Cortex™-A7 MPCore™

Revision: r0p4

Technical Reference Manual

ARM®

ARM DDI 0464E (ID112412)

Cortex-A7 MPCore
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
03 October 2011	A	Non-Confidential	First release for r0p0
09 November 2011	B	Non-Confidential	First release for r0p1
11 January 2012	C	Non-Confidential	First release for r0p2
15 May 2012	D	Non-Confidential	First release for r0p3
19 November 2012	E	Confidential-Draft	First release for r0p4

Proprietary Notice

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
Cortex-A7 MPCore Technical Reference Manual

Preface

This preface introduces the *Cortex-A7 MPCore Technical Reference Manual*. It contains the following sections:

About this book

This book is for the Cortex-A7 MPCore processor. This is a multiprocessor device that has between one to four processors.

Product revision status

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

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

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

Intended audience

This book is written for hardware and software engineers implementing the Cortex-A7 MPCore processor in system designs. It provides information that enables designers to integrate the Cortex-A7 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-A7 MPCore processor and descriptions of the major features.

Chapter 2 *Functional Description*

Read this for a description of the functionality of the Cortex-A7 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 AXI Coherency Extensions (ACE) attributes.

Chapter 8 *Generic Interrupt Controller*

Read this for a description of the *Generic Interrupt Controller* (GIC).

Chapter 9 *Generic Timer*

Read this for a description of the timers.

Chapter 10 *Debug*

Read this for a description of the support for debug.

Chapter 11 *Performance Monitoring Unit*

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

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

Conventions that this book can use are described in:

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

Typographical

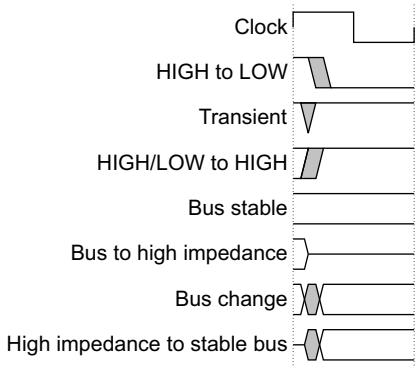
The following table describes the typographical conventions:

Style	Purpose
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace` *`italic`*	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

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:

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

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

Additional reading

This section lists 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:

- *AMBA® APB Protocol Specification* (ARM IHI 0024).
- *AMBA AXI and ACE Protocol Specification, AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite* (ARM IHI 0022).
- *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* (ARM DDI 0406).

- *ARM Generic Interrupt Controller Architecture Specification* (ARM IHI 0048).
- *CoreSight™ Architecture Specification* (ARM IHI 0029).
- *CoreSight ETM-A7 Technical Reference Manual* (ARM DDI 0468).
- *CoreSight ETM-A7 Configuration and Sign-off Guide* (ARM DII 0261).
- *CoreSight SoC Technical Reference Manual* (ARM DDI 0480).
- *Embedded Trace Macrocell Architecture Specification* (ARM IHI 0014).
- *Cortex-A Series Programmer's Guide* (ARM DEN 0013).
- *Cortex-A7 MPCore Configuration and Sign-off Guide* (ARM DII 0256).
- *Cortex-A7 MPCore Floating-Point Unit Technical Reference Manual* (ARM DDI 0463).
- *Cortex-A7 MPCore Integration Manual* (ARM DIT 0017).
- *Cortex-A7 MPCore NEON Media Processing Engine Technical Reference Manual* (ARM DDI 0462).

Other publications

This section lists relevant documents published by third parties:

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

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`. Give:

- The title.
- The number, ARM DDI 0464E.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

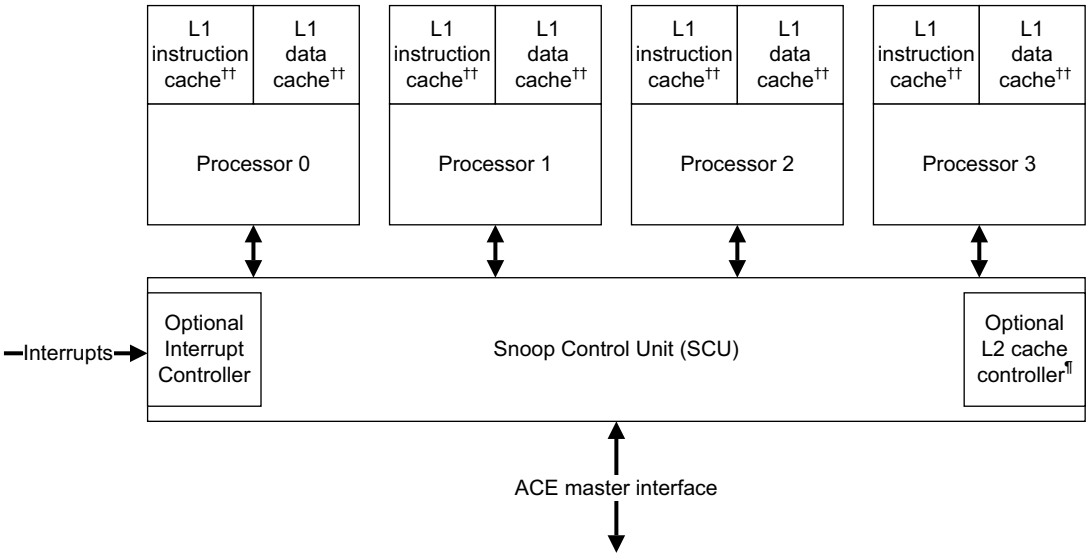
Chapter 1
Introduction

This chapter introduces the Cortex-A7 MPCore processor and its features. It contains the following sections:

1.1 About the Cortex-A7 MPCore processor

The Cortex-A7 MPCore processor is a high-performance, low-power processor that implements the ARMv7-A architecture. The Cortex-A7 MPCore processor has one to four processors in a single multiprocessor device with a L1 cache subsystem, an optional integrated GIC, and an optional L2 cache controller.

Figure 1-1 shows an example of a Cortex-A7 MPCore configuration with four processors.



††Configurable L1 cache size 8KB, 16KB, 32KB, or 64KB
¶Configurable L2 cache size None, 128KB, 256KB, 512KB, 1024KB

Figure 1-1 Example multiprocessor configuration

You can implement one processor in a Cortex-A7 MPCore processor design. In this configuration, the SCU is still provided. See *Configurable options* on page 1-7 for more information.

See *Processor components* on page 2-2 for more information about the functional components.

1.2 Compliance

The Cortex-A7 MPCore processor complies with, or implements, the specifications described in:

- *ARM architecture*.
- *Advanced Microcontroller Bus Architectures*.
- *Debug architecture*.
- *Generic Interrupt Controller architecture* on page 1-4.
- *Generic Timer architecture* on page 1-4.

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

1.2.1 ARM architecture

The Cortex-A7 MPCore processor implements the ARMv7-A architecture with the following architecture extensions:

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

  Note

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

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

See the *ARM Architecture Reference Manual*.

1.2.2 Advanced Microcontroller Bus Architectures

The Cortex-A7 MPCore processor ACE and debug interfaces comply with the:

- AMBA 4 *Advanced eXtensible Interface* (AXI) protocol. See the *AMBA AXI Protocol Specification*.
- AMBA 3 *Advanced Peripheral Bus* (APB) protocol. See the *AMBA APB Protocol Specification.*

1.2.3 Debug architecture

The Cortex-A7 MPCore processor implements the ARMv7.1 ARM Debug architecture, that complies with the CoreSight architecture. See the *CoreSight Architecture Specification* and the *ARM Architecture Reference Manual*.

1.2.4 Generic Interrupt Controller architecture

The Cortex-A7 MPCore processor implements the *Generic Interrupt Controller* (GIC) v2.0 architecture that includes support for the Virtualization Extensions. See the *ARM Generic Interrupt Controller Architecture Specification*.

1.2.5 Generic Timer architecture

The Cortex-A7 MPCore processor implements the ARM Generic Timer architecture that includes support for the Virtualization Extensions. See the *ARM Architecture Reference Manual*.

1.3 Features

The processor includes the following features:

- Full implementation of the ARMv7-A architecture instruction set with the architecture extensions listed in *Compliance* on page 1-3.
- In-order pipeline with direct and indirect branch prediction.
- Harvard *Level 1* (L1) memory system with a *Memory Management Unit* (MMU).
- *Level 2* (L2) memory system.
- APB debug interface that supports integer processor clock ratios up to and including 1:1.
- Trace support through an *Embedded Trace Macrocell* (ETM) interface.
- Optional VFPv4-D16 FPU with trapless execution or *Media Processing Engine* (MPE) with NEON technology.

  Note

  When FPU option is selected without NEON, the FPU is VFPv4-D16 and uses 16 double-precision registers. When the FPU is implemented with NEON, the FPU is VFPv4-D32 and uses 32 double-precision registers. This register bank is shared with NEON.

1.4 Interfaces

The Cortex-A7 MPCore processor has the following external interfaces:

- ACE.
- APB.
- ETM.
- *Design For Test* (DFT).
- *Memory Built-In Self Test* (MBIST) controller.

See *Interfaces* on page 2-8 for more information on these interfaces.

1.5 Configurable options

Table 1-1 shows the Cortex-A7 MPCore RTL configurable options.

Table 1-1 Configurable options for the Cortex-A7 MPCore RTL

Feature	Range of options	Default value
Processor-level configuration options:[a]		
Floating-Point Unit (FPU) or *Media Processing Engine* (NEON)	Neither, FPU only, FPU and NEON[b]	FPU and NEON
Global configuration options:		
Instruction cache size	8KB, 16KB, 32KB, or 64KB	32KB
Data cache size	8KB, 16KB, 32KB, or 64KB	32KB
L2 cache controller	Present or not present	Present
L2 cache sizes	None, 128KB, 256KB, 512KB, 1024KB	256KB
L2 data RAM cycle latency	2 cycles or 3 cycles	2 cycles
Shared Peripheral Interrupts	0-480 in steps of 32	0
Number of processors	1, 2, 3, or 4	2
Integrated GIC	Present or not present	Not present

a. These options can be configured independently for each processor in the Cortex-A7 MPCore processor.
b. If NEON is selected, FPU is included.

1.6 Test features

The Cortex-A7 MPCore processor is delivered as fully-synthesizable RTL and is a fully-static design. Scan-chains and test wrappers for production test can be inserted into the design by the synthesis tools during implementation. See the relevant implementation reference methodology documentation for more information.

You can perform production test of the cache and TLB RAMs, and the L2 cache RAMs if present, through a dedicated MBIST interface. See the *Cortex-A7 MPCore Integration Manual* for more information about this interface, and how to control it.

1.7 Product documentation and design flow

This section describes the Cortex-A7 MPCore books, how they relate to the design flow, and the relevant architectural standards and protocols.

See *Additional reading* on page ix for more information about the books described in this section.

1.7.1 Documentation

The Cortex-A7 MPCore 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-A7 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-A7 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-A7 MPCore processor into a SoC. It includes a description of the pins that the integrator must tie off to configure the macrocell for the required integration. Some of the integration is affected by the configuration options used when implementing the Cortex-A7 MPCore processor.

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

1.7.2 Design flow

The Cortex-A7 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 might include integrating the RAMs into the design.

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

Programming The system programmer develops the software required to configure and initialize the 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-A7 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 processor by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made. They can also limit the options available to the software.

Software configuration

The programmer configures the processor by programming particular values into software-visible registers. This affects the behavior of the 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.

1.8 Product revisions

This section describes the differences in functionality between product revisions:

r0p0-r0p1 Functional changes are:

- ID register value changed to reflect product revision status:
  Main ID Register `0x410FC071`
  Debug Peripheral ID 2 `0x1B`
  Performance Monitors Peripheral ID2 `0x1B`
- support for a redundant internal GIC, see *GIC configuration* on page 8-5
- various engineering errata fixes.

r0p1-r0p2 Functional changes are:

- ID register value changed to reflect product revision status:
  Main ID Register `0x410FC072`
  Debug Peripheral ID 2 `0x2B`
  Performance Monitors Peripheral ID2 `0x2B`
- various engineering errata fixes.

r0p2-r0p3 Functional changes are:

- ID register value changed to reflect product revision status:
  Main ID Register `0x410FC073`
  Debug Peripheral ID 2 `0x3B`
  Performance Monitors Peripheral ID2 `0x3B`
- enhancement of the read allocate mode operation to optimize performance
- various engineering errata fixes.

r0p3-r0p4 Functional changes are:

- ID register value changed to reflect product revision status:
  Main ID Register `0x410FC074`
  Debug Peripheral ID 2 `0x4B`
  Performance Monitors Peripheral ID2 `0x4B`
- various engineering errata fixes.

Chapter 2
Functional Description

This chapter describes the functionality of the Cortex-A7 MPCore processor. It contains the following sections:

2.1 About the Cortex-A7 MPCore processor functions

Figure 2-1 shows a top-level functional diagram of the Cortex-A7 MPCore processor.

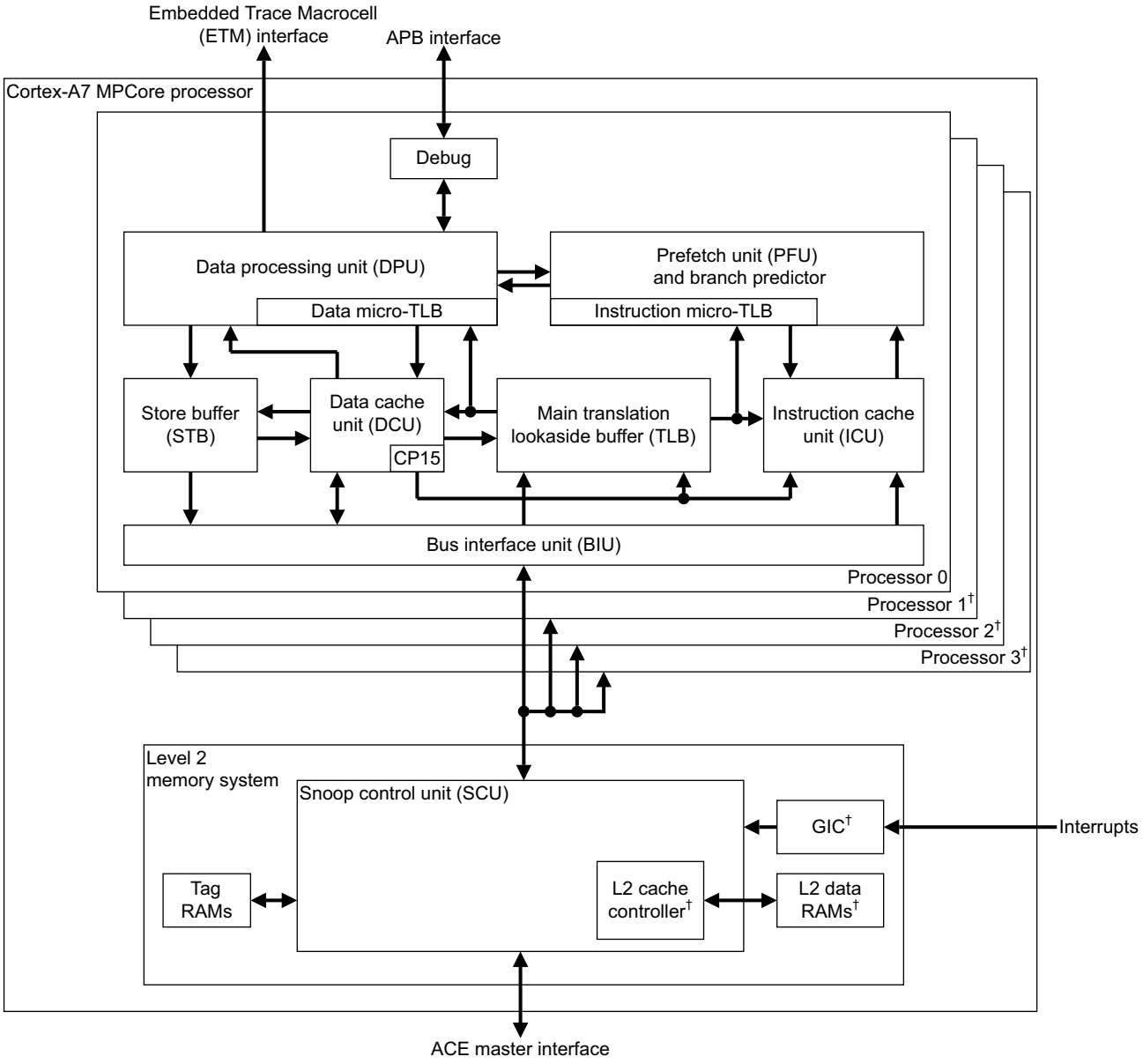


Figure 2-1 Cortex-A7 MPCore processor top-level diagram

2.1.1 Processor components

The following sections describe the main components and their functions:

- *Data Processing Unit* on page 2-3
- *System control coprocessor* on page 2-3
- *Instruction side memory system* on page 2-3
- *Data side memory system* on page 2-4
- *L1 memory system* on page 2-6

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 *Prefetch Unit* (PFU). The DPU executes instructions that require data to be transferred to or from the memory system by interfacing to the *Data Cache Unit* (DCU), which manages all load and store operations. See Chapter 3 *Programmers Model* for more information.

System control coprocessor

The system control coprocessor, CP15, provides configuration and control of the memory system and its associated functionality.

See Chapter 4 *System Control* for more information.

Instruction side memory system

The instruction side memory system includes:

- *Instruction Cache Unit*.
- *Prefetch Unit*.

Instruction Cache Unit

The *Instruction Cache Unit* (ICU) contains the Instruction Cache controller and its associated linefill buffer. The Cortex-A7 MPCore ICache is two-way set associative and uses *Virtually Indexed Physically Tagged* (VIPT) cache-lines holding up to 8 ARM or Thumb 32-bit instructions or up to 16 Thumb 16-bit instructions.

Prefetch Unit

The *Prefetch Unit* (PFU) obtains instructions from the instruction cache or from external memory and predicts the outcome of branches in the instruction stream, then passes the instructions to the DPU for processing. In any given cycle, up to a maximum of four instructions can be fetched and two can be passed to the DPU.

Branch Target Instruction Cache

The PFU also contains a four-entry deep *Branch Target Instruction Cache* (BTIC). Each entry stores up to two instruction cache fetches and enables the branch shadow of predicted taken B and BL instructions to be eliminated. The BTIC implementation is architecturally transparent, so it does not have to be flushed on a context switch.

Branch Target Address Cache

The PFU also contains a eight-entry deep *Branch Target Address Cache* (BTAC) used to predict the target address of certain indirect branches. The BTAC implementation is architecturally transparent, so it does not have to be flushed on a context switch.

Branch prediction The branch predictor is a global type that uses history registers and a 256-entry pattern history table.

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

Data side memory system

This section describes the following:

- *Data Cache Unit*.
- *Store Buffer* on page 2-6.
- *Bus Interface Unit and SCU interface* on page 2-6.

Data Cache Unit

The *Data Cache Unit* (DCU) consists of the following sub-blocks:

- The *Level 1* (L1) data cache controller, which generates the control signals for the associated embedded tag, data, and dirty memory (RAMs) and arbitrates between the different sources requesting access to the memory resources. The data cache is 4-way set associative and uses a *Physically Indexed Physically Tagged* (PIPT) scheme for lookup which enables unambiguous address management in the system.
- The load/store pipeline that interfaces with the DPU and main TLB.
- The system coprocessor controller that performs cache maintenance operations directly on the data cache and indirectly on the instruction cache through an interface with the ICU.
- An interface to receive coherency requests from the *Snoop Control Unit* (SCU).

The DCU contains a combined local and global exclusive monitor. This monitor can be set to the exclusive state only by a `LDREX` instruction executing on the local processor, and can be cleared to the open access state by:

- A `STREX` instruction on the local processor or a store to the same shared cache line on another processor.
- The cache line being evicted for other reasons.
- A `CLREX` instruction.

The Cortex-A7 MPCore processor uses the MOESI protocol, with ACE modified equivalents of MOESI states, to maintain data coherency between multiple processors. MOESI describes the state that a shareable line in a L1 data cache can be in:

M Modified/UniqueDirty (UD). The line is only in this cache and is dirty.

O Owned/SharedDirty (SD). The line is possibly in more than one cache and is dirty.

E Exclusive/UniqueClean (UC). The line is only in this cache and is clean.

S Shared/SharedClean (SC). The line is possibly in more than one cache and is clean.

I Invalid/Invalid (I). The line is not in this cache.

The DCU stores the MOESI state of the cache line in the tag and dirty RAMs.

Read allocate mode

The L1 data cache only supports a Write-Back policy. It normally allocates a cache line on either a read miss or a write miss. However, there are some situations where allocating on writes is undesirable, such as executing the C standard library `memset()` function to clear a large block of memory to a known value. Writing large blocks of data like this can pollute the cache with unnecessary data. It can also waste power and performance if a linefill must be performed only to discard the linefill data because the entire line was subsequently written by the `memset()`.

To prevent this, the *Bus Interface Unit* (BIU) includes logic to detect when a full cache line has been written by the processor before the linefill has completed. If this situation is detected on three consecutive linefills, it switches into read allocate mode. When in read allocate mode, loads behave as normal and can still cause linefills, and writes still lookup in the cache but, if they miss, they write out to L2 rather than starting a linefill.

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

A secondary read allocate mode applies when the L2 cache is integrated. After 127 consecutive cache line sized writes to L2 are detected, L2 read allocate mode is entered. When in L2 read allocate mode, loads behave as normal and can still cause linefills, and writes still lookup in the cache but, if they miss, they write out to L3 rather than starting a linefill. L2 read allocate mode continues until there is a cacheable write burst that is not a full cache line, or there is a load to the same line as is currently being written to L3.

Note

The number of consecutive cache line sized writes to enter a secondary read allocate mode was 7 prior to product revision r0p3.

Data cache invalidate on reset

The ARMv7 *Virtual Memory System Architecture* (VMSA) does not support a CP15 operation to invalidate the entire data cache. If this function is required in software, it must be constructed by iterating over the cache geometry and executing a series of individual CP15 invalidate by set/way instructions.

In normal usage the only time the entire data cache must be invalidated is on reset. The processor provides this functionality by default. If it is not required on reset the invalidate operation can be disabled by asserting and holding the appropriate external **L1RSTDISABLE** signal for a processor when the corresponding reset signal is deasserted.

In parallel to the data cache invalidate, the DCU also sends an invalidate-all request to the ICU and the TLB, unless **L1RSTDISABLE** is asserted.

Store Buffer

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

The STB can merge:

- Several store transactions into a single transaction if they are to the same 64-bit aligned address. The STB is also used to queue up CP15 maintenance operations before they are broadcast to other processors in the multiprocessor device.
- Multiple writes into an AXI write burst.

Bus Interface Unit and SCU interface

The *Bus Interface Unit* (BIU) contains the SCU interface and buffers to decouple the interface from the cache and STB. The BIU interface and the SCU always operate at the processor frequency.

A write buffer is available to hold:

- Data from cache evictions or non-cacheable write bursts before they are written out to the SCU.
- The addresses of outstanding ACE write transactions to permit hazard checking against other outstanding requests in the system.

L1 memory system

The processor L1 memory system includes the following features:

- Separate instruction and data caches.
- Export of memory attributes for system caches.

The caches have the following features:

- Support for instruction and data cache sizes between 8KB and 64KB.
- Pseudo-random cache replacement policy.
- Ability to disable each cache independently.
- Streaming of sequential data from `LDM` and `LDRD` operations, and sequential instruction fetches.
- Critical word first linefill on a cache miss.
- All the cache RAM blocks and associated tag and valid RAM blocks if implemented using standard RAM compilers.

See Chapter 6 *L1 Memory System* for more information.

L2 memory system

The L2 memory system contains:

- The SCU that connects between one to four processors to the external memory system through the ACE master interface. The SCU maintains data cache coherency between the processors and arbitrates L2 requests from the processors.

  The SCU includes support for data security using the implemented Security Extensions.

Note

The SCU does not support hardware management of coherency of the instruction caches.

- An optional L2 cache that:
  - Has configurable cache RAM sizes of 128KB, 256KB, 512KB, or 1MB
  - Is 8-way set associative.
  - Supports 64-byte cache lines.
- One ACE master interface. All transactions are routed through the interface.

See Chapter 7 *L2 Memory System* for more information.

Optional Generic Interrupt Controller

The optional integrated GIC manages interrupt sources and behavior, and can route interrupts to individual processors. It permits software to mask, enable and disable interrupts from individual sources, to prioritize, in hardware, individual sources and to generate software interrupts. It also provides support for the Security and Virtualization Extensions. The GIC accepts interrupts asserted at the system level and can signal them to each processor it is connected to. This can result in an IRQ or FIQ exception being taken.

See Chapter 8 *Generic Interrupt Controller* for more information.

Media Processing Engine

The optional *Media Processing Engine* (MPE) implements ARM NEON technology, a media and signal processing architecture that adds instructions targeted at audio, video, 3-D graphics, image, and speech processing. Advanced SIMD instructions are available in both ARM and Thumb states.

See the *Cortex-A7 MPCore NEON Media Processing Engine Technical Reference Manual* for more information.

Floating-Point Unit

The optional *Floating-Point Unit* (FPU) implements the ARMv7 VFPv4-D16 architecture and includes the VFP register file and status registers. It performs floating-point operations on the data held in the VFP register file.

See the *Cortex-A7 MPCore Floating-Point Unit Technical Reference Manual* for more information.

Debug

The Cortex-A7 MPCore processor has a CoreSight compliant *Advanced Peripheral Bus version 3* (APBv3) debug interface. This permits system access to debug resources, for example, the setting of watchpoints and breakpoints. The processor provides extensive support for real-time debug and performance profiling.

See Chapter 10 *Debug* for more information.

Performance monitoring

The Cortex-A7 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 11 *Performance Monitoring Unit* for more information.

2.2 Interfaces

The Cortex-A7 MPCore processor has the following external interfaces:

- *ACE*.
- *APB*.
- *ETM*.
- *DFT*.
- *MBIST controller*.

2.2.1 ACE

The processor implements the AMBA 4 *AXI Coherency Extension*s (ACE) interface.

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

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

See *ACE master interface signals* on page A-8 for more information on the signals.

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

2.2.2 APB

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

2.2.3 ETM

The *Embedded Trace Macrocell* (ETM) interface enables you to connect an external ETM unit to the Cortex-A7 MPCore processor for real-time code and data tracing of the processor in an embedded system.

Each processor in the Cortex-A7 MPCore processor exports its own ETM interface. Each interface runs at the full speed of the Cortex-A7 MPCore processor. The ETM interface connects directly to the external CoreSight ETM-A7 without any additional glue logic.

See *ETM interface signals* on page A-16 for more information on the signals.

See the *CoreSight ETM-A7 Technical Reference Manual* for more information on the ETM interface.

2.2.4 DFT

The processor implements a *Design For Test* (DFT) interface that enables an industry standard *Automatic Test Pattern Generation* (ATPG) tool to test logic outside of the embedded memories. See *DFT interface* on page A-17 for information on these test signals.

2.2.5 MBIST controller

The *Memory Built In Self Test* (MBIST) controller interface provides support for manufacturing test of the memories embedded in the Cortex-A7 MPCore processor. MBIST is the industry standard method of testing embedded memories. MBIST works by performing sequences of reads and writes to the memory based on test algorithms. See *MBIST interface* on page A-17 for information on this interface.

2.3 Clocking and resets

The following sections describe clocking and resets:

- *Clocking*.
- *Resets* on page 2-10.

2.3.1 Clocking

The Cortex-A7 MPCore processor has a single clock input, **CLKIN**. All processors in the Cortex-A7 MPCore processor and the SCU are clocked with a distributed version of **CLKIN**. The Cortex-A7 MPCore processor synchronizes the input signals:

- **nCOREPORESET[3:0]**.
- **nCORERESET[3:0]**.
- **nDBGRESET[3:0]**.
- **nL2RESET**.
- **nMBISTRESET**.

All other external signals must be synchronous with reference to **CLKIN**.

ACE master interface clocking

The SCU interface supports integer ratios of the **CLKIN** frequency , for example 1:1, 2:1, 3:1. These ratios are configured through external clock enable signals. In all cases AXI transfers remain synchronous. The ACE master interface includes the **ACLKENM** clock enable signal.

ACLKENM asserts one **CLKIN** cycle prior to the rising edge of the external ACE clock signal, **ACLKM**. Software can change the **CLKIN** to **ACLKM** frequency ratio dynamically using **ACLKENM**.

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

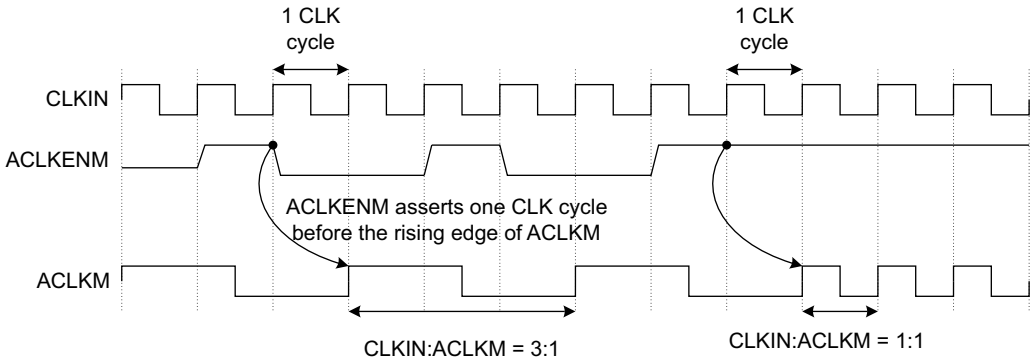


Figure 2-2 ACLKENM with CLKIN:ACLKM ratio changing from 3:1 to 1:1

Note

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

Debug interface clocking

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

2.3.2 Resets

The Cortex-A7 MPCore processor has multiple reset domains with the following reset input signals:

nCOREPORESET[3:0]

These power-on reset signals initialize all the processor logic, including CPU Debug, and breakpoint and watchpoint logic in the processor power domains. They do not reset debug logic in the debug power domain.

nCORERESET[3:0]

These are the primary reset signals which initialize the processor logic in the processor power domains, not including the debug, breakpoint and watchpoint logic.

nDBGRESET[3:0] At the Cortex-A7 level, these signals reset only the debug, and breakpoint and watchpoint logic in the processor power domain. At the Cortex-A7 integration layer level, these signals also reset the debug logic for each processor, which is in the debug power domain.

These reset signals are 4-bit signals, where each bit represents one processor in the multiprocessor device.

The following reset input signals are single-bit fanouts to all the processors in the multiprocessor device:

nMBISTRESET This signal resets the device for entry into MBIST mode.

nL2RESET This signal resets the L2 memory system and the logic in the SCU.

All of these resets:

- Can be asynchronously asserted and de-asserted.
- Are active-LOW signals that reset logic in the appropriate domain of the Cortex-A7 MPCore processor.

ARM recommends that you assert the reset signals for at least four **CLKIN** cycles to ensure correct reset behavior.

In Table 2-1 on page 2-11, [3:0] specifies the processor configuration and [n] designates the processor that is reset.

Table 2-1 Valid reset combinations

Reset combination	Signals	Value	Description
All processor power-on reset	**nCOREPORESET [3:0]** **nCORERESET [3:0]** **nDBGRESET [3:0]** **nL2RESET**	all = 0 all = 0[a] all = 0[a] 0	All logic is held in reset.
Individual processor power-on reset with Debug reset	**nCOREPORESET [3:0]** **nCORERESET [3:0]** **nDBGRESET [3:0]** **nL2RESET**	[n] = 0 [n] = 0[a] [n] = 0[a] 1	Individual processor and Debug are held in reset.
All processor power-on and L2 reset with Debug active	**nCOREPORESET [3:0]** **nCORERESET [3:0]** **nDBGRESET [3:0]** **nL2RESET**	all = 0 all = 0 all = 1 0	All processors and L2 are held in reset, so they can be powered up. This enables external debug over power down for all processors.
Individual processor power-on reset with Debug active	**nCOREPORESET [3:0]** **nCORERESET [3:0]** **nDBGRESET [3:0]** **nL2RESET**	[n] = 0 [n] = 0 [n] = 1 1	Individual processor is held in reset, so that the processor can be powered up. This enables external debug over power down for the processor that is held in reset.
All processors software reset	**nCOREPORESET [3:0]** **nCORERESET [3:0]** **nDBGRESET [3:0]** **nL2RESET**	all = 1 all = 0 all = 1 1	All logic excluding Debug and L2 memory system is held in reset. All breakpoints and watchpoints are retained.
All processors software reset and L2 reset	**nCOREPORESET [3:0]** **nCORERESET [3:0]** **nDBGRESET [3:0]** **nL2RESET**	all = 1 all = 0 all = 1 0	All logic excluding Debug is held in reset. All breakpoints and watchpoints are retained.
Individual processor software reset	**nCOREPORESET [3:0]** **nCORERESET [3:0]** **nDBGRESET [3:0]** **nL2RESET**	[n] = 1 [n] = 0 [n] = 1 1	Individual processor logic excluding Debug and ETM is held in reset. Breakpoints and watchpoints for that processor are retained.
All processors debug reset	**nCOREPORESET [3:0]** **nCORERESET [3:0]** **nDBGRESET [3:0]** **nL2RESET**	all = 1 all = 1 all = 0 1	Debug is held in reset.
Individual processor Debug reset	**nCOREPORESET [3:0]** **nCORERESET [3:0]** **nDBGRESET [3:0]** **nL2RESET**	[n] = 1 [n] = 1 [n] = 0 1	Individual processor Debug is held in reset.
Run mode	**nCOREPORESET [3:0]** **nCORERESET [3:0]** **nDBGRESET [3:0]** **nL2RESET**	1 1 1 1	No logic is held in reset.

a. For power-on reset or processor reset, **nCOREPORESET** must be asserted. The remaining processor resets, **nCORERESET** and **nDBGRESET** can be asserted, but is not required.

2.4 Power management

The Cortex-A7 MPCore processor provides mechanisms and support to control both dynamic and static power dissipation. The individual processors in the Cortex-A7 MPCore processor support four main levels of power management. This section describes:

- *Power domains*.
- *Power modes* on page 2-13.
- *Event communication using WFE or SEV* on page 2-20.
- *Communication to the Power Management Controller* on page 2-20.

2.4.1 Power domains

The power domains that the Cortex-A7 MPCore processor supports are:

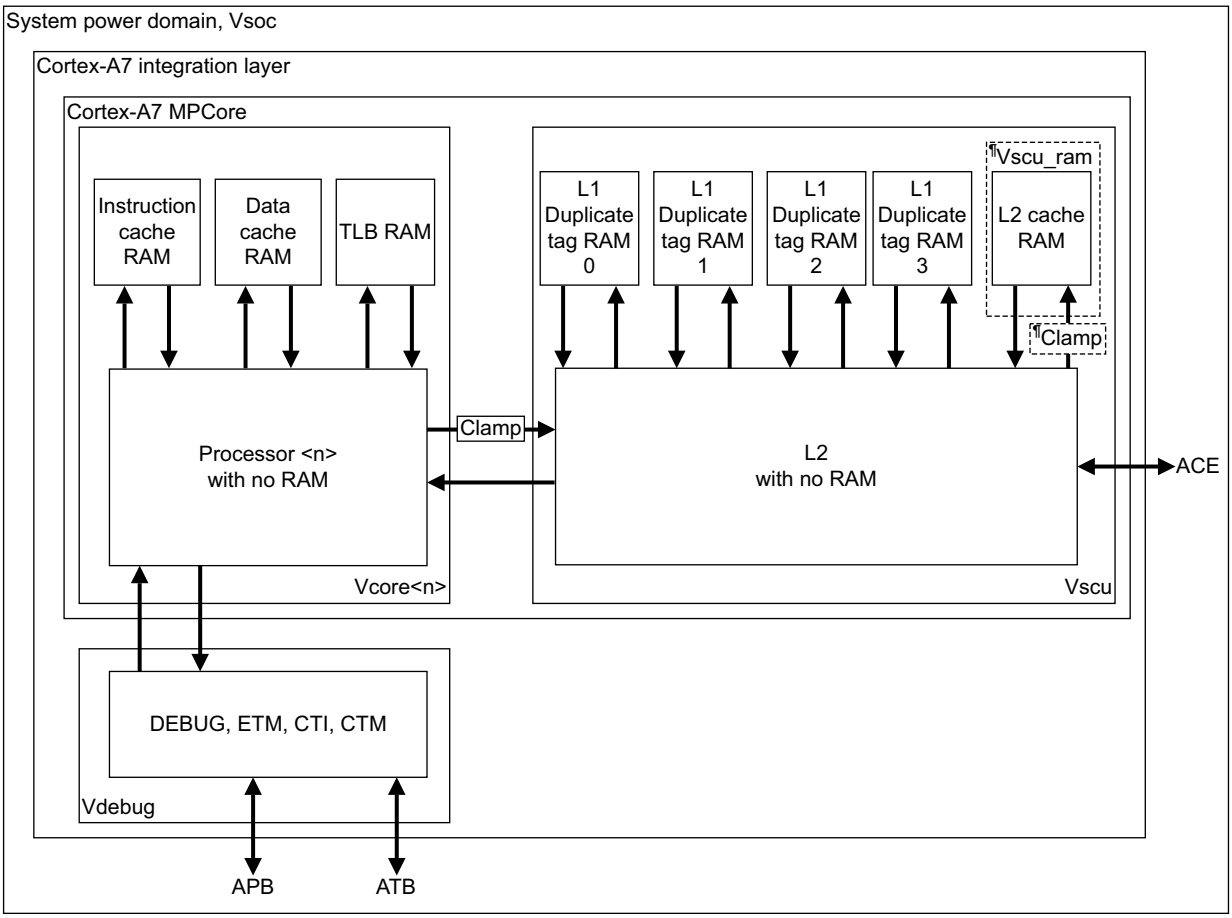
- Vcore<n>, that represents the processor <n>, which includes the optional Advanced SIMD and VFP Extensions and the L1 TLB and cache RAM.
- Vscu that represents the Cortex-A7 L2 with the duplicate tag RAMs, which includes the SCU, the optional L2 cache controller, and the optional integrated GIC.
- Vscu_ram, which includes the L2 cache RAM.

The Cortex-A7 integration layer supports the following power domain:

- Vdebug, which includes support for v7.1 Debug architecture, CoreSight debug and trace components.

The separate SCU power domains can remain active even when all the processors are powered down. This enables the Cortex-A7 MPCore processor to accept snoops from an external device to control the L2 cache.

Figure 2-3 on page 2-13 shows an example of the domains embedded in a *System-on-Chip* (SoC) power domain.



¶For optional Dormant mode support only.

Figure 2-3 Power domains

Note

Figure 2-3 shows the Vdebug power domain for example only. It is a domain in the Cortex-A7 integration layer and is not described in this manual.

Each Vcore<n> domain uses a single clock that is architecturally gated at the top level of the processor to minimize dynamic power consumption without removing power completely from the processor.

At the SoC integration level, the processor logic in Vcore<n> and Vscu can be isolated and powered down completely through the instantiation of clamps on all of the external interfaces inside the Vsoc domain. .

2.4.2 Power modes

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

Table 2-2 .shows the supported power configurations for the different possible modes of operation.

Table 2-2 Supported power modes

Mode	Vsoc	Vscu	Vscu_ram	Vcore<n[a]>
Run mode	Powered up	Powered up	Powered up	Powered up
Processor n[a] in shutdown mode	Powered up	Powered up	Powered up	Powered down
Cortex-A7 MPCore processor in shutdown mode	Powered up	Powered down	Powered down	Powered down
Cortex-A7 MPCore processor in Dormant mode	Powered up	Powered down	Powered up	Powered down

a. Where n represents 0-3.

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

The dynamic power management and power-up and power-down sequences in the following sections are the only power sequences that ARM recommends. Any deviation from these sequences can lead to UNPREDICTABLE results.

The supported power modes are:

- *Run mode*.
- *Standby mode*.
- *Individual processor shutdown mode* on page 2-17.
- *Multiprocessor device shutdown mode* on page 2-18.
- *Dormant mode* on page 2-19.

Run mode

This is the normal mode of operation where all of the processor functionality is available. The Cortex-A7 MPCore processor uses gated clocks and gates to disable inputs to unused functional blocks. Only the logic in use to perform an operation consumes any dynamic power.

Standby mode

The following sections describes the methods of entering standby mode:

- *Processor Wait for Interrupt*.
- *Processor Wait for Event* on page 2-15.
- *L2 Wait for Interrupt* on page 2-16.

Processor Wait for Interrupt

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

A processor enters into WFI mode by executing the `WFI` instruction.

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

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

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

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

- An L2 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 mode occurs when the processor detects a reset or one of the WFI wake up events as described in the *ARM Architecture Reference Manual*.

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

Figure 2-4 shows the upper bound for the **STANDBYWFI** deassertion timing after the assertion of **nIRQ** or **nFIQ** inputs.

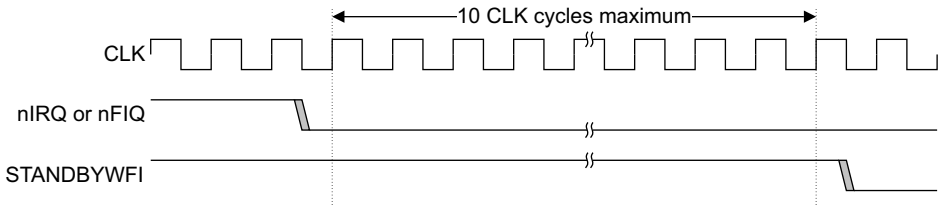


Figure 2-4 STANDBYWFI deassertion timing

Processor Wait for Event

Wait for Event is a feature of the ARMv7-A architecture that uses a locking mechanism based on events to put the processor in a low power state by disabling most of the clocks in the processor while keeping the processor powered up. This reduces the power drawn to the static leakage current, leaving a small clock power overhead to enable the processor to wake up from WFE mode.

A processor enters into WFE mode by executing the WFE instruction. When executing the WFE instruction, the processor waits for all instructions in the processor to complete before entering the idle or low power state.

While the processor is in WFE mode, the clocks in the processor are temporarily enabled without causing the processor to exit WFE mode, when any of the following events are detected:

- An L2 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 WFE mode occurs when the processor detects a reset, the assertion of the **EVENTI** input signal, or one of the WFI wake up events as described in the *ARM Architecture Reference Manual*.

On entry into WFE mode, **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 because of an L2 snoop request, cache, and TLB maintenance operation or an APB access.

The upper bound for the **STANDBYWFE** deassertion timing after the assertion of **nIRQ** or **nFIQ** inputs is identical to **STANDBYWFI** as shown in Figure 2-4 on page 2-15.

L2 Wait for Interrupt

When all the processors are in WFI mode, the shared L2 memory system logic that is common to all the processors can also enter a WFI mode. In L2 WFI mode, all internal clocks in the processor are disabled.

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

- All processors are in WFI mode and therefore, all the processors **STANDBYWFI** outputs are asserted. Assertion of all the processors **STANDBYWFI** outputs guarantee that all the processors are in idle and low power state. All clocks in the processor, with the exception of a small amount of clock wake up logic, are disabled.
- The SoC asserts the input pin **ACINACTM** to idle the AXI master interface. This prevents the L2 memory system from accepting any new requests from the AXI master interface.
- When the L2 memory system completed the outstanding transactions for AXI interfaces, it can then enter the low power state, L2 WFI mode. On entry into L2 WFI mode, **STANDBYWFIL2** is asserted. Assertion of **STANDBYWFIL2** guarantees that the L2 memory system is in idle and does not accept any new transactions.

Exit from L2 WFI mode occurs on one of the following WFI wake up events:

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

When the processor exits from WFI mode, **STANDBYWFI** for that processor is deasserted. When the L2 memory system logic exits from WFI mode, **STANDBYWFIL2** is deasserted. The SoC must continue to assert **ACINACTM** until **STANDBWFIL2** has deasserted.

Figure 2-5 on page 2-17 shows the L2 WFI timing for a 4-processor configuration.

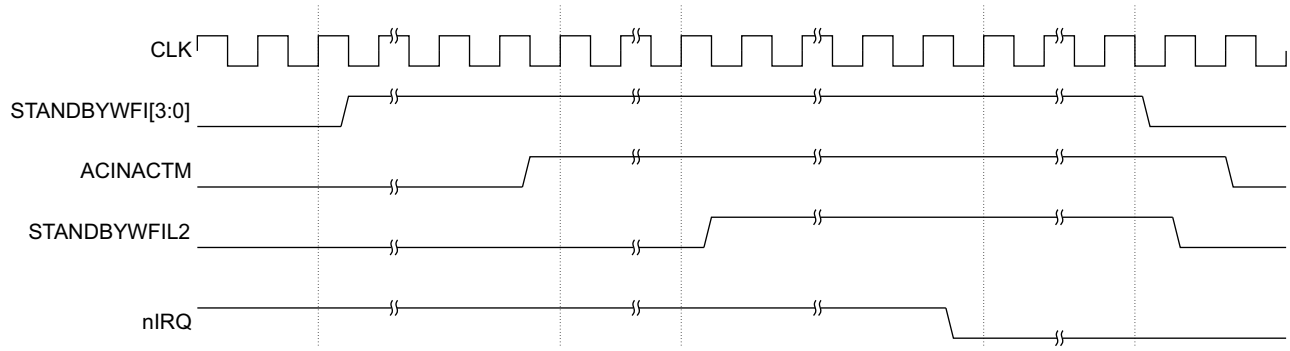


Figure 2-5 L2 Wait For Interrupt timing

Individual processor shutdown mode

This is the mode where the Vcore 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 indicate that the processor is idle.

To power down the processor, apply the following sequence:

1. Clear the SCTLR.C bit, or HSCTLR.C bit if in Hyp mode, to prevent further data cache allocation.
2. Clean and invalidate all data from the L1 data cache. The L2 duplicate snoop tag RAM for this processor is now empty. This prevents any new data cache snoops or data cache maintenance operations from other processors in the multiprocessor device being issued to this processor.
3. Execute a `CLREX` instruction.
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.
5. Execute an `ISB` instruction to ensure that all of the CP15 register changes from the previous steps have been committed.
6. Execute a `DSB` instruction to ensure that all cache, TLB and branch predictor maintenance operations issued by any processor in the multiprocessor device before the SMP bit was cleared have completed.
7. Execute a WFI instruction and wait until the **STANDBYWFI** output is asserted to indicate that the processor is in idle and low power state.
8. Deassert **DBGPWRDUP** LOW. This prevents any external debug access to the processor.

   Note

   The **DBGPWRDUP** signal is an integration layer signal. See the *Cortex-A7 MPCore Integration Manual* for more information.

9. Activate the processor output clamps.

10. Remove power from the Vcore power domain.

To power up the processor, apply the following sequence:

1. Assert **nCOREPORESET** LOW and hold **L1RSTDISABLE** LOW. Ensure **DBGPWRDUP** is held LOW to prevent any external debug access to the processor.
2. Apply power to the Vcore power domain. Keep the state of the signals **nCOREPORESET**, **L1RSTDISABLE** and **DBGPWRDUP** LOW.
3. When the power domain has stabilized and reset has been asserted for four or more cycles, release the processor output clamps.
4. De-assert resets.
5. Assert **DBGPWRDUP** HIGH to allow external debug access to the processor.
6. If required use software to restore the state of the processor prior to power-down.
7. Assert ACTLR.SMP bit HIGH for SMP mode. Continue a normal power-on reset sequence.

Note

The **DBGPWRDUP** signal is an integration layer signal. See the Cortex-A7 MPCore Integration Manual for more information.

