ARM® Cortex®-A9 MPCore

Revision: r4p1

Technical Reference Manual

ARM

ARM 100486_0401_10_en

ARM® Cortex®-A9 MPCore

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
A	04 April 2008	Non-Confidential	First release for r0p0
B	08 July 2008	Non-Confidential	First release for r0p1
C	16 December 2008	Non-Confidential	First release for r1p0
D	02 October 2009	Non-Confidential	First release for r2p0
E	27 November 2009	Non-Confidential	Second release for r2p0
F	30 April 2010	Non-Confidential	First release for r2p2
G	19 July 2011	Non-Confidential	First release for r3p0
H	23 March 2012	Non-Confidential	First release for r4p0
I	15 June 2012	Non-Confidential	First release for r4p1
0401-10	06 January 2016	Non-Confidential	Converted to DITA. Document number is changed to 100486. Second release for r4p1.

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

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

Unrestricted Access is an ARM internal classification.

Product Status

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

Web Address

http://www.arm.com

Contents

ARM® Cortex®-A9 MPCore Technical Reference Manual

Preface

This preface introduces the *ARM® Cortex®-A9 MPCore Technical Reference Manual*.

It contains the following:

About this book

This book is for the Cortex®-A9 MPCore. The Cortex-A9 MPCore consists of between one and four Cortex-A9 processors and a *Snoop Control Unit* (SCU) and other peripherals.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*m* Identifies the major revision of the product, for example, r1.

p*n* Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This book is written for hardware and software engineers implementing Cortex®-A9 system designs. The manual describes the external functionality of the Cortex-A9 MPCore. It provides information that enables designers to integrate the processor into a target system.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction
This chapter introduces the Cortex-A9 MPCore processor and its features.

Chapter 2 Snoop Control Unit
This chapter describes the *Snoop Control Unit* (SCU).

Chapter 3 Interrupt Controller
This chapter describes the implementation-defined features of the Interrupt Controller.

Chapter 4 Global timer, private timers, and watchdog registers
This chapter describes the timers and watchdog registers.

Chapter 5 Clocks, Resets, and Power Management
This chapter describes the clocks, resets, and power management features of the Cortex-A9 MPCore.

Chapter 6 Debug
This chapter describes some of the debug and trace considerations in Cortex-A9 MPCore designs.

Appendix A Signal Descriptions
This appendix describes the Cortex-A9 MPCore signals.

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

Glossary

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

See the *ARM Glossary* for more information.

Typographic conventions

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

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

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

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

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

`monospace bold`
: Denotes language keywords when used outside example code.

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

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

SMALL CAPITALS
: Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

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

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

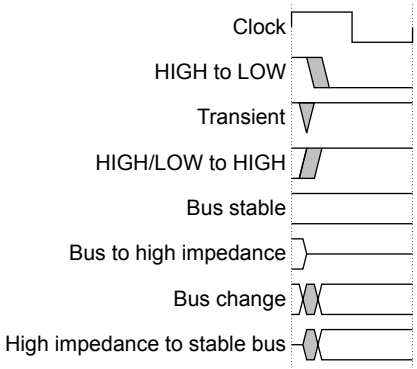

Figure 1 Key to timing diagram conventions

Signals

The signal conventions are:

