled.</p>
[19]	L2 ECC disable <sup>a</sup>	<p>L2 ECC disable error detection and correction.</p> <p><b>0</b> L2 ECC is enabled. This is the reset state.</p> <p><b>1</b> L2 ECC is disabled.</p>
[18]	L2 cache disable <sup>b</sup>	<p>L2 cache disable</p> <p><b>0</b> L2 cache is enabled. This is the reset state.</p> <p><b>1</b> L2 cache is disabled.</p>
[17:15]	Tag RAM setup latency	<p>L2 tag RAM setup latency:</p> <p>0b000 - 0b111 1 cycle to 8 cycles.</p> <p>The reset value is 0b111.</p>
[14:12]	Tag RAM read latency	<p>L2 tag RAM read latency:</p> <p>0b000 - 0b111 1 cycle to 8 cycles.</p> <p>The reset value is 0b111.</p>
[11:9]	Tag RAM write latency	<p>L2 tag RAM write latency:</p> <p>0b000 - 0b111 1 cycle to 8 cycles.</p> <p>The reset value is 0b111.</p>
[8:6]	Data RAM setup latency	<p>L2 data RAM setup latency:</p> <p>0b000 - 0b111 1 cycle to 8 cycles.</p> <p>The reset value is 0b111.</p>
[5:3]	Data RAM read latency	<p>L2 data RAM read latency:</p> <p>0b000 - 0b111 1 cycle to 8 cycles.</p> <p>The reset value is 0b111.</p>
[2:0]	Data RAM write latency	<p>L2 data RAM write latency:</p> <p>0b001 - 0b111 1 to 8 cycles.</p> <p>The reset value is 0b111.</p>

a. L2 ECC is an implementation option.

b. The L2 cache must never be disabled in a multi-cluster configuration.

To access the L2CTLR, read the CP15 register with:

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

### 4.3.48 L2 Extended Control Register

The L2ECTLR characteristics are:

**Purpose** Provides additional control options for the L2 memory system.

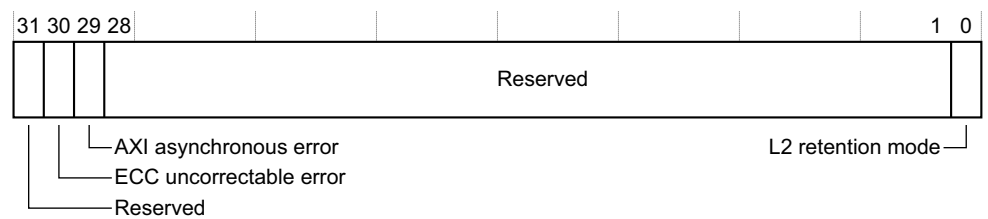
**Usage constraints** The L2ECTLR is:

- A read/write register.
- Common to the Secure and Non-secure states.
- Common to all the processors in the multiprocessor device.
- 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.

**Configurations** Available in all configurations.

**Attributes** See the register summary in [Table 4-10 on page 4-11](#).

[Figure 4-38](#) shows the L2ECTLR bit assignments.



**Figure 4-38 L2ECTLR bit assignments**

[Table 4-69](#) shows the L2ECTLR bit assignments.

**Table 4-69 L2ECTLR bit assignments**

Bits	Name	Function
[31]	-	Reserved, RAZ/WI.
[30]	ECC uncorrectable error	ECC uncorrectable error indication: <b>0</b> No pending ECC error. This is the reset value. <b>1</b> An uncorrectable ECC error has occurred. A write of 0 clears this bit. A write of 1 is ignored.
[29]	AXI asynchronous error	AXI asynchronous error indication: <b>0</b> No pending AXI asynchronous error. This is the reset value. <b>1</b> AXI asynchronous error has occurred. A write of 0 clears this bit. A write of 1 is ignored.
[28:1]	-	Reserved, RAZ/WI.
[0]	L2 dynamic RAM retention mode	Disable L2 dynamic RAM retention: <b>0</b> L2 dynamic RAM retention enabled. This is the reset value. <b>1</b> L2 dynamic RAM retention disabled.

To access the L2ECTL, 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

#### 4.3.49 SCU Control Register

The SCUCTLR characteristics are:

**Purpose** Provides additional control and information on the cluster.

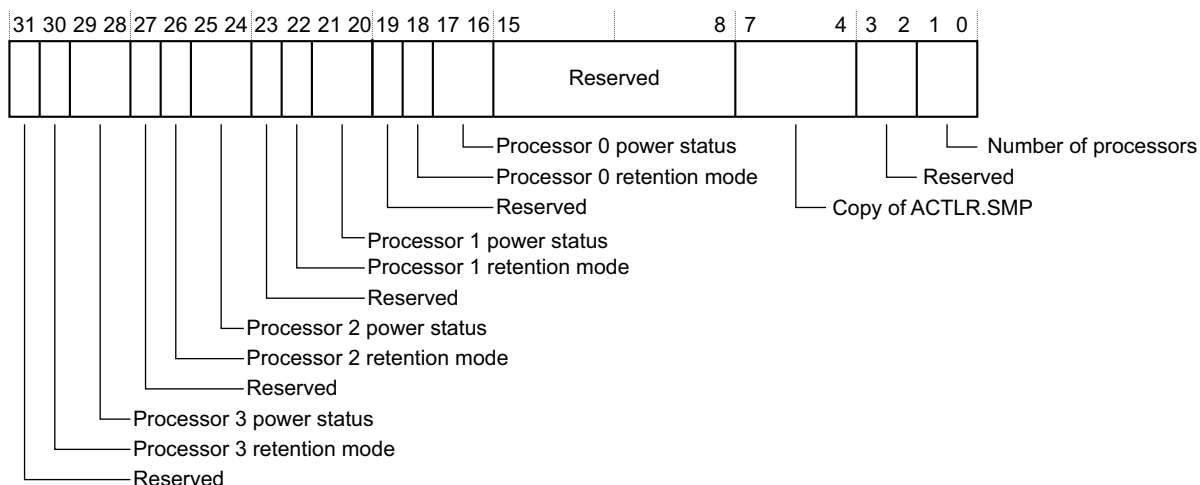
**Usage constraints** The SCUCTLR is:

- A read/write register.
- Common to the Secure and Non-secure states.
- Common to all the processors in the multiprocessor device.
- 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 4-70 on page 4-83](#).

[Figure 4-39](#) shows the SCUCTLR bit assignments.



**Figure 4-39** SCUCTLR bit assignments



Table 4-70 shows the SCUCTLR bit assignments.

**Table 4-70 SCUCTLR bit assignments**

Bits	Name	Function
[31]	-	Reserved, RAZ/WI.
[30]	Processor 3 retention mode	Disable processor 3 retention: <b>0</b> Processor 3 retention enabled. This is the reset value. <b>1</b> Processor 3 retention disabled.
[29:28]	Processor 3 power status	Power status of processor 3: <b>0b00</b> Normal mode. This is the reset value. <b>0b01</b> The processor is not present. <b>0b10</b> The processor is about to enter, or is in, retention mode. <b>0b11</b> The processor is about to enter, or is in, powerdown mode.
[27]	-	Reserved, RAZ/WI.
[26]	Processor 2 retention mode	Disable processor 2 retention: <b>0</b> Processor 2 retention enabled. This is the reset value. <b>1</b> Processor 2 retention disabled.
[25:24]	Processor 2 power status	Power status of processor 2: <b>0b00</b> Normal mode. This is the reset value. <b>0b01</b> The processor is not present. <b>0b10</b> The processor is about to enter, or is in, retention mode. <b>0b11</b> The processor is about to enter, or is in, powerdown mode.
[23]	-	Reserved, RAZ/WI.
[22]	Processor 1 retention mode	Disable processor 1 retention: <b>0</b> Processor 1 retention enabled. This is the reset value. <b>1</b> Processor 1 retention disabled.
[21:20]	Processor 1 power status	Power status of processor 1: <b>0b00</b> Normal mode. This is the reset value. <b>0b01</b> The processor is not present. <b>0b10</b> The processor is about to enter, or is in, retention mode. <b>0b11</b> The processor is about to enter, or is in, powerdown mode.
[19]	-	Reserved, RAZ/WI.
[18]	Processor 0 retention mode	Disable processor 0 retention: <b>0</b> Processor 0 retention enabled. This is the reset value. <b>1</b> Processor 0 retention disabled.
[17:16]	Processor 0 power status	Power status of processor 0: <b>0b00</b> Normal mode. This is the reset value. <b>0b01</b> The processor is not present. <b>0b10</b> The processor is about to enter, or is in, retention mode. <b>0b11</b> The processor is about to enter, or is in, powerdown mode.
[15:8]	-	Reserved, RAZ/WI.

**Table 4-70 SCUCTLR bit assignments (continued)**

Bits	Name	Function
[7:4]	Copy of ACTLR.SMP	Copy of the ACTLR.SMP for each processor.
[3:2]	-	Reserved, RAZ/WI.
[1:0]	Number of processors	Number of processors present.

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

MRC p15, 1, <Rt>, c9, c0, 4; Read SCU Control Register  
MCR p15, 1, <Rt>, c9, c0, 4; Write SCU Control Register

#### 4.3.50 Diagnostic control registers

The DGNCTLRn characteristics are:

**Purpose** Reserved. These registers must only be used under guidance from ARM.

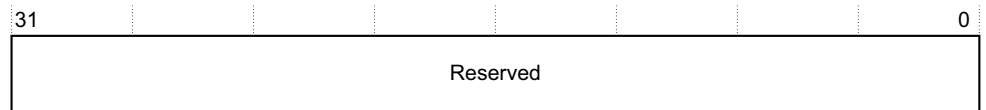
**Usage constraints** The DGNCTLRn is:

- Read/write in Secure PL1 mode.
- Read-only in Non-secure, PL1 or PL2 mode.

**Configurations** Available in all configurations.

**Attributes** See the register summary in [Table 4-14 on page 4-14](#).

[Figure 4-40](#) shows the DGNCTLRn bit assignments.

**Figure 4-40 DGNCTLRn bit assignments**

[Table 4-71](#) shows the DGNCTLRn bit assignments.

**Table 4-71 DGNCTLRn bit assignments**

Bits	Name	Function
[32:0]	-	Reserved, UNK.

To access the DGNCTLR0, read or write the CP15 register with:

MRC p15, 0, <Rt>, c15, c0, 0; Read DGNCTLR0 Register  
MCR p15, 0, <Rt>, c15, c0, 0; Write DGNCTLR0 Register

To access the DGNCTLR1, read or write the CP15 register with:

MRC p15, 0, <Rt>, c15, c0, 1; Read DGNCTLR1 Register  
MCR p15, 0, <Rt>, c15, c0, 1; Write DGNCTLR1 Register

To access the DGNCTLR2, read or write the CP15 register with:

MRC p15, 0, <Rt>, c15, c0, 2; Read DGNCTLR2 Register  
MCR p15, 0, <Rt>, c15, c0, 2; Write DGNCTLR2 Register

### 4.3.51 Diagnostic common control register

The DGNCCTLR characteristics are:

<b>Purpose</b>	Reserved. This register must only be used under guidance from ARM.
<b>Usage constraints</b>	The DGNCCTLR is: <ul style="list-style-type: none"> <li>• Common to all the processors in the multiprocessor device.</li> <li>• Read/write in Secure PL1 mode.</li> <li>• Read-only in Non-secure, PL1 or PL2 mode.</li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 4-14 on page 4-14</a> .

[Figure 4-41](#) shows the DGNCCTLR bit assignments.



**Figure 4-41 DGNCCTLR bit assignments**

[Table 4-72](#) shows the DGNCCTLR bit assignments.

**Table 4-72 DGNCCTLR bit assignments**

Bits	Name	Function
[32:0]	-	Reserved, UNK.

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

MRC p15, 0, <Rt>, c15, c0, 4; Read DGNCCTLR Register

MCR p15, 0, <Rt>, c15, c0, 4; Write DGNCCTLR Register

### 4.3.52 Peripheral port start address register

The FILASTARTR characteristics are:

<b>Purpose</b>	Holds the start address of the peripheral port physical memory region.
<b>Usage constraints</b>	The FILASTARTR is: <ul style="list-style-type: none"> <li>• A read/write register.</li> <li>• Banked for the Secure and Non-secure states.</li> <li>• Common to all the processors in the multiprocessor device.</li> <li>• Only accessible from PL1 or higher.</li> </ul>
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 4-14 on page 4-14</a> .

[Figure 4-42 on page 4-86](#) shows the FILASTARTR bit assignments.

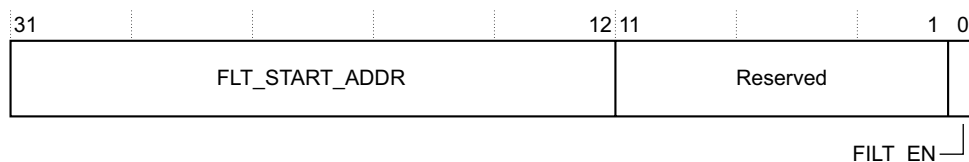


Figure 4-42 FILASTARTR bit assignments

Table 4-73 shows the FILASTARTR bit assignments.

Table 4-73 FILASTARTR bit assignments

Bits	Name	Function
[31:12]	FLT_START_ADDR	Start address of the peripheral port physical memory region. Secure copy resets to <b>CFGADDRFLTSTARTS</b> [39:20]. Non-secure copy resets to <b>CFGADDRFLTSTARTNS</b> [39:20].
[11:1]	-	Reserved, UNK/SBZP.
[0]	FILT_EN	Indicates that FLT_START_ADDR and FLT_END_ADDR are valid. Secure copy resets to <b>CFGADDRFLTENS</b> [39:20]. Non-secure copy resets to <b>CFGADDRFLTENNS</b> [39:20].

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

MRC p15, 4, <Rt>, c15, c0, 1; Read FILASTARTR Register  
MCR p15, 4, <Rt>, c15, c0, 1; Write FILASTARTR Register

### 4.3.53 Peripheral port end address register

The FILAENDR characteristics are:

**Purpose** Holds the end address of the peripheral port physical memory region.

**Usage constraints** The FILAENDR is:

- A read/write register.
- Banked for the Secure and Non-secure states.
- Common to all the processors in the multiprocessor device.
- Only accessible from PL1 or higher.

**Configurations** Available in all configurations.

**Attributes** See the register summary in [Table 4-14 on page 4-14](#).

Figure 4-43 shows the FILAENDR bit assignments.

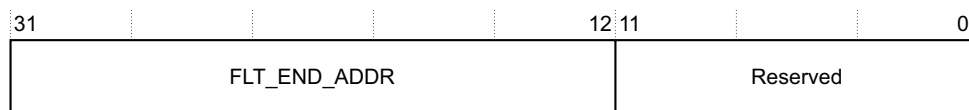


Figure 4-43 FILAENDR bit assignments

Table 4-74 shows the FILAENDR bit assignments.

**Table 4-74 FILAENDR bit assignments**

Bits	Name	Function
[31:12]	FLT_END_ADDR	End address of the peripheral port physical memory region. This includes the 1MB block starting at FLT_END_ADDR. Secure copy resets to <b>CFGADDRFLTENDS[39:20]</b> . Non-secure copy resets to <b>CFGADDRFLTENDNS[39:20]</b> .
[11:0]	-	Reserved, UNK/SBZP.

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

MRC p15, 4, <Rt>, c15, c0, 2; Read FILAENDR Register

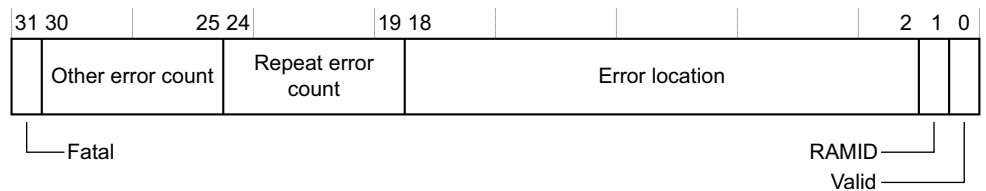
MCR p15, 4, <Rt>, c15, c0, 2; Write FILAENDR Register

#### 4.3.54 L2 Memory Error Syndrome Register

The L2MRERRSR characteristics are:

<b>Purpose</b>	Holds the number of memory errors that have occurred in the following L2 RAMs: <ul style="list-style-type: none"> <li>L2 tag RAM.</li> <li>L2 data RAM.</li> </ul>
<b>Usage constraints</b>	The L2MRERRSR is: <ul style="list-style-type: none"> <li>A read/write register.</li> <li>Common to the Secure and Non-secure states.</li> <li>Common to all the processors in the multiprocessor device.</li> <li>Only accessible from PL1 or higher.</li> <li>A write of any value to the register updates the register to 0.</li> </ul>
<b>Configurations</b>	Only available if L2 ECC is implemented. Access is Read-only, RAZ if not implemented.
<b>Attributes</b>	See the register summary in <a href="#">Table 4-75 on page 4-88</a> .

[Figure 4-44 on page 4-88](#) shows the L2MRERRSR bit assignments.

**Figure 4-44 L2MRERRSR bit assignment****Table 4-75 L2MRERRSR bit assignment**

Bits	Name	Function
[31]	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.
[30:25]	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, way, or index information in this register while the sticky Valid bit is set. The value saturates to 63. The reset value is 0.
[24:19]	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, way, or index information in this register while the sticky Valid bit is set. The value saturates to 63. The reset value is 0.
[18:2]	Error location	Indicates the location at which the first memory error is detected. <b>[18:6]</b> Index. <b>[5:2]</b> Way.
[1]	RAMID	RAM Identifier. Indicates the RAM where the first memory error occurred: <b>0</b> Tag RAM. <b>1</b> Data RAM.
[0]	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.

**Note**

- Because L2 RAM accesses are speculative, an error can be detected several times before being corrected. This means that the error counter might be incremented several times for the same error.
- If several errors are detected while the error detection function is active, only one error might be reported.

To access the L2MRERRSR, read or write the CP15 register with:

MRC p15, 1, <Rt>, c9, c0, 6; Read L2 Memory Error Syndrome Register

MCR p15, 1, <Rt>, c9, c0, 6; Write L2 Memory Error Syndrome Register

**4.3.55 Auxiliary Memory Attribute Indirection Register 0**

The processor does not implement AMAIR0, so this register is always UNK/SBZP.

**4.3.56 Auxiliary Memory Attribute Indirection Register 1**

The processor does not implement AMAIR1, so this register is always UNK/SBZP.

**4.3.57 Hyp Auxiliary Memory Attribute Indirection Register 0**

The processor does not implement HAMAIR0, so this register is always UNK/SBZP.

**4.3.58 Hyp Auxiliary Memory Attribute Indirection Register 1**

The processor does not implement HAMAIR1, so this register is always UNK/SBZP.

**4.3.59 FCSE Process ID Register**

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

# Chapter 5

## Memory Management Unit

This chapter describes the *Memory Management Unit* (MMU) and the address translation process. It contains the following sections:

- *About the MMU* on page 5-2.
- *TLB organization* on page 5-3.
- *TLB match process* on page 5-4.
- *Memory access sequence* on page 5-5.
- *MMU enabling and disabling* on page 5-7.
- *Memory types* on page 5-10.
- *Memory region attributes* on page 5-11.



## 5.1 About the MMU

The Cortex-A17 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.

The MMU controls table walk hardware that accesses translation tables in memory. The MMU translates virtual addresses to physical addresses. The MMU provides fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes held in translation tables. The instruction micro TLB, data micro TLB, and main TLB provide caching for translations and translation table entries.

The Cortex-A17 MMU features include the following:

- 32, 64 or 48 entry fully-associative instruction micro TLB.
- 32 entry fully-associative data micro TLB.
- 4-way set-associative, 1024 entry, main TLB.
- 2-way set-associative 128 entry intermediate table walk cache.
- The TLB entries contain a global indicator or an *Address Space Identifier* (ASID) to permit context switches without requiring the TLB to be invalidated.
- The TLB entries contain a *Virtual Machine Identifier* (VMID) to permit virtual machine switches without requiring the TLB to be invalidated.

## 5.2 TLB organization

The Cortex-A17 MPCore processor implements a 2-level TLB structure. The TLBs, at either the micro or the main level, do not require to be invalidated on a context or virtual machine switch. The Cortex-A17 MMU does not support locking TLB entries in either the micro TLBs or the main TLB.

This section describes the TLB organization in:

- *Instruction micro TLB.*
- *Data micro TLB.*
- *Unified main TLB.*

### 5.2.1 Instruction micro TLB

The instruction micro TLB is implemented as a 32, 48 or 64 entry, fully-associative structure. This TLB caches entries at the 4KB and 1MB granularity of *Virtual Address (VA)* to *Physical Address (PA)* mapping only. If the translation tables map the memory region to a larger granularity than 4KB or 1MB, it only allocates one mapping for the particular 4KB 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, generating a Prefetch Abort exception if access is not permitted.

### 5.2.2 Data micro TLB

The data micro TLB is a 32 entry fully-associative TLB that is used for data loads and stores. The cache entries have a 4KB and 1MB granularity of VA to PA mappings only.

A hit in the data TLB provides a single **CLK** cycle access to the translation, and returns the physical address to the data cache for comparison. It also checks the access permissions, generating a Data Abort exception if access is not permitted.

### 5.2.3 Unified main TLB

Misses from the instruction and data micro TLBs are handled by a unified main TLB. This is a 1024 entry 4-way set-associative structure. The main 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 main TLB take a variable number of cycles, based on the competing requests from each of the micro TLBs, TLB maintenance operations in flight, and the different page size mappings in use. The main TLB can service multiple requests in parallel, including translation table walks.

### 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 invalidation on a context or virtual machine switch. Each TLB entry contains a VA, page size, PA, and a set of memory properties that include the memory type and access permissions. Each entry is associated with a particular ASID, or as global, for all application spaces. The TLB entry also contains a field to store the *Virtual Machine Identifier* (VMID) 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 VA, moderated by the page size such as the VA 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, *Translation Table Base Register 0* (TTBR0) or *Translation Table Base Register 1* (TTBR1) if 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 VA, current ASID, current VMID, and security state in the relevant instruction or data micro TLB.
2. Performs a lookup for the requested VA, current ASID, current VMID, and security state in the unified main TLB if there is a miss in the relevant micro TLB.
3. Performs a hardware translation table walk if there is a miss in the main TLB.

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 TTBR0 and TTBR1.
- The Long-descriptor translation table format, by setting the IRGN bits in the relevant 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 a main 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 Shareable, Outer Shareable or not.
  - Normal memory, Device, or Strongly-ordered.

- The TLB translates the VA to a PA for the memory access.

## 5.5 MMU enabling and disabling

The global enable for the MMU is controlled by SCTLR.M. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

## 5.6 Intermediate table walk cache

The Cortex-A17 MPCore processor implements a dedicated cache that stores intermediate levels of translation table entries as part of a table walk. Cached entries are associated with an ASID. The TLB and the intermediate cache 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 VA 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 Memory types

Although various different memory types can be specified in the translation tables, the Cortex-A17 MPCore processor does not implement all possible combinations:

- All Write-Through mappings, including mappings to instruction memory, are treated as Non-cacheable. *The Physical Address Register* (PAR) reports the memory attributes as Normal, Non-cacheable.
- All Inner Write-Back memory is treated as Write-Back Write-Allocate ignoring any cache allocate hint, though this can dynamically switch to no Write-Allocate, if more than three full cache lines are written in succession. See [L1 data memory system on page 6-6](#). All Inner Write-Back is reported as Write-Back Write-Allocate in the PAR.
- For Non-cacheable memory, accesses are not looked up in L1 caches.  
Normal memory, which is both Inner Non-cacheable and Outer Non-cacheable is always reported as Outer Shareable irrespective of the translation table settings.

The attribute behavior for Strongly-ordered and Device memory types are architecturally-defined, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

---

### Note

---

Cortex-A17 MPCore processor implements the IMPLEMENTATION DEFINED memory attribute: TEX[2:0] = 0b001, C = 1, B = 0 as Strongly-ordered.

---

## 5.8 Memory region attributes

The memory region attributes control the memory type, accesses to the caches, and whether the memory region is Shareable.

Table 5-1 shows the C, B, and TEX[2:0] encodings for Short-descriptor format memory region without TEX remapping.

**Table 5-1 TEX, C, and B encodings when SCTL.RTRE is set to 0**

TEX[2:0]	C	B	Description	Memory type	Page Shareable
000	0	0	Strongly-ordered	Strongly-ordered	Shareable
		1	Shareable Device	Device	Shareable
	1	0	Outer and Inner Write-Through, no Write-Allocate	Normal	S bit <sup>a</sup>
		1	Outer and Inner Write-Back, no Write-Allocate	Normal	S bit <sup>a</sup>
001	0	0	Outer and Inner Non-cacheable	Normal	S bit <sup>a</sup>
		1	Reserved	-	-
	1	0	Strongly-ordered	Strongly-ordered	Shareable
		1	Outer and Inner Write-Back, Write-Allocate	Normal	S bit <sup>a</sup>
010	0	0	Non-shareable Device	Device	Non-shareable
		1	Reserved	-	-
	1	x	Reserved	-	-
011	x	x	Reserved	-	-
1BB	A	A	Cacheability <sup>b</sup>	Normal	S bit <sup>a</sup>

a. The S bit selects outer shareability. It is not possible to specify inner shareability when TEX remap is set to 0.

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

# Chapter 6

## L1 Memory System

This chapter describes the L1 Memory System. It contains the following sections:

- *About the L1 memory system on page 6-2.*
- *Cache features on page 6-3.*
- *L1 instruction memory system on page 6-4.*
- *L1 data memory system on page 6-6.*
- *Data prefetching on page 6-9.*
- *Direct access to internal memory on page 6-10.*

## 6.1 About the L1 memory system

The L1 memory system consists of separate instruction and data caches. The size of the instruction cache is implemented as either 32KB or 64KB. The size of the data cache is 32KB.

The L1 memory system has the following features:

- Support for four sizes of memory page.
- Export of memory attributes for external memory systems.
- Support for Security Extensions.

The L1 instruction cache has the following features:

- Instruction side cache line length of 64-bytes.
- Virtually indexed and physically tagged instruction cache.
- Pseudorandom cache replacement policy.
- 4-way set-associative instruction cache.
- The Instruction cache can be enabled and disabled by the system control coprocessor. See [System Control Register on page 4-52](#).

The L1 data cache has the following features:

- Data side cache line length of 64-bytes.
- Physically indexed and physically tagged data cache.
- Pseudorandom cache replacement policy.
- 4-way set-associative data cache.
- One 64-byte linefill buffer and one 64-byte eviction buffer.
- An 8-entry, 64-bit merging store buffer.
- The Data cache can be enabled and disabled by the system control coprocessor. See [System Control Register on page 4-52](#).

## 6.2 Cache features

This section describes the implementation-specific features of the instruction and data caches:

- At reset the instruction and data caches are disabled.
- Both caches are automatically invalidated immediately after reset.
- You can enable or disable each cache independently. See [System Control Register on page 4-52](#).
- Cache lockdown is not supported.
- On a cache miss, data for the cache linefill is requested in critical word-first order.

## 6.3 L1 instruction memory system

The instruction cache has a 64-byte line length. Cache data is banked across multiple RAMs so that instruction fetch lookups do not require full cache lines to be read. The instruction cache can source up to 128 bits per fetch depending on alignment.

On cache miss the I-side requests linefill data from the L2 memory system in 64-byte transactions. A cache miss is non-blocking and there can be up to four linefill requests outstanding.

### 6.3.1 Instruction cache disabled behavior

At reset the L1 instruction cache is disabled. CP15 instruction cache maintenance operations can operate with the instruction cache disabled. The instruction cache is enabled by setting the SCR.I bit. See [System Control Register on page 4-52](#).

### 6.3.2 Instruction cache speculative memory accesses

Instructions are prefetched from executable memory (memory with access control, *Execute-never* (XN) = 0) before the processor can determine if they are to be executed. This is referred to as speculative memory accesses. Blocks of instructions are speculatively fetched and held in the instruction cache. This has two effects on instruction memory. Firstly, memory might be read beyond the range of any valid program code. Secondly, if the cache is invalidated the same block can be read multiple times. Because of this, you must not place read-sensitive devices in the same memory region as code. You must mark regions that contain read-sensitive devices with the memory access control, XN = 1. To avoid speculative fetches to read-sensitive devices when address translation is disabled, device and code areas must be separated in the physical memory map. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for more information.

### 6.3.3 Program flow prediction

The processor contains program flow prediction hardware, also known as branch prediction. Branch prediction increases overall performance and reduces power consumption. With program flow prediction disabled, all taken branches incur a penalty associated with flushing the pipeline. To avoid this penalty, the branch prediction hardware predicts if a conditional branch is to be taken and predicts the address that the branch is to go to, known as the branch target address. The hardware contains the following functionality:

- A *Branch Target Address Cache* (BTAC) holding the branch target address of previously taken branches.
- Dynamic branch predictor history stored in RAMs.
- The return stack. A stack of nested subroutine return addresses.
- Static branch predictor.

The following sections describe program flow prediction:

- [Predicted and non-predicted instructions](#).
- [Thumb state conditional branches on page 6-5](#).
- [Return stack predictions on page 6-5](#).

#### Predicted and non-predicted instructions

This section shows the instructions that the processor predicts. Unless otherwise specified, the list applies to ARM, Thumb and ThumbEE instructions. As a general rule, the flow prediction hardware predicts all branch instructions regardless of the addressing mode, including:

- Conditional branches.

- Unconditional branches.
- Immediate branches including function call instructions.
- Indirect branches including function return instructions.
- Branches that switch between ARM and Thumb states.

However, some branch instructions are not predicted:

- Instructions with the S suffix are not predicted because they are typically used to return from exceptions and have side-effects that can change privilege mode and security state.
- All mode-changing instructions.

### Thumb state conditional branches

In Thumb state, a branch that is normally encoded as unconditional can be made conditional by inclusion in an *If-Then* (IT) block. Then it is treated as a normal conditional branch.

### 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 equal to the link register value stored in r14. The following instructions cause a return stack push if predicted:

- BL immediate.
- BLX immediate.
- BLX register.

The following instructions cause a return stack pop if predicted:

- BX r14.
- POP {...,pc}.
- LDR pc, [r13].
- MOV pc, r14.

The LDR instruction can use any of the addressing modes, as long as r13 is the base register.

Because return from exception instructions can change processor privilege mode and security state, they are not predicted. This includes the LDM R<n>, {...,pc}<sup>^</sup> instruction, RFE, and the MOVS pc, r14 instruction.

## 6.3.4 Enabling program flow prediction

Program flow prediction is always enabled when the MMU is enabled by setting the SCTL.RM bit to 1. See [System Control Register on page 4-52](#).

## 6.4 L1 data memory system

The L1 data cache is organized as a physically indexed and physically tagged cache. The data micro TLB produces the physical address from the virtual address before performing the cache access.

### 6.4.1 Behavior for different memory types

The L1 data memory system uses memory attributes from the MMU to determine how to behave 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 6-7.](#)
- [Non-cacheable on page 6-7.](#)
- [Strongly-ordered and Device on page 6-7.](#)

**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-A17 MPCore processor behavior
Strongly-ordered	-	-	Strongly-ordered
Device	-	-	Device
Normal	Non-cacheable	-	Normal Non-cacheable
Normal	Write-Through	Read-Allocate	Normal Non-cacheable
Normal	Write-Through	Write-Allocate	Normal Non-cacheable
Normal	Write-Through	Read-Write-Allocate	Normal Non-cacheable
Normal	Write-Through	No-Allocate	Normal Non-cacheable
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 Read-Write-Allocate

The L1 and L2 data memory system uses the inner memory attributes from the MMU to determine its behavior.