Multiprocessor device shutdown mode

This is the mode where the Vscu, Vscu_ram, and Vcore 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 2-17.
2. Follow steps 1. to 3. in *Individual processor shutdown mode* on page 2-17.
3. Clean and invalidate all data from L2 data cache.
4. Follow steps 4 to 9. in *Individual processor shutdown mode* on page 2-17.
5. Assert **ACINACTM** and wait until the **STANDBYWFIL2** output is asserted to indicate that the L2 memory system is idle.
6. Activate the multiprocessor device output clamps.
7. Remove power from the Vscu and Vscu_ram power domains.

Note

For device power-down, all operations on a 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 and hold **L1RSTDISABLE** LOW.
2. For the lead processor in the multiprocessor device, assert **nL2RESET** LOW and hold **L2RSTDISABLE** LOW.

3. Apply power to the Vscu, Vscu_ram, and Vcore domains while keeping the signals described in steps 1. and 2. LOW.
4. When the power domain has stabilized and reset has been asserted for four or more cycles, release the processor output clamps.
5. Continue a normal power-on reset sequence.
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 power-down 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 power-on reset sequence. The SoC must assert the reset signals until power is restored. After power is restored, the processor exits the power-on 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. Clean and invalidate all data from the L1 data cache. The L2 duplicate snoop tag RAM for this processor is now empty. This prevents any new data cache snoops or data cache maintenance operations from other processors in the multiprocessor device being issued to this processor.
3. Execute a `CLREX` instruction.
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 an ISB instruction to ensure that all of the CP15 register changes from the previous steps have been committed.
7. 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.
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. Assert **ACINACTM** and wait until the **STANDBYWFIL2** output is asserted to indicate that the L2 memory system is idle.
11. When all processors **STANDBYWFI** and **STANDBYWFIL2** are asserted, the multiprocessor device is ready to enter Dormant mode.
12. Activate the L2 cache RAM input clamps.
13. Remove power from the Vcore and Vscu power domains.

To exit Dormant mode, apply the following sequence:

1. Apply a normal power-on 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 and reset has been asserted for four or more clock cycles, release the L2 cache RAM input clamps.
3. Continue a normal power-on reset sequence with **L2RSTDISABLE** held HIGH.
4. The architectural state must be restored, if required.

2.4.3 Event communication using WFE or SEV

An external agent can use the the **EVENTI** pin to participate in a WFE or SEV event communication of the Cortex-A7 MPCore processor. When this pin is asserted, it sends an event message to all the Cortex-A7 MPCore 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 to the processors that it has released a semaphore and that the processors can leave the WFE standby power saving mode. The **EVENTI** input pin must remain HIGH at least one **CLK** clock cycle to be visible by the processors.

The external agent can determine that at least one of the Cortex-A7 MPCore processors in the cluster has executed an SEV instruction by checking 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 Cortex-A7 MPCore processor in the cluster executes an SEV instruction.

2.4.4 Communication to the Power Management Controller

Communication between the Cortex-A7 MPCore processor and the system power management controller can be performed using the **STANDBYWFI[3:0]** and **STANDBYWFIL2** signals.

The **STANDBYWFI[n]** signal indicates when an individual processor is in idle and low power state. The power management controller can remove power from an individual processor when **STANDBYWFI[n]** is asserted. See *Individual processor shutdown mode* on page 2-17 for more information.

The **STANDBYWFIL2** signal indicates when all individual processors and the L2 memory system are in idle and low power state. A power management controller can remove power from the Cortex-A7 MPCore processor when **STANDBYWFIL2** is asserted. See *Multiprocessor device shutdown mode* on page 2-18 for more information.

Note

The Cortex-A7 MPCore processor includes a minimal L2 memory system in configurations without an L2 cache. Therefore, the power management controller must always wait for assertion of **STANDBYWFIL2** before removing power from the Cortex-A7 MPCore processor.

Figure 2-6 shows how **STANDBYWFI[3:0]** and **STANDBYWFIL2** correspond to individual processors and the Cortex-A7 MPCore processor.

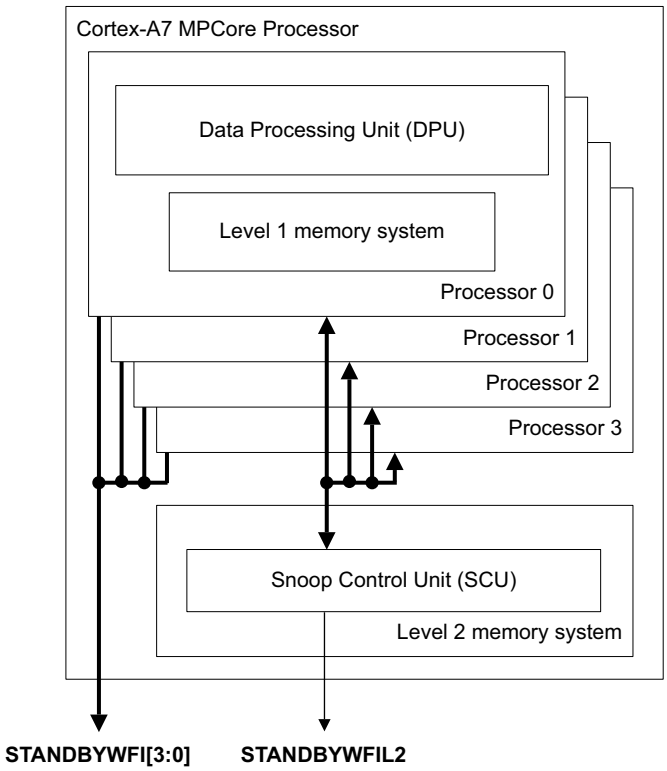


Figure 2-6 STANDBYWFI[3:0] and STANDBYWFIL2 signals

Chapter 3
Programmers Model

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

3.1 About the programmers model

The Cortex-A7 MPCore processor implements the ARMv7-A architecture. This includes:

- The 32-bit ARM instruction set.
- The Thumb instruction set that has both 16-bit and 32-bit instructions.
- The ThumbEE instruction set.
- A trivial implementation of the Jazelle 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* for more information.

3.2 Execution environment support

For the Cortex-A7 MPCore processor, ARM deprecates any use of the ThumbEE instruction set and provides a trivial implementation of the Jazelle Extension.

This means that:

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

See the *ARM Architecture Reference Manual*.

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

Table 3-1 Jazelle register instruction summary

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

a. Accessible from all privilege levels.
b. Can cause a trap to Hyp mode depending on the TID0 bit set in HCR.
c. Write-only accessible in unprivileged level, read-write accessible at PL1 or higher.
d. Can cause a trap to Hyp mode depending on the TJDBX bit set in HSTR.
e. Accessible from PL1 or higher.

Note

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

3.3 Advanced SIMD and VFP Extensions

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* for more information.

See the *Cortex-A7 MPCore Floating-Point Unit Technical Reference Manual* and *Cortex-A7 MPCore NEON Media Processing Engine Technical Reference Manual* for implementation-specific information.

3.4 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* for more information.

3.4.1 System boot sequence

Caution

Security Extensions computing enable a 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 4-63. This means that code that does not attempt to use the Security Extensions always runs in the Secure state. If the software uses both Secure and Non-secure states, the less trusted software, such as a complex operating system and application code running under that operating system, executes in Non-secure state, and the most trusted software executes in the Secure state.

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

1. Exit from reset in Secure state.
2. Configure the security state of memory and peripherals. Some memory and peripherals are accessible only to the software running in Secure state.
3. Initialize the secure operating system. The required operations depend on the operating system, and include initialization of caches, MMU, exception vectors, and stacks.
4. Initialize Secure Monitor software to handle exceptions that switch execution between the Secure and Non-secure operating systems.
5. Optionally lock aspects of the secure state environment 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.

3.4.2 Security Extensions write access disable

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

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

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

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

3.5 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 10 *Debug*.
- The PMUv2 Performance Monitors, see Chapter 11 *Performance Monitoring Unit*.

See the *ARM Architecture Reference Manual* for more information.

3.6 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* for more information.

3.7 Multiprocessing Extensions

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

See the *ARM Architecture Reference Manual* for more information.

3.8 Modes of operation

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

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

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

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

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

Table 3-2 CPSR J and T bit encoding

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

Note

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

3.9 Memory model

The Cortex-A7 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* for more information about big-endian and little-endian memory systems.

Note

Instructions are always treated as little-endian.

Chapter 4
System Control

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

4.1 About system control

The 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* 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 4-4.
- *c1 registers* on page 4-5.
- *c2 registers* on page 4-6.
- *c3 registers* on page 4-6.
- *c4 registers* on page 4-6.
- *c5 registers* on page 4-7.
- *c6 registers* on page 4-7.
- *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-A7 MPCore processor supports the *Virtualization Extensions* (VE) and the *Large Physical Address Extension* (LPAE). See *Virtualization Extensions architecture* on page 3-7 and *Large Physical Address Extension architecture* on page 3-8 and Chapter 9 *Generic Timer* for more information. The VE, LPAE, and Generic Timer contain a number of 64-bit registers. The following subsection describes these registers and provides cross references to individual register descriptions:

- *64-bit registers* on page 4-14.

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 4-17.
- *Cache maintenance operations* on page 4-17.
- *TLB maintenance operations* on page 4-18.
- *Address translation operations* on page 4-19.
- *Miscellaneous operations* on page 4-20.
- *Performance monitor registers* on page 4-21.
- *Security Extensions registers* on page 4-22.
- *Virtualization Extensions registers* on page 4-22.
- *TLB maintenance operations* on page 4-24.
- *Generic Timer registers* on page 4-24.
- *Implementation defined registers* on page 4-24.

Table 4-1 describes the column headings in the CP15 register summary tables use 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	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	0x410FC073	*Main ID Register* on page 4-26
			1	CTR	0x84448003	*Cache Type Register* on page 4-27
			2	TCMTR	0x00000000	*TCM Type Register* on page 4-28
			3	TLBTR	0x00000000	*TLB Type Register* on page 4-28
			5	MPIDR	-[a]	*Multiprocessor Affinity Register* on page 4-28
			6	REVIDR	0x00000000	*Revision ID Register* on page 4-29
			4, 7	MIDR	0x410FC074	Aliases of Main ID Register, *Main ID Register* on page 4-26
		c1	0	ID_PFR0	0x00001131	*Processor Feature Register 0* on page 4-30
			1	ID_PFR1	0x00011011	*Processor Feature Register 1* on page 4-31
			2	ID_DFR0	0x02010555	*Debug Feature Register 0* on page 4-32
			3	ID_AFR0	0x00000000	*Auxiliary Feature Register 0* on page 4-33
			4	ID_MMFR0	0x10101105	*Memory Model Feature Register 0* on page 4-33
			5	ID_MMFR1	0x40000000	*Memory Model Feature Register 1* on page 4-34
			6	ID_MMFR2	0x01240000	*Memory Model Feature Register 2* on page 4-36
			7	ID_MMFR3	0x02102211	*Memory Model Feature Register 3* on page 4-37
		c2	0	ID_ISAR0	0x01101110	*Instruction Set Attribute Register 0* on page 4-39
			1	ID_ISAR1	0x13112111	*Instruction Set Attribute Register 1* on page 4-40
			2	ID_ISAR2	0x21232041	*Instruction Set Attribute Register 2* on page 4-42
			3	ID_ISAR3	0x11112131	*Instruction Set Attribute Register 3* on page 4-43

Table 4-2 c0 register summary (continued)

CRn	Op1	CRm	Op2	Name	Reset	Description
c0	0	c2	4	ID_ISAR4	0x10011142	*Instruction Set Attribute Register 4* on page 4-45
			5	ID_ISAR5	0x00000000	*Instruction Set Attribute Register 5* on page 4-46
	1	c0	0	CCSIDR	UNK	*Cache Size ID Register* on page 4-46
			1	CLIDR	-[b]	*Cache Level ID Register* on page 4-48
			7	AIDR	0x00000000	*Auxiliary ID Register* on page 4-49
	2	c0	0	CSSELR	UNK	*Cache Size Selection Register* on page 4-49
	4	c0	0	VPIDR	-[c]	*Virtualization Processor ID Register* on page 4-50
			5	VMPIDR	-[d]	*Virtualization Multiprocessor ID Register* on page 4-51

a. The reset value depends on the primary input CLUSTER ID and the CPU ID value in the Multiprocessor Affinity Register.
b. The reset value depends on whether L2 cache is implemented.
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[a]	*System Control Register* on page 4-51
			1	ACTLR	0x00000000	*Auxiliary Control Register* on page 4-59
			2	CPACR	0x00000000[b]	*Coprocessor Access Control Register* on page 4-61
		c1	0	SCR	0x00000000	*Secure Configuration Register* on page 4-63
			1	SDER	UNK	Secure Debug Enable Register, see the *ARM Architecture Reference Manual*
			2	NSACR	0x00000000[c]	*Non-Secure Access Control Register* on page 4-65
	4	c0	0	HSCTLR	UNK	*Hyp System Control Register* on page 4-66
			1	HACTLR	UNK	*Hyp Auxiliary Configuration Register* on page 4-68
		c1	0	HCR	0x00000000	Hyp Configuration Register, see the *ARM Architecture Reference Manual*
			1	HDCR	0x00000006[d]	*Hyp Debug Control Register* on page 4-68
			2	HCPTR	0x000033FF[e]	*Hyp Coprocessor Trap Register* on page 4-70
			3	HSTR	0x00000000	Hyp System Trap Register, see the *ARM Architecture Reference Manual*
			7	HACR	UNK	*Hyp Coprocessor Trap Register* on page 4-70

a. The reset value depends on primary inputs, **CFGTE**, **CFGEND**, and **VINITHI**. The value shown in Table 4-2 on page 4-4 assumes these signals are set to zero.

b. The reset value depends on the FPU and NEON configuration. If FPU and Advanced SIMD are implemented, the reset value is `0x00000000`. If FPU is implemented but Advanced SIMD is not implemented, the reset value is `0xC0000000`. If FPU and Advanced SIMD are not implemented, the reset value is Unknown.

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

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

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

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 *ARM Architecture Reference Manual*
			1	TTBR1	UNK	Translation Table Base Register 1, see the *ARM Architecture Reference Manual*
			2	TTBCR	`0x00000000`[a]	Translation Table Base Control Register, see the *ARM Architecture Reference Manual*
	4	c0	2	HTCR	UNK	*Hyp Translation Control Register* on page 4-73
		c1	2	VTCR	UNK	Virtualization Translation Control Register, see the *ARM Architecture Reference Manual*

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 *ARM Architecture Reference Manual*

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	*Data Fault Status Register* on page 4-73
			1	IFSR	UNK	*Instruction Fault Status Register* on page 4-77
		c1	0	ADFSR	UNK	*Auxiliary Data Fault Status Register* on page 4-79
			1	AIFSR	0x00000000	*Auxiliary Instruction Fault Status Register* on page 4-79
c5	4	c1	0	HADFSR	UNK	*Hyp Auxiliary Data Fault Status Syndrome Register* on page 4-79
			1	HAIFSR	UNK	*Hyp Auxiliary Instruction Fault Status Syndrome Register* on page 4-79
		c2	0	HSR	UNK	*Hyp Syndrome Register* on page 4-80

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 *ARM Architecture Reference Manual*
			2	IFAR	UNK	Instruction Fault Address Register, see the *ARM Architecture Reference Manual*
	4	c0	0	HDFAR	UNK	Hyp Data Fault Address Register, see the *ARM Architecture Reference Manual*
			2	HIFAR	UNK	Hyp Instruction Fault Address Register, see the *ARM Architecture Reference Manual*
			4	HPFAR	UNK	Hyp IPA Fault Address Register, see the *ARM Architecture Reference Manual*

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 *ARM Architecture Reference Manual*
		c1	0	ICIALLUIS	UNK	Instruction cache invalidate all to PoU[a] Inner Shareable, see the *ARM Architecture Reference Manual*
			6	BPIALLIS	UNK	Branch predictor invalidate all Inner Shareable, see the *ARM Architecture Reference Manual*
		c4	0	PAR	UNK	*Physical Address Register* on page 4-81
		c5	0	ICIALLU	UNK	Instruction cache invalidate all to PoU, see the *ARM Architecture Reference Manual*
			1	ICIMVAU	UNK	Instruction cache invalidate by MVA to PoU, see the *ARM Architecture Reference Manual*
			4	CP15ISB	UNK	Instruction Synchronization Barrier operation, see the *ARM Architecture Reference Manual*
			6	BPIALL	UNK	Branch predictor invalidate all, see the *ARM Architecture Reference Manual*
			7	BPIMVA	UNK	Branch predictor invalidate by MVA, see the *ARM Architecture Reference Manual*
		c6	1	DCIMVAC	UNK	Data cache invalidate by MVA to PoC[b], see the *ARM Architecture Reference Manual*
			2	DCISW	UNK	Data cache invalidate line by set/way, see the *ARM Architecture Reference Manual*
		c8	0	ATS1CPR	UNK	Stage 1 current state PL1 read, see the *ARM Architecture Reference Manual*
			1	ATS1CPW	UNK	Stage 1 current state PL1 write, see the *ARM Architecture Reference Manual*
			2	ATS1CUR	UNK	Stage 1 current state unprivileged (PL0) read, see the *ARM Architecture Reference Manual*
			3	ATS1CUW	UNK	Stage 1 current state unprivileged (PL0) write, see the *ARM Architecture Reference Manual*
			4	ATS12NSOPR	UNK	Stages 1 and 2 Non-secure PL1 read, see the *ARM Architecture Reference Manual*
			5	ATS12NSOPW	UNK	Stages 1 and 2 Non-secure PL1 write, see the *ARM Architecture Reference Manual*
			6	ATS12NSOUR	UNK	Stages 1 and 2 Non-secure unprivileged (PL0) read, see the *ARM Architecture Reference Manual*
			7	ATS12NSOUW	UNK	Stages 1 and 2 Non-secure unprivileged (PL0) write, see the *ARM Architecture Reference Manual*
		c10	1	DCCMVAC	UNK	Data cache clean line by MVA to PoC, see the *ARM Architecture Reference Manual*

Table 4-8 c7 register summary (continued)

CRn	Op1	CRm	Op2	Name	Reset	Description
c7	0	c10	2	DCCSW	UNK	Data cache clean line by set/way, see the *ARM Architecture Reference Manual*
			4	CP15DSB	UNK	Data Synchronization Barrier operation, see the *ARM Architecture Reference Manual*
			5	CP15DMB	UNK	Data Memory Barrier operation, see the *ARM Architecture Reference Manual*
		c11	1	DCCMVAU	UNK	Clean data cache line by MVA to PoU, see the *ARM Architecture Reference Manual*
		c13	1	NOP	UNK	No Operation, see the *ARM Architecture Reference Manual*
		c14	1	DCCIMVAC	UNK	Data cache clean and invalidate line by MVA to PoC, see the *ARM Architecture Reference Manual*
			2	DCCISW	UNK	Data cache clean and invalidate line by set/way, see the *ARM Architecture Reference Manual*
	4	c8	0	ATS1HR	UNK	Add translation stage 1 Hyp mode read, see the *ARM Architecture Reference Manual*
			1	ATS1HW	UNK	Add translation stage 1 Hyp mode write, see the *ARM Architecture Reference Manual*

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

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

4.2.9 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 *ARM Architecture Reference Manual*
			1	TLBIMVAIS	UNK	Invalidate unified TLB entry by MVA Inner Shareable, see the *ARM Architecture Reference Manual*
			2	TLBIASIDIS	UNK	Invalidate unified TLB by ASID match Inner Shareable, see the *ARM Architecture Reference Manual*
			3	TLBIMVAAIS	UNK	Invalidate unified TLB by MVA all ASID Inner Shareable, see the *ARM Architecture Reference Manual*
		c5	0	ITLBIALL	UNK	Invalidate instruction TLB, see the *ARM Architecture Reference Manual*
			1	ITLBIMVA	UNK	Invalidate instruction TLB entry by MVA, see the *ARM Architecture Reference Manual*
			2	ITLBIASID	UNK	Invalidate instruction TLB by ASID match, see the *ARM Architecture Reference Manual*
		c6	0	DTLBIALL	UNK	Invalidate data TLB, see the *ARM Architecture Reference Manual*
			1	DTLBIMVA	UNK	Invalidate data TLB entry by MVA, see the *ARM Architecture Reference Manual*
			2	DTLBIASID	UNK	Invalidate data TLB by ASID match, see the *ARM Architecture Reference Manual*
		c7	0	TLBIALL	UNK	Invalidate unified TLB, see the *ARM Architecture Reference Manual*
			1	TLBIMVA	UNK	Invalidate unified TLB entry by MVA, see the *ARM Architecture Reference Manual*
			2	TLBIASID	UNK	Invalidate unified TLB by ASID match, see the *ARM Architecture Reference Manual*
			3	TLBIMVAA	UNK	Invalidate unified TLB by MVA all ASID, see the *ARM Architecture Reference Manual*
	4	c3	0	TLBIALLHIS	UNK	Invalidate entire Hyp Unified TLB Inner Shareable, see the *ARM Architecture Reference Manual*
			1	TLBIMVAHIS	UNK	Invalidate Unified Hyp TLB entry by MVA Inner Shareable, see the *ARM Architecture Reference Manual*
			4	TLBIALLNSNHIS	UNK	Invalidate entire NS Non-Hyp Unified TLB Inner Shareable, see the *ARM Architecture Reference Manual*

Table 4-9 c8 register summary (continued)

CRn	Op1	CRm	Op2	Name	Reset	Description
c8	4	c7	0	TLBIALLH	UNK	Invalidate entire Hyp Unified TLB, see the *ARM Architecture Reference Manual*
			1	TLBIMVAH	UNK	Invalidate Unified Hyp TLB entry by MVA, see the *ARM Architecture Reference Manual*
			4	TLBIALLNSNH	UNK	Invalidate entire NS Non-Hyp Unified TLB, see the *ARM Architecture Reference Manual*

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	0x41072000	*Performance Monitor Control Register* on page 11-7
			1	PMNCNTENSET	UNK	Count Enable Set Register, see the *ARM Architecture Reference Manual*
			2	PMNCNTENCLR	UNK	Count Enable Clear Register, see the *ARM Architecture Reference Manual*
			3	PMOVSR	UNK	Overflow Flag Status Register, see the *ARM Architecture Reference Manual*
			4	PMSWINC	UNK	Software Increment Register, see the *ARM Architecture Reference Manual*
			5	PMSELR	UNK	Event Counter Selection Register, see the *ARM Architecture Reference Manual*
			6	PMCEID0	0x3FFF0F3F	Common Event Identification Register 0, see the *ARM Architecture Reference Manual*
			7	PMCEID1	0x00000000	Common Event Identification Register 1, see the *ARM Architecture Reference Manual*
		c13	0	PMCCNTR	UNK	Cycle Count Register, see the *ARM Architecture Reference Manual*
			1	PMXEVTYPER	UNK	Event Type Selection Register, see the *ARM Architecture Reference Manual*
			2	PMXEVCNTR	UNK	Event Count Register, see the *ARM Architecture Reference Manual*
		c14	0	PMUSERENR	0x00000000	User Enable Register, see the *ARM Architecture Reference Manual*

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 *ARM Architecture Reference Manual*
			2	PMINTENCLR	UNK	Interrupt Enable Clear Register, see the *ARM Architecture Reference Manual*
			3	PMOVSSET	UNK	Performance Monitor Overflow Flag Status Set Register, see the *ARM Architecture Reference Manual*
	1	c0	2	L2CTLR	0x00000000[a]	*L2 Control Register* on page 4-81
			3	L2ECTLR	0x00000000	*L2 Extended Control Register* on page 4-82

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* on page 4-54
			0	MAIR0	UNK	*MAIR0 and MAIR1, Memory Attribute Indirection Registers 0 and 1* on page 4-56
			1	NMRR	UNK	*Normal Memory Remap Register* on page 4-58
			1	MAIR1	UNK	*MAIR0 and MAIR1, Memory Attribute Indirection Registers 0 and 1* on page 4-56
		c3	0	AMAIR0	UNK	*Auxiliary Memory Attribute Indirection Register 0* on page 4-83
			1	AMAIR1	UNK	*Auxiliary Memory Attribute Indirection Register 1* on page 4-83
	4	c2	0	HMAIR0	UNK	Hyp Memory Attribute Indirection Register 0, see the *ARM Architecture Reference Manual*
			1	HMAIR1	UNK	Hyp Memory Attribute Indirection Register 1, see the *ARM Architecture Reference Manual*
		c3	0	HAMAIR0	UNK	*Hyp Auxiliary Memory Attribute Indirection Register 0* on page 4-83
			1	HAMAIR1	UNK	*Hyp Auxiliary Memory Attribute Indirection Register 1* on page 4-83

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[a]	Vector Base Address Register, see the *ARM Architecture Reference Manual*
			1	MVBAR	UNK	Monitor Vector Base Address Register, see the *ARM Architecture Reference Manual*
		c1	0	ISR	UNK	Interrupt Status Register, see the *ARM Architecture Reference Manual*
	4	c0	0	HVBAR	UNK	Hyp Vector Base Address Register, see the *ARM Architecture Reference Manual*

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	*FCSE Process ID Register* on page 4-83
			1	CONTEXTIDR	UNK	FCSE Process ID Register, see the *ARM Architecture Reference Manual*
			2	TPIDRURW	UNK	User Read/Write Thread ID Register, see the *ARM Architecture Reference Manual*
			3	TPIDRURO	UNK	User Read Only Thread ID Register, see the *ARM Architecture Reference Manual*
			4	TPIDRPRW	UNK	Privileged Only Thread ID Register, see the *ARM Architecture Reference Manual*
	4	c0	2	HTPIDR	UNK	Hyp Software Thread ID Register, see the *ARM Architecture Reference Manual*

4.2.15 c14 registers

See Chapter 9 *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	3[a]	c0	0	CDBGDR0	UNK	Data Register 0, see *Direct access to internal memory* on page 6-9
			1	CDBGDR1	UNK	Data Register 1, see *Direct access to internal memory* on page 6-9
			2	CDBGDR2	UNK	Data Register 2, see *Direct access to internal memory* on page 6-9
		c2	0	CDBGDCT	UNK	Data Cache Tag Read Operation Register, see *Direct access to internal memory* on page 6-9
			1	CDBGICT	UNK	Instruction Cache Tag Read Operation Register, see *Direct access to internal memory* on page 6-9
		c4	0	CDBGDCD	UNK	Data Cache Data Read Operation Register, see *Direct access to internal memory* on page 6-9
			1	CDBGICD	UNK	Instruction Cache Data Read Operation Register, see *Direct access to internal memory* on page 6-9
			2	CDBGTD	UNK	TLB Data Read Operation Register, see *Direct access to internal memory* on page 6-9
	4	c0	0	CBAR	-[b]	*Configuration Base Address Register* on page 4-83

a. See *Direct access to internal memory* on page 6-9 for information on how these registers are used.

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

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 *ARM Architecture Reference Manual*
-	1	c2	-	TTBR1	UNK	Translation Table Base Register 1, see the *ARM Architecture Reference Manual*
-	4	c2	-	HTTBR	UNK	Hyp Translation Table Base Register, see the *ARM Architecture Reference Manual*
-	6	c2	-	VTTBR	UNK[a]	Virtualization Translation Table Base Register, see the *ARM Architecture Reference Manual*
-	0	c7	-	PAR	UNK	*Physical Address Register* on page 4-81
-	0	c14	-	CNTPCT	UNK	Counter Physical Count Register, see the *ARM Architecture Reference Manual*
-	1	c14	-	CNTVCT	UNK	Counter Virtual Count Register, see the *ARM Architecture Reference Manual*

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

CRn	Op1	CRm	Op2	Name	Reset	Description
-	2	c14	-	CNTP_CVAL	UNK	Counter PL1 Physical Compare Value Register, see the *ARM Architecture Reference Manual*
-	3	c14	-	CNTV_CVAL	UNK	Counter PL1 Virtual Compare Value Register, see the *ARM Architecture Reference Manual*
-	4	c14	-	CNTVOFF	UNK	Counter Virtual Offset Register, see the *ARM Architecture Reference Manual*
-	6	c14	-	CNTHP_CVAL	UNK	Counter Non-secure PL2 Physical Compare Value Register, see the *ARM Architecture Reference Manual*

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	`0x410FC073`	*Main ID Register* on page 4-26
CTR				1	`0x84448003`	*Cache Type Register* on page 4-27
TCMTR				2	`0x00000000`	*TCM Type Register* on page 4-28
TLBTR				3	`0x00000000`	*TLB Type Register* on page 4-28
MPIDR				5	-[a]	*Multiprocessor Affinity Register* on page 4-28
REVIDR				6	`0x00000000`	*Revision ID Register* on page 4-29
MIDR				4, 7	`0x410FC074`	Aliases of Main ID Register, *Main ID Register* on page 4-26
ID_PFR0			c1	0	`0x00001131`	*Processor Feature Register 0* on page 4-30
ID_PFR1				1	`0x00011011`	*Processor Feature Register 1* on page 4-31
ID_DFR0				2	`0x02010555`	*Debug Feature Register 0* on page 4-32
ID_AFR0				3	`0x00000000`	*Auxiliary Feature Register 0* on page 4-33
ID_MMFR0				4	`0x10101105`	*Memory Model Feature Register 0* on page 4-33
ID_MMFR1				5	`0x40000000`	*Memory Model Feature Register 1* on page 4-34
ID_MMFR2				6	`0x01240000`	*Memory Model Feature Register 2* on page 4-36
ID_MMFR3				7	`0x02102211`	*Memory Model Feature Register 3* on page 4-37
ID_ISAR0			c2	0	`0x01101110`	*Instruction Set Attribute Register 0* on page 4-39
ID_ISAR1				1	`0x13112111`	*Instruction Set Attribute Register 1* on page 4-40
ID_ISAR2				2	`0x21232041`	*Instruction Set Attribute Register 2* on page 4-42
ID_ISAR3				3	`0x11112131`	*Instruction Set Attribute Register 3* on page 4-43
ID_ISAR4				4	`0x10011142`	*Instruction Set Attribute Register 4* on page 4-45

Table 4-16 Identification registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
ID_ISAR5	c0	0	c2	5	0x00000000	*Instruction Set Attribute Register 5* on page 4-46
CCSIDR		1	c0	0	UNK	*Cache Size ID Register* on page 4-46
CLIDR				1	0x0A200023	*Cache Level ID Register* on page 4-48
AIDR				7	0x00000000	*Auxiliary ID Register* on page 4-49
CSSELR		2	c0	0	UNK	*Cache Size Selection Register* on page 4-49

a. The reset value depends on the primary input, **CLUSTERID**, and the number of configured processors in the Cortex-A7 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	0x00C50078[a]	32 bit	*System Control Register* on page 4-51
TTBR0	c2	0	c0	0	UNK	32 bit	Translation Table Base Register 0, see the *ARM Architecture Reference Manual*
	-	0	c2	-		64 bit	
TTBR1		0	c0	1	UNK	32 bit	Translation Table Base Register 1, see the *ARM Architecture Reference Manual*
	-	1	c2	-		64 bit	
TTBCR		0	c0	2	0x00000000[b]	32 bit	Translation Table Base Control Register, see the *ARM Architecture Reference Manual*
DACR	c3	0	c0	0	UNK	32 bit	Domain Access Control Register, see the *ARM Architecture Reference Manual*
PRRR	c10	0	c2	0	UNK	32 bit	*Primary Region Remap Register* on page 4-54
MAIR0				0	UNK	32 bit	*MAIR0 and MAIR1, Memory Attribute Indirection Registers 0 and 1* on page 4-56
NMRR				1	UNK	32 bit	*Normal Memory Remap Register* on page 4-58
MAIR1				1	UNK	32 bit	*MAIR0 and MAIR1, Memory Attribute Indirection Registers 0 and 1* on page 4-56
CONTEXTIDR	c13	0	c0	1	UNK	32 bit	Process ID Register, see the *ARM Architecture Reference Manual*

a. The reset value depends on primary inputs, **CFGTE**, **CFGEND**, and **VINITHI**. The value shown in Table 4-17 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	*Data Fault Status Register* on page 4-73
IFSR				1	UNK	*Instruction Fault Status Register* on page 4-77
ADFSR			c1	0	UNK	*Auxiliary Data Fault Status Register* on page 4-79
AIFSR				1	0x00000000	*Auxiliary Instruction Fault Status Register* on page 4-79
DFAR	c6	0	c0	0	UNK	Data Fault Address Register, see the *ARM Architecture Reference Manual*
IFAR				2	UNK	*Instruction Fault Address Register, see the ARM Architecture Reference Manual*

The Virtualization Extensions include additional fault handling registers. For more information see *Virtualization Extensions registers* on page 4-22.

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	0x00000000	*Auxiliary Control Register* on page 4-59
CPACR				2	-[a]	*Coprocessor Access Control Register* on page 4-61
FCSEIDR	c13	0	c0	0	0x00000000	*FCSE Process ID Register* on page 4-83

a. The reset value depends on the FPU and NEON configuration. If FPU and Advanced SIMD are implemented, the reset value is 0x00000000. If FPU is implemented but Advanced SIMD is not implemented, the reset value is 0xC0000000. If FPU and Advanced SIMD are not implemented, the reset value is Unknown.

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
ICIALLUIS	c7	0	c1	0	UNK	Instruction cache invalidate all to PoU[a] Inner Shareable, see the *ARM Architecture Reference Manual*
BPIALLIS				6	UNK	Branch predictor invalidate all Inner Shareable, see the *ARM Architecture Reference Manual*
ICIALLU			c5	0	UNK	Instruction cache invalidate all to PoU, see the *ARM Architecture Reference Manual*
ICIMVAU				1	UNK	Instruction cache invalidate by MVA to PoU, see the *ARM Architecture Reference Manual*

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

Name	CRn	Op1	CRm	Op2	Reset	Description
BPIALL	c7	0	c5	6	UNK	Branch predictor invalidate all, see the *ARM Architecture Reference Manual*
BPIMVA				7	UNK	Branch predictor invalidate by MVA, see the *ARM Architecture Reference Manual*
DCIMVAC			c6	1	UNK	Data cache invalidate by MVA to PoC[b], see the *ARM Architecture Reference Manual*
DCISW				2	UNK	Data cache invalidate by set/way, see the *ARM Architecture Reference Manual*
DCCMVAC			c10	1	UNK	Data cache clean by MVA to PoC, see the *ARM Architecture Reference Manual*
DCCSW				2	UNK	Data cache clean by set/way, see the *ARM Architecture Reference Manual*
DCCMVAU			c11	1	UNK	Data cache clean by MVA to PoU, see the *ARM Architecture Reference Manual*
DCCIMVAC			c14	1	UNK	Data cache clean and invalidate by MVA to PoC, see the *ARM Architecture Reference Manual*
DCCISW				2	UNK	Data cache clean and invalidate by set/way, see the *ARM Architecture Reference Manual*

a. PoU = Point of Unification. PoU is set by the **BROADCASTINNER** pin and can be in the L1 data cache or outside of the processor, in which case PoU is dependent on the external memory system.

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

4.2.23 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 *ARM Architecture Reference Manual*
TLBIMVAIS				1	UNK	Invalidate unified TLB by MVA Inner Shareable, see the *ARM Architecture Reference Manual*
TLBIASIDIS				2	UNK	Invalidate unified TLB by ASID Inner Shareable, see the *ARM Architecture Reference Manual*
TLBIMVAAIS				3	UNK	Invalidate unified TLB by MVA all ASID Inner Shareable, see the *ARM Architecture Reference Manual*
ITLBIALL			c5	0	UNK	Invalidate entire instruction TLB, see the *ARM Architecture Reference Manual*
ITLBIMVA				1	UNK	Invalidate instruction TLB entry by MVA, see the *ARM Architecture Reference Manual*
ITLBIASID				2	UNK	Invalidate instruction TLB by ASID, see the *ARM Architecture Reference Manual*

Table 4-21 TLB maintenance operations (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
DTLBIALL	c8	0	c6	0	UNK	Invalidate entire data TLB, see the *ARM Architecture Reference Manual*
DTLBIMVA				1	UNK	Invalidate data TLB entry by MVA, see the *ARM Architecture Reference Manual*
DTLBIASID				2	UNK	Invalidate data TLB by ASID, see the *ARM Architecture Reference Manual*
TLBIALL			c7	0	UNK	Invalidate entire unified TLB, see the *ARM Architecture Reference Manual*
TLBIMVA				1	UNK	Invalidate unified TLB by MVA, see the *ARM Architecture Reference Manual*
TLBIASID				2	UNK	Invalidate unified TLB by ASID, see the *ARM Architecture Reference Manual*
TLBIMVAA				3	UNK	Invalidate unified TLB by MVA all ASID, see the *ARM Architecture Reference Manual*
TLBIALLHIS		4	c3	0	UNK	Invalidate entire Hyp unified TLB Inner Shareable, see the *ARM Architecture Reference Manual*
TLBIMVAHIS				1	UNK	Invalidate Hyp unified TLB entry by MVA Inner Shareable, see the *ARM Architecture Reference Manual*
TLBIALLNSNHIS				4	UNK	Invalidate entire Non-secure Non-Hyp unified TLB Inner Shareable, see the *ARM Architecture Reference Manual*
TLBIALLH			c7	0	UNK	Invalidate entire Hyp unified TLB, see the *ARM Architecture Reference Manual*
TLBIMVAH				1	UNK	Invalidate Hyp unified TLB entry by MVA, see the *ARM Architecture Reference Manual*
TLBIALLNSNH				4	UNK	Invalidate entire Non-secure Non-Hyp unified TLB, see the *ARM Architecture Reference Manual*

The Virtualization Extensions include additional TLB operations for use in Hyp mode. For more information, see *Virtualization Extensions registers* on page 4-22.

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	*Physical Address Register* on page 4-81
	-		c7	-		64-bit	
ATS1CPR			c8	0	UNK	32-bit	Stage 1 current state PL1 read, see the *ARM Architecture Reference Manual*
ATS1CPW				1	UNK	32-bit	Stage 1 current state PL1 write, see the *ARM Architecture Reference Manual*

Table 4-22 Address translation operations (continued)

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
ATS1CUR	c7	0	c8	2	UNK	32-bit	Stage 1 current state unprivileged (PL0) read, see the *ARM Architecture Reference Manual*
ATS1CUW				3	UNK	32-bit	Stage 1 current state unprivileged (PL0) write, see the *ARM Architecture Reference Manual*
ATS12NSOPR				4	UNK	32-bit	Stages 1 and 2 Non-secure PL1 read, see the *ARM Architecture Reference Manual*
ATS12NSOPW				5	UNK	32-bit	Stages 1 and 2 Non-secure PL1 write, see the *ARM Architecture Reference Manual*
ATS12NSOUR				6	UNK	32-bit	Stages 1 and 2 Non-secure unprivileged (PL0) read, see the *ARM Architecture Reference Manual*
ATS12NSOUW				7	UNK	32-bit	Stages 1 and 2 Non-secure unprivileged (PL0) write, see the *ARM Architecture Reference Manual*
ATS1HR		4	c8	0	UNK	32-bit	Stage 1 Hyp mode read, see the *ARM Architecture Reference Manual*
ATS1HW				1	UNK	32-bit	Stage 1 Hyp mode write, see the *ARM Architecture Reference Manual*

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 *ARM Architecture Reference Manual*
CP15ISB			c5	4	UNK	Instruction Synchronization Barrier operation, see the *ARM Architecture Reference Manual*
CP15DSB			c10	4	UNK	Data Synchronization Barrier operation, see the *ARM Architecture Reference Manual*
CP15DMB				5	UNK	Data Memory Barrier operation, see the *ARM Architecture Reference Manual*
NOP			c13	1	UNK	*System control No Operation* (NOP), see the *ARM Architecture Reference Manual*
TPIDRURW	c13	0	c0	2	UNK	User Read/Write Thread ID Register, see the *ARM Architecture Reference Manual*
TPIDRURO				3	UNK	User Read-Only Thread ID Register, see the *ARM Architecture Reference Manual*
TPIDRPRW				4	UNK	PL1 only Thread ID Register, see the *ARM Architecture Reference Manual*
HTPIDR		4	c0	2	UNK	Hyp Software Thread ID Register, see the *ARM Architecture Reference Manual*

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	0x41072000	*Performance Monitor Control Register* on page 11-7
PMNCNTENSET				1	UNK	Count Enable Set Register, see the *ARM Architecture Reference Manual*
PMNCNTENCLR				2	UNK	Count Enable Clear Register, see the *ARM Architecture Reference Manual*
PMOVSR				3	UNK	Overflow Flag Status Register, see the *ARM Architecture Reference Manual*
PMSWINC				4	UNK	Software Increment Register, see the *ARM Architecture Reference Manual*
PMSELR				5	UNK	Event Counter Selection Register, see the *ARM Architecture Reference Manual*
PMCEID0				6	0x3FFF0F3F	Common Event Identification Register 0, see the *ARM Architecture Reference Manual*
PMCEID1				7	0x00000000	Common Event Identification Register 1, see the *ARM Architecture Reference Manual*
PMCCNTR			c13	0	UNK	Cycle Count Register, see the *ARM Architecture Reference Manual*
PMXEVTYPER				1	UNK	Event Type Select Register, see the *ARM Architecture Reference Manual*
PMXEVCNTR				2	UNK	Event Count Register, see the *ARM Architecture Reference Manual*
PMUSERENR			c14	0	0x00000000	User Enable Register, see the *ARM Architecture Reference Manual*
PMINTENSET				1	UNK	Interrupt Enable Set Register, see the *ARM Architecture Reference Manual*
PMINTENCLR				2	UNK	Interrupt Enable Clear Register, see the *ARM Architecture Reference Manual*
PMOVSSET				3	UNK	Performance Monitor Overflow Flag Status Set Register, see the *ARM Architecture Reference Manual*

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	*Secure Configuration Register* on page 4-63
SDER				1	UNK	Secure Debug Enable Register, see the *ARM Architecture Reference Manual*
NSACR				2	0x00000000[a]	*Non-Secure Access Control Register* on page 4-65
VBAR	c12	0	c0	0	0x00000000[b]	Vector Base Address Register, see the *ARM Architecture Reference Manual*
MVBAR				1	UNK	Monitor Vector Base Address Register, see the *ARM Architecture Reference Manual*
ISR			c1	0	UNK	Interrupt Status Register, see the *ARM Architecture Reference Manual*

a. The reset value depends on the FPU and NEON configuration. If FPU and Advanced SIMD are implemented, the reset value is 0x00000000. If FPU is implemented but Advanced SIMD is not implemented, the reset value is 0x00008000. If FPU and Advanced SIMD are not implemented, the reset value is 0x00000000.

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

4.2.28 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	-[a]	32-bit	*Virtualization Processor ID Register* on page 4-50*l*
VMPIDR				5	-[b]	32-bit	*Virtualization Multiprocessor ID Register* on page 4-51
HSCTLR	c1	4	c0	0	UNK	32-bit	*Hyp System Control Register* on page 4-66
HACTLR				1	UNK		*Hyp Auxiliary Configuration Register* on page 4-68
HCR			c1	0	0x00000000	32-bit	Hyp Configuration Register, see the *ARM Architecture Reference Manual*
HDCR				1	0x00000006[c]	32-bit	*Hyp Debug Control Register* on page 4-68
HCPTR				2	0x000033FF[d]	32-bit	*Hyp Coprocessor Trap Register* on page 4-70
HSTR				3	0x00000000	32-bit	Hyp System Trap Register, see the *ARM Architecture Reference Manual*
HTCR	c2	4	c0	2	UNK	32-bit	*Hyp Translation Control Register* on page 4-73
VTCR			c1	2	UNK	32-bit	Virtualization Translation Control Register, see the *ARM Architecture Reference Manual*
HTTBR	-	4	c2	-	UNK	64-bit	Hyp Translation Table Base Register, see the *ARM Architecture Reference Manual*

Table 4-26 Virtualization Extensions registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Width	Description
VTTBR	-	6	c2	-	UNK[e]	64-bit	Virtualization Translation Table Base Register, see the *ARM Architecture Reference Manual*
HADFSR	c5	4	c1	0	UNK	32-bit	*Hyp Auxiliary Data Fault Status Syndrome Register* on page 4-79
HAIFSR				1	UNK	32-bit	*Hyp Auxiliary Instruction Fault Status Syndrome Register* on page 4-79
HSR			c2	0	UNK	32-bit	*Hyp Syndrome Register* on page 4-80
HDFAR	c6	4	c0	0	UNK	32-bit	*Hyp Data Fault Address Register, see the ARM Architecture Reference Manual*
HIFAR				2	UNK	32-bit	*Hyp Instruction Fault Address Register, see the ARM Architecture Reference Manual*
HPFAR				4	UNK	32-bit	*Hyp IPA Fault Address Register, see the ARM Architecture Reference Manual*
HMAIR0	c10	4	c2	0	UNK	32-bit	Hyp Memory Attribute Indirection Register 0, see the *ARM Architecture Reference Manual*
HMAIR1				1	UNK	32-bit	Hyp Memory Attribute Indirection Register 1, see the *ARM Architecture Reference Manual*
HAMAIR0			c3	0	UNK	32-bit	*Hyp Auxiliary Memory Attribute Indirection Register 0* on page 4-83
HAMAIR1				1	UNK	32-bit	*Hyp Auxiliary Memory Attribute Indirection Register 1* on page 4-83
HVBAR	c12	4	c0	0	UNK	32-bit	Hyp Vector Base Address Register, see the *ARM Architecture Reference Manual*

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 FPU and NEON configuration. If FPU and Advanced SIMD are implemented, the reset value is `0x000033FF`. If FPU is implemented but Advanced SIMD is not implemented, the reset value is `0x0000B3FF`. If FPU and Advanced SIMD are not implemented, the reset value is `0x0000BFFF`.

e. The reset value for bits [54:48] is 0b00000000.

4.2.29 TLB maintenance operations

Table 4-28 shows the 32-bit wide TLB maintenance operation registers added for Virtualization Extensions.

Table 4-27 TLB maintenance operations

Name	CRn	Op1	CRm	Op2	Reset	Description
TLBIALLHIS	c8	4	c3	0	UNK	Invalidate entire Hyp unified TLB Inner Shareable, see the *ARM Architecture Reference Manual*
TLBIMVAHIS				1	UNK	Invalidate Hyp unified TLB by MVA Inner Shareable, see the *ARM Architecture Reference Manual*
TLBIALLNSNHIS				4	UNK	Invalidate entire Non-secure Non-Hyp unified TLB Inner Shareable, see the *ARM Architecture Reference Manual*
TLBIALLH			c7	0	UNK	Invalidate entire Hyp unified TLB, see the *ARM Architecture Reference Manual*
TLBIMVAH				1	UNK	Invalidate Hyp unified TLB by MVA, see the *ARM Architecture Reference Manual*
TLBIALLNSNH				4	UNK	Invalidate entire Non-secure Non-Hyp unified TLB, see the *ARM Architecture Reference Manual*

4.2.30 Generic Timer registers

See Chapter 9 *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-A7 MPCore processor.

Table 4-28 Memory access registers

Name	CRn	Op1	CRm	Op2	Reset	Description
L2CTLR	c9	1	c0	2	0x00000000[a]	*L2 Control Register* on page 4-81
L2ECTLR				3	0x00000000	*L2 Extended Control Register* on page 4-82
CDBGDR0	c15	3[b]	c0	0	UNK	Data Register 0, see *Direct access to internal memory* on page 6-9
CDBGDR1				1	UNK	Data Register 1, see *Direct access to internal memory* on page 6-9
CDBGDR2				2	UNK	Data Register 2, see *Direct access to internal memory* on page 6-9
CDBGDCT			c2	0	UNK	Data Cache Tag Read Operation Register, see *Direct access to internal memory* on page 6-9
CDBGICT				1	UNK	Instruction Cache Tag Read Operation Register, see *Direct access to internal memory* on page 6-9
CDBGDCD			c4	0	UNK	Data Cache Data Read Operation Register, see *Direct access to internal memory* on page 6-9

Table 4-28 Memory access registers (continued)

Name	CRn	Op1	CRm	Op2	Reset	Description
CDBGICD	c9	3	c4	1	UNK	Instruction Cache Data Read Operation Register, see *Direct access to internal memory* on page 6-9
CDBGTD				2	UNK	TLB Data Read Operation Register, see *Direct access to internal memory* on page 6-9
CBAR		4	c0	0	-[c]	*Configuration Base Address Register* on page 4-83

a. The reset value depends on the processor configuration.

b. See *Direct access to internal memory* on page 6-9 for information on how these registers are used.