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

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

Additional reading

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

ARM publications

- *ARM® Architecture Reference Manual, ARMv7-A and ARMv7-R edition* (ARM DDI 0406).
- *ARM® Cortex®-A9 Technical Reference Manual* (ARM DDI 0388).
- *ARM® Cortex®-A9 Floating-Point Unit Technical Reference Manual* (ARM DDI 0408).
- *ARM® Cortex®-A9 NEON™ Media Processing Engine Technical Reference Manual* (ARM DDI 0409).
- *ARM® Cortex®-A9 MBIST Technical Reference Manual* (ARM DDI 0414).
- *ARM® Cortex®-A9 Configuration and Sign-Off Guide* (ARM DII 0146).
- *AMBA® AXI Protocol Specification* (ARM IHI 0022).
- *ARM® Generic Interrupt Controller Architecture Specification* (ARM IHI 0048).
- *CoreSight™ PTM-A9 Technical Reference Manual* (ARM DDI 0401).
- *CoreSight™ PTM-A9 Integration Manual* (ARM DII 0162).
- *CoreSight™ Program Flow Trace Architecture Specification* (ARM IHI 0035).
- *CoreSight™ Technology System Design Guide* (ARM DGI 0012).
- *CoreSight™ Architecture Specification* (ARM IHI 0029).
- *ARM Debug Interface v5 Architecture Specification* (ARM IHI 0031).
- *CoreLink™ Level 2 Cache Controller L2C-310 Technical Reference Manual* (ARM DDI 0246).
- *RealView ICE and RealView Trace User Guide* (ARM DUI 0155).

Other publications

- JEP106M, *Standard Manufacture's Identification Code, JEDEC Solid State Technology Association*.

Feedback

Feedback on this product

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

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

Feedback on content

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

- The title *ARM® Cortex®-A9 MPCore Technical Reference Manual*.
- The number ARM 100486_0401_10_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note

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

Chapter 1
Introduction

This chapter introduces the Cortex-A9 MPCore processor and its features.

It contains the following sections:

1.1 About the Cortex-A9 MPCore processor

Description of the processor components and the example configuration.

This section contains the following subsections:

1.1.1 Processor components

The Cortex-A9 MPCore processor has three components.

These components are the following:

- From one to four Cortex-A9 processors in a cluster and a *Snoop Control Unit* (SCU) that can be used to ensure coherency within the cluster.
- A set of private memory-mapped peripherals, including a global timer, and a watchdog and private timer for each Cortex-A9 processor present in the cluster.
- An integrated Interrupt Controller that is an implementation of the Generic Interrupt Controller architecture. The integrated Interrupt Controller registers are in the private memory region of the Cortex-A9 MPCore processor.

Individual Cortex-A9 processors in the Cortex-A9 MPCore cluster can be implemented with their own hardware configurations. See the *ARM® Cortex®-A9 Technical Reference Manual* for additional information on possible Cortex-A9 processor configurations. ARM recommends that you implement uniform configurations for software ease of use.

There are other configuration options that affect Cortex-A9 MPCore system integration. The major options are:

- One or two AXI master port interfaces, with address filtering capabilities.
- An optional *Accelerator Coherency Port* (ACP) suitable for coherent memory transfers.
- A configurable number of interrupt lines.

Related references

1.3 Configurable options on page 1-15.

1.1.2 Example configuration

Figure showing an example multiprocessor configuration.