If a memory instruction executes an operation that crosses a 4KB page boundary between two pages with different memory types such as Normal or Strongly-ordered, the result is unpredictable.

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

Write-Back No-Allocate is implemented as Write-Back Read-Write-Allocate.

### Write-Through

The Cortex-A17 processor memory system treats all Write-Through pages as Normal Non-cacheable.

### Non-cacheable

Normal Non-cacheable memory is not looked-up in any cache. The requests are sent directly to external 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.

### 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-A17 MPCore processor, there is no distinction between Shareable and Non-shareable devices.

All the memory operations for a single instruction can be sent to external memory as a burst request. No Strongly-ordered or Device read or write request on the interconnect can cross an aligned 64-byte boundary.

## 6.4.2 Internal exclusive monitor

The Cortex-A17 MPCore processor L1 memory system has an internal exclusive monitor. This is a two-state, open and exclusive, state machine that manages load/store exclusive (LDREXB, LDREXH, LDREX, LDREXD, STREXB, STREXH, STREX, and STREXD) accesses and clear exclusive (CLREX) instructions. You can use these instructions to construct semaphores, ensuring synchronization between different processes running on the processor, and also between different processors that are using the same coherent memory locations for the semaphore. A Load-Exclusive instruction tags a small block of memory for exclusive access. The size of the tagged block is defined by CTR.ERG as 64 bytes, one cache line.

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

---

**Note**

---

Code must not assume that failure of the STREX instruction indicates other exclusive activity. The Cortex-A17 multiprocessor might clear an exclusive monitor, between the LDREX and the STREX, without any application-related cause.

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

---

ARM recommends that no load or store instructions are placed between the LDREX and STREX because these additional instructions can cause a cache eviction. Data cache maintenance instructions can also clear the exclusive monitor.

**Treatment of intervening STR operations**

In cases where there is an intervening STR operation in an LDREX/STREX code sequence, the intermediate STR does not produce any direct effect on the internal exclusive monitor. The local monitor is in the Exclusive Access state after the LDREX, remains in the Exclusive Access state after the STR, and returns to the Open Access state only after a STREX, a CLEAREX, or an eviction of the cache line.

## 6.5 Data prefetching

This section describes:

- [PLD, PLDW, and PLI instructions](#).
- [Data prefetching and monitoring](#).

### 6.5.1 PLD, PLDW, and PLI instructions

PLD and PLDW instructions, for cacheable data, perform a lookup in the cache and start a linefill if they miss. The PLD or PLDW instruction retires as soon as its linefill is started, rather than waiting for data to be returned. This enables other instructions to execute while the linefill continues in the background. There are four PLD slots to allow further PLD instructions to execute if a preceding PLD causes a translation table walk. Any linefill started by a PLDW instruction causes the data to be invalidated in other processors, so that the line is ready to be written to.

The PLI instruction performs preloading in the L2 cache for cacheable accesses if they miss in both the L1 instruction cache and L2 cache. Instruction preloading is performed in the background.

### 6.5.2 Data prefetching and monitoring

The processor data cache implements an automatic prefetcher that monitors cache misses in the processor. When a pattern is detected, the automatic prefetcher starts linefills in the background. The prefetcher recognizes a sequence of three data cache misses at a fixed stride pattern that lies within eight cache lines, plus or minus. Any intervening stores or loads that hit in the data cache do not interfere with the recognition of the cache miss pattern. Up to eight independent streams can be handled by the prefetcher. It can be deactivated in software using a CP15 Auxiliary Control Register bit. See [Auxiliary Control Register on page 4-55](#).

Use the PLD instruction for data prefetching where short sequences or irregular pattern fetches are required.

## 6.6 Direct access to internal memory

The Cortex-A17 MPCore processor provides a mechanism to read the internal memory used by the L1 cache and TLB structures through the implementation-defined region of the system coprocessor interface. This functionality can be useful when investigating issues where the coherency between the data in the cache and data in system memory is broken.

The appropriate memory block and location is selected using a number of write-only CP15 registers and the data is read from read-only CP15 registers as shown in [Table 6-2](#). These operations are only available in secure privileged modes. In all other modes, executing the CP15 instruction results in an Undefined Instruction exception.

**Table 6-2 Cortex-A17 MPCore system coprocessor CP15 registers used to access internal memory**

Function	Access	CP15 operation	Rd Data
Data Register 0	Read-only	MRC p15, 3, <Rd>, c15, c0, 0	Data
Data Register 1	Read-only	MRC p15, 3, <Rd>, c15, c0, 1	Data
Data Register 2	Read-only	MRC p15, 3, <Rd>, c15, c0, 2	Data
Data Cache Tag Read Operation Register	Write-only	MCR p15, 3, <Rd>, c15, c2, 0	Set/Way
Instruction Cache Tag Read Operation Register	Write-only	MCR p15, 3, <Rd>, c15, c2, 1	Set/Way
Data Cache Data Read Operation Register	Write-only	MCR p15, 3, <Rd>, c15, c4, 0	Set/Way/Offset
Instruction Cache Data Read Operation Register	Write-only	MCR p15, 3, <Rd>, c15, c4, 1	Set/Way/Offset
TLB Data Read Operation Register	Write-only	MCR p15, 3, <Rd>, c15, c4, 2	Index/Way

The following sections describe the encodings for the operations and the format for the data read from the memory:

- [Data cache tag and data encoding](#).
- [Instruction cache tag and data encoding on page 6-11](#).
- [TLB data encoding on page 6-12](#).

### 6.6.1 Data cache tag and data encoding

The L1 data cache consists of a 4-way set-associative structure. The encoding, set in Rd in the appropriate MCR instruction, used to locate the required cache data entry for tag and data memory is shown in [Table 6-3](#). It is very similar for both the tag and data RAM access. Data RAM access includes an additional field to locate the appropriate doubleword in the cache line.

**Table 6-3 Data cache tag and data location encoding**

Bit-field of Rd	Description
[31:30]	Cache way
[29:13]	Unused
[12:6]	Set index
[5:3]	Cache doubleword data offset, Data Register only
[2:0]	Unused

Data cache reads return 64 bits of data in Data Register 0 and Data Register 1. The tag information, MOESI coherency state, outer attributes, and valid, for the selected cache line is returned using Data Register 0 and Data Register 1 using the format shown in [Table 6-4](#).

**Table 6-4 Data cache tag data format**

Register	Bit-field	Description
Data Register 0	[31]	Partial MOESI state, Dirty
	[30]	Partial Outer memory attribute
	[29]	Partial MOESI state, Exclusive
	[28]	Partial MOESI state, Valid
	[27]	Non-secure state, NS
	[26:0]	Tag address [39:13]
Data Register 1	[31:1]	Unused
	[0]	Partial MOESI state, Globally shared

### 6.6.2 Instruction cache tag and data encoding

The data format of the instruction cache is significantly different from the data cache. This is shown in the encodings and data format used in the CP15 operations used to access the tag and data memories. [Table 6-5](#) shows the encoding required to select a given cache line. The number of sets in each way depends on the implemented size of the cache.

The set-index range parameter (S) is determined by:

$$S = \log_2((\text{Instruction cache size (Byte)} / \text{cache line size (Byte)}) / 4).$$

**Table 6-5 Instruction cache tag and data location encoding**

Bit-field of Rd	Description
[31:30]	Cache Way
[29:S+6]	Unused
[S+5:6]	Set index
[5:3]	Cache doubleword data offset, Data Register only
[2:0]	Unused

[Table 6-6](#) shows the tag and valid bits format for the selected cache line.

**Table 6-6 Instruction cache tag data format**

Bit	Description
[31:30]	Unused
[29]	Valid
[28]	Non-secure state (NS)
[27:0]	Tag address[39:12]

The CP15 Instruction Cache Data Read Operation returns a 64-bit entry from the cache in Data Register 0 and Data Register 1.

The CP15 Instruction Cache Tag Read Operation returns two entries from the cache in Data Register 0 and Data Register 1 corresponding to the 16-bit aligned offset in the cache line:

**Data Register 0** Bits[31:0] data from cache offset+ 0b000.

**Data Register 1** Bits[31:0] data from cache offset+ 0b100.

### 6.6.3 TLB data encoding

The unified main TLB is built from two RAMs:

**RAM 0** This is a 4-way set-associative (4 x 256) RAM array, for 4KB and 64KB page translation entries.

RAM0 exclusively holds TLB regular format entries.

**RAM 1** This is a 2-way set-associative (2 x 64) RAM array for 1MB, 2MB, 16MB, and 1GB page translation entries, walk cache and *Intermediate Physical Address* (IPA) cache.

RAM1 holds:

- TLB regular format entries.
- TLB walk format entries.
- TLB IPA format entries.

The TLB walk format entry is identified by the Entry type field set to 1.

The TLB IPA format entry is identified by the Entry type field set to 0 and the S1 level field set to 0b11.

To read the individual entries into the data registers software must write to the TLB Data Read Operation Register. Table 6-7 shows the TLB Data Read Operation Register location encoding.

**Table 6-7 TLB Data Read Operation Register location encoding**

Bit-field of Rd	Description
[31:30] <sup>a</sup>	TLB way.
[30:9]	Unused.
[8]	Type: <b>0</b> RAM0. <b>1</b> RAM1.
[7:0] <sup>b</sup>	TLB index.

a. Bit [31] unused if bit [8] = 1.

b. Bits [7:6] unused if bit [8] = 1.

Both RAM0 and RAM1 are 90 bits wide. The data is returned in the data registers:

**Data Register 0**[31:0] RAM data [31:0].

**Data Register 1**[31:0] RAM data[63:32].

**Data Register 2**[25:0] RAM data[89:64].

## TLB regular format

Table 6-8 shows the TLB regular format encoding.

**Table 6-8 TLB regular format encoding**

Bits	Name	Description
[89]	Valid entry	Indicates that the entry is valid: <b>0</b> Entry is not valid. <b>1</b> Entry is valid.
[88]	Entry type	Indicates the entry type: <b>0</b> Main or IPA cache entry. <b>1</b> Walk cache entry.
[87:72]	VA[31:16]	Indicates the virtual address.
[71:64]	VMID[7:0]	Indicates the virtual machine identifier.
[63:56]	ASID[7:0]	Indicates the address space identifier.
[55]	nG	Indicates the Non-global bit.
[54]	LPAE	Indicates the format of the entry: <b>0</b> VMSAv7 format. <b>1</b> LPAE format.
[53]	Outer Shareable	Indicates the Outer Shareable memory attribute.
[52:47]	MemAttr[7:2]	Indicates the memory attributes. See <a href="#">Table 6-11 on page 6-16</a> .
[46:45]	HAP[1:0]	Indicates the hypervisor access permissions, from stage 2 translation.
[44:43]	S2 level[1:0]	Indicates the stage 2 level that gave this translation: <b>0b00</b> No stage 2 translation performed. <b>0b01</b> Level 1. <b>0b10</b> Level 2. <b>0b11</b> Level 3.
[42:41]	S1 level[1:0]	Indicates the stage 1 level that gave this translation: <b>LPAE == 0</b> <b>0b00</b> Level 2. <b>0b01</b> Level 1. <b>0b10</b> - <b>0b11</b> No stage 1 translation. <b>LPAE == 1</b> <b>0b00</b> Level 3. <b>0b01</b> Level 2. <b>0b10</b> Level 1. <b>0b11</b> IPA to PA translation entry. No stage 1 translation.
[40]	Page size	Indicates the page size or contiguous hint, depending on LPAE field of the entry: <b>LPAE == 0</b> 16MB or 64KB page. <b>LPAE == 1</b> Contiguous hint.
[39]	PXN	Indicates the privileged execute-never attribute, from stage 1 translation.
[38]	XN1	Indicates the Execute-never attribute, from stage 1 translation.
[37]	XN2	Indicates the Execute-never attribute, from stage 2 translation.
[36:34]	AP[2:0]	Indicates the access permissions, from stage 1 translation.

Table 6-8 TLB regular format encoding (continued)

Bits	Name	Description
[33:30]	Domain[3:0]	Indicates the memory regions domain, depending on LPAE field of the entry: <b>LPAE == 0</b> Memory regions domain. <b>LPAE == 1</b> UNKNOWN.
[29]	TIB	Indicates additional descriptor information. <b>NS == 0</b> <b>0</b> NS Descriptor. <b>1</b> NS Descriptor. <b>NS == 1</b> <b>0</b> Non-secure PL0 or PL1 entry. <b>1</b> PL2 entry.
[28]	NS	Non-secure VA.
[27:00]	PA[39:12]	Indicates the physical address.

### TLB Walk format

Table 6-9 shows the encoding for a translation table walk entry.

Table 6-9 TLB Walk format encoding

Bit	Name	Description
[89]	Valid entry	Indicates that the entry is valid: <b>0</b> Entry is not valid. <b>1</b> Entry is valid.
[88]	Entry type	Indicates the entry type: <b>0</b> Regular or IPA TLB format entry. <b>1</b> Walk cache entry.
[87:72]	VA[31:16]	Indicates the virtual address.
[71:64]	VMID[7:0]	Indicates the virtual machine identifier.
[63:56]	ASID[7:0]	Indicates the address space identifier.
[55]	nG	Indicates the Non-global bit. Always 1 for walk-cache entries.
[54]	LPAE	Indicates the format of the entry: <b>0</b> VMSAv7 format. <b>1</b> LPAE format.
[53]	Outer Shareable	Indicates the Outer Shareable memory attribute.
[52:47]	MemAttr[7:2]	Indicates the memory attributes. See <a href="#">Table 6-11 on page 6-16</a> .
[46:45]	PA[11:10]	Indicates the physical address, depending on LPAE field of the entry: <b>LPAE == 0</b> Physical address[11:10]. <b>LPAE == 1</b> UNKNOWN.
[44:40]	-	UNKNOWN.
[39]	PXNTable	Indicates the privileged execute-never attribute.
[38]	XNTable	Indicates the Execute-never attribute.
[37]	-	UNKNOWN.



**Table 6-9 TLB Walk format encoding (continued)**

Bit	Name	Description
[35:34]	APTable	Indicates the access permissions.
[33:30]	Domain[3:0]	Indicates the memory regions domain, depending on LPAE field of the entry: <b>LPAE == 0</b> Memory regions domain. <b>LPAE == 1</b> UNKNOWN.
[29]	TIB	Indicates additional descriptor information. <b>NS == 0</b> <b>0</b> NS Descriptor. <b>1</b> NS Descriptor. <b>NS == 1</b> <b>0</b> Non-secure PL0 or PL1 entry. <b>1</b> PL2 entry.
[28]	NS	Non-secure VA.
[27:00]	PA[39:12]	Indicates the physical address.

**TLB IPA format**

[Table 6-10](#) shows the encoding for an IPA translation entry.

**Table 6-10 TLB IPA format encoding**

Bits	Name	Description
[89]	Valid entry	Indicates that the entry is valid: <b>0</b> Entry is not valid. <b>1</b> Entry is valid.
[88]	Entry type	Indicates the entry type: <b>0</b> Main or IPA cache entry. <b>1</b> Walk cache entry.
[87:72]	IPA[31:16]	Indicates the intermediate physical address.
[71:64]	VMID[7:0]	Indicates the virtual machine identifier.
[63:56]	IPA[39:32]	Indicates the intermediate physical address.
[55]	-	UNKNOWN.
[54]	LPAE	Indicates the format of the entry: <b>0</b> VMSAv7 format. <b>1</b> LPAE format.
[53]	Outer Shareable	Indicates the Outer Shareable memory attribute.
[52:47]	MemAttr[7:2]	Indicates the memory attributes. See <a href="#">Table 6-11 on page 6-16</a> .
[46:45]	HAP[1:0]	Indicates the hypervisor access permissions.
[44:43]	S2 level[1:0]	Indicates the stage 2 level that gave this translation: <b>0b00</b> No stage 2 translation performed. <b>0b01</b> Level 1. <b>0b10</b> Level 2. <b>0b11</b> Level 3.

**Table 6-10 TLB IPA format encoding (continued)**

Bits	Name	Description
[42:41]	S1 level[1:0]	Always 0b11.
[40]	-	UNKNOWN.
[39:38]	MemAttr[1:0]	Indicates the memory attributes. See <a href="#">Table 6-11</a> .
[37]	XN	Indicates the Execute-never attribute.
[36:30]	-	UNKNOWN.
[29]	TIB	Indicates additional descriptor information.
		NS == 0      0      NS Descriptor.
		1      NS Descriptor.
		NS == 1      0      Non-secure PL0 or PL1 entry.
		1      PL2 entry.
[28]	NS	Non-secure VA (always 1).
[27:00]	PA[39:12]	Indicates the physical address.

### Memory attributes encoding

[Table 6-11](#) shows the encoding of the memory attributes field of a TLB entry.

**Table 6-11 Memory attributes**

MemAttrs[7:0]	Description
0b00000000	Strongly-ordered
0b00010001	Device
0b00110011	Outer Non-cacheable, Inner Non-cacheable
0b00110111	Outer Non-cacheable, Inner Non-cacheable
0b00111111	Outer Non-cacheable, Inner Non-cacheable
0b01110011	Outer Write-Through, Inner Non-cacheable
0b01110111	Outer Write-Through, Inner Write-Through
0b01111111	Outer Write-Through, Inner Write-Back
0b01110011	Outer Write-Back, Inner Non-cacheable
0b11110111	Outer Write-Back, Inner Write-Through
0b11111111	Outer Write-Back, Inner Write-Back

# Chapter 7

## L2 Memory System

This chapter describes the L2 Memory System. It contains the following sections:

- *About the L2 Memory system on page 7-2.*
- *L2 cache on page 7-3.*
- *Cache coherency on page 7-6.*
- *ACE master interface on page 7-7.*
- *ACP on page 7-12.*
- *Peripheral port on page 7-14.*
- *External aborts and asynchronous errors on page 7-15.*

## 7.1 About the L2 Memory system

The L2 memory system consists of:

- An AMBA ACE master interface 128-bit bus.
- An AMBA AXI4 Peripheral port master interface 128-bit bus.
- An AMBA AXI *Accelerator Coherency Port* (ACP) slave interface 128-bit bus.
- Integrated *Snoop Control Unit* (SCU), connecting all the processors within a cluster:
  - It maintains L1 data cache coherency between processors by snooping the L1 caches.
  - It holds duplicate copies of the L1 data cache tags for efficient coherency support.
  - It arbitrates requests from the processors and the ACP interface.
  - It does not support hardware management of coherency of the instruction caches.
- An integrated L2 cache:
  - The cache size is implemented as either 256KB, 512KB, 1MB, 2MB, 4MB or 8MB.
  - The cache is implemented with or without ECC.
  - A fixed line length of 64 bytes.
  - Physically indexed and tagged cache.
  - 16-way set-associative cache structure.
  - Pseudo-random cache replacement policy.

The L2 memory system has a synchronous abort mechanism and an asynchronous abort mechanism, see [External aborts and asynchronous errors on page 7-15](#).

## 7.2 L2 cache

The integrated L2 cache is the Point of Unification for the Cortex-A17 MPCore processor. It handles both instruction and data requests from the I-Side and D-side LSU respectively. Because the L2 cache is integrated into the multiprocessor it is controlled by CP15 operations.

- Data is allocated in the L2 cache on linefills from the L1 cache.
- Instructions are allocated to the L2 cache when fetched from the system.
- There are 16 linefill buffers in single-processor implementations and 32 in multiprocessor implementations. In all implementations there are 16 eviction buffers.
- The L2 cache tags are looked up in parallel with the SCU duplicate tags. If both the L2 tag and SCU duplicate tag hit, the L1 cache data is used, because the L2 cache data might not be up to date if there has been a write since the last linefill.
- Additionally, speculative requests to the system from the SCU are not made until the system checks the L2 cache.
- L2 RAMs are invalidated automatically at reset unless the **L2RSTDISABLE** signal is set HIGH when the **nL2RESET** signal is deasserted.

### 7.2.1 L2 error correction

L2 ECC is an implementation option. Single-bit error correction and double-bit error detection is provided for the L2 cache tag and data RAMs. On the data RAM ECCs are applied with 8-byte granularity.

If a single error is detected, the transfer is blocked and the cache line in error is corrected and, if valid and dirty, evicted to the external memory. If the line is clean it is invalidated and the line is read again from the external memory.

If a double error is detected, and the cache line is dirty, an asynchronous error is generated by asserting **nECCERRIRQ**.

If an ECC error is detected. The error location (index and way) is recorded in the L2MRERRSR register.

See [L2 Memory Error Syndrome Register on page 4-87](#).

### 7.2.2 L2 hardware cache flush

The multiprocessor provides an efficient way to fully clean and invalidate the L2 cache in preparation for powerdown, without requiring the processor to be woken up to perform the clean and invalidate through software.

Use of L2 hardware cache flush can only occur if specific requirements are met and the following sequence applied:

1. Execute a DSB instruction to ensure completion of any prior prefetch requests.
2. All processors are in the WFI low-power state, so all the processor **STANDBYWFI** outputs are asserted.
3. When all outstanding ACP transactions are complete, the SoC asserts the **AINACTS** signal to idle the ACP. This is necessary to prevent ACP transactions from allocating new entries in the L2 cache while the hardware cache flush is occurring.
4. The SoC can now assert the **L2FLUSHREQ** input.

5. The L2 performs a series of internal clean and invalidate operations to each set and way of the L2 cache. Any dirty cache lines are written back to the system using WriteBack operations.
6. When the L2 completes the clean and invalidate sequence, it asserts the **L2FLUSHDONE** signal. The SoC can now deassert **L2FLUSHREQ** signal and then the L2 deasserts **L2FLUSHDONE**.
7. When all outstanding snoop transactions are completed, the SoC can assert the **ACINACTM** signal. In response, the L2 asserts the **STANDBYWFI2** signal.

Figure 7-1 shows the L2 hardware cache flush timing.

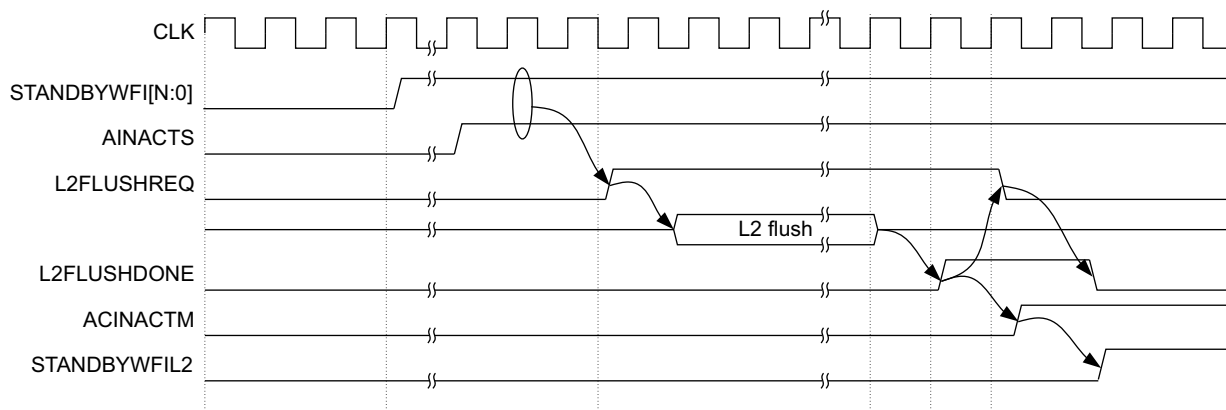


Figure 7-1 L2 hardware cache flush timing

### 7.2.3 L2 cache RAM latency settings

The L2 cache controller supports a range of data and tag RAM sizes. Each RAM size might have different timing requirements. To ensure that the L2 cache operates correctly with the data and tag RAMs implemented in your macrocell, you must set the appropriate latency values in **L2CTLR**. See [L2 Control Register on page 4-77](#).

There are three latency settings which must be applied to the L2 Cache data and tag RAMs:

#### Setup latency

The number of cycles that the driving signals are asserted before the L2 cache controller applies the rising edge of the clock to the RAM.

#### Write access latency

The time needed by the RAMs to internally compute the write control and data signals. It is the minimum number of cycles before the next rising clock edge is applied to the RAM, following a write access. The RAMs do not sample new data and control levels before this time has elapsed.

#### Read access latency

The minimum number of cycles before the next rising clock edge is applied to the RAM, following a read access. Read data is sampled in the last cycle.

#### ———— Note ————

- L2 tag RAM latencies are measured in CLK cycles.
- L2 data RAM latencies are measured in L2RAMCLK cycles.

The relationship between L2RAMCLKEN, L2RAMCLK and CLK is described in [Clocking and resets on page 2-8](#).

---

## 7.3 Cache coherency

Cortex-A17 MPCore processor supports between one and four individual processors with L1 data cache coherency maintained by the SCU.

The SCU maintains coherency between the individual data caches in the processor using ACE equivalents of a modified MOESI state:

<b>M</b>	Modified/UniqueDirty (UD). The line is only in this cache and is dirty.
<b>O</b>	Owned/SharedDirty (SD). The line is possibly in more than one cache and is dirty.
<b>E</b>	Exclusive/UniqueClean (UC). The line is only in this cache and is clean.
<b>S</b>	Shared/SharedClean (SC). The line is possibly in more than one cache. In this cache the line is not eligible for eviction.
<b>I</b>	Invalid/Invalid (I). The line is not in this cache.

The SCU contains buffers that can handle direct cache-to-cache transfers between processors without having to read or write any data to the external memory system. Cache line migration enables dirty cache lines to be transferred directly between processors while remaining in the MOESI modified state, and there is no requirement to write back transferred cache line data to the external memory system.

Each processor has tag and dirty RAMs that contain the MOESI state of the cache line. Rather than access these for each snoop request the SCU contains a set of duplicate tags that permit each coherent data request to be checked against the contents of the other caches in the cluster. The duplicate tags filter coherent requests from the system so that the processors and system can function efficiently even with a high volume of snoops from the system.

When an external snoop hits in the duplicate tags a request is made to the appropriate processor. The processor prioritizes external requests from the SCU over internal requests from the DPU.



## 7.4 ACE master interface

This section describes the properties of the ACE master interface. The ACE interface to the system can be clocked at integer ratios of the **CLK** frequency.

### 7.4.1 Memory interface attributes

[Table 7-1](#) shows the ACE master interface attributes for the Cortex-A17 MPCore processor.  $n$  is the number of processors implemented, in the range 1 to 4.

**Table 7-1 ACE master interface attributes**

Attribute	Value	Comments
Write issuing capability	$8 * n + 16$	Each processor can perform: <ul style="list-style-type: none"> <li>8 Strongly-ordered, Device or Non-cacheable write accesses.</li> </ul> The whole cluster capability is: <ul style="list-style-type: none"> <li><math>8 * n</math> Strongly-ordered, Device, Non-cacheable write accesses.</li> <li>16 cacheable write accesses.</li> </ul>
Read issuing capability	26 ( $n=1$ ) $10 * n + 32$ ( $n > 1$ )	Each processor can perform: <ul style="list-style-type: none"> <li>10 Strongly-ordered, Device or Non-cacheable data reads.</li> </ul> The whole cluster capability is: <ul style="list-style-type: none"> <li>16 cacheable reads, for a single processor implementation.</li> <li>32 cacheable reads, for two, or more processors implemented.</li> </ul>
Exclusive thread capability	$n$	Each processor can have 1 exclusive access sequence in progress.
Write ID capability	$26 * n + 16$	The Write ID capability is made up of: <ul style="list-style-type: none"> <li>1 for Strongly-ordered write or exclusive Device or Non-cacheable for each processor.</li> <li>1 for Device write for each processor.</li> <li>8 for Non-cacheable writes for each processor.</li> <li>16 for cacheable streams or maintenance evictions for each processor.</li> <li>16 for cacheable L2 evictions.</li> </ul>
Write ID width	8	The ID encodes the source of the memory transaction. See <a href="#">Table 7-2 on page 7-8</a> .
Read ID capability	$18 * n + 8^a + 16$	The Read ID capability is made up of: <ul style="list-style-type: none"> <li>1 for Strongly-ordered, Device or Non-cacheable data reads for each processor.</li> <li>4 for Non-cacheable instructions reads for each processor.</li> <li>1 for TLB Non-cacheable reads for each processor.</li> <li>8 for data linefills for each processor.</li> <li>4 for instr linefills for each processor.</li> <li>8 for data linefills for ACP.</li> <li>16 for L2 prefetches.</li> </ul> If the L2 cache is not present the read ID capability value changes to $10n + 2$ .
Read ID width	8	The ID encodes the source of the memory transaction. See <a href="#">Table 7-3 on page 7-9</a> .

a. If ACP active.

### 7.4.2 ACE transfers

Cortex-A17 does not generate any FIXED bursts. All WRAP bursts fetch a complete cache line starting with the critical word first. A burst does not cross a cache line boundary.

The cache linefill fetch length is always 64 bytes.

The Cortex-A17 MPCore processor generates only a subset of all possible AXI transactions on the master interface.

For Write-Back (inner cacheable) transfers the supported transfers are:

- WRAP 4 128-bit for read transfers.
- INCR 4 128-bit for write transfers.

For Non-cacheable transactions:

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

For Device or Strongly-ordered transactions:

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

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

The following points apply to AXI transactions:

- WRAP bursts are only 128-bit.
- INCR 1 can be any size up to 128-bits for read or write.
- INCR burst, more than one transfer, are only 32-bit or 128-bit.
- No transaction is marked as FIXED.
- Write transfers with none, some, or all byte strobes low can occur.

### 7.4.3 AXI transaction IDs

The ACE master interface and the peripheral port use the same AXI ID encodings. The peripheral port implements a subset of these codes.

The AXI ID signal encodings are described in [Table 7-2](#) and [Table 7-3 on page 7-9](#).

[Table 7-2](#) shows the encoding for **AWIDM[7:0]** and **AWIDMP[7:0]**

**Table 7-2 Encoding for AWIDM[7:0] and AWIDMP[7:0]**

AWID	Comments
[7:5]	Source:
0b000	Processor 0.
0b001	Processor 1.
0b010	Processor 2.
0b011	Processor 3.
0b100	ACP <sup>a</sup> .
0b111	L2 <sup>a</sup> .
[4:0]	Used for internal tracking purposes only.

a. The peripheral port does not implement this value.

Table 7-3 shows the encoding for **ARIDM[7:0]** and **ARIDMP[7:0]**

**Table 7-3 Encoding for ARIDM[7:0] and ARIDMP[7:0]**

ARID	Comments
[7:5]	Source:
0b000	Processor 0.
0b001	Processor 1.
0b010	Processor 2.
0b011	Processor 3.
0b100	ACP <sup>a</sup> .
0b111	L2 <sup>a</sup> .
[4:0]	Used for internal tracking purposes only.

a. The peripheral port does not implement this value.

#### 7.4.4 Write response

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

#### 7.4.5 AXI privilege information

AXI provides information about the privilege level of an access on the **ARPROTM[2:0]** and **AWPROTM[2:0]** signals. However, when accesses might be cached or merged together, the resulting transaction can have both privileged and user data combined. If this happens, the Cortex-A17 MPCore processor marks the transaction as privileged, even if it was initiated by a user process.

Table 7-4 shows the processor modes and corresponding **ARPROTM[2:0]** and **AWPROTM[2:0]** values.