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

4.3 Register descriptions

This section describes all the CP15 system control registers by coprocessor register number order. Table 4-2 on page 4-4 to Table 4-15 on page 4-14 provide cross references to individual registers.

4.3.1 Main ID Register

The MIDR characteristics are:

Purpose Provides identification information for the processor, including an implementer code for the device and a part 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 4-4.

Figure 4-1 shows the MIDR bit assignments.

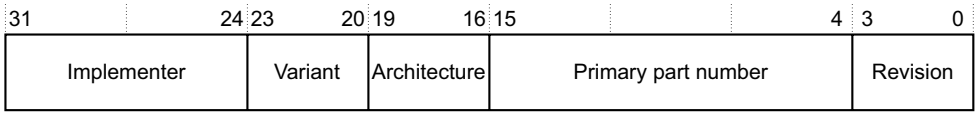


Figure 4-1 MIDR bit assignments

Table 4-29 shows the MIDR bit assignments.

Table 4-29 MIDR bit assignments

Bits	Name	Function
[31:24]	Implementer	Indicates the implementer code: `0x41` ARM.
[23:20]	Variant	Indicates the variant number of the processor. This is the major revision number *n* in the *rn* part of the rnpn description of the product revision status: `0x0` Major revision number.
[19:16]	Architecture	Indicates the architecture code: `0xF` ARMv7.
[15:4]	Primary part number	Indicates the primary part number: `0xC07` Cortex-A7 MPCore part number.
[3:0]	Revision	Indicates the revision number of the processor. This is the minor revision number *n* in the *pn* part of the *rnpn* description of the product revision status: `0x4` Minor revision number.

To access the MIDR, read the CP15 register with:

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

4.3.2 Cache Type Register

The CTR characteristics are:

Purpose Provides information about the architecture of the caches.

Usage constraints The CTR is:

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

Configurations Available in all configurations.

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

Figure 4-2 shows the CTR bit assignments.

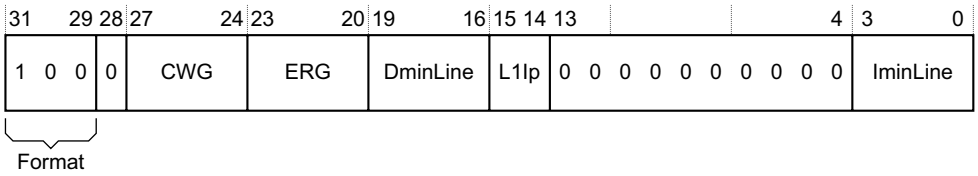


Figure 4-2 CTR bit assignments

Table 4-30 shows the CTR bit assignments.

Table 4-30 CTR bit assignments

Bits	Name	Function
[31:29]	Format	Indicates the CTR format: `0x4` ARMv7 format.
[28]	-	Reserved, RAZ.
[27:24]	CWG	Cache Write-Back granule. Log_2 of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified: `0x4` Cache Write-Back granule size is 16 words.
[23:20]	ERG	Exclusives Reservation Granule. Log_2 of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions: `0x4` Exclusive reservation granule size is 16 words.
[19:16]	DminLine	Log_2 of the number of words in the smallest cache line of all the data and unified caches that the processor controls: `0x4` Smallest data cache line size is 16 words.
[15:14]	L1Ip	L1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache: `b10` *Virtually Indexed Physically Tagged* (VIPT).
[13:4]	-	Reserved, RAZ.
[3:0]	IminLine	Log_2 of the number of words in the smallest cache line of all the instruction caches that the processor controls. `0x3` Smallest instruction cache line size is 8 words.

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 4-5.

Figure 4-3 shows the TLBTR bit assignments.

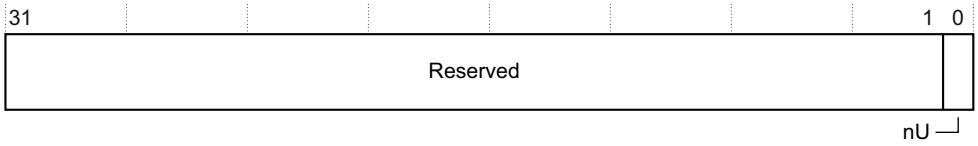


Figure 4-3 TLBTR bit assignments

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 4-4.

Figure 4-4 shows the MPIDR bit assignments.

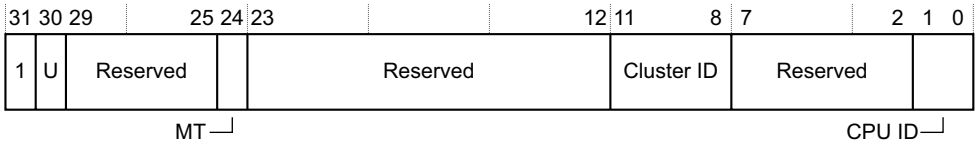


Figure 4-4 MPIDR bit assignments

Table 4-32 shows the MPIDR bit assignments.

Table 4-32 MPIDR bit assignments

Bits	Name	Function
[31]	-	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 **CLUSTERID** configuration pin. It identifies each Cortex-A7 MPCore processor in a system with more than one processor present. That is, there are other processors in the multiprocessor system that might not be Cortex-A7 MPCore processors.
[7:2]	-	Reserved, RAZ.
[1:0]	CPU ID	Indicates the processor number in the Cortex-A7 MPCore processor. For: • One processor, the CPU ID is `0x0`. • Two processors, the CPU IDs are `0x0` and `0x1`. • Three processors, the CPU IDs are `0x0`, `0x1`, and `0x2`. • Four processors, the CPU IDs are `0x0`, `0x1`, `0x2`, and `0x3`. .

To access the MPIDR, read the CP15 registers with:

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

4.3.6 Revision ID Register

The REVIDR characteristics are:

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

Usage constraints The REVIDR is:
- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

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

Figure 4-5 shows the REVIDR bit assignments.

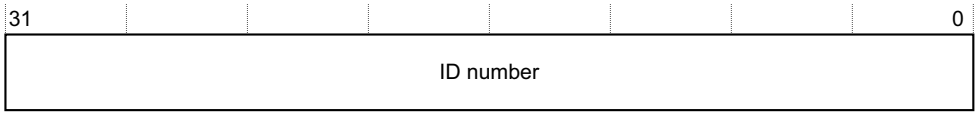


Figure 4-5 REVIDR bit assignments

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-A7 MPCore implementation.

To access the REVIDR, read the CP15 register with:

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

4.3.7 Processor Feature Register 0

The ID_PFR0 characteristics are:

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

Usage constraints The ID_PFR0 is:

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

Configurations Available in all configurations.

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

Figure 4-6 shows the ID_PFR0 bit assignments.

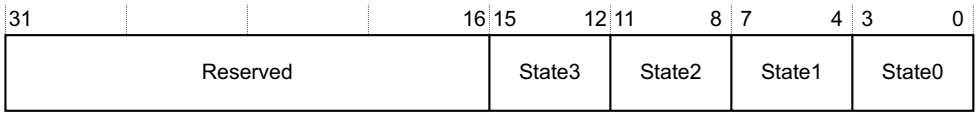


Figure 4-6 ID_PFR0 bit assignments

Table 4-34 shows the ID_PFR0 bit assignments.

Table 4-34 ID_PFR0 bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RAZ.
[15:12]	State3	Indicates support for *Thumb Execution Environment* (ThumbEE) instruction set: `0x1` 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 4-4.

Figure 4-7 shows the ID_PFR1 bit assignments.

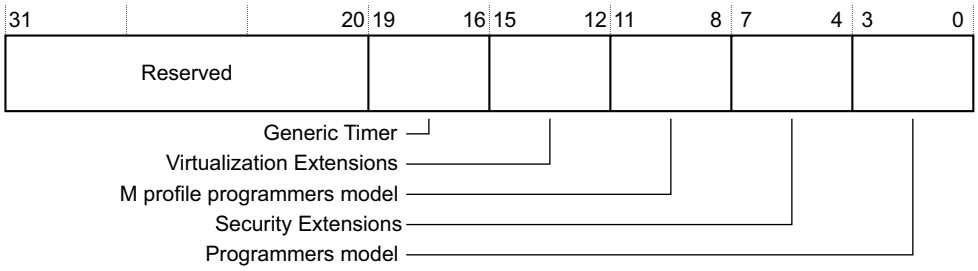


Figure 4-7 ID_PFR1 bit assignments

Table 4-35 shows the ID_PFR1 bit assignments.

Table 4-35 ID_PFR1 bit assignments

Bits	Name	Function
[31:20]	-	Reserved, RAZ.
[19:16]	Generic Timer	Indicates support for Generic Timer: 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 4-4.

Figure 4-8 shows the ID_DFR0 bit assignments.

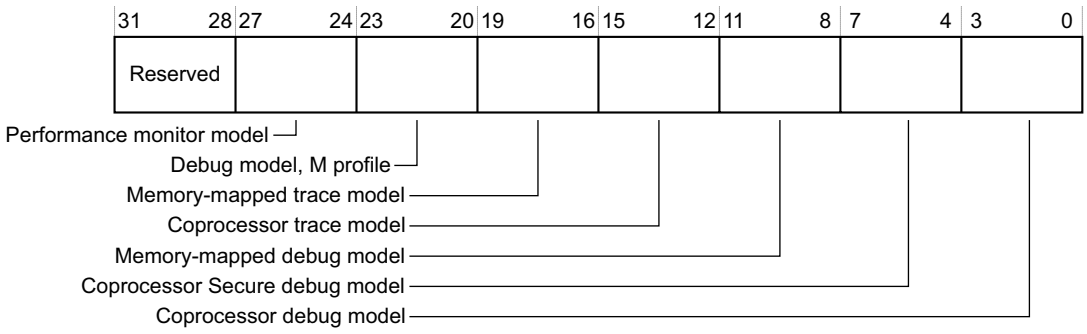


Figure 4-8 ID_DFR0 bit assignments

Table 4-36 shows the ID_DFR0 bit assignments.

Table 4-36 ID_DFR0 bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RAZ.
[27:24]	Performance monitor model	Indicates support for performance monitor model: `0x2` Processor supports *Performance Monitor Unit version 2* (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:

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

Usage constraints The ID_MMFR0 is:

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

Configurations Available in all configurations.

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

Figure 4-9 on page 4-34 shows the ID_MMFR0 bit assignments.

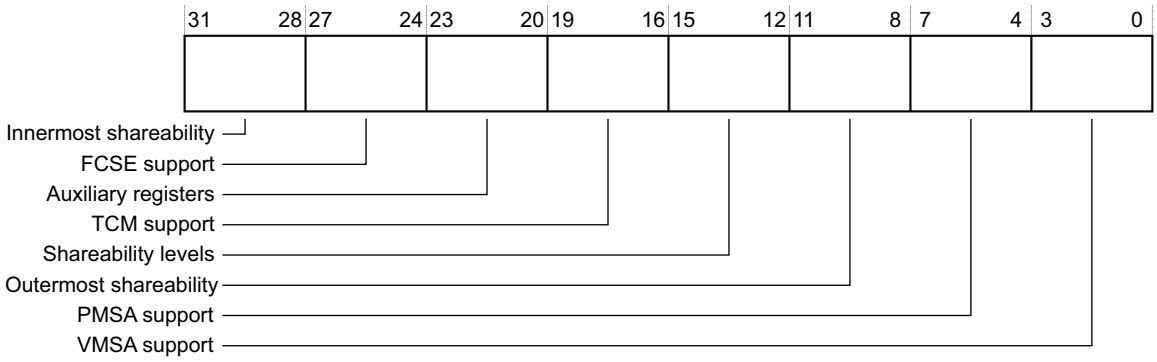


Figure 4-9 ID_MMFR0 bit assignments

Table 4-37 shows the ID_MMFR0 bit assignments.

Table 4-37 ID_MMFR0 bit assignments

Bits	Name	Function
[31:28]	Innermost shareability	Indicates the innermost shareability domain implemented: `0x1` Processor implements hardware coherency support.
[27:24]	FCSE	Indicates support for *Fast Context Switch Extension* (FCSE): `0x0` Processor does not support FCSE.
[23:20]	Auxiliary registers	Indicates support for Auxiliary registers: `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 *Protected Memory System Architecture* (PMSA): `0x0` Processor does not support PMSA.
[3:0]	VMSA support	Indicates support for a *Virtual Memory System Architecture* (VMSA). `0x5` Processor supports: • VMSAv7, with support for remapping and the Access flag. • *Privileged Execute Never* (PXN) within the first level descriptors. • 64-bit address translation descriptors.

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 4-4.

Figure 4-10 shows the ID_MMFR1 bit assignments.

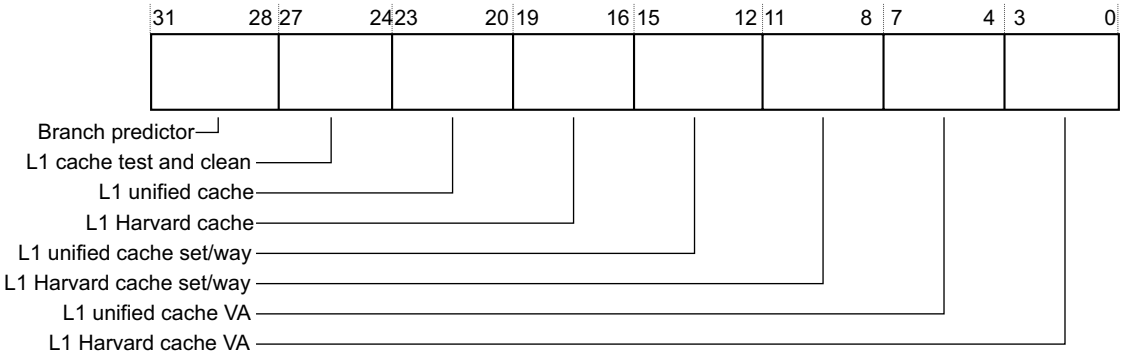


Figure 4-10 ID_MMFR1 bit assignments

Table 4-38 shows the ID_MMFR1 bit assignments.

Table 4-38 ID_MMFR1 bit assignments

Bits	Name	Function
[31:28]	Branch predictor	Indicates branch predictor management requirements: `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 4-4.

Figure 4-11 shows the ID_MMFR2 bit assignments.

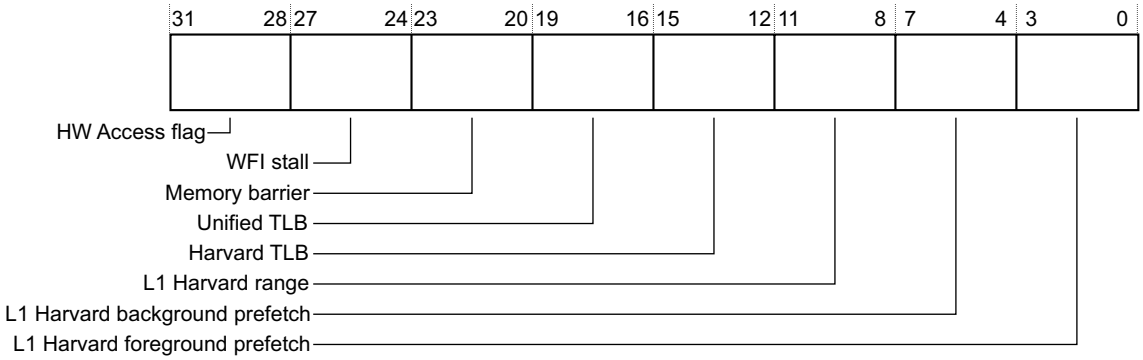


Figure 4-11 ID_MMFR2 bit assignments

Table 4-39 shows the ID_MMFR2 bit assignments.

Table 4-39 ID_MMFR2 bit assignments

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

Figure 4-12 shows the ID_MMFR3 bit assignments.

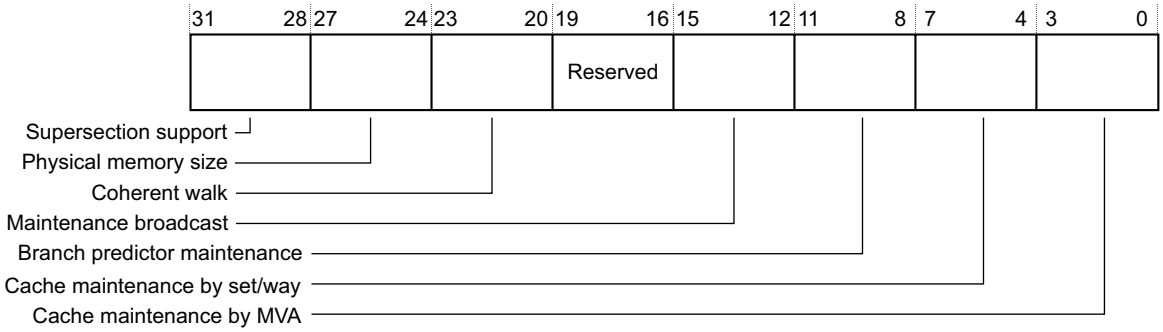


Figure 4-12 ID_MMFR3 bit assignments

Table 4-40 shows the ID_MMFR3 bit assignments.

Table 4-40 ID_MMFR3 bit assignments

Bits	Name	Function
[31:28]	Supersection support	Indicates support for supersections: `0x0` Processor supports supersections.
[27:24]	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: • Invalidate entire branch predictor array. • Invalidate branch predictor by MVA.
[7:4]	Cache maintenance by set/way	Indicates the supported cache maintenance operations by set/way. `0x1` Processor supports: • Invalidate data cache by set/way. • Clean data cache by set/way. • Clean and invalidate data cache by set/way.
[3:0]	Cache maintenance by MVA	Indicates the supported cache maintenance operations by MVA. `0x1` Processor supports: • Invalidate data cache by MVA. • Clean data cache by MVA. • Clean and invalidate data cache by MVA. • Invalidate instruction cache by MVA. • Invalidate all instruction cache entries.

To access the ID_MMFR3, read the CP15 register with:

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

4.3.15 Instruction Set Attribute Register 0

The ID_ISAR0 characteristics are:

Purpose Provides information about the instruction set that the processor supports 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 4-4.

Figure 4-13 shows the ID_ISAR0 bit assignments.

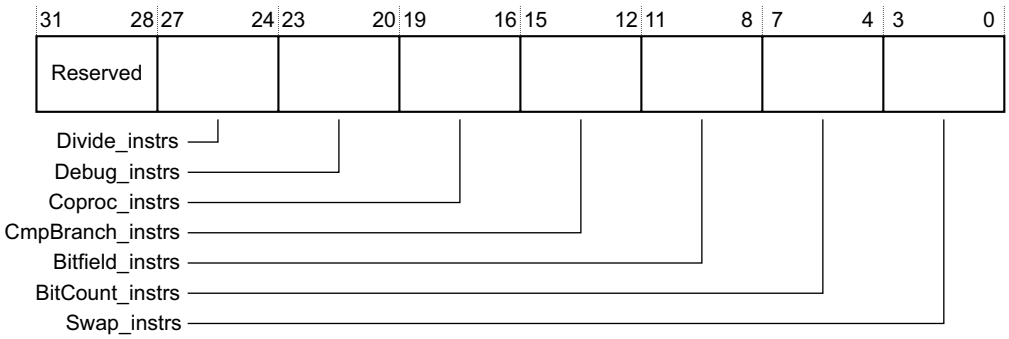


Figure 4-13 ID_ISAR0 bit assignments

Table 4-41 shows the ID_ISAR0 bit assignments.

Table 4-41 ID_ISAR0 bit assignments

Bits	Name	Function
[31:28]	-	Reserved, RAZ.
[27:24]	Divide_instrs	Indicates support for Divide instructions: 0x2 Processor supports: • SDIV and UDIV in the Thumb instruction set. • SDIV and UDIV in the ARM instruction set.
[23:20]	Debug_instrs	Indicates the supported Debug instructions: 0x1 Processor supports BKPT instruction.
[19:16]	Coproc_instrs	Indicates the supported Coprocessor instructions: 0x0 None supported, except for separately attributed architectures including CP15, CP14, and Advanced SIMD and VFP. .
[15:12]	CmpBranch_instrs	Indicates the supported combined Compare and Branch instructions in the Thumb instruction set: 0x1 Processor supports CBNZ and CBZ instructions.
[11:8]	Bitfield_instrs	Indicates the supported 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 supported fractionally[a].

a. The SWP instruction only produces a read followed by a write that are not locked on the bus, if enabled in the SCTLR. See the description of ISAR4:SWP_frac in *Instruction Set Attribute Register 4* on page 4-45.

To access the ID_ISAR0, read the CP15 register with:

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

4.3.16 Instruction Set Attribute Register 1

The ID_ISAR1 characteristics are:

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

Usage constraints The ID_ISAR1 is:
- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

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

Figure 4-14 on page 4-41 shows the ID_ISAR1 bit assignments.

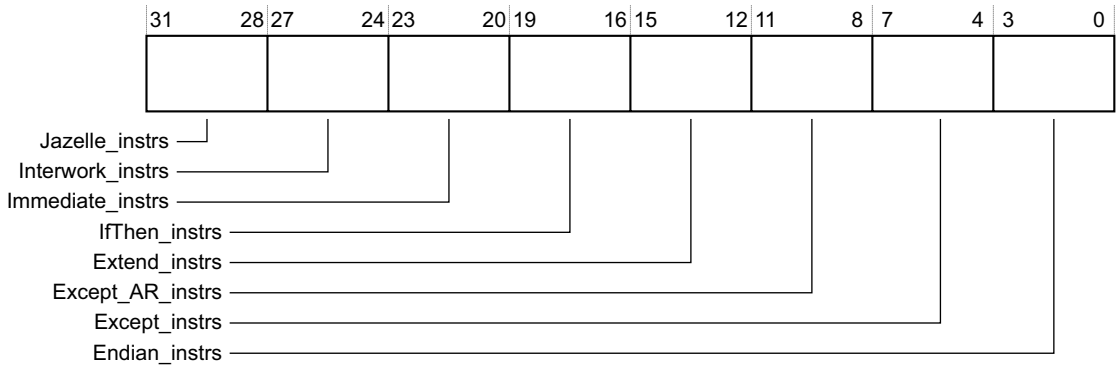


Figure 4-14 ID_ISAR1 bit assignments

Table 4-42 shows the ID_ISAR1 bit assignments.

Table 4-42 ID_ISAR1 bit assignments

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

To access the ID_ISAR1, read the CP15 register with:

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

4.3.17 Instruction Set Attribute Register 2

The ID_ISAR2 characteristics are:

Purpose Provides information about the instruction set that the processor supports 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 4-4.

Figure 4-15 shows the ID_ISAR2 bit assignments.

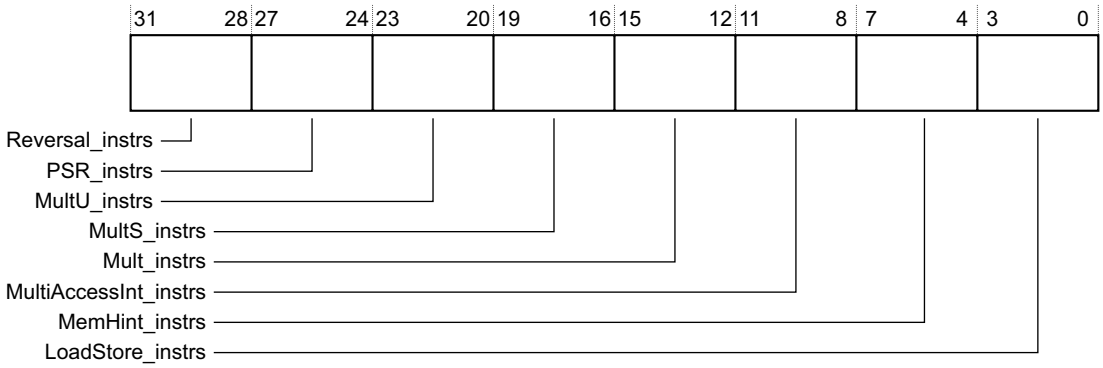


Figure 4-15 ID_ISAR2 bit assignments

Table 4-43 shows the ID_ISAR2 bit assignments.

Table 4-43 ID_ISAR2 bit assignments

Bits	Name	Function
[31:28]	Reversal_instrs	Indicates the supported Reversal instructions: 0x2 Processor supports REV, REV16, REVSH, and RBIT instructions.
[27:24]	PSR_instrs	Indicates the supported PSR instructions: 0x1 Processor supports MRS and MSR instructions, and the exception return forms of data-processing instructions. **Note** The exception return forms of the data-processing instructions are: • In the 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. • In the Thumb instruction set, the SUBS PC, LR, #N instruction.
[23:20]	MultU_instrs	Indicates the supported advanced unsigned Multiply instructions: 0x2 Processor supports UMULL, UMLAL, and UMAAL instructions.

Table 4-43 ID_ISAR2 bit assignments (continued)

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

To access the ID_ISAR2, read the CP15 register with:

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

4.3.18 Instruction Set Attribute Register 3

The ID_ISAR3 characteristics are:

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

Usage constraints The ID_ISAR3 is:
- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations Available in all configurations.

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

Figure 4-16 on page 4-44 shows the ID_ISAR3 bit assignments.

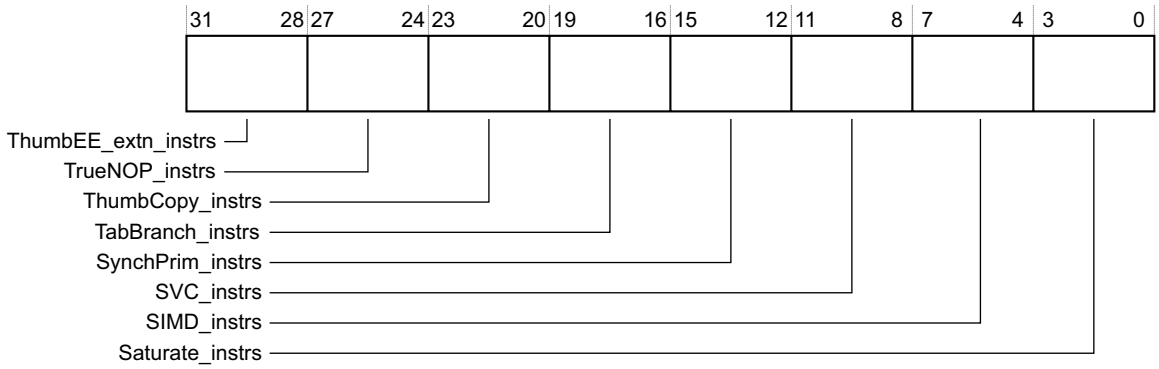


Figure 4-16 ID_ISAR3 bit assignments

Table 4-44 shows the ID_ISAR3 bit assignments.

Table 4-44 ID_ISAR3 bit assignments

Bits	Name	Function
[31:28]	ThumbEE_extn_instrs	Indicates the supported *Thumb Execution Environment* (ThumbEE) extension instructions: `0x1` Processor supports `ENTERX` and `LEAVEX` instructions, and modifies the load behavior to include null checking.
[27:24]	TrueNOP_instrs	Indicates support for True NOP instructions: `0x1` Processor supports true `NOP` instructions in both the ARM and Thumb instruction sets, and the capability for additional NOP-compatible hints.
[23:20]	ThumbCopy_instrs	Indicates the supported Thumb non flag-setting `MOV` instructions: `0x1` Processor supports Thumb instruction set encoding T1 of the `MOV` (register) instruction, copying from a low register to a low register.
[19:16]	TabBranch_instrs	Indicates the supported Table Branch instructions in the Thumb instruction set. `0x1` Processor supports `TBB` and `TBH` instructions.
[15:12]	SynchPrim_instrs	Indicates the supported Synchronization Primitive instructions. `0x2` Processor supports: • `LDREX` and `STREX` instructions. • `CLREX`, `LDREXB`, `LDREXH`, `STREXB`, and `STREXH` instructions. • `LDREXD` and `STREXD` instructions.
[11:8]	SVC_instrs	Indicates the supported SVC instructions: `0x1` Processor supports `SVC` instruction.
[7:4]	SIMD_instrs	Indicates the supported *Single Instruction Multiple Data* (SIMD) instructions. `0x3` Processor supports: • `SSAT` and `USAT` instructions, and the Q bit in the PSRs. • `PKHBT`, `PKHTB`, `QADD16`, `QADD8`, `QASX`, `QSUB16`, `QSUB8`, `QSAX`, `SADD16`, `SADD8`, `SASX`, `SEL`, `SHADD16`, `SHADD8`, `SHASX`, `SHSUB16`, `SHSUB8`, `SHSAX`, `SSAT16`, `SSUB16`, `SSUB8`, `SSAX`, `SXTAB16`, `SXTB16`, `UADD16`, `UADD8`, `UASX`, `UHADD16`, `UHADD8`, `UHASX`, `UHSUB16`, `UHSUB8`, `UHSAX`, `UQADD16`, `UQADD8`, `UQASX`, `UQSUB16`, `UQSUB8`, `UQSAX`, `USAD8`, `USADA8`, `USAT16`, `USUB16`, `USUB8`, `USAX`, `UXTAB16`, `UXTB16` instructions, and the GE[3:0] bits in the PSRs.
[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 4-4.

Figure 4-17 shows the ID_ISAR4 bit assignments.

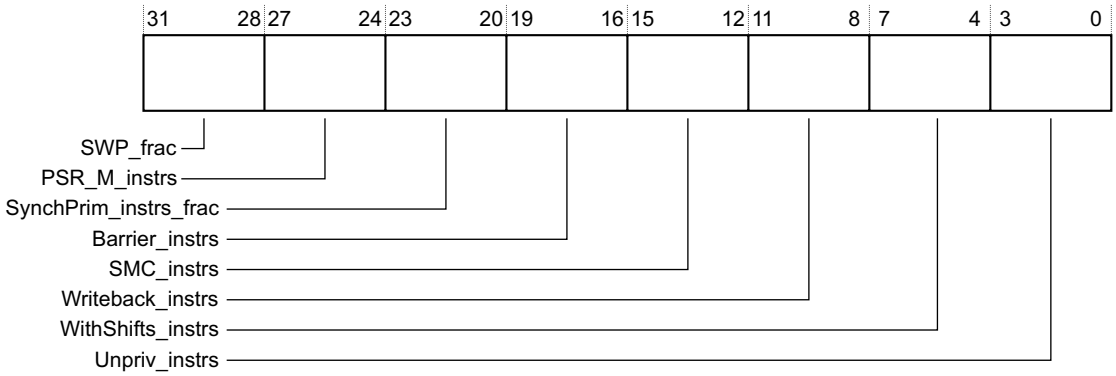


Figure 4-17 ID_ISAR4 bit assignments

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: 0x1 Processor supports SWP and SWPB instruction but only in a uniprocessor context. SWP and SWPB do not guarantee whether memory accesses from other masters can come between the load memory access and the store memory access of the SWP or SWPB instruction.
[27:24]	PSR_M_instrs	Indicates the supported M profile instructions to modify the PSRs: 0x0 None supported.
[23:20]	SynchPrim_instrs_frac	This field is used with the SynchPrim_instrs field of ID_ISAR3 to indicate the supported Synchronization Primitive instructions: 0x0 Processor supports: • LDREX and STREX instructions. • CLREX, LDREXB, LDREXH, STREXB, and STREXH instructions. • LDREXD and STREXD instructions.

Table 4-45 ID_ISAR4 bit assignments (continued)

Bits	Name	Function
[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.
[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: • Shifts of loads and stores over the range LSL 0-3. • Constant shift options, both on load/store and other instructions. • Register-controlled shift options.
[3:0]	Unpriv_instrs	Indicates the supported unprivileged instructions. 0x2 Processor supports: • LDRBT, LDRT, STRBT, and STRT instructions. • LDRHT, LDRSBT, LDRSHT, and STRHT instructions.

To access the ID_ISAR4, read the CP15 register with:

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

4.3.20 Instruction Set Attribute Register 5

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

4.3.21 Cache Size ID Register

The CCSIDR characteristics are:

Purpose Provides information about the architecture of the caches.

Usage constraints The CCSIDR is:
- A read-only register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.

Configurations There is one CCSIDR for each cache size and level of cache. The CSSELR determines which CCSIDR is accessible.

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

Figure 4-18 on page 4-47 shows the CCSIDR bit assignments.

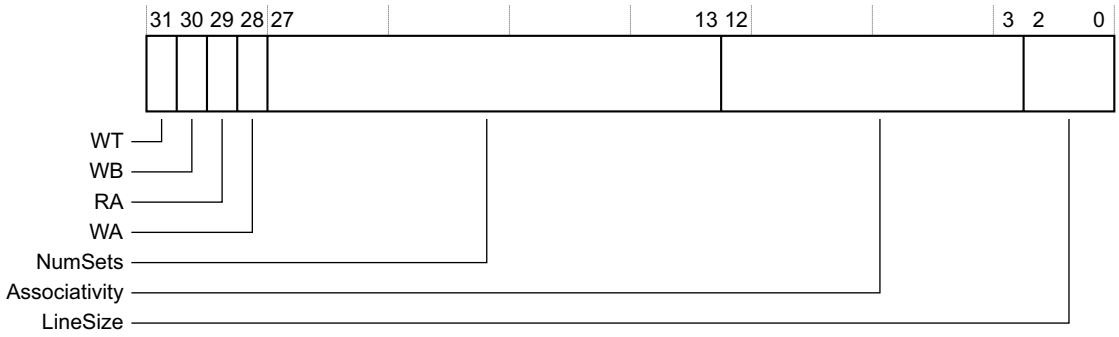


Figure 4-18 CCSIDR bit assignments

Table 4-46 shows the CCSIDR bit assignments.

Table 4-46 CCSIDR bit assignments

Bits	Name	Function
[31]	WT	Indicates support for Write-Through: **0** Cache level does not support Write-Through. **1** Cache level supports Write-Through.
[30]	WB	Indicates support for Write-Back: **0** Cache level does not support Write-Back. **1** Cache level supports Write-Back.
[29]	RA	Indicates support for Read-Allocation: **0** Cache level does not support Read-Allocation. **1** Cache level supports Read-Allocation.
[28]	WA	Indicates support for Write-Allocation: **0** Cache level does not support. **1** Cache level supports Write-Allocation.
[27:13]	NumSets[a]	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[a]	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: `0b0000000001` 2-ways. `0b0000000011` 4-ways. `0b0000000111` 8-ways.
[2:0]	LineSize[a]	Indicates the ($\log_2$ (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 on page 4-48.

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	8KB	0x7003E01A	0	1	1	1	0x1F	0x3	0x2
		16KB	0x7007E01A					0x3F	0x3	0x2
		32KB	0x700FE01A					0x7F	0x3	0x2
		64KB	0x701FE01A					0xFF	0x3	0x2
L1 instruction cache	0x1	8KB	0x200FE009	0	0	1	0	0x7F	0x1	0x1
		16KB	0x201FE009					0xFF	0x1	0x1
		32KB	0x203FE009					0x1FF	0x1	0x1
		64KB	0x207FE009					0x3FF	0x1	0x1
L2 cache	0x2	128KB	0x701FE03A	0	1	1	1	0xFF	0x7	0x2
		256KB	0x703FE03A					0x1FF	0x7	0x2
		512KB	0x707FE03A					0x3FF	0x7	0x2
		1024KB	0x70FFE03A					0x7FF	0x7	0x2
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:

Purpose Identifies:

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

Usage constraints The CLIDR is:

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

Configurations Available in all configurations.

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

Figure 4-19 on page 4-49 shows the CLIDR bit assignments.

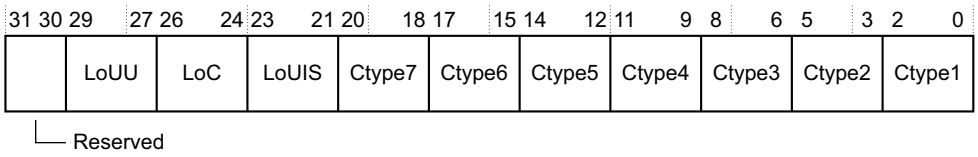


Figure 4-19 CLIDR bit assignments

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: `0b001` L2 cache not implemented on the processor. `0b010` L2 cache coherency.
[23:21]	LoUIS	Indicates the Level of Unification Inner Shareable for the cache hierarchy: `0b001` L2 cache.
[20:6]	-	Reserved, RAZ.
[5:3]	Ctype2	Indicates the type of cache if the processor implements level 2 cache: `0b000` L2 cache not implemented on the processor. `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:

Purpose Selects the current CCSIDR, see *Cache Size ID Register* on page 4-46, by specifying:

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

Usage constraints The CSSELR is:

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

Configurations Available in all configurations.

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

Figure 4-20 shows the CSSELR bit assignments.

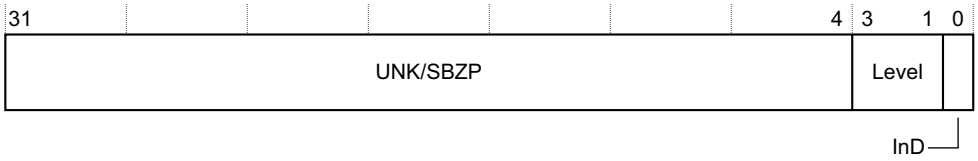


Figure 4-20 CSSELR bit assignments

Table 4-49 shows the CSSELR bit assignments.

Table 4-49 CSSELR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, UNK/SBZP
[3:1]	Level	Cache level of required cache: `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 4-4.

Figure 4-21 shows the VPIDR bit assignments.

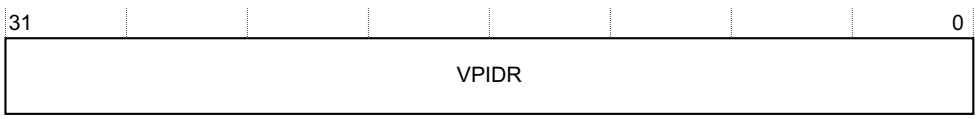


Figure 4-21 VPIDR bit assignments

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 4-4.

Figure 4-22 shows the VMPIDR bit assignments.

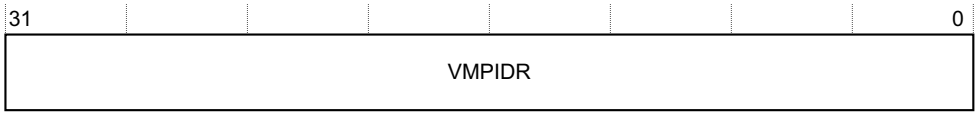


Figure 4-22 VMPIDR bit assignments

Table 4-51 shows the VMPIDR bit assignments.

Table 4-51 VMPIDR bit assignments

Bits	Name	Function
[31:0]	VMPIDR	MPIDR value returned by Non-secure 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.

- Only accessible from PL1 or higher.
- Has write access to the Secure copy of the register disabled when the **CP15SDISABLE** signal is asserted HIGH. Attempts to write to this register in Secure PL1 modes when **CP15SDISABLE** is HIGH result in an Undefined Instruction exception.

Configurations Available in all configurations.

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

Figure 4-23 shows the SCTLR bit assignments.

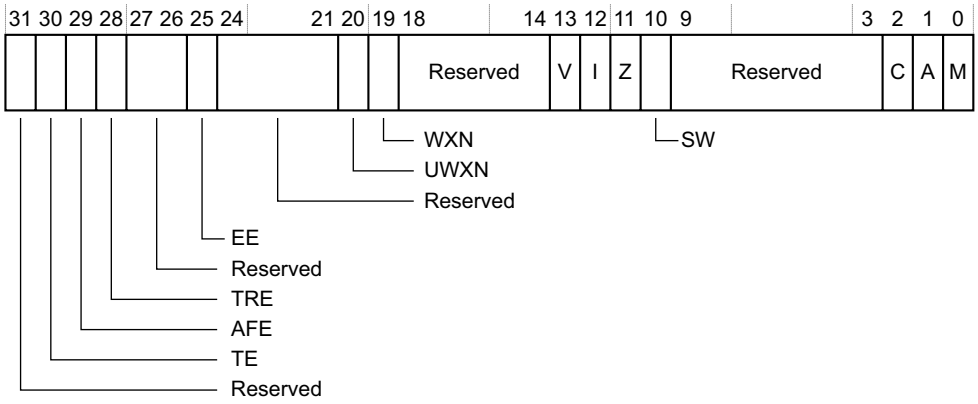


Figure 4-23 SCTLR bit assignments

Table 4-52 shows the SCTLR bit assignments.

Table 4-52 SCTLR bit assignments

Bits	Name	Function
[31]	-	Reserved, UNK/SBZP.
[30]	TE	Thumb Exception enable. This bit controls whether exceptions are taken in ARM or Thumb state: **0** Exceptions, including reset, taken in ARM state. **1** Exceptions, including reset, taken in Thumb state. The primary input **CFGTE** defines the reset value of the TE bit.
[29]	AFE	Access flag enable bit. This bit enables use of the AP[0] bit in the translation table descriptors as the Access flag. **0** In the translation table descriptors, AP[0] is an access permissions bit. The full range of access permissions is supported. No Access flag is implemented. This is the reset value. **1** In the translation table descriptors, AP[0] is the Access flag. Only the simplified model for access permissions is supported.
[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: **0** TEX remap disabled. TEX[2:0] are used, with the C and B bits, to describe the memory region attributes. This is the reset value. **1** TEX remap enabled. TEX[2:1] are reassigned for use as bits managed by the operating system. The TEX[0], C and B bits are used to describe the memory region attributes, with the MMU remap registers.
[27:26]	-	Reserved, 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: **0** Little endian. **1** Big endian. The primary input **CFGEND** defines the reset value of the EE bit.
[24]	-	Reserved, RAZ/WI.
[23:22]	-	Reserved, RAO/SBOP.
[21]	-	Reserved, RAZ/WI.
[20]	UWXN	Unprivileged write permission implies PL1 *Execute Never* (XN). This bit can be used to require all memory regions with unprivileged write permissions to be treated as XN for accesses from software executing at PL1. **0** Regions with unprivileged write permission are not forced to be XN, this is the reset value. **1** Regions with unprivileged write permission are forced to be XN for accesses from software executing at PL1
[19]	WXN	Write permission implies *Execute Never* (XN). This bit can be used to require all memory regions with write permissions to be treated as XN. **0** Regions with write permission are not forced to be XN, this is the reset value. **1** Regions with write permissions are forced to be XN.
[18]	-	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: **0** Normal exception vectors, base address `0x00000000`. Software can remap this base address using the VBAR. **1** High exception vectors, base address `0xFFFF0000`. This base address is never remapped. The primary input **VINITHI** defines the reset value of the V bit.
[12]	I	Instruction cache enable bit. This is a global enable bit for instruction caches: **0** Instruction caches disabled, this is the reset value. **1** 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]	SW	SWP and SWPB enable bit. This bit enables the use of `SWP` and `SWPB` instructions: **0** `SWP` and `SWPB` are UNDEFINED. This is the reset value. **1** `SWP` and `SWPB` perform normally[a].
[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: **0** Data and unified caches disabled, this is the reset value. **1** Data and unified caches enabled.
[1]	A	Alignment bit. This is the enable bit for Alignment fault checking: **0** Alignment fault checking disabled, this is the reset value. **1** Alignment fault checking enabled.
[0]	M	Address translation enable bit. This is a global enable bit for the MMU stage 1 address translation: **0** Address translation disabled, this is the reset value. **1** Address translation enabled.

a. SWP and SWPB do not cause bus locked transfers.

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 Primary Region Remap Register

The PRRR characteristics are:

Purpose Controls the top level mapping of the TEX[0], C, and B memory region attributes.

Usage constraints The PRRR is:

- Only accessible from PL1 or higher.
- Not accessible when the Long-descriptor translation table format is in use. See, instead, *MAIR0 and MAIR1, Memory Attribute Indirection Registers 0 and 1* on page 4-56.

Configurations The PRRR:

- Is Banked.
- Has write access to the Secure copy of the register disabled when the **CP15SDISABLE** signal is asserted HIGH.

Attributes See the register summary in Table 4-11 on page 4-12.

Figure 4-24 shows the PRRR bit assignments.

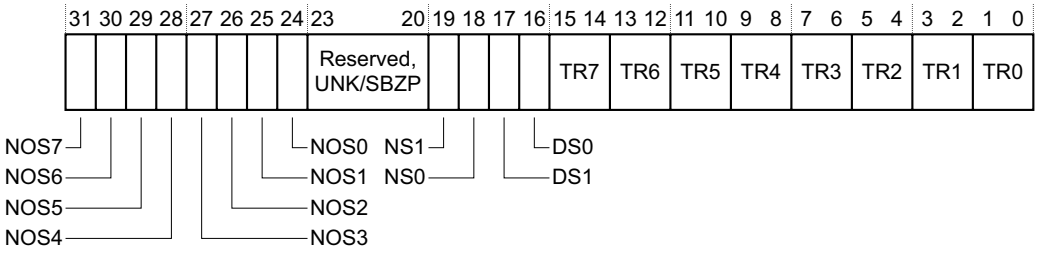


Figure 4-24 PRRR bit assignments

Table 4-53 shows the PRRR bit assignments.

Table 4-53 PRRR bit assignments

Bits	Name	Function
[24+*n*][a]	NOS*n*	Outer Shareable property mapping for memory attributes *n*, if the region is mapped as Normal Shareable. *n* is the value of the TEX[0], C and B bits, see Table 4-54 on page 4-56. The possible values of each NOS*n* bit are: **0** Memory region is Outer Shareable. **1** Memory region is Inner Shareable. See Table 5-2 on page 5-4 for more information about this field.
[23:20]	-	Reserved, UNK/SBZP.
[19]	NS1	Mapping of S = 1 attribute for Normal memory. This bit gives the mapped Shareable attribute for a region of memory that: • Is mapped as Normal memory. • Has the S bit set to 1. The possible values of the bit are: **0** Region is not Shareable. **1** Region is Shareable.
[18]	NS0	Mapping of S = 0 attribute for Normal memory. This bit gives the mapped Shareable attribute for a region of memory that: • Is mapped as Normal memory. • Has the S bit set to 0. The possible values of the bit are the same as those given for the NS1 bit, bit[19].
[17]	DS1	Mapping of S = 1 attribute for Device memory. This bit gives the mapped Shareable attribute for a region of memory that: • Is mapped as Device memory. • Has the S bit set to 1. **Note** This field has no significance in the processor.
[16]	DS0	Mapping of S = 0 attribute for Device memory. This bit gives the mapped Shareable attribute for a region of memory that: • Is mapped as Device memory. • Has the S bit set to 0. **Note** This field has no significance in the processor.
[2*n*+1:2*n*][a]	TR*n*	Primary TEX mapping for memory attributes *n*. *n* is the value of the TEX[0], C and B bits, see Table 4-54 on page 4-56. This field defines the mapped memory type for a region with attributes *n*. The possible values of the field are: `0b00` Strongly-ordered. `0b01` Device. `0b10` Normal Memory. `0b11` Reserved, effect is UNPREDICTABLE. See Table 5-2 on page 5-4 for more information about this field.

a. Where n is 0-7

Table 4-54 shows the mapping between the memory region attributes and the *n* value used in the PRRR.nOS*n* and PRRR.TR*n* field descriptions.

Table 4-54 Memory attributes and the *n* value for the PRRR field descriptions

Attributes			*n* value
TEX[0]	C	B	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Large Physical Address Extension, address translations use Long-descriptor translation table formats and MAIR0 replaces the PRRR, and MAIR1 replaces the NMRR. For more information see *MAIR0 and MAIR1, Memory Attribute Indirection Registers 0 and 1*.