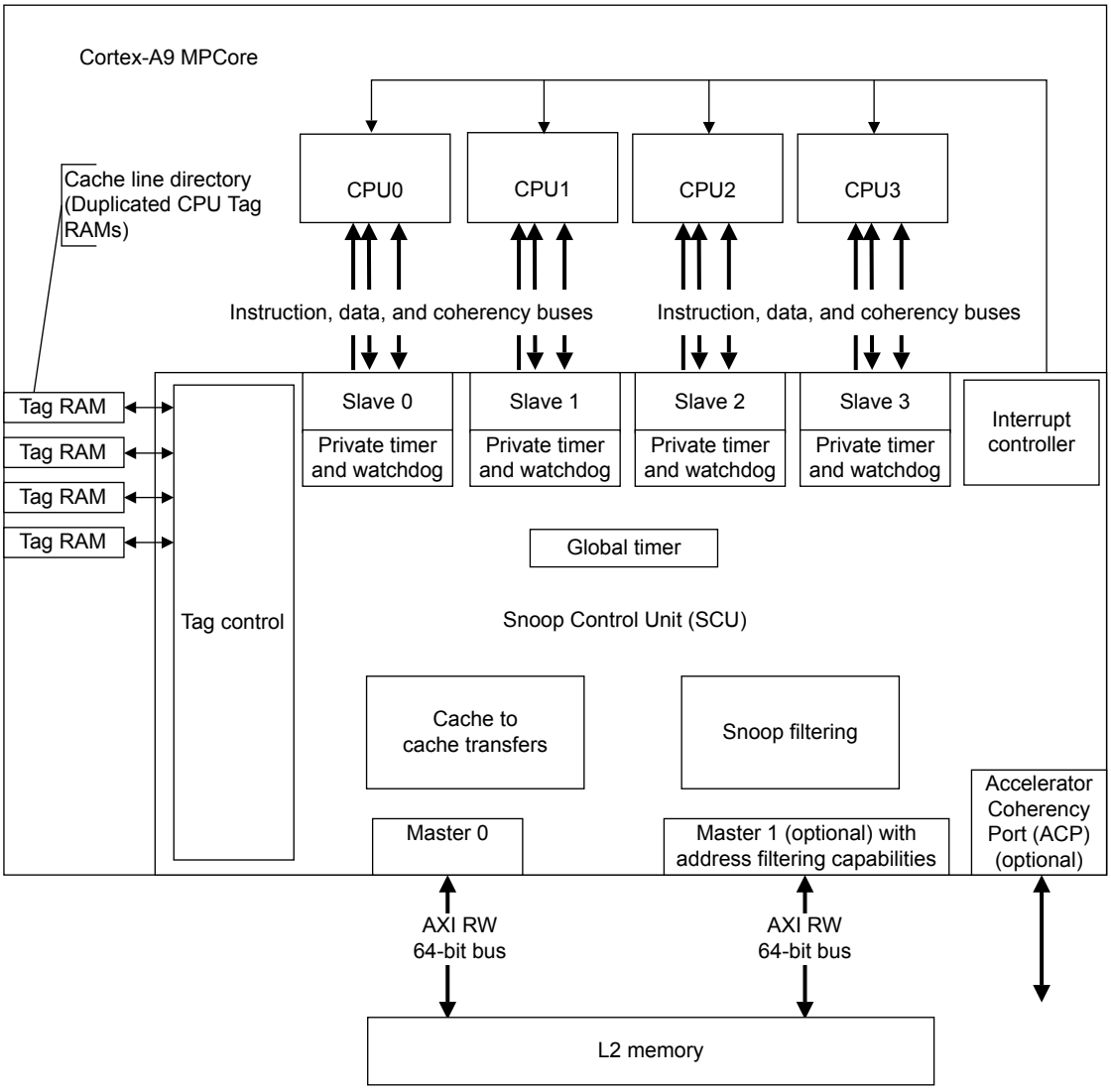

Figure 1-1 Example multiprocessor configuration

Note

It is possible to implement only one Cortex-A9 processor in a Cortex-A9 MPCore processor design. In this configuration, an SCU is still provided. The ACP, and an additional master port, are still available as configuration options.

1.2 Compliance

The Cortex-A9 processor complies with, or implements, several specifications. This manual complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

This section contains the following subsections:

1.2.1 ARM architecture

The Cortex-A9 processor implements the ARMv7-A architecture profile that includes the several architecture extensions.

These architecture extensions are the following:

- Advanced *Single Instruction Multiple Data* (SIMD) architecture extension for integer and floating-point vector operations.
- *Vector Floating-Point version 3* (VFPv3) architecture extension for floating-point computation that is fully compliant with the IEEE 754 standard.
- Security Extensions for enhanced security.
- Multiprocessing Extensions for multiprocessing functionality.

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

1.2.2 Advanced Microcontroller Bus Architecture

The Cortex-A9 processor complies with the AMBA 3 protocol.

See the *AMBA® AXI Protocol Specification*.

1.2.3 Program Flow Trace architecture

The Cortex-A9 processor implements the *Program Trace Macrocell* (PTM) based on the *Program Flow Trace* (PFT) v1.0 architecture profile.

See the *CoreSight™ Program Flow Trace Architecture Specification*.

1.2.4 Debug architecture

The Cortex-A9 processor implements the ARMv7 Debug architecture profile, that includes support for Security Extensions and CoreSight. See the CoreSight Architecture Specification.