**Table 7-4 Processor mode and ARPROT and AWPROT values**

Processor mode	Type of access	Value of ARPROT[1] and AWPROT[1]
PL0, PL1, PL2	Cacheable read access	Privileged access
PL0	Device, Strongly-ordered, or normal Non-cacheable read access	Normal access
PL1, PL2		Privileged access
PL0, PL1, PL2	Cacheable write access	Privileged access
PL0	Device or Strongly-ordered write	Normal access
PL1, PL2		Privileged access
PL0	Normal Non-cacheable write, except for STREX, STREXB, STREXH, and STREXD to shareable memory	Privileged access

Table 7-4 Processor mode and ARPROT and AWPROT values (continued)

Processor mode	Type of access	Value of ARPROT[1] and AWPROT[1]
PL0	Normal Non-cacheable write for STREX, STREXB, STREXH, and STREXD to shareable memory	Normal access
PL1, PL2	Normal Non-cacheable write	Privileged access
PL0, PL1, PL2	TLB pagewalk	Privileged access

#### 7.4.6 AXI3 Compatibility mode

The Cortex-A17 MPCore processor implements an AXI3 compatibility mode that enables you to use the processor in a standalone environment where the AMBA 4 ACE interface is not required and the processor does not propagate barriers outside of the cluster. To enable this mode you must set the **SYSBARDISABLE** input pin to 1 on the boundary of the processor. When **SYSBARDISABLE** is set to 1, you must ensure the **BROADCASTINNER**, **BROADCASTOUTER**, and **BROADCASTCACHEMAINT** input pins are set to 0.

#### 7.4.7 ACE configuration signals

The Cortex-A17 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-A17 MPCore processor.

Table 7-5 Supported ACE configurations

Signal	Feature		
	AXI3 mode	ACE non-coherent <sup>a</sup>	ACE inner coherent
<b>SYSBARDISABLE</b> <sup>b</sup>	-	-	-
<b>BROADCASTCACHEMAINT</b> <sup>c</sup>	0	0	0
<b>BROADCASTOUTER</b> <sup>d</sup>	0	0	1
<b>BROADCASTINNER</b>	0	0	1

a. ACE non-coherent is compatible with connecting to an ACE-Lite interconnect.

b. Cortex-A17 does not broadcast barriers. **SYSBARDISABLE** has no effect. ARM recommends setting this signal HIGH.

c. **BROADCASTCACHEMAINT** is always LOW.

d. **BROADCASTOUTER** and **BROADCASTINNER** must always be equal.

Table 7-6 on page 7-11 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	ACE inner coherent
AXI3 compliance	Y	N	N
AXI4 and ACE compliance	N	Y	Y
Barriers on AR and AW channels	N	N	N
Cache maintenance requests on AR channel	N	N	N
Snoops on AC channel	N	N	Y
Coherent requests on AR or AW channel	N	N	Y
DVM requests on AR channel	N	N	Y

## 7.5 ACP

*Accelerator Coherency Port* (ACP) is implemented as an AXI3 slave interface that supports the following features:

- 128-bit read and write data buses.
- INCR burst type only.

---

### Note

---

- ACP does not support Non-cacheable accesses, only cacheable accesses are supported.
  - ACP does not support FIXED burst type transactions to Normal memory.
  - The ACP does not support write interleaving. The **WIDS** signals, required in an AXI3 slave interface to support write interleaving, are not implemented.
- 

### 7.5.1 Accelerator Coherency Port Interface Restrictions

The requirements of the ACP interface are:

- Read data bus width and write data bus width must be 128-bit.
- All transactions have burst size equal to the data bus width, **AxSIZE** = 0b100.
- All transaction burst addresses must be aligned to the burst length.
- All transactions have burst type INCR, **AxBURST** = 0b01.
- Exclusive transactions are not supported, **AxLOCK** = 0b00.
- All transactions are memory type Write-back with read and write allocation hint, **AxCACHE[3:0]** = 0b1111.
- All transactions are permitted to be secure or non-secure, **AxPROT** can take any value.
- A transaction ID width of up to 5 bits is supported.
- Write interleaving is not supported.
- Two burst lengths are supported. All bursts must be aligned.
  - 16 byte burst, **AxLEN** = 0b00000000 and **AxADDR[3:0]** = 0b0000.
  - 64 byte burst, **AxLEN** = 0b00000011 and **AxADDR[5:0]** = 0b000000.
- For write bursts:
  - 16-byte write bursts are permitted to have any combination of valid bytes. However, write bursts with no valid bytes are not supported.
  - 64-byte write bursts must have all bytes valid.

### 7.5.2 ACP requests

All read and write requests performed on the ACP must be cacheable. If the request type, in **ARCACHESC[3:0]** or **AWCACHESC[3:0]**, is not cacheable write-back, allocate on both reads and writes (0b1111), SLVERR is asserted in **RRESPC[1:0]** or **BRESPC[1:0]**.

Requests are treated as coherent requests:

#### ACP coherent read requests

In this case, the SCU enforces coherency.

When the data is present in one of the processors within multiprocessor the data is read directly from the relevant processor and returned to the ACP.

When the data is not present in any of processors, the read request is issued to the L2. If data is present in L2 cache, it is returned to the ACP. If the data is not present, the L2 cache performs a linefill before returning the data.

### ACP coherent write requests

In this case, the SCU enforces coherency.

When the data is present in one of the processors within the multiprocessor the data is first cleaned and invalidated from the relevant processor.

When the data is not present in any of the processors, or when it has been cleaned and invalidated, the write request is issued to the L2. In the case of a L2 cache hit the line is updated. On a miss a line is allocated in L2 cache.

#### 7.5.3 ACP read and write acceptance

<b>Read</b>	Read acceptance 8 cacheable reads. To ensure the maximum acceptance, ARM recommends that you use the lower 8 ID values.
<b>Write</b>	Write acceptance 1 cacheable write. ARM recommends that you use a different ID from any pending read in progress.

## 7.6 Peripheral port

The peripheral port is an AXI master interface. It can be mapped into the Secure and Non-secure physical address spaces.

The peripheral port supports the following memory types and attributes:

- Normal, Non-cacheable.
- Strongly-ordered.
- Device.

You must ensure that the peripheral port memory region is correctly defined. Accesses to memory types not supported by the peripheral port generates an abort.

### 7.6.1 Mapping the peripheral port

Two pairs of configuration input buses map the peripheral port into physical address space.

For Non-secure access, **CFGADDRFILSTARTNS[39:20]** is the start of the peripheral port region and **CFGADDRFILTENDNS[39:20]** is the end of the region. Mapping is enabled by **CFGADDRFILTENNS**.

For Secure access, **CFGADDRFILSTARTS[39:20]** is the start of the peripheral port region and **CFGADDRFILTENDS[39:20]** is the end of the region. Mapping is enabled by **CFGADDRFILTENS**.

Both pairs of buses are sampled at reset, any subsequent changes are ignored.

See [Configuration signals on page A-4](#).

The address regions can be read by software using the FILASTARTR and FILAENDR system control registers. See [Peripheral port start address register on page 4-85](#) and [Peripheral port end address register on page 4-86](#).

### 7.6.2 Peripheral port read issuing capability

<b>Read</b>	10 Strongly-ordered, Device or Non-cacheable reads per processor.
<b>Write</b>	8 Strongly-ordered, Device or Non-cacheable writes per processor.

### 7.6.3 Inter-dependencies between the Peripheral Port and the ACE master interface

Any transfer initiated on either of these ports must be able to complete independently of any accesses made on the other port.



## 7.7 External aborts and asynchronous errors

### 7.7.1 External aborts

The L2 memory system handles external aborts depending on the instruction executed and attributes of the memory region of the access:

- External aborts on writes to Non-cacheable Normal memory or Device memory are indicated by asserting the **nAXIERRIRQ** signal.
- External aborts on evictions from the caches are indicated by asserting the **nAXIERRIRQ** signal.
- All other external aborts are reported to the processor as asynchronous imprecise aborts.

You can configure external aborts to trap to Monitor mode by setting the EA bit in the Secure Configuration Register to 1. See [Secure Configuration Register on page 4-58](#) for more information.

### 7.7.2 Asynchronous errors

The L2 memory system has two outputs that indicate asynchronous error conditions. An asynchronous external error condition exists when either:

- The **nAXIERRIRQ** output is asserted LOW.
- The **nECCERRIRQ** output is asserted LOW.

If an asynchronous error condition is detected, the corresponding bit in the L2 Extended Control Register is asserted. The asynchronous error condition can be cleared by writing 0 to the corresponding bit of the L2ECTLR. Software can only clear the L2ECTLR. Any attempt to assert the error by writing the L2ECTLR is ignored. See [L2 Extended Control Register on page 4-81](#) for more information.

External errors on the ACE write response channel or on the ACE read data channel, associated with a write-allocate memory transaction, cause the **nAXIERRIRQ** signal to be asserted LOW.

Double-bit ECC errors cause the **nECCERRIRQ** signal to be asserted LOW.

Any external error associated with a load instruction is reported back to the requestor along with an error response and this might trigger an abort.

# Chapter 8

## Generic Timer

This chapter describes the Generic Timer for the Cortex-A17 MPCore processor. It contains the following sections:

- *About the Generic Timer on page 8-2.*
- *Generic Timer functional description on page 8-3.*
- *Timer programmers model on page 8-4.*

## 8.1 About the Generic Timer

The Generic Timer can schedule events and trigger interrupts based on an incrementing counter value. It provides:

- Generation of timer events as interrupt outputs.
- Generation of event streams.
- Support for Virtualization Extensions.

The Generic Timer is compliant with the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

This chapter only describes features that are specific to the Cortex-A17 MPCore processor implementation.

## 8.2 Generic Timer functional description

The Cortex-A17 MPCore processor provides a set of four timers for each processor in the cluster.

- Physical Timer for use in Secure and Non-secure PL1 modes. The registers for the Physical Timer are banked to provide Secure and Non-secure copies.
- Virtual Timer for use in Non-secure PL1 modes.
- Physical Timer for use in Hyp mode.

The counter value is distributed to the processor with a synchronous binary encoded 64-bit bus, **CNTVALUEB[63:0]**.

[Table A-5 on page A-7](#) describes the Generic Timer signals.

### 8.3 Timer programmers model

Within each processor, a set of Timer registers are allocated to the CP15 coprocessor space. The Timer registers are either 32 bits wide or 64 bits wide.

Table 8-1 shows the Generic Timer registers.

**Table 8-1 Generic Timer registers**

CRn	Op1	CRm	Op2	Name	Reset	Width	Description
c14	0	c0	0	CNTFRQ	UNK	32-bit	Counter Frequency Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			c1	CNTKCTL	-a	32-bit	Counter Non-secure PL1 Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c2	0	CNTP_TVAL	UNK	32-bit	Counter PL1 Physical Timer Value Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	CNTP_CTL	-b	32-bit	Counter PL1 Physical Timer Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
	4	c3	0	CNTV_TVAL	UNK	32-bit	Counter PL1 Virtual Timer Value Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	CNTV_CTL	-b	32-bit	Counter PL1 Virtual Timer Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
		c1	0	CNTHCTL	-c	32-bit	Counter Non-secure PL2 Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			c2	CNTHP_TVAL	UNK	32-bit	Counter Non-secure PL2 Physical Timer Value Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
			1	CNTHP_CTL	-b	32-bit	Counter Non-secure PL2 Physical Timer Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

a. The reset value for bits [2:0] is 0b000.

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

c. The reset value for bit [2] is 0 and for bits [1:0] is 0b11.

# Chapter 9

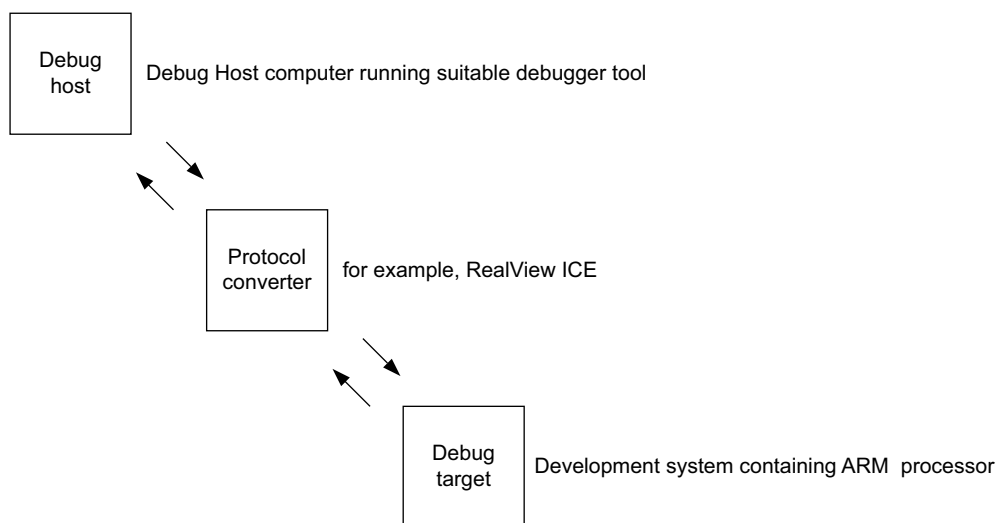
## Debug

This chapter describes debug, the registers that it uses and examples of how to use it. It contains the following sections:

- *About debug* on page 9-2.
- *Debug register interfaces* on page 9-4.
- *Debug register summary* on page 9-6.
- *Debug register descriptions* on page 9-10.
- *Debug events* on page 9-30.
- *External debug interface* on page 9-31.

## 9.1 About debug

This section gives an overview of debug and describes the debug components. The processor forms one component of a debug system. Figure 9-1 shows a typical system.



**Figure 9-1 Typical debug system**

This typical system has several parts:

- *Debug host.*
- *Protocol converter.*
- *Debug target.*
- *The debug unit.*

### 9.1.1 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.

### 9.1.2 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.

### 9.1.3 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.

### 9.1.4 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.



## 9.2 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 9-31](#).

This section describes:

- [Processor interfaces](#).
- [Breakpoints and watchpoints](#).
- [Effects of resets on debug registers](#).

### 9.2.1 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 9-31](#).

#### 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 9-31](#).

#### Trace registers

This interface is memory-mapped. See [External debug interface on page 9-31](#).

### 9.2.2 Breakpoints and watchpoints

The processor supports six breakpoints, four watchpoints, and a standard *Debug Communications Channel* (DCC).

A breakpoint consists of a breakpoint control register and a breakpoint value register. These two registers are referred to as a *Breakpoint Register Pair* (BRP).

Four of the breakpoints (BRP 0 to 3) match only to virtual address and the other two (BRP 4 and 5) match against either virtual address or context ID, or *Virtual Machine Identifier* (VMID). All the watchpoints can be linked to two breakpoints (BRP 4 and 5) to enable a memory request to be trapped in a given process context.

### 9.2.3 Effects of resets on debug registers

The processor has the following reset signals that affect the debug registers:

**nCOREPORESET[N:0]** This signal initializes all processor logic, including the debug, breakpoint, watchpoint logic, and performance monitors logic.

**nCORERESET[N:0]** This signal initializes the processor logic, including the performance monitors logic.

**nCOREPRESETDBG[N:0]** This signal resets the processor debug logic, including the breakpoint and watchpoint logic. Performance monitors logic is not affected.

**nTOPPRESETDBG** Resets the shared Debug-APB, the top debug logic, the CTI and CTM logic within the PCLKDBG clock domain.

The whole debug part of the processor is reset by asserting LOW **nCOREPRESETDBG** and **nTOPPRESETDBG**.

### 9.3 Debug register summary

Table 9-1 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. Each processor has an address range for the debug registers mapped onto the External debug interface. The offset references a register within a processor. See Table 9-22 on page 9-31.

**Table 9-1 CP14 debug register summary**

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
0	0x000	c0	0	c0	0	DBGDIDR	RO	<a href="#">Debug Identification Register on page 9-10</a>
1	0x004	c0	0	c1	0	DBGDSCR internal view	RO	Debug Status and Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
2-4	0x008-0x010	-	-	-	-	-	-	Reserved
5	0x014	c0	0	c5	0	DBGDTRTX internal view	WO	Target to Host Data Transfer Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
						DBGDTRRX internal view	RO	Host to Target Data Transfer Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
6	0x018	c0	0	c6	0	DBGWFAR	-	RAZ/WI
7	0x01C	c0	0	c7	0	DBGVCR	RW	Vector Catch Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
8	0x020	-	-	-	-	-	-	Reserved
9	0x024	-	-	-	-	DBGECR	RW	Event Catch Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
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 Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

Table 9-1 CP14 debug register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
33	0x084	-	-	-	-	DBGITR	WO	Instruction Transfer Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
						DBGPCSR	RO	<a href="#">Program Counter Sampling Register on page 9-11</a>
34	0x088	c0	0	c2	2	DBGDSCR external view	RW	Debug Status and Control Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
35	0x08C	c0	0	c3	2	DBGDTRTX external view	RW	Target to Host Data Transfer Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
36	0x090	-	-	-	-	DBGDRCR	WO	<a href="#">Debug Run Control Register on page 9-12</a>
37	0x094	-	-	-	-	-	-	Reserved
38-39	0x098-0x09C	-	-	-	-	-	-	Reserved
40	0x0A0	-	-	-	-	DBGPCSR	RO	<a href="#">Program Counter Sampling Register on page 9-11</a>
41	0x0A4	-	-	-	-	DBGCIDSR	RO	Context ID Sampling Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
42	0x0A8	-	-	-	-	DBGVIDSR	RO	Virtualization ID Sampling Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
43-63	0x0AC-0x0FC	-	-	-	-	-	-	Reserved
64-69	0x100-0x114	c0	0	c0-c5	4	DBGBVRn	RW	<a href="#">Breakpoint Value Registers on page 9-13</a>
70-71	0x118-0x11C	-	-	-	-	-	-	Reserved
72-79	0x120-0x13C	-	-	-	-	-	-	Reserved
80-85	0x140-0x154	c0	0	c0-c5	5	DBGBCRn	RW	<a href="#">Breakpoint Control Registers on page 9-14</a>
86-95	0x158-0x17C	-	-	-	-	-	-	Reserved
96-99	0x180-0x18C	c0	0	c0-c3	6	DBGWVRn	RW	<a href="#">Watchpoint Value Registers on page 9-16</a>
100-111	0x190-0x1BC	-	-	-	-	-	-	Reserved

Table 9-1 CP14 debug register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
112-115	0x1C0-0x1CC	c0	0	c0-c3	7	DBGWCRn	RW	<a href="#">Watchpoint Control Registers on page 9-17</a>
116-127	0x1D0-0x1FC	-	-	-	-	-	-	Reserved
128	0x200	c1	0	c0	0	DBGDRAR(MRC)	RO	<a href="#">Debug ROM Address Register on page 9-19</a>
128	0x200	-	0	c1	-	DBGDRAR(MRRC)		<a href="#">Debug ROM Address Register on page 9-19</a>
129-147	0x204-0x24C	-	-	-	-	-	-	Reserved
148-149	0x250-0x254	c1	0	c4-c5	1	DBGBXVRn	RW	<a href="#">Breakpoint Extended Value Registers on page 9-20</a>
150-191	0x258-0x2FC	-	-	-	-	-	-	Reserved
192	0x300	c1	0	c0	4	DBGOSLAR	WO	<a href="#">OS Lock Access Register on page 9-21</a>
193	0x304	c1	0	c1	4	DBGOSLSR	RO	<a href="#">OS Lock Status Register on page 9-22</a>
194	0x308	-	-	-	-	-	-	Reserved
195	0x30C	c1	0	c3	4	DBGOSDLR	RW	OS Double Lock Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
196	0x310	c1	0	c4	4	DBGPRCR	RW	<a href="#">Device Powerdown and Reset Control Register on page 9-23</a>
197	0x314	-	-	-	-	DBGPRSR	RO	Device Powerdown and Reset Status Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
198-255	0x318-0x03C	-	-	-	-	-	-	Reserved
256	0x400	c2	0	c0	0	DBGDSAR(MRC)	RO	<a href="#">Debug Self Address Offset Register on page 9-25</a>
256	0x400	-	0	c2	-	DBGDSAR(MRRC)	RO	<a href="#">Debug Self Address Offset Register on page 9-25</a>
257-511	0x404-0x7FC	-	-	-	-	-	-	Reserved
512-575	0x800-0x8FC	-	-	-	-	-	-	Reserved
576-831	0x900-0xCFC	-	-	-	-	-	-	Reserved
832-895	0xD00-0xDFC	-	-	-	-	Processor ID registers	RO	Processor ID registers, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
896-999	0xE00-0xF9C	-	-	-	-	-	-	Reserved

Table 9-1 CP14 debug register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
1000	0xFA0	c7	0	c8	6	DBGCLAIMSET	RW	<a href="#">Claim Tag Set Register on page 9-26</a>
1001	0xFA4	c7	0	c9	6	DBGCLAIMCLR	RW	<a href="#">Claim Tag Clear Register on page 9-26</a>
1002-1003	0xFA8-0xFAC	-	-	-	-	-	-	Reserved
1004	0xFB0	-	-	-	-	DBGLAR	WO	Lock Access Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
1005	0xFB4	-	-	-	-	DBGLSR	RO	Lock Status Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
1006	0xFB8	c7	0	c14	6	DBGAUTHSTATUS	RO	Authentication Status Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
1007	0xFBC	-	-	-	-	-	-	Reserved
1008	0xFC0	c7	0	c0	7	DBGDEVID2	RO	UNK
1009	0xFC4	c7	0	c1	7	DBGDEVID1	RO	<a href="#">Debug Device ID Register 1 on page 9-27</a>
1010	0xFC8	c7	0	c2	7	DBGDEVID0	RO	<a href="#">Debug Device ID Register 0 on page 9-28</a>
1011	0xFCC	-	-	-	-	DBGDEVTYPE	RO	Device Type Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
1012	0xFD0	-	-	-	-	DBGPID4	RO	<a href="#">Debug Peripheral Identification Registers on page 9-29</a>
1013-1015	0xFD4-0xFDC	-	-	-	-	DBGPID5-7	-	Reserved
1016	0xFE0	-	-	-	-	DBGPID0	RO	<a href="#">Debug Peripheral Identification Registers on page 9-29</a>
1017	0xFE4	-	-	-	-	DBGPID1	RO	
1018	0xFE8	-	-	-	-	DBGPID2	RO	
1019	0xFEC	-	-	-	-	DBGPID3	RO	
1020	0xFF0	-	-	-	-	DBGCID0	RO	<a href="#">Debug Component Identification Registers on page 9-29</a>
1021	0xFF4	-	-	-	-	DBGCID1	RO	
1022	0xFF8	-	-	-	-	DBGCID2	RO	
1023	0xFFC	-	-	-	-	DBGCID3	RO	

# 9.4 Debug register descriptions

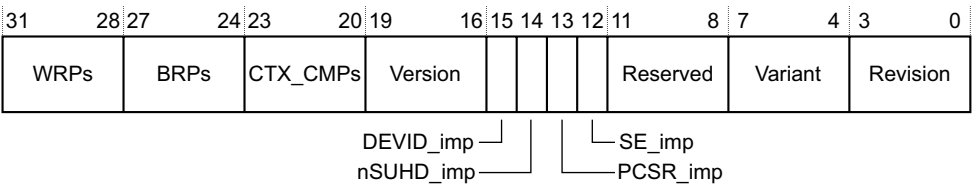
This section describes the debug registers. [Table 9-1 on page 9-6](#) provides cross-references to individual registers.

## 9.4.1 Debug Identification 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.  
 Accessible when the processor is powered down.  
 See [Debug Device ID Register 1 on page 9-27](#) and [Debug Device ID Register 0 on page 9-28](#) for more information about the debug implementation.
- Configurations** Available in all configurations.
- Attributes** See the register summary in [Table 9-1 on page 9-6](#).

[Figure 9-2](#) shows the DBGDIDR bit assignments.



**Figure 9-2** DBGDIDR bit assignments

[Table 9-2](#) shows the DBGDIDR bit assignments.

**Table 9-2** DBGDIDR bit assignments

Bits	Name	Function
[31:28]	WRPs	Indicates the number of <i>Watchpoint Register Pairs</i> (WRPs) implemented: 0x3 The processor implements 4 WRPs.
[27:24]	BRPs	Indicates the number of <i>Breakpoint Register Pairs</i> (BRPs) implemented: 0x5 The processor implements 6 BRPs.
[23:20]	CTX_CMPs	Indicates the number of BRPs that can be used for Context matching: 0x1 The processor implements 2 breakpoints with Context matching.
[19:16]	Version	Indicates the Debug architecture version: 0x5 The processor implements ARMv7.1 Debug architecture.
[15]	DEVID_imp	Indicates if the Debug Device ID Register (DBGDEVID) is implemented: 1 DBGDEVID is implemented.
[14]	nSUHD_imp	Indicates if the Secure User Halting Debug is implemented: 1 Secure User halting debug is not implemented.

Table 9-2 DBGDIDR bit assignments (continued)

Bits	Name	Function
[13]	PCSR_imp	Indicates if the Program Counter Sampling Register (DBGPCSR) implemented as register 33: <b>1</b> DBGPCSR is implemented as register 33.
[12]	SE_imp	Security Extensions implemented bit: <b>1</b> 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 <a href="#">Main ID Register on page 4-27</a> 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 <a href="#">Main ID Register on page 4-27</a> for more information.

#### 9.4.2 Program Counter Sampling Register

The DBGPCSR characteristics are:

<b>Purpose</b>	Enables a debugger to sample the <i>Program Counter</i> (PC).
<b>Usage constraints</b>	ARM deprecates reading a PC sample through register 33 when the DBGPCSR is also implemented as register 40.
<b>Configurations</b>	DBGPCSR is implemented as both debug register 33 and 40.
<b>Attributes</b>	See the register summary in <a href="#">Table 9-1 on page 9-6</a> .

[Figure 9-3](#) shows the DBGPCSR bit assignments.

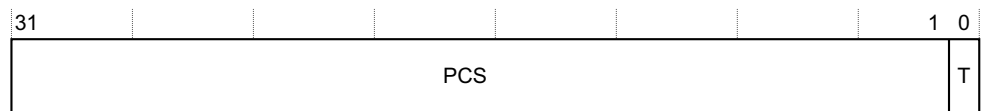


Figure 9-3 DBGPCSR bit assignments

[Table 9-3](#) shows the DBGPCSR bit assignments.

Table 9-3 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-A17 MPCore processor, no offset is applied.
[0]	T	This bit indicates whether the sampled address is an ARM instruction, or a Thumb or ThumbEE instruction: <b>0</b> DBGPCSR[1] = 0, the sampled address, DBGPCSR[31:0], is an ARM instruction. <b>1</b> (DBGPCSR[31:1] << 1) is the address of the sampled Thumb or ThumbEE instruction. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.

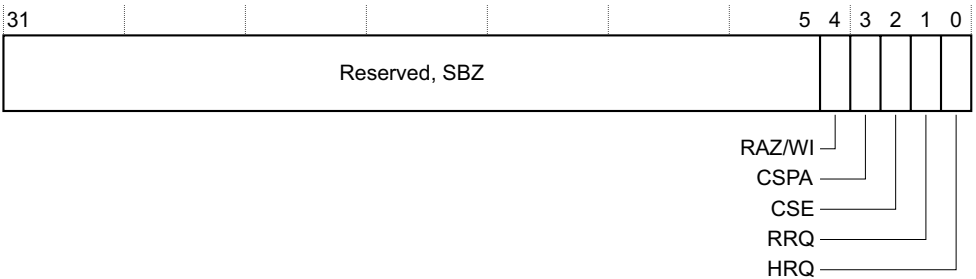


### 9.4.3 Debug Run Control Register

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.
- Usage constraints** The DBGDRCR is accessible when the processor is powered down. DBGDRCR is not accessible on extended CP14 interface.
- Configurations** Required in all implementations.
- Attributes** See the register summary in [Table 9-1 on page 9-6](#).

[Figure 9-4](#) shows the DBGDRCR bit assignments.



**Figure 9-4** DBGDRCR bit assignments

[Table 9-4](#) shows the DBGDRCR bit assignments.

**Table 9-4** DBGDRCR bit assignments

Bits	Name	Function
[31:5]	-	SBZ.
[4]	-	RAZ/WI.
[3]	CSPA	Clear Sticky Pipeline Advance bit. This bit sets the DBGDSCR.PipeAdv bit to 0. The actions on writing to this bit are: <ul style="list-style-type: none"> <li><b>0</b> No action.</li> <li><b>1</b> Sets the DBGDSCR.PipeAdv bit to 0.</li> </ul> Writes to this bit are ignored when the processor is powered down or OSDLK is set.

Table 9-4 DBGDSCR bit assignments (continued)

Bits	Name	Function
[2]	CSE	<p>Clear Sticky Exceptions bits. This bit sets the DBGDSCR sticky exceptions bits to 0. The actions on writing to this bit are:</p> <p><b>0</b> No action.</p> <p><b>1</b> Sets the DBGDSCR[8:6] to 0b000.</p> <p>See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information on the DBGDSCR.</p>
[1]	RRQ	<p>Restart Request. The actions on writing to this bit are:</p> <p><b>0</b> No action.</p> <p><b>1</b> Request exit from Debug state.</p> <p>Writing 1 to this bit requests that the processor exits Debug state. This request is held until the processor exits Debug state. Once the request has been made, the debugger can poll the DBGDSCR.RESTARTED bit until it reads as 1.</p> <p>The processor ignores writes to this bit if it is in Non-debug state.</p>
[0]	HRQ	<p>Halt Request. The actions on writing to this bit are:</p> <p><b>0</b> No action.</p> <p><b>1</b> Request exit from Debug state.</p> <p>Writing 1 to this bit requests that the processor enters Debug state. This request is held until the processor enters Debug state.</p> <p>Once the request has been made, the debugger can poll the DBGDSCR.HALTED bit until it reads 1.</p> <p>The processor ignores writes to this bit if it is already in Debug state, or if invasive debug is disabled.</p> <p>See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information on the DBGDSCR.HRQ and the Halting debug events.</p>

#### 9.4.4 Breakpoint Value Registers

The DBGBVR characteristics are:

<b>Purpose</b>	Holds a value for use in breakpoint matching, either an instruction address or a Context ID.
<b>Usage constraints</b>	Used in conjunction with a DBGBCR, see <a href="#">Breakpoint Control Registers on page 9-14</a> . Each DBGBVR is associated with a DBGBCR to form a <i>Breakpoint Register Pair</i> (BRP). DBGBVRn is associated with DBGBCRn to form BRPn.
<b>Configurations</b>	The processor implements 6 BRPs, and is specified by the DBGDIDR.BRPs field, see <a href="#">Debug Identification Register on page 9-10</a> .
<b>Attributes</b>	See the register summary in <a href="#">Table 9-1 on page 9-6</a> . The debug logic reset value of a DBGBVR is UNK.

[Figure 9-5 on page 9-14](#) shows the DBGBVR bit assignments.



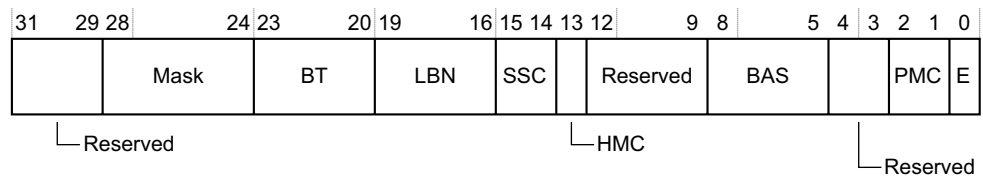


Figure 9-6 DBGBCR bit assignments

Table 9-7 shows the DBGBCR bit assignments.