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

```
MRC p15, 0, <Rt>, c10, c2, 0    ; Read Primary Region Remap Register
MCR p15, 0, <Rt>, c10, c2, 0    ; Write Primary Region Remap Register
```

4.3.29 MAIR0 and MAIR1, Memory Attribute Indirection Registers 0 and 1

The MAIR0 and MAIR1 characteristics are:

Purpose To provide the memory attribute encodings corresponding to the possible AttrIndx values in a Long-descriptor format translation table entry for stage 1 translations.

Usage constraints MAIR0 and MAIR1 are:

- Only accessible from PL1 or higher and when using the Long-descriptor translation table format. When using the Short-descriptor format see, instead, *Primary Region Remap Register* on page 4-54 and *Normal Memory Remap Register* on page 4-58.
- The Secure copies of the registers give the values for memory accesses from Secure state.
- The Non-secure copies of the registers give the values for memory accesses from Non-secure modes other than Hyp mode.

Configurations MAIR0 and MAIR1:

- Are Banked
- Have write access to the Secure copy of the register disabled when the **CP15SDISABLE** signal is asserted HIGH.

Attributes See the register summary in Table 4-11 on page 4-12.

Figure 4-25 shows the MAIR0 and MAIR1 bit assignments.

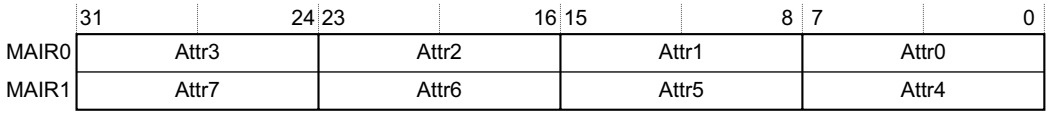


Figure 4-25 MAIR0 and MAIR1 bit assignments

Table 4-55 shows the MAIR0 and MAIR1 bit assignments.

Table 4-55 MAIR0 and MAIR1 bit assignments

Bits	Name	Description
[7:0]	Attr*m*[a]	The memory attribute encoding for an AttrIndx[2:0] entry in a Long descriptor format translation table entry, where: • AttrIndx[2] selects the appropriate MAIR: — Setting AttrIndx[2] to 0 selects MAIR0. — Setting AttrIndx[2] to 1 selects MAIR1. • AttrIndx[2:0] gives the value of *m* in Attr*m*.

a. Where m is 0-7

Table 4-56 shows the MAIRn.Attrm[7:4] encoding.

Table 4-56 MAIR*n*.Attr*m*[7:4] encoding

Attr*m*[7:4][ab]	Meaning
0000	Strongly-ordered or Device memory, see encoding of Attr*m*[3:0].
00RW, RW not 00	Normal memory, Outer Write-Through Cacheable[c]
0100	Normal memory, Outer[d] Non-cacheable[c]
01RW, RW not 00	Normal memory, Outer Write-Back Cacheable[c]
10RW	Normal memory, Outer[d] Write-Through Cacheable[c]
11RW	Normal memory, Outer[d] Write-Back Cacheable[bc]

a. Where *m* is 0-7.

b. R defines the Outer Read-Allocate policy, and W defined the Outer Write-Allocate policy, see Table 4-58 on page 4-58.

c. Transient attribute is ignored. See the *ARM Architecture reference Manual* for more information.

d. See encoding of Attr*m*[3:0], shown in Table 4-57, for Inner cacheability policies.

Table 4-57 shows the MAIRn.Attrm[7:4] encoding The encoding of Attr*n*[3:0] depends on the value of Attr*n*[7:4], as Table 4-57 shows.

Table 4-57 MAIR*n*.Attr*m*[3:0] encoding

Attr*m*[3:0][ab]	Meaning when Attr*m*[7:4] is `0b0000`	Meaning when Attr*m*[7:4] is not `0b0000`
0000	Strongly-ordered memory	UNPREDICTABLE.
00RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Cacheable[c]
0100	Device memory	Normal memory, Inner Non-cacheable[c]

Table 4-57 MAIR*n*.Attr*m*[3:0] encoding (continued)

Attr*m*[3:0][ab]	Meaning when Attr*m*[7:4] is 0b0000	Meaning when Attr*m*[7:4] is not 0b0000
01RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Cacheable[c]
10RW	UNPREDICTABLE	Normal memory, Inner Write-Through Cacheable[c]
11RW	UNPREDICTABLE	Normal memory, Inner Write-Back Cacheable[c]

a. Where *m* is 0-7.

b. R defines the Inner Read-Allocate policy, and W defines the Inner Write-Allocate policy, see Table 4-58.

c. Transient attribute is ignored. See the *ARM Architecture reference Manual* for more information.

Table 4-58 shows the encoding of the R and W bits that are used, in some Attr*m* encodings in Table 4-56 on page 4-57 and Table 4-57 on page 4-57, to define the read-allocate and write-allocate policies:

Table 4-58 Encoding of R and W bits in some Attr*m* fields

R or W	Meaning
0	Do not allocate
1	Allocate

To access MAIR0 read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c10, c2, 0    ; Read Memory Attribute Indirection Register 0
MCR p15, 0, <Rt>, c10, c2, 0    ; Write Memory Attribute Indirection Register 0
```

To access MAIR1, read or write the CP15 register with:

```
MRC p15, 0, <Rt>, c10, c2, 1    ; Read Memory Attribute Indirection Register 1
MCR p15, 0, <Rt>, c10, c2, 1    ; Write Memory Attribute Indirection Register 1
```

4.3.30 Normal Memory Remap Register

The NMRR characteristics are:

Purpose Provides additional mapping controls for memory regions that are mapped as Normal memory by their entry in the PRRR.

Usage constraints The NMRR is:

- Only accessible from PL1 or higher.
- Used in conjunction with the PRRR.
- Not accessible when the Long-descriptor translation table format is in use. See, instead, *MAIR0 and MAIR1, Memory Attribute Indirection Registers 0 and 1* on page 4-56.

Configurations The NMRR:

- Is Banked.
- Has write access to the Secure copy of the register disabled when the **CP15SDISABLE** signal is asserted HIGH.

Attributes See the register summary in Table 4-11 on page 4-12.

Figure 4-26 on page 4-59 shows the NMRR bit assignments.

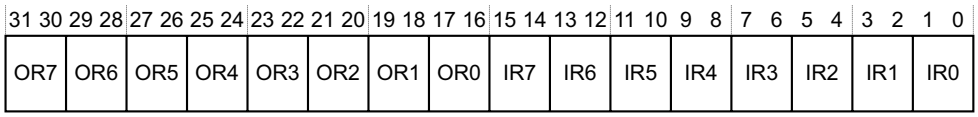


Figure 4-26 NMRR bit assignments

Table 4-59 shows the NMRR bit assignments.

Table 4-59 NMRR bit assignments

Bits	Name	Description
[2n+17:2n+16][a]	ORn	Outer Cacheable property mapping for memory attributes *n*, if the region is mapped as Normal memory by the PRRR.TR*n* entry. *n* is the value of the TEX[0], C and B bits, see Table 4-54 on page 4-56. The possible values of this field are: `0b00` Region is Non-cacheable. `0b01` Region is Write-Back, Write-Allocate. `0b10` Region is Write-Through, no Write-Allocate. `0b11` Region is Write-Back, no Write-Allocate. See Table 5-2 on page 5-4 for more information about this field.
[2n+1:2n][a]	IRn	Inner Cacheable property mapping for memory attributes *n*, if the region is mapped as Normal Memory by the PRRR.TR*n* entry. *n* is the value of the TEX[0], C and B bits, see Table 4-54 on page 4-56. The possible values of this field are the same as those given for the OR*n* field. See Table 5-2 on page 5-4 for more information about this field.

a. Where n is 0-7

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

```
MRC p15, 0, <Rt>, c10, c2, 1    ; Read Normal Memory Remap Register
MCR p15, 0, <Rt>, c10, c2, 1    ; Write Normal Memory Remap Register
```

4.3.31 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, with access rights that depend on the mode:
  - Read/write in Secure PL1 modes.
  - Read-only and write-ignored in Non-secure PL1 and PL2 modes if NSACR.NS_SMP is 0.
  - Read/write in Non-secure PL1 and PL2 modes if NSACR.NS_SMP is 1. In this case, all bits are write-ignored except for the SMP bit.

Configurations Available in all configurations.

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

Figure 4-27 shows the ACTLR bit assignments.

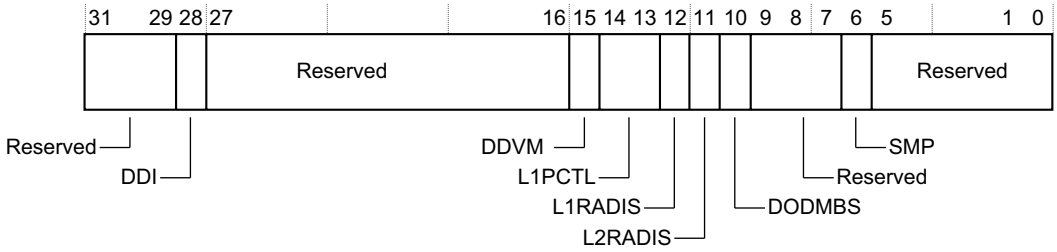


Figure 4-27 ACTLR bit assignments

Table 4-60 shows the ACTLR bit assignments.

Table 4-60 ACTLR bit assignments

Bits	Name	Function
[31:29]	-	Reserved, RAZ/WI.
[28]	DDI	Disable dual issue: **0** Enables dual issue, this is the reset value. **1** Disables dual issue.
[27:16]	-	Reserved, RAZ/WI.
[15]	DDVM	Disable *Distributed Virtual Memory* (DVM) transactions: **0** Enables DVM, this is the reset value. **1** Disables DVM.
[14:13]	L1PCTL	L1 Data prefetch control. The value of this field determines the maximum number of outstanding data prefetches permitted in the L1 memory system, not counting those generated by software load or PLD instructions: `0b00` Prefetch disabled. `0b01` 1 outstanding pre-fetch permitted. `0b10` 2 outstanding pre-fetches permitted. `0b11` 3 outstanding pre-fetches permitted, this is the reset value.
[12]	L1RADIS	L1 Data Cache read-allocate mode disable[a]: **0** Enables L1 data cache read-allocate mode, this is the reset value. **1** Disables L1 data cache read-allocate mode.
[11]	L2RADIS	L2 Data Cache read-allocate mode disable[a]: **0** Enables L2 data cache read-allocate mode, this is the reset value. **1** Disables L2 data cache read-allocate mode.
[10]	DODMBS	Disable optimized data memory barrier behavior: **0** Enables optimized data memory barrier behavior, this is the reset value. **1** Disables optimized data memory barrier.

Table 4-60 ACTLR bit assignments (continued)

Bits	Name	Function
[9:7]	-	Reserved, RAZ/WI.
[6]	SMP	Enables coherent requests to the processor: **0** Disables coherent requests to the processor. This is the reset value. **1** Enables coherent requests to the processor. When coherent requests are disabled: • loads to cacheable memory are not cached by the processor. • Load-Exclusive instructions take a precise abort if the memory attributes are: — Inner Write-Back and Outer Shareable. — Inner Write-Through and Outer Shareable. — Outer Write-Back and Outer Shareable. — Outer Write-Through and Outer Shareable. — Inner Write-Back and Inner Shareable. — Inner Write-Through and Inner Shareable. — Outer Write-Back and Inner Shareable. — Outer Write-Back and Inner Shareable. **Note** 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 power-down sequence. See *Power management* on page 2-12.
[5:0]	-	Reserved, RAZ/WI.

a. See *Data Cache Unit* on page 2-4 for more information.

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

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

4.3.32 Coprocessor Access Control Register

The CPACR characteristics are:

Purpose Controls access to coprocessors CP10 and CP11.

Usage constraints The CPACR is:
- A read/write register.
- Common to the Secure and Non-secure states.
- Only accessible from PL1 or higher.
- Has no effect on instructions executed in Hyp mode.

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

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

Figure 4-28 on page 4-62 shows the CPACR bit assignments.

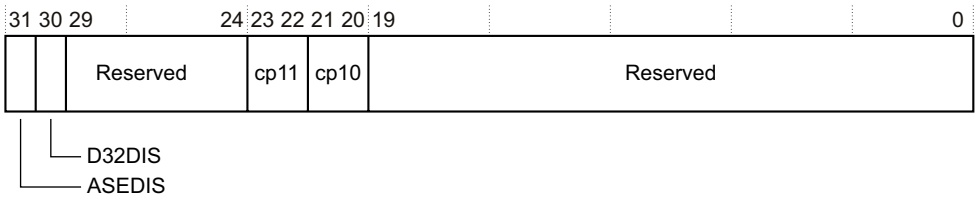


Figure 4-28 CPACR bit assignments

Table 4-61 shows the CPACR bit assignments.

Table 4-61 CPACR bit assignments

Bits	Name	Function
[31]	ASEDIS	Disable Advanced SIMD Functionality: **0** All Advanced SIMD and VFP instructions execute normally. **1** All Advanced SIMD instructions executed take an Undefined instruction exception. See the *Cortex-A7 MPCore Floating-Point Unit Technical Reference Manual* and *Cortex-A7 MPCore NEON Media Processing Engine Technical Reference Manual* for more information. If FPU is implemented and Advanced SIMD is not implemented, this bit is RAO/WI. If FPU and Advanced SIMD are not implemented, this bit is UNK/SBZP.
[30]	D32DIS	Disable use of registers D16-D31 of the VFP register file: **0** All instructions accessing D0-D31 execute normally. **1** Any VFP instruction that attempts to access any of registers D16-D31 is UNDEFINED. See the *Cortex-A7 MPCore Floating-Point Unit Technical Reference Manual* and *Cortex-A7 MPCore NEON Media Processing Engine Technical Reference Manual* for more information. If FPU is implemented and Advanced SIMD is implemented, ARM deprecates writing a value that is not zero to this bit. If FPU is implemented and Advanced SIMD is not implemented, this bit is RAO/WI. If FPU and Advanced SIMD are not implemented, this bit is UNK/SBZP.
[29:24]	-	Reserved, RAZ/WI.
[23:22]	cp11	Defines the access rights for coprocessor 11: `0b00` Access denied. Attempted accesses generate an Undefined Instruction exception. This is the reset value. `0b01` Access at PL1 or higher only. Attempted accesses in User mode generate an Undefined Instruction exception. Any attempt to access the coprocessor from software executing at PL0 generates an Undefined Instruction exception. `0b10` Reserved. The effect of this value is UNPREDICTABLE. `0b11` Full access. If FPU and Advanced SIMD are not implemented, this field is RAZ/WI.
[21:20]	cp10	Defines the access rights for coprocessor 10: `0b00` Access denied. Attempted accesses generate an Undefined Instruction exception. This is the reset value. `0b01` Access at PL1 or higher only. Any attempt to access the coprocessor from software executing at PL0 generates an Undefined Instruction exception. `0b10` Reserved. The effect of this value is UNPREDICTABLE. `0b11` Full access. If FPU and Advanced SIMD are not implemented, this field is RAZ/WI.
[19:0]	-	Reserved, RAZ/WI.

Note

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

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

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

4.3.33 Secure Configuration Register

The SCR characteristics are:

Purpose Defines the configuration of the current security state. It specifies:

- The security state of the processor, Secure or Non-secure.
- What mode the processor branches to, if an IRQ, FIQ or external abort occurs.
- Whether the CPSR.F and CPSR.A bits can be modified when SCR.NS = 1.

Usage constraints The SCR is:

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

Configurations Available in all configurations.

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

Figure 4-29 shows the SCR bit assignments.

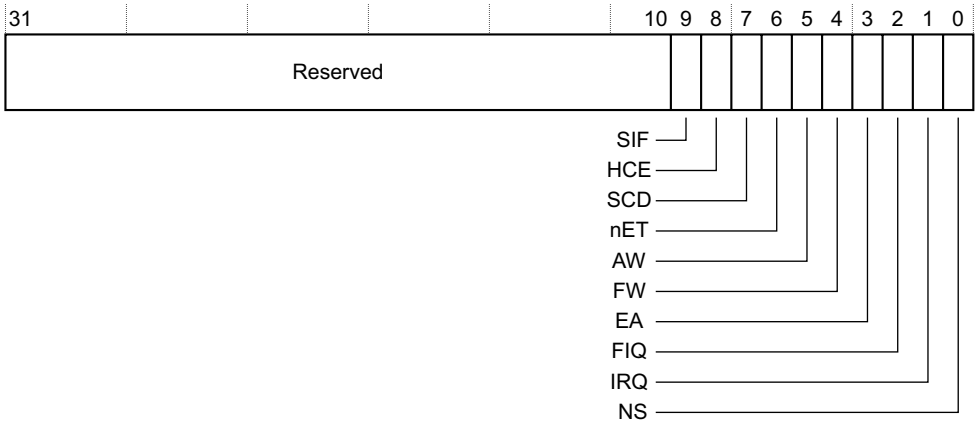


Figure 4-29 SCR bit assignments

Table 4-62 shows the SCR bit assignments.

Table 4-62 SCR bit assignments

Bits	Name	Function
[31:10]	-	Reserved, UNK/SBZP.
[9]	SIF	Secure Instruction Fetch bit: **0** Secure state instruction fetches from Non-secure memory are permitted, this is the reset value. **1** Secure state instruction fetches from Non-secure memory are not permitted.
[8]	HCE	Hyp Call enable. This bit enables the use of HVC instruction from Non-secure PL1 modes: **0** The HVC instruction is UNDEFINED in Non-secure PL1 mode, this is the reset value. **1** 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: **0** The SMC instruction executes normally in Non-Secure state, and performs a Secure Monitor Call, this is the reset value. **1** The SMC instruction is UNDEFINED in Non-secure state.
[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 the A bit in the CPSR can be modified in Non-secure state: **0** The CPSR.A bit can be modified only in Secure state, this is the reset value. **1** The CPSR.A bit can be modified in any security state.
[4]	FW	F bit writable. This bit controls whether the F bit in the CPSR can be modified in Non-secure state: **0** The CPSR.F bit can be modified only in Secure state, this is the reset value. **1** The CPSR.F bit can be modified in any security state.
[3]	EA	External Abort handler. This bit controls which mode takes external aborts: **0** External aborts taken in Abort mode, this is the reset value. **1** External aborts taken in Monitor mode.
[2]	FIQ	FIQ handler. This bit controls which mode takes FIQ exceptions: **0** FIQs taken in FIQ mode, this is the reset value. **1** FIQs taken in Monitor mode.
[1]	IRQ	IRQ handler. This bit controls which mode takes IRQ exceptions: **0** IRQs taken in IRQ mode, this is the reset value. **1** IRQs taken in Monitor mode.
[0]	NS	Non Secure bit. Except when the processor is in Monitor mode, this bit determines the security state of the processor. **0** Secure, this is the reset value. **1** Non-secure. **Note** When the processor is in Monitor mode, it is always in Secure state, regardless of the value of the NS bit. The value of the NS bit also affects the accessibility of the Banked CP15 registers in Monitor mode. See the *ARM Architecture Reference Manual* for more information on the NS bit.

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

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

4.3.34 Non-Secure Access Control Register

The NSACR characteristics are:

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

- ACTLR.SMP, see *Auxiliary Control Register* on page 4-59.
- L2ECTLR.AXI asynchronous error, see *L2 Extended Control Register* on page 4-82.
- CPACR.ASEDIS, see *Coprocessor Access Control Register* on page 4-61.

Usage constraints The NSACR is only accessible from PL1 or higher, with access rights that depend on the mode and security state:

- Read/write in Secure PL1 modes.
- Read-only in Non-secure PL1 and PL2 modes.

Configurations Available in all configurations.

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

Figure 4-30 shows the NSACR bit assignments.

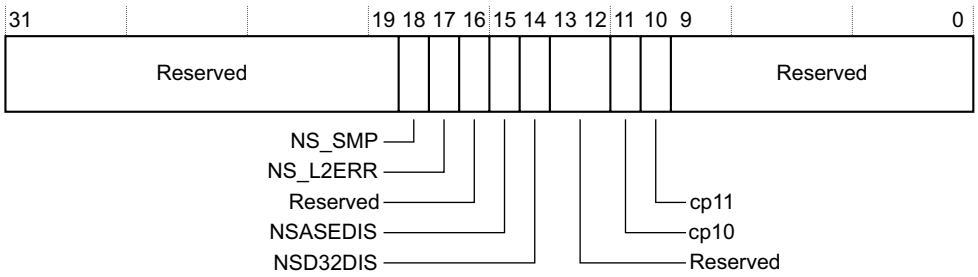


Figure 4-30 NSACR bit assignments

Table 4-63 shows the NSACR bit assignments.

Table 4-63 NSACR bit assignments

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

Table 4-63 NSACR bit assignments (continued)

Bits	Name	Function
[15]	NSASEDIS	Disable Non-secure Advanced SIMD functionality: **0** This bit has no effect on the ability to write CPACR.ASEDIS, this is the reset value. **1** When executing in Non-secure state, the CPACR.ASEDIS bit has a fixed value of 1 and writes to it are ignored. If FPU is implemented and Advanced SIMD is not implemented, this bit is RAO/WI. If FPU and Advanced SIMD are not implemented, this bit is UNK/SBZP.
[14]	NSD32DIS	Disable the Non-secure use of D16-D31 of the VFP register file: **0** This bit has no effect on the ability to write CPACR. D32DIS. This is the reset value. **1** The CPACR.D32DIS bit when executing in Non-secure state has a fixed value of 1 and writes to it are ignored. If FPU is implemented and Advanced SIMD is implemented, ARM deprecates writing a value that is not zero to this bit. If FPU is implemented and Advanced SIMD is not implemented, this bit is RAO/WI. If FPU and Advanced SIMD are not implemented, this bit is UNK/SBZP.
[13:12]	-	Reserved, RAZ/WI.
[11]	cp11	Non-secure access to coprocessor 11 enable: **0** Secure access only. Any attempt to access coprocessor 11 in Non-secure state results in an Undefined Instruction exception. If the processor is in Non-secure state, the corresponding bits in the CPACR ignore writes and read as `0b00`, access denied. This is the reset value. **1** Secure or Non-secure access. If FPU and Advanced SIMD are not implemented, this bit is RAZ/WI.
[10]	cp10	Non-secure access to coprocessor 10 enable: **0** Secure access only. Any attempt to access coprocessor 10 in Non-secure state results in an Undefined Instruction exception. If the processor is in Non-secure state, the corresponding bits in the CPACR ignore writes and read as `0b00`, access denied. This is the reset value. **1** Secure or Non-secure access. If FPU and Advanced SIMD 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
```

4.3.35 Hyp System Control Register

The HSCTLR characteristics are:

Purpose Provides the top level control of the system operation in Hyp mode. In Hype 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 4-5.

Figure 4-31 shows the HSCTLR bit assignments.

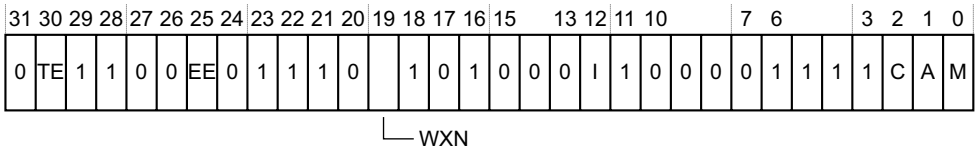


Figure 4-31 HSCTLR bit assignments

Table 4-64 shows the HSCTLR bit assignments.

Table 4-64 HSCTLR bit assignments

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

Table 4-64 HSCTLR bit assignments (continued)

Bits	Name	Function
[11]	-	Reserved, RAO/WI.
[10:7]	-	Reserved, RAZ/WI.
[6:3]	-	Reserved, RAO/WI.
[2]	C	Data and unified cache enable bit for memory accesses made in Hyp mode: **0** Data and unified caches disabled. **1** Data and unified caches enabled.
[1]	A	Alignment fault checking enable bit for memory accesses made in Hyp mode: **0** Alignment fault checking disabled. **1** Alignment fault checking enabled.
[0]	M	MMU stage 1 address trasnlation enable bit for memory accesses made in Hyp mode: **0** Address translation disabled. **1** 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
```

4.3.36 Hyp Auxiliary Configuration 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.

4.3.37 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 4-5.

Figure 4-32 on page 4-69 shows the HDCR bit assignments.

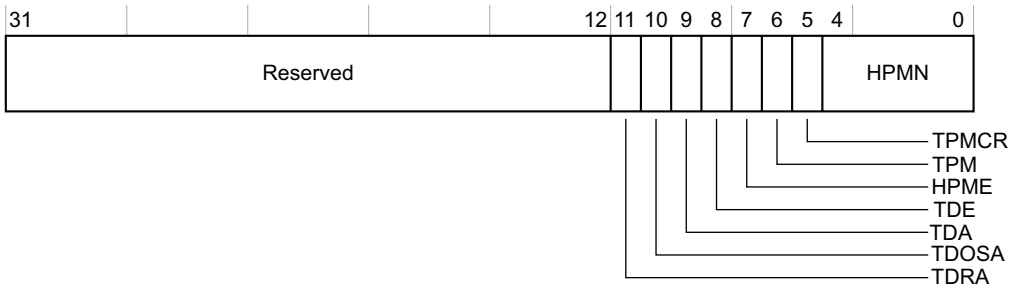


Figure 4-32 HDCR bit assignments

Table 4-65 shows the HDCR bit assignments.

Table 4-65 HDCR bit assignments

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

Table 4-65 HDCR bit assignments (continued)

Bits	Name	Function
[7]	HPME	Hypervisor Performance Monitor Enable: **0** Hyp mode performance monitor counters disabled. **1** Hyp mode performance monitor counters enabled. When this bit is set to 1, access to the performance monitors that are reserved for use from Hyp mode is enabled. For more information, see the description of the HPMN field. The reset value of this bit is UNKNOWN.
[6]	TPM	Trap Performance Monitor accesses: **0** Has no effect on performance monitor accesses. **1** Trap valid Non-secure performance monitor accesses to Hyp mode. When this bit is set to 1, any valid Non-secure access to the Performance Monitor registers is trapped to Hyp mode. This bit resets to 0. See the *ARM Architecture Reference Manual* for more information.
[5]	TPMCR	Trap Performance Monitor Control Register accesses: **0** Has no effect on PMCR accesses. **1** Trap valid Non-secure PMCR accesses to Hyp mode. When this bit is set to 1, any valid Non-secure access to the PMCR is trapped to Hyp mode. This bit resets to 0. See the *ARM Architecture Reference Manual* for more information.
[4:0]	HPMN	Hyp Performance Monitor count. Specifies the number of performance monitor counters that are accessible from Non-secure PL1 modes. In Non-secure state, HPMN divides the performance monitor counters. For example, If PMnEVCNTR is performance monitor counter *n* then, in Non-secure state: • If *n* is in the range $0 \le n <$ HPMN, the counter is accessible from PL1 and PL2, and from PL0 if unprivileged access to the counters is enabled. • If *n* is in the range HPMN $\le n <$PMCR.N, the counter is accessible only from PL2. The HPME bit enables access to the counters in this range. Behavior of the Performance Monitors counters is UNPREDICTABLE if this field is set to a value greater than PMCR.N. This field resets to `0x4`, which is the value of PMCR.N.

a. The permitted values of the HDCR.{TDRA, TDOSA, TDA, TDE} bits are `0b0000`, `0b0100`, `0b1000`, `0b1100`, `0b1110`, and `0b1111`. If these bits are set to any other values, behavior is UNPREDICTABLE.

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
```

4.3.38 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 4-61

- Are affected by settings in the NSACR. See *Non-Secure Access Control Register* on page 4-65. 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 4-5.

Figure 4-33 shows the HCPTR bit assignments.

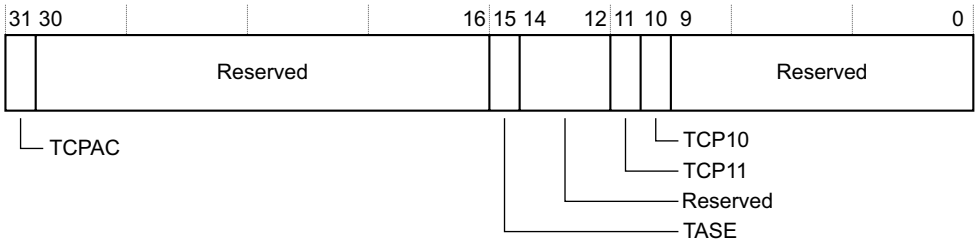


Figure 4-33 HCPTR bit assignments

Table 4-66 shows the HCPTR bit assignments.

Table 4-66 HCPTR bit assignments

Bits	Name	Function
[31]	TCPAC	Trap Coprocessor Access Control Register accesses: **0** Has no effect on CPACR accesses. **1** Trap valid Non-secure PL1 CPACR accesses to Hyp mode. When this bit is set to 1, any valid Non-secure PL1 or PL0 access to the CPACR is trapped to Hyp mode. See the *ARM Architecture Reference Manual* for more information.
[30:16]	-	Reserved, UNK/SBZP.
[15]	TASE	Trap Advanced SIMD Extension: **0** If the NSACR settings permit Non-secure use of the Advanced SIMD functionality then Hyp mode can access that functionality, regardless of any settings in the CPACR **Note** This bit value has no effect on possible use of the Advanced SIMD functionality from Non-secure PL1 and PL0 modes. **1** Trap valid Non-secure accesses to Advanced SIMD functionality to Hyp mode. When this bit is set to 1, any otherwise-valid access to Advanced SIMD functionality from: • A Non-secure PL1 or PL0 mode is trapped to Hyp mode. • Hyp mode generates an Undefined Instruction exception, taken in Hyp mode. If FPU is implemented and Advanced SIMD is not implemented, this bit is RAO/WI. If FPU and Advanced SIMD 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 4-66 HCPTR bit assignments (continued)

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

4.3.39 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.

4.3.40 Hyp Translation Control Register

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

4.3.41 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 4-75.

DFSR when using the Short-descriptor translation table format

Figure 4-34 shows the DFSR bit assignments when using the Short-descriptor translation table format.

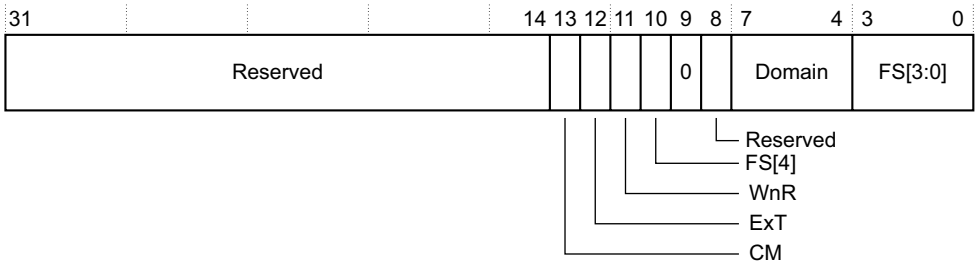


Figure 4-34 DFSR bit assignments for Short-descriptor translation table format

Table 4-67 shows the DFSR bit assignments when using the Short-descriptor translation table format.

Table 4-67 DFSR bit assignments for Short-descriptor translation table format

Bits	Name	Function
[31:14]	-	Reserved, UNK/SBZP.
[13]	CM	Cache maintenance fault. For synchronous faults, this bit indicates whether a cache maintenance operation generated the fault: **0** Abort not caused by a cache maintenance operation. **1** Abort caused by a cache maintenance operation.
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: **0** External abort marked as DECERR. **1** External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11]	WnR	Write not Read bit. This field indicates whether the abort was caused by a write or a read access: **0** Abort caused by a read access. **1** Abort caused by a write access. For faults on CP15 cache maintenance operations, including the VA to PA translation operations, this bit always returns a value of 1.
[10]	FS[4]	Part of the Fault Status field. See bits [3:0] in this table.

Table 4-67 DFSR bit assignments for Short-descriptor translation table format (continued)

Bits	Name	Function
[9]	-	RAZ.
[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: `0b00001` Alignment fault. `0b00010` Debug event. `0b00011` Access flag fault, section. `0b00100` Instruction cache maintenance fault. `0b00101` Translation fault, section. `0b00110` Access flag fault, page. `0b00111` Translation fault, page. `0b01000` Synchronous external abort, non-translation. `0b01001` Domain fault, section. `0b01011` Domain fault, page. `0b01100` Synchronous external abort on translation table walk, 1st level. `0b01101` Permission fault, section. `0b01110` Synchronous external abort on translation table walk, 2nd level. `0b01111` Permission fault, page. `0b10110` Asynchronous external abort.

DFSR when using the Long-descriptor translation table format

Figure 4-35 shows the DFSR bit assignments when using the Long-descriptor translation table format.

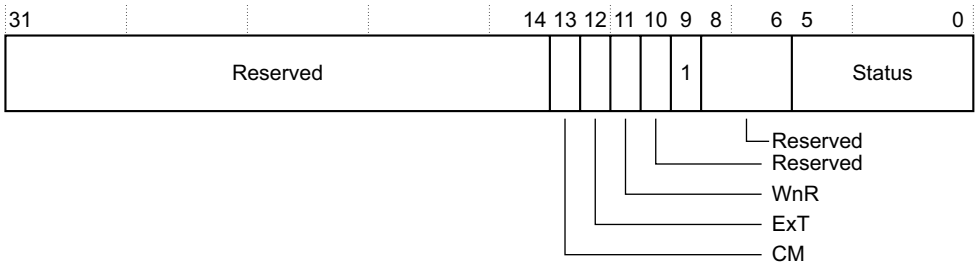


Figure 4-35 DFSR bit assignments for Long-descriptor translation table format

Table 4-68 shows the DFSR bit assignments when using the Long-descriptor translation table format.

Table 4-68 DFSR bit assignments for Long-descriptor translation table format

Bits	Name	Function
[31:14]	-	Reserved, UNK/SBZP.
[13]	CM	Cache maintenance fault. For synchronous faults, this bit indicates whether a cache maintenance operation generated the fault: **0** Abort not caused by a cache maintenance operation. **1** Abort caused by a cache maintenance operation.
[12]	ExT	External abort type. This field indicates whether an AXI Decode or Slave error caused an abort: **0** External abort marked as DECERR. **1** External abort marked as SLVERR. For aborts other than external aborts this bit always returns 0.
[11]	WnR	Write not Read bit. This field indicates whether the abort was caused by a write or a read access: **0** Abort caused by a read access. **1** Abort caused by a write access. For faults on CP15 cache maintenance operations, including the VA to PA translation operations, this bit always returns a value of 1.
[10]	-	Reserved, UNK/SBZP.
[9]	-	RAO.
[8:6]	-	Reserved, UNK/SBZP.
[5:0]	Status	Fault Status bits. This field indicates the type of exception generated. Any encoding not listed is reserved. `0b0001LL` Translation fault, LL bits indicate level. `0b0010LL` Access fault flag, LL bits indicate level. `0b0011LL` Permission fault, LL bits indicate level. `0b010000` Synchronous external abort. `0b010001` Asynchronous external abort. `0b0101LL` Synchronous external abort on translation table walk, LL bits indicate level. `0b100001` Alignment fault. `0b100010` Debug event.

Table 4-69 shows how the LL bits in the Status field encode the lookup level associated with the MMU fault.

Table 4-69 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
```

4.3.42 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 4-78.

IFSR when using the Short-descriptor translation table format

Figure 4-36 shows the IFSR bit assignments when using the Short-descriptor translation table format.

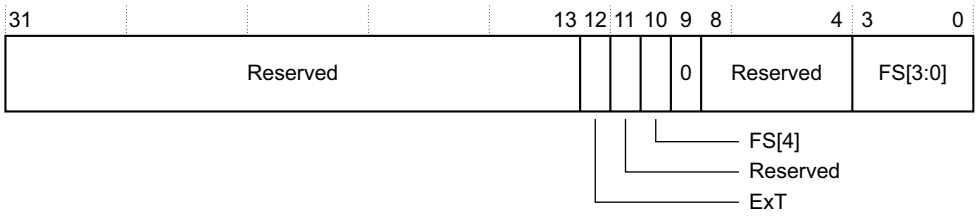


Figure 4-36 IFSR bit assignments for Short-descriptor translation table format

Table 4-70 shows the IFSR bit assignments when using the Short-descriptor translation table format.

Table 4-70 IFSR bit assignments for Long-descriptor translation table format

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

Table 4-70 IFSR bit assignments for Long-descriptor translation table format (continued)

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

IFSR when using the Long-descriptor translation table format

Figure 4-37 shows the IFSR bit assignments when using the Long-descriptor translation table format.

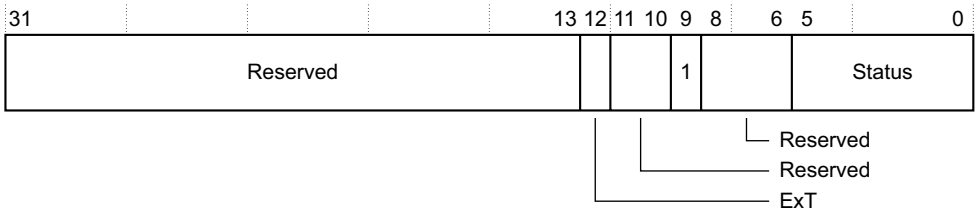


Figure 4-37 IFSR bit assignments for Long-descriptor translation table format

Table 4-71 shows the IFSR bit assignments when using the Long-descriptor translation table format.

Table 4-71 IFSR bit assignments for Long-descriptor translation table format

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

Table 4-71 IFSR bit assignments for Long-descriptor translation table format (continued)

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

Table 4-72 shows how the LL bits in the Status field encode the lookup level associated with the MMU fault.

Table 4-72 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
```

4.3.43 Auxiliary Data Fault Status Register

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

4.3.44 Auxiliary Instruction Fault Status Register

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

4.3.45 Hyp Auxiliary Data Fault Status Syndrome Register

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

4.3.46 Hyp Auxiliary Instruction Fault Status Syndrome Register

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

4.3.47 Hyp Syndrome Register

The HSR characteristics are:

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

Usage constraints The HSR is:

- A read/write register.
- Only accessible from Hyp mode or from Monitor mode when SCR.NS is 1.
- Unknown when executing in Non-secure modes other than Hyp mode.

Configurations Available in all configurations.

Attributes See the register summary in Table 4-26 on page 4-22.

Figure 4-38 shows the HSR bit assignments.

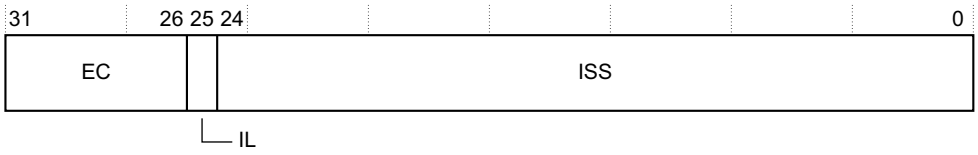


Figure 4-38 HSR bit assignments

Table 4-73 shows the HSR bit assignments.

Table 4-73 HSR bit assignments

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

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`.

4.3.48 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* for more information.

4.3.49 L2 Control Register

The L2CTLR characteristics are:

Purpose Defines the implemented options of the multiprocessor device.

Usage constraints The L2CTLR is:

- A read only register.
- Common to the 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.

Figure 4-39 shows the L2CTLR bit assignments.

Figure 4-39 L2CTLR bit assignments

Table 4-74 shows the L2CTLR bit assignments.

Table 4-74 L2CTLR bit assignments

Bits	Name	Function
[31:26]	-	Reserved, RAZ/WI.
[25:24]	Number of processors	Number of processors present: `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. These bits are read-only and the reset value of this field is set to the number of processors present in the configuration.
[23]	Interrupt controller	Interrupt controller: **0** Interrupt Controller not present. **1** Interrupt Controller present.
[22:1]		Reserved, RAZ/WI.
0]	Data RAM latency	L2 data RAM latency: **0** 2 cycles. **1** 3 cycles.

To access the L2CTLR, read the CP15 register with:

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

4.3.50 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. In this case, all bits are write-ignored except for bit[29].

Configurations Available in all configurations.

Attributes See the register summary in Table 4-10 on page 4-11.

Figure 4-40 on page 4-83 shows the L2ECTLR bit assignments.

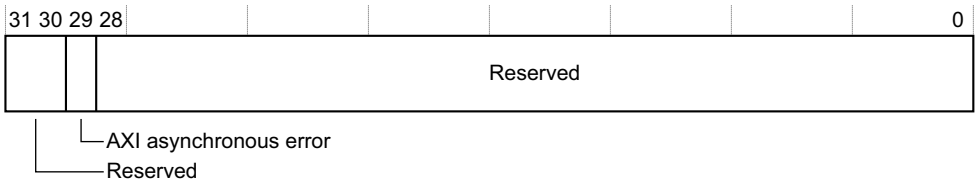


Figure 4-40 L2ECTLR bit assignments

Table 4-75 shows the L2ECTLR bit assignments.

Table 4-75 L2ECTLR bit assignments

Bits	Name	Function
[31:30]	-	Reserved, RAZ/WI.
[29]	AXI asynchronous error	AXI asynchronous error indication: **0** No pending AXI asynchronous error. This is the reset value. **1** AXI asynchronous error has occurred. A write of 0 clears this bit. A write of 1 is ignored.
[28:0]	-	Reserved, RAZ/WI.

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

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

4.3.51 Auxiliary Memory Attribute Indirection Register 0

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

4.3.52 Auxiliary Memory Attribute Indirection Register 1

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

4.3.53 Hyp Auxiliary Memory Attribute Indirection Register 0

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

4.3.54 Hyp Auxiliary Memory Attribute Indirection Register 1

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

4.3.55 FCSE Process ID Register

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

4.3.56 Configuration Base Address Register

The CBAR characteristics are:

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

Usage constraints The CBAR is:

- A read-only register.

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

Configurations Available in all configurations.

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

Figure 4-41 shows the CBAR bit assignments.

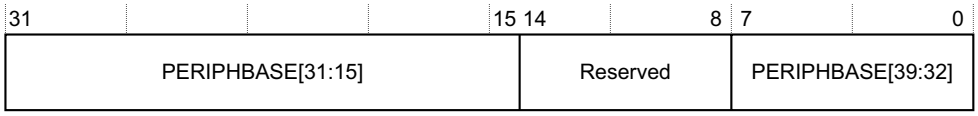


Figure 4-41 CBAR bit assignments

Table 4-76 shows the CBAR bit assignments.

Table 4-76 CBAR bit assignments

Bits	Name	Function
[31:15]	PERIPHBASE[31:15]	The primary input **PERIPHBASE[31:15]** determines the reset value
[11:8]	-	Reserved, UNK/SBZP
[7:0]	PERIPHBASE[39:32]	The primary input **PERIPHBASE[39:32]** determines the reset value

To access the CBAR, read the CP15 register with:

```
MRC p15, 4, <Rt>, c15, c0, 0; Read Configuration Base Address Register
```

Chapter 5
Memory Management Unit

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

5.1 About the MMU

The Cortex-A7 MPCore processor implements the Extended VMSAv7 MMU, which includes the ARMv7-A *Virtual Memory System Architecture* (VMSA), the Security Extensions, the *Large Physical Address Extensions* (LPAE), and the 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* for a full architectural description of the Extended VMSAv7.

The MMU controls table walk hardware that accesses translation tables in main memory. The MMU works with the L1 and L2 memory system to translate virtual addresses to physical addresses. The MMU enables fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes held in the *Translation Look-aside Buffers* (TLBs).

The MMU features in each processor of the multiprocessor device include the following:

- 10-entry fully-associative micro instruction TLB.
- 10-entry fully-associative micro data TLB.
- 2-way set-associative 256-entry unified main TLB.
- 2-way set-associative 64-entry walk cache.
- 2-way set-associative 64-entry IPA cache.
- the TLB entries include global and application specific identifiers to prevent context switch TLB flushes.
- *Virtual Machine Identifier* (VMID) to prevent TLB flushes on virtual machine switches by the hypervisor.

5.2 Memory management system

The Cortex-A7 MPCore processor supports the ARM v7 VMSA including the Security extensions and LPAE. The translation of a *Virtual Address* (VA) used by the instruction set architecture to a *Physical Address* (PA) used in the memory system and the management of the associated attributes and permissions is carried out using a two-level MMU.

The first level MMU uses a Harvard design with separate micro TLB structures in the PFU for instruction fetches (IuTLB) and in the DPU for data read and write requests (DuTLB).

A miss in the micro TLB results in a request to the main unified TLB shared between the data and instruction sides of the memory system. The TLB consists of a 256-entry 2-way set-associative RAM based structure. The TLB page-walk mechanism supports page descriptors held in the L1 data cache. These cached descriptors are coherent across the cores in the Cortex-A7 MPCore processor. The caching of page descriptors is configured globally for each translation table base register, TTBRx, in the system coprocessor, CP15.

5.2.1 Memory types and attributes

Although various different memory types can be specified in the page tables, the Cortex-A7 MPCore processor does not implement all possible combinations:

- Write-Through caches. Any memory marked as Inner Write-Through is not cached on the data side. On the instruction side areas marked as Inner Write-Through and Inner Write-Back are treated identically and both cached in the L1 instruction cache. However, instruction side areas marked as Inner Write-Through are not cached in the L2 cache when present. All memory marked as Inner Write-Through is reported as Inner Write-Through by the PAR even though it is not cached.
- 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 *Data Cache Unit* on page 2-4. All Inner Write-Back is reported as Write-Back Write-Allocate in the PAR.
- Non-cacheable. memory does not access L1 caches.

  Normal memory, that 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 See the *ARM Architecture Reference Manual* for more information.

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 SCTRL.TRE 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[a]
		1	Outer and Inner Write-Back, no Write-Allocate	Normal	S bit[a]

Table 5-1 TEX, C, and B encodings when SCTRL.TRE is set to 0 (continued)

TEX[2:0]	C	B	Description		Memory type	Page Shareable
001	0	0	Outer and Inner Non-cacheable		Normal	S bit[a]
		1	Reserved		-	-
	1	0[b]	Outer and Inner Write-Back, Write-Allocate		Normal	S bit[a]
		1				
010	0	0	Non-shareable Device		Device	Non-shareable
		1	Reserved		-	-
	1	x	Reserved		-	-
011	x	x	Reserved		-	-
1BB	A	A	Cacheability[c]	AA = Inner attribute BB = Outer attribute	Normal	S bit[a]

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

b. For architectural compatibility with other processors, ARM recommends that this encoding is not used.

c. See the *ARM Architecture Reference Manual* for more information on cacheability attributes.

Table 5-2 shows the C, B, and TEX[2:0] encodings for Short-descriptor format memory region with TEX remapping.

Table 5-2 TEX, C, and B encodings when SCTRL.TRE is set to 1

Encoding			Memory type[a]	Cache attributes[a, b]:		Outer Shareable attribute[a, c]
TEX[0]	C	B		Inner cache	Outer cache	
0	0	0	PRRR[1:0]	NMRR[1:0]	NMRR[17:16]	NOT(PRRR[24])
		1	PRRR[3:2]	NMRR[3:2]	NMRR[19:18]	NOT(PRRR[25])
	1	0	PRRR[5:4]	NMRR[5:4]	NMRR[21:20]	NOT(PRRR[26])
		1	PRRR[7:6]	NMRR[7:6]	NMRR[23:22]	NOT(PRRR[27])
1	0	0	PRRR[9:8]	NMRR[9:8]	NMRR[25:24]	NOT(PRRR[28])
		1	PRRR[11:10]	NMRR[11:10]	NMRR[27:26]	NOT(PRRR[29])
	1	0	PRRR[13:12]	NMRR[13:12]	NMRR[29:28]	NOT(PRRR[30])
		1	PRRR[15:14]	NMRR[15:14]	NMRR[31:30]	NOT(PRRR[31])

a. For details of the *Memory type* and *Outer Shareable* encodings see *Primary Region Remap Register* on page 4-54.

b. Applies only if the memory type for the region is mapped as Normal memory.

c. Applies only if the memory type for the region is mapped as Normal or Device memory and the region is Shareable.