1.2.5 Generic Interrupt Controller architecture

The Cortex-A9 processor implements the ARM *Generic Interrupt Controller* (GIC) v1.0 architecture profile.

1.3 Configurable options

List of Cortex-A9 MPCore processor configurable options.

Table 1-1 Configurable options for the Cortex-A9 MPCore processor

Feature	Options
Cortex-A9 processors	One to four
Instruction cache size per Cortex-A9 processor	16KB, 32KB, or 64KB
Data cache size per Cortex-A9 processor	16KB, 32KB, or 64KB
TLB size per Cortex-A9 processor	64, 128, 256 or 512 entries
BTAC size per Cortex-A9 processor	512, 1024, 2048 or 4096 entries
GHB size	1024, 2048, 4096, 8192 or 16384 descriptors
Instruction micro TLB per Cortex-A9 processor	32 or 64 entries
Media Processing Engine with NEON technology per Cortex-A9 processor	Included or not [a]
FPU per Cortex-A9 processor	Included or not [b]
Preload Engine per Cortex-A9 processor	Included or not
Number of entries in the Preload Engine FIFO per Cortex-A9 processor	16, 8, or 4
Jazelle® DBX extension per Cortex-A9 processor	Full or trivial
Program Trace Macrocell (PTM) interface per Cortex-A9 processor	Included or not
Power off and dormant mode wrappers	Included or not
Support for parity error detection	Included or not [c]
ARM_BIST	Included or not
Master ports	One or two
Accelerator Coherency Port	One, included or not
Shared Peripheral Interrupts (SPIs)	0-224, in steps of 32

[a] Includes support for floating-point operations. If this option is implemented, then the FPU option cannot also be implemented.
[b] If this option is implemented then the Media Processing Engine with NEON technology option cannot also be implemented.
[c] The *ARM® Cortex®-A9 Technical Reference Manual* describes the parity error scheme. See *A.11 Parity error signals* on page Appx-A-112 for a description of the signals.

Parity error detection is not supported on the GHB RAMs when implementing an 8192 or 16384-entry GHB configuration.

1.4 Test features

The Cortex-A9 processor provides test signals that enable the use of both ATPG and MBIST to test the Cortex-A9 processor and its memory arrays.

See the *Cortex®-A9 MBIST Controller Technical Reference Manual*.

Related references

Appendix A Signal Descriptions on page Appx-A-90.

1.5 Private Memory Region

All registers accessible by all Cortex-A9 processors within the Cortex-A9 MPCore are grouped into two contiguous 4KB pages accessed through a dedicated internal bus. The base address of these pages is defined by the pins **PERIPHBASE[31:13]**.

Cortex-A9 MPCore global control and peripherals must be accessed through memory-mapped transfers to the Cortex-A9 MPCore private memory region.

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

Access to the private memory region is little-endian only.

Access these registers with single load/store instructions. Load or store multiple accesses cause an abort to the requesting Cortex-A9 processor and the Fault Status Register shows this as a SLVERR.

The following table shows the permitted access sizes for the private memory regions.

Table 1-2 Permitted access sizes for private memory regions

Private memory region	Permitted access sizes			
	Byte	Halfword	Word	Doubleword
Global timer, private timers, and watchdogs	No	No	Yes	No
SCU registers	Yes	No	Yes	No
Cortex-A9 processor interrupt interfaces				
Interrupt distributor				

Note

Halfword or doubleword accesses cause an abort to the requesting Cortex-A9 processor and the Fault Status Register shows this as a SLVERR.

A word access with strobes not all set causes an abort to the requesting Cortex-A9 processor and the Fault Status Register shows this as a SLVERR.

The *Accelerator Coherency Port* (ACP) cannot access any of the registers in this memory region.

The following table shows register addresses for the Cortex-A9 MPCore processor relative to this base address.