Table 9-7 DBGBCR bit assignments

Bits	Name	Function
[31:29]	-	Reserved.
[28:24]	Mask	Address mask. The processor does not support address range masking.
[23:20]	BT	<p>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.</p> <p>See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for the breakpoint type values and meanings.</p>
[19:16]	LBN	<p>Linked Breakpoint Number. If this BRP 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 ID to be used in the combined instruction address and Context matching. Otherwise, this field must be programmed to 0b0000.</p> <p>Reading this register returns an UNKNOWN value for this field, and the generation of Breakpoint debug events is UNPREDICTABLE, if either:</p> <ul style="list-style-type: none"> <li>this breakpoint is not programmed for Linked instruction address match or mismatch and this field is not programmed to 0b0000.</li> <li>this breakpoint is programmed for Linked instruction address match or mismatch and the BRP indicated by this field does not support Context matching or is not programmed for Linked Context ID match.</li> </ul> <p>See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.</p>
[15:14]	SSC	<p>Security State Control. This field enables the watchpoint to be conditional on the security state of the processor. This field is used with the <i>Hyp Mode Control</i> (HMC), and <i>Privileged Mode Control</i> (PMC), fields. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for possible values of the fields, and the mode and security states that can be tested.</p> <p>This field must be programmed to 0b00 if DBGBCR.BT is programmed for Linked Context match. If this is not done, the generation of debug events by this breakpoint is unpredictable.</p> <p style="text-align: center;"><b>Note</b></p> <p>When this field is set to a value other than 0b00, the SSC field controls the processor security state in which the access matches, not the required security attribute of the access.</p>
[13]	HMC	<p>Hyp Mode Control bit. This field is used with the SSC and PMC fields. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> 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.</p>
[12:9]	-	Reserved.

### Table 9-7 DBGBCR bit assignments (continued)

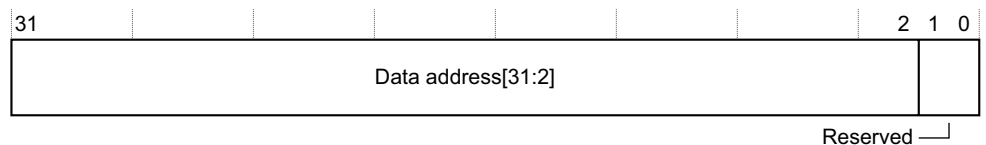
Bits	Name	Function
[8:5]	BAS	<p>Byte Address Select. This field enables match or mismatch comparisons on only certain bytes of the word address held in the DBGBVR. The operation of this field depends also on:</p> <ul style="list-style-type: none"> <li>The DBGBVR meaning field being programmed for instruction address match or mismatch.</li> <li>The Address range mask field being programmed to 0b00000, no mask.</li> <li>The instruction set state of the processor, indicated by the CPSR.J and CPSR.T bits.</li> </ul> <p>This field must be programmed to 0b1111 if either:</p> <ul style="list-style-type: none"> <li>The DBGBVR meaning field, bits[23:20], is programmed for Linked or Unlinked Context ID match.</li> <li>The Address range mask field, bits[28:24], is programmed to a value other than 0b00000.</li> </ul> <p>If this is not done, the generation of Breakpoint debug events is UNPREDICTABLE.</p>
[4:3]	-	Reserved.
[2:1]	PMC	<p>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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for possible values of the fields, and the mode and security states that can be tested.</p>
[0]	E	<p>Breakpoint Enable. This field enables the BRP:</p> <p><b>0</b> BRP disabled.</p> <p><b>1</b> BRP enabled.</p> <p>A BRP never generates a Breakpoint debug event when it is disabled.</p>

### 9.4.6 Watchpoint Value Registers

The DBGWVR characteristics are:

<b>Purpose</b>	Holds a data address value for use in watchpoint matching. The address used must be the virtual address of the data.
<b>Usage constraints</b>	Used in conjunction with a DBGWCR, see <a href="#">Watchpoint Control Registers on page 9-17</a> . Each DBGWVR is associated with a DBGWCR to form a <i>Watchpoint Register Pair</i> (WRP). DBGWVRn is associated with DBGWCRn to form WRPn.
<b>Configurations</b>	The processor implements four WRPs, and is specified by the DBGDIDR.WRPs field, see <a href="#">Debug Identification Register on page 9-10</a> .
<b>Attributes</b>	See the register summary in <a href="#">Table 9-1 on page 9-6</a> . The debug logic reset value of a DBGWVR is UNK.

Figure 9-7 shows the DBGWVR bit assignments.



**Figure 9-7 DBGWVR bit assignments**

### Table 9-8 DBGWVR bit assignments

### Figure 9-8 DBGWCR bit assignments

Table 9-9 shows the DBGWCR bit assignments.

**Table 9-9 DBGWCR bit assignments**

Bits	Name	Function																				
[31:29]	-	Reserved.																				
[28:24]	Mask	<p>Address mask. The processor supports watchpoint address masking. This field can be used to 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 0b00001 and 0b00010 are reserved. Therefore, the meaning of Watchpoint Address range mask values are:</p> <table><tr><td>0b00000</td><td>No mask.</td></tr><tr><td>0b00001</td><td>Reserved.</td></tr><tr><td>0b00010</td><td>Reserved.</td></tr><tr><td>0b00011</td><td>0x00000007, mask for data address, three bits masked.</td></tr><tr><td>0b00100</td><td>0x0000000F, mask for data address, four bits masked.</td></tr><tr><td>0b00101</td><td>0x0000001F, mask for data address, five bits masked.</td></tr><tr><td>-</td><td></td></tr><tr><td>-</td><td></td></tr><tr><td>-</td><td></td></tr><tr><td>0b11111</td><td>0x7FFFFFFF, mask for data address, 32 bits masked.</td></tr></table>	0b00000	No mask.	0b00001	Reserved.	0b00010	Reserved.	0b00011	0x00000007, mask for data address, three bits masked.	0b00100	0x0000000F, mask for data address, four bits masked.	0b00101	0x0000001F, mask for data address, five bits masked.	-		-		-		0b11111	0x7FFFFFFF, mask for data address, 32 bits masked.
0b00000	No mask.																					
0b00001	Reserved.																					
0b00010	Reserved.																					
0b00011	0x00000007, mask for data address, three bits masked.																					
0b00100	0x0000000F, mask for data address, four bits masked.																					
0b00101	0x0000001F, mask for data address, five bits masked.																					
-																						
-																						
-																						
0b11111	0x7FFFFFFF, mask for data address, 32 bits masked.																					
[23:21]	-	Reserved.																				
[20]	WT	<p>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:</p> <table><tr><td>0</td><td>Unlinked data address match.</td></tr><tr><td>1</td><td>Linked data address match.</td></tr></table> <p>When this bit is set to 1 the linked breakpoint number field indicates the breakpoint to which this watchpoint is linked. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.</p>	0	Unlinked data address match.	1	Linked data address match.																
0	Unlinked data address match.																					
1	Linked data address match.																					
[19:16]	LBN	<p>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 0b0000.</p> <p>Reading this register returns an UNKNOWN value for this field, and the generation of Watchpoint debug events is UNPREDICTABLE, if either:</p> <ul style="list-style-type: none"><li>This watchpoint does not have linking enabled and this field is not programmed to 0b0000.</li><li>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.</li></ul> <p>See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.</p>																				
[15:14]	SSC	<p>Security State Control. This field enables the watchpoint to be conditional on the security state of the processor. This field is used with the HMC and <i>Privileged Access Control</i> (PAC) fields.</p> <p>See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for possible values of the fields, and the access modes and security states that can be tested.</p>																				
[13]	HMC	<p>Hyp Mode Control. This field is used with the SSC and PAC fields. The value of DBGWCR.PAC has no effect for accesses made in Hyp mode.</p> <p>See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for possible values of the fields, and the access modes and security states that can be tested.</p>																				
[12:9]	-	RAZ/WI.																				





Figure 9-10 shows the DBGDRAR bit assignments as a 64-bit register.

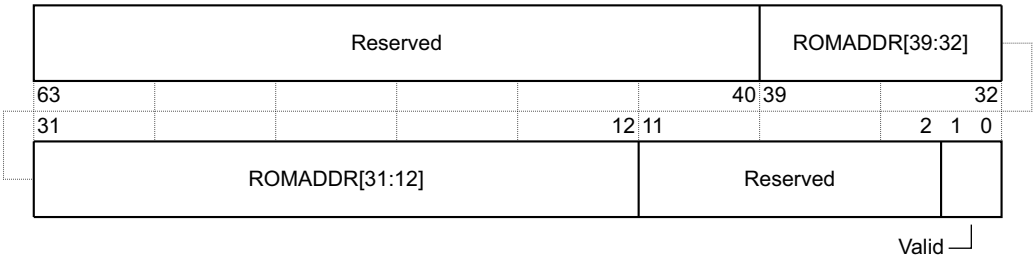


Figure 9-10 DBGDRAR 64-bit assignments

Table 9-10 shows the DBGDRAR bit assignments.

Table 9-10 DBGDRAR bit assignments

Bits	Name	Function
[63:40]	-	Reserved.
[39:32]	ROMADDR[39:32]	Bits[39:32] of the ROM table 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 ROM table physical address. Bits[11:0] of the address are zero. 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 ROM table address is valid: <div> <div>0b00</div> <div>ROM table address is not valid.</div> </div> <div> <div>0b11</div> <div>ROM table address is valid.</div> </div>
<p><b>Note</b></p> <p>ROMADDRV must be set to 1 if ROMADDR[39:12] is set to a valid value.</p>		

### 9.4.9 Breakpoint Extended Value Registers

The DBG BXVR characteristics are:

<b>Purpose</b>	Holds a value for use in VMID matching for BRP that supports Context matching.
<b>Usage constraints</b>	Used in conjunction with a DBGBCR, see <a href="#">Breakpoint Control Registers on page 9-14</a> , and <a href="#">Breakpoint Value Registers on page 9-13</a> . Each DBG BXVR is associated with a DBGBCR and DBG BVR to form a <i>Breakpoint Register Pair</i> (BRP). DBG BXVRn is associated with DBGBCRn and DBG BVRn to form BRPn.
<b>Configurations</b>	The processor implements two BRPs, BRP 4 and 5, that can be used for Context matching, and is specified by the DBGDIDR.CTX_CMPs field, see <a href="#">Debug Identification Register on page 9-10</a> .
<b>Attributes</b>	See the register summary in <a href="#">Table 9-1 on page 9-6</a> .

Figure 9-11 on page 9-21 shows the DBG BXVR bit assignments.

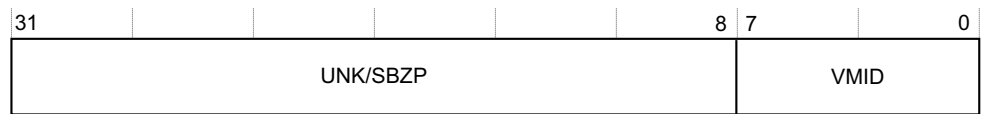


Figure 9-11 DBGBXVR bit assignments

Table 9-11 shows the DBGBXVR bit assignments.

Table 9-11 DBGBXVR bit assignments

Bits	Name	Function
[31:8]	-	UNK/SBZP.
[7:0]	VMID	VMID value. Used to compare with the <i>Virtual Machine ID</i> (VMID) field, held in the <i>Virtualization Translation Table Base Register</i> (VTTBR). See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.

#### 9.4.10 OS Lock Access Register

The DBGOSLAR characteristics are:

<b>Purpose</b>	Provides a lock for the debug registers and disables Software debug events.
<b>Usage constraints</b>	Use DBGOSLSR to check the current status of the lock, see <a href="#">OS Lock Status Register on page 9-22</a> . Not accessible when the processor is powered down.
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 9-1 on page 9-6</a> .

Figure 9-12 shows the DBGOSLAR bit assignments.

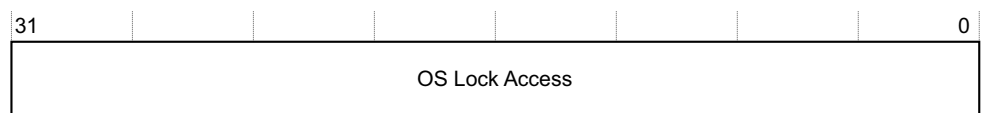


Figure 9-12 DBGOSLAR bit assignments

Table 9-12 shows the DBGOSLAR bit assignments.

Table 9-12 DBGOSLAR bit assignments

Bits	Name	Function
[31:0]	OS Lock Access	Writing the key value 0xC5ACCE55 to this field locks the debug registers for the OS Save or Restore operation. Writing any other value to this register unlocks the debug registers if they are locked. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.

9.4.11 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, reading DBGOSLSR.OSLM returns zero.
- Usage constraints

There are no usage constraints.  
Accessible when the processor is powered down.
- Configurations

Available in all configurations.
- Attributes

See the register summary in [Table 9-1 on page 9-6](#).

[Figure 9-13](#) shows the DBGOSLSR bit assignments.

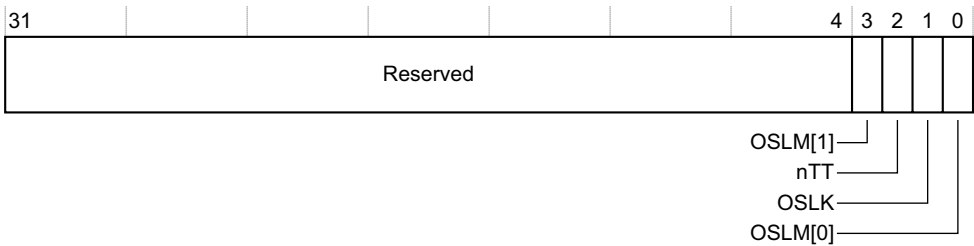


Figure 9-13 DBGOSLSR bit assignments

[Table 9-13](#) shows the DBGOSLSR bit assignments.

Table 9-13 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: 0b10           The processor implements the OS Lock Model but does not implement DBGOSSRR.  —— <b>Note</b> —— This field splits across bits [3] and [0].

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	<p>This bit indicates the status of the OS Lock:</p> <p><b>0</b> Lock not set.</p> <p><b>1</b> Lock set.</p> <p>The OS Lock is set or cleared by writing to the DBGOSLAR, see <i>OS Lock Access Register on page 9-21</i>. The OS Lock is set to 1 on a core powerup reset.</p> <p>Setting the OS Lock restricts access to Debug registers. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.</p> <p>This bit is UNKNOWN when the processor is powered down or OSDLK is set.</p>
[0]	OSLM[0]	<p>OS Lock Model implemented bit. This field identifies the form of OS Save and Restore mechanism implemented:</p> <p>0b10 The processor implements the OS Lock Model but does not implement DBGOSSRR.</p> <p><b>Note</b></p> <p>This field splits across bits [3] and [0].</p>

The DBGPRCR characteristics are:

**Usage constraints** Accessible when the processor is powered down.

ARM deprecates using the Extended CP14 interface to write to bit [1] of the DBGPRCR. Bits [3:2] are not defined for Extended CP14 interface.

**Attributes** See the register summary in [Table 9-1 on page 9-6](#).

Diagram of the 32-bit COREPDRQ register structure:

- Bit 31: Reserved
- Bit 30: Reserved
- Bit 29: Reserved
- Bit 28: Reserved
- Bit 27: Reserved
- Bit 26: Reserved
- Bit 25: Reserved
- Bit 24: Reserved
- Bit 23: Reserved
- Bit 22: Reserved
- Bit 21: Reserved
- Bit 20: Reserved
- Bit 19: Reserved
- Bit 18: Reserved
- Bit 17: Reserved
- Bit 16: Reserved
- Bit 15: Reserved
- Bit 14: Reserved
- Bit 13: Reserved
- Bit 12: Reserved
- Bit 11: Reserved
- Bit 10: Reserved
- Bit 9: Reserved
- Bit 8: Reserved
- Bit 7: Reserved
- Bit 6: Reserved
- Bit 5: Reserved
- Bit 4: COREPURQ
- Bit 3: HCWR
- Bit 2: CWRR
- Bit 1: CORENPDRQ
- Bit 0: Reserved

Table 9-14 on page 9-24 shows the DBGPRCR bit assignments.

Table 9-14 DBGPRCR bit assignments

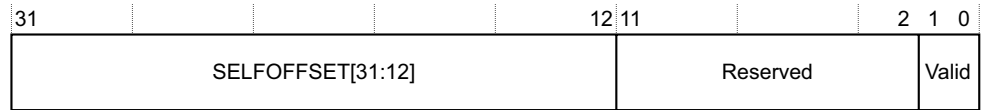
Bits	Name	Function
[31:4]	-	Reserved, UNK/SBZP.
[3]	COREPURQ	<p>Core Powerup Request bit. This bit enables a debugger to request that the power controller powers up the core, enabling access to the debug registers in the core power domain:</p> <p><b>0</b> <b>DBGPWRUPREQ</b> is LOW, this is the reset value.</p> <p><b>1</b> <b>DBGPWRUPREQ</b> 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.</p> <p>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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.</p>
[2]	HCWR	<p>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 powerup or warm reset:</p> <p><b>0</b> Does not hold the non-debug logic in reset on a powerup or warm reset.</p> <p><b>1</b> Holds the non-debug logic of the processor in reset on a powerup or warm reset. The processor is held in this state until this bit is cleared to 0.</p> <p>For accesses to DBGPRCR from CP14, this bit is UNK/SBZP.</p> <p>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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.</p>
[1]	CWRR	<p>Core Warm Reset Request bit. Writing 1 to this bit issues a request for a warm reset:</p> <p><b>0</b> No action.</p> <p><b>1</b> Request internal reset by asserting the <b>DBGIRSTREQ</b> output HIGH. This bit is reset to 0 by when the warm reset is completed.</p> <p>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:</p> <ul style="list-style-type: none"> <li>• The core power domain is powered down.</li> <li>• DBGPRSR.DLK, OS Double Lock status bit, is set to 1.</li> <li>• For the external debug interface, the OS lock is set.</li> </ul> <p>See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.</p>
[0]	CORENPDRQ	<p>Core No Powerdown Request bit. When set to 1, the <b>DBGNOPWRDWN</b> 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 and PTM are not actually powered down when requested by software or hardware handshakes:</p> <p><b>0</b> <b>DBGNOPWRDWN</b> is LOW. This is the reset value.</p> <p><b>1</b> <b>DBGNOPWRDWN</b> is HIGH.</p> <p>This bit is UNKNOWN on reads and ignores writes when any of the following apply:</p> <ul style="list-style-type: none"> <li>• The core power domain is powered down. If the CORENPDRQ bit is 1, it loses this value through the powerdown.</li> <li>• DBGPRSR.DLK, OS Double Lock status bit is set to 1.</li> <li>• For the external debug interface, the OS Lock is set.</li> </ul>

### 9.4.13 Debug Self Address Offset Register

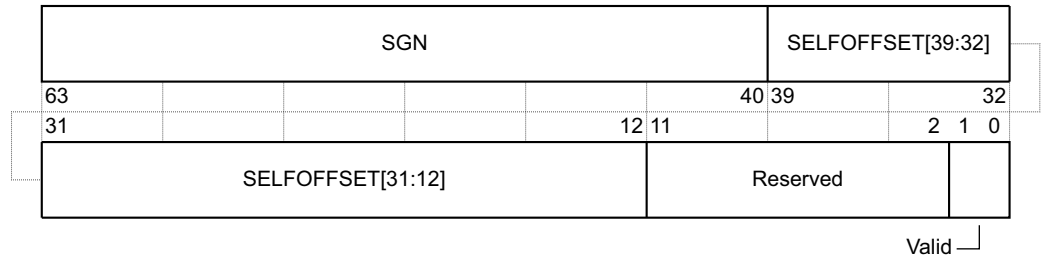
The DBGDSAR characteristics are:

- 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 9-19](#).
- Usage constraints** This register is only visible in the CP14 interface.
- Configurations** The DBGDSAR is:
- a 64-bit register accessible using MRRC instruction
  - a 32-bit register available in a 32-bit view using MRC instruction.
- Attributes** See the register summary in [Table 9-1 on page 9-6](#). This register is only visible in the CP14 interface, and therefore does not have a memory offset.

[Figure 9-15](#) shows the DBGDSAR bit assignments as a 32-bit register.



**Figure 9-15 DBGDSAR 32-bit assignments**



**Figure 9-16 DBGDSAR 64-bit assignments**

[Table 9-15](#) shows the DBGDSAR bit assignments.

**Table 9-15 DBGDSAR bit assignments**

Bits	Name	Function
[63:40]	SGN	Sign extension.
[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 <a href="#">Debug ROM Address Register on page 9-19</a> . If Valid[1:0] = 0b00, value of this field is UNKNOWN.
[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 <a href="#">Debug ROM Address Register on page 9-19</a> . If Valid = 0b00, the value of this field is UNKNOWN.
[11:2]	-	Reserved.
[1:0]	Valid	Valid bits. This field indicates whether the debug self address offset is valid: 0b00 Offset is not valid. 0b11 Offset is valid.

### 9.4.14 Claim Tag Set Register

The DBGCLAIMSET characteristics are:

<b>Purpose</b>	Used by software to set CLAIM bits to 1.
<b>Usage constraints</b>	Use in conjunction with the DBGCLAIMCLR register. See <a href="#">DBGCLAIMCLR bit assignments on page 9-27</a> . Not accessible when the processor is powered down.
<b>Configurations</b>	The processor implements bits [7:0] as claim tags.
<b>Attributes</b>	See the register summary in <a href="#">Table 9-1 on page 9-6</a> .

[Figure 9-17](#) shows the DBGCLAIMSET bit assignments.



**Figure 9-17 DBGCLAIMSET bit assignments**

[Table 9-16](#) shows the DBGCLAIMSET bit assignments.

**Table 9-16 DBGCLAIMSET bit assignments**

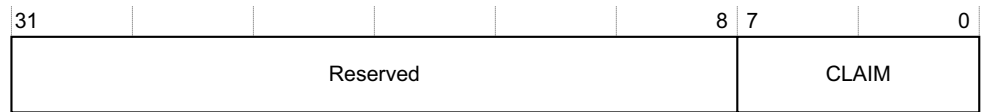
Bits	Name	Function
[31:8]	-	Reserved.
[7:0]	CLAIM	CLAIM bits. Writing a 1 to one of these bits sets the corresponding CLAIM bit to 1. A single write operation can set multiple bits to 1. Writing 0 to one of these bits has no effect. The CLAIM bits do not have any specific functionality. ARM expects the 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 <a href="#">Claim Tag Clear Register</a> for information on how to: <ul style="list-style-type: none"> <li>Clear CLAIM bits to 0.</li> <li>Read the current values of the CLAIM bits.</li> </ul>

### 9.4.15 Claim Tag Clear Register

The DBGCLAIMCLR characteristics are:

<b>Purpose</b>	Used by software to read the values of the CLAIM bits, and to clear these bits to zero.
<b>Usage constraints</b>	Use in conjunction with the DBGCLAIMSET register. See <a href="#">DBGCLAIMSET bit assignments</a> . Not accessible when the processor is powered down.
<b>Configurations</b>	The processor implements bits [7:0] as claim tags.
<b>Attributes</b>	See the register summary in <a href="#">Table 9-1 on page 9-6</a> .

[Figure 9-18 on page 9-27](#) shows the DBGCLAIMCLR bit assignments.



### Figure 9-18 DBGCLAIMCLR bit assignments

Table 9-17 shows the DBGCLAIMCLR bit assignments.

### Table 9-17 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 and how they might be used, see <a href="#">Claim Tag Set Register on page 9-26</a> .

#### 9.4.16 Debug Device ID Register 1

The DBGDEVID1 characteristics are:

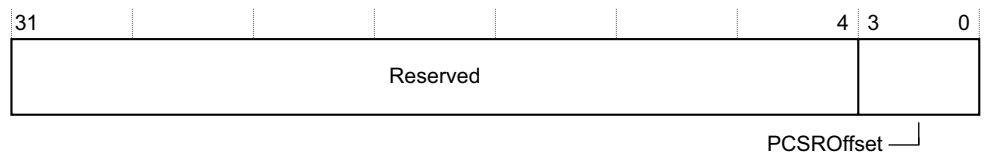
<b>Purpose</b>	Adds to the information given by the DBGDIDR by describing other features of the debug implementation.
----------------	--

<b>Usage constraints</b>	There are no usage constraints. Accessible when the processor is powered down.
--------------------------	---

<b>Configurations</b>	Available in all configurations.
-----------------------	----------------------------------

**Attributes** See the register summary in [Table 9-18](#).

Figure 9-19 shows the DBGDEVID1 bit assignments.



**Figure 9-19 DBGDEVID1 bit assignments**

Table 9-18 shows the DBGDEVID1 bit assignments.

### Table 9-18 DBGDEVID1 bit assignments

Bits	Name	Function
[31:4]	-	Reserved.
[3:0]	PCSROffset	Defines the offset applied to DBGPCSR samples: 0b0001 DBGPCSR samples have no offset applied.

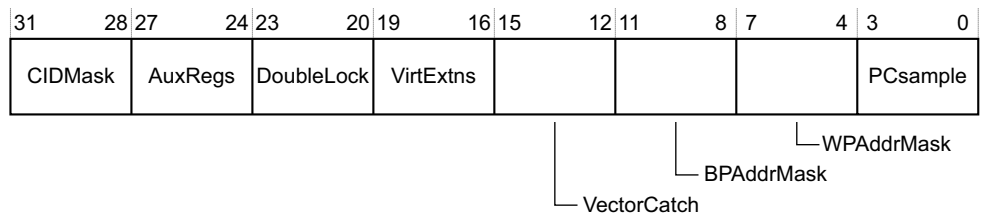


### 9.4.17 Debug Device ID Register 0

The DBGDEVID0 characteristics are:

<b>Purpose</b>	Extends the DBGDIDR by describing other features of the debug implementation.
<b>Usage constraints</b>	Use in conjunction with DBGDIDR to find the features of the debug implementation. See <a href="#">DBGDIDR bit assignments on page 9-10</a> . Accessible when the processor is powered down.
<b>Configurations</b>	Available in all configurations.
<b>Attributes</b>	See the register summary in <a href="#">Table 9-1 on page 9-6</a> .

Figure 9-20 shows the DBGDEVID0 bit assignments.



**Figure 9-20 DBGDEVID0 bit assignments**

Table 9-19 shows the DBGDEVID0 bit assignments.

**Table 9-19 DBGDEVID0 bit assignments**

Bits	Name	Function
[31:28]	CIDMask	This field indicates the level of support for the Context ID matching breakpoint masking capability. 0b0000 Context ID masking not implemented.
[27:24]	AuxRegs	Specifies support for the Debug External Auxiliary Control Register. 0b0000 The processor does not support the Debug External Auxiliary Control Register.
[23:20]	DoubleLock	Specifies support for the Debug OS Double Lock Register: 0b0001 The processor supports Debug OS Double-lock Register.
[19:16]	VirExtns	Specifies the implementation of the Virtualization Extensions to the Debug architecture: 0b0001 The processor implements the Virtualization Extensions to the Debug architecture.
[15:12]	VectorCatch	Defines the form of the vector catch event implemented: 0b0000 The processor implements address matching form of vector catch.
[11:8]	BPAAddrMask	Indicates the level of support for the <i>Immediate Virtual Address</i> (IVA) matching breakpoint masking capability: 0b1111 Breakpoint address masking not implemented. DBGBCRn[28:24] are UNK/SBZP.
[7:4]	WPAAddrMask	Indicates the level of support for the DVA matching watchpoint masking capability: 0b0001 Watchpoint address mask implemented.
[3:0]	PCSample	Indicates the level of support for Program Counter sampling using debug registers 40 and 41: 0b0011 DBGPCSR, DBGCIDSr and DBGVIDSR are implemented as debug registers 40, 41, and 42.

### 9.4.18 Debug Peripheral Identification Registers

The Debug 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 9-20](#).

**Table 9-20 Summary of the Debug Peripheral Identification Registers**

Register	Value	Offset
Debug Peripheral ID4	0x04	0xFD0
Debug Peripheral ID5	0x00	0xFD4
Debug Peripheral ID6	0x00	0xFD8
Debug Peripheral ID7	0x00	0xFDC
Debug Peripheral ID0	0x0E	0xFE0
Debug Peripheral ID1	0xBC	0xFE4
Debug Peripheral ID2	0x0B	0xFE8
Debug Peripheral ID3	0x00	0xFEC

Only bits [7:0] of each Debug Peripheral ID Register are used, with bits [31:8] reserved. Together, the eight Debug Peripheral ID Registers define a single 64-bit Debug Peripheral ID.

The *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* describes these registers.

### 9.4.19 Debug Component Identification Registers

There are four read-only Debug Component Identification Registers, Component ID0 through Component ID3. [Table 9-21](#) shows these registers.

**Table 9-21 Summary of the Component Identification Registers**

Register	Value	Offset
Debug Component ID0	0x0D	0xFF0
Debug Component ID1	0x90	0xFF4
Debug Component ID2	0x05	0xFF8
Debug Component ID3	0xB1	0xFFC

The Debug 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.

## 9.5 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.

### 9.5.1 Watchpoint debug events

In the 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.

### 9.5.2 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.

### 9.5.3 Debug OS lock

Debug OS Lock is set by the reset, **nCOREPORESET[N:0]**, see [Resets on page 2-12](#). 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*.

## 9.6 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 9-21 shows the debug interface implemented in each processor. For more information on these signals, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

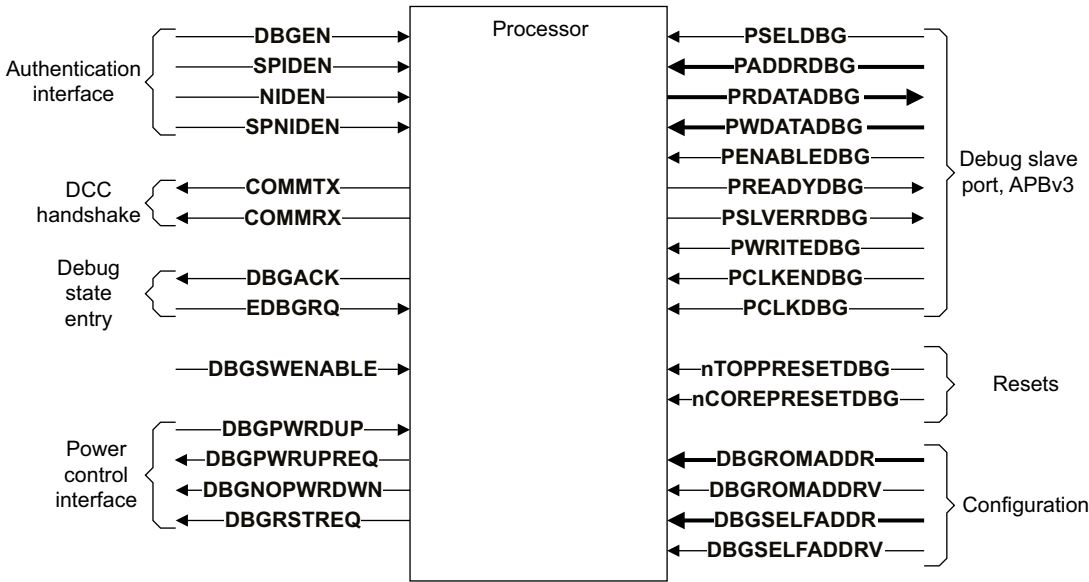


Figure 9-21 External debug interface, including APBv3 slave port

### 9.6.1 Memory map

The basic memory map supports up to four processors in a multiprocessor device. Table 9-22 shows the address mapping for the debug trace components.