5.3 TLB organization

TLB organization is described in the following sections:

- *Micro TLB*.
- *Main TLB*.

5.3.1 Micro TLB

The first level of caching for the page table information is a micro TLB of 10 entries that is implemented on each of the instruction and data sides. These blocks provide a lookup of the virtual addresses in a single cycle.

The micro TLB returns the physical address to the cache for the address comparison, and also checks the access permissions to signal either a Prefetch Abort or a Data Abort.

All main TLB related maintenance operations affect both the instruction and data micro TLBs, causing them to be flushed.

5.3.2 Main TLB

Misses from the micro TLBs are handled by a unified main TLB. This is a 256- entry 2-way set-associative structure. The main TLB supports all the VMSAv7 page sizes of 4KB, 64KB, 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.
- The TLB maintenance operations in flight.
- The different page size mappings in use.

5.3.3 IPA cache RAM

The IPA cache RAM holds mappings between intermediate physical addresses and physical addresses. Only translations performed in non-secure non-hypervisor modes use this. When a stage 2 translation is completed it is updated, and checked whenever a stage 2 translation is required.

Similarly to the main TLB, it can hold entries for different sizes, and maintains a hitmap of the sizes present so that it does not have to lookup for sizes that are known not to be present.

5.3.4 Walk cache RAM

The walk cache RAM holds the partial result of a stage 1 translation, which excludes the last level. If the stage 1 translation results in a section or larger mapping then nothing is placed in the walk cache.

The walk cache holds entries fetched from secure and non-secure state, including hypervisor mode.

5.4 TLB match process

The Virtualization Extensions and the Security Extensions provide for multiple virtual address spaces that are translated differently. The main TLB entries store all the required context information to facilitate a match and avoid the need for a TLB flush on a context or virtual machine switch. Each TLB entry contains a virtual address, page size, physical address, and a set of memory properties that include the memory type and access permissions. Each entry is marked as being associated with a particular *Application Space ID* (ASID), or as global for all application spaces. The TLB entry also contains a field to store the *Virtual Machine Identifier* (VMID) that brought in the entry, applicable to accesses made from the non-secure state, as defined by the Virtualization extensions. There is also a bit that records whether that TLB entry is allocated on a Hyp mode request. A TLB entry match occurs when the following conditions are met:

- Its virtual address, moderated by the page size such as the virtual address bits [31:N], where N is log2 of the page size for that translation stored in the TLB entry, matches that of the requested address.
- The Non-secure TLB ID, NSTID, matches the Secure or Non-secure state of the requests.
- The Hyp mode bit matches whether the request was made from Hyp mode.
- The ASID matches the current ASID held in the CONTEXTIDR, TTBR0, or TTBR1 register or the entry is marked global.
- The VMID matches the current VMID held in the VTTBR register.

Note

- For a request originating from Hyp mode, the ASID and VMID match are ignored.
- For a request originating from Secure state, the VMID match is ignored.

5.5 Memory access sequence

When the processor generates a memory access, the MMU:

1. Performs a lookup for the requested virtual address, current ASID, current VMID, and security state in the relevant instruction or data micro TLB.
2. Performs a lookup for the requested virtual address, 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* (TTBR). See the *ARM Architecture Reference Manual* for information on translation table formats.

Note

Translations in Hyp mode and Stage 2 translations are always performed with the Long-descriptor translation table format as specified by the LPAE.

You can configure the MMU to perform translation table walks in cacheable regions using:

- The Short-descriptor translation table format by setting the IRGN bits in the *Translation Table Base Register 0* (TTBR0) and *Translation Table Base Register 1* (TTBR1).
- The Long-descriptor translation table format by setting the IRGN bits in the relevant *Translation Table Base Control Register* (TTBCR).

If the encoding of the IRGN bits is Write-Back, an L1 data cache lookup is performed and data is read from the data cache. If the encoding of the IRGN bits is Write-Through or Non-cacheable, an access to external memory is performed.

In the case of 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 setting the PD0 or PD1 bit in the TTBCR to 0.
- Long-descriptor translation table format by setting the EPD0 or EPD1 bit in the TTBCR to 0.

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 enabled. If the matching entry does not pass the permission checks, the MMU signals a memory abort. See the *ARM Architecture Reference Manual* for a description of access permission bits, abort types and priorities, and for a description of the *Instruction Fault Status Register* (IFSR) and *Data Fault Status Register* (DFSR).
- The memory region attributes specified in the TLB entry determine if the access is:
  - Secure or Non-secure.
  - Inner, Outer Shared or not.
  - Normal Memory, Device, or Strongly-ordered.
- The TLB translates the virtual address to a physical address for the memory access.

5.6 MMU enabling and disabling

You can enable or disable the MMU. See the *ARM Architecture Reference Manual* for more information.

5.7 External aborts

External memory errors are defined as those that occur in the memory system rather than those that are detected by the MMU. External memory errors are expected to be extremely rare. External aborts are caused by errors flagged by the ACE master interface when the request goes external to the Cortex-A7 MPCore processor. You can configure External aborts to trap to Monitor mode by setting the SCR.EA bit to 1.

5.7.1 External aborts on data write

Externally generated errors during a data write can be asynchronous. This means that the r14_abt on entry into the abort handler on such an abort might not hold the address of the instruction that caused the exception.

External memory errors on cacheable writes do not cause an abort, but do cause the **nAXIERRIRQ** signal to be asserted.

The DFAR is UNPREDICTABLE when an asynchronous abort occurs.

5.7.2 External aborts on data read

Externally generated errors on data load instructions are always synchronous. The address captured in the DFAR matches the address that generated the external abort.

External memory errors on read channel accesses that do not cause an abort cause the **nAXIERRIRQ** signal to be asserted. This includes:

- L2 linefills triggered by an L1 instruction fetch where data is received from the interconnect in a dirty state. Instruction data can be marked as dirty as a result of self-modifying code or a line containing a mixture of data and instructions. If an error response is received on any part of the line, the dirty data might be lost.
- DVM operations.

5.7.3 Synchronous and asynchronous aborts

To determine a fault type, read the DFSR for a data abort or the IFSR for a Prefetch Abort.

While in debug state with DBGDSCR.ADAdiscard set to 1, the behavior of the **nAXIERRIRQ** signal is not affected.

5.8 MMU software accessible registers

The system control coprocessor registers, CP15, in conjunction with page table descriptors stored in memory, control the MMU as shown in Table 5-3.

You can access 32-bit registers with instructions of the form:

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

You can access 64-bit registers with instructions of the form:

```
MRCC p15, 0, <Rt>, <Rt2>, <CRn>
MCRR p15, 0, <R>, <CRn>, <CRm>
```

CRn is the system control coprocessor register. Unless specified otherwise, CRm and Opcode_2 Should Be Zero.

Table 5-3 CP15 register functions

Register	Cross reference
Context ID Register	See the *ARM Architecture Reference Manual*
Control Register	
Data Fault Address Register	
Data Fault Status Register	
Domain Access Control Register	
Hyp Auxiliary DFSR	
Hyp Auxiliary IFSR	
Hyp Data Fault Address Register	
Hyp Instruction Fault Address Register	
Hyp Instruction Fault Address Register	
Hyp Memory Attribute Indirection Register 0	
Hyp Memory Attribute Indirection Register 1	
Hyp Translation Control Register	
Hyp Translation Table Base Register	
Virtualization Translation Control Register	
Virtualization Translation Table Base Register	
Hyp Syndrome Register	
Instruction Fault Address Register	
Instruction Fault Status Register	
Non-secure Access Control Register	
Normal Memory Remap Register	
Primary Region Remap Register	
TLB operations	

Table 5-3 CP15 register functions (continued)

Register	Cross reference
TLB Type Register	See the *ARM Architecture Reference Manual*
Translation Table Base Control Register	
Translation Table Base Register 0	
Translation Table Base Register 1	

Chapter 6
L1 Memory System

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

6.1 About the L1 memory system

The L1 memory system consists of separate instruction and data caches. You can configure the instruction and data caches independently during implementation to sizes of 8KB, 16KB, 32KB, or 64KB.

The L1 memory system has a store buffer that has four 64-bit slots with data merging capability. It handles writes to Device, Strongly-ordered, Cacheable and Non-cacheable memory.

The L1 instruction memory system has the following features:

- Instruction side cache line length of 32-bytes.
- Virtually indexed and physically tagged instruction cache.
- Pseudo random cache replacement policy.
- 2-way set-associative instruction cache.
- Support for four sizes of memory page.
- Export of memory attributes for external memory systems.
- Support for Security Extensions.
- Can be disabled independently, using the system control coprocessor. See *System Control Register* on page 4-51
- On a cache miss, critical word first filling of the cache is performed

The L1 data memory system has the following features:

- Data side cache line length of 64-bytes.
- Physically indexed and physically tagged data cache.
- Pseudo random cache replacement policy.
- 4-way set-associative data cache.
- Two 32-byte linefill buffers and one 64-byte eviction buffer.
- A 4-entry, 64-bit merging store buffer.
- Can be disabled independently, using the system control coprocessor. See *System Control Register* on page 4-51.
- On a cache miss, critical word first filling of the cache is performed.

6.2 Cache behavior

You can disable each cache independently. See *System Control Register* on page 4-51. On a cache miss, critical word-first filling of the cache is performed.

If the cache reports a hit on a memory location that is marked as Non-Cacheable, Device, or Strongly-ordered, this is called an unexpected cache hit. In this architecturally unpredictable case, the cache might return incorrect data. Because the caches are physically addressed, improper page table configuration is the only way to create this scenario.

6.2.1 Instruction cache disabled behavior

The SCR.I bit, see *System Control Register* on page 4-51, enables or disables the L1 instruction cache. If the I bit is disabled, fetches cannot access any of the instruction cache arrays. An exception to this rule is the CP15 instruction cache operations. If the instruction cache is disabled, the instruction cache maintenance operations can still execute normally.

6.2.2 Instruction cache speculative memory accesses

An instruction remains in the pipeline between the fetch and the execute stages. Because there can be several unresolved branches in the pipeline, instruction fetches are speculative, meaning there is no guarantee that they are executed. A branch or exceptional instruction in the code stream can cause a pipeline flush, discarding the currently fetched instructions.

Because of the aggressive prefetching behavior, you must not place read-sensitive devices in the same page as code. Pages with Device or Strongly-ordered memory type attributes are treated as Non-Cacheable Normal Memory. You must mark pages that contain read-sensitive devices with the TLB XN (Execute Never) attribute bit.

To avoid speculative fetches to read-sensitive devices when address translation is disabled, these devices and code that is fetched must be separated in the physical memory map. See the *ARM Architecture Reference Manual* for more information.

6.3 L1 instruction memory system

The L1 instruction side memory system is responsible for providing an instruction stream to the Cortex-A7 MPCore processor. To increase overall performance and to reduce power consumption, it contains the following functionality:

- Dynamic branch prediction
- Instruction caching.

The instruction side comprises the following:

Prefetch Unit (PFU)

The PFU implements a 2-level prediction mechanism, comprising the following :

- A 256-entry branch pattern history table.
- A 4-entry BTIC.
- A 8-entry BTAC.
- A 8-entry return stack.

The prediction scheme is available in ARM state, Thumb state, and ThumbEE state. It is also capable of predicting state changes from ARM to Thumb, and from Thumb to ARM. It does not predict any other state changes, or any instruction that changes the mode of the processor. See *Program flow prediction*.

Instruction Cache Controller

The instruction cache controller fetches the instructions from memory depending on the program flow predicted by the PFU.

The instruction cache is 2-way set-associative. It comprises the following features:

- Configurable size of 8KB, 16KB, 32KB, or 64KB.
- *Virtually Indexed Physically Tagged* (VIPT).
- 64-bit native accesses to provide up to four instructions per cycle to the PFU.
- Security Extensions support.
- No lockdown support.

6.3.1 Enabling program flow prediction

Program flow prediction is always enabled when the MMU is enabled by setting the SCTLR.M bit to 1. See *System Control Register* on page 4-51.

6.3.2 Program flow prediction

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 and Thumb instructions. As a general rule, the flow prediction hardware predicts all branch instructions regardless of the addressing mode, including:

- Conditional branches.
- Unconditional branches.

- Indirect branches associated with function-call and return instructions.
- Branches that switch between ARM and Thumb states.

However, some branch instructions are not predicted:

- PC destination data processing operations.
- 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]`.

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}^` instruction, `RFE`, and the `MOVS pc, r14` instruction.

6.4 L1 data memory system

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

6.4.1 Internal exclusive monitor

The Cortex-A7 MPCore 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 16 words, one cache line.

A Load-Exclusive instruction that causes a transaction with ARLOCK[0] set to 1 is expected to receive an EXOKAY response. An OKAY response to a transaction with ARLOCK[0] set to 1 indicates that exclusive accesses are not supported at the address of the transaction and causes a precise abort to be taken. A Load-Exclusive instruction causes ARLOCK[0] to be set to 1 if the memory attributes are:

- Inner Non-cacheable and Outer Non-cacheable.
- Device or Strongly Ordered.
- Inner Write-Back and Outer Shareable and **BROADCASTOUTER** is set to 1.
- Inner Write-Through and Outer Shareable and **BROADCASTOUTER** is set to 1.
- Outer Write-Back and Outer Shareable and **BROADCASTOUTER** is set to 1.
- Outer Write-Through and Outer Shareable and **BROADCASTOUTER** is set to 1.
- Inner Write-Back and Inner Shareable and **BROADCASTINNER** is set to 1.
- Inner Write-Through and Inner Shareable and **BROADCASTINNER** is set to 1.
- Outer Write-Back and Inner Shareable and **BROADCASTINNER** is set to 1.
- Outer Write-Through and Inner Shareable and **BROADCASTINNER** is set to 1.

A Load-Exclusive instruction might also take a precise abort if ACTLR.SMP bit is clear. See Table 4-60 on page 4-60 for more information.

See the *ARM Architecture Reference Manual* for more information about these instructions.

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 the STREX.

However, if the address LDREX/STREX code sequence is in cacheable memory, any eviction of the cache line containing that address clears the monitor. It is therefore recommended that no load or store instructions are placed between the LDREX and STREX because these additional instructions can cause a cache eviction. Any data cache maintenance instruction can also clear the exclusive monitor.

6.4.2 ACE transactions

Table 6-1 shows the ACE transactions that each type of memory access generates .

Table 6-1 ACE transactions

Attributes		ACE transaction				
Memory type	**Shareability**	**Domain**	**Load**	**Store**	**Load exclusive**	**Store exclusive**
Strongly Ordered	-	System	ReadNoSnoop	WriteNoSnoop	ReadNoSnoop and **ARLOCK[0]** set to 1	WriteNoSnoop and **AWLOCK[0]** set to 1
Device	-					
Normal, inner Non-cacheable, outer Non-cacheable	Non-shared	System	ReadNoSnoop	WriteNoSnoop	ReadNoSnoop and **ARLOCK[0]** set to 1	WriteNoSnoop and **AWLOCK[0]** set to 1
	Inner-shared					
	Outer-shared					
Normal, inner Non-cacheable, outer Write-Back or Write-Through, or Normal, inner Write-Through, outer Write-Back, Write-Through or Non-cacheable	Non-shared	Non-shareable	ReadNoSnoop	WriteNoSnoop	ReadNoSnoop and **ARLOCK[0]** set to 0	WriteNoSnoop and **AWLOCK[0]** set to 0
	Inner-shared	Inner shareable	ReadOnce	CleanUnique followed by Write-Back	ReadClean with **ARLOCK[0]** set to 1[a]	CleanUnique with **ARLOCK[0]** set to 1 followed by Write-Back[a]
	Outer-shared	Outer shareable				
Normal, inner Write-Back, outer Non-cacheable, Write-Through, or Write-Back	Non-shared	Non-shareable	ReadNoSnoop	WriteNoSnoop	ReadNoSnoop and **ARLOCK[0]** set to 0	WriteNoSnoop and **AWLOCK[0]** set to 0
	Inner-shared	Inner Shareable	ReadShared	CleanUnique if required, then a Write-Back when the line is evicted	ReadShared with **ARLOCK[0]** set to 1[a]	CleanUnique with **ARLOCK[0]** set to 1 if required, then a Write-Back when the line is evicted[a]
	Outer-shared	Outer Shareable				

a. For Inner Shareable transactions where **BROADCASTINNER** is set to 0, **ARLOCK[0]**, is set to 0. For Outer Shareable transactions where **BROADCASTOUTER** is set to 0, **ARLOCK[0]**, is set to 0.

See the *AMBA AXI and ACE Protocol Specification* for more information about ACE transactions.

6.5 Data prefetching

This section describes:

- *PLD and PLDW instructions*.
- *Data prefetching and monitoring*.

6.5.1 PLD and PLDW instructions

PLD and PLDW instructions lookup in the cache, and start a linefill if they miss and are to a cacheable address. However, the PLD or PLDW instruction retires as soon as its linefill is started rather than waiting for data to be returned, which enables other instructions to execute while the linefill continues in the background. If the memory type is Shareable, then any linefill started by a PLDW instruction also causes the data to be invalidated in other processors, so that the line is ready for writing to.

6.5.2 Data prefetching and monitoring

The Cortex-A7 MPCore 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 in four cache lines, plus or minus. Occasionally these linefills might be dropped before the data is allocated into the cache. Any intervening stores or loads that hit in the data cache do not interfere with the recognition of the cache miss pattern. It can be deactivated in software using a CP15 Auxiliary Control Register bit. See *Auxiliary Control Register* on page 4-59.

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-A7 MPCore processor provides a mechanism to read the internal memory used by the 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-A7 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 6-10.
- *TLB RAM accesses* on page 6-11.

6.6.1 Data cache tag and data encoding

The Cortex-A7 MPCore processor data cache consists of a 4-way set-associative structure. The number of sets in each way depends on the configured size of the cache. The encoding, set in Rd in the appropriate `MCR` instruction, used to locate the required cache data entry for tag and data memory is shown in 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. The set-index range parameter (S) is determined by:

$S = \log_2(\text{Data cache size} / 4)$.

Table 6-3 Data cache tag and data location encoding

Bit-field of Rd	Description
[31:30]	Cache way
[29:S+5]	Unused

Table 6-3 Data cache tag and data location encoding (continued)

Bit-field of Rd	Description
[S+5:6]	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. The Cortex-A7 MPCore processor encodes the 4-bit MOESI coherency state across two fields of Data Register 0 and Data Register 1.

Table 6-4 Data cache tag data format

Register	Bit-field	Description
Data Register 0	[31:5]	Unused
Data Register 0	[4:2]	Outer memory attributes
Data Register 0	[1:0]	Partial MOESI state, from dirty RAM
Data Register 1	[31:30]	Partial MOESI state, from tag RAM
Data Register 1	[29]	TrustZone Non-secure state, NS
Data Register 1	[28:0]	Tag address [39:11[a]]

a. Bottom N bits not valid on larger cache sizes, where N is (log2(cache size/8KB)).

6.6.2 Instruction cache tag and data encoding

The Cortex-A7 MPCore processor instruction cache is significantly different from the data cache and this is shown in the encodings and data format used in the CP15 operations used to access the tag and data memories. Table 6-5 shows the encoding required to select a given cache line. The set-index range parameter (S) is determined by:

S = $\log_2$(Instruction cache size (Byte) / 2*32)) for the 2-way set-associative cache.

Table 6-5 Instruction cache tag and data location encoding

Bit-field of Rd	Description
[31]	Cache Way
[30:S+5]	Unused
[S+4:5]	Set index
[4:2]	Cache data element offset, Data Register only
[1:0]	Unused

Table 6-6 shows the tag and valid bits format for the selected cache line using only Data Register 0.

Table 6-6 Instruction cache tag data format

Bit-field of Data Register 0	Description
[31]	Unused
[30]	Valid
[29]	Cache line instruction set mode: **0** ARM. **1** Thumb.
[28]	TrustZone Non-secure state (NS)
[27:0]	Tag address

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

Data Register 0 Bits[17:0] data from cache offset+ `0b00`.

Data Register 1 Bits[17:0] data from cache offset+ `0b10`.

In ARM mode these two fields combined always represent a single instruction. In Thumb, they can represent any combination of 16-bit and partial or full 32-bit instructions.

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

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

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

The 64 bits of cache data is returned in Data register 0 and Data register 1.

6.6.3 TLB RAM accesses

The Cortex-A7 MPCore processor unified TLB is built from a 2-way set-associative RAM based structure. To read the individual entries into the data registers software must write to the TLB Data Read Operation Register. Table 6-7 shows the write TLB Data Read Operation Register location encoding.

Table 6-7 TLB Data Read Operation Register location encoding

Bit-field of Rd	Description
[31]	TLB way
[30:8]	Unused
[7:0]	TLB index

The TLB RAM contains the data for the main TLB, the walk cache, and the *Intermediate Physical Address* (IPA) cache RAMs. Table 6-8 shows the TLB indexes that determines the format of the TLB RAM accesses.

Table 6-8 TLB RAM format

TLB index[7:0]	Format
0-127	Main TLB RAM, see *Main TLB RAM*
128-159	Walk cache RAM, see *Walk cache RAM* on page 6-14
160-191	IPA cache RAM, see *IPA cache RAM* on page 6-15
192-255	Unused

Main TLB RAM

The main TLB RAM uses a 86-bit encoding for the descriptor that is returned in the Data Registers:

Data Register 0[31:0] TLB Descriptor[31:0].

Data Register 1[31:0] TLB Descriptor[62:32].

Data Register 2[19:0] TLB Descriptor[85:64].

Table 6-9 shows the data fields in the TLB descriptor.

Table 6-9 Main TLB descriptor data fields

Bits	Name	Description
[85:84]	S2 Level	The stage 2 level that gave this translation: `0b00` No stage 2 translation performed. `0b01` Level 1. `0b10` Level 2. `0b11` Level 3.
[83:82]	S1 Size	The stage 1 size that gave this translation: `0b00` 4KB. `0b01` 64KB. `0b10` 1MB (VMSAv7) or 2MB (LPAE). `0b11` 16MB (VMSAv7) or 1GB (LPAE).
[81:78]	Domain	Only valid if the entry was fetched in VMSAv7 format.

Table 6-9 Main TLB descriptor data fields (continued)

Bits	Name	Description
[77:72]	Memory Type and shareability	Bits[77:76] are the inner type: 0b00 Non-cacheable. 0b01 Write-Back Write-Allocate. 0b10 Write-Through. 0b11 Device or Strongly Ordered.
		If bits[77:76] == 0b11 (Device or Strongly Ordered) Bit[75] is set if stage-1 translation was overridden by a stage-2 translation. Bits[74:72] encode the type: 0b010 Device. 0b110 Strongly ordered.
		If bits[77:76] != 0b11 (not Device or not Strongly Ordered) Bits[75:74] are the outer type: 0b00 Non-cacheable. 0b01 Write-Back Write-Allocate. 0b10 Write-Through. 0b11 Write-Back no Write-Allocate. Bits[73:72] are for shareability: 0b00 Non-shareable. 0b01 Unused. 0b10 Outer Shareable. 0b11 Inner Shareable.
[71]	XN2[a]	Stage-2 translation Execute Never bit.
[70]	XN1[b]	Stage-1 translation Execute Never bit.
[69]	PXN[b]	Privileged Execute Never.
[68:41]	PA	Physical Address.
[40]	NS, descriptor[b]	Security state allocated to memory region.
[39:38]	HAP[a]	Hypervisor access permissions from the stage-2 translation.
[37:35]	AP or HYP[b]	Access permissions from stage-1 translation and HYP mode flag.
[34]	nG[b]	Not global.
[33:26]	ASID	Address Space Identifier.
[25:18]	VMID	Virtual Machine Identifier.
[17:5]	VA	Virtual address.

Table 6-9 Main TLB descriptor data fields (continued)

Bits	Name	Description
[4]	NS, walk	Security state that the entry was fetched in.
[3:1]	Size	This field indicates the VMSA v7 or LPAE TLB RAM size. VMSA v7: 0b000 4KB. 0b010 64KB. 0b100 1MB. 0b110 16MB. LPAE: 0b001 4KB. 0b011 64KB. 0b101 2MB. 0b111 1GB.
[0]	Valid	Valid bit, when set to 1 the entry contains valid data.

a. This is from the Stage 2 page table. See the *ARM Architecture Reference Manual* for more information.
b. This is from the Stage 1 page table. See the *ARM Architecture Reference Manual* for more information.

Walk cache RAM

The walk cache RAM uses a 86-bit encoding. Table 6-10 shows the data fields in the Walk cache descriptor.

Table 6-10 Walk cache descriptor fields

Bits	Name	Description
[85:82]	Unused	-
[81:78]	Domain	Only valid if the entry was fetched in VMSAv7 format
[77:48]	PA	The physical address of the level 3 (LPAE) or level 2 (VMSAv7) table
[47:41]	VA	Virtual address
[40]	-	Unused
[39]	NSTable	Combined NSTable bits from first and second level stage 1 tables (LPAE) or NS descriptor (VMSAv7)
[38]	PXNTable	Combined PXNTable bit from first and second level stage 1 tables
[37]	XNTable	Combined XNTable bit from first and second level stage 1 tables
[36:35]	APTable	Combined APTable bits from first and second level stage 1 tables
[34]	HYP	HYP bit, when set to 1 indicates the entry was fetched in HYP mode
[33:26]	ASID	Indicates the Address Space Identifier
[25:18]	VMID	Virtual Machine Identifier
[17:12]	Attrs	Physical attributes of the final level stage 1 table
[11:5]	-	Unused
[4]	NS, walk	Security state that the entry was fetched in

Table 6-10 Walk cache descriptor fields (continued)

Bits	Name	Description
[3:2]	-	Unused
[1]	LPAE	LPAE bit. Indicates what format the entry was fetched in: **0** VMSAv7 format. **1** LPAE format.
[0]	Valid	Valid bit. When set to 1 the entry contains valid data.

IPA cache RAM

The *Intermediate Physical Address* (IPA) cache RAM uses a 86-bit encoding. Table 6-10 on page 6-14 shows the data fields in the IPA cache descriptor.

Table 6-11 IPA cache descriptor fields

Bits	Name	Description
[85:82]	Memattrs	Memory attributes
[81:59]	IPA	Unused lower bits, page size dependent, must be zero
[58:31]	PA	Physical address
[30]	XN	Execute Never
[29:28]	HAP	Hypervisor access permissions
[27:26]	SH	Shareability
[25:18]	VMID	Virtual Machine Identifier
[17:4]	Unused	-
[3:1]	Size	The size values are: `0b001` 4KB. `0b011` 64KB. `0b101` 2MB. `0b111` 1GB.
[0]	Valid	The entry contains valid data

Chapter 7
L2 Memory System

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

7.1 About the L2 Memory system

The L2 memory system consists of an:

- Integrated *Snoop Control Unit* (SCU), connecting up to four processors within a cluster. The SCU also has duplicate copies of the L1 data cache directories for coherency support. The L2 memory system interfaces with an AMBA *AXI Coherency Extension* (ACE) interconnect on a 128-bit wide bus.
- Optional tightly-coupled L2 cache that includes:
  - Configurable L2 cache size of 128KB, 256KB, 512KB, and 1MB.
  - Fixed line length of 64 bytes.
  - Physically indexed and tagged cache.
  - 8-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 handling* on page 7-10.

7.2 Snoop Control Unit

Cortex-A7 MPCore processor supports between one and four individual processors with L1 data cache coherency maintained by the SCU. The SCU is clocked synchronously and at the same frequency as the processors.

The SCU maintains coherency between the individual data caches in the processor using ACE modified equivalents of MOESI state, as described in *Data Cache Unit* on page 2-4.

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.

The SCU also controls snoop and maintenance requests to the system using the external **BROADCASTINNER**, **BROADCASTOUTER**, and **BROADCASTCACHEMAINT** pins:

- When you set the **BROADCASTINNER** pin to 1 the inner shareability domain extends beyond the Cortex-A7 cluster and Inner Shareable snoop and maintenance operations are broadcast externally. When you set the **BROADCASTINNER** pin to 0 the inner shareability domain does not extend beyond the Cortex-A7 cluster.
- When you set the **BROADCASTOUTER** pin to 1 the outer shareability domain extends beyond the Cortex-A7 cluster and outer shareable snoop and maintenance operations are broadcast externally. When you set the **BROADCASTOUTER** pin to 0 the outer shareability domain does not extend beyond the Cortex-A7 cluster.
- When you set the **BROADCASTCACHEMAINT** pin to 1 this indicates to the Cortex-A7 cluster that there are external downstream caches and maintenance operations are broadcast externally. When you set the **BROADCASTCACHEMAINT** pin to 0 there are no downstream caches external to the Cortex-A7 cluster.

Note

- If you set the **BROADCASTINNER** pin to 1 you must also set the **BROADCASTOUTER** pin to 1.
- In a system that contains Cortex-A15 and Cortex-A7 MPCore processor clusters, you must ensure the **BROADCASTINNER** and **BROADCASTOUTER** pins on both processor clusters are set to 1 so that both clusters are in the same Inner Shareable domain.

7.2.1 ACE configuration signals

The Cortex-A7 MPCore processor implements the following ACE configuration signals:

- **BROADCASTINNER**.
- **BROADCASTOUTER**.
- **BROADCASTCACHEMAINT**.
- **SYSBARDISABLE**.

Table 7-1 shows the permitted combinations of these signals and the supported configurations in the Cortex-A7 MPCore processor.

Table 7-1 Supported ACE configurations

Signal	Feature						
	AXI3 mode[a]	ACE non-coherent[b]		ACE outer coherent		ACE inner coherent	
		No L3 Cache	With L3 Cache	No L3 Cache	With L3 Cache	No L3 Cache	With L3 Cache
BROADCASTCACHEMAINT	0	0	1	0	1	0	1
BROADCASTOUTER	0	0	0	1	1	1	1
BROADCASTINNER	0	0	0	0	0	1	1

a. **SYSBARDISABLE** must be set to 1 in AXI3 mode.

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

Table 7-2 shows the key features in each of the supported ACE configurations.

Table 7-2 Supported features in the ACE configurations

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

7.3 Master interface

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

7.3.1 Memory interface attributes

Table 7-3 shows the ACE master interface attributes for the Cortex-A7 MPCore processor. The table lists the maximum possible values for the read and write issuing capabilities if the processor includes four processors.

Table 7-3 ACE master interface attributes

Attribute	Value	Comments
Write issuing capability	$33+n^a$	The write issue capability is made up of: • 16 Non-cacheable, Device or Strongly-ordered writes[b]. • 16 Cacheable (either inner or outer) writes. • 1 barrier operation for each processor. 1 barrier is generated from the cluster.
Read issuing capability	$8n^a + 66$	8 for each processor in the multiprocessor device including up to: • 5 data linefills. • 1 Non-cacheable, Device, or Strongly ordered data read. • 1 Non-cacheable TLB page-walk read. • 1 instruction fetch, either cacheable linefill or Non-cacheable. • 4 coherency operations. • 1 barrier operation. 1 barrier operation is generated from the cluster and up to 63 outstanding DVM messages. Up to 2 Level 2 linefills generated by the cluster if L2 is present
Exclusive thread capability	n^a	Each processor can have 1 exclusive access sequence in progress
Write ID capability	$3n^a + 17$	The Write ID capability is made up of: • 1 for Non-cacheable writes for each processor. • 1 for Device or Strongly-ordered writes for each processor. • 1 barrier operation for each processor. • 1 for barriers across the whole cluster. • 16 for cacheable writes across the whole cluster.
Write ID width	5	The ID encodes the source of the memory transaction. See Table 7-4 on page 7-6.
Read ID capability	$10n^a + 5$	10 for each processor in the multiprocessor device including: • 3 for the L2 cache. • 1 for DVM messages. • 1 for barriers. If the L2 cache is not present the read ID capability value changes to 10n + 2.
Read ID width	6	The ID encodes the source of the memory transaction. See Table 7-5 on page 7-6.

a. n is the processor number 1-4.

b. For a single processor, there is a limit of 15 normal non-cacheable writes

Table 7-4 shows the Encodings for **AWIDM[4:0]**

Table 7-4 Encodings for AWIDM[4:0]

Attribute	Value	Comments
Write ID	0b000nn[a]	Processor nn Non-cacheable or STREX
	0b001nn[a]	Processor nn writes to Device and Strongly-ordered memory
	0b010nn[a]	Processor nn write portion of the barrier transactions
	0b01111	Write portion of barrier caused by external DVM synchronization
	0b1bbbb[b]	Writes to cacheable memory

a. Where nn is the processor number 0b00, 0b01, 0b10, or 0b11.
b. Where bbbb is an arbitrary value between 0x0-0xF.

Table 7-5 shows the Encodings for **ARIDM[5:0]**

Table 7-5 Encodings for ARIDM[5:0]

Attribute	Value	Comments
Read ID	0b0000nn[a]	Processor nn Non-cacheable, Device or Strongly-ordered, including LDREX
	0b0001nn[a]	Processor nn TLB
	0b0010nn[a]	Processor nn read portion of barrier transactions
	0b001111	Read portion of barrier transaction caused by external DVM synchronization
	0b0100nn[a]	Processor nn Line-Fill Buffer 0
	0b0101nn[a]	Processor nn Line-Fill Buffer 1
	0b0110nn[a]	Processor nn instruction
	0b1000nn[a]	Processor nn STB0
	0b1001nn[a]	Processor nn STB1
	0b1010nn[a]	Processor nn STB2
	0b1011nn[a]	Processor nn STB3
	0b1100nn[a]	Processor nn DVM request
	0b110100	DVM Complete messages
	0b111mmm[b]	L2 Line-Fill Buffer

a. Where nn is the processor number 0b00, 0b01, 0b10, or 0b11.
b. Where mmm is L2 encoding of the Line-Fill Buffer.

See the *AMBA AXI and ACE Protocol Specification* for more information about the ACE and AXI signals described in this manual.

7.3.2 ACE transfers

Cortex-A7 does not generate any FIXED bursts and all WRAP bursts fetch a complete cache line starting with the critical word first. A burst does not cross a cache line boundary. The instruction linefill length is:

- 32-bytes on the instruction side when no L2 cache is present.
- 64-bytes on the instruction side when there is a L2 cache present.

In a system with L2 cache, Instruction side linefills cause 32-byte fetches if the L2 cache is turned off.

The cache linefill fetch length is always 64-bytes on the data side.

The Cortex-A7 MPCore processor generates only a subset of all possible AXI transactions on the master interface.

For Write-Back Write-Allocate transfers the supported transfers are:

- WRAP2 128-bit for read transfers (linefills, no L2 cache present, or L2 cache turned off).
- WRAP4 128-bit for read transfers (linefills).
- WRAP4 128-bit for write transfers (L1 evictions).
- INCR4 128-bit for write transfers (L2 evictions).
- INCR N (N:1-4)128-bit write transfers (read allocate).

For Non-cacheable transactions:

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

For Device or Strongly-ordered transactions:

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

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

The following points apply to AXI transactions:

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

7.3.3 ACE channel properties

Table 7-6 show the properties of the ACE channels.

Table 7-6 ACE channel properties

Property	Value	Comment
Snoop acceptance capability	6	The SCU can accept and process a maximum of six snoop requests from the system.
Snoop latency	Hit	When there is a hit in L2 cache, the best case for response and data is eight processor cycles. When there is a miss in L2 cache and a hit in L1 cache, the best case for response and data is ten processor cycles. **Note** Latencies can be higher if hazards occur or if there are not enough buffers to absorb requests.
	Miss	Best case six processor cycles as the SCU duplicate tags and L2 tags indicate the miss.
	DVM	The multiprocessor device takes: • Three cycles to provide an AC-CR response to non-DVM synchronization packets providing that there are no ongoing snoops in to a processor. • Minimum of six cycles to provide a CR-AR response to DVM synchronization packets.
Snoop filter	Supported	The multiprocessor device provides support for an external snoop filter in an interconnect. It indicates when clean lines are evicted from the processor by sending Evict transactions on the ACE write channel. However there are some cases where incorrect software can prevent an Evict transaction from being sent, therefore you must ensure that any external snoop filter is built to handle a capacity overflow that sends a back-invalidation to the processor if it runs out of storage.
Supported transactions	-	All transactions described by the ACE protocol: • Are accepted on the ACE master interface from the system. • Can be produced on the ACE master interface except: — WriteUnique — WriteLineUnique — ReadNotSharedDirty — MakeUnique.
DVM issue	63	DVM messages, except for DVM synchronization, can be issued back-to-back continuously as Cortex-A7 MPCore processor does not require a response from the interconnect fabric to retire a single DVM message or multi-part DVM messages.

See the *AMBA AXI and ACE Protocol Specification* for more information about the ACE channel.

7.3.4 AXI transaction IDs

The AXI ID signal encodings are described in Table 7-4 on page 7-6 and Table 7-5 on page 7-6.

7.3.5 Write response

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

7.3.6 AXI3 Compatibility mode

The Cortex-A7 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 and the **BROADCASTINNER**, **BROADCASTOUTER**, and **BROADCASTCACHEMAINT** input pins to 0 on the boundary of the processor.

7.4 Optional integrated L2 cache

The optional integrated L2 configurable caches sizes are 128KB, 256KB, 512KB, and 1MB.

Data is only allocated to the L2 cache when evicted from the L1 memory system, not when first fetched from the system. The L1 cache can prefetch data from the system without data being evicted from the L2 cache.

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

The L2 cache is 8-way set associative. The L2 cache tags are looked up in parallel with the SCU duplicate tags. If both the L2 tag and SCU duplicate tag hit the L2 tag hit takes priority.

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.

7.4.1 External aborts handling

The L2 memory system handles two types of external abort depending on the attributes of the memory region of the access:

- All load accesses use the synchronous abort mechanism.
- All `STREX`, `STREXB`, `STREXH`, and `STREXD` instructions use the synchronous abort mechanism.
- All store accesses to Device, Strongly-ordered, or inner and outer Non-cacheable normal memory use the asynchronous abort mechanism, except for `STREX`, `STREXB`, `STREXH`, and `STREXD`.
- All store accesses to normal memory that is either inner cacheable or outer cacheable and any evictions from L1 or L2 cache do not cause an abort in the processor, instead they assert the **nAXIERRIRQ** pin. This is because the access that aborts might not relate directly back to a specific processor in the cluster.
- L2 linefills triggered by an L1 instruction fetch assert the **nAXIERRIRQ** pin if the data is received from the interconnect in a dirty state. Instruction data can be marked as dirty as a result of self-modifying code or a line containing a mixture of data and instructions. If an error response is received on any part of the line, the dirty data might be lost.
- DVM operations that receive an error response assert the **nAXIERRIRQ** pin.

7.5 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-A7 MPCore processor marks the transaction as privileged, even it if was initiated by a user process.

Table 7-7 shows Cortex-A7 MPCore processor modes and corresponding **ARPROTM[2:0]** and **AWPROTM[2:0]** values.

Table 7-7 Cortex-A7 MPCore 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
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

Chapter 8
Generic Interrupt Controller

This chapter describes the optionally integrated *Generic Interrupt Controller* (GIC). It contains the following sections:

8.1 About the GIC

The integrated GIC collates and arbitrates from a large number of interrupt sources. It provides:

- Masking of interrupts.
- Prioritization of interrupts.
- Distribution of the interrupts to the target processors.
- Tracking the status of interrupts.
- Generation of interrupts by software.
- Support for Security Extensions.
- Support for Virtualization Extensions.

The integrated GIC is compliant with the version 2.0 of the *ARM Generic Interrupt Controller (GIC) Architecture Specification.*

This chapter only describes features that are specific to the Cortex-A7 MPCore implementation.

8.2 GIC functional description

This section provides a functional description of the Cortex-A7 MPCore GIC in:

- *GIC memory-map*.
- *Interrupt sources* on page 8-4.
- *Interrupt priority formats* on page 8-5.
- *GIC configuration* on page 8-5.

The GIC is a single functional unit located in the Cortex-A7 MPCore processor. The GIC consists of a shared Distributor block and several CPU interfaces. For each processor in the multiprocessor device, there is:

- One CPU Interface.
- A virtual interface control.
- A virtual CPU interface.

8.2.1 GIC memory-map

The GIC registers are memory-mapped, and the base address is specified by **PERIPHBASE[39:15]**. This input must be tied to a constant value. The **PERIPHBASE** value is sampled during reset into the *Configuration Base Address* (CBAR) for each processor in the cluster. See *Configuration Base Address Register* on page 4-83.

Memory regions used for these registers must be marked as Device or Strongly-ordered in the translation tables.

Memory regions marked as Normal Memory cannot access any of the GIC registers, instead access caches or external memory as required.

Table 8-1 lists the address offsets for the GIC blocks relative to the **PERIPHBASE** base address. Access to reserved regions can result in a data abort exception to the requesting processor.

Table 8-1 GIC memory map

Offset from PERIPHBASE[39:15]	GIC block
0x0000-0x0FFF	Reserved
0x1000-0x1FFF	Distributor
0x2000-0x3FFF	CPU interface
0x4000-0x4FFF	Virtual interface control, common base address
0x5000-0x5FFF	Virtual interface control, processor-specific base address
0x6000-0x7FFF	Virtual CPU interface

The GIC provides two aliases of the GIC virtual interface control registers:

- An alias that provides access to the Virtual CPU interface of the assessing processor using a single base address for all processors in the multiprocessor device. This base address is at offset 0x4000 relative to the **PERIPHBASE** address.
- An alias that permits the Virtual CPU interface of any processor in the multiprocessor device to be accessed explicitly from any other processor, using a different base address for each processor. The starting base address is at offset 0x5000 relative to the **PERIPHBASE** address, with address bits[11:9] as the CPU ID decode.

8.2.2 Interrupt sources

The Cortex-A7 MPCore processor can support up to 480 *Shared Peripheral Interrupts* (SPIs). All interrupt sources are identified by a unique ID.

The Cortex-A7 MPCore processor has the following interrupt sources:

Software Generated Interrupts

SGIs are generated by writing to the *Software Generated Interrupt Register* (GICD_SGIR). A maximum of 16 SGIs, ID0-ID15, can be generated for each processor interface. An SGI has edge-triggered properties. The software triggering of the interrupt is equivalent to the edge transition of the interrupt signal on a peripheral input.

Private Peripheral Interrupts

A PPI is an interrupt generated by a peripheral that is specific to a single processor. There are seven PPIs for each CPU interface:

Legacy nFIQ signal (PPI0)

When the GIC interrupt bypass is in effect, such as after reset, the external **nFIQ** signal bypasses the interrupt distributor logic and directly drives the interrupt request to the corresponding processor.

When a processor uses the GIC rather than the external **nFIQ** signal, by enabling its own CPU interface, the **nFIQ** signal is treated like other interrupt lines and uses ID28. The interrupt is active-LOW level-sensitive.

Secure Physical Timer event (PPI1)

This is the event generated from the Secure Physical Timer and uses ID29. The interrupt is level-sensitive.

Non-secure Physical Timer event (PPI2)

This is the event generated from the Non-secure Physical Timer and uses ID30. The interrupt is level-sensitive.

Legacy nIRQ signal (PPI3)

When the GIC interrupt bypass is in effect, such as after reset, the external **nIRQ** signal bypasses the interrupt distributor logic and directly drives the interrupt request to the corresponding processor.

When a processor uses the GIC rather than the external **nIRQ** signal, by enabling its own CPU interface, the **nIRQ** signal is treated like other interrupt lines and uses ID31. The interrupt is active-LOW level-sensitive.

Virtual Timer event (PPI4)

This is the event generated from the Virtual Timer and uses ID27. The interrupt is level-sensitive.

Hypervisor Timer event (PPI5)

This is the event generated from the Physical Timer in Hypervisor mode and uses ID26. The interrupt is level-sensitive.

Virtual Maintenance Interrupt (PPI6)

The Virtualization Extensions support in the *ARM Generic Interrupt Controller Architecture Specification* permits for a maintenance interrupt to be generated under several conditions. The virtual maintenance interrupt uses ID25. The interrupt is level-sensitive.

Shared Peripheral Interrupts

SPIs are triggered by events generated on associated interrupt input lines. The GIC can support up to 480 SPIs corresponding to the external **IRQS** signal. The number of SPIs available depends on the implemented configuration of the Cortex-A7 MPCore processor. The permitted values are 0-480, in steps of 32. SPIs start at ID32. The SPIs can be configured to be edge-triggered or active-HIGH level-sensitive.

8.2.3 Interrupt priority formats

The Cortex-A7 MPCore processor implements a 5-bit version of the interrupt priority field for 32 interrupt priority levels.

8.2.4 GIC configuration

Table 8-2 lists the configurable options for the GIC.

Table 8-2 GIC configurable options

Feature	Range of options
GIC	Present or not present
SPIs	0 to 480, in steps of 32

It is implementation-defined to whether the GIC is present or not present.

If you configure the design to exclude the GIC:

- SPI signals are not available.
- **PERIPHBASE[39:15]** are retained and software can read the Configuration Base Address Register to determine the location of the GIC if it exists in the external system. See *Configuration Base Address Register* on page 4-83.

The Cortex-A7 MPCore processor always includes the virtual interrupt signals, **nVIRQ** and **nVFIQ**, regardless of whether the GIC is present or not. There is one **nVIRQ** and one **nVFIQ** for each processor. If you configure the processor to exclude the GIC, the input pins **nVIRQ** and **nVFIQ** can be tied off to HIGH if unused, or can be driven by an external GIC in the SoC.

If you configure the processor to include the GIC, and the GIC is used, the input pins **nVIRQ** and **nVFIQ** must be tied off to HIGH. This is because the internal GIC generates the virtual interrupt signals to the processors. If you configure the processor to include the GIC, and the GIC is not used, the input pins **nVIRQ** and **nVFIQ** can be driven by an external GIC in the SoC.

8.3 GIC programmers model

This section describes the programmers model for the Cortex-A7 MPCore GIC in:

- *Distributor register summary*.
- *Distributor register descriptions* on page 8-8.
- *CPU Interface register summary* on page 8-13.
- *CPU Interface register descriptions* on page 8-14.
- *Virtual interface control register summary* on page 8-16.
- *Virtual Interface Control Register description* on page 8-17.
- *Virtual CPU interface register summary* on page 8-18.
- *Virtual CPU interface register description* on page 8-19.

8.3.1 Distributor register summary

The Distributor centralizes all interrupt sources, determines the priority of each interrupt, and for each CPU interface forwards the interrupt with the highest priority to the interface for priority masking and preemption handling.

The Distributor provides a programming interface for:

- Globally enabling the forwarding of interrupts to the CPU interfaces.
- Enabling or disabling each interrupt.
- Setting the priority level of each interrupt.
- Setting the target processor list of each interrupt.
- Setting each peripheral interrupt to be level-sensitive or edge-triggered.
- Setting each interrupt as either Group 0 or Group 1, see the *ARM Generic Interrupt Controller Architecture Specification* for information on interrupt grouping.
- Sending an SGI to one or more target processors.
- Visibility of the state of each interrupt.
- A mechanism for software to set or clear the pending state of a peripheral interrupt.

Table 8-3 shows the register map for the Distributor. The offsets in this table are relative to the Distributor block base address as shown in Table 8-1 on page 8-3.

The GICD_IPRIORITYRn, GICD_ITARGETSRn, GICD_CPENDSGIRn, and GICD_SPENDSGIRn registers are byte-accessible and halfword-accessible. All other registers in Table 8-3 are word-accessible. Registers not described in Table 8-3 are RAZ/WI.