Table 1-3 Cortex-A9 MPCore private memory region

Offset from PERIPHBASE[31:13]	Peripheral	Description
0x0000 - 0x00FC	SCU registers	*Chapter 2 Snoop Control Unit* on page 2-24
0x0100 - 0x01FF	Interrupt controller interfaces	*Chapter 3 Interrupt Controller* on page 3-47
0x0200 - 0x02FF	Global timer	*4.3 About the Global Timer* on page 4-70
0x0300 - 0x03FF	-	-
0x0400 - 0x04FF	-	-
0x0500 - 0x05FF	-	-

Table 1-3 Cortex-A9 MPCore private memory region (continued)

Offset from PERIPHBASE[31:13]	Peripheral	Description
0x0600 - 0x06FF	Private timers and watchdogs	*4.2 Private timer and watchdog registers* on page 4-64
0x0700 - 0x07FF	Reserved	Double word and halfword accesses generate a SLVERR data abort. Byte and word accesses complete without error.
0x0800 - 0x08FF		
0x0900 - 0x09FF		
0x0A00 - 0x0AFF		
0x0B00 - 0x0FFF		
0x1000 - 0x1FFF	Interrupt Distributor	*3.1.2 Interrupt Distributor interrupt sources* on page 3-48

Related references

A.4 Configuration signals on page Appx-A-95.

1.6 Interfaces

The Cortex-A9 MPCore processor has the four different interfaces. These interfaces are the AMBA AXI interfaces, the interrupts interface, the debug interfaces, and the Design for Test interface.

This section contains the following subsections:

1.6.1 AMBA AXI interfaces

The AMBA AXI interfaces include one or two AXI Master port interfaces, and one *Accelerator Coherency* (ACP) AXI Slave port.

See the *AMBA® AXI Protocol Specification*.

Related concepts

2.3 AMBA AXI Master Port Interfaces on page 2-36.

1.6.2 Interrupts interface

The Cortex-A9 MPCore processor provides the legacy nIRQ and nFIQ interrupt lines for each individual Cortex-A9 processor present in the cluster.

The Cortex-A9 MPCore processor also provides a separate interrupt interface, with a configurable number of interrupts lines, up to 224, connected to its internal Interrupt Controller.

Related references

Chapter 3 Interrupt Controller on page 3-47.

1.6.3 Debug interfaces

The external debug interface of the Cortex-A9 MPCore processor is compliant with the ARMv7 Debug Architecture that includes support for Security Extensions and CoreSight.

Except for a few debug configuration signals, the debug interfaces of the individual Cortex-A9 processors are presented externally so that each processor can be debugged independently.

The Cortex-A9 MPCore processor also provides an external Debug APB interface for memory-mapped accesses to debug and performance monitor registers.

Related references

Chapter 6 Debug on page 6-86.

1.6.4 Design for Test interface

Gives information about the *Memory Built In Self Test* (MBIST) interface.

See the *ARM® Cortex®-A9 MBIST Controller Technical Reference Manual*.

1.7 MPCore considerations

You must consider the processor coherency, the registers, and the maintenance operations when you use multiprocessing.

This section contains the following subsections:

1.7.1 About Cortex-A9 MPCore coherency

Memory coherency in a Cortex-A9 MPCore processor is maintained following a weakly ordered memory consistency model.

Cache coherency among L1 data caches of the Cortex-A9 processors in the cluster is maintained when the Cortex-A9 processors are operating in *Symmetric Multi-Processing* (SMP) mode. This mode is controlled by the SMP bit of the Auxiliary Control Register.

To be kept coherent, the memory must be marked as Write-Back, Shareable, Normal memory.

——— **Note** ———

When the Shareable attribute is applied to a memory region that is not Write-Back Normal memory, data held in this region is treated as Noncacheable.

———————————

1.7.2 Registers with multiprocessor uses

The Auxiliary Control Register, the Configuration Base Address Register, and the Multiprocessor Affinity Register have multiprocessor uses.

See *ARM® Cortex®-A9 Technical Reference Manual* for more information.

1.7.3 Maintenance operations broadcasting

All processors working in SMP mode on the same coherent domain can send and receive TLB and Cache Maintenance operations.