Table 9-22 Address mapping for debug trace components

Address range	Component <sup>a</sup>
0x00000 - 0x00FFF	ROM table
0x01000 - 0x0FFFF	Reserved
0x10000 - 0x10FFF	Processor 0 Debug
0x11000 - 0x11FFF	Processor 0 PMU
0x12000 - 0x12FFF	Processor 1 Debug
0x13000 - 0x13FFF	Processor 1 PMU
0x14000 - 0x14FFF	Processor 2 Debug
0x15000 - 0x15FFF	Processor 2 PMU
0x16000 - 0x16FFF	Processor 3 Debug
0x17000 - 0x17FFF	Processor 3 PMU
0x18000 - 0x18FFF	Processor 0 CTI
0x19000 - 0x19FFF	Processor 1 CTI

Table 9-22 Address mapping for debug trace components (continued)

Address range	Component <sup>a</sup>
0x1A000 - 0x1AFFF	Processor 2 CTI
0x1B000 - 0x1BFFF	Processor 3 CTI
0x1C000 - 0x1CFFF	Processor 0 Trace
0x1D000 - 0x1DFFF	Processor 1 Trace
0x1E000 - 0x1EFFF	Processor 2 Trace
0x1F000 - 0x1FFFF	Processor 3 Trace

a. Indicates the mapped component if present, otherwise reserved.

### 9.6.2 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.

If software running on the processor has control over an external device that drives the authentication signals, it must make the change using a safe sequence:

1. Execute a sequence of instructions, specific to your system, to change the signal value. For example, this might be a single STR instruction that writes certain value to a control register in a system peripheral.
2. If step 1 involves any memory operation, issue a DSB instruction.
3. Poll the DBGDSCR or Authentication Status Register to check whether the processor has already detected the changed value of these signals. This is required because the system might not issue the signal change to the processor until several cycles after the DSB instruction completes.
4. Issue an ISB instruction exception entry or exception return.

The software cannot perform debug or analysis operations that depend on the new value of the authentication signals until this procedure is complete. The same rules apply when the debugger has control of the processor through the Instruction Transfer Register, DBGITR, while in Debug state. The relevant combinations of the **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN** values can be determined by polling DSCR[17:16], DSCR[15:14], or the Authentication Status Register.

# Chapter 10

## Performance Monitoring Unit

This chapter describes the *Performance Monitoring Unit* (PMU) and the registers that it uses. It contains the following sections:

- [\*About the Performance Monitoring Unit\* on page 10-2.](#)
- [\*PMU functional description\* on page 10-3.](#)
- [\*PMU registers summary\* on page 10-4.](#)
- [\*PMU register descriptions\* on page 10-7.](#)
- [\*Events\* on page 10-10.](#)
- [\*Interrupts\* on page 10-14.](#)
- [\*Exporting PMU events\* on page 10-15.](#)

## 10.1 About the Performance Monitoring Unit

Each processor in the Cortex-A17 MPCore processor includes a PMU that implements the ARM-v7 *Performance Monitors Extension version 2* (PMUv2) architecture. The PMU provides information about the behavior of the processor that you can use when debugging or profiling code.

The PMU provides six counters. You can program each counter to count any of the performance monitoring events available in the processor.

---

**Note**

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.

---

## 10.2 PMU functional description

This section describes the functionality of the PMU in:

- *Event interface.*
- *CP15 and APB interface.*
- *Counters.*

Figure 10-1 shows the major blocks inside the PMU.

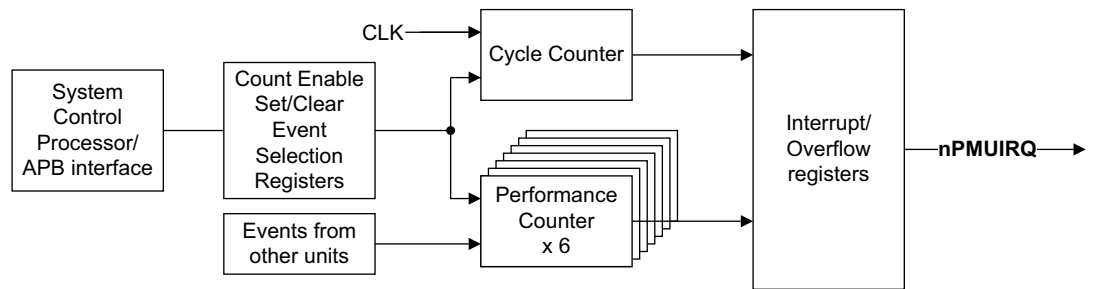


Figure 10-1 PMU block diagram

### 10.2.1 Event interface

Events from all other units from across the design are provided to the PMU.

### 10.2.2 CP15 and APB interface

The PMU registers are programmed using the CP15 system control coprocessor or external APB interface.

### 10.2.3 Counters

The PMU has:

- Six 32-bit counters that are incremented by enabled events.
- One cycle counter that increments on the processor clock.



## 10.3 PMU registers summary

The PMU counters and their associated control registers are accessible from the internal CP15 interface and from the Debug APB interface.

Table 10-1 gives a summary of the Cortex-A17 MPCore processor PMU registers.

**Table 10-1 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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
1	0x004	c9	0	c13	2	PMXEVCNTR1	RW	
2	0x008	c9	0	c13	2	PMXEVCNTR2	RW	
3	0x00C	c9	0	c13	2	PMXEVCNTR3	RW	
4-30	0x010-0x78	-	-	-	-	-	-	Reserved
31	0x07C	c9	0	c13	0	PMCCNTR	RW	Cycle Count Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
32-255	0x080-0x3FC	-	-	-	-	-	-	Reserved
256	0x400	c9	0	c13	1	PMXEVTYPER0	RW	Event Type Selection Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
257	0x404	c9	0	c13	1	PMXEVTYPER1	RW	
258	0x408	c9	0	c13	1	PMXEVTYPER2	RW	
259	0x40C	c9	0	c13	1	PMXEVTYPER3	RW	
258-286	0x410-0x478	-	-	-	-	-	-	Reserved
287	0x47C	c9	0	c13	1	PMXEVTYPER31	RW	Performance Monitors Event TypeSelect Register 31, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
288-767	0x480-0xBFC	-	-	-	-	-	-	Reserved
768	0xC00	c9	0	c12	1	PMCNTENSET	RW	Count Enable Set Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
769-775	0xC04-0xC1C	-	-	-	-	-	-	Reserved
776	0xC20	c9	0	c12	2	PMCNTENCLR	RW	Count Enable Clear Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
777-783	0xC24-0xC3C	-	-	-	-	-	-	Reserved
784	0xC40	c9	0	c14	1	PMINTENSET	RW	Interrupt Enable Set Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
785-791	0xC44-0xC5C	-	-	-	-	-	-	Reserved

Table 10-1 PMU register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
792	0xC60	c9	0	c14	2	PMINTENCLR	RW	Interrupt Enable Clear Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
793-799	0xC64-0xC7C	-	-	-	-	-	-	Reserved
800	0xC80	c9	0	c12	3	PMOVSr	RW	Overflow Flag Status Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
801-807	0xC84-0xC9C	-	-	-	-	-	-	Reserved
808	0xCA0	c9	0	c12	4	PMSWINC	WO	Software Increment Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
809-895	0xCA4-0xDFC	-	-	-	-	-	-	Reserved
896	0xE00	-	-	-	-	PMCFGR	RO	Performance Monitor Configuration Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
897	0xE04	c9	0	c12	0	PMCR	RW	<a href="#">Performance Monitor Control Register on page 10-7</a>
898	0xE08	c9	0	c14	0	PMUSERENR	RW	User Enable Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
899-903	0xE0C-0xE1C	-	-	-	-	-	-	Reserved
904	0xE20	c9	0	c12	6	PMCEID0	RO	Common Event Identification Register 0, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
905	0xE24	c9	0	c12	7	PMCEID1	RO	Common Event Identification Register 1, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
906-1003	0xE28-0xFAC	-	-	-	-	-	-	Reserved
1004	0xFB0	-	-	-	-	PMLAR	WO	Lock Access Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
1005	0xFB4	-	-	-	-	PMLSR	RO	Lock Status Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
1006	0xFB8					PMAUTHSTATUS	RO	Authentication Status Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>

Table 10-1 PMU register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
1007-1010	0xFBC-0xFC8	-	-	-	-	-	-	Reserved
1011	0xFCC	-	-	-	-	PMDEVTYPE	RO	Device Type Register, see the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i>
1012	0xFD0	-	-	-	-	PMPID4	RO	<i>Performance Monitors Peripheral Identification Registers on page 10-8</i>
1013	0xFD4	-	-	-	-	PMPID5	RO	
1014	0xFD8	-	-	-	-	PMPID6	RO	
1015	0xFDC	-	-	-	-	PMPID7	RO	
1016	0xFE0	-	-	-	-	PMPID0	RO	
1017	0xFE4	-	-	-	-	PMPID1	RO	<i>Performance Monitors Component Identification Registers on page 10-9</i>
1018	0xFE8	-	-	-	-	PMPID2	RO	
1019	0xFEC	-	-	-	-	PMPID3	RO	
1020	0xFF0	-	-	-	-	PMCID0	RO	
1021	0xFF4	-	-	-	-	PMCID1	RO	
1022	0xFF8	-	-	-	-	PMCID2	RO	
1023	0xFFC	-	-	-	-	PMCID3	RO	

## 10.4 PMU register descriptions

This section describes the Cortex-A17 MPCore processor PMU registers. [Table 10-1 on page 10-4](#) provides cross-references to individual PMU registers.

### 10.4.1 Performance Monitor Control Register

The PMCR characteristics are:

- |                          |  |
|--------------------------|--|
| <b>Purpose</b>           | <ul style="list-style-type: none"> <li>Provides details of the performance monitor implementation, including the number of counters implemented.</li> <li>Configures and controls the counters.</li> </ul>   |
| <b>Usage constraints</b> | <p>The PMCR is:</p> <ul style="list-style-type: none"> <li>A read/write register.</li> <li>Common to the Secure and Non-secure states.</li> <li>Accessible from PL1 or higher.</li> <li>Accessible in User mode only when the PMUSERENR.EN bit is set to 1.</li> </ul> |
| <b>Configurations</b>    | Available in all configurations.   |
| <b>Attributes</b>        | See the register summary in <a href="#">Table 10-1 on page 10-4</a> .  |

[Figure 10-2](#) shows the PMCR bit assignments.

31		24	23		16	15		11	10		6	5	4	3	2	1	0
IMP				IDCODE				N		Reserved		DP	X	D	C	P	E

**Figure 10-2 Performance Monitor Control Register bit assignments**

[Table 10-2](#) shows the PMCR bit assignments.

**Table 10-2 PMCR bit assignments**

Bits	Name	Function
[31:24]	IMP	Implementer code. 0x41 ARM. This is a read-only field.
[23:16]	IDCODE	Identification code. 0x0E Cortex-A17 MPCore processor identification code. This is a read-only field.
[15:11]	N	Number of event counters. In Secure state and Hyp mode, this field returns 0x6 that indicates the number of counters implemented. In Non-secure modes other than Hyp mode, this field reads the value of HDCR.HPMN. See <a href="#">Hyp Debug Control Register on page 4-64</a> . This is a read-only field.
[10:6]	-	Reserved, UNK/SBZP.

**Table 10-2 PMCR bit assignments (continued)**

Bits	Name	Function
[5]	DP	<p>Disable cycle counter, PMCCNTR, in regions of software when prohibited:</p> <p><b>0</b> Count is enabled in prohibited regions. This is the reset value.</p> <p><b>1</b> Count is disabled in prohibited regions.</p> <p>This bit is read/write.</p>
[4]	X	<p>Export enable. This bit permits events to be exported to another debug device, such as a trace macrocell, over an event bus:</p> <p><b>0</b> Export of events is disabled. This is the reset value.</p> <p><b>1</b> Export of events is enabled.</p> <p>This bit is read/write.</p>
[3]	D	<p>Clock divider:</p> <p><b>0</b> When enabled, PMCCNTR counts every clock cycle. This is the reset value.</p> <p><b>1</b> When enabled, PMCCNTR counts once every 64 clock cycles.</p> <p>This bit is read/write.</p>
[2]	C	<p>Clock counter reset:</p> <p><b>0</b> No action. This is the reset value.</p> <p><b>1</b> Reset PMCCNTR to 0.</p> <p>This bit is write-only, and always RAZ.</p>
[1]	P	<p>Event counter reset:</p> <p><b>0</b> No action. This is the reset value.</p> <p><b>1</b> Reset all event counters, not including PMCCNTR, to 0.</p> <p>In Non-secure modes other than Hyp mode, writing a 1 to this bit does not reset event counters that the HDCR.HPMN field reserves for Hyp mode use. See <a href="#">Hyp Debug Control Register on page 4-64</a>.</p> <p>In Secure state and Hyp mode, writing a 1 to this bit resets all event counters.</p> <p>This bit is write-only, and always RAZ.</p>
[0]	E	<p>Enable bit.</p> <p><b>0</b> All counters, including PMCCNTR, are disabled. This is the reset value.</p> <p><b>1</b> All counters are enabled.</p> <p>This bit is read/write.</p>

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

## 10.4.2 Performance Monitors Peripheral Identification Registers

The Performance Monitors Peripheral Identification Registers provide standard information required for all components that conform to the PMUv2 architecture. They are a set of eight registers, listed in register number order in [Table 10-3](#).

**Table 10-3 Summary of the Performance Monitors Peripheral Identification Registers**

Register	Value	Offset
Performance Monitors Peripheral ID4	0x04	0xFD0
Performance Monitors Peripheral ID5	0x00	0xFD4
Performance Monitors Peripheral ID6	0x00	0xFD8

**Table 10-3 Summary of the Performance Monitors Peripheral Identification Registers**

Register	Value	Offset
Performance Monitors Peripheral ID7	0x00	0xFDC
Performance Monitors Peripheral ID0	0xAE	0xFE0
Performance Monitors Peripheral ID1	0xB9	0xFE4
Performance Monitors Peripheral ID2	0x0B	0xFE8
Performance Monitors Peripheral ID3	0x00	0xFEC

Only bits [7:0] of each Performance Monitors Peripheral ID Register are used, with bits [31:8] reserved. Together, the eight Performance Monitors Peripheral ID Registers define a single 64-bit Peripheral ID.

The *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* describes these registers.

### 10.4.3 Performance Monitors Component Identification Registers

There are four read-only Performance Monitors Component Identification Registers, Component ID0 through Component ID3. [Table 10-4](#) shows these registers.

**Table 10-4 Summary of the Performance Monitors Component ID Registers**

Register	Value	Offset
Performance Monitors Component ID0	0x0D	0xFF0
Performance Monitors Component ID1	0x90	0xFF4
Performance Monitors Component ID2	0x05	0xFF8
Performance Monitors Component ID3	0xB1	0xFFC

The Performance Monitors Component Identification Registers identify Performance Monitor as PMUv2 architecture. The *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* describes these registers.

## 10.5 Events

Table 10-5 shows the events that are generated and the numbers that the PMU uses to reference the events. The table also shows the bit position of each event on the event bus, **PMUEVENT[37:0]**. Event reference numbers that are not listed are reserved.

Where an event is output as a bit-field, the field represents the number of events generated within a **CORECLK** cycle. To correctly count the total number of events generated, you must sample the field every **CORECLK** cycle, summing the values sampled.

**Table 10-5 Performance monitor events**

Event ID	PMUEVENT bit position	Description
0x01	[0]	Instruction fetch that causes a refill at (at least) the lowest level of instruction or unified cache. Includes speculative linefills in the count.
0x02	[1]	Instruction fetch that causes a TLB refill at (at least) the lowest level of TLB. Includes the speculative requests in the count.
0x03	[2]	Data read or write operation that causes a refill at (at least) the lowest level of data or unified cache. Counts the number of allocations performed in the Data Cache because of a read or a write.
0x04	[4:3]	Data read or write operation that causes a cache access at (at least) the lowest level of data or unified cache. This includes speculative reads.
0x05	[5]	Data read or write operation that causes a TLB refill at (at least) the lowest level of TLB. This does not include micro TLB misses because of PLD, PLI, CP15 Cache operation by MVA and CP15 VA to PA operations.
0x08	[16:10]	Instruction architecturally executed.
0x09	[17]	Exception taken. Counts the number of exceptions architecturally taken.
0x0A	[18]	Exception return architecturally executed. The following instructions are reported on this event: <ul style="list-style-type: none"> <li>• LDM {..., pc}^</li> <li>• RFE</li> <li>• ERET</li> <li>• Any data processing instruction with S bit set and PC as destination.</li> </ul>
0x0B	[19]	Change to ContextID retired. Counts the number of instructions architecturally executed writing into the ContextID register.
0x10	[26]	Branch mispredicted or not predicted. Counts the number of mispredicted or not-predicted branches executed. This includes the branches which are flushed because of a previous load/store which aborts late.
0x11	-	Cycle counter.
0x12	[27]	Branches or other change in program flow that could have been predicted by the branch prediction resources of the processor. This includes the branches which are flushed because of a previous load/store which aborts late.
0x13	[30:28]	Level 1 data cache access.
0x14	[31]	Level 1 instruction cache access.
0x15	[32]	Level 1 data cache eviction.
0x16	-	Level 2 data cache access.
0x17	-	Level 2 data cache refill.

Table 10-5 Performance monitor events (continued)

Event ID	PMUEVENT bit position	Description
0x18	-	Level 2 data cache write-back. Data transfers made as a result of a coherency request from the Level 2 caches to outside of the Level 1 and Level 2 caches are not counted. Write-backs made as a result of CP15 cache maintenance operations are counted.
0x19	-	Bus accesses. Single transfer bus accesses on either of the ACE read or write channels might increment twice in one cycle if both the read and write channels are active simultaneously. Operations that utilise the bus that do not explicitly transfer data, such as barrier or coherency operations are counted as bus accesses.
0x1B	-	Instructions speculatively executed.
0x1C	[20]	Write to translation table register (TTBR0 or TTBR1).
0x1D	-	Bus cycle.
0x40	-	Level 1 data cache access - Read.
0x41	-	Level 1 data cache access - Write.
0x50	-	Level 2 data cache access - Read.
0x51	-	Level 2 data cache access - Write.
0x56	-	Level 2 data cache write-back - Victim.
0x57	-	Level 2 data cache write-back - Cleaning and coherency.
0x58	-	Level 2 data cache invalidate.
0x60	-	Reserved.
0x62	-	Bus access - Normal Cacheable.
0x63	-	Bus access - Not Cacheable.
0x64	-	Bus access - Normal.
0x65	-	Bus access - Peripheral.
0x66	-	Data memory access - Read.
0x67	-	Data memory access - Write.
0x68	-	Unaligned access - Read.
0x69	-	Unaligned access - Write.
0x6A	[25:24]	Unaligned access.
0x6C	-	Exclusive instruction speculatively executed - LDREX.
0x6E	-	Exclusive instruction speculatively executed - STREX fail.
0x6F	-	Exclusive instruction speculatively executed - STREX.
0x70	[7:6]	Load instruction speculatively executed.
0x71	[9:8]	Store instruction speculatively executed.
0x72	-	Instruction speculatively executed - Load or store.
0x73	-	Instruction speculatively executed - Data processing.
0x74	-	Instruction speculatively executed - Advanced SIMD.



Table 10-5 Performance monitor events (continued)

Event ID	PMUEVENT bit position	Description
0x75	-	Instruction speculatively executed - VFP.
0x76	[21]	Instruction speculatively executed - Software change of the PC.
0x78	[22]	Branch speculatively executed - Immediate branch.
0x79	[23]	Branch speculatively executed - Procedure return.
0x7A	-	Branch speculatively executed - Indirect branch.
0x7C	-	Barrier speculatively executed - ISB.
0x7D	-	Barrier speculatively executed - DSB.
0x7E	-	Barrier speculatively executed - DMB.
0x81	-	Exception taken - Undefined Instruction.
0x8A	-	Exception taken - Hypervisor Call.
0xC0	-	Instruction side stalled due to a linefill.
0xC1	-	Instruction side stalled due to a translation table walk.
0xC2	-	Number of set of 4 ways read in the instruction cache - Tag RAM.
0xC3	-	Number of ways read in the instruction cache - Data RAM.
0xC4	-	Number of ways read in the instruction BTAC RAM.
0xCA	-	Data snooped from other processor. This event counts memory-read operations that read data from another processor within the local Cortex-A17 cluster, rather than accessing the L2 cache or issuing an external read. It increments on each transaction, rather than on each beat of data.
0xD3	-	Duration during which all slots in the Load-Store Unit are busy.
0xD8	-	Duration during which all slots in the Load-Store Issue queue are busy.
0xD9	-	Duration during which all slots in the Data Processing issue queue are busy.
0xDA	-	Duration during which all slots in the Data Engine issue queue are busy.
0xDB	-	Number of NEON instruction which fail their condition code and lead to a flush of the DE pipe.
0xDC	-	Number of Trap to hypervisor.
0xDE	-	PTM EXTOUT 0.
0xDF	-	PTM EXTOUT 1.
0xE0	-	Duration during which the MMU handle a translation table walk.
0xE1	-	Duration during which the MMU handle a Stage1 translation table walk.
0xE2	-	Duration during which the MMU handle a Stage2 translation table walk.
0xE3	-	Duration during which the MMU handle a translation table walk requested by the Load Store Unit.
0xE4	-	Duration during which the MMU handle a translation table walk requested by the Instruction side.
0xE5	-	Duration during which the MMU handle a translation table walk requested by a Preload instruction or Prefetch request.

Table 10-5 Performance monitor events (continued)

Event ID	PMUEVENT bit position	Description
0xE6	-	Duration during which the MMU handle a translation table walk requested by a CP15 operation (maintenance by MVA and VA-to-PA operation).
0xE7	-	Level 1 PLD TLB refill.
0xE8	-	Level 1 CP15 TLB refill.
0xE9	-	Level 1 TLB flush.
0xEA	-	Level 2 TLB access.
0xEB	-	Level 2 TLB miss.
-	[37-33]	Reserved.

———— **Note** ————

L2 events cannot be counted when the processor clock is slower than **CLK** (**CLK** to **CORECLKEN** ratio is not 1:1).

## 10.6 Interrupts

The Cortex-A17 MPCore processor asserts the **nPMUIRQ** signal when an interrupt is generated by the PMU. Your implementation might route this signal to an interrupt controller for prioritization and masking. This is the only mechanism that signals this interrupt to the processor.

———— **Note** ————

Performance monitor overflow interrupts are only signaled when PMCR.E is set to 1.

—————

## 10.7 Exporting PMU events

This section describes exporting of PMU events in:

- [External hardware](#).
- [Debug trace hardware](#).

### 10.7.1 External hardware

In addition to the counters in the processor, [Table 10-5 on page 10-10](#) shows the events on the **PMUEVENT** bus that your implementation might connect to external hardware.

### 10.7.2 Debug trace hardware

Some of the events in [Table 10-5 on page 10-10](#) might be exported to other external debug, or trace hardware, to enable the events to be monitored. See the *ARM® CoreSight™ SoC Technical Reference Manual* for more information.

# Chapter 11

## Program Trace Macrocell

This chapter describes the *Program Trace Macrocell* (PTM) for the Cortex-A17 MPCore processor. It contains the following sections:

- [About PTM on page 11-2.](#)
- [PTM options on page 11-3.](#)
- [PTM functional description on page 11-4.](#)
- [Reset on page 11-6.](#)
- [PTM programmers model on page 11-7.](#)
- [Register summary on page 11-11.](#)
- [Register descriptions on page 11-15.](#)

## 11.1 About PTM

The PTM is a real-time instruction flow trace module that complies with 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, DS-5.

For more information see:

- *ARM® CoreSight™ Architecture Specification* (ARM IHI 0029).
- *ARM® CoreSight™ SoC Technical Reference Manual* (ARM DDI 0480).

## 11.2 PTM options

Table 11-1 shows the options implemented in the Cortex-A17 PTM.

**Table 11-1 Cortex-A17 MPCore processor 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	38
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

## 11.3 PTM functional description

This section describes the PTM in:

- [Processor interface](#).
- [Trace generation](#).
- [Filtering and triggering resources on page 11-5](#).
- [FIFO on page 11-5](#).
- [Trace out on page 11-5](#).

Figure 11-1 shows the main functional blocks of the PTM.

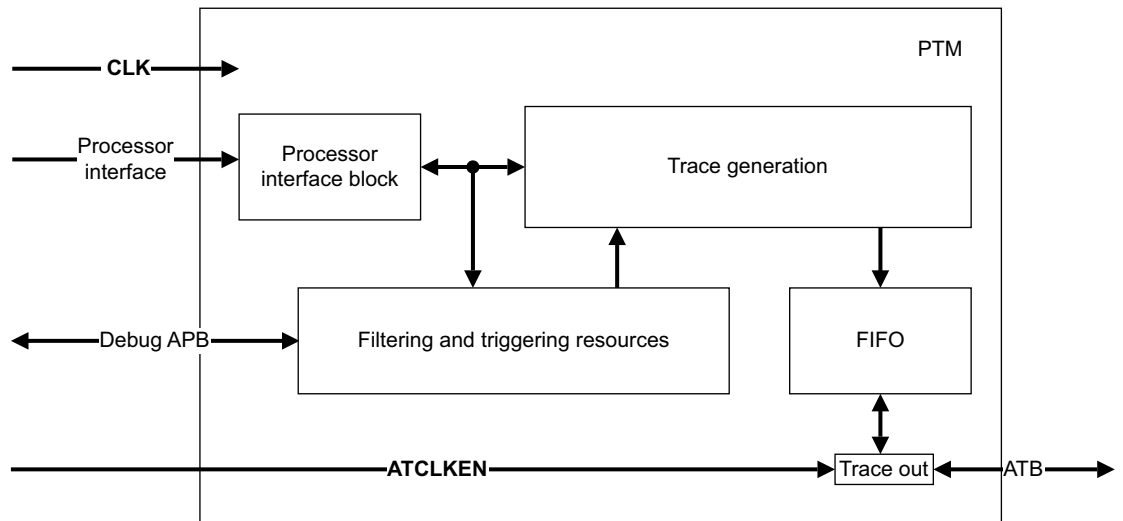


Figure 11-1 PTM functional blocks

### 11.3.1 Processor interface

This block monitors the behavior of the processor and generates waypoint information.

### 11.3.2 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 barrier instructions.
- 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.

### 11.3.3 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.

### 11.3.4 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.

### 11.3.5 Trace out

Trace from FIFO is output on the synchronous AMBA ATB interface.

## 11.4 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.

## 11.5 PTM programmers model

The programmers model enables you to use the PTM registers to control the macrocell. This section describes the mechanisms for programming the registers used to set up the trace and triggering facilities of the macrocell in:

- [Programming the PTM.](#)
- [Event definitions on page 11-8.](#)
- [Disabling the PTM on page 11-9.](#)
- [Interaction with the performance monitoring unit on page 11-9.](#)

### 11.5.1 Programming the PTM

When the PTM is powered up or reset, you must program all PTM registers before you enable tracing. If you do not, the trace results are UNPREDICTABLE.

When programming the PTM registers you must enable all the changes at the same time. For example, if the counter is reprogrammed before the trigger condition has been correctly set up, it might start to count based on incorrect events.

You access the PTM registers through the CoreSight Debug APB bus. The PTM implements the CoreSight lock access mechanism, and can distinguish between memory-mapped accesses from on-chip software and memory-mapped accesses from a debugger, for example by using the CoreSight *Debug Access Port* (DAP).

See the *ARM® CoreSight™ Program Flow Trace Architecture Specification* for more information about programming the PTM.

The following sections describe how you control PTM programming:

- [Using the Programming bit.](#)
- [Programming registers.](#)

#### Using the Programming bit

Use the Programming bit in the Main Control Register, see [Main Control Register on page 11-15](#), to disable all operations during programming.

When the Programming bit is set to 0 you must not write to registers other than the Main Control Register, because this can lead to unpredictable behavior.

When setting the Programming bit, you must not change any other bits of the Main Control Register. You must only change the value of bits other than the Programming bit of the Control Register when bit[1] of the Status Register is set to 1. ARM recommends that you use a read-modify-write procedure when changing the Main Control Register. For information on the Status Register, see the *ARM® CoreSight™ Program Flow Trace Architecture Specification*.

When the Programming bit is set to 1:

- The FIFO is permitted to empty and no more trace is produced.
- The counters, sequencer, and start/stop block are held in their current state.
- The external outputs are forced LOW.

#### 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 11-15.](#)

- [Synchronization Frequency Register on page 11-20.](#)
- CoreSight Trace ID Register, see the *ARM® CoreSight™ Program Flow Trace Architecture Specification*.
- The CoreSight registers at offsets 0xF00 to 0xFFC.
- [Peripheral Identification Registers on page 11-31.](#)
- [Component Identification Registers on page 11-31.](#)

To start tracing, you must program the following registers to avoid unpredictable behavior:

- [Main Control Register on page 11-15.](#)
- Trigger Event Register, TTER, see the *ARM® CoreSight™ Program Flow Trace Architecture Specification*.
- [TraceEnable Start/Stop Control Register on page 11-18.](#)
- [TraceEnable Control Register 1 on page 11-19.](#)
- TraceEnable Event Register, see the *ARM® CoreSight™ Program Flow Trace Architecture Specification*.
- CoreSight Trace ID Register, see the *ARM® CoreSight™ Program Flow Trace Architecture Specification*.

You might also need to program the following registers:

- 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, if timestamping is used.
- [Extended External Input Selection Register on page 11-23](#) if the extended external inputs are used.

## 11.5.2 Event definitions

As described in the *ARM® CoreSight™ Program Flow Trace Architecture Specification*, there are several event registers that you can program to select specific inputs as control events.

[Table 11-2](#) shows the event resources defined for the PTM.

**Table 11-2 Event resource definitions**

Resource type	Index values	Description
0b000	0-7	Single address comparator 1-8
0b001	0-3	Address range comparator 1-4
0b100	0-1	Counter 1-2 at zero

**Table 11-2 Event resource definitions (continued)**

Resource type	Index values	Description
0b101	0-2	Sequencer in states 1-3
	8	Context ID comparator
	11	VMID comparator
	15	Trace start/stop resource
0b110	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)

### 11.5.3 Disabling the PTM

During normal operation, the PTM buffers individual bytes of trace, and only generates output when at least four 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 or OS lock bit are 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.

### 11.5.4 Interaction with the performance monitoring unit

The processor includes a 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® Architecture Reference Manual ARMv7-A and ARMv7-R edition* 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.

### 11.5.5 Effect of debug 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 debug OS double lock is set. For more information on debug double lock, see the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

## 11.6 Register summary

This section summarizes the PTM registers. For full descriptions of the PTM registers, see:

- [Register descriptions on page 11-15](#), for the implementation-defined registers.
- The *ARM® 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 11-3](#), access type is described as follows:
 

<b>RW</b>	Read and write.
<b>RO</b>	Read only.
<b>WO</b>	Write only.