Table 8-3 Distributor register summary

Offset	Name	Type	Reset	Full name
0x000	GICD_CTLR	RW	0x00000000[a]	Distributor Control Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x004	GICD_TYPER	RO	IMPLEMENTATION DEFINED	*Interrupt Controller Type Register* on page 8-8
0x008	GICD_IIDR	RO	0x0100143B	*Distributor Implementer Identification Register* on page 8-10
0x080-0x0BC	GICD_IGROUPRn	RW	0x00000000	Interrupt Group Registers, see *ARM Generic Interrupt Controller Architecture Specification*[b]

Table 8-3 Distributor register summary (continued)

Offset	Name	Type	Reset	Full name
0x100	GICD_ISENABLERn	RW[c]	0x0000FFFF[d]	Interrupt Set-Enable Registers, see the *ARM Generic Interrupt Controller Architecture Specification*
0x104-0x13C			0x00000000	
0x180	GICD_ICENABLERn	RW[c]	0x0000FFFF[d]	Interrupt Clear-Enable Registers, see the *ARM Generic Interrupt Controller Architecture Specification*
0x184-0x1BC			0x00000000	
0x200-0x23C	GICD_ISPENDRn	RW	0x00000000	Interrupt Set-Pending Registers, see the *ARM Generic Interrupt Controller Architecture Specification*
0x280-0x2BC	GICD_ICPENDRn	RW	0x00000000	Interrupt Clear-Pending Registers, see the *ARM Generic Interrupt Controller Architecture Specification*
0x300-0x33C	GICD_ISACTIVERn	RW	0x00000000	Interrupt Set-Active Registers, see the *ARM Generic Interrupt Controller Architecture Specification*
0x380-0x3BC	GICD_ICACTIVERn	RW	0x00000000	Interrupt Clear-Active Registers, see the *ARM Generic Interrupt Controller Architecture Specification*
0x400-0x5FC	GICD_IPRIORITYRn[e]	RW	0x00000000	Interrupt Priority Registers, see *ARM Generic Interrupt Controller Architecture Specification*
0x800-0x81C	GICD_ITARGETSRn	RO[f]	-	Interrupt Processor Targets Registers, see the *ARM Generic Interrupt Controller Architecture Specification*
0x820-0x9FC		RW	0x00000000	
0xC00	GICD_ICFGRn	RO	0xAAAAAAAA[g]	*Interrupt configuration registers* on page 8-10
0xC04		RO	0x55540000[g]	
0xC08-0xC7C		RW	0x55555555[g]	
0xD00	GICD_PPISR	RO	0x00000000	*Private Peripheral Interrupt Status Register* on page 8-11
0xD04-0xD3C	GICD_SPISRn	RO	0x00000000	*Shared Peripheral Interrupt Status Registers* on page 8-12
0xF00	GICD_SGIR	WO	-	Software Generated Interrupt Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xF10-0xF1C	GICD_CPENDSGIRn	RW	0x00000000	SGI Clear-Pending Registers, see the *ARM Generic Interrupt Controller Architecture Specification*
0xF20-0xF2C	GICD_SPENDSGIRn	RW	0x00000000	SGI Set-Pending Registers, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFD0	GICD_PIDR4	RO	0x00000004	Peripheral ID4 Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFD4	GICD_PIDR5	RO	0x00000000	Peripheral ID5 Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFD8	GICD_PIDR6	RO	0x00000000	Peripheral ID6 Register, see the *ARM Generic Interrupt Controller Architecture Specification*

Table 8-3 Distributor register summary (continued)

Offset	Name	Type	Reset	Full name
0xFDC	GICD_PIDR7	RO	0x00000000	Peripheral ID7 Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFE0	GICD_PIDR0	RO	0x00000090	Peripheral ID0 Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFE4	GICD_PIDR1	RO	0x000000B4	Peripheral ID1 Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFE8	GICD_PIDR2	RO	0x0000002B	Peripheral ID2 Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFEC	GICD_PIDR3	RO	0x00000000	Peripheral ID3 Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFF0	GICD_CIDR0	RO	0x0000000D	Component ID0 Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFF4	GICD_CIDR1	RO	0x000000F0	Component ID1 Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFF8	GICD_CIDR2	RO	0x00000005	Component ID2 Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0xFFC	GICD_CIDR3	RO	0x000000B1	Component ID3 Register, see the *ARM Generic Interrupt Controller Architecture Specification*

a. You cannot modify the EnableGrp0 bit if **CFGSDISABLE** is asserted.

b. This register is only accessible from a Secure access.

c. Writes to bits corresponding to the SGIs are ignored.

d. The reset value for the register that contains the SGI and PPI interrupts is 0x0000FFFF because SGIs are always enabled. However, SGIs are Group 0 on reset, so the reset value for Non-secure reads is 0x00000000.

e. Writing to the GICD_IPRIORITYR does not affect the priority of an active interrupt.

f. The register that contains the SGI and PPI interrupts is read-only and the value is implementation defined. For Cortex-A7 configurations with only one processor, these registers are RAZ/WI.

g. The reset value for the register that contains the SGI interrupts is 0xAAAAAAAA. The reset value for the register that contains the PPI interrupts is 0x55540000. The reset value for the registers that contain the SPI interrupts is 0x55555555.

8.3.2 Distributor register descriptions

This section only describes registers whose implementation is specific to the Cortex-A7 MPCore processor. All other registers are described in the *ARM Generic Interrupt Controller Architecture Specification*.

Interrupt Controller Type Register

The GICD_TYPER characteristics are:

Purpose Provides information about the configuration of the integrated GIC. It indicates:

- Whether the GIC implements the Security Extensions.
- The maximum number of interrupt IDs that the GIC supports.
- The maximum number of processor interfaces implemented.
- If the GIC implements the Security Extensions, the maximum number of implemented *Lockable Shared Peripheral Interrupts* (LSPIs).

Usage constraints There are no usage constraints.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-3 on page 8-6.

Figure 8-1 shows the GICD_TYPER bit assignments.

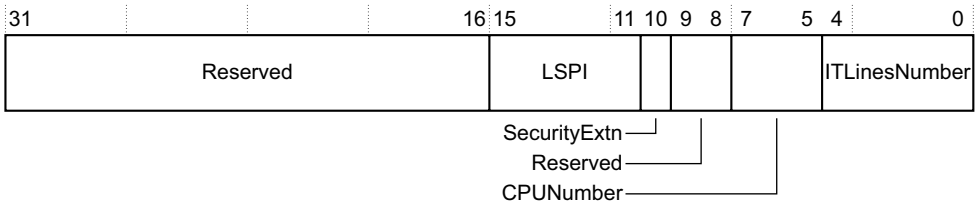


Figure 8-1 GICD_TYPER bit assignments

Table 8-4 shows the GICD_TYPER bit assignments.

Table 8-4 GICD_TYPER bit assignments

Bits	Name	Function
[31:16]	-	Reserved, RAZ.
[15:11]	LSPI	Returns the number of *Lockable Shared Peripheral Interrupts* (LSPIs) that the Interrupt Controller contains: `0b11111` 31 LSPIs, these are the interrupts of IDs 32-62. When **CFGSDISABLE** is asserted, the GIC prevents writes to any register locations that control the operating state of an LSPI.
[10]	SecurityExtn	Indicates whether the GIC implements the Security Extensions. This bit always returns a value of 1, indicating that the Security Extensions are implemented.
[9:8]	-	Reserved, RAZ.
[7:5]	CPUNumber	Indicates the number of implemented processors: `0b000` The Cortex-A7 MPCore configuration contains one processor. `0b001` The Cortex-A7 MPCore configuration contains two processors. `0b010` The Cortex-A7 MPCore configuration contains three processors. `0b011` The Cortex-A7 MPCore configuration contains four processors. All other values are reserved for future expansions.
[4:0]	ITLinesNumber	Indicates the number of interrupts that the GIC supports: `0b00000` Up to 32 interrupts[a], no external interrupt lines. `0b00001` Up to 64 interrupts, 32 external interrupt lines. `0b00010` Up to 96 interrupts, 64 external interrupt lines. . . . `0b01111` Up to 512 interrupts, 480 external interrupt lines. All other values are reserved for future expansions.

a. The Distributor always uses interrupts of IDs 0 to 31 to control any SGIs and PPIs that the GIC might contain.

Distributor Implementer Identification Register

The GICD_IIDR characteristics are:

Purpose Provides information about the implementer and revision of the Distributor.

Usage constraints There are no usage constraints.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-3 on page 8-6.

Figure 8-2 shows the GICD_IIDR bit assignments.

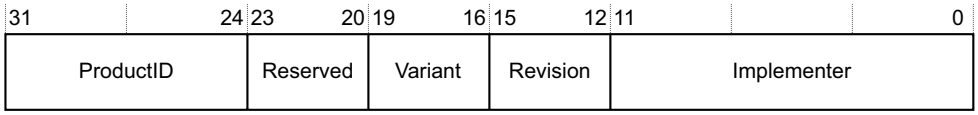


Figure 8-2 GICD_IIDR bit assignments

Table 8-5 shows the GICD_IIDR bit assignments.

Table 8-5 GICD_IIDR bit assignments

Bits	Name	Description
[31:24]	ProductID	Indicates the product ID of the GIC: `0x01` Cortex-A7.
[23:20]	-	Reserved, RAZ.
[19:16]	Variant	Indicates the major revision number of the GIC: `0x0` Variant number.
[15:12]	Revision	Indicates the minor revision number of the GIC: `0x1` Revision number.
[11:0]	Implementer	Indicates the implementer: `0x43B` ARM implementation.

Interrupt configuration registers

The GICD_ICFGR provides a 2-bit field that describes the configuration for each interrupt that the GIC supports.

The options for each bit-pair depend on the interrupt type as follows:

SGI The bits are read-only and a bit-pair always reads as `b10` because SGIs are edge-triggered.

PPI The bits are read-only and a bit-pair always reads as `b01` because the PPIs are implemented as level-sensitive.

SPI The *Least Significant Bit* (LSB) of the bit-pair is read-only and is always 1. You can program the *Most Significant Bit* (MSB) of the bit-pair to alter the triggering sensitivity as follows:

`b01` Interrupt is active-HIGH level-sensitive.

`b11` Interrupt is rising edge-sensitive.

See the *ARM Generic Interrupt Controller Architecture Specification* for more information.

Private Peripheral Interrupt Status Register

The GICD_PPISR characteristics are:

Purpose Enables a Cortex-A7 MPCore processor to access the status of the PPI inputs on the Distributor. Non-secure accesses can only read the status of Group 1 interrupts.

Usage constraints A processor can only read the status of its own PPI and therefore cannot read the status of the PPI for other processors. .

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-3 on page 8-6.

Figure 8-3 shows the GICD_PPISR bit assignments.

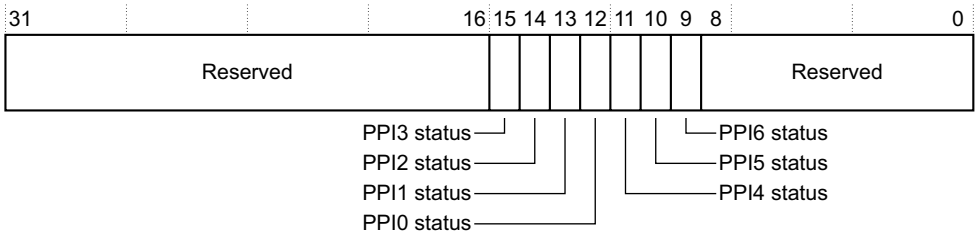


Figure 8-3 GICD_PPISR bit assignments

Table 8-6 shows the GICD_PPISR bit assignments.

Table 8-6 GICD_PPISR bit assignments

Bits	Name	Description
[31:16]	-	Reserved, RAZ
[15:9]	PPI status	Assert when the **PPI [6:0]** inputs to the Distributor are asserted: **PPI6** Virtual Maintenance Interrupt. **PPI5** Hypervisor timer event. **PPI4** Virtual timer event. **PPI3** **nIRQ**. **PPI2** Non-secure physical timer event. **PPI1** Secure physical timer event. **PPI0** **nFIQ**. **Note** These bits return the actual status of the **PPI[6:0]** signals. The GICD_ISPR and GICD_ICPR can also provide the **PPI[6:0]** status, although you can write to these registers, they might not contain the true status of the PPI input signals.
[8:0]	-	Reserved, RAZ

Shared Peripheral Interrupt Status Registers

The GICD_SPISRn characteristics are:

Purpose Enables a Cortex-A7 MPCore processor to access the status of **IRQS[479:0]** signals on the Distributor.

Usage constraints Non-secure accesses can only read the status of Group 1 interrupts.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-3 on page 8-6.

Figure 8-4 shows the GICD_SPISRn register bit assignments.

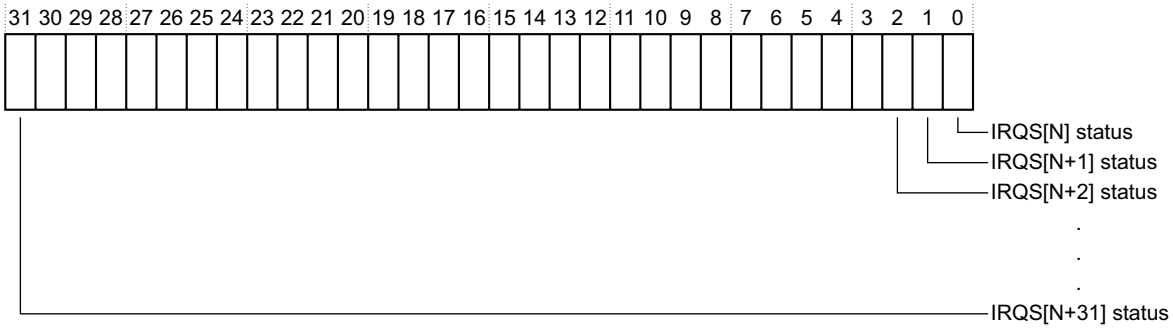


Figure 8-4 GICD_SPISRn bit assignments

Table 8-7 shows the GICD_SPISRn register bit assignments.

Table 8-7 GICD_SPISRn bit assignments

Bits	Name	Function
[31:0]	IRQS[N+31:N]	Returns the status of the **IRQS[479:0]** signals on the Distributor. For each bit: 0 **IRQS** is LOW. 1 **IRQS** is HIGH. **Note** • The **IRQS** that a bit refers to depends on its bit position and the base address offset of the GICD_SPISRn. • These bits return the actual status of the **IRQS[479:0]** signals. The GICD_ISPENDRn and GICD_ICPENDRn can also provide the **IRQS[479:0]** status, although you can write to these registers, they might not contain the actual status of the **IRQS[479:0]** signals.

Figure 8-5 on page 8-13 shows the address map that the Distributor provides for the SPIs.

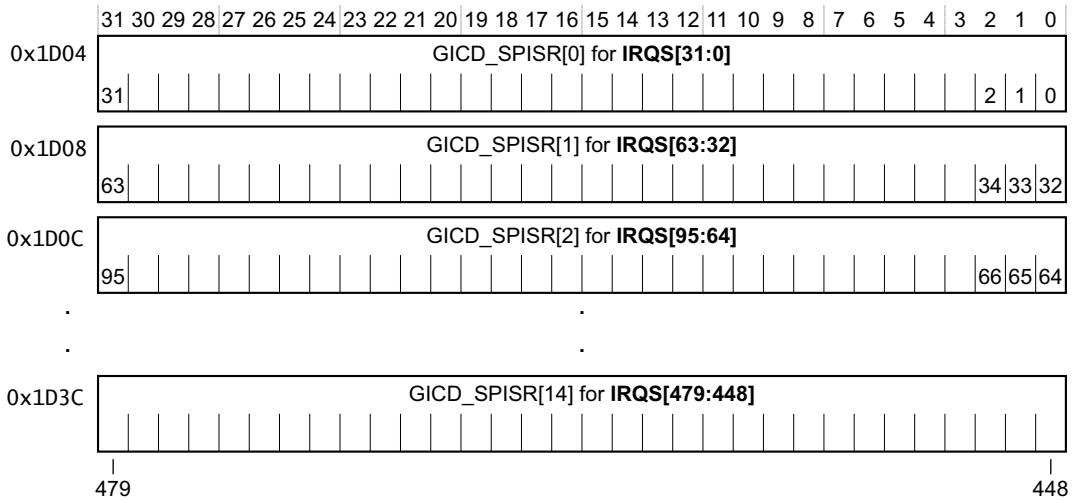


Figure 8-5 GICD_SPISRn address map

The Distributor provides up to 15 registers to support 480 SPIs. If you configure the GIC to use fewer than 480 SPIs, it reduces the number of registers accordingly. For locations where interrupts are not implemented, the bit is RAZ/WI.

8.3.3 CPU Interface register summary

Each CPU interface provides a programming interface for:

- Enabling the signaling of interrupt requests by the CPU interface.
- Acknowledging an interrupt.
- Indicating completion of the processing of an interrupt.
- Setting an interrupt priority mask for the processor.
- Defining the preemption policy for the processor.
- Determining the highest priority pending interrupt for the processor.

For more information on CPU interfaces, see the *ARM Generic Interrupt Controller Architecture Specification*.

Table 8-8 shows the register map for the CPU Interface. The offsets in this table are relative to the CPU Interface block base address as shown in Table 8-1 on page 8-3.

All the registers in Table 8-8 are word-accessible. Registers not described in this table are RAZ/WI.

Table 8-8 CPU Interface register summary

Offset	Name	Type	Reset	Description
0x0000	GICC_CTLR	RW	0x00000000	CPU Interface Control Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0004	GICC_PMRn	RW	0x00000000	Interrupt Priority Mask Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0008	GICC_BPR	RW	0x00000002 (S)[a] 0x00000003 (NS)[b]	Binary Point Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x000C	GICC_IAR	RO	0x000003FF	Interrupt Acknowledge Register, see the *ARM Generic Interrupt Controller Architecture Specification*

Table 8-8 CPU Interface register summary (continued)

Offset	Name	Type	Reset	Description
0x0010	GICC_EOIR	WO	-	End Of Interrupt Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0014	GICC_RPR	RO	0x000000FF	Running Priority Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0018	GICC_HPPIR	RO	0x000003FF	Highest Priority Pending Interrupt Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x001C	GICC_ABPR	RW[c]	0x00000003	Aliased Binary Point Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0020	GICC_AIAR	RO	0x000003FF	Aliased Interrupt Acknowledge Register
0x0024	GICC_AEOIR	WO	-	Aliased End of Interrupt Register
0x0028	GICC_AHPPIR	RO	0x000003FF	Aliased Highest Priority Pending Interrupt Register
0x00D0-0x00DC	GICC_APR0	RW	0x00000000	*Active Priority Register*
0x00E0	GICC_NSAPR0	RW[c]	0x00000000	*Non-secure Active Priority Register* on page 8-15
0x00FC	GICC_IIDR	RO	0x0102143B	*CPU Interface Identification Register* on page 8-15
0x1000	GICC_DIR	WO	-	Deactivate Interrupt Register, see the *ARM Generic Interrupt Controller Architecture Specification*

a. (S) = Secure
b. (NS) = Non-secure
c. This register is only accessible from a Secure access.

8.3.4 CPU Interface register descriptions

This section only describes registers whose implementation is specific to the Cortex-A7 MPCore processor. All other registers are described in the *ARM Generic Interrupt Controller Architecture Specification*. Table 8-8 on page 8-13 provides cross references to individual registers.

Active Priority Register

The GICC_APR0 characteristics are:

Purpose Provides support for preserving and restoring state in power-management applications.

Usage constraints. This register is banked to provide Secure and Non-secure copies. This ensures that Non-secure accesses do not interfere with Secure operation.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-8 on page 8-13.

The Cortex-A7 MPCore processor implements the GICC_APR0 according to the recommendations described in the ARM Generic Interrupt Controller Architecture Specification.

Table 8-9 shows the Cortex-A7 GICC_APR0 implementation.

Table 8-9 Active Priority Register implementation

Number of group priority bits	Preemption levels	Minimum legal value of Secure GICC_BPR	Minimum legal value of Non-secure GICC_BPR	Active Priority Registers implemented	View of Active Priority Registers for Non-secure accesses
5	32	2	3	GICC_APR0 [31:0]	GICC_NSAPR0 [31:16] appears as GICC_APR0 [15:0]

Non-secure Active Priority Register

The GICC_NSAPR0 characteristics are:

Purpose Provides support for preserving and restoring state in power-management applications.

Usage constraints. This register is only accessible from a Secure access.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-8 on page 8-13.

The Cortex-A7 MPCore processor implements the GICC_NSAPR0 according to the recommendations described in the *ARM Generic Interrupt Controller Architecture Specification*. It is consistent with the GICC_APR0 Register.

CPU Interface Identification Register

The GICC_IIDR characteristics are:

Purpose Provides information about the implementer and revision of the CPU interface.

Usage constraints. There are no usage constraints.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-8 on page 8-13.

Figure 8-6 shows the GICC_IIDR bit assignments.

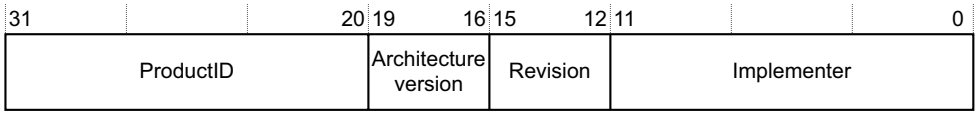


Figure 8-6 GICC_IIDR bit assignments

Table 8-10 shows the GICC_IIDR bit assignments.

Table 8-10 GICC_IIDR bit assignments

Bits	Name	Function
[31:20]	ProductID	Identifies the product ID: `0x010` Cortex-A7 MPCore product ID.
[19:16]	Architecture version	Identifies the architecture version of the GIC: `0x2` Version 2.0.
[15:12]	Revision	Identifies the revision number for the CPU interface: `0x1` Revision r0p1.
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the CPU interface. For an ARM implementation, these values are: **Bits [11:8]** = `0x4` The JEP106 continuation code of the implementer. **Bit [7]** Always 0. **Bits [6:0]** = `0x3B` The JEP106 identity code of the implementer.

8.3.5 Virtual interface control register summary

The virtual interface control registers are management registers. Configuration software on the Cortex-A7 MPCore processor must ensure they are accessible only by a hypervisor, or similar software.

Table 8-11 shows the register map for the virtual interface control registers. The offsets in this table are relative to the virtual interface control registers block base address as shown in Table 8-1 on page 8-3.

All the registers in Table 8-11 are word-accessible. Registers not described in this table are RAZ/WI.

Table 8-11 Virtual interface control register summary

Offset	Name	Type	Reset	Description
`0x000`	GICH_HCR	RW	`0x00000000`	Hypervisor Control Register, see the *ARM Generic Interrupt Controller Architecture Specification*
`0x004`	GICH_VTR	RO	`0x90000003`	*VGIC Type Register* on page 8-17
`0x008`	GICH_VMCR	RW	`0x004C0000`	Virtual Machine Control Register, see the *ARM Generic Interrupt Controller Architecture Specification*
`0x010`	GICH_MISR	RO	`0x00000000`	Maintenance Interrupt Status Register, see the *ARM Generic Interrupt Controller Architecture Specification*
`0x020`	GICH_EISR0	RO	`0x00000000`	End of Interrupt Status Register, see the *ARM Generic Interrupt Controller Architecture Specification*
`0x030`	GICH_ELSR0	RO	`0x0000000F`	Empty List register Status Register, see the *ARM Generic Interrupt Controller Architecture Specification*
`0x0F0`	GICH_APR0	RW	`0x00000000`	Active Priorities Register, see the *ARM Generic Interrupt Controller Architecture Specification*
`0x100`	GICH_LR0	RW	`0x00000000`	List Register 0, see the *ARM Generic Interrupt Controller Architecture Specification*

Table 8-11 Virtual interface control register summary (continued)

Offset	Name	Type	Reset	Description
0x104	GICH_LR1	RW	0x00000000	List Register 1, see the *ARM Generic Interrupt Controller Architecture Specification*
0x108	GICH_LR2	RW	0x00000000	List Register 2, see the *ARM Generic Interrupt Controller Architecture Specification*
0x10C	GICH_LR3	RW	0x00000000	List Register 3, see the *ARM Generic Interrupt Controller Architecture Specification*

8.3.6 Virtual Interface Control Register description

This section only describes registers whose implementation is specific to the Cortex-A7 MPCore processor. All other registers are described in the *ARM Generic Interrupt Controller Architecture Specification*.

VGIC Type Register

The GICH_VTR characteristics are:

Purpose Holds information about the number of priority bits, number of pre-emption bits, and number of List Registers implemented.

Usage constraints There are no usage constraints.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-11 on page 8-16.

Figure 8-7 shows the GICH_VTR bit assignments.

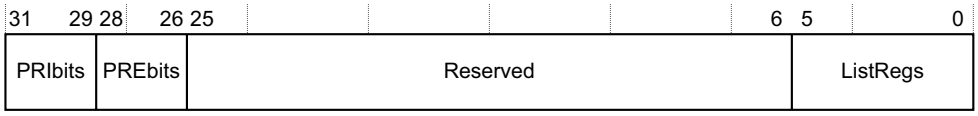


Figure 8-7 GICH_VTR bit assignments

Table 8-12 shows the GICH_VTR bit assignments.

Table 8-12 GICH_VTR bit assignments

Bits	Name	Function
[31:29]	PRIbits	Indicates the number of priority bits implemented, minus one: 0x4 5 bits of priority and 32 priority levels.
[28:26]	PREbits	Indicates the number of pre-emption bits implemented, minus one: 0x4 5 bits of pre-emption and 32 pre-emption levels.
[25:6]	-	Reserved, RAZ.
[5:0]	ListRegs	Indicates the number of implemented List Registers, minus one: 0x3 4 List Registers.

8.3.7 Virtual CPU interface register summary

The virtual CPU interface forwards virtual interrupts to a connected Cortex-A7 MPCore processor, subject to the normal GIC handling and prioritization rules. The virtual interface control registers control virtual CPU interface operation, and in particular, the virtual CPU interface uses the contents of the List registers to determine when to signal virtual interrupts. When a processor accesses the virtual CPU interface, the List registers are updated. For more information on virtual CPU interface, see the ARM Generic Interrupt Controller Architecture Specification.

Table 8-13 shows the register map for the virtual interface control registers. The offsets in this table are relative to the virtual interface control registers block base address as shown in Table 8-1 on page 8-3.

All the registers in Table 8-13 are word-accessible. Registers not described in this table are RAZ/WI.

Table 8-13 Virtual CPU interface register summary

Offset	Name	Type	Reset	Description
0x0000	GICV_CTLR	RW	0x00000000	VM Control Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0004	GICV_PMR	RW	0x00000000	VM Priority Mask Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0008	GICV_BPR	RW	0x00000002	VM Binary Point Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x000C	GICV_IAR	RO	0x000003FF	VM Interrupt Acknowledge Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0010	GICV_EOIR	WO	-	VM End Of Interrupt Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0014	GICV_RPR	RO	0x000000FF	VM Running Priority Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0018	GICV_HPPIR	RO	0x000003FF	VM Highest Priority Pending Interrupt Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x001C	GICV_ABPR	RW	0x00000003	VM Aliased Binary Point Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0020	GICV_AIAR	RO	0x000003FF	VM Aliased Interrupt Acknowledge Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0024	GICV_AEOIR	WO	-	VM Aliased End of Interrupt Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x0028	GICV_AHPPIR	RO	0x000003FF	VM Aliased Highest Priority Pending Interrupt Register, see the *ARM Generic Interrupt Controller Architecture Specification*
0x00D0	GICV_APR0	RW	0x00000000	*VM Active Priority Register* on page 8-19
0x00FC	GICV_IIDR	RO	0x0102143B	*VM CPU Interface Identification Register* on page 8-19
0x1000	GICV_DIR	WO	-	VM Deactivate Interrupt Register, see the *ARM Generic Interrupt Controller Architecture Specification*

8.3.8 Virtual CPU interface register description

This section only describes the registers whose implementation is specific to the Cortex-A7 MPCore processor. All other registers are described in the *ARM Generic Interrupt Controller Architecture Specification*.

VM Active Priority Register

The GICV_APR0 characteristics are:

Purpose For software compatibility, this register is present in the virtual CPU interface. However, in virtualized system, it is not used in the preserving and restoring state.

Usage constraints. Reading the content of this register and then writing the same values must not change any state because there is no requirement to preserve and restore state during a power down.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-13 on page 8-18.

The Cortex-A7 MPCore processor implements the GICV_APR0 as an alias of GICH_APR.

VM CPU Interface Identification Register

The GICV_IIDR characteristics are:

Purpose Provides information about the implementer and revision of the Virtual CPU interface.

Usage constraints There are no usage constraints.

Configurations Available if the GIC is implemented.

Attributes See the register summary in Table 8-13 on page 8-18.

The bit assignments for the VM CPU Interface Identification Register are identical to the corresponding register in the CPU Interface, see *CPU Interface Identification Register* on page 8-15.

Chapter 9
Generic Timer

This chapter describes the Generic Timer for the Cortex-A7 MPCore processor. It contains the following sections:

9.1 About the Generic Timer

The Generic Timer can schedule events and trigger interrupts based on an incrementing counter value. It provides:

- Generation of timer events as interrupt outputs.
- Generation of event streams.
- Support for Virtualization Extensions.

The Cortex-A7 MPCore Timer is compliant with the *ARM Architecture Reference Manual*.

This chapter only describes features that are specific to the Cortex-A7 MPCore implementation.

9.2 Generic timer functional description

The Cortex-A7 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-4 on page A-6 shows the signals that are the external interrupt output pins.

9.3 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 9-1 shows the System Timer registers.

Table 9-1 System Timer registers

CRn	Op1	CRm	Op2	Name	Reset	Width	Description
c14	0	c0	0	CNTFRQ	UNK	32-bit	Counter Frequency Register, see the *ARM Architecture Reference Manual*
-	0	c14	-	CNTPCT	UNK	64-bit	Counter Physical Count Register, see the *ARM Architecture Reference Manual*
c14	0	c1	0	CNTKCTL	-[a]	32-bit	Counter Non-secure PL1 Control Register, see the *ARM Architecture Reference Manual*
		c2	0	CNTP_TVAL	UNK	32-bit	Counter PL1 Physical Timer Value Register, see the *ARM Architecture Reference Manual*
			1	CNTP_CTL	-[b]	32-bit	Counter PL1 Physical Timer Control Register, see the *ARM Architecture Reference Manual*
		c3	0	CNTV_TVAL	UNK	32-bit	Counter PL1 Virtual Timer Value Register, see the *ARM Architecture Reference Manual*
			1	CNTV_CTL	-[b]	32-bit	Counter PL1 Virtual Timer Control Register, see the *ARM Architecture Reference Manual*
-	1	c14	-	CNTVCT	UNK	64-bit	Counter Virtual Count Register, see the *ARM Architecture Reference Manual*
	2			CNTP_CVAL	UNK	64-bit	Counter PL1 Physical Compare Value Register, see the *ARM Architecture Reference Manual*
	3			CNTV_CVAL	UNK	64-bit	Counter PL1 Virtual Compare Value Register, see the *ARM Architecture Reference Manual*
	4			CNTVOFF	UNK	64-bit	Counter Virtual Offset Register, see the *ARM Architecture Reference Manual*
c14	4	c1	0	CNTHCTL	-[c]	32-bit	Counter Non-secure PL2 Control Register, see the *ARM Architecture Reference Manual*
		c2	0	CNTHP_TVAL	UNK	32-bit	Counter Non-secure PL2 Physical Timer Value Register, see the *ARM Architecture Reference Manual*
			1	CNTHP_CTL	-[b]	32-bit	Counter Non-secure PL2 Physical Timer Control Register, see the *ARM Architecture Reference Manual*
-	6	c14	-	CNTHP_CVAL	UNK	64-bit	Counter Non-secure PL2 Physical Compare Value Register, see the *ARM Architecture Reference Manual*

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 10
Debug

This chapter describes debug, the registers that it uses and examples of how to use it. It contains the following sections:

10.1 About debug

This section gives an overview of debug and describes the debug components. The processor forms one component of a debug system. Figure 10-1 shows a typical system.

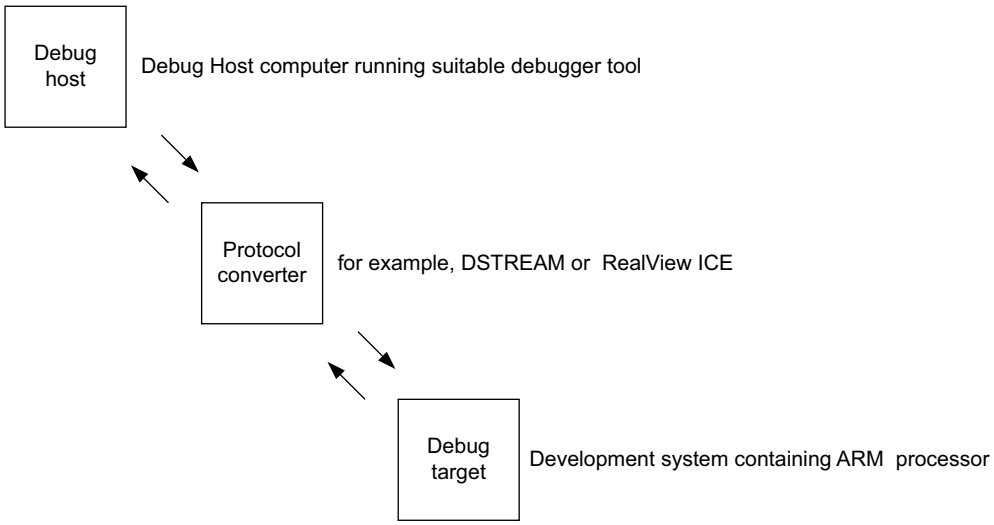


Figure 10-1 Typical debug system

This typical system has several parts:

- *Debug host*.
- *Protocol converter*.
- *Debug target*.
- *The debug unit*.

10.1.1 Debug host

The debug host is a computer, for example a personal computer, running a software debugger such as RealView Debugger. The debug host enables you to issue high-level commands such as setting breakpoint at a certain location, or examining the contents of a memory address.

10.1.2 Protocol converter

The debug host sends messages to the debug target using an interface such as Ethernet. However, the debug target typically implements a different interface protocol. A device such as DSTREAM or RealView ICE is required to convert between the two protocols.

10.1.3 Debug target

The debug target is the lowest level of the system. An example of a debug target is a development system with a test chip or a silicon part with a processor.

The debug target implements system support for the protocol converter to access the debug unit using the *Advanced Peripheral Bus* (APB) slave interface.

10.1.4 The debug unit

The processor debug unit assists in debugging software running on the processor. You can use the processor debug unit, in combination with a software debugger program, to debug:

- Application software.

- Operating systems.
- Hardware systems based on an ARM processor.

The debug unit enables you to:

- Stop program execution.
- Examine and alter process and coprocessor state.
- Examine and alter memory and input/output peripheral state.
- Restart the processor.

10.2 Debug register interfaces

The processor implements the ARMv7.1 Debug architecture and debug events as described in the *ARM Architecture Reference Manual*.

The Debug architecture defines a set of debug registers. The debug register interfaces provide access to these registers from:

- Software running on the processor, see *Processor interfaces*.
- An external debugger, see *External debug interface* on page 10-34.

This section describes:

- *Processor interfaces*.
- *Breakpoints and watchpoints*.
- *Effects of resets on debug registers*.

10.2.1 Processor interfaces

The processor has the following interfaces to the debug, performance monitor, and trace registers:

Debug registers

This interface is Baseline CP14, Extended CP14, and memory-mapped. You can access the debug register map using the APB slave port. See *External debug interface* on page 10-34.

Performance monitor

This interface is CP15 based and memory-mapped. You can access the performance monitor registers using the APB slave port. See *External debug interface* on page 10-34.

Trace registers

This interface is memory-mapped. See *External debug interface* on page 10-34.

10.2.2 Breakpoints and watchpoints

The processor supports six breakpoints, four watchpoints, and a standard *Debug Communications Channel* (DCC). Four of the breakpoints match only to virtual address and the other two match against either virtual address or context ID, or *Virtual Machine Identifier* (VMID). All the watchpoints can be linked to two breakpoints to enable a memory request to be trapped in a given process context.

10.2.3 Effects of resets on debug registers

The processor has the following reset signals that affect the debug registers:

nCOREPORESET This signal initializes all processor logic, including the debug, breakpoint, watchpoint logic, and performance monitors logic.

nCORERESET This signal initializes the processor logic, including the performance monitors logic.

nDBGRESET This signal resets the debug logic in the processor, including the breakpoint and watchpoint logic. Performance monitors logic is not affected.

10.3 Debug register summary

Table 10-1 shows the 32-bit or 64-bit wide CP14 interface registers, accessed by the MCR, MRC, MCCR, or MRRC instructions in the order of CRn, Op1, CRm, Op2.

Table 10-1 CP14 debug register summary

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
0	0x000	c0	0	c0	0	DBGDIDR	RO	*Debug Identification Register* on page 10-9
1	0x004	c0	0	c1	0	DBGDSCR internal view	RO	*Debug Status and Control Register, see the ARM Architecture Reference Manual*
2-4	0x008-0x010	-	-	-	-	-	-	Reserved
5	0x014	c0	0	c5	0	DBGDTRTX internal view	WO	*Target to Host Data Transfer Register, see the ARM Architecture Reference Manual*
						DBGDTRRX internal view	RO	Host to Target Data Transfer Register, see the ARM Architecture Reference Manual
6	0x018	c0	0	c6	0	DBGWFAR	-	UNK/SBZP
7	0x01C	c0	0	c7	0	DBGVCR	RW	Vector Catch Register, see the *ARM Architecture Reference Manual*
8	0x020	-	-	-	-	-	-	Reserved
9	0x024	-	-	-	-	DBGECR	RW	Event Catch Register, see the *ARM Architecture Reference Manual*
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 *ARM Architecture Reference Manual*
33	0x084	-	-	-	-	DBGITR	WO	*Instruction Transfer Register, see the ARM Architecture Reference Manual*
						DBGPCSR	RO	*Program Counter Sampling Register* on page 10-10
34	0x088	c0	0	c2	2	DBGDSCR external view	RW	Debug Status and Control Register, see the *ARM Architecture Reference Manual*
35	0x08C	c0	0	c3	2	DBGDTRTX external view	RW	*Target to Host Data Transfer Register, see the ARM Architecture Reference Manual*

Table 10-1 CP14 debug register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
36	0x090	-	-	-	-	DBGDRCR	WO	*Debug Run Control Register* on page 10-11
37	0x094	-	-	-	-	DBGEACR	RW	*Debug External Auxiliary Control Register* on page 10-12
38-39	0x098-0x09C	-	-	-	-	-	-	Reserved
40	0x0A0	-	-	-	-	DBGPCSR	RO	*Program Counter Sampling Register* on page 10-10
41	0x0A4	-	-	-	-	DBGCIDSR	RO	*Context ID Sampling Register, see the ARM Architecture Reference Manual*
42	0x0A8	-	-	-	-	DBGVIDSR	RO	*Virtualization ID Sampling Register, see the ARM Architecture Reference Manual*
43-63	0x0AC-0x0FC	-	-	-	-	-	-	Reserved
64-69	0x100-0x114	c0	0	c0-c5	4	DBGBVRn	RW	*Breakpoint Value Registers* on page 10-13
70-71	0x118-0x11C	-	-	-	-	-	-	Reserved
72-79	0x120-0x13C	-	-	-	-	-	-	Reserved
80-85	0x140-0x154	c0	0	c0-c5	5	DBGBCRn	RW	*Breakpoint Control Registers* on page 10-14
86-95	0x158-0x17C	-	-	-	-	-	-	Reserved
96-99	0x180-0x18C	c0	0	c0-c3	6	DBGWVRn	RW	*Watchpoint Value Registers* on page 10-16
100-111	0x190-0x1BC	-	-	-	-	-	-	Reserved
112-115	0x1C0-0x1CC	c0	0	c0-c3	7	DBGWCRn	RW	*Watchpoint Control Registers* on page 10-17
116-127	0x1D0-0x1FC	-	-	-	-	-	-	Reserved
128	0x200	-	-	-	-	-	-	Reserved
129-147	0x204-0x24C	-	-	-	-	-	-	Reserved
148-149	0x250-0x254	c1	0	c4-c5	1	DBGBXVRn	RW	*Breakpoint Extended Value Registers* on page 10-20
150-191	0x258-0x2FC	-	-	-	-	-	-	Reserved
192	0x300	c1	0	c0	4	DBGOSLAR	WO	*OS Lock Access Register* on page 10-21
193	0x304	c1	0	c1	4	DBGOSLSR	RO	*OS Lock Status Register* on page 10-21
194	0x308	-	-	-	-	-	-	Not implemented

Table 10-1 CP14 debug register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
195	0x30C	-	-	-	-	-	-	Reserved
196	0x310	c1	0	c4	4	DBGPRCR	RW	*Device Power-down and Reset Control Register* on page 10-22
197	0x314	-	-	-	-	DBGPRSR	RO	*Device Powerdown and Reset Status Register, see the ARM Architecture Reference Manual*
198-255	0x318-0x03C	-	-	-	-	-	-	Reserved
256-511	0x400-0x7FC	-	-	-	-	-	-	Reserved
512-575	0x800-0x8FC	-	-	-	-	-	-	Reserved
576-831	0x900-0xCFC	-	-	-	-	-	-	Reserved
832-895	0xD00-0xDFC	-	-	-	-	Processor ID registers	RO	Processor ID registers, see the *ARM Architecture Reference Manual*
896-927	0xE00-0xE7C	-	-	-	-	-	-	Reserved
928-957	0xE80-0xEF4	-	-	-	-	-	-	Reserved
958	0xEF8	-	-	-	-	DBGITMISCOUT	WO	*Integration Miscellaneous Signals Register* on page 10-26
959	0xEFC	-	-	-	-	DBGITMISCIN	RO	*Integration Miscellaneous Signals Input Register* on page 10-27
960	0xF00	-	-	-	-	DBGITCTRL	RW	*Integration Mode Control Register* on page 10-27
961-999	0xF04-0xF9C	-	-	-	-	-	-	Reserved
1000	0xFA0	c7	0	c8	6	DBGCLAIMSET	RW	*Claim Tag Set Register* on page 10-28
1001	0xFA4	c7	0	c9	6	DBGCLAIMCLR	RW	*Claim Tag Clear Register* on page 10-29
1002-1003	0xFA8-0xFAC	-	-	-	-	-	-	Reserved
1004	0xFB0	-	-	-	-	DBGLAR	WO	Lock Access Register, see the *ARM Architecture Reference Manual*
1005	0xFB4	-	-	-	-	DBGLSR	RO	*Lock Status Register, see the ARM Architecture Reference Manual*
1006	0xFB8	c7	0	c14	6	DBGAUTHSTATUS	RO	Authentication Status Register, see the *ARM Architecture Reference Manual*
1007	0xFBC	-	-	-	-	-	-	Reserved
1008	0xFC0	c7	0	c0	7	DBGDEVID2	RO	UNK

Table 10-1 CP14 debug register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
1009	0xFC4	c7	0	c1	7	DBGDEVID1	RO	*Debug Device ID Register 1* on page 10-29
1010	0xFC8	c7	0	c2	7	DBGDEVID	RO	*Debug Device ID Register* on page 10-30
1011	0xFCC	-	-	-	-	DBGDEVTYPE	RO	Device Type Register, see the *ARM Architecture Reference Manual*
1012	0xFD0	-	-	-	-	DBGPID4	RO	*Debug Peripheral Identification Registers* on page 10-31
1013-1015	0xFD4-0xFDC	-	-	-	-	DBGPID5-7	-	Reserved
1016	0xFE0	-	-	-	-	DBGPID0	RO	*Debug Peripheral Identification Registers* on page 10-31
1017	0xFE4	-	-	-	-	DBGPID1	RO	
1018	0xFE8	-	-	-	-	DBGPID2	RO	
1019	0xFEC	-	-	-	-	DBGPID3	RO	
1020	0xFF0	-	-	-	-	DBGCID0	RO	*Debug Component Identification Registers* on page 10-32
1021	0xFF4	-	-	-	-	DBGCID1	RO	
1022	0xFF8	-	-	-	-	DBGCID2	RO	
1023	0xFFC	-	-	-	-	DBGCID3	RO	
-	-	c0	0	c1	0	DBGDSCR internal view	RO	Debug Status and Control Register, see the *ARM Architecture Reference Manual*
-	-	c0	0	c5	0	DBGDTRRX internal view	WO	Host to Target Data Transfer, see the *ARM Architecture Reference Manual*
						DBGDTRTX internal view	RO	Target to Host Transfer, see the *ARM Architecture Reference Manual*
-	-	c1	0	c0	0	DBGDRAR (MRC)	RO	*Debug ROM Address Register* on page 10-19
		c1	0			DBGDRAR (MRRC)	RO	*Debug ROM Address Register* on page 10-19
-	-	c1	0	c3	4	DBGOSDLR	RW	OS Double Lock Register, see the *ARM Architecture Reference Manual*
-	-	c2	0	c0	0	DBGDSAR (MRC)	RO	*Debug Self Address Offset Register* on page 10-25
		c2	0			DBGDSAR (MRRC)	RO	*Debug Self Address Offset Register* on page 10-25

10.4 Debug register descriptions

This section describes the debug registers. Table 10-1 on page 10-5 provides cross references to individual registers.

10.4.1 Debug Identification Register

The DBGDIDR characteristics are:

Purpose Specifies:
- Which version of the Debug architecture is implemented.
- Some features of the debug implementation.

Usage constraints There are no usage constraints. See *Debug Device ID Register 1* on page 10-29 and *Debug Device ID Register* on page 10-30 for more information about the debug implementation.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-2 shows the DBGDIDR bit assignments.

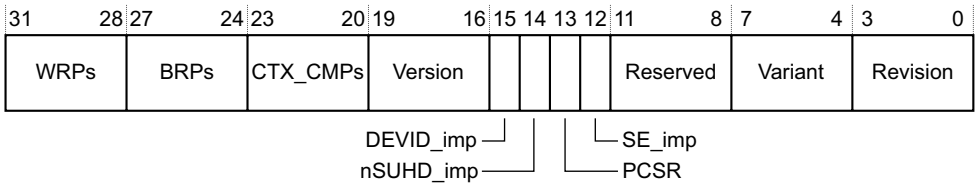


Figure 10-2 DBGDIDR bit assignments

Table 10-2 shows the DBGDIDR bit assignments.

Table 10-2 DBGDIDR bit assignments

Bits	Name	Function
[31:28]	WRPs	Indicates the number of *Watchpoint Register Pairs* (WRPs) implemented: `0x3` The processor implements 4 WRPs.
[27:24]	BRPs	Indicates the number of *Breakpoint Register Pairs* (BRPs) implemented: `0x5` The processor implements 6 BRPs.
[23:20]	CTX_CMPs	Indicates the number of BRPs that can be used for Context ID comparison: `0x1` The processor implements 2 breakpoints with Context ID comparison.
[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.
[13]	PCSR_imp	Indicates if the *Program Counter Sampling Register* (DBGPCSR) implemented as register 33: **1** DBGPCSR is implemented as register 33.

Table 10-2 DBGDIDR bit assignments (continued)

Bits	Name	Function
[12]	SE_imp	Security Extensions implemented bit: **1** The processor implements Security Extensions.
[11:8]	-	Reserved.
[7:4]	Variant	This field indicates the variant number of the processor. This number is incremented on functional changes. The value matches bits [23:20] of the CP15 Main ID Register. See *Main ID Register* on page 4-26 for more information.
[3:0]	Revision	This field indicates the revision number of the processor. This number is incremented on bug fixes. The value matches bits [3:0] of the CP15 Main ID Register. See *Main ID Register* on page 4-26 for more information.

10.4.2 Program Counter Sampling Register

The DBGPCSR characteristics are:

Purpose Enables a debugger to sample the *Program Counter* (PC).

Usage constraints ARM deprecates reading a PC sample through register 33 when the DBGPCSR is also implemented as register 40.

Configurations DBGPCSR is implemented as both debug register 33 and 40.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-3 shows the DBGPCSR bit assignments.