The *ARM® Architecture Reference Manual, ARMv7-A and ARMv7-R edition* gives detailed information on broadcast operations.A Cortex-A9 processor in the A9-MP cluster broadcasts broadcastable maintenance operation when it operates in SMP mode (ACTLR.SMP=1) and when the maintenance operation broadcasting is enabled (ACTLR.FW=1). A Cortex-A9 processor can receive and execute broadcast maintenance operations when it operates in SMP mode, ACTLR.SMP=1.

1.8 Product documentation and design flow

List of Cortex-A9 MPCore documents and how these documents relate to the design flow.

See the list of external documents for more information about the books described in this section. For information on the relevant architectural standards and protocols, see *1.2 Compliance* on page 1-14.

This section contains the following subsections:

1.8.1 Documentation

The Cortex-A9 MPCore documentation is summarized in the *Technical Reference Manual* (TRM).

Technical Reference Manual

The TRM describes the functionality and the effects of functional options on the behavior of the Cortex-A9 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-A9 MPCore processor, then contact:

- The implementer to determine:
  - The build configuration of the implementation.
  - What integration, if any, was performed before implementing the Cortex-A9 MPCore processor.
- The integrator to determine the pin configuration of the device that you are using.

1.8.2 Design flow

The Cortex-A9 MPCore processor is delivered as synthesizable RT. Before it can be used in a product, it must go through the several processes.

These processes are the following:

Implementation

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

This is the last process. The system programmer develops the software required to configure and initialize the Cortex-A9 MPCore processor, and tests the required application software.

Each process:

- Can be performed by a different party.
- Can include implementation and integration choices affect the behavior and features of the Cortex-A9 MPCore processor.

For MCUs, often a single design team integrates the processor before synthesizing the complete design. Alternatively, the team can synthesize the processor on its own or partially integrated, to produce a macrocell that is then integrated, possibly by a separate team.

The operation of the final device depends on:

Build configuration

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

Configuration inputs

The integrator configures some features of the Cortex-A9 MPCore processor by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made. They can also limit the options available to the software.

Software configuration

The programmer configures the Cortex-A9 MPCore processor by programming particular values into registers. This affects the behavior of the Cortex-A9 MPCore processor.

——— **Note** ———

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

———————————

1.9 Product revisions

List of differences in functionality between product revisions.

These differences are in addition to those described in the *ARM® Cortex®-A9 Technical Reference Manual*:

r0p0 First release.

r0p0-r0p1 The differences between the two revisions are:

- r0p1 includes fixes for all known engineering errata relating to r0p0
- r0p1 includes an upgrade of the micro TLB entries from 8 to 32 entries, on both the Instruction and Data side.

Neither of these changes affect the functionality described in this document.

r0p1-r1p0 Functional changes are:

- In r1p0, there is a global timer. See *4.3 About the Global Timer* on page 4-70.
- In the Interrupt Controller, **INT** becomes **IRQS**. See *3.3.10 SPI Status Registers* on page 3-57.
- SCU CPU Power Status Register bits reassigned. See *2.2.4 SCU CPU Power Status Register* on page 2-29.

r1p0-r2p0 Functional changes are:

- Conditions for coherent snoop for ACP requests amended. See *2.4.1 ACP requests* on page 2-43.
- SCU Control register updated. See *2.2.2 SCU Control Register* on page 2-27:
  - Bit 6 to enable additional clock gating on GIC,
  - Bit 5 to enable additional clock gating on SCU.
- SCU Secure Access Control Register renamed to SCU Non-secure Access Control Register. See *2.2.9 SCU Non-secure Access Control Register* on page 2-34.
- Removal of SCU Invalidate All Registers in Non-secure State Register and functionality. See *2.2 SCU registers* on page 2-26.
- Added speculative linefill feature to optimize L1 miss and L2 hit latency, See *2.2.2 SCU Control Register* on page 2-27. Bit 3.
- Added **SCUIDLE output.** See *2.2.4 SCU CPU Power Status Register* on page 2-29
- Added Filtering capabilities in the SCU for Device accesses. See *2.3.6 Device accesses filtering* on page 2-40.
- **PERIPHCLK** can be turned off. See *5.1 Clocks* on page 5-76
- Change to the behavior of the comparators for each processor with the global timer. See *4.3 About the Global Timer* on page 4-70
- **Added PMUEVENT** See *A.9 Performance monitoring signals* on page Appx-A-110

r2p0-r2p1 No change.