All PTM registers are 32 bits wide. The PTM registers are defined in the *ARM® CoreSight™ Program Flow Trace Architecture Specification*. [Table 11-3](#) 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 11-3 PTM register summary**

Base offset	Function	Type	Description
PTM configuration			
0x000	Main Control	RW	<a href="#">Main Control Register on page 11-15</a>
0x004	Configuration Code	RO	<a href="#">Configuration Code Register on page 11-16</a>
0x008	Trigger Event	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x010	Status	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x014	System Configuration	RO	<a href="#">System Configuration Register on page 11-18</a>
TraceEnable control			
0x018	TraceEnable Start/Stop Control	RW	<a href="#">TraceEnable Start/Stop Control Register on page 11-18</a>
0x020	TraceEnable Event	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x024	TraceEnable Control	RW	<a href="#">TraceEnable Control Register 1 on page 11-19</a>
Address comparators			
0x040-0x05C	Address Comparator Value 1-8	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x080-0x09C	Address Comparator Access Type 1-8	RW	
Counters			

Table 11-3 PTM register summary (continued)

Base offset	Function	Type	Description
0x140-0x144	Counter Reload Value 1-2	RW	ARM® 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			
0x180-0x194	Sequencer State Transition Event 1-6	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x19C	Current Sequencer State	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
External output event			
0x1A0-0x1A4	External Output Event 1-2	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
Context ID comparators			
0x1B0	Context ID Comparator Value 1	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x1BC	Context ID Comparator Mask	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
General control			
0x1E0	Synchronization Frequency	RW	<a href="#">Synchronization Frequency Register on page 11-20</a>
0x1E4	ID	RO	See <a href="#">ETM ID Register on page 11-20</a>
0x1E8	Configuration Code Extension	RO	<a href="#">Configuration Code Extension Register on page 11-21</a>
0x1EC	Extended External Input Selection	RW	<a href="#">Extended External Input Selection Register on page 11-23</a>
0x1F8	Timestamp Event	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x1FC	Auxiliary Control Register	RW	<a href="#">Auxiliary Control Register on page 11-23</a>
0x200	CoreSight Trace ID	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x204	VMID Comparator value	RW	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x300	OS Lock Access Specification	WO	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x304	OS Lock Status	RO	ARM® CoreSight™ Program Flow Trace Architecture Specification
0x310	Power Down Control	RW	<a href="#">Power Down Control Register on page 11-25</a>
0x314	Power Down Status	RO	ARM® CoreSight™ Program Flow Trace Architecture Specification



Table 11-3 PTM register summary (continued)

Base offset	Function	Type	Description
Integration registers			
0xEDC	Miscellaneous Outputs	WO	<a href="#">Miscellaneous Output Register on page 11-26</a>
0xEE0	Miscellaneous Inputs	RO	<a href="#">Miscellaneous Input Register on page 11-27</a>
0xEE8	Trigger	WO	<a href="#">Trigger Register on page 11-27</a>
0xEEC	ATB Data 0	WO	<a href="#">ITATBDATA0 bit assignments on page 11-28</a>
0xEF0	ATB Control 2	RO	<a href="#">ATB Control Register 2 on page 11-29</a>
0xEF4	ATB Identification	WO	<a href="#">ATB Identification Register on page 11-29</a>
0xEF8	ATB Control 0	WO	<a href="#">ATB Control Register 0 on page 11-30</a>
0xF00	Integration Mode Control	RW	<a href="#">Integration Mode Control Register on page 11-31</a>
0xFA0	Claim Tag Set	RW	<a href="#">ARM® CoreSight™ Program Flow Trace Architecture Specification</a>
0xFA4	Claim Tag Clear	RW	<a href="#">ARM® CoreSight™ Program Flow Trace Architecture Specification</a>
0xFB0	Lock Access	WO	<a href="#">ARM® CoreSight™ Program Flow Trace Architecture Specification</a>
0xFB4	Lock Status	RO	<a href="#">ARM® CoreSight™ Program Flow Trace Architecture Specification</a>
0xFB8	Authentication Status	RO	<a href="#">ARM® CoreSight™ Program Flow Trace Architecture Specification</a>
0xFC8	Device Configuration	RO	<a href="#">ARM® CoreSight™ Program Flow Trace Architecture Specification</a>
0xFCC	Device Type	RO	<a href="#">ARM® CoreSight™ Program Flow Trace Architecture Specification</a>
Peripheral and Component ID registers			
0xFD0	Peripheral ID4	RO	<a href="#">Peripheral Identification Registers on page 11-31</a>
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	

Table 11-3 PTM register summary (continued)

Base offset	Function	Type	Description
0xFF0	Component ID0	RO	<a href="#">Component Identification Registers on page 11-31</a>
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 *ARM® CoreSight™ Program Flow Trace Architecture Specification*.

## 11.7 Register descriptions

This section describes the implementation-specific PTM registers in the Cortex-A17 MPCore processor. [Table 11-3 on page 11-11](#) provides cross-references to individual registers.

The *ARM® CoreSight™ Program Flow Trace Architecture Specification* describes the other PTM registers.

### 11.7.1 Main Control Register

The ETMCR characteristics are:

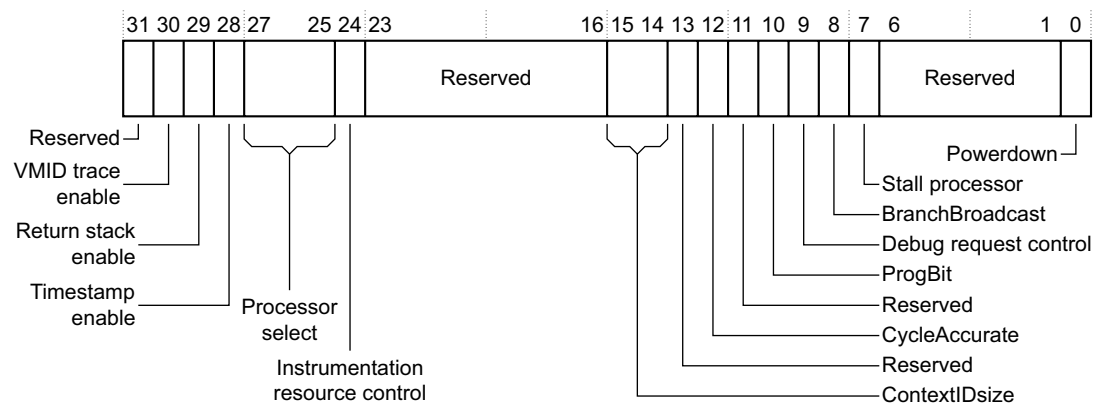
**Purpose** Controls general operation of the PTM, such as whether tracing is enabled or is cycle-accurate.

**Usage constraints** There are no usage constraints.

**Configurations** Available in all PTM configurations.

**Attributes** See the register summary in [Table 11-3 on page 11-11](#).

[Figure 11-2](#) shows the ETMCR bit assignments.



**Figure 11-2 ETMCR bit assignments**

[Table 11-4](#) shows the ETMCR bit assignments.

**Table 11-4 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. These bits are not implemented.
[24]	Instrumentation resources access control	RAZ. This bit is not implemented.
[23:16]	-	SBZP.

Table 11-4 ETMCR bit assignments (continued)

Bits	Name	Function
[15:14]	ContextIDsize	<p>The possible value of this field are:</p> <p>0b00            No Context ID tracing.</p> <p>0b01            One byte traced, Context ID bits[7:0].</p> <p>0b10            Two bytes traced, Context ID bits[15:0].</p> <p>0b11            Four bytes traced, Context ID bits[31:0].</p> <p>The reset value is 0b00.</p> <p>———— <b>Note</b> ————</p> <p>The PTM traces only the number of bytes specified, even if the new Context ID value is larger than this.</p>
[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 <b>DBGREQ</b> output is asserted until <b>DBGACK</b> 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	<p>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.</p> <p>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:</p> <ul style="list-style-type: none"> <li>• ETMCR, bit[0] and bits[27:25].</li> <li>• ETMLAR.</li> <li>• ETMCLAIMSET.</li> <li>• ETMCLAIMCLR.</li> <li>• ETMOSLAR.</li> </ul> <p>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.</p>

## 11.7.2 Configuration Code Register

The ETMCCR characteristics are:

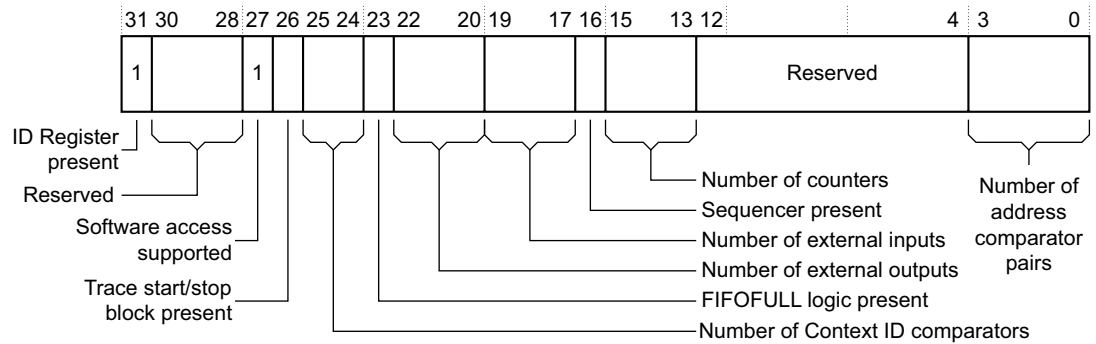
<b>Purpose</b>	Enables software to read the implementation-defined configuration of the PTM, giving the number of each type of resource. Where a value indicates the number of instances of a particular resource, zero indicates that there are no implemented resources of that resource type.
----------------	---

**Usage constraints** There are no usage constraints.

**Configurations** Available in all PTM configurations.

**Attributes** See the register summary in [Table 11-3 on page 11-11](#).

[Figure 11-3](#) shows the ETMCCR bit assignments.



**Figure 11-3 ETMCCR bit assignments**

[Table 11-5](#) shows the ETMCCR bit assignments.

**Table 11-5 ETMCCR bit assignments**

Bits	Name	Function
[31]	ID Register present	ID Register: <b>1</b> ID Register is present. See <a href="#">ETM ID Register on page 11-20</a> for more information.
[30:28]	-	RAZ.
[27]	Software access supported	Software access support: <b>1</b> The Cortex-A17 MPCore processor supports software access.
[26]	Trace start/stop block present	Trace start/stop block: <b>1</b> The trace start/stop block is present.
[25:24]	Number of Context ID comparators	Indicates the number of Context ID comparators. The Cortex-A17 MPCore processor supports one.
[23]	FIFOFULL logic present	<b>0</b> The Cortex-A17 MPCore processor does not implement the FIFOFULL logic.
[22:20]	Number of external outputs	Indicates the number of external outputs. The Cortex-A17 MPCore processor supports two.
[19:17]	Number of external inputs	Indicates the number of external inputs. The Cortex-A17 MPCore processor supports four.
[16]	Sequencer present	Sequencer: <b>1</b> The sequencer is present.
[15:13]	Number of counters	Indicates the number of counters. The Cortex-A17 MPCore processor supports two.
[12:4]	-	SBZP.
[3:0]	Number of address comparator pairs	Indicates the number of address comparator pairs. The Cortex-A17 MPCore processor supports four.

### 11.7.3 System Configuration Register

The ETMSCR characteristics are:

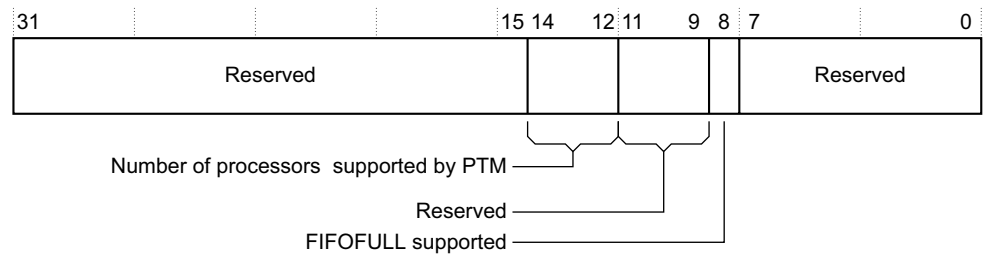
**Purpose** Shows the PTM features supported by the PTM macrocell. The contents of this register are based on inputs provided by the ASIC.

**Usage constraints** There are no usage constraints.

**Configurations** Available in all PTM configurations.

**Attributes** See the register summary in [Table 11-3 on page 11-11](#).

[Figure 11-4](#) shows the ETMSCR bit assignments.



**Figure 11-4 ETMSCR bit assignments**

[Table 11-6](#) shows the ETMSCR bit assignments.

**Table 11-6 ETMSCR bit assignments**

Bits	Name	Function
[31:15]	-	SBZP.
[14:12]	Number of processors supported by PTM	Indicates the number of processors supported minus one. <b>0</b> The Cortex-A17 MPCore processor supports one processor per PTM.
[11:9]	-	SBZP.
[8]	FIFOFULL supported	<b>0</b> The Cortex-A17 MPCore processor does not implement the FIFOFULL logic.
[7:0]	-	SBZP.

### 11.7.4 TraceEnable Start/Stop Control Register

The ETMTSSCR 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 11-3 on page 11-11](#).

[Figure 11-5 on page 11-19](#) shows the ETMTSSCR bit assignments.

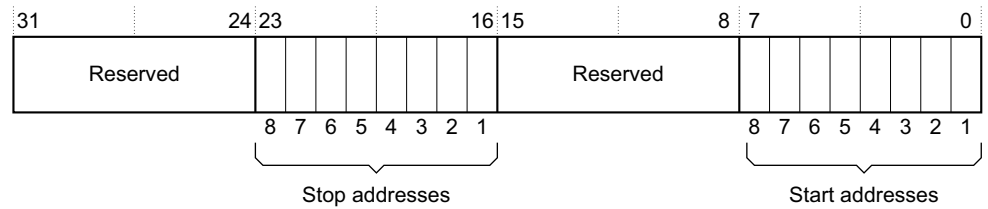


Figure 11-5 ETMSSCR bit assignments

Table 11-7 shows the ETMTSSCR bit assignments.

Table 11-7 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.

### 11.7.5 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 11-3 on page 11-11](#).

Figure 11-6 shows the ETMECR1 bit assignments.

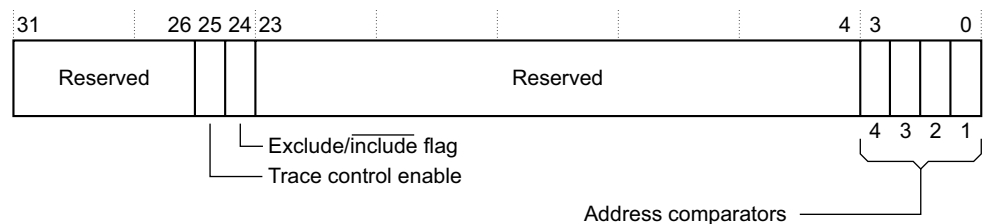


Figure 11-6 ETMECR1 bit assignments

Table 11-8 shows the ETMECR1 bit assignments.

**Table 11-8 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: <b>0</b> Tracing is unaffected by the trace start/stop logic. <b>1</b> 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: <b>0</b> Include. The specified address range comparators indicate the regions where tracing can occur. No tracing occurs outside this region. <b>1</b> 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 ETMTREEVER 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.

## 11.7.6 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 11-3 on page 11-11](#).

Bits[2:0] of this register are not implemented and read as zero. In all other respects, the *ARM® CoreSight™ Program Flow Trace Architecture Specification* describes the operation of this register in the PTM.

## 11.7.7 ETM ID Register

The ETMIDR characteristics are:

**Purpose**

- Holds the PTM architecture variant.
- Defines the programmers model for the PTM.

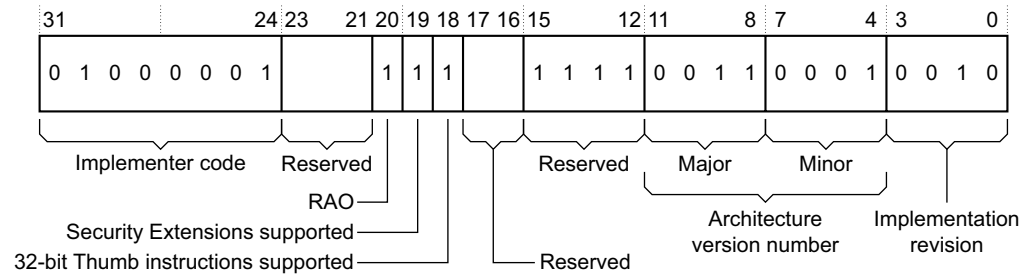
**Usage constraints** There are no usage constraints.



**Configurations** Available in all PTM configurations.

**Attributes** See the register summary in [Table 11-3 on page 11-11](#).

[Figure 11-7](#) shows the ETMIDR bit assignments.



**Figure 11-7 ETMIDR bit assignments**

[Table 11-9](#) shows the ETMIDR bit assignments.

**Table 11-9 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	Indicates support for Security Extensions. 1 Processor implements Security Extensions.
[18]	32-bit Thumb instructions supported	Indicates support for 32-bit Thumb instructions. 1 Processor supports 32-bit Thumb instruction traced as single instructions.
[17:16]	-	RAZ.
[15:12]	-	This field reads as 0b1111.
[11:8]	Major architecture version	Indicates the major architecture version number: 0b0011.
[7:4]	Minor architecture version	Indicates the minor architecture version number: 0b0001.
[3:0]	Implementation revision	Indicates the implementation revision: 0b0000.

### 11.7.8 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 11-16](#).

**Usage constraints** Software uses this register with the ETMCCR.

**Configurations** Available in all PTM configurations.

**Attributes** See the register summary in [Table 11-3 on page 11-11](#).

[Figure 11-8 on page 11-22](#) shows the ETMCCER bit assignments.

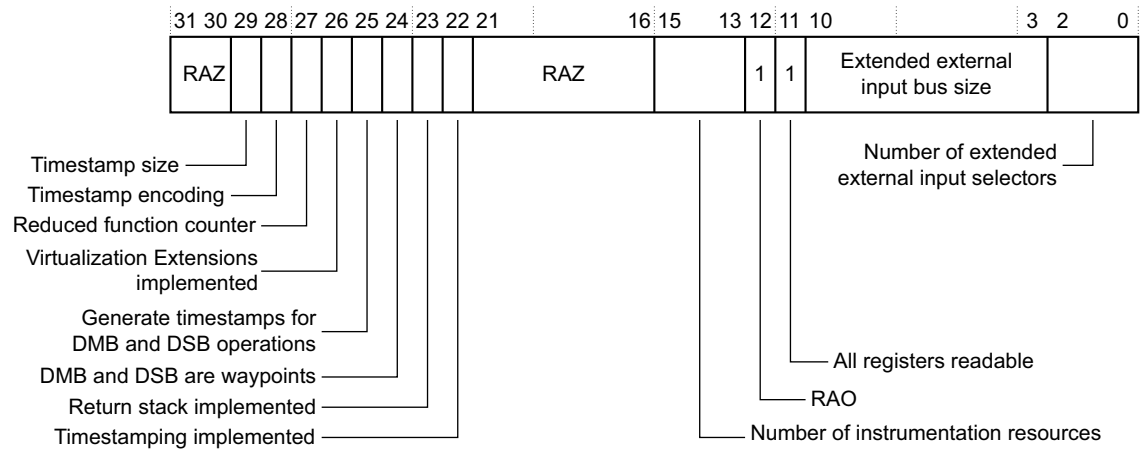


Figure 11-8 ETMCCER bit assignments

Table 11-10 shows the ETMCCER bit assignments.

Table 11-10 ETMCCER bit assignments

Bits	Name	Function	
[31:30]	-	SBZP.	
[29]	Timestamp size	1	The Cortex-A17 MPCore processor timestamp size is 64 bits.
[28]	Timestamp encoding	1	The Cortex-A17 MPCore processor timestamp size is encoded as a natural binary number.
[27]	Reduced function counter	0	The Cortex-A17 MPCore processor does not implement counters as full function counters.
[26]	Virtualization Extensions implemented	1	The Cortex-A17 MPCore processor implements Virtualization Extensions.
[25]	Generate timestamp for DMB and DSB operations	0	The Cortex-A17 MPCore processor does not generate timestamps for DMB and DSB operations.
[24]	DMB and DSB are waypoints	0	The Cortex-A17 MPCore processor does not treat DMB and DSB instructions as waypoints.
[23]	Return stack implemented	1	The Cortex-A17 MPCore processor implements a return stack.
[22]	Timestamping implemented	1	The Cortex-A17 MPCore processor implements timestamping.
[21:16]	-	RAZ.	
[15:13]	Number of instrumentation resources	0	The Cortex-A17 MPCore processor does not implement any instrumentation resources.
[12]	-	RAO.	

**Table 11-10 ETMCCER bit assignments (continued)**

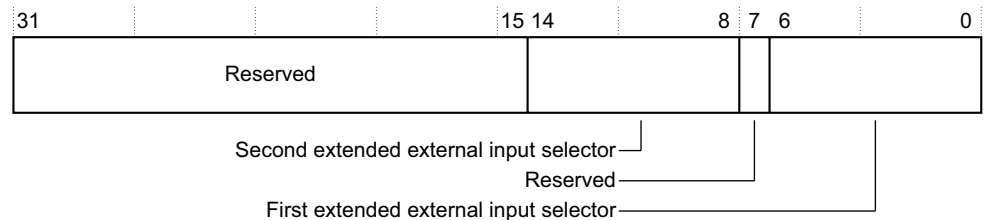
Bits	Name	Function
[11]	All registers readable	Indicates that all registers, except some Integration Test Registers, are readable. Registers with names that start with IT are the Integration Test Registers, for example ITATBCTR1.
[10:3]	Extended External input bus size	Indicates the number of bits input from the <b>PMUEVENT</b> bus: <b>38</b> <b>PMUEVENT[37:0]</b> .
[2:0]	Number of extended external input selectors	Indicates the number of extended external input selectors: <b>0b010</b> Two selectors.

### 11.7.9 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 11-3 on page 11-11](#).

[Figure 11-9](#) shows the ETMEXTINSELR bit assignments.

**Figure 11-9 ETMEXTINSELR bit assignments**

[Table 11-11](#) shows the ETMEXTINSELR bit assignments.

**Table 11-11 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

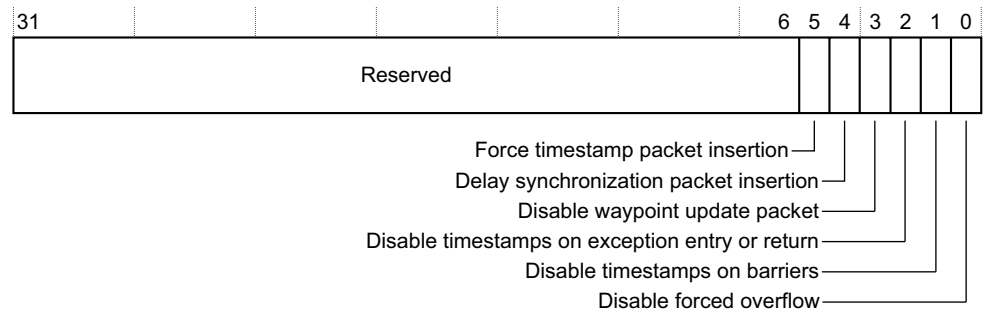
### 11.7.10 Auxiliary Control Register

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 11-3 on page 11-11](#).

[Figure 11-10](#) shows the ETMAUXCR bit assignments.



**Figure 11-10 ETMAUXCR bit assignments**

[Table 11-12](#) shows the ETMAUXCR bit assignments.

**Table 11-12 ETMAUXCR bit assignments**

Bits	Name	Function
[31:6]	-	Reserved.
[5]	Force timestamp packet insertion	<p>Force insertion of timestamp packets, regardless of current trace activity:</p> <p><b>0</b> Timestamp packets delayed when trace activity is high. This is the reset value.</p> <p><b>1</b> Timestamp packets inserted regardless of trace activity.</p> <p>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.</p>
[4]	Delay synchronization packet insertion	<p>Delays insertion of synchronization packets:</p> <p><b>0</b> Synchronization packets inserted normally regardless of trace activity. This is the reset value.</p> <p><b>1</b> Synchronization packets delayed when trace activity is high.</p> <p>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.</p>
[3]	Disable waypoint update packet	<p>Specifies whether the PTM issues waypoint update packets if there are more than 4096 bytes between waypoints:</p> <p><b>0</b> PTM always issues update packets if there are more than 4096 bytes between waypoints. This is the reset value.</p> <p><b>1</b> PTM does not issue waypoint update packets unless required to do so as the result of an exception or debug entry.</p>

Table 11-12 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:
		<b>0</b> PTM issues timestamps on exception entry or return. This is the reset value.
		<b>1</b> 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:
		<b>0</b> PTM issues timestamps on barrier instructions. This is the reset value.
		<b>1</b> 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:
		<b>0</b> Forced overflow enabled. This is the reset value.
		<b>1</b> Forced overflow disabled.

When setting any of bits[3:0] of this register the PTM behavior might contradict the *ARM® 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 *ARM® 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 required 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.

### 11.7.11 Power Down Control Register

The ETMPDCR characteristics are:

<b>Purpose</b>	Controls powerdown of the PTM.
<b>Usage constraints</b>	There are no usage constraints.
<b>Configurations</b>	Available in all PTM configurations that have full support for trace over powerdown.
<b>Attributes</b>	See the register summary in <a href="#">Table 11-3 on page 11-11</a> .

[Figure 11-11 on page 11-26](#) shows the ETMPDCR bit assignments.

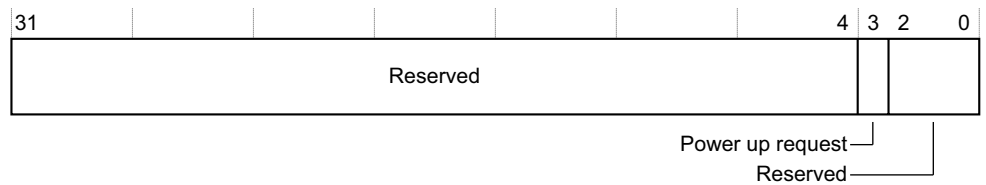


Figure 11-11 ETMPDCR bit assignments

Table 11-13 shows the ETMPDCR bit assignments.

Table 11-13 ETMPDCR bit assignments

Bits	Name	Function
[31:4]	-	Reserved.
[3]	Power up request	Core powerup 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: <b>0</b> <b>DBGPWRUPREQ</b> is LOW. This is the reset value. <b>1</b> <b>DBGPWRUPREQ</b> is HIGH.
[2:0]	-	Reserved.

### 11.7.12 Miscellaneous Output Register

The ITMISCOUT characteristics are:

**Purpose** Controls signal outputs when bit[0] of the Integration Mode Control Register is set. See [Integration Mode Control Register](#) on page 11-31.

**Usage constraints** There are no usage constraints.

**Configurations** Available in all PTM configurations.

**Attributes** See the register summary in [Table 11-3 on page 11-11](#).

[Figure 11-12](#) shows the ITMISCOUT bit assignments.



Figure 11-12 ITMISCOUT bit assignments

[Table 11-14](#) shows the ITMISCOUT bit assignments.

Table 11-14 ITMISCOUT bit assignments

Bits	Name	Function
[31:10]	-	Reserved
[9:8]	<b>PTMEXTOUT[1:0]</b>	Drives the <b>PTMEXTOUT[1:0]</b> outputs <sup>a</sup>
[7:0]	-	Reserved

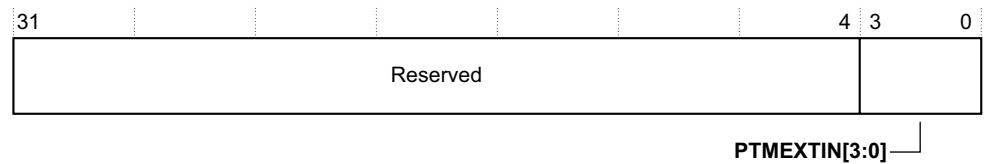
a. Connected to the integrated CTI. See [Chapter 12 Cross Trigger](#).

### 11.7.13 Miscellaneous Input Register

The ITMISCIN characteristics are:

- Purpose** Enables the values of signal inputs to be read when bit[0] of the Integration Mode Control Register is set. See [Integration Mode Control Register on page 11-31](#).
- Usage constraints** There are no usage constraints.
- Configurations** Available in all PTM configurations.
- Attributes** See the register summary in [Table 11-3 on page 11-11](#).

[Figure 11-13](#) shows the ITMISCIN bit assignments.



**Figure 11-13 ITMISCIN bit assignments**

[Table 11-15](#) shows the ITMISCIN bit assignments.

**Table 11-15 ITMISCIN bit assignments**

Bits	Name	Function
[31:4]	-	Reserved
[3:0]	PTMEXTIN[3:0]	Returns the value of the <b>EXTIN[3:0]</b> inputs <sup>a</sup>

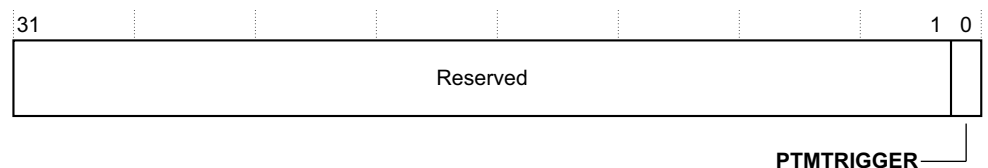
a. Connected to the integrated CTI. See [Chapter 12 Cross Trigger](#).

### 11.7.14 Trigger Register

The ITTRIGGER characteristics are:

- Purpose** Controls signal outputs when bit[0] of the Integration Mode Control Register is set. See [Integration Mode Control Register on page 11-31](#).
- Usage constraints** There are no usage constraints.
- Configurations** Available in all PTM configurations.
- Attributes** See the register summary in [Table 11-3 on page 11-11](#).

[Figure 11-14](#) shows the ITTRIGGER bit assignments.



**Figure 11-14 ITTRIGGER bit assignments**

Table 11-16 shows the ITTRIGGER bit assignments.

**Table 11-16 ITTRIGGER bit assignments**

Bits	Name	Function
[31:1]	-	Reserved
[0]	PTMTRIGGER	Drives the <b>PTMTRIGGER</b> output <sup>a</sup>

a. Connected to the integrated CTI. See [Chapter 12 Cross Trigger](#).

### 11.7.15 ATB Data Register 0

The ITATBDATA0 characteristics are:

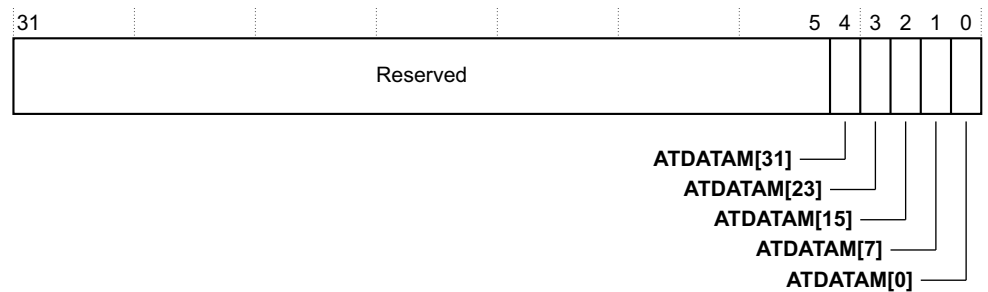
**Purpose** Controls signal outputs when bit[0] of the Integration Mode Control Register is set. See [Integration Mode Control Register](#) on page 11-31.

**Usage constraints** There are no usage constraints.

**Configurations** Available in all PTM configurations.

**Attributes** See the register summary in [Table 11-3](#) on page 11-11.

[Figure 11-15](#) shows the ITATBDATA0 bit assignments.



**Figure 11-15 ITATBDATA0 bit assignments**

[Table 11-17](#) shows the ITATBDATA0 bit assignments.