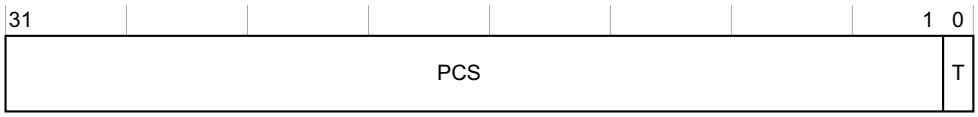


Figure 10-3 DBGPCSR bit assignments

Table 10-3 shows the DBGPCSR bit assignments.

Table 10-3 DBGPCSR bit assignments

Bits	Name	Function
[31:1]	PCS	Program Counter sample value. The sampled value of bits[31:1] of the PC. The sampled value is either the virtual address of an instruction, or the virtual address of an instruction address plus an offset that depends on the processor instruction set state. DBGDEVID1.PCSROffset indicates whether an offset is applied to the sampled addresses.
[0]	T	This bit indicates whether the sampled address is an ARM instruction, or a Thumb or ThumbEE instruction: **0** If DBGPCSR[1] is 0, the sampled address is an ARM instruction. **1** The sampled address is a Thumb or ThumbEE instruction If T is 0 then DBGPCSR[1] is 0, (DBGPCSR[31:2] << 2) is the address of the sampled ARM instruction If T is 1, (DBGPCSR[31:1] << 1) is the address of the sampled Thumb or ThumbEE instruction..

10.4.3 Debug Run Control Register

The DBGDRCR characteristics are:

Purpose Software uses this register to:

- Request the processor to enter or exit Debug state
- Clear to 0 the sticky exception bits in the DBGDSCR, see the *ARM Architecture Reference Manual* for information on the DBGDSCR.

Usage constraints The DBGDRCR is accessible when the processor is powered off. DBGDRCR is not accessible on extended CP14 interface.

Configurations Required in all implementations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-4 shows the DBGDRCR bit assignments.

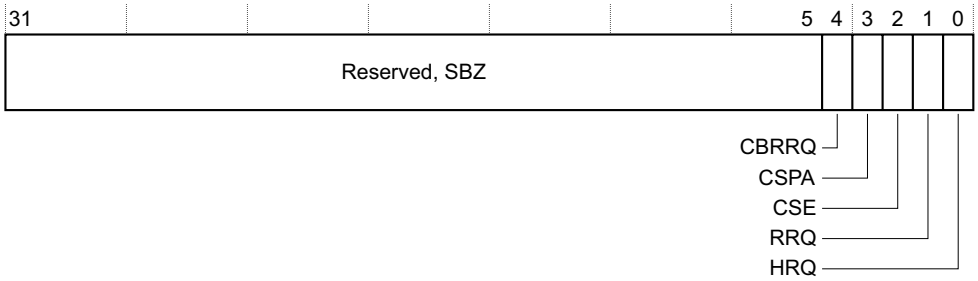


Figure 10-4 DBGDRCR bit assignments

Table 10-4 shows the DBGDRCR bit assignments.

Table 10-4 DBGDRCR bit assignments

Bits	Name	Function
[31:5]	-	Reserved, SBZ.
[4]	CBRRQ	Cancel Bus Requests Request. The actions on writing to this bit are: **0** No action. **1** Request cancel of pending accesses.
[3]	CSPA	Clear Sticky Pipeline Advance bit. This bit sets the DBGDSCR.PipeAdv bit to 0. The actions on writing to this bit are: **0** No action. **1** Sets the DBGDSCR.PipeAdv bit to 0. When the processor is powered down, it is UNPREDICTABLE whether a write of 1 to this bit sets DBGDSCR.PipeAdv to 0.

Table 10-4 DBGDRCR bit assignments (continued)

Bits	Name	Function
[2]	CSE	Clear Sticky Exceptions bits. This bit sets the DBGDSCR sticky exceptions bits to 0. The actions on writing to this bit are: **0** No action. **1** Sets the DBGDSCR[8:6] to `0b000`. See the *ARM Architecture Reference Manual* for more information on the DBGDSCR.
[1]	RRQ	Restart request. The actions on writing to this bit are: **0** No action. **1** Request exit from Debug state. Writing 1 to this bit requests that the processor exits Debug state. This request is held until the processor exits Debug state. After the request has been made, the debugger can poll the DBGDSCR.RESTARTED bit until it reads as 1. The processor ignores writes to this bit if it is in Non-debug state.
[0]	HRQ	Halt request. The actions on writing to this bit are: **0** No action. **1** Request exit from Debug state. Writing 1 to this bit requests that the processor enters Debug state. This request is held until the processor enters Debug state. After the request has been made, the debugger can poll the DBGDSCR.HALTED bit until it reads 1. The processor ignores writes to this bit if it is already in Debug state.

10.4.4 Debug External Auxiliary Control Register

The DBGEACR characteristics are:

Purpose Provides implementation-defined configuration and control options.

Usage constraints The DBGEACR is accessible when the CPU is powered off. The DBGEACR is not accessible from the CP14 interface.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-5 shows the DBGEACR bit assignments.

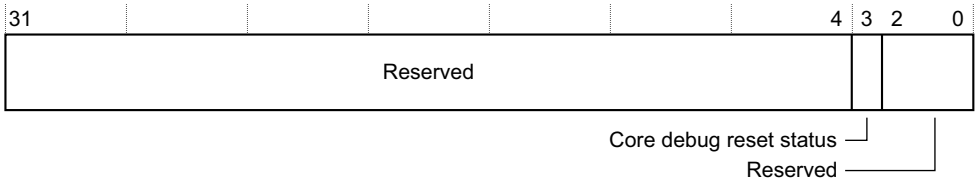


Figure 10-5 DBGEACR bit assignments

Table 10-5 shows the DBGEACR bit assignments.

Table 10-5 DBGEACR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, RAZ/WI.
[3]	Core debug reset status	Read-only status bit that reflects the current reset state of the debug logic in the CPU power domain: **0** Debug logic in CPU power domain is not in reset state. **1** Debug logic in CPU power domain is currently in reset state.
[2:0]	-	Reserved, RAZ/WI.

10.4.5 Breakpoint Value Registers

The DBGBVR characteristics are:

Purpose Holds a value for use in breakpoint matching, either an instruction address or a Context ID.

Usage constraints Used in conjunction with a DBGBCR, see *Breakpoint Control Registers* on page 10-14. Each DBGBVR is associated with a DBGBCR to form a *Breakpoint Register Pair* (BRP). DBGBVRn is associated with DBGBCRn to form breakpoint n.

Configurations The processor implements 6 BRPs, and is specified by the DBGDIDR.BRPs field, see *Debug Identification Register* on page 10-9.

Attributes See the register summary in Table 10-1 on page 10-5. The debug logic reset value of a DBGBVR is UNKNOWN.

Figure 10-6 shows the DBGBVR bit assignments.

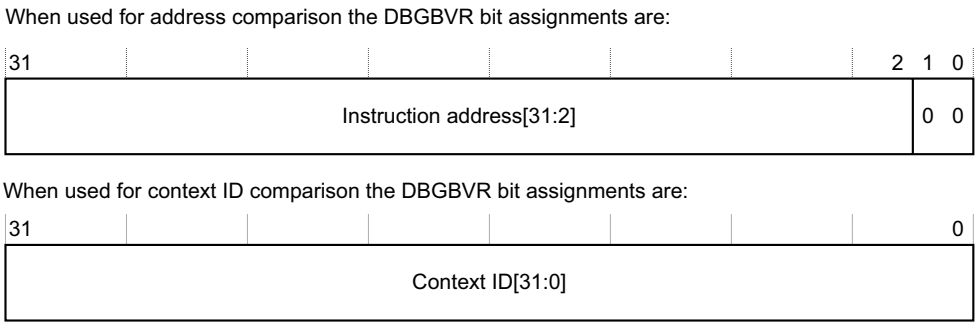


Figure 10-6 DBGBVR bit assignments

Table 10-6 shows the DBGBVR bit assignments when the register is used for address comparison.

Table 10-6 DBGBVR bit assignments when register is used for address comparison

Bits	Name	Function
[31:2]	Instruction address[31:2]	This field indicates bits[31:2] of the address value for comparison
[1:0]	-	This field must be written as `0b00`, otherwise the generation of breakpoint debug events by this breakpoint is UNPREDICTABLE

Table 10-7 shows the DBGBVR bit assignments when the register is used for Context ID comparison.

Table 10-7 DBGBVR bit assignments when register is used for Context ID comparison

Bits	Name	Function
[31:0]	Context ID[31:0]	This field indicates bits[31:0] of the Context ID value for comparison

10.4.6 Breakpoint Control Registers

The DBGBCR characteristics are:

Purpose Holds control information for a breakpoint.

Usage constraints Used in conjunction with a DBGBVR, see *Breakpoint Value Registers* on page 10-13. Each DBGBVR is associated with a DBGBCR to form a *Breakpoint Register Pair* (BRP). DBGBVRn is associated with DBGBCRn to form breakpoint n.

Configurations The processor implements 6 BRPs, and is specified by the DBGDIDR.BRPs field, see *Debug Identification Register* on page 10-9.

Attributes See the register summary in Table 10-1 on page 10-5. The debug logic reset value of a DBGBCR is UNKNOWN.

Figure 10-7 shows the DBGBCR bit assignments.

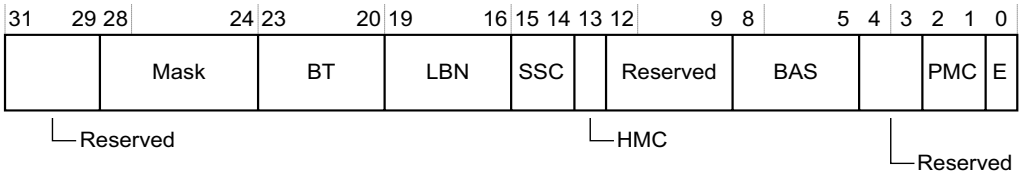


Figure 10-7 DBGBCR bit assignments

Table 10-8 shows the DBGBCR bit assignments.

Table 10-8 DBGBCR bit assignments

Bits	Name	Function
[31:29]	-	Reserved.
[28:24]	Mask	Address mask. The processor does not support address range masking.
[23:20]	BT	Breakpoint Type. This field controls the behavior of Breakpoint debug event generation. This includes the meaning of the value held in the associated DBGBVR, indicating whether it is an instruction address match or mismatch or a Context match. It also controls whether the breakpoint is linked to another breakpoint. See the *ARM Architecture Reference Manual* for the meaning of bits[23:20].

Table 10-8 DBGBCR bit assignments (continued)

Bits	Name	Function
[19:16]	LBN	Linked Breakpoint Number. If this breakpoint is programmed for Linked instruction address match or mismatch then this field must be programmed with the number of the breakpoint that holds the Context match to be used in the combined instruction address and Context comparison. Otherwise, this field must be programmed to `0b0000`. Reading this register returns an UNKNOWN value for this field, and the generation of debug events is UNPREDICTABLE, if either: • This breakpoint is not programmed for Linked instruction address match or mismatch and this field is not programmed to `0b0000`. • This breakpoint is programmed for Linked instruction address match or mismatch and the breakpoint indicated by this field does not support Context matching or is not programmed for Linked Context ID match. See the *ARM Architecture Reference Manual* for more information.
[15:14]	SSC	Security State Control. This field enables the watchpoint to be conditional on the security state of the processor. This field is used with the *Hyp Mode Control* (HMC), and *Privileged Mode Control* (PMC), fields. See the *ARM Architecture Reference Manual* for possible values of the fields, and the mode and security states that can be tested. This field must be programmed to `b00` if DBGBCR.BT is programmed for Linked Context match. If this is not done, the generation of debug events by this breakpoint is UNPREDICTABLE. **Note** When this field is set to a value other than `b00`, the SSC field controls the processor security state in which the access matches, not the required security attribute of the access.
[13]	HMC	Hyp Mode Control. This field is used with the SSC and PMC fields. See the *ARM Architecture Reference Manual* for possible values of the fields, and the access modes 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.
[12:9]	-	Reserved.
[8:5]	BAS	Byte Address Select. This field enables match or mismatch comparisons on only certain bytes of the word address held in the DBGBVR. The operation of this field depends also on: • The breakpoint type field being programmed for instruction address match or mismatch. • The Address range mask field being programmed to `0b00000`, no mask. • The instruction set state of the processor, indicated by the CPSR.J and CPSR.T bits. This field must be programmed to `0b1111` if either: • The DBGBVR.BT is programmed for Linked or Unlinked Context ID match. • DBGBCR.MASK is programmed to a value other than `0b00000`. If this is not done, the generation of Breakpoint debug events is UNPREDICTABLE.

Table 10-8 DBGBCR bit assignments (continued)

Bits	Name	Function
[4:3]	-	Reserved.
[2:1]	PMC	Privileged Mode Control. This field enables breakpoint matching conditional on the mode of the processor: This field is used with the SSC and HMC fields. See the *ARM Architecture Reference Manual* for possible values of the fields, and the mode and security states that can be tested. This field must be programmed to `0b11` if DBGBCR.BT is programmed for Linked Context ID match. If this is not done, the generation of debug events by this breakpoint is UNPREDICTABLE. **Note** Bits[2:1] has no effect for accesses made in Hyp mode.
[0]	E	Breakpoint Enable. This bit enables the breakpoint: **0** Breakpoint disabled. **1** Breakpoint enabled. A breakpoint never generates a debug event when it is disabled. **Note** The value of the DBGBCR.E bit is UNKNOWN on reset. A debugger must ensure that DBGBCR.E has a defined value before it programs DBGDSCR[15:14] to enable debug.

10.4.7 Watchpoint Value Registers

The DBGWVR characteristics are:

Purpose Holds a data address value for use in watchpoint matching. The address used must be the virtual address of the data.

Usage constraints Used in conjunction with a DBGWCR to form a watchpoint, see *Watchpoint Control Registers* on page 10-17. Each DBGWVR is associated with a DBGWCR to form a *Watchpoint Register Pair* (WRP). DBGWVRn is associated with DBGWCRn to form watchpoint n.

Configurations The processor implements 4 WRPs, and is specified by the DBGDIDR.WRPs field, see *Debug Identification Register* on page 10-9.

Attributes See the register summary in Table 10-1 on page 10-5. The debug logic reset value of a DBGWVR is UNKNOWN.

Figure 10-8 shows the DBGWVR bit assignments.

Figure 10-8 DBGWVR bit assignments

Table 10-9 shows the DBGWVR bit assignments.

Table 10-9 DBGWVR bit assignments

Bits	Name	Function
[31:2]	Data address[31:2]	This field indicates bits[31:2] of the address value for comparison
[1:0]	-	RAZ on reads and SBZP for writes

10.4.8 Watchpoint Control Registers

The DBGWCR characteristics are:

Purpose Holds control information for a watchpoint.

Usage constraints Used in conjunction with a DBGWVR, see *Watchpoint Value Registers* on page 10-16. Each DBGWCR is associated with a DBGWVR to form a *Watchpoint Register Pair* (WRP). DBGWCRn is associated with DBGWVRn to form watchpoint n.

Configurations The processor implements 4 WRPs, and is specified by the DBGDIDR.WRPs field, see *Debug Identification Register* on page 10-9.

Attributes See the register summary in Table 10-1 on page 10-5. The debug logic reset value of a DBGWCR is UNKNOWN.

Figure 10-9 shows the DBGWCR bit assignments.

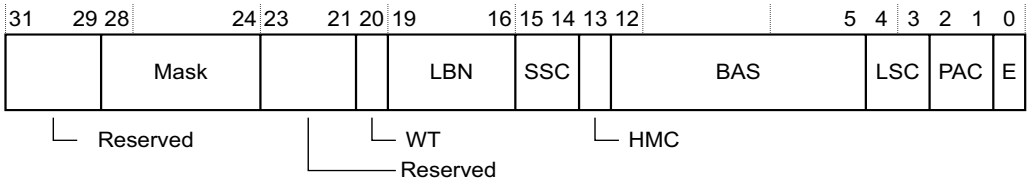


Figure 10-9 DBGWCR bit assignments

Table 10-10 shows the DBGWCR bit assignments.

Table 10-10 DBGWCR bit assignments

Bits	Name	Function
[31:29]	-	Reserved.
[28:24]	Mask	Address 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 1 and 2 are reserved. Therefore, the meaning of Watchpoint Address range mask values are: `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, 31 bits masked. See the *ARM Architecture Reference Manual* for the meanings of watchpoint address range mask values.
[23:21]	-	Reserved.
[20]	WT	Watchpoint Type. This bit is set to 1 to link the watchpoint to a breakpoint to create a linked watchpoint that requires both data address matching and Context matching: **0** Linking disabled. **1** Linking enabled. When this bit is set to 1 the linked breakpoint number field indicates the breakpoint to which this watchpoint is linked. See the *ARM Architecture Reference Manual* for more information.
[19:16]	LBN	Linked Breakpoint Number. If this watchpoint is programmed with 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`. Reading this register returns an UNKNOWN value for this field, and the generation of Watchpoint debug events is UNPREDICTABLE, if either: • This watchpoint does not have linking enabled and this field is not programmed to `0b0000`. • This watchpoint has linking enabled and the breakpoint indicated by this field does not support Context matching, is not programmed for Context matching, or does not exist. See the *ARM Architecture Reference Manual* for more information.
[15:14]	SSC	Security State Control. This field enables the watchpoint to be conditional on the security state of the processor. This field is used with the *Hyp Mode Control* (HMC) and *Privileged Access Control* (PAC) fields. See the *ARM Architecture Reference Manual* for possible values of the fields, and the access modes and security states that can be tested.
[13]	HMC	Hyp Mode Control. This field is used with the *Security State Control* (SSC) and PAC fields. The value of DBGWCR.PAC has no effect for accesses made in Hyp mode. See the *ARM Architecture Reference Manual* for possible values of the fields, and the access modes and security states that can be tested.

Table 10-10 DBGWCR bit assignments (continued)

Bits	Name	Function
[12:5]	BAS	Byte Address Select. The processor implements an 8-bit Byte address select field, DBGWCR[12:5]. A DBGWVR is programmed with a word-aligned address. This field enables thewatchpoint to hit only if certain bytes of the addressed word are accessed. The watchpoint hits if an access hits any byte being watched, even if: • The access size is larger than the size of the region being watched. • The access is unaligned, and the base address of the access is not in the same word of memory as the address in the DBGWVR. See the *ARM Architecture Reference Manual* for more information.
[4:3]	LSC	Load/store access control. This field enables watchpoint matching conditional on the type of access being made: `0b00` Reserved. `0b01` Match on any load, Load-Exclusive, or swap. `0b10` Match on any store, Store-Exclusive or swap `0b11` Match on either type of access.
[2:1]	PAC	Privileged Access Control. This field enables watchpoint matching conditional on the mode of the processor. This field is used with the SSC and HMC fields. See the *ARM Architecture Reference Manual* for possible values of the fields, and the access modes and security states that can be tested.
[0]	E	Watchpoint Enable. This bit enables the watchpoint: **0** Watchpoint disabled. **1** Watchpoint enabled. A watchpoint never generates a Watchpoint debug event when it is disabled. For more information about possible watchpoints values, see *Watchpoint Value Registers* on page 10-16. **Note** The value of DBGWCR.E is UNKNOWN on reset. A debugger must ensure that DBGWCR.E has a defined value before it programs DBGDSCR[15:14] to enable debug.

10.4.9 Debug ROM Address Register

The DBGDRAR characteristics are:

Purpose Defines the base address of a ROM table, that locates and describes the debug components in the system.

Usage constraints There are no usage constraints.

Configurations The DBGDRAR 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 10-1 on page 10-5. This register is only visible in the CP14 interface, and therefore does not have a memory offset.

Figure 10-10 shows the DBGDRAR bit assignments as a 32-bit register.

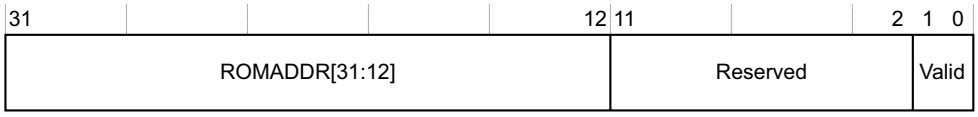


Figure 10-10 DBGDRAR 32-bit assignments

Figure 10-11 shows the DBGDRAR bit assignments as a 64-bit register.

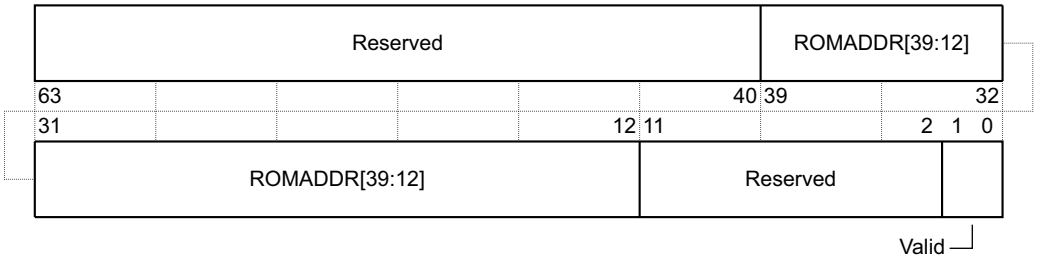


Figure 10-11 DBGDRAR 64-bit assignments

Table 10-11 shows the DBGDRAR bit assignments.

Table 10-11 DBGDRAR bit assignments

Bits	Name	Function
[63:40]	-	Reserved.
[39:32]	ROMADDR[39:32]	Bits[39:32] of the 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: `0b00` ROM table address is not valid. `0b11` ROM table address is valid. **Note** **ROMADDRV** must be set to 1 if **ROMADDR[39:12]** is set to a valid value.

10.4.10 Breakpoint Extended Value Registers

The DBGBXVR characteristics are:

Purpose Holds a value for use in VMID matching for BRP that supports Context ID comparison.

Usage constraints Used in conjunction with a DBGBCR, see *Breakpoint Control Registers* on page 10-14, and *Breakpoint Value Registers* on page 10-13. Each DBGBXVR is associated with a DBGBCR and DBGBVR to form a *Breakpoint Register Pair* (BRP). DBGBXVRn is associated with DBGBCRn and DBGBVRn to form BRPn.

Configurations The processor implements 2 BRPs that can be used for Context ID comparisons, and is specified by the DBGDIDR.CTX_CMPs field, see *Debug Identification Register* on page 10-9.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-12 on page 10-21 shows the DBGBXVR bit assignments.

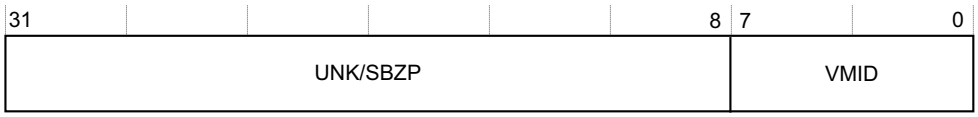


Figure 10-12 DBGBXVR bit assignments

Table 10-12 shows the DBGBXVR bit assignments.

Table 10-12 DBGBXVR bit assignments

Bits	Name	Function
[31:8]	-	UNK/SBZP
[7:0]	VMID	VMID value. Used to compare with the *Virtual Machine ID* (VMID) field, held in the *Virtualization Translation Table Base Register* (VTTBR). See the *ARM Architecture Reference Manual* for more information.

10.4.11 OS Lock Access Register

The DBGOSLAR characteristics are:

Purpose Provides a lock for the debug registers.

Usage constraints Use DBGOSLSR to check the current status of the lock, see *OS Lock Status Register*.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-13 shows the DBGOSLAR bit assignments.

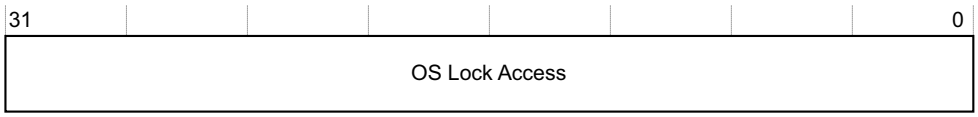


Figure 10-13 DBGOSLAR bit assignments

Table 10-13 shows the DBGOSLAR bit assignments.

Table 10-13 DBGOSLAR bit assignments

Bits	Name	Function
[31:0]	OS Lock Access	Writing the key value `0xC5ACCE55` to this field locks the debug registers. Writing any other value to this register unlocks the debug registers if they are locked. See the *ARM Architecture Reference Manual* for more information.

10.4.12 OS Lock Status Register

The DBGOSLSR characteristics are:

Purpose Provides status information for the OS Lock. Software can read this register to detect whether the OS Save and Restore mechanism is implemented. If it is not implemented, the read of DBGOSLSR.OSLM returns zero.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-14 shows the DBGOSLSR bit assignments.

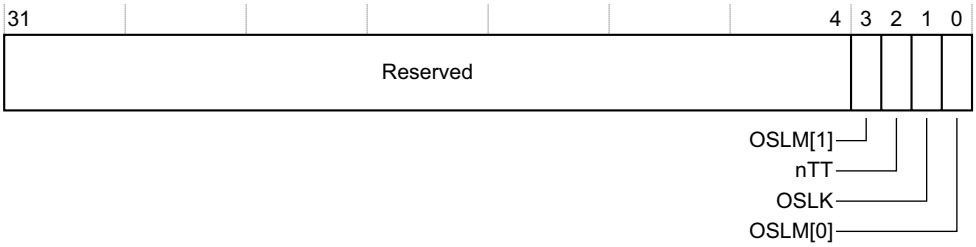


Figure 10-14 DBGOSLSR bit assignments

Table 10-14 shows the DBGOSLSR bit assignments.

Table 10-14 DBGOSLSR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, UNK.
[3]	OSLM[1]	OS Lock Model implemented bit. This field identifies the form of OS Save and Restore mechanism implemented: `0b10` The processor implements the OS Lock Model but does not implement DBGOSSRR. **Note** This field splits across the two non-contiguous bits in the register.
[2]	nTT	This bit is always RAZ. It indicates that a 32-bit access is needed to write the key to the OS Lock Access Register.
[1]	OSLK	This bit indicates the status of the OS Lock: **0** Lock not set. **1** Lock set. The OS Lock is set or cleared by writing to the DBGOSLAR, see *OS Lock Access Register* on page 10-21. The OS Lock is set to 1 on a core power up reset. Setting the OS Lock restricts access to Debug registers. See the *ARM Architecture Reference Manual* for more information.
[0]	OSLM[0]	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. **Note** This field splits across the two non-contiguous bits in the register.

10.4.13 Device Power-down and Reset Control Register

The DBGPRCR characteristics are:

Purpose Controls processor functionality related to reset and power-down.

Usage constraints 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.

Configurations Required in all configurations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-15 shows the DBGPRCR bit assignments.

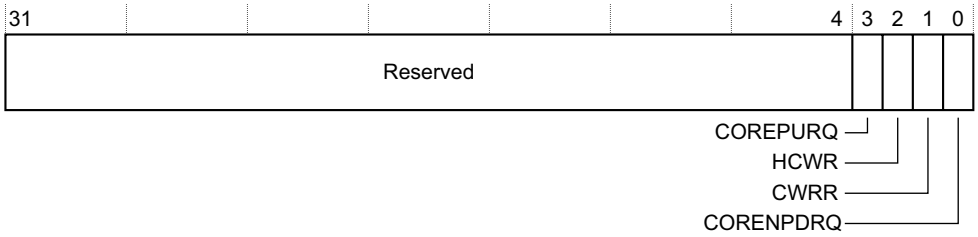


Figure 10-15 DBGPRCR bit assignments

Table 10-15 shows the DBGPRCR bit assignments.

Table 10-15 DBGPRCR bit assignments

Bits	Name	Function
[31:4]	-	Reserved, UNK/SBZP.
[3]	COREPURQ	Power-up request bit. This bit enables a debugger to request that the power controller powers up the processor, enabling access to debug registers in the processor power domain: **0** **DBGPWRUPREQ** is LOW, this is the reset value. **1** **DBGPWRUPREQ** is HIGH. This bit is only defined for the memory-mapped and external debug interfaces. **Note** This bit never affects system power-up, because when implemented it resets to 0. For accesses to DBGPRCR from CP14, this bit is UNK/SBZP. See the *ARM Architecture Reference Manual* for more information.

Table 10-15 DBGPRCR bit assignments (continued)

Bits	Name	Function
[2]	HCWR	Hold reset bit. Writing 1 to this bit means the non-debug logic of the processor is held in reset after a power-up or warm reset: **0** Do not hold the non-debug logic reset on power-up or warm reset. **1** Hold the non-debug logic of the processor in reset on power-up or warm reset. The processor is held in this state until this bit is set to 0 **Note** This bit never affects system power-up, because when implemented it resets to 0. For accesses to DBGPRCR from CP14, this bit is UNK/SBZP. See the *ARM Architecture Reference Manual* for more information.
[1]	CWRR	Reset request bit. Writing 1 to this bit issues a request for a warm reset: **0** No action. **1** Request internal reset using the memory mapped interface or CP14. Reads from this bit are UNKNOWN, and writes to this bit from the memory-mapped or external debug interface are ignored when any of the following apply: • The core power domain is off. • DBGPRSR.DLK, OS Double Lock status bit, is set to 1. • For the external debug interface, the OS lock is set. See the *ARM Architecture Reference Manual* for more information.
[0]	CORENPDRQ	Hold power-up request bit. When set to 1, the **DBGNOPWRDWN** output signal is HIGH. This output is connected to the system power controller and is interpreted as a request to operate in emulate mode. In this mode, the processor that includes ETM are not actually powered down when requested by software or hardware handshakes: **0** **DBGNOPWRDWN** is LOW. This is the reset value. **1** **DBGNOPWRDWN** is HIGH. This bit is UNKNOWN on reads and ignores writes when any of the following apply: • The core power domain is powered down. If the CORENPDRQ bit is 1, it loses this value through the power down. • DBGPRSR.DLK, OS Double Lock status bit is set to 1. • For the external debug interface, the OS Lock is set.

10.4.14 Debug Self Address Offset Register

The DBGDSAR characteristics are:

Purpose Defines the offset from the base address defined by DBGDRAR of the physical base address of the debug registers for the processor. See *Debug ROM Address Register* on page 10-19.

Usage constraints There are no usage constraints.

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 10-1 on page 10-5.This register is only visible in the CP14 interface, and therefore does not have a memory offset.

Figure 10-16 shows the DBGDSAR bit assignments as a 32-bit register.

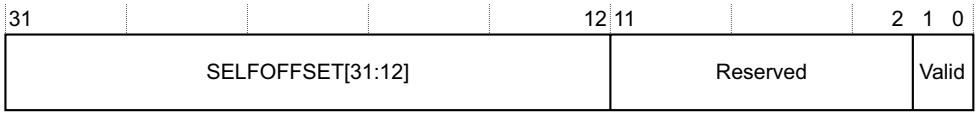


Figure 10-16 DBGDSAR 32-bit assignments

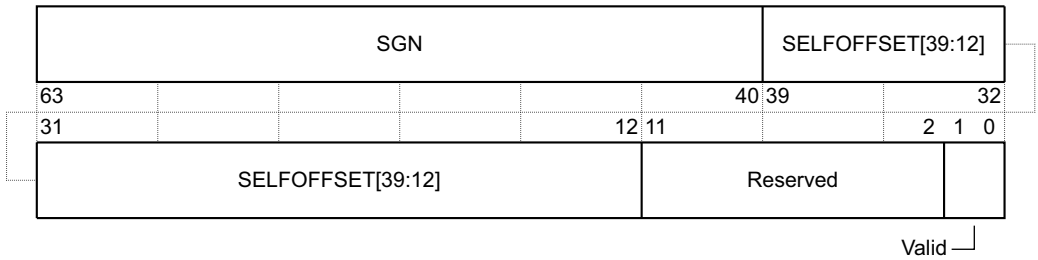


Figure 10-17 DBGDSAR 64-bit assignments

Table 10-16 shows the DBGDSAR bit assignments.

Table 10-16 DBGDSAR bit assignments

Bits	Name	Function
[63:40]	SGN	Sign extension.
[39:12]	SELFOFFSET[39:12]	Bits [39:32] of the two's complement offset from the base address defined by DBGDRAR to the physical address where the debug registers are mapped. Bits[11:0] of the address are zero. See *Debug ROM Address Register* on page 10-19. If the DBGDSAR.Valid is zero, is zero the value of this field is UNKNOWN.

Table 10-16 DBGDSAR bit assignments (continued)

Bits	Name	Function
[31:12]	SELFOFFSET[31:12]	Bits [31:12] of the two's complement offset from the base address defined by DBGDRAR to the physical address where the debug registers are mapped. Bits [11:0] of the address are zero. See *Debug ROM Address Register* on page 10-19 If DBGDSAR.Valid is zero, the value of this field is UNKNOWN.
[11:2]	-	Reserved.
[1:0]	Valid	Valid bit. This field indicates whether the debug self address offset is valid: `0b00` Offset is not valid. `0b11` Offset is valid.

10.4.15 Integration Miscellaneous Signals Register

The DBGITMISCOUT characteristics are:

Purpose Controls signal outputs when bit [0] of the Integration Mode Control Register is set.

Usage constraints DBGITMISCOUT is not accessible on the extended CP14 interface.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-18 shows the DBGITMISCOUT bit assignments.

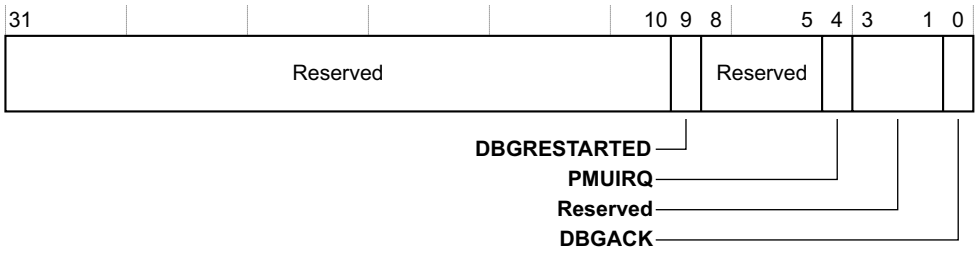


Figure 10-18 DBGITMISCOUT bit assignments

Table 10-17 shows the DBGITMISCOUT bit assignments.

Table 10-17 DBGITMISCOUT bit assignments

Bits	Name	Function
[31:10]	-	Reserved, SBZP.
[9]	DBGRESTARTED	Set value of the **DBGRESTARTED** output pin. Handshake for DBGRESTART that is the same as the DSCR[1] bit. The reset value is 0.
[8:5]	-	Reserved, SBZP.
[4]	PMUIRQ	Set value of **PMUIRQ** output pin. When this bit is set to 1, the corresponding **nPMUIRQ** signal is cleared to 0. The reset value is 1.
[3:1]	-	Reserved, SBZP.
[0]	DBGACK	Set value of the **DBGACK** output pin

10.4.16 Integration Miscellaneous Signals Input Register

The DBGITMISCIN characteristics are:

Purpose Enables the values of signal inputs to be read when bit [0] of the Integration Mode Control Register is set to 1.

Usage constraints DBGITMISCIN is not accessible on the extended CP14 interface.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-19 shows the DBGITMISCIN bit assignments.

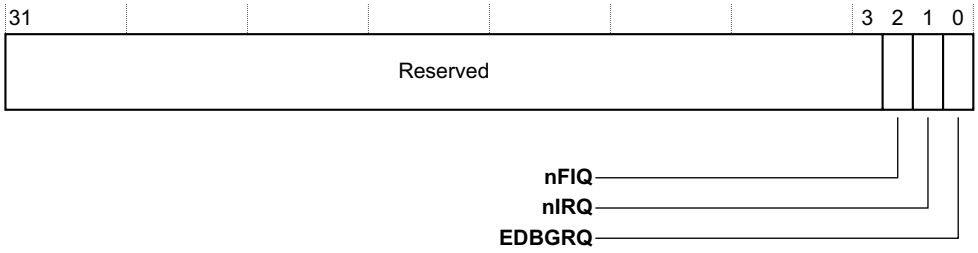


Figure 10-19 DBGITMISCIN bit assignments

Table 10-18 shows the DBGITMISCIN bit assignments.

Table 10-18 DBGITMISCIN bit assignments

Bits	Name	Function
[31:3]	-	Reserved, RAZ.
[2]	nFIQ	Read value of **nFIQ** input pin.
[1]	nIRQ	Read value of **nIRQ** input pin.
[0]	EDBGRQ	Read value of **EDBGRQ** input pin

10.4.17 Integration Mode Control Register

The DBGITCTRL characteristics are:

Purpose Switches the processor from its default functional mode into integration mode, where test software can control directly the inputs and outputs of the processor, for integration testing or topology detection. When the processor is in integration mode, the test software uses the integration registers to drive output values and to read inputs.

Usage constraints DBGITCTRL is not accessible on the extended CP14 interface.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-20 on page 10-28 shows the DBGITCTRL bit assignments.

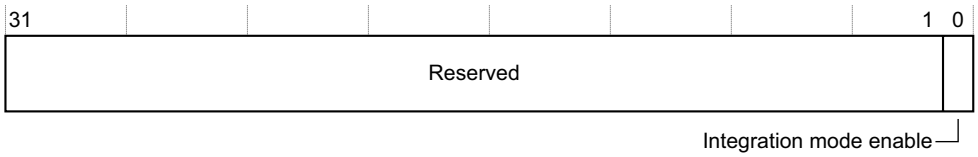


Figure 10-20 DBGITCTRL bit assignments

Table 10-19 shows the DBGITCTRL bit assignments.

Table 10-19 DBGITCTRL bit assignments

Bits	Name	Function
[31:1]	-	Reserved.
[0]	Integration mode enable	When this bit is set to 1, the device reverts to an integration mode to enable integration testing or topology detection: **0** Normal operation. **1** Integration mode enabled.

10.4.18 Claim Tag Set Register

The DBGCLAIMSET characteristics are:

Purpose Used by software to set CLAIM bits to 1.

Usage constraints Use in conjunction with the DBGCLAIMLR register. See *DBGCLAIMCLR bit assignments* on page 10-29.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-21 shows the DBGCLAIMSET bit assignments.

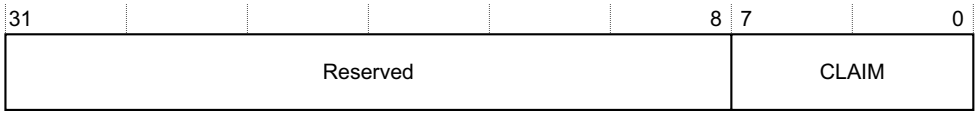


Figure 10-21 DBGCLAIMSET bit assignments

Table 10-20 shows the DBGCLAIMSET bit assignments.

Table 10-20 DBGCLAIMSET bit assignments

Bits	Name	Function
[31:8]	-	Reserved.
[7:0]	CLAIM	CLAIM bits. Writing a 1 to one of these bits sets the corresponding CLAIM bit to 1. A single write operation can set multiple bits to 1. The CLAIM bits do not have any specific functionality. ARM expects the usage model to be that an external debugger and a debug monitor can set specific bits to 1 to claim the corresponding debug resources. This field is RAO. See *Claim Tag Clear Register* on page 10-29 for information on how to: • Clear CLAIM bits to 0. • Read the current values of the CLAIM bits.

10.4.19 Claim Tag Clear Register

The DBGCLAIMCLR characteristics are:

Purpose Used by software to read the values of the CLAIM bits, and to clear these bits to zero.

Usage constraints Use in conjunction with the DBGCLAIMSET register. See *DBGCLAIMSET bit assignments* on page 10-28.

Configurations The processor implements bits [7:0] as claim tags.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-22 shows the DBGCLAIMCLR bit assignments.

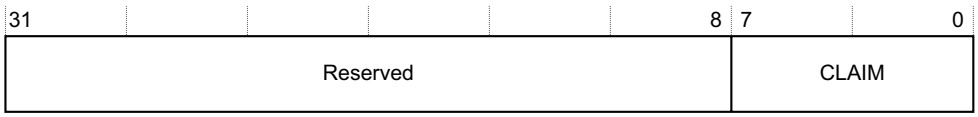


Figure 10-22 DBGCLAIMCLR bit assignments

Table 10-21 shows the DBGCLAIMCLR bit assignments.

Table 10-21 DBGCLAIMCLR bit assignments

Bits	Name	Function
[31:8]	-	Reserved, RAZ/SBZP.
[7:0]	CLAIM	CLAIM bits. Writing a 1 to one of these bits sets 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 *Claim Tag Set Register* on page 10-28

10.4.20 Debug Device ID Register 1

The DBGDEVID1 characteristics are:

Purpose Adds to the information given by the DBGDIDR by describing other features of the debug implementation.

Usage constraints There are no usage constraints.

Configurations Available in all configurations.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-23 shows the DBGDEVID1 bit assignments.

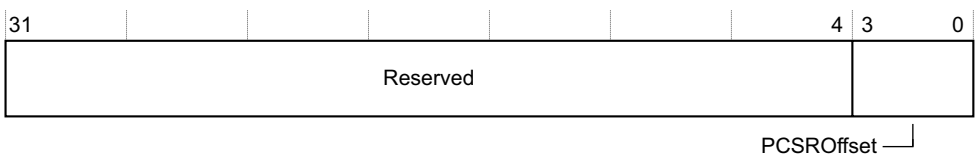


Figure 10-23 DBGDEVID1 bit assignments

Table 10-22 shows the DBGDEVID1 bit assignments.

Table 10-22 DBGDEVID1 bit assignments

Bits	Name	Function
[31:4]	-	Reserved.
[3:0]	PCSROffset	Defines the offset applied to DBGPCSR samples: 0b0001 DBGPCSR samples have no offset applied.

10.4.21 Debug Device ID Register

The DBGDEVID characteristics are:

Purpose Extends the DBGDIDR by describing other features of the debug implementation.

Usage constraints Use in conjunction with DBGDIDR to find the features of the debug implementation. See *DBGDIDR bit assignments* on page 10-9.

Configurations DBGDEVID is an optional register.

Attributes See the register summary in Table 10-1 on page 10-5.

Figure 10-24 shows the DBGDEVID bit assignments.

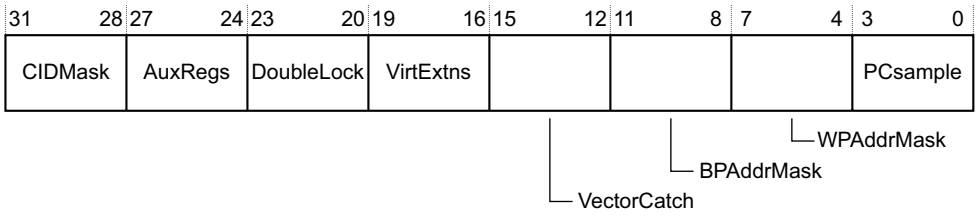


Figure 10-24 DBGDEVID bit assignments

Table 10-23 shows the DBGDEVID bit assignments.

Table 10-23 DBGDEVID bit assignments

Bits	Name	Function
[31:28]	CIDMask	Indicates the level of support for the Context ID matching breakpoint masking capability: 0b0000 Context ID masking is not implemented.
[27:24]	AuxRegs	Specifies support for the Debug External Auxiliary Control Register. See *Debug External Auxiliary Control Register* on page 10-12: 0b0001 The processor supports 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.

Table 10-23 DBGDEVID bit assignments (continued)

Bits	Name	Function
[11:8]	BPAddrMask	Indicates the level of support for the *Immediate Virtual Address* (IVA) matching breakpoint masking capability: `0b1111` Breakpoint address masking not implemented. DBGBCRn[28:24] are UNK/SBZP.
[7:4]	WPAddrMask	Indicates the level of support for the DVA matching watchpoint masking capability: `0b0001` Watchpoint address mask implemented.
[3:0]	PCSample	Indicates the level of support for Program Counter sampling using debug registers 40, 41, and 42: `0b0011` DBGPCSR, DBGCIDSR and DBGVIDSR are implemented as debug registers 40, 41, and 42.

10.4.22 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 10-24.

Table 10-24 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	`0x07`	`0xFE0`
Debug Peripheral ID1	`0xBC`	`0xFE4`
Debug Peripheral ID2[a]	`0x4B`	`0xFE8`
Debug Peripheral ID3	`0x00`	`0xFEC`

a. Bits [7:4] of this value match the revision field in the Debug Identification Register, see *Debug Identification Register* on page 10-9.

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* describes these registers.

10.4.23 Debug Component Identification Registers

There are four read-only Debug Component Identification Registers, Component ID0 through Component ID3. Table 10-25 shows these registers.

Table 10-25 Summary of the Component Identification Registers

Register	Value	Offset
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* describes these registers.

10.5 Debug events

A debug event can be either:

- A software debug event.
- A halting debug event.

A processor responds to a debug event in one of the following ways:

- Ignores the debug event.
- Takes a debug exception.
- Enters Debug state.

This section describes debug events in:

- *Watchpoint debug events*.
- *Asynchronous aborts*.

See the *ARM Architecture Reference Manual* for more information on debug events.

10.5.1 Watchpoint debug events

In the Cortex-A7 MPCore processor, watchpoint debug events are always synchronous. Memory hint instructions and cache clean operations do not generate watchpoint debug events. Store exclusive instructions generate a watchpoint debug event even when the check for the control of exclusive monitor fails.

10.5.2 Asynchronous aborts

The processor ensures that all possible outstanding asynchronous data aborts are recognized before entry to the Debug state. The debug asynchronous aborts discarded bit, DBGDSCR.ADAdiscard, is set to 1 on entry to Debug state.

10.6 External debug interface

The system can access memory-mapped debug registers through the APB interface. The APB interface is compliant with the APB interface.

Figure 10-25 shows the debug interface implemented in the Cortex-A7 MPCore processor. For more information on these signals, see the *ARM Architecture Reference Manual*.

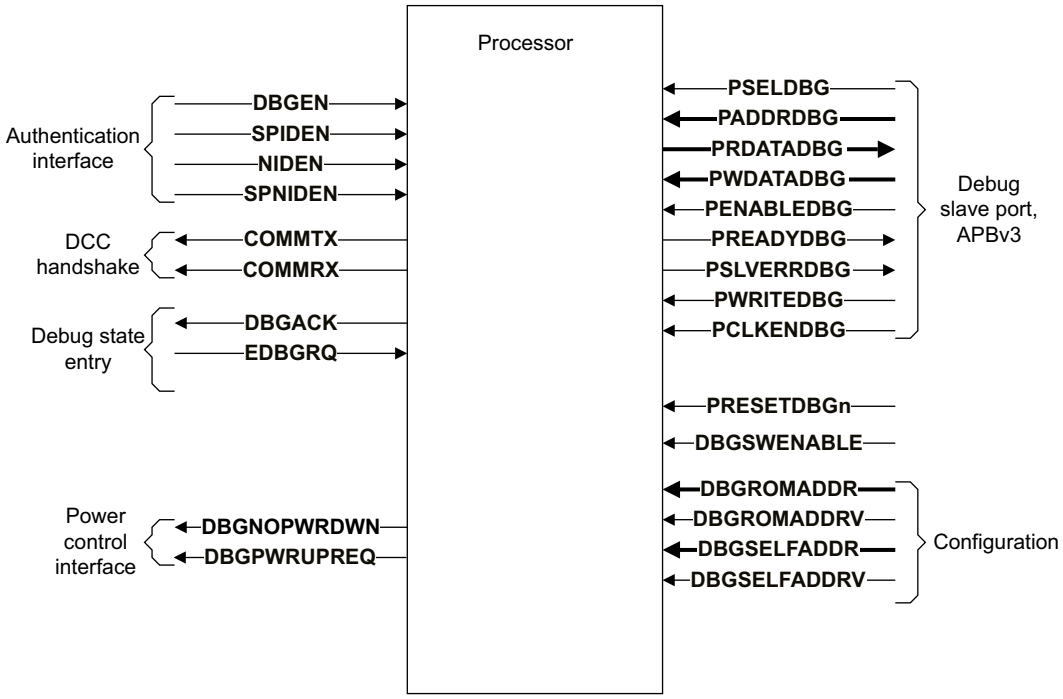


Figure 10-25 External debug interface, including APBv3 slave port

10.6.1 Memory map

The basic memory map supports up to four processors in a multiprocessor device. Table 10-26 shows the address mapping for the debug trace components.