r2p1-r2p2 No change.

r2p2-r3p0 No change.

r3p0-r4p0 Added configuration options for the TLB, BTAC, and GHB sizes. See *1.3 Configurable options* on page 1-15.

r4p0-r4p1 No change.

Chapter 2
Snoop Control Unit

This chapter describes the *Snoop Control Unit* (SCU).

It contains the following sections:

2.1 About the SCU

The SCU connects one to four Cortex-A9 processors to the memory system through the AXI interfaces.

The SCU functions are to:

- Maintain data cache coherency between the Cortex-A9 processors.
- Initiate L2 AXI memory accesses.
- Arbitrate between Cortex-A9 processors requesting L2 accesses.
- Manage ACP accesses.

Note

The Cortex-A9 SCU does not support hardware management of coherency of the instruction cache.

This section contains the following subsections:

- *2.1.1 TrustZone® extensions* on page 2-25.
- *2.1.2 SCU event monitoring* on page 2-25.

2.1.1 TrustZone® extensions

The SCU implements support for the ARM Architecture security extensions.

Related references

2.2.8 SCU Access Control Register (SAC) on page 2-33.
2.2.9 SCU Non-secure Access Control Register on page 2-34.

2.1.2 SCU event monitoring

The individual CPU event monitors can be configured to gather statistics on the operation of the SCU.

The *ARM® Cortex®-A9 Technical Reference Manual* describes event monitoring.

2.2 SCU registers

Summary of SCU registers with information on purpose, usage contraints, configurations, and attributes for each register.

This section contains the following subsections:

2.2.1 SCU register summary

List of SCU registers. Addresses are relative to the base address of the region for the SCU memory map, that is **PERIPHBASE[31:13]**.

All SCU registers are byte accessible and are reset by **nSCURESET**.

Table 2-1 SCU registers summary

Offset from PERIPHBASE [31:13]	Name	Security state		Reset value	Banked	Page
		Secure	Non-secure			
0x00	SCU Control Register	RW	RW	Implementation defined	No	*2.2.2 SCU Control Register* on page 2-27
0x04	SCU Configuration Register	RO	RO	Implementation defined	No	*2.2.3 SCU Configuration Register* on page 2-28
0x08	SCU CPU Power Status Register	RW	RW	Implementation defined	No	*2.2.4 SCU CPU Power Status Register* on page 2-29
0x0C	SCU Invalidate All Registers in Secure State	WO	-	0x00000000	No	*2.2.5 SCU Invalidate All Registers in Secure State Register* on page 2-31
0x40	Filtering Start Address Register	RW	RW	Defined by **FILTERSTART** input	No	*2.2.6 Filtering Start Address Register* on page 2-31
0x44	Filtering End Address Register	RW	RW	Defined by **FILTEREND** input	No	*2.2.7 Filtering End Address Register* on page 2-32
0x50	*SCU Access Control* (SAC) Register	RW	RW	0x0000000F	No	*2.2.8 SCU Access Control Register (SAC)* on page 2-33
0x54	*SCU Non-secure Access Control* (SNSAC) Register	RW	RO	0x00000000	No	*2.2.9 SCU Non-secure Access Control Register* on page 2-34

Note

Secure RW and WO registers are writable if the relevant bits in the SAC are set.

SCU registers must not be written with NEON STR instructions.

2.2.2 SCU Control Register

Characteristics and bit assignments of the SCU Control Register.

Purpose

- Enables speculative linefills to L2 with L2C-310.
- Enables Force all Device to port0.
- Enables IC standby mode.
- Enables SCU standby mode.
- Enables SCU RAM parity support.
- Enables address filtering.
- Enables the SCU.