**Table 11-17 ITATBDATA0 bit assignments**

Bits	Name	Function
[31:5]	-	Reserved
[4]	ATDATAM[31]	Drives the <b>ATDATAM[31]</b> output
[3]	ATDATAM[23]	Drives the <b>ATDATAM[23]</b> output
[2]	ATDATAM[15]	Drives the <b>ATDATAM[15]</b> output
[1]	ATDATAM[7]	Drives the <b>ATDATAM[7]</b> output
[0]	ATDATAM[0]	Drives the <b>ATDATAM[0]</b> output

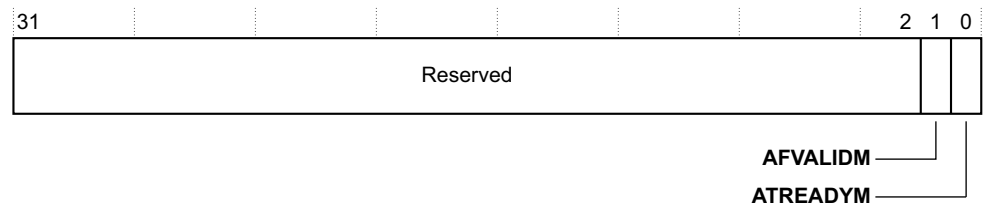


### 11.7.16 ATB Control Register 2

The ITATBCTR2 characteristics are:

- Purpose** Enables the values of signal inputs to be read when bit[0] of the Integration Mode Control Register is set. See [Integration Mode Control Register on page 11-31](#).
- Usage constraints** There are no usage constraints.
- Configurations** Available in all PTM configurations.
- Attributes** See the register summary in [Table 11-3 on page 11-11](#).

[Figure 11-16](#) shows the ITATBCTR2 bit assignments.



**Figure 11-16** ITATBCTR2 bit assignments

[Table 11-18](#) shows the ITATBCTR2 bit assignments.

**Table 11-18** ITATBCTR2 bit assignments

Bits	Name	Function
[31:2]	-	Reserved
[1]	AFVALIDM	Returns the value of <b>AFVALIDM</b> input
[0]	ATREADYM	Returns the value of <b>ATREADYM</b> input <sup>a</sup>

- a. To sample **ATREADYM** correctly from the Cortex-A17 MPCore processor signals, **ATVALIDM** must be asserted.

### 11.7.17 ATB Identification Register

The ITATBID characteristics are:

- Purpose** Controls signal outputs when bit[0] of the Integration Mode Control Register is set. See [Integration Mode Control Register on page 11-31](#).
- Usage constraints** There are no usage constraints.
- Configurations** Available in all PTM configurations.
- Attributes** See the register summary in [Table 11-3 on page 11-11](#).

[Figure 11-17 on page 11-30](#) shows the ITATBID bit assignments.

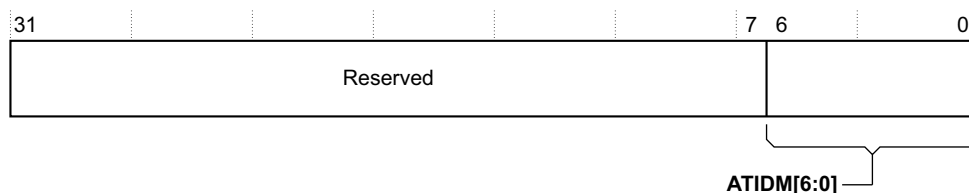


Figure 11-17 ITATBID bit assignments

Table 11-19 shows the ITATBID bit assignments.

Table 11-19 ITATBID bit assignments

Bits	Name	Function
[31:7]	-	Reserved
[6:0]	ATIDM[6:0]	Drives the <b>ATIDM[6:0]</b> outputs

### 11.7.18 ATB Control Register 0

The ITATBCTR0 characteristics are:

- Purpose** Controls signal outputs when bit[0] of the Integration Mode Control Register is set. See [Integration Mode Control Register on page 11-31](#).
- Usage constraints** There are no usage constraints.
- Configurations** Available in all PTM configurations.
- Attributes** See the register summary in [Table 11-3 on page 11-11](#).

Figure 11-18 shows the ITATBCTR0 bit assignments.

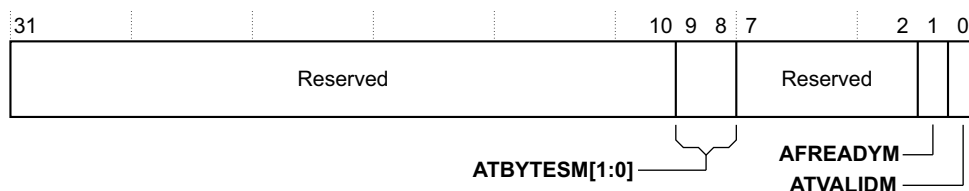


Figure 11-18 ITATBCTR0 bit assignments

Table 11-20 shows the ITATBCTR0 bit assignments.

Table 11-20 ITATBCTR0 bit assignments

Bits	Name	Function
[31:10]	-	Reserved
[9:8]	<b>ATBYTESM[1:0]</b>	Drives the <b>ATBYTESM</b> outputs
[7:2]	-	Reserved
[1]	<b>AFREADYM</b>	Drives the <b>AFREADYM</b> output
[0]	<b>ATVALIDM</b>	Drives the <b>ATVALIDM</b> output

### 11.7.19 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 11-15](#).

After leaving integration mode, the PTM must be reset before attempting to perform tracing.

The *ARM® CoreSight™ Program Flow Trace Architecture Specification* describes the operation of this register in the PTM.

### 11.7.20 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 11-21](#).

**Table 11-21 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	0x5B	0xFE0
Peripheral ID1	0xB9	0xFE4
Peripheral ID2	0x0B	0xFE8
Peripheral ID3	0x00	0xFEC

Only bits[7:0] of each Peripheral ID Register are used, with bits[31:8] reserved. Together, the eight Peripheral ID Registers define a single 64-bit Peripheral ID.

The *ARM® CoreSight™ Program Flow Trace Architecture Specification* describes these registers.

### 11.7.21 Component Identification Registers

There are four read-only Component Identification Registers, Component ID0 to Component ID3. [Table 11-22](#) shows these registers.

**Table 11-22 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 *ARM® CoreSight™ Program Flow Trace Architecture Specification* describes these registers.

# Chapter 12

## Cross Trigger

This chapter describes the cross trigger interfaces for the Cortex-A17 MPCore processor. It contains the following sections:

- [\*About the cross trigger on page 12-2.\*](#)
- [\*Trigger inputs and outputs on page 12-3.\*](#)
- [\*Cortex-A17 CTI on page 12-4.\*](#)
- [\*Cortex-A17 CTM on page 12-5.\*](#)

## 12.1 About the cross trigger

The Cortex-A17 MPCore processor has a single external 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). A number of trigger inputs and trigger outputs are connected between debug components in the Cortex-A17 MPCore processor and CoreSight CTI blocks.

The 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 12-1 shows the debug system components and the available trigger inputs and trigger outputs.

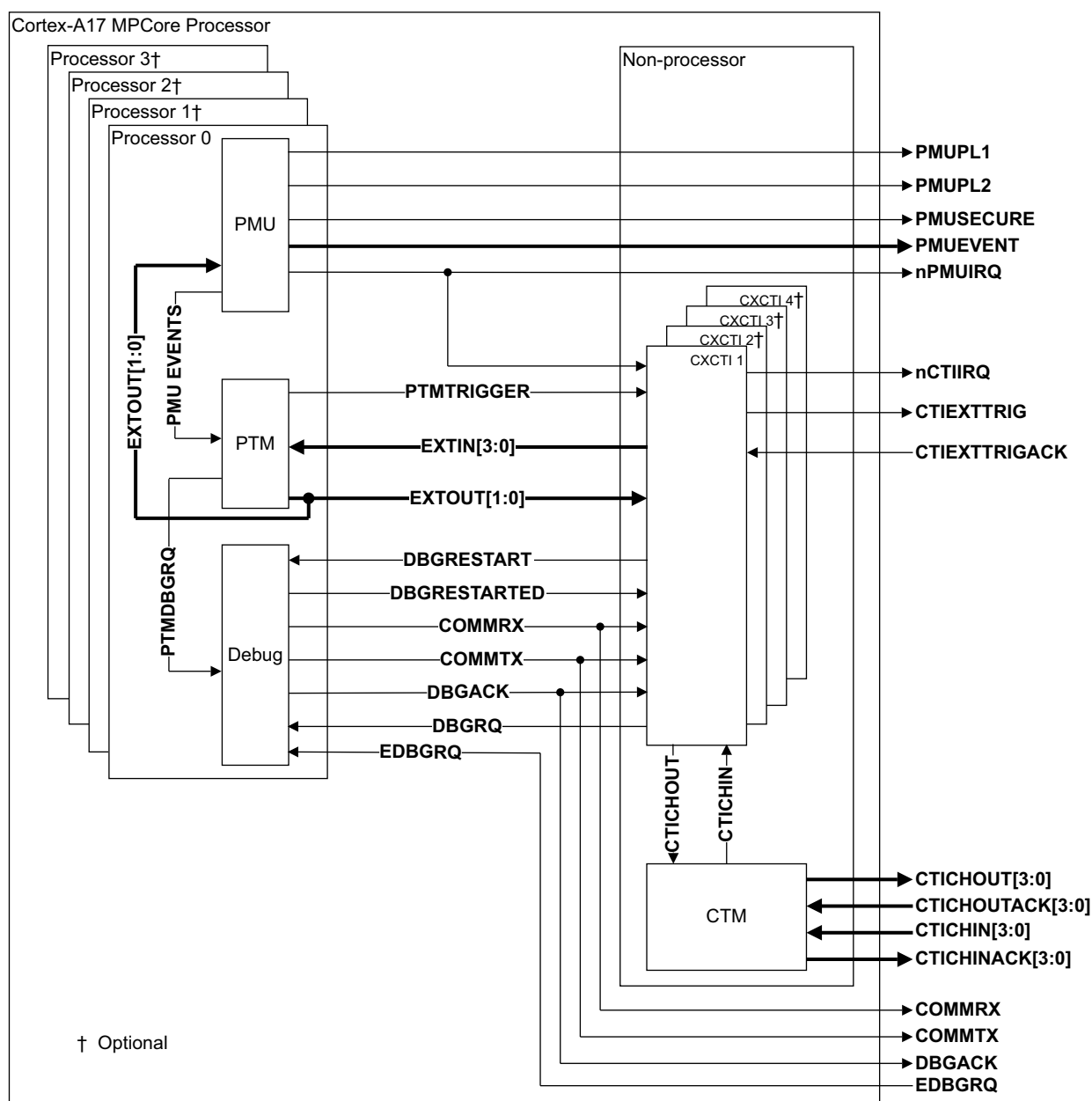


Figure 12-1 Debug system components

## 12.2 Trigger inputs and outputs

This section describes the trigger inputs and outputs that are available to the CTI.

Table 12-1 shows the CTI inputs.

**Table 12-1 Trigger inputs**

CTI input	Name	Description
0	-	-
1	<b>PMUIRQ<sup>a</sup></b>	PMU generated interrupt
2	<b>EXTOUT[0]</b>	PTM output
3	<b>EXTOUT[1]</b>	PTM output
4	<b>COMMTX</b>	Debug communication transmit channel is empty
5	<b>COMMRX</b>	Debug communication receive channel is full
6	<b>PTMTRIGGER</b>	PTM trigger
7	-	-

a. This signal is the same as **nPMUIRQ** with inverted polarity.

Table 12-2 shows the CTI outputs.

**Table 12-2 Trigger outputs**

CTI output	Name	Description
0	<b>DBGRRQ<sup>a</sup></b>	Causes the processor to enter debug state
1	<b>EXTIN[0]</b>	PTM input
2	<b>EXTIN[1]</b>	PTM input
3	<b>EXTIN[2]</b>	PTM input
4	<b>EXTIN[3]</b>	PTM input
5	<b>CTIEXTTRIG</b>	CTI external trigger
6	<b>nCTIIRQ</b>	CTI interrupt
7	<b>DBGRESTART</b>	Causes the processor to exit debug state

a. **DBGRRQ** is OR gated with **EDBGRQ** in the corresponding processor.

## 12.3 Cortex-A17 CTI

In the Cortex-A17 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 the 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**.



## 12.4 Cortex-A17 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-A17 multiprocessor. 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 13

## NEON and Floating-point Unit

This chapter describes the Cortex-A17 NEON and floating-point unit, their features, and the registers that they use. It contains the following sections:

- [\*About NEON and floating-point unit on page 13-2.\*](#)
- [\*Programmers model for NEON and floating-point unit on page 13-3.\*](#)

## 13.1 About NEON and floating-point unit

NEON technology is the implementation of the Advanced *Single Instruction Multiple Data* (SIMD) extension to the ARMv7 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.

Advanced SIMD and VFP extensions are supported as a single implementation option in the macrocell, meaning that, they cannot be independently implemented.

This section describes the following:

- [Advanced SIMDv2 support](#).
- [VFPv4 support](#).

### 13.1.1 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.

### 13.1.2 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-A17 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-A17 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.

---

## 13.2 Programmers model for NEON and floating-point unit

This section describes the programmers model for the Cortex-A17 NEON and VFP unit in:

- [Accessing the Advanced SIMD and VFP feature identification registers.](#)
- [Enabling Advanced SIMD and VFP extensions.](#)
- [Register summary on page 13-5.](#)
- [Register descriptions on page 13-5.](#)

### 13.2.1 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 13-1](#) lists the feature identification registers for the Advanced SIMD and VFP extensions.

**Table 13-1 Advanced SIMD and VFP feature identification registers**

Name	Description
FPSID	See <a href="#">Floating-Point System ID Register on page 13-5</a>
MVFR0	See <a href="#">Media and VFP Feature Register 0 on page 13-9</a>
MVFR1	See <a href="#">Media and VFP Feature Register 1 on page 13-8</a>

### 13.2.2 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 4-57](#).
- 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 4-60](#).
- 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 4-66](#).

To enable Advanced SIMD and VFP operations, software must set the FPEXC.EN bit to 1. See [Floating-Point Exception Register on page 13-10](#).

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 4-57](#) and [Floating-Point Exception Register on page 13-10](#).