Table 10-26 Address mapping for debug trace components

Address range	Component[a]
0x00000 - 0x00FFF	ROM table
0x01000 - 0x0FFFF	Reserved
0x10000 - 0x10FFF	CPU 0 Debug
0x11000 - 0x11FFF	CPU 0 PMU
0x12000 - 0x12FFF	CPU 1 Debug
0x13000 - 0x13FFF	CPU 1 PMU
0x14000 - 0x14FFF	CPU 2 Debug
0x15000 - 0x15FFF	CPU 2 PMU
0x16000 - 0x16FFF	CPU 3 debug

Table 10-26 Address mapping for debug trace components (continued)

Address range	Component[a]
0x17000 - 0x17FFF	CPU 3 PMU
0x18000 - 0x18FFF	CPU 0 CTI
0x19000 - 0x19FFF	CPU 1 CTI
0x1A000 - 0x1AFFF	CPU 2 CTI
0x1B000 - 0x1BFFF	CPU 3 CTI
0x1C000 - 0x1CFFF	CPU 0 Trace
0x1D000 - 0x1DFFF	CPU 1 Trace
0x1E000 - 0x1EFFF	CPU 2 Trace
0x1F000 - 0x1FFFF	CPU 3 Trace

a. Indicates the mapped component if present, otherwise reserved.

10.6.2 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 an implementation-specific sequence of instructions to change the signal value. For example, this might be a single `STR` instruction that writes certain value to a control register in a system peripheral.
2. If 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 11
Performance Monitoring Unit

This chapter describes the *Performance Monitoring Unit* (PMU) and the registers that it uses. It contains the following sections:

11.1 About the Performance Monitoring Unit

The processor includes logic to gather various statistics on the operation of the processor and memory system during runtime, based on the PMUv2 architecture. These events provide useful information about the behavior of the processor that you can use when debugging or profiling code.

The processor PMU provides four counters. Each counter can count any of the 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.

11.2 PMU functional description

This section describes the functionality of the PMU in:

- *Event interface*.
- *CP15 and APB interface*.
- *Counters*.

Figure 11-1 shows the major blocks inside the PMU.

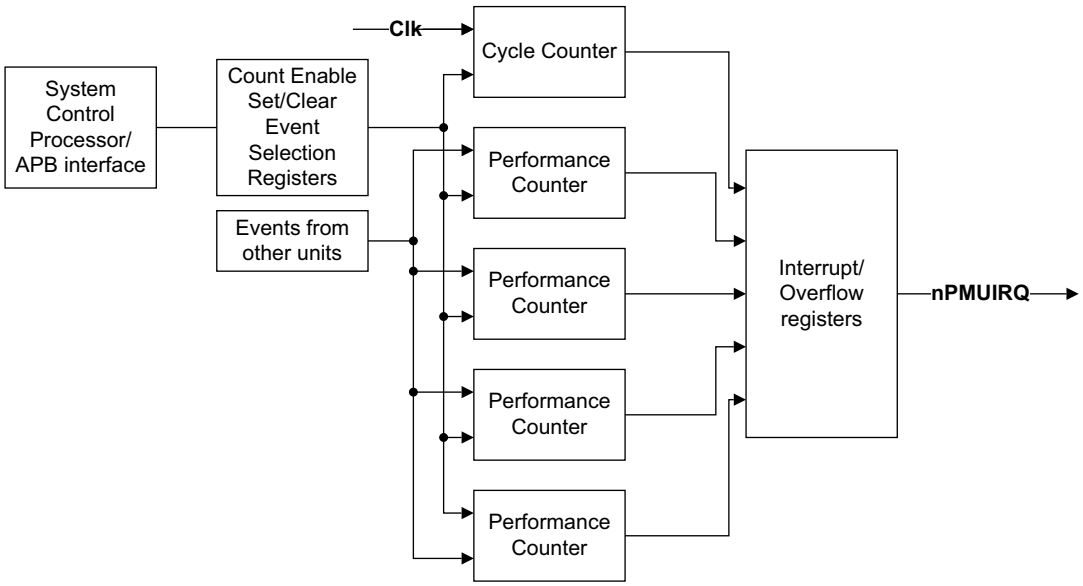


Figure 11-1 PMU block diagram

11.2.1 Event interface

Events from all other units from across the design are provided to the PMU.

11.2.2 CP15 and APB interface

The PMU registers can be programmed using the CP15 system control coprocessor or external APB interface.

11.2.3 Counters

The PMU has:

- Four 32-bit counters that increment when they are enabled based on events.
- One cycle counter that increments on the processor clock

11.3 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 11-1 gives a summary of the Cortex-A7 MPCore PMU registers.

Table 11-1 PMU register summary

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
0	0x000	c9	0	c13	2	PMXEVCNTR0	RW	Event Count Register, see the *ARM Architecture Reference Manual*
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 *ARM Architecture Reference Manual*
32-255	0x080-0x3FC	-	-	-	-	-		Reserved
256	0x400	c9	0	c13	1	PMXEVTYPER0	RW	Event Type Selection Register, see the *ARM Architecture Reference Manual*
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 Type Select Register 31, see the *ARM Architecture Reference Manual*
288-767	0x480-0xBFC	-	-	-	-	-	-	Reserved
768	0xC00	c9	0	c12	1	PMCNTENSET	RW	Count Enable Set Register, see the *ARM Architecture Reference Manual*
769-775	0xC04-0xC1C	-	-	-	-	-	-	Reserved
776	0xC20	c9	0	c12	2	PMCNTENCLR	RW	Count Enable Clear Register, see the *ARM Architecture Reference Manual*
777-783	0xC24-0xC3C	-	-	-	-	-	-	Reserved
784	0xC40	c9	0	c14	1	PMINTENSET	RW	Interrupt Enable Set Register, see the *ARM Architecture Reference Manual*
785-791	0xC44-0xC5C	-	-	-	-	-	-	Reserved
792	0xC60	c9	0	c14	2	PMINTENCLR	RW	Interrupt Enable Clear Register, see the *ARM Architecture Reference Manual*
793-799	0xC64-0xC7C	-	-	-	-	-	-	Reserved

Table 11-1 PMU register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
800	0xC80	c9	0	c12	3	PMOVSR	RW	Overflow Flag Status Register, see the *ARM Architecture Reference Manual*
801-807	0xC84-0xC9C	-	-	-	-	-	-	Reserved
808	0xCA0	c9	0	c12	4	PMSWINC	WO	Software Increment Register, see the *ARM Architecture Reference Manual*
809-895	0xCA4-0xDFC	-	-	-	-	-	-	Reserved
896	0xE00	-	-	-	-	PMCFGR	RO	Performance Monitor Configuration Register, see the *ARM Architecture Reference Manual*
897	0xE04	c9	0	c12	0	PMCR	RW	*Performance Monitor Control Register* on page 11-7
898	0xE08	c9	0	c14	0	PMUSERENR	RW	User Enable Register, see the *ARM Architecture Reference Manual*
899-903	0xE0C-0xE1C	-	-	-	-	-	-	Reserved
904	0xE20	c9	0	c12	6	PMCEID0	RO	Common Event Identification Register 0, see the *ARM Architecture Reference Manual*
905	0xE24	c9	0	c12	7	PMCEID1	RO	Common Event Identification Register 1, see the *ARM Architecture Reference Manual*
906-1003	0xE28-0xFAC	-	-	-	-	-	-	Reserved
1004	0xFB0	-	-	-	-	PMLAR	WO	Lock Access Register, see the *ARM Architecture Reference Manual*
1005	0xFB4	-	-	-	-	PMLSR	RO	Lock Status Register, see the *ARM Architecture Reference Manual*
1006	0xFB8					PMAUTHSTATUS	RO	Authentication Status Register, see the *ARM Architecture Reference Manual*
1007-1010	0xFBC-0xFC8	-	-	-	-	-	-	Reserved
1011	0xFCC	-	-	-	-	PMDEVTYPE	RO	Device Type Register, see the *ARM Architecture Reference Manual*

Table 11-1 PMU register summary (continued)

Register number	Offset	CRn	Op1	CRm	Op2	Name	Type	Description
1012	0xFD0	-	-	-	-	PMPID4	RO	*Performance Monitors Peripheral Identification Registers* on page 11-8
1013	0xFD4	-	-	-	-	PMPID5	RO	
1014	0xFD8	-	-	-	-	PMPID6	RO	
1015	0xFDC	-	-	-	-	PMPID7	RO	
1016	0xFE0	-	-	-	-	PMPID0	RO	
1017	0xFE4	-	-	-	-	PMPID1	RO	
1018	0xFE8	-	-	-	-	PMPID2	RO	
1019	0xFEC	-	-	-	-	PMPID3	RO	
1020	0xFF0	-	-	-	-	PMCID0	RO	*Performance Monitors Component Identification Registers* on page 11-9
1021	0xFF4	-	-	-	-	PMCID1	RO	
1022	0xFF8	-	-	-	-	PMCID2	RO	
1023	0xFFC	-	-	-	-	PMCID3	RO	

11.4 PMU register descriptions

This section describes the Cortex-A7 MPCore PMU registers. Table 11-1 on page 11-4 provides cross references to individual PMU registers.

11.4.1 Performance Monitor Control Register

The PMCR characteristics are:

Purpose

- Provides details of the performance monitor implementation, including the number of counters implemented.
- Configures and controls the counters.

Usage constraints The PMCR is:

- A read/write register.
- Common to the Secure and Non-secure states.
- Accessible from PL1 or higher.
- accessible in User mode only when the PMUSERENR.EN bit is set to 1.

Configurations Available in all configurations.

Attributes See the register summary in Table 11-1 on page 11-4.

Figure 11-2 shows the PMCR bit assignments.

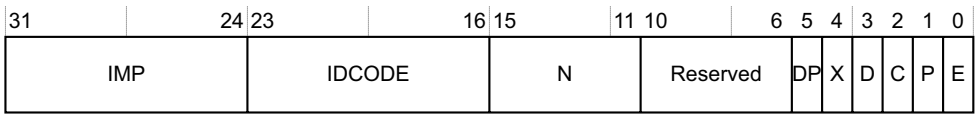


Figure 11-2 Performance Monitor Control Register bit assignments

Table 11-2 shows the PMCR bit assignments.

Table 11-2 PMCR bit assignments

Bits	Name	Function
[31:24]	IMP	Implementer code. `0x41` ARM. This is a read-only field.
[23:16]	IDCODE	Identification code. `0x07` Cortex-A7 MPCore identification code. This is a read-only field.
[15:11]	N	Number of event counters. In Secure state and Hyp mode, this field returns `0x4` that indicates the number of counters implemented. In Non-secure modes other than Hyp mode, this field reads the value of HDCR.HPMN. See *Hyp Debug Control Register* on page 4-68. This is a read-only field.
[10:6]	-	Reserved, UNK/SBZP.
[5]	DP	Disable cycle counter, PMCCNTR, in regions of software when prohibited: **0** Count is enabled in prohibited regions. This is the reset value. **1** Count is disabled in prohibited regions. This bit is read/write.

Table 11-2 PMCR bit assignments (continued)

Bits	Name	Function
[4]	X	Export enable. This bit permits events to be exported to another debug device, such as a trace macrocell, over an event bus: **0** Export of events is disabled. This is the reset value. **1** Export of events is enabled. This bit is read/write.
[3]	D	Clock divider: **0** When enabled, PMCCNTR counts every clock cycle. This is the reset value. **1** When enabled, PMCCNTR counts once every 64 clock cycles. This bit is read/write.
[2]	C	Clock counter reset: **0** No action. This is the reset value. **1** Reset PMCCNTR to 0. This bit is write-only, and always RAZ.
[1]	P	Event counter reset: **0** No action. This is the reset value. **1** Reset all event counters, not including PMCCNTR, to 0. 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 *Hyp Debug Control Register* on page 4-68. In Secure state and Hyp mode, writing a 1 to this bit resets all event counters. This bit is write-only, and always RAZ.
[0]	E	Enable bit. Performance monitor overflow IRQs are only signaled when the enable bit is set to 1. **0** All counters, including PMCCNTR, are disabled. This is the reset value. **1** All counters are enabled. This bit is read/write.

To access the PMCR, read or write the CP15 registers with:

```
MRC p15, 0, <Rt>, c9, c12, 0; Read Performance Monitor Control Register
MCR p15, 0, <Rt>, c9, c12, 0; Write Performance Monitor Control Register
```

11.4.2 Performance Monitors Peripheral Identification Registers

The Performance Monitors Peripheral Identification Registers provide standard information required for all components that conform to the ARM PMUv2 architecture. They are a set of eight registers, listed in register number order in Table 11-3.

Table 11-3 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
Performance Monitors Peripheral ID7	0x00	0xFDC
Performance Monitors Peripheral ID0	0xA7	0xFE0

Table 11-3 Summary of the Performance Monitors Peripheral Identification Registers

Register	Value	Offset
Performance Monitors Peripheral ID1	0xB9	0xFE4
Performance Monitors Peripheral ID2	0x4B	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* describes these registers.

11.4.3 Performance Monitors Component Identification Registers

There are four read-only Performance Monitors Component Identification Registers, Component ID0 through Component ID3. Table 11-4 shows these registers.

Table 11-4 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 ARM PMUv2 architecture. The *ARM Architecture Reference Manual* describes these registers.

11.5 Events

Table 11-5 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. Event reference numbers that are not listed are reserved.

Table 11-5 Performance monitor events

Event ID	PMUEVENT bit position	Description
0x00	-	Software increment. The register is incremented only on writes to the Software Increment Register. See the *ARM Architecture Reference Manual*.
0x01	[0]	Instruction fetch that causes a refill at (at least) the lowest level of instruction or unified cache. Includes the 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	[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	[4]	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.
0x06	[5]	Data read architecturally executed. Counts the number of data read instructions accepted by the Load Store Unit. This includes counting the speculative and aborted LDR/LDM, and the reads because of the SWP instructions.
0x07	[6]	Data write architecturally executed. Counts the number of data write instructions accepted by the Load Store Unit. This includes counting the speculative and aborted STR/STM, and the writes because of the SWP instructions.
0x08	[7]	Instruction architecturally executed.
0x09	[8]	Exception taken. Counts the number of exceptions architecturally taken.
0x0A	[9]	Exception return architecturally executed. The following instructions are reported on this event: • LDM {..., pc}^ • RFE • DP S pc
0x0B	[10]	Change to ContextID retired. Counts the number of instructions architecturally executed writing into the ContextID Register.
0x0C	[11]	Software change of PC.
0x0D	[12]	Immediate branch architecturally executed (taken or not taken). This includes the branches which are flushed due to a previous load/store which aborts late.
0x0E	[13]	Procedure return (other than exception returns) architecturally executed.
0x0F	[14]	Unaligned load-store.
0x10	[15]	Branch mispredicted/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.

Table 11-5 Performance monitor events (continued)

Event ID	PMUEVENT bit position	Description
0x12	[16]	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	[17]	Data memory access.
0x14	[18]	Instruction Cache access.
0x15	[19]	Data cache eviction.
0x16	-	Level 2 data cache access
0x17	-	Level 2 data cache refill
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.
0x1D	-	Bus cycle
0x60	-	Bus access, read
0x61	-	Bus access, write
0x86	[20]	IRQ exception taken.
0x87	[21]	FIQ exception taken.
0xC0	[22]	External memory request.
0xC1	[23]	Non-cacheable external memory request.
0xC2	[24]	Linefill because of prefetch.
0xC3	[25]	Prefetch linefill dropped.
0xC4	[26]	Entering read allocate mode.
0xC5	[27]	Read allocate mode.
0xC6	[28]	Reserved.
0xC7	-	ETM Ext Out[0].
0xC8	-	ETM Ext Out[1].
0xC9	[29]	Data Write operation that stalls the pipeline because the store buffer is full.
0xCA	-	Data snooped from other processor. This event counts memory-read operations that read data from another processor within the local Cortex-A7 cluster, rather than accessing the L2 cache or issuing an external read. It increments on each transaction, rather than on each beat of data.

11.6 Interrupts

The Cortex-A7 MPCore processor asserts the **nPMUIRQ** signal when an interrupt is generated by the PMU. You can route this signal to an interrupt controller for prioritization and masking. This is the only mechanism that signals this interrupt to the processor.

11.7 Exporting PMU events

This section describes exporting of PMU events in:

- *External hardware*.
- *Debug trace hardware*.

11.7.1 External hardware

In addition to the counters in the processor, Table 11-5 on page 11-10 shows the events on the **PMUEVENT** bus that you can connect to external hardware.

11.7.2 Debug trace hardware

Some of the events in Table 11-5 on page 11-10 can be exported to other external debug, or trace hardware, to enable the events to be monitored. See the *CoreSight SoC Technical Reference Manual* for more information.

Appendix A
Signal Descriptions

This appendix describes the Cortex-A7 MPCore signals. It contains the following sections:

A.1 About the signal descriptions

The tables in this appendix list the Cortex-A7 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-A7 MPCore configuration, that is, four processors, an integrated GIC, and an ACE master interface. 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, [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-1 Clock and reset signals

Signal	Direction	Description
CLKIN	Input	Global clock.
nCOREPORESET[3:0]	Input	All processor reset: **0** Apply reset to all processor logic that includes NEON and VFP, Debug, ETM, breakpoint and watchpoint logic. **1** Do not apply reset to all processor logic that includes NEON and VFP, Debug, ETM, breakpoint and watchpoint logic.
nCORERESET[3:0]	Input	Individual processor resets excluding Debug and ETM: **0** Apply reset to processor that includes NEON and VFP, but excludes Debug, ETM, breakpoint and watchpoint logic. **1** Do not apply reset to processor that includes NEON and VFP, but excludes Debug, ETM, breakpoint and watchpoint logic.
nDBGRESET[3:0]	Input	Debug logic resets: **0** Apply reset to debug, breakpoint and watchpoint logic. **1** Do not apply reset to debug, breakpoint and watchpoint logic.
nL2RESET	Input	SCU global reset: **0** Apply reset to shared L2 memory system controller. **1** Do not apply reset to shared L2 memory system controller.
L1RSTDISABLE[3:0]	Input	Disable automatic L1 cache invalidate at reset: **0** L1 cache is reset by hardware. **1** L1 cache is not reset by hardware.
L2RSTDISABLE	Input	Disable automatic L2 cache invalidate at reset: **0** L2 cache is reset by hardware. **1** L2 cache is not reset by hardware.

A.3 Configuration signals

Table A-2 shows the configuration 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-2 Configuration signals

Signal	Direction	Description
CFGEND[3:0]	Input	Endianness configuration at reset. It sets the initial value of the EE bit in the CP15 *System Control Register* (SCTLR): **0** EE bit is LOW. **1** EE bit is HIGH. This pin is only sampled during reset of the processor.
VINITHI[3:0]	Input	Location of the exception vectors at reset. It sets the initial value of the V bit in the CP15 *System Control Register* (SCTLR): **0** Exception vectors start at address `0x00000000`. **1** Exception vectors start at address `0xFFFF0000`. This pin is only sampled during reset of the processor.
CFGTE[3:0]	Input	Default exception handling state. It sets the initial value of the TE bit in the CP15 *System Control Register* (SCTLR): **0** TE bit is LOW. **1** TE bit is HIGH. This pin is only sampled during reset of the processor.
CLUSTERID[3:0]	Input	Value read in the Cluster ID field, bits [11:8], of the CP15 *Multiprocessor Affinity Register* (MPDIR). This pin is only sampled during reset of the processor.
CP15SDISABLE[3:0]	Input	Disable write access to some secure CP15 registers.
SMPnAMP[3:0]	Output	Signals *Symmetric MultiProcessing* (SMP) mode or *Asymmetric MultiProcessing* (AMP) for each processor in the Cortex-A7 MPCore processor.

A.4 Generic Interrupt Controller signals

Table A-3 shows the *Generic Interrupt Controller* (GIC) 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-3 GIC signals

Signal	Direction	Description
CFGSDISABLE[a]	Input	Disables write access to some secure GIC registers.
IRQS[N:0][b]	Input	Interrupt request input lines for the GIC where **N** can be 31, 63, up to 479 by increments of 32.
nFIQ[3:0]	Input	FIQ request. Active-LOW, fast interrupt request: **0** Activate fast interrupt. **1** Do not activate fast interrupt. The processor treats the **nFIQ** input as level-sensitive. To guarantee that an interrupt is taken, ensure the **nFIQ** input remains asserted until the processor acknowledges the interrupt.
nIRQ[3:0]	Input	IRQ request input lines. Active-LOW, interrupt request: **0** Activate interrupt. **1** Do not activate interrupt. The processor treats the **nIRQ** input as level-sensitive. To guarantee that an interrupt is taken, ensure the **nIRQ** input remains asserted until the processor acknowledges the interrupt.
nFIQOUT[3:0][a]	Output	Active-LOW output of individual processor nFIQ from the GIC. For use when processors are powered off and interrupts are handled by the GIC under the control of an external power controller.
nIRQOUT[3:0][a]	Output	Active-LOW output of individual processor nIRQ from the GIC. For use when processors are powered off and interrupts are handled by the GIC under the control of an external power controller.
nVFIQ[3:0]	Input	Virtual FIQ request. Active-LOW, fast interrupt request: **0** Activate fast interrupt. **1** Do not activate fast interrupt. The processor treats the **nVFIQ** input as level-sensitive. The **nVFIQ** input must be asserted until the processor acknowledges the interrupt. If the Cortex-A7 MPCore processor is configured to include the GIC, and the GIC is used, the input pins **nVIRQ** and **nVFIQ** must be tied off to HIGH. If the processor is configured to include the GIC, and the GIC is not used, the input pins **nVIRQ** and **nVFIQ** can be driven by an external GIC in the SoC. See *GIC configuration* on page 8-5 for more information.
nVIRQ[3:0]	Input	Virtual IRQ request. Active-LOW, interrupt request: **0** Activate interrupt. **1** Do not activate interrupt. The processor treats the **nVIRQ** input as level-sensitive. The **nVIRQ** input must be asserted until the processor acknowledges the interrupt. If the Cortex-A7 MPCore processor is configured to include the GIC, and the GIC is used, the input pins **nVIRQ** and **nVFIQ** must be tied off to HIGH. If the processor is configured to include the GIC, and the GIC is not used, the input pins **nVIRQ** and **nVFIQ** can be driven by an external GIC in the SoC. See *GIC configuration* on page 8-5 for more information.
PERIPHBASE[39:15]	Input	Specifies the base address for the GIC registers. This value is sampled into the CP15 *Configuration Base Address Register* (CBAR) at reset.

a. Not used if a GIC is not present.

b. **N** is 0 if there is no GIC present.

A.5 Generic timer signals

Table A-4 shows the generic timer 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-4 Generic timer signals

Signal	Direction	Description
nCNTHPIRQ[3:0]	Output	Hypervisor physical timer event
nCNTPNSIRQ[3:0]	Output	Non-secure physical timer event
nCNTPSIRQ[3:0]	Output	Secure physical timer event
nCNTVIRQ[3:0]	Output	Virtual timer event
CNTVALUEB[63:0]	Input	Global system counter value in binary format

A.6 Power control signals

Table A-5 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-5 Power control signals

Signal	Direction	Description
EVENTI	Input	Event input for processor wake-up from WFE state. See *Event communication using WFE or SEV* on page 2-20 for more information.
EVENTO	Output	Event output. Active when a SEV instruction is executed. See *Event communication using WFE or SEV* on page 2-20 for more information.
STANDBYWFI[3:0]	Output	Indicates if a processor is in WFI standby mode: **0** Processor not in WFI standby mode. **1** Processor in WFI standby mode.
STANDBYWFE[3:0]	Output	Indicates if a processor is in WFE standby mode: **0** Processor not in WFE standby mode. **1** Processor in WFE standby mode.
STANDBYWFIL2	Output	Indicates if the L2 memory system is in WFI standby mode. This signal is active when the following conditions are met: • All processors are in standby WFI. • **ACINACTM** is asserted HIGH. • L2 memory system is idle.
DBGPWRUPREQ[3:0]	Input	Power up request: **0** Power down debug request to the power controller. **1** Power up request to the power controller.
DBGNOPWRDWN[3:0]	Output	No power-down request: **0** On a power-down request, the SoC power controller powers down the processor **1** On a power-down request, the SoC power controller does not power down the processor.

A.7 ACE master interface signals

This section describes the ACE master interface signals:

- *Clock and configuration signals*.
- *Write address channel signals* on page A-9.
- *Write data channel signals* on page A-9.
- *Write data response channel signals* on page A-10.
- *Read address channel signals* on page A-10.
- *Read data channel signals* on page A-11.
- *Snoop address channel signals* on page A-11.
- *Snoop response channel signals* on page A-11.
- *Snoop data channel signals* on page A-12.
- *Read/write acknowledge signals* on page A-12.

For a complete description of the AXI interface signals, see the *AMBA AXI and ACE Protocol Specification*.

A.7.1 Clock and configuration signals

Table A-6 shows the clock and configuration signals for the ACE master interface.

Table A-6 Clock and configuration signals

Signal	Direction	Description
ACLKENM	Input	AXI master bus clock enable. See *Clocking* on page 2-9 for more information.
ACINACTM	Input	Snoop interface is inactive and no longer accepting requests.
BROADCASTINNER[a]	Input	Enable broadcasting of Inner Shareable transactions: **0** Inner Shareable transactions are not broadcasted externally. **1** Inner Shareable transactions are broadcasted externally. If **BROADCASTINNER** is tied HIGH, you must also tie **BROADCASTOUTER** HIGH.
BROADCASTOUTER[a]	Input	Enable broadcasting of outer shareable transactions: **0** Outer Shareable transactions are not broadcasted externally. **1** Outer Shareable transactions are broadcasted externally.
BROADCASTCACHEMAINT[a]	Input	Enable broadcasting of cache maintenance operations to downstream caches: **0** Cache maintenance operations are not broadcasted to downstream caches. **1** Cache maintenance operations are broadcasted to downstream caches.
SYSBARDISABLE[ab]	Input	Disable broadcasting of barriers onto system bus: **0** Barriers are broadcast onto system bus, this requires an AMBA4 interconnect. **1** Barriers are not broadcast onto the system bus. This is compatible with an AXI3 interconnect.

a. This pin is only sampled during reset of the processor. See Table 7-1 on page 7-4 for more information.

b. For AXI3 compatibility, **SYSBARDISABLE** must be tied HIGH and **BROADCASTINNER**, **BROADCASTOUTER**, and **BROADCASTCACHEMAINT** must be tied LOW.

A.7.2 Asynchronous error signals

Table A-7 shows the asynchronous error signals.

Table A-7 Asynchronous error signals

Signal	Direction	Description
nAXIERRIRQ	Output	Error indicator for AXI transactions with an error condition that might not map to an individual processor in the multiprocessor device. See *External aborts handling* on page 7-10 for more information.

A.7.3 Write address channel signals

Table A-8 shows the address channel signals for the ACE master interface.

Table A-8 address channel signals

Signal	Direction	Description
AWADDRM[39:0]	Output	Address
AWBARM[1:0]	Output	Barrier type
AWBURSTM[1:0]	Output	Burst type
AWCACHEM[3:0]	Output	Cache type
AWDOMAINM[1:0]	Output	Domain type
AWIDM[4:0]	Output	Request ID
AWLENM[7:0]	Output	Burst length
AWLOCKM	Output	Lock type
AWPROTM[2:0]	Output	Protection type
AWREADYM	Input	Address ready
AWSIZEM[2:0]	Output	Burst size
AWSNOOPM[2:0]	Output	Snoop request type
AWVALIDM	Output	Address valid

A.7.4 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	Direction	Description
WDATAM[127:0]	Output	Write data
WIDM[4:0]	Output	Write ID
WLASTM	Output	Write last indication

Table A-9 Write data channel signals (continued)

Signal	Direction	Description
WREADYM	Input	Write ready
WSTRBM[15:0]	Output	Write byte-lane strobes
WVALIDM	Output	Write valid

A.7.5 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	Direction	Description
BIDM[4:0]	Input	Response ID
BREADYM	Output	Response ready
BRESPM[1:0]	Input	Write response
BVALIDM	Input	Response valid

A.7.6 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	Direction	Description
ARADDRM[39:0]	Output	Address
ARBARM[1:0]	Output	Barrier type
ARBURSTM[1:0]	Output	Burst type
ARCACHEM[3:0]	Output	Cache type
ARDOMAINM[1:0]	Output	Domain type
ARIDM[5:0]	Output	Request ID
ARLENM[7:0]	Output	Burst length
ARLOCKM	Output	Lock type
ARPROTM[2:0]	Output	Protection type
ARREADYM	Input	Address ready
ARSIZEM[2:0]	Output	Burst size
ARSNOOPM[3:0]	Output	Snoop request type
ARVALIDM	Output	Address valid

A.7.7 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	Direction	Description
RDATAM[127:0]	Input	Read data
RIDM[5:0]	Input	Read data ID
RLASTM	Input	Read last indication
RREADYM	Output	Read ready
RRESPM[3:0]	Input	Read response
RVALIDM	Input	Read valid

A.7.8 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	Direction	Description
ACADDRM[39:0]	Input	Address
ACPROTM[2:0]	Input	Protection type
ACREADYM	Output	Address ready
ACSNOOPM[3:0]	Input	Transaction type
ACVALIDM	Input	Address valid

A.7.9 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	Direction	Description
CRREADYM	Input	Response ready
CRVALIDM	Output	Response valid
CRRESPM[4:0]	Output	Snoop response

A.7.10 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	Direction	Description
CDDATAM[127:0]	Output	Snoop data
CDLASTM	Output	Snoop last indication
CDREADYM	Input	Snoop ready
CDVALIDM	Output	Snoop valid

A.7.11 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	Direction	Description
RACKM	Output	Read acknowledge
WACKM	Output	Write acknowledge

A.8 External debug interface

The following sections describe the external debug interface signals:

- *APB Interface signals*.
- *Authentication interface signals* on page A-14.
- *Miscellaneous Debug signals* on page A-14.
- *ETM interface signals* on page A-16.
- *PMU signals* on page A-16.

A.8.1 APB Interface signals

Table A-17 shows the APB Interface signals.

Table A-17 APB Interface signals

Signal	Direction	Description
PADDRDBG[14:2]	Input	APB Address bus bits[14:2]
PADDRDBG31	Input	APB address bus bit[31]: **0** Not an external debugger access. **1** External debugger access.
PCLKENDBG	Input	APB clock enable
PENABLEDBG	Input	Indicates the second and subsequent cycles of an APB transfer.
PRDATADBG[31:0]	Output	APB read data bus
PREADYDBG	Output	APB slave ready. An APB slave can assert **PREADYDBG** to extend a transfer by inserting wait states.
PSELDBG	Input	Debug bus access
PSLVERRDBG	Output	APB slave transfer error: **0** No transfer error. **1** Transfer error.
PWDATADBG[31:0]	Input	APB write data bus
PWRITEDBG	Input	APB read or write signal: **0** Reads from APB. **1** Writes to APB.

A.8.2 Authentication interface signals

Table A-18 shows the 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-18 Authentication interface signals

Signal	Direction	Description
DBGEN[3:0]	Input	Invasive debug enable: **0** Not enabled. **1** Enabled.
NIDEN[3:0]	Input	Non-invasive debug enable: **0** Not enabled. **1** Enabled.
SPIDEN[3:0]	Input	Secure privileged invasive debug enable: **0** Not enabled. **1** Enabled.
SPNIDEN[3:0]	Input	Secure privileged non-invasive debug enable: **0** Not enabled. **1** Enabled.

A.8.3 Miscellaneous Debug signals

Table A-19 shows the miscellaneous Debug 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-19 Miscellaneous Debug signals

Signal	Direction	Description
COMMRX[3:0]	Output	Communications channel receive. Receive portion of Data Transfer Register full flag: **0** Empty. **1** Full.
COMMTX[3:0]	Output	Communication transmit channel. Transmit portion of Data Transfer Register empty flag: **0** Full. **1** Empty.
APBACTIVE[3:0]	Output	Processor debug busy. **0** Not active. **1** Active.
DBGACK[3:0]	Output	Debug acknowledge: **0** External debug request not acknowledged. **1** External debug request acknowledged.
DBGOSUNLOCKCATCH[3:0]	Input	Debug OS unlock catch enable: **0** Debug OS unlock catch disabled. **1** Debug OS unlock catch enabled.

Table A-19 Miscellaneous Debug signals (continued)

Signal	Direction	Description
DBGHALTREQ[3:0]	Input	Debug halt request: **0** No debug halt request. **1** Debug halt request.
DBGLOCKSET[3:0]	Input	Debug software lock status: **0** Lock clear. **1** Lock set.
DBGHOLDRST[3:0]	Input	Debug hold core warm reset: **0** Processor is not held in reset after a power-up or warm reset. **1** Processor is held in reset after a power-up or warm reset.
DBGSWENABLE[3:0]	Input	Debug software access enable: **0** Not enabled. **1** Enabled, access by the software through the Extended CP14 interface is permitted.
EDBGRQ[3:0]	Input	External debug request: **0** No external debug request. **1** External debug request. The processor treats the **EDBGRQ** input as level-sensitive. The **EDBGRQ** input must be asserted until the processor asserts **DBGACK.**
DBGROMADDR[39:12]	Input	Specifies bits [39:12] of the ROM table physical address. If the address cannot be determined, tie this signal off to 0. This pin is only sampled during reset of the processor.
DBGROMADDRV	Input	Valid signal for **DBGROMADDR**. If the address cannot be determined, tie this signal LOW. This pin is only sampled during reset of the processor.
DBGSELFADDR[39:15]	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 cannot be determined, tie this signal off to 0. This pin is only sampled during reset of the processor.
DBGSELFADDRV	Input	Valid signal for **DBGSELFADDR**. If the offset cannot be determined, tie this signal LOW. This pin is only sampled during reset of the processor.
DBGRESTART[3:0]	Input	External restart requests
DBGRESTARTED[3:0]	Output	Handshake for DBGRESTART.
DBGTRIGGER[3:0]	Output	Debug external request taken.

A.8.4 ETM interface signals

Table A-20 shows the ETM interface signals.

Table A-20 ETM interface signals

Signal	Direction	Description
ETMICTLx[20:0][a]	Output	ETM instruction control bus
ETMIAx[31:1]	Output	ETM instruction address
ETMDCTLx[10:0]	Output	ETM data control bus
ETMDAx[31:0]	Output	ETM data address
ETMDDx[63:0]	Output	ETM data write data value
ETMCIDx[31:0]	Output	Current processor Context ID
ETMVMIDx[7:0]	Output	Current processor Virtual Machine ID
ETMWFXPENDINGx	Output	Processor is attempting to enter WFI or WFE state
ETMPWRUPx	Input	Power up processor ETM interface
ETMEXTOUTx[1:0]	Input	ETM external event to be monitored
PMUEVENTx[29:0]	Output	Performance monitor unit output.

a. x is 0, 1, 2, or 3 to reference processor 0-3.

A.8.5 PMU signals

Table A-21 shows the performance monitoring signals.

Table A-21 Performance Monitoring Unit signals

Signal	Direction	Description
nPMUIRQ[3:0][a]	Output	PMU interrupt signals

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.

A.9 DFT and MBIST interface signals

This section section describes:

- *DFT interface*.
- *MBIST interface*.

A.9.1 DFT interface

Table A-22 shows the DFT interface signals.

Table A-22 DFT interface signals

Signal name	Direction	Description
DFTRAMHOLD	Input	Disables the RAM chip selects during scan testing
DFTRSTDISABLE	Input	Disables internal synchronized reset during scan shift
DFTSE	Input	Scan shift enable, forces on the clock grids during scan shift

A.9.2 MBIST interface

Table A-23 shows the MBIST interface signals.

Table A-23 MBIST interface signals

Signal name	Direction	Description
MBISTACK	Output	MBIST test acknowledge
MBISTADDR[13:0]	Input	MBIST logical address in array
MBISTARRAY[8:0]	Input	MBIST array selector
MBISTBE[3:0]	Input	MBIST bit write enable
MBISTCFG	Input	MBIST all RAMs enable
MBISTINDATA[85:0]	Input	MBIST data in
MBISTOUTDATA[85:0]	Output	MBIST data out
MBISTREADEN	Input	MBIST Read enable
MBISTREQ	Input	MBIST test request
MBISTWRITEEN	Input	MBIST write enable
nMBISTRESET	Input	MBIST reset

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
Clarified reset signal descriptions	*Resets* on page 2-10	All
Updated valid reset combinations	Table 2-1 on page 2-11	All
Updated reset value of the Main ID Register	Table 4-2 on page 4-4 Table 4-16 on page 4-15	r0p1
Updated bits[3:0] of the Main ID Register	*Main ID Register* on page 4-26	r0p1
Updated descriptions for Processor Feature Register 0 bits[15:12] and bits[11:8]	*Processor Feature Register 0* on page 4-30	All
Clarified the CSSELR value and the complete register encoding for L1 instruction cache and L1 instruction cache selection in the CCSIDR	Table 4-47 on page 4-48	All
Clarified **BROADCASTCACHEMAINT** pin description	*Snoop Control Unit* on page 7-3	All

Table B-2 Differences between issue A and issue B (continued)

Change	Location	Affects
Enhancement in the use of **nVFIQ** and **nVIRQ** signals	*GIC configuration* on page 8-5 Table A-3 on page A-5	r0p1
Clarified DBGDIDR, DBGPCSR, DBGBVR, DBGDRAR, and DBGDSAR register descriptions	Chapter 10 *Debug*	All
Updated the value for Peripheral ID2 register	Table 10-24 on page 10-31 Table 11-3 on page 11-8	r0p1
Renamed PMCCFILTR to PMXEVTYPER31 in the PMU register summary table	Table 11-1 on page 11-4	All
Updated the **EVENTI** and **EVENTO** signal descriptions	Table A-5 on page A-7	All
Clarified the **DBGHOLDRST[3:0]** signal description	Table A-19 on page A-14	All

Table B-3 Differences between issue B and issue C

Change	Location	Affects
Added the Debug an Performance Monitors Peripheral ID2 Register values	*Product revisions* on page 1-11	r0p1
Clarified the power-down sequences	*Individual processor shutdown mode* on page 2-17 *Dormant mode* on page 2-19	All
Updated the reset value of the Main ID Register	Table 4-2 on page 4-4 Table 4-16 on page 4-15	r0p2
Updated the PRRR, NMRR, MAIR0 and MAIR1 reset values	Table 4-11 on page 4-12 Table 4-17 on page 4-16	All
Clarified the ID_ISAR0.Divide_instrs bit description	Table 4-41 on page 4-40	All
Added PRRR description	*Primary Region Remap Register* on page 4-54	All
Added MAIR0 and MAIR1 description	*MAIR0 and MAIR1, Memory Attribute Indirection Registers 0 and 1* on page 4-56	All
Added NMRR description	*Normal Memory Remap Register* on page 4-58	All
Updated bits[3:0] of the Main ID Register	*Main ID Register* on page 4-26	r0p2
Updated the ACTLR.DDI bit description	Table 4-60 on page 4-60	All
Clarified the CPACR.ASEDIS bit description	Table 4-61 on page 4-62	All
Clarified the SCR.nET bit and SCR.NS bit descriptions	Table 4-62 on page 4-64	All
Clarified the NSACR.cp11 bit and the NSACR.cp10 bit descriptions	Table 4-63 on page 4-65	All
Updated the type of exceptions reported in the DFSR	Table 4-67 on page 4-74 Table 4-68 on page 4-76	All
Updated the type of exceptions reported in the IFSR	Table 4-70 on page 4-77 Table 4-71 on page 4-78	All
Added memory region attributes description	Memory region attributes on page 5-3	All

Table B-3 Differences between issue B and issue C (continued)

Change	Location	Affects
Updated the representation of the **BROADCASTINNER**, **BROADCASTOUTER**, **BROADCASTCACHEMAINT**, and **SYSBARDISABLE** signals	Table 7-1 on page 7-4	All
Added the key features in each of the supported ACE configurations	Table 7-2 on page 7-4	All
Updated write and read issuing capability comments	Table 7-3 on page 7-5	All
Clarified the DBGOSLSR.OSLK bit description	Table 10-14 on page 10-22	All
Clarified the peripheral and component ID registers used for Debug	*Debug Peripheral Identification Registers* on page 10-31 *Debug Component Identification Registers* on page 10-32	All
Clarified the peripheral and component ID registers used for Performance Monitors	*Performance Monitors Peripheral Identification Registers* on page 11-8 *Performance Monitors Component Identification Registers* on page 11-9	All
Updated the Debug Peripheral ID2 value	Table 10-24 on page 10-31	r0p2
Updated the Performance Monitors Peripheral ID2 value	Table 11-3 on page 11-8	r0p2

Table B-4 Differences between issue C and issue D

Change	Location	Affects
Changed the number of consecutive cache line sized writes to enter secondary read allocate mode from seven to 127. Also clarified the secondary read allocate mode description.	*Data side memory system* on page 2-4	r0p3
Updated the power up and power down sequences required for individual processor shutdown mode	*Individual processor shutdown mode* on page 2-17	All
Updated the reset value of the Main ID Register.	Table 4-2 on page 4-4 Table 4-16 on page 4-15	r0p3
Clarified SCTLR reset value.	Table 4-3 on page 4-5	All
Clarified access to system control registers when CRn is c7.	Table 4-8 on page 4-8	All
Updated the CCSIDR usage constraints.	*Cache Size ID Register* on page 4-46	All
Updated the CPACR usage constraints.	*Coprocessor Access Control Register* on page 4-61	All
Clarified the use of the SCTLR.Z bit.	Table 4-52 on page 4-52	All
Updated bits[3:0] of the Main ID Register.	*Main ID Register* on page 4-26	r0p3
Updated the ACTLR.SMP bit description.	Table 4-60 on page 4-60	All
Updated CPACR.D32DIS bit description. This bit is UNK/SBZP for implementations with VFP and Advanced SIMD, only r0p3 and later versions enforce this.	Table 4-61 on page 4-62	r0p3
Updated NSACR.NSD32DIS bit description. This bit is UNK/SBZP for implementations with VFP and Advanced SIMD, but only r0p3 enforces this.	Table 4-63 on page 4-65	r0p3

Table B-4 Differences between issue C and issue D (continued)

Change	Location	Affects
Clarified ACE transactions.	Table 6-1 on page 6-7	All
Clarified Main TLB descriptor memory type and shareability encodings	Table 6-9 on page 6-12	All
Clarified how the Load-Exclusive instruction affects the internal exclusive monitor.	*Internal exclusive monitor* on page 6-6	All
Clarified the use of registers that are offset `0xF10-0xFFC` in the GIC Distributor.	Table 8-3 on page 8-6	All
Updated the Debug Peripheral ID2 value.	Table 10-24 on page 10-31	r0p3
Updated the Performance Monitors Peripheral ID2 value.	Table 11-3 on page 11-8	r0p3
Updated the **nFIQ[3:0]** and **nIRQ[3:0]** signal descriptions.	*Generic Interrupt Controller signals* on page A-5	All
Clarified snoop response channel signals.	Table A-14 on page A-11	All

Table B-5 Differences between issue D and issue E

Change	Location	Affects
Clarified input signals synchronized by the Cortex-A7 MPCore processor.	*Clocking* on page 2-9	All
Added recommendation that the reset signals are asserted for at least four **CLKIN** cycles.	*Resets* on page 2-10	All
Updated the processor power down sequence.	*Individual processor shutdown mode* on page 2-17 *Multiprocessor device shutdown mode* on page 2-18	All
Clarified the sequences required for both entering and exiting Dormant mode.	*Dormant mode* on page 2-19	All
Clarified how the **STANDBYWFI[3:0]** and **STANDBYWFIL2** signals are used to communicate between the Cortex-A7 MPCore processor and the system power management controller.	*Communication to the Power Management Controller* on page 2-20	All
Combined the ThumbEE architecture section with the Jazelle Extension section from Chapter 3 *Programmers Model* into one section called Execution environment support.	*Execution environment support* on page 3-3	All
Updated reset value of the Main ID Register.	Table 4-2 on page 4-4 Table 4-16 on page 4-15	r0p4
Added clarification to the description of the PoU and PoC footnotes.	Table 4-8 on page 4-8 Table 4-21 on page 4-18	All
Updated bits[3:0] of the Main ID Register.	*Main ID Register* on page 4-26	r0p4
Clarified the value the LoC field takes to indicate L2 cache is not implemented on the processor.	Table 4-48 on page 4-49	All
Clarified the Coprocessor Access Control Register description.	*Coprocessor Access Control Register* on page 4-61	All

Table B-5 Differences between issue D and issue E (continued)

Change	Location	Affects
Clarified the Non-Secure Access Control Register description.	*Non-Secure Access Control Register* on page 4-65	All
Clarified the HDCR.TDRA, TDOSA, TDA, and TDE bit field descriptions.	Table 4-65 on page 4-69	All
Clarified the HSR.EC and IL bit field descriptions.	Table 4-73 on page 4-80	All
Added a footnote about not using SCTLR bit encodings TEX = 001, C = 1, and B = 0. This is to insure compatibility with other processors.	Table 5-1 on page 5-3	All
Combined the Memory types section with the Attributes section from Chapter 5 *Memory Management Unit* into one section called Memory types and attributes.	*Memory types and attributes* on page 5-3	All
Clarified the Walk cache RAM description.	*Walk cache RAM* on page 5-5	All
Updated synchronous and asynchronous aborts description.	*Synchronous and asynchronous aborts* on page 5-9	All
Clarified the instruction cache speculative memory accesses description.	*Instruction cache speculative memory accesses* on page 6-3	All
Clarified the **SYSBARDISABLE** signal setting for non-AXI3 modes, that is AXI configurations.	Table 7-1 on page 7-4 *AXI3 Compatibility mode* on page 7-9	All
Updated the GICD_IIDR value.	Table 8-3 on page 8-6	r0p4
Updated the GICD_IIDR.ProductID bit field value.	Table 8-5 on page 8-10	All
Updated the GICD_IIDR.Revision bit field value.	Table 8-5 on page 8-10	r0p4
Updated the GICC_IIDR value.	Table 8-8 on page 8-13	r0p4
Updated the GICC_IIDR.Revision bit field value.	Table 8-10 on page 8-16	r0p4
Updated the GICV_IIDR value.	Table 8-13 on page 8-18	r0p4
Clarified the DBGPCSR.PCS and T bit field descriptions.	Table 10-3 on page 10-10	All
Clarified the DBGBVR usage constraints and attributes.	*Breakpoint Value Registers* on page 10-13	All
Clarified the DBGBCR.BT, LBN, SCC, BAC, PMC, and E bit field descriptions.	Table 10-8 on page 10-14	All
Clarified the DBGWVR purpose, usage constraints, and attributes.	*Watchpoint Value Registers* on page 10-16	All
Clarified the DBGWCR usage constraints and attributes.	*Watchpoint Control Registers* on page 10-17	All
Clarified the DBGWCR.MASK, WT, LBN, SCC, HMC, PAC, and E bit field descriptions.	Table 10-10 on page 10-18	All
Updated the DBGOSLAR.OS Lock Access bit description.	Table 10-13 on page 10-21	All
Updated the Debug Peripheral ID2 value.	Table 10-24 on page 10-31	r0p4
Correct register name Debug Device ID Register 0.	*Debug Device ID Register* on page 10-30	r0p4
Updated the Performance Monitors Peripheral ID2 value.	Table 11-3 on page 11-8	r0p4
Removed the asynchronous terminology from the **nFIQ**, **nIRQ**, **nVFIQ**, and **nVIRQ** descriptions.	Table A-3 on page A-5	All