1. Enable access to CP10 and CP11 and clear the ASEDIS bit in the CPACR:

```
MOV r0, 0x0F00000
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 4-60](#), [Coprocessor Access Control Register on page 4-57](#), and [Floating-Point Exception Register on page 13-10](#).

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, 0x0F00000
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.

---

### 13.2.3 Register summary

Table 13-2 gives a summary of the Cortex-A17 Advanced SIMD and VFP system registers.

**Table 13-2 Advanced SIMD and VFP system registers**

Name	Type	Reset	Description
FPSID	RO	0x41033000	See <i>Floating-Point System ID Register</i>
FPSCR	RW	0x00000000	See <i>Floating-Point Status and Control Register</i> on page 13-6
MVFR1	RO	0x11111111	See <i>Media and VFP Feature Register 1</i> on page 13-8
MVFR0	RO	0x10110222	See <i>Media and VFP Feature Register 0</i> on page 13-9
FPEXC	RW	0x00000000	See <i>Floating-Point Exception Register</i> on page 13-10

---

**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.

### 13.2.4 Register descriptions

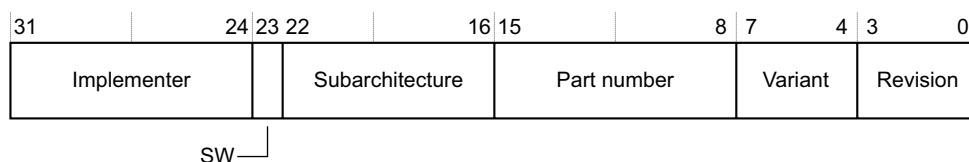
This section describes the Cortex-A17 Advanced SIMD and VFP system registers. Table 13-2 provides cross-references to individual registers.

#### Floating-Point System ID Register

The FPSID characteristics are:

<b>Purpose</b>	Provides top-level information about the floating-point implementation.
<b>Usage constraints</b>	Only accessible from PL1 or higher.
<b>Configurations</b>	Available if VFP is implemented.
<b>Attributes</b>	See the register summary in Table 13-2.

Figure 13-1 on page 13-6 shows the FPSID bit assignments.

**Figure 13-1 FPSID bit assignments**

[Table 13-3](#) shows the FPSID bit assignments.

**Table 13-3 FPSID bit assignments**

Bits	Name	Function
[31:24]	Implementer	Indicates the implementer: 0x41 ARM Limited.
[23]	SW	Software 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: 0x03 VFP architecture v3 or later, with no subarchitecture. The floating-point implementation is in hardware. No software support code is required. 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: 0xE Cortex-A17.
[3:0]	Revision	Indicates the revision number for the floating-point implementation: 0x0 Revision.

## Floating-Point Status and Control Register

The FPSCR characteristics are:

<b>Purpose</b>	Provides status information and control of unprivileged execution for the floating-point system.
<b>Usage constraints</b>	There are no usage constraints.
<b>Configurations</b>	Available if VFP is implemented.
<b>Attributes</b>	See the register summary in <a href="#">Table 13-2 on page 13-5</a> .

[Figure 13-2 on page 13-7](#) shows the FPSCR bit assignments.

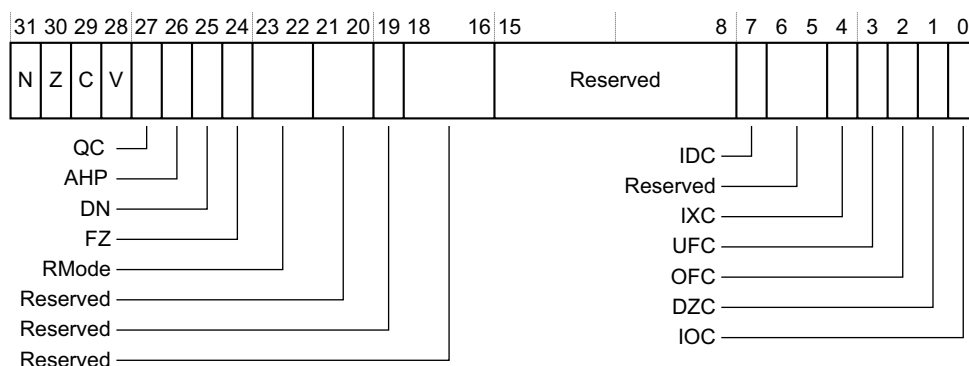


Figure 13-2 FPSCR bit assignments

Table 13-4 shows the FPSCR bit assignments.

Table 13-4 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.
[26]	AHP	Alternative Half-Precision control bit: <div> <div>0</div> <div>1</div> </div> <div> <div>IEEE half-precision format selected.</div> <div>Alternative half-precision format selected.</div> </div>
[25]	DN	Default NaN mode control bit: <div> <div>0</div> <div>1</div> </div> <div> <div>NaN operands propagate through to the output of a floating-point operation.</div> <div>Any operation involving one or more NaNs returns the Default NaN.</div> </div> 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: <div> <div>0</div> <div>1</div> </div> <div> <div>Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.</div> <div>Flush-to-zero mode enable.</div> </div> 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.



Table 13-4 FPSCR bit assignments (continued)

Bits	Field	Function
[23:22]	RMode	Rounding Mode control field: 0b00 <i>Round to Nearest (RN) mode.</i> 0b01 <i>Round towards Plus Infinity (RP) mode.</i> 0b10 <i>Round towards Minus Infinity (RM) mode.</i> 0b11 <i>Round towards Zero (RZ) mode.</i> 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]	-	RAZ/WI.
[19]	-	UNK/SBZP.
[18:16]	-	RAZ/WI.
[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.

**Attributes** See the register summary in [Table 13-2 on page 13-5](#).

[Figure 13-3](#) shows the MVFR1 bit assignments.

31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
A_SIMD FMAC	VFP HPFP		A_SIMD HPFP	A_SIMD SPFP		A_SIMD integer		A_SIMD load/store		D_NaN mode		FtZ mode			

Figure 13-3 MVFR1 bit assignments

Table 13-5 shows the MVFR1 bit assignments.

**Table 13-5 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.
[19:16]	A_SIMD SPFP	Indicates whether the Advanced SIMD extension supports single-precision floating-point operations: 0x1 Supported.
[15:12]	A_SIMD integer	Indicates whether the Advanced SIMD extension supports integer operations: 0x1 Supported.
[11:8]	A_SIMD load/store	Indicates whether the Advanced SIMD extension supports load/store instructions: 0x1 Supported.
[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.

**Attributes** See the register summary in [Table 13-2 on page 13-5](#).

Figure 13-4 shows the MVFR0 bit assignments.

31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
VFP rounding modes			Short vectors		Square root		Divide		VFP exception trapping		Double precision	Single precision	A_SIMD registers		

**Figure 13-4 MVFR0 bit assignments**

Table 13-6 shows the MVFR0 bit assignments.

**Table 13-6 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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.
[7:4]	Single precision	Indicates the hardware support for VFP single-precision operations: 0x2 VFPv4 single-precision supported. See the <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> 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 <i>ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition</i> for more information.

### Floating-Point Exception Register

The FPEXC characteristics are:

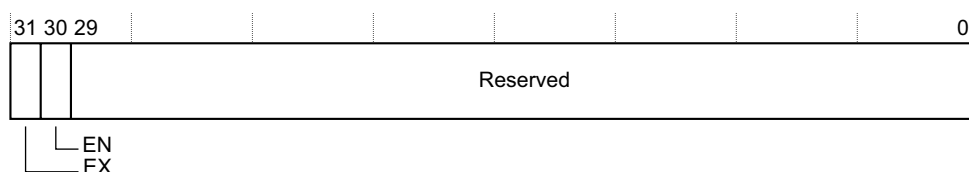
**Purpose** Provides a global enable for the Advanced SIMD and VFP extensions, and indicates how the state of these extensions is recorded.

**Usage constraints** Only accessible from PL1 or higher.

**Configurations** Available if VFP is implemented.

**Attributes** See the register summary in [Table 13-2 on page 13-5](#).

[Figure 13-5](#) shows the FPEXC bit assignments.



**Figure 13-5 FPEXC bit assignments**

Table 13-7 shows the FPEXC Register bit assignments.

**Table 13-7 FPEXC bit assignments**

Bits	Name	Function
[31]	EX	Exception bit. The Cortex-A17 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: <b>0</b> The Advanced SIMD and VFP extensions are disabled. <b>1</b> 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-A17 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-A17 MPCore processor signals. It contains the following sections:

- *About the signal descriptions on page A-2.*
- *Clock and reset signals on page A-3.*
- *Configuration signals on page A-4.*
- *Interrupt signals on page A-5.*
- *Asynchronous error signals on page A-6.*
- *Generic Timer signals on page A-7.*
- *Power control signals on page A-8.*
- *ACE master interface signals on page A-9.*
- *Peripheral port AXI master interface signals on page A-14.*
- *ACP slave interface signals on page A-17.*
- *External debug interface on page A-20.*
- *DFT interface signals on page A-25.*
- *MBIST interface signals on page A-26.*

## A.1 About the signal descriptions

The tables in this appendix list the Cortex-A17 MPCore processor signals, along with their direction, input or output, and a high-level description.

The external interface contains all the signals required for the maximum Cortex-A17 MPCore processor configuration. Signals representing features which are not configured are unused.

## A.2 Clock and reset signals

Table A-1 shows the clock, reset and reset control signals. All signals which include a 4-bit field, [N:0], encode up to four processors. For these signals, bit[0] represents processor 0, bit[1] represents processor 1, bit[2] represents processor 2, and bit[3] represents processor 3.

**Table A-1 Clock and reset signals**

Signal	Type	Description
<b>CLK</b>	Input	Global clock.
<b>CORECLK[N:0]</b>	Input	Clock to individual processor.
<b>CORECLKEN[N:0]</b>	Input	Clock enable to individual processor.
<b>L2RAMCLK</b>	Input	Clock for L2 data RAM.
<b>L2RAMCLKEN</b>	Input	Clock enable for L2 data RAM.
<b>nCOREPORESET[N:0]</b>	Input	All processor reset:
		<b>0</b> Apply reset to all processor logic that includes NEON and VFP, Debug, PTM, breakpoint and watchpoint logic. <b>1</b> Do not apply reset to all processor logic that includes NEON and VFP, Debug, PTM, breakpoint and watchpoint logic.
<b>nCORERESET[N:0]</b>	Input	Individual processor resets excluding Debug and ETM:
		<b>0</b> Apply reset to processor that includes NEON and VFP, but excludes Debug, PTM, breakpoint and watchpoint logic. <b>1</b> Do not apply reset to processor that includes NEON and VFP, but excludes Debug, PTM, breakpoint and watchpoint logic.
<b>nCOREPRESETDBG[N:0]</b>	Input	Debug logic resets:
		<b>0</b> Apply reset to debug, breakpoint and watchpoint logic. <b>1</b> Do not apply reset to debug, breakpoint and watchpoint logic.
<b>nL2RESET</b>	Input	L2 reset:
		<b>0</b> Apply reset to shared L2 memory system controller. <b>1</b> Do not apply reset to shared L2 memory system controller.
<b>nMBISTRESET</b>	Input	Force all logic into BIST mode.
<b>nTOPPRESETDBG</b>	Input	Resets the shared Debug-APB, CTI and CTM logic within the PCLKDBG clock domain
		<b>0</b> Apply reset to Debug-APB, CTI and CTM. <b>1</b> Do not apply reset to Debug-APB, CTI and CTM.

### A.3 Configuration signals

Table A-2 shows the configuration signals. All signals which include a 4-bit field, [N:0], encode up to four processors. For these signals, bit[0] represents processor 0, bit[1] represents processor 1, bit[2] represents processor 2, and bit[3] represents processor 3.

**Table A-2 Configuration signals**

Signal	Type	Description
CFGADDFILTENDNS[39:20]	Input	Peripheral port Non-secure end address.
CFGADDFILTENDS[39:20]	Input	Peripheral port Secure end address.
CFGADDRFILTEENS	Input	Enables peripheral port Non-secure address filtering.
CFGADDRFILTEENS	Input	Enables peripheral port Secure address filtering.
CFGADDRFILSTARTNS[39:20]	Input	Peripheral port Non-secure start address.
CFGADDRFILSTARTS[39:20]	Input	Peripheral port Secure start address.
CFGEND[N:0]	Input	Endianness configuration at reset. It sets the initial value of the EE bit in the CP15 <i>System Control Register</i> (SCTLR): <b>0</b> EE bit is LOW. <b>1</b> EE bit is HIGH. This signal is sampled at reset.
TEINIT[N:0]	Input	Default exception handling state. It sets the initial value of the TE bit in the CP15 <i>System Control Register</i> (SCTLR): <b>0</b> TE bit is LOW. <b>1</b> TE bit is HIGH. This signal is sampled at reset.
CLUSTERID[3:0]	Input	Value read in the Cluster ID field, bits [11:8], of the CP15 <i>Multiprocessor Affinity Register</i> (MPDIR). This bus is sampled at reset.
CP15SSDISABLE[N:0]	Input	Unused. Must be tied LOW.
IMINLN	Input	Sets the instruction cache minimum size in IminLine field CP15 <i>Cache Type Register</i> (CTR). <b>0</b> 32 bytes. <b>1</b> 64 bytes. This signal is sampled at reset.
L1RSTDISABLE[N:0]	Input	Disable L1 cache hardware invalidation at reset. This signal is sampled at reset. <p style="text-align: center;"><b>Note</b></p> ARM does not recommend disabling L1 cache hardware invalidation at reset.
L2RSTDISABLE	Input	Disable L2 cache hardware invalidation at reset. This signal is sampled at reset.
VINITHI[N:0]	Input	Location of the exception vectors at reset. It sets the initial value of the V bit in the CP15 <i>System Control Register</i> (SCTLR): <b>0</b> Exception vectors start at address 0x00000000. <b>1</b> Exception vectors start at address 0xFFFF0000. This signal is sampled at reset.



## A.4 Interrupt signals

Table A-3 shows the interrupt signals.

**Table A-3 Interrupt signals**

Signal	Type	Description
<b>nFIQ[N:0]</b>	Input	<p>FIQ request. Active-LOW, asynchronous fast interrupt request:</p> <p><b>0</b>                    Activate fast interrupt.</p> <p><b>1</b>                    Do not activate fast interrupt.</p> <p>The processor treats the <b>nFIQ</b> input as level-sensitive. To guarantee that an interrupt is taken, ensure the <b>nFIQ</b> input remains asserted until the processor acknowledges the interrupt.</p>
<b>nIRQ[N:0]</b>	Input	<p>IRQ request input lines. Active-LOW, asynchronous interrupt request:</p> <p><b>0</b>                    Activate interrupt.</p> <p><b>1</b>                    Do not activate interrupt.</p> <p>The processor treats the <b>nIRQ</b> input as level-sensitive. To guarantee that an interrupt is taken, ensure the <b>nIRQ</b> input remains asserted until the processor acknowledges the interrupt.</p>
<b>nVFIQ[N:0]</b>	Input	<p>Virtual FIQ request. Active-LOW, asynchronous fast interrupt request:</p> <p><b>0</b>                    Activate fast interrupt.</p> <p><b>1</b>                    Do not activate fast interrupt.</p> <p>The processor treats the <b>nVFIQ</b> input as level-sensitive. The <b>nVFIQ</b> input must be asserted until the processor acknowledges the interrupt.</p>
<b>nVIRQ[N:0]</b>	Input	<p>Virtual IRQ request. Active-LOW, asynchronous interrupt request:</p> <p><b>0</b>                    Activate interrupt.</p> <p><b>1</b>                    Do not activate interrupt.</p> <p>The processor treats the <b>nVIRQ</b> input as level-sensitive. The <b>nVIRQ</b> input must be asserted until the processor acknowledges the interrupt.</p>

## A.5 Asynchronous error signals

Table A-4 shows the asynchronous error signals.

**Table A-4 Asynchronous error signals**

Signal	Type	Description
<b>nAXIERRIRQ</b>	Output	Indicates an asynchronous error condition on an ACE transaction
<b>nECCERRIRQ<sup>a</sup></b>	Output	Indicates L2 RAM double-bit ECC error

a. This signal is only present if ECC is implemented.

## A.6 Generic Timer signals

Table A-5 shows the Generic Timer signals. All signals which include a 4-bit field, [N:0], encode up to four processors. For these signals, bit[0] represents processor 0, bit[1] represents processor 1, bit[2] represents processor 2, and bit[3] represents processor 3.

**Table A-5 Generic Timer signals**

Signal	Type	Description
CNTCLKEN	Input	Counter clock enable
CNTVALUEB[63:0]	Input	Global system counter value in binary format
nCNTHPIRQ[N:0]	Output	Hypervisor physical timer event
nCNTPSIRQ[N:0]	Output	Non-secure physical timer event
nCNTPSIRQ[N:0]	Output	Secure physical timer event
nCNTVIRQ[N:0]	Output	Virtual timer event

## A.7 Power control signals

Table A-6 shows the power control signals. All signals which include a 4-bit field, [3:0], encode up to four processors. For these signals, bit[0] represents processor 0, bit[1] represents processor 1, bit[2] represents processor 2, and bit[3] represents processor 3.

**Table A-6 Power control signals**

Signal	Type	Description
<b>CPUQACCEPTn[N:0]</b>	Output	Indicates that processor<N> accepts power controller request.
<b>CPUQACTIVE[N:0]</b>	Output	Indicates that processor<N> is active.
<b>CPUQDENY[N:0]</b>	Output	Indicates that processor<N> denies power controller retention request.
<b>CPUQREQn[N:0]</b>	Input	Indicates that power controller is ready to enter or exit retention for processor<n>.
<b>EVENTI</b>	Input	Event input for processor wake-up from WFE state. See <i>Event communication using WFE or SEV on page 2-20</i> for more information.
<b>EVENTO</b>	Output	Event output. Active when a SEV instruction is executed. See <i>Event communication using WFE or SEV on page 2-20</i> for more information.
<b>L2FLUSHREQ</b>	Input	Requests a L2 cache hardware flush.
<b>L2FLUSHDONE</b>	Output	Indicates that the L2 cache hardware flush is complete.
<b>L2QACCEPTn</b>	Output	Indicates that L2 RAM accepts power controller request.
<b>L2QACTIVE</b>	Output	Indicates that L2 RAM is active: <b>0</b> The power controller can request L2 cache RAM retention. <b>1</b> Power domain must be powered up.
<b>L2QDENY</b>	Output	Indicates that L2 cache RAM denies power controller retention request.
<b>L2QREQn</b>	Input	Indicates that power controller is ready to enter or exit retention for L2 cache RAM
<b>STANDBYWFE[N:0]</b>	Output	Indicates if a processor is in WFE standby mode: <b>0</b> Processor not in WFE standby mode. <b>1</b> Processor in WFE standby mode.
<b>STANDBYWFI[N:0]</b>	Output	Indicates if a processor is in WFI standby mode: <b>0</b> Processor not in WFI standby mode. <b>1</b> Processor in WFI standby mode.
<b>STANDBYWFIL2</b>	Output	Indicates if the L2 memory system is in WFI standby mode. This signal is active when the following conditions are met: <ul style="list-style-type: none"> <li>• All processors are in standby WFI.</li> <li>• <b>ACINACTM</b> and <b>AINACTSC</b> are asserted HIGH.</li> <li>• L2 memory system is idle.</li> </ul>

## A.8 ACE master interface signals

This section describes the ACE master interface signals:

- [Clock and configuration signals.](#)
- [Write address channel signals on page A-10.](#)
- [Write data channel signals on page A-10.](#)
- [Write data response channel signals on page A-11.](#)
- [Read address channel signals on page A-11.](#)
- [Read data channel signals on page A-11.](#)
- [Snoop address channel signals on page A-12.](#)
- [Snoop response channel signals on page A-12.](#)
- [Snoop data channel signals on page A-12.](#)
- [Read/write acknowledge signals on page A-13.](#)

For a complete description of the ACE interface signals, see the *ARM® AMBA® AXI and ACE Protocol Specification, AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite*.

### A.8.1 Clock and configuration signals

[Table A-7](#) shows the clock and configuration signals for the ACE master interface.

**Table A-7 Clock and configuration signals**

Signal	Type	Description
<b>ACINACTM</b>	Input	Snoop interface is inactive and no longer accepting requests.
<b>ACLKENM</b>	Input	ACE master bus clock enable. See <a href="#">Clocks on page 2-8</a> for more information.
<b>BROADCASTCACHEMAINT</b>	Input	Enable broadcasting of cache maintenance operations to downstream caches: <b>0</b> Cache maintenance operations are not broadcasted to downstream caches. This signal must be tied LOW.
<b>BROADCASTINNER<sup>a</sup></b>	Input	Enable broadcasting of Inner Shareable transactions: <b>0</b> Inner Shareable transactions are not broadcasted externally. <b>1</b> Inner Shareable transactions are broadcasted externally. If <b>BROADCASTINNER</b> is tied HIGH, you must also tie <b>BROADCASTOUTER</b> HIGH.
<b>BROADCASTOUTER<sup>a</sup></b>	Input	Enable broadcasting of Outer Shareable transactions: <b>0</b> Outer Shareable transactions are not broadcasted externally. <b>1</b> Outer Shareable transactions are broadcasted externally.
<b>SYSBARDISABLE</b>	Input	Disable broadcasting of barriers onto system bus: <b>1</b> Barriers are not broadcast onto the system bus. This is compatible with an AXI3 interconnect. This signal has no effect. ARM recommends that you tie it HIGH.

a. **BROADCASTINNER** and **BROADCASTOUTER** must always be equal.

## A.8.2 Write address channel signals

Table A-8 shows the address channel signals for the ACE master interface.

**Table A-8 address channel signals**

Signal	Type	Description
<b>AWADDRM[39:0]</b>	Output	Address
<b>AWBARM[1:0]</b>	Output	Barrier type
<b>AWBURSTM[1:0]</b>	Output	Burst type
<b>AWCACHM[3:0]</b>	Output	Cache type
<b>AWDOMAINM[1:0]</b>	Output	Domain type
<b>AWIDM[7:0]</b>	Output	Request ID
<b>AWLENM[7:0]</b>	Output	Burst length
<b>AWLOCKM</b>	Output	Lock type
<b>AWPROTM[2:0]</b>	Output	Protection type
<b>AWREADYM</b>	Input	Address ready
<b>AWSIZEM[2:0]</b>	Output	Burst size
<b>AWSNOOPM[2:0]</b>	Output	Snoop request type
<b>AWVALIDM</b>	Output	Address valid

## A.8.3 Write data channel signals

Table A-9 shows the write data channel signals for the ACE master interface.

**Table A-9 Write data channel signals**

Signal	Type	Description
<b>WDATAM[127:0]</b>	Output	Write data
<b>WIDM[7:0]</b>	Output	Write ID
<b>WLASTM</b>	Output	Write last indication
<b>WREADYM</b>	Input	Write ready
<b>WSTRBM[15:0]</b>	Output	Write byte-lane strobes
<b>WVALIDM</b>	Output	Write valid

## A.8.4 Write data response channel signals

Table A-10 shows the write data response channel signals for the ACE master interface.

**Table A-10 Write data response channel signals**

Signal	Type	Description
<b>BIDM[7:0]</b>	Input	Response ID
<b>BREADYM</b>	Output	Response ready
<b>BRESPM[1:0]</b>	Input	Write response
<b>BVALIDM</b>	Input	Response valid

## A.8.5 Read address channel signals

Table A-11 shows the read address channel signals for the ACE master interface.

**Table A-11 Read address channel signals**

Signal	Type	Description
<b>ARADDRM[39:0]</b>	Output	Address
<b>ARBARM[1:0]</b>	Output	Barrier type
<b>ARBURSTM[1:0]</b>	Output	Burst type
<b>ARCACHEM[3:0]</b>	Output	Cache type
<b>ARDOMAINM[1:0]</b>	Output	Domain type
<b>ARIDM[7:0]</b>	Output	Request ID
<b>ARLENM[7:0]</b>	Output	Burst length
<b>ARLOCKM</b>	Output	Lock type
<b>ARPROTM[2:0]</b>	Output	Protection type
<b>ARREADYM</b>	Input	Address ready
<b>ARSIZEM[2:0]</b>	Output	Burst size
<b>ARSNOOPM[3:0]</b>	Output	Snoop request type
<b>ARVALIDM</b>	Output	Address valid

## A.8.6 Read data channel signals

Table A-12 shows the read data channel signals for the ACE master interface.

**Table A-12 Read data channel signals**

Signal	Type	Description
<b>RDATAM[127:0]</b>	Input	Read data
<b>RIDM[7:0]</b>	Input	Read data ID
<b>RLASTM</b>	Input	Read last indication

**Table A-12 Read data channel signals (continued)**

Signal	Type	Description
<b>RREADYM</b>	Output	Read ready
<b>RRESPM[3:0]</b>	Input	Read response
<b>RVALIDM</b>	Input	Read valid

### A.8.7 Snoop address channel signals

Table A-13 shows the snoop address channel signals for the ACE master interface.

**Table A-13 Snoop address channel signals**

Signal	Type	Description
<b>ACADDRM[39:0]</b>	Input	Address
<b>ACPROTM[2:0]</b>	Input	Protection type
<b>ACREADYM</b>	Output	Address ready
<b>ACSNOOPM[3:0]</b>	Input	Transaction type
<b>ACVALIDM</b>	Input	Address valid

### A.8.8 Snoop response channel signals

Table A-14 shows the Snoop response channel signals for the ACE master interface.

**Table A-14 Snoop response channel signals**

Signal	Type	Description
<b>CRREADYM</b>	Input	Response ready
<b>CRRESPM[4:0]</b>	Output	Snoop response
<b>CRVALIDM</b>	Output	Response valid

### A.8.9 Snoop data channel signals

Table A-15 shows the coherency data channel handshake signals for the ACE master interface.

**Table A-15 Snoop data channel signals**

Signal	Type	Description
<b>CDDATAM[127:0]</b>	Output	Snoop data
<b>CDLASTM</b>	Output	Snoop last indication
<b>CDREADYM</b>	Input	Snoop ready
<b>CDVALIDM</b>	Output	Snoop valid



### A.8.10 Read/write acknowledge signals

Table A-16 shows the read/write acknowledge signals for the ACE master interface.

**Table A-16 Read/write acknowledge signals**

Signal	Type	Description
<b>RACKM</b>	Output	Read acknowledge
<b>WACKM</b>	Output	Write acknowledge

## A.9 Peripheral port AXI master interface signals

This section describes the peripheral port AXI master interface signals:

- [Clock and configuration signals.](#)
- [Write address channel signals.](#)
- [Write data channel signals on page A-15.](#)
- [Write data response channel signals on page A-15.](#)
- [Read address channel signals on page A-15.](#)
- [Read data channel signals on page A-16.](#)
- [Read/write acknowledge signals on page A-13.](#)

For a complete description of the AXI interface signals, see the *ARM® AMBA® AXI and ACE Protocol Specification, AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite*.

### A.9.1 Clock and configuration signals

[Table A-7 on page A-9](#) shows the clock and configuration signals for the peripheral port AXI master interface.

**Table A-17 Clock and configuration signals**

Signal	Type	Description
ACLKENMP	Input	AXI master bus clock enable. See <a href="#">Clocks on page 2-8</a> for more information

### A.9.2 Write address channel signals

[Table A-8 on page A-10](#) shows the address channel signals for the peripheral port AXI master interface.

**Table A-18 address channel signals**

Signal	Type	Description
AWADDRMP[39:0]	Output	Address
AWBURSTMP[1:0]	Output	Burst type
AWCACHEMP[3:0]	Output	Cache type
AWIDMP[7:0]	Output	Request ID
AWLENMP[3:0]	Output	Burst length
AWLOCKMP[1:0]	Output	Lock type
AWPROTMP[2:0]	Output	Protection type
AWREADYMP	Input	Address ready
AWSIZEMP[2:0]	Output	Burst size
AWVALIDMP	Output	Address valid

### A.9.3 Write data channel signals

[Table A-9 on page A-10](#) shows the write data channel signals for the peripheral port AXI master interface.

**Table A-19 Write data channel signals**

Signal	Type	Description
<b>WDATAMP[127:0]</b>	Output	Write data
<b>WIDMP[7:0]</b>	Output	Write ID
<b>WLASTMP</b>	Output	Write last indication
<b>WREADYMP</b>	Input	Write ready
<b>WSTRBMP[15:0]</b>	Output	Write byte-lane strobes
<b>WVALIDMP</b>	Output	Write valid

### A.9.4 Write data response channel signals

[Table A-10 on page A-11](#) shows the write data response channel signals for the peripheral port AXI master interface.

**Table A-20 Write data response channel signals**

Signal	Type	Description
<b>BIDMP[7:0]</b>	Input	Response ID
<b>BREADYMP</b>	Output	Response ready
<b>BRESPMP[1:0]</b>	Input	Write response
<b>BVALIDMP</b>	Input	Response valid

### A.9.5 Read address channel signals

[Table A-11 on page A-11](#) shows the read address channel signals for the peripheral port AXI master interface.

**Table A-21 Read address channel signals**

Signal	Type	Description
<b>ARADDRMP[39:0]</b>	Output	Address
<b>ARBURSTMP[1:0]</b>	Output	Burst type
<b>ARCACHEMP[3:0]</b>	Output	Cache type
<b>ARIDMP[7:0]</b>	Output	Request ID
<b>ARLENMP[3:0]</b>	Output	Burst length
<b>ARLOCKMP[1:0]</b>	Output	Lock type
<b>ARPROTMP[2:0]</b>	Output	Protection type

**Table A-21 Read address channel signals (continued)**

Signal	Type	Description
<b>ARREADYMP</b>	Input	Address ready
<b>ARSIZEMP[2:0]</b>	Output	Burst size
<b>ARVALIDMP</b>	Output	Address valid

### A.9.6 Read data channel signals

[Table A-12 on page A-11](#) shows the read data channel signals for the peripheral port AXI master interface.

**Table A-22 Read data channel signals**

Signal	Type	Description
<b>RDATAMP[127:0]</b>	Input	Read data
<b>RIDMP[7:0]</b>	Input	Read data ID
<b>RLASTMP</b>	Input	Read last indication
<b>RREADYMP</b>	Output	Read ready
<b>RRESPMP[1:0]</b>	Input	Read response
<b>RVALIDMP</b>	Input	Read valid

## A.10 ACP slave interface signals

The ACP is an implementation option. These signals are only present if the ACP is implemented.

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-18.](#)
- [Read data channel signals on page A-19.](#)

### A.10.1 Clock and configuration signals

[Table A-23](#) shows the clock and configuration signals for the ACP.

**Table A-23 Clock and configuration signals**

Signal	Type	Description
ACLKENS	Input	AXI slave bus clock enable
AINACTSC	Input	AXI slave inactive and no longer accepting requests

### A.10.2 Write address channel signals

[Table A-24](#) shows the write address channel signals for the ACP.

**Table A-24 Write address channel signals**

Signal	Type	Description
AWADDRSC[39:0]	Input	Address
AWBURSTSC[1:0]	Input	Burst type
AWCACHESC[3:0]	Input	Cache type
AWIDSC[4:0]	Input	Request ID
AWLENSC[1:0]	Input	Burst length
AWLOCKSC	Input	Write lock type
AWPROTSC[2:0]	Input	Protection type
AWREADYSC	Output	Address ready
AWSIZESC[2:0]	Input	Burst size
AWVALIDSC	Input	Address valid

### A.10.3 Write data channel signals

Table A-25 shows the write data channel signals for the ACP.

**Table A-25 Write data channel signals**

Signal	Type	Description
WDATASC[127:0]	Input	Write data
WLASTSC	Input	Write last
WREADYSC	Output	Write ready
WSTRBSC[15:0]	Input	Write strobes
WVALIDSC	Input	Write valid

### A.10.4 Write response channel signals

Table A-26 shows the write response channel signals for the ACP.

**Table A-26 Write response channel signals**

Signal	Type	Description
BIDSC[4:0]	Output	Response ID
BREADYSC	Input	Response ready
BRESPSC[1:0]	Output	Write response
BVALIDSC	Output	Response valid

### A.10.5 Read address channel signals

Table A-27 shows the read address channel signals for the ACP.

**Table A-27 Read address channel signals**

Signal	Type	Description
ARADDRSC[39:0]	Input	Address
ARBURSTSC[1:0]	Input	Burst type
ARCACHESC[3:0]	Input	Cache type
ARIDSC[4:0]	Input	Request ID
ARLENSC[1:0]	Input	Burst length
ARLOCKSC	Input	Read lock type
ARPROTSC[2:0]	Input	Protection type
ARREADYSC	Output	Address ready
ARSIZE[2:0]	Input	Burst size
ARVALIDSC	Input	Address valid

## A.10.6 Read data channel signals

Table A-28 shows the read data channel signals for the ACP.

**Table A-28 Read data channel signals**

Signal	Type	Description
<b>RDATASC[127:0]</b>	Output	Read data
<b>RIDSC[4:0]</b>	Output	Read ID
<b>RLASTSC</b>	Output	Read last
<b>RREADYSC</b>	Input	Read ready
<b>RRESPSC[1:0]</b>	Output	Read response
<b>RVALIDSC</b>	Output	Read valid

## A.11 External debug interface

The following sections describe the external debug interface signals:

- [APB Interface signals](#).
- [Debug authentication interface signals on page A-21](#).
- [Miscellaneous Debug signals on page A-21](#).
- [PTM interface signals on page A-22](#).
- [PMU signals on page A-23](#).

### A.11.1 APB Interface signals

Table A-29 shows the APB Interface signals.

**Table A-29 APB Interface signals**

Signal	Type	Description
<b>PADDRDBG[16:2]</b>	Input	APB Address bus bits[16:2].
<b>PADDRDBG31</b>	Input	APB address bus bit[31]: <b>0</b> Not an external debugger access. <b>1</b> External debugger access.
<b>PCLKDBG</b>	Input	APB clock.
<b>PCLKENDBG</b>	Input	APB clock enable.
<b>PENABLEDBG</b>	Input	Indicates the second and subsequent cycles of an APB transfer.
<b>PRDATADB[31:0]</b>	Output	APB read data bus.
<b>PREADYDBG</b>	Output	APB slave ready. An APB slave can assert <b>PREADYDBG</b> to extend a transfer by inserting wait states.
<b>PSELDBG</b>	Input	Debug bus access.
<b>PSLVERRDBG</b>	Output	APB slave transfer error: <b>0</b> No transfer error. <b>1</b> Transfer error.
<b>PWDATADB[31:0]</b>	Input	APB write data bus
<b>PWRITEDBG</b>	Input	APB read or write signal: <b>0</b> Reads from APB. <b>1</b> Writes to APB.



### A.11.2 Debug authentication interface signals

Table A-30 shows the debug authentication interface signals. All signals which include a 4-bit field, [3:0], encode up to four processors. For these signals, bit[0] represents processor 0, bit[1] represents processor 1, bit[2] represents processor 2, and bit[3] represents processor 3.

**Table A-30 Debug authentication interface signals**

Signal	Type	Description
DBGEN[N:0]	Input	Invasive debug enable:
		<b>0</b> Not enabled.
		<b>1</b> Enabled.
NIDEN[N:0]	Input	Non-invasive debug enable:
		<b>0</b> Not enabled.
		<b>1</b> Enabled.
SPIDEN[N:0]	Input	Secure privileged invasive debug enable:
		<b>0</b> Not enabled.
		<b>1</b> Enabled.
SPNIDEN[N:0]	Input	Secure privileged non-invasive debug enable:
		<b>0</b> Not enabled.
		<b>1</b> Enabled.

### A.11.3 Miscellaneous Debug signals

Table A-31 shows the miscellaneous Debug signals. All signals which include an N+1-bit field, [N:0], encode up to N+1 processors. For these signals, bit[0] represents processor 0, bit[1] represents processor 1, bit[2] represents processor 2, and bit[N] represents processor N.

**Table A-31 Miscellaneous Debug signals**

Signal	Type	Description
COMMRX[N:0]	Output	Communications channel receive. Receive portion of Data Transfer Register full flag:
		<b>0</b> Empty.
		<b>1</b> Full.
COMMTX[N:0]	Output	Communication transmit channel. Transmit portion of Data Transfer Register empty flag:
		<b>0</b> Full.
		<b>1</b> Empty.
DBGACK[N:0]	Output	Debug acknowledge:
		<b>0</b> External debug request not acknowledged.
		<b>1</b> External debug request acknowledged.
DBGNOPWRDWN[N:0]	Output	No power-down request.
DBGPWRDUP[N:0]	Input	Processor powered up.
DBGPWRUPREQ[N:0]	Output	Processor powerup request.
DBGROMADDR[39:12]	Input	Specifies bits [39:12] of the ROM table physical address. If the address cannot be determined, tie these signals off to 0. These pins are only sampled during reset of the processor.

Table A-31 Miscellaneous Debug signals (continued)

Signal	Type	Description
<b>DBGROMADDRV</b>	Input	Valid signal for <b>DBGROMADDR</b> . If the address cannot be determined, tie this signal LOW. This pin is only sampled during reset of the processor.
<b>DBGRSTREQ[N:0]</b>	Output	Warm reset request.
<b>DBGSELFADDR[39:15]</b>	Input	Specifies bits [39:15] 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 is used, bits [16:15] must be tied to 0b10. If the offset cannot be determined, tie <b>DBGSELFADDR[39:15]</b> to 0. These pins are only sampled during reset of the processor.
<b>DBGSELFADDRV</b>	Input	Valid signal for <b>DBGSELFADDR</b> . If the offset cannot be determined, tie this signal LOW. This pin is only sampled during reset of the processor.
<b>DBGSWENABLE[N:0]</b>	Input	Debug software access enable: <b>0</b> Not enabled. <b>1</b> Enabled, access by the software through the Extended CP14 interface and memory mapped interface is permitted.
<b>EDBGRQ[N:0]</b>	Input	External debug request: <b>0</b> No external debug request. <b>1</b> External debug request. The processor treats the <b>EDBGRQ</b> input as level-sensitive. The <b>EDBGRQ</b> input must be asserted until the processor asserts <b>DBGACK</b> .

#### A.11.4 PTM interface signals

This section describes the PTM interface in:

- [ATB interface](#).
- [Miscellaneous PTM interface on page A-23](#).

#### ATB interface

[Table A-32](#) 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-32 ATB interface signals

Signal	Type	Description
<b>AFREADYMx</b>	Output	FIFO flush acknowledge: <b>0</b> FIFO flush not complete. <b>1</b> FIFO flush complete.
<b>AFVALIDMx</b>	Input	FIFO flush request.
<b>ATBYTESMx[1:0]</b>	Output	CoreSight ATB device data size: 0b00 1 byte. 0b01 2 byte. 0b10 3 byte. 0b11 4 byte.

Table A-32 ATB interface signals (continued)

Signal	Type	Description
ATCLKEN	Input	ATB clock enable.
ATDATAMx[31:0]	Output	ATB data bus.
ATIDMx[6:0]	Output	ATB trace source identification.
ATREADYMx	Input	ATB device ready: <b>0</b> Not ready. <b>1</b> Ready.
ATVALIDMx	Output	ATB valid data: <b>0</b> No valid data. <b>1</b> Valid data.

### Miscellaneous PTM interface

Table A-33 shows the miscellaneous Program Trace Macrocell signals.

Table A-33 Miscellaneous PTM interface signals

Signal	Type	Description
SYNCREQx	Input	Synchronization request. The input must be driven HIGH for one <b>ATCLKEN</b> cycle.
TSCLKCHANGE	Input	<b>CLK</b> frequency change indicator. The input must be driven HIGH for a single <b>ATCLKEN</b> cycle if the frequency of <b>CLK</b> is changed. If not used, this input must be tied LOW.
TSVALUEB[63:0]	Input	Global system timestamp value in binary format.

### A.11.5 PMU signals

Table A-34 shows the performance monitoring signals.

Table A-34 Performance Monitoring Unit signals

Signal	Type	Description
nPMUIRQ[N:0] <sup>a</sup>	Output	PMU interrupt signals
PMUEVENTx[37:0] <sup>b</sup>	Output	PMU Event signals <sup>c</sup>
PMUPL1[N:0]	Output	PMU Privilege level 1 event
PMUPL2[N:0]	Output	PMU Privilege level 2 event
PMUSECURE[N:0]	Output	PMU Secure event

- a. The 4-bit field, [3:0], encodes up to four processors. Bit[0] represents processor 0, bit[1] represents processor 1, bit[2] represents processor 2, and bit[3] represents processor 3.
- b. PMU Event signals are asserted for one **CORECLK** cycle.
- c. **PMUEVENT[37:33]** are reserved.

## A.12 Cross trigger channel interface

Table A-35 shows the cross trigger channel interface signals. The value of **N** is one less than the number of processors in your design.

**Table A-35 Cross trigger channel interface signals**

Signal	Type	Description
<b>CIHSBYPASS[3:0]</b>	Input	Cross trigger channel interface handshake bypass.
<b>CISBYPASS</b>	Input	Cross trigger channel interface sync bypass.
<b>CTICHIN[3:0]</b>	Input	Cross trigger channel input. Each bit represents a valid channel input: <b>0</b> Channel input inactive. <b>1</b> Channel input active.
<b>CTICHINACK[3:0]</b>	Output	Cross trigger channel input acknowledge.
<b>CTICHOUT[3:0]</b>	Output	Cross trigger channel output. Each bit represents a valid channel output: <b>0</b> Channel output inactive. <b>1</b> Channel output active.
<b>CTICHOUTACK[3:0]</b>	Input	Cross trigger channel output acknowledge.
<b>CTIEXTTRIG[N:0]</b>	Output	Cross trigger external trigger output.
<b>CTIEXTTRIGACK[N:0]</b>	Input	Cross trigger external trigger output acknowledge.
<b>nCTIIRQ[N:0]</b>	Output	Active-LOW cross trigger interrupt output: <b>0</b> Interrupt active. <b>1</b> Interrupt not active.

## A.13 DFT interface signals

Table A-36 shows the DFT interface signals.

**Table A-36 DFT interface signals**

Signal name	Type	Description
DFTRAMHOLD	Input	Disables the RAM chip selects during scan testing
DFTRSTDISABLE[1:0]	Input	Disables internal synchronized reset during scan shift
DFTSE	Input	Scan shift enable, forces on the clock grids during scan shift

## A.14 MBIST interface signals

MBIST is an implementation option. These signals are only present when MBIST is implemented.

Table A-37 shows the common MBIST interface signals.

**Table A-37 MBIST interface signals**

Signal	Type	Description
<b>nMBISTRESET</b>	Input	MBIST reset

Table A-38 shows the MBIST L1 interface signals.

**Table A-38 MBIST L1 interface signals**

Signal	Type	Description
<b>MBISTACK1[3:0]</b>	Output	MBIST test acknowledge
<b>MBISTADDR1[10:0]</b>	Input	MBIST logical address in array
<b>MBISTARRAY[6:0]</b>	Input	MBIST array selector
<b>MBISTBE1[69:0]</b>	Input	MBIST bit write enable
<b>MBISTCFG1[1:0]</b>	Input	MBIST all RAMs enable
<b>MBISTINDATA1[127:0]</b>	Input	MBIST data in
<b>MBISTOUTDATA1[127:0]</b>	Output	MBIST data out
<b>MBISTREADEN1</b>	Input	MBIST Read enable
<b>MBISTREQ1[3:0]</b>	Input	MBIST test request
<b>MBISTWRITEEN1</b>	Input	MBIST write enable

Table A-39 shows the MBIST L2 Data and Buffer interface signals.

**Table A-39 MBIST L2 Data and Buffer interface signals**

Signal	Type	Description
<b>MBISTACK2</b>	Output	MBIST test acknowledge
<b>MBISTADDR2[16:0]</b>	Input	MBIST logical address in array
<b>MBISTARRAY2[2:0]</b>	Input	MBIST array selector
<b>MBISTBE2</b>	Input	MBIST bit write enable
<b>MBISTCFG2[10:0]</b>	Input	MBIST all RAMs enable
<b>MBISTINDATA2[511:0]</b>	Input	MBIST data in. L2 ECC disabled
<b>MBISTINDATA2[575:0]</b>		MBIST data in. L2 ECC enabled
<b>MBISTOUTDATA2[511:0]</b>	Output	MBIST data out. L2 ECC disabled
<b>MBISTOUTDATA2[575:0]</b>		MBIST data out. L2 ECC enabled

**Table A-39 MBIST L2 Data and Buffer interface signals (continued)**

Signal	Type	Description
<b>MBISTREADEN2</b>	Input	MBIST Read enable
<b>MBISTREQ2</b>	Input	MBIST test request
<b>MBISTWRITEEN2</b>	Input	MBIST write enable

Table A-40 shows the MBIST L2 Tag and SCU interface signals.

**Table A-40 MBIST L2 Tag and SCU interface signals**

Signal	Type	Description
<b>MBISTACK3</b>	Output	MBIST test acknowledge
<b>MBISTADDR3[12:0]</b>	Input	MBIST logical address in array
<b>MBISTARRAY3[7:0]</b>	Input	MBIST array selector
<b>MBISTB3E[123:0]</b>	Input	MBIST bit write enable
<b>MBISTCFG3[10:0]</b>	Input	MBIST all RAMs enable
<b>MBISTINDATA3[123:0]</b>	Input	MBIST data in
<b>MBISTOUTDATA3[123:0]</b>	Output	MBIST data out
<b>MBISTREADEN3</b>	Input	MBIST Read enable
<b>MBISTREQ3</b>	Input	MBIST test request
<b>MBISTWRITEEN3</b>	Input	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 text clarifying the use of <b>CORECLKEN[N:0]</b>	<a href="#">Clock enables on page 2-8</a>	All revisions
Added text to end of section regarding interface signals	<a href="#">Clock enables on page 2-8</a>	All revisions
Updated first bullet point and added second bullet point	<a href="#">Resets on page 2-12</a>	All revisions
Updated first sentence of section L2 RAMs dynamic retention	<a href="#">State retention control on page 2-20</a>	All revisions
Added text regarding assertion of <b>nCORERESET[N:0]</b> and <b>nCOREPRESETDBG[N:0]</b> in multiprocessor device shutdown mode step 7.	<a href="#">Power modes on page 2-25</a>	All revisions
Changed ACTLR[3] from reserved to ASSE	<a href="#">Auxiliary Control Register on page 4-55</a>	r1p0
Updated functional description of bit FLT_END_ADDR	<a href="#">Table 4-74 on page 4-87</a>	All revisions
Updated Usage constraints, fourth bullet point	<a href="#">L2 Control Register on page 4-77</a>	All revisions



Table B-2 Differences between issue A and issue B (continued)

Change	Location	Affects
Changed L2CTLR[20] from reserved to SFEN	<a href="#">L2 Control Register on page 4-77</a>	r1p0
Updated first bullet point	<a href="#">Memory types on page 5-10</a>	All revisions
Updated table row AWID[4:0] and ARID[4:0]	<a href="#">Table 7-2 on page 7-8 and Table 7-3 on page 7-9</a>	All revisions
Revised text describing reset requirements	<a href="#">Resets on page 2-12</a>	All revisions
Added timing requirement for logic state in powerup reset	<a href="#">Resets on page 2-12</a>	All revisions
Revised description	<a href="#">Processor Wait for Event on page 2-18</a>	All revisions
Revised description	<a href="#">Event communication using WFE or SEV on page 2-20</a>	All revisions
Updated MIDR.Variant for major revision 1	<a href="#">Table 4-29 on page 4-27</a>	r1p0
Added description of L2CTLR.IWINC	<a href="#">Table 4-68 on page 4-79</a>	All revisions
Revised description of implementation defined TEX encoding	<a href="#">Table 5-1 on page 5-11</a>	All revisions
Revised description of PLI instruction	<a href="#">PLD, PLDW, and PLI instructions on page 6-9</a>	All revisions
Revised description of ACP error	<a href="#">ACP requests on page 7-12</a>	All revisions
Corrected the Extended External input bus size and revised the description	<a href="#">Table 11-10 on page 11-22</a>	All revisions

Table B-3 Differences between issue B and issue C

Change	Location	Affects
Clarified state of clock enables during low-power state	<a href="#">Clock enables on page 2-8</a>	r1p0
Corrected figure	<a href="#">Figure 9-8 on page 9-17</a>	All revisions
Corrected reset value of MIDR	<a href="#">Table 4-29 on page 4-27</a>	r1p0
Updated description of CNTCLKEN	<a href="#">CNTCLKEN on page 2-12</a>	All revisions
Updated description of reset repeaters	<a href="#">Resets on page 2-12</a>	All revisions
Added note	<a href="#">Individual processor shutdown mode on page 2-25</a>	All revisions
Added register descriptions	<a href="#">Diagnostic control registers on page 4-84</a>	r1p1
Added register description	<a href="#">Diagnostic common control register on page 4-85</a>	r1p1
Corrected reset value HSCTLR[21] in figure	<a href="#">Figure 4-28 on page 4-62</a>	All revisions
Corrected figure	<a href="#">Figure 9-10 on page 9-20</a>	All revisions
Updated description of PCLKENDBG	<a href="#">Clock enables on page 2-8</a>	r1p1
Updated figure	<a href="#">Figure 2-6 on page 2-11</a>	r1p1
Updated figure	<a href="#">Figure 2-7 on page 2-11</a>	r1p1