Usage constraints

- This register is writable in Secure state if the relevant bit in the SAC register is set.
- This register is writable in Non-secure state if the relevant bits in the SAC and SNSAC registers are set.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *2.2.1 SCU register summary* on page 2-26.

The following figure shows the SCU Control Register bit assignments.

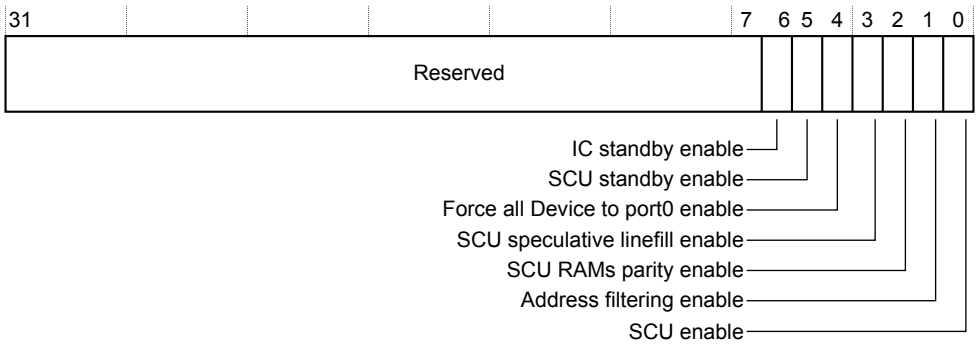

Figure 2-1 SCU Control Register bit assignments

The following table shows the SCU Control Register bit assignments.

Table 2-2 SCU Control Register bit assignments

Bits	Name	Function
[31:7]	-	Reserved
[6]	IC standby enable	When set, this stops the Interrupt Controller clock when no interrupts are pending, and no CPU is performing a read/write request. This bit is set to 0 by default
[5]	SCU standby enable	When set, **SCU CLK** is turned off when all processors are in WFI mode, there is no pending request on the ACP, if implemented, and there is no remaining activity in the SCU. When **SCU CLK** is off, **ARREADYS**, **AWREADYS,** and **WREADYS** on the ACP are forced LOW. The clock is turned on when any processor leaves WFI mode, or if there is a new request on the ACP. This bit is set to 0 by default

Table 2-2 SCU Control Register bit assignments (continued)

Bits	Name	Function
[4]	Force all Device to port0 enable	When set, all requests from the ACP or processors with **AxCACHE** = Noncacheable Bufferable are forced to be issued on the AXI Master port M0. See *2.3.5 Address filtering capabilities* on page 2-40. This bit is set to 0 by default
[3]	SCU Speculative linefills enable	When set, coherent linefill requests are sent speculatively to the L2C-310 in parallel with the tag look-up. If the tag look-up misses, the confirmed linefill is sent to the L2C-310 and gets RDATA earlier because the data request was already initiated by the speculative request. This feature works only if the L2C-310 is present in the design. This bit is set to 0 by default
[2]	SCU RAMs Parity enable	**0** Parity off. This is the default setting. **1** Parity on. This bit is always zero if support for parity is not implemented.
[1]	Address filtering enable	**0** Addressing filtering off. **1** Addressing filtering on. The default value is the value of **FILTEREN** sampled when **nSCURESET** is deasserted. This bit is always zero if the SCU is implemented in the single master port configuration. See *2.3.5 Address filtering capabilities* on page 2-40.
[0]	SCU enable	**0** SCU disable. This is the default setting. **1** SCU enable.

2.2.3 SCU Configuration Register

Characteristics and bit assignments of the SCU Configuration Register.

Purpose
- Read tag RAM sizes for the Cortex-A9 processors that are present.
- Determine the Cortex-A9 processors that are taking part in coherency.
- Read the number of Cortex-A9 processors present.

Usage constraints This register is read-only.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *2.2.1 SCU register summary* on page 2-26.

The following figure shows the SCU Configuration Register bit assignments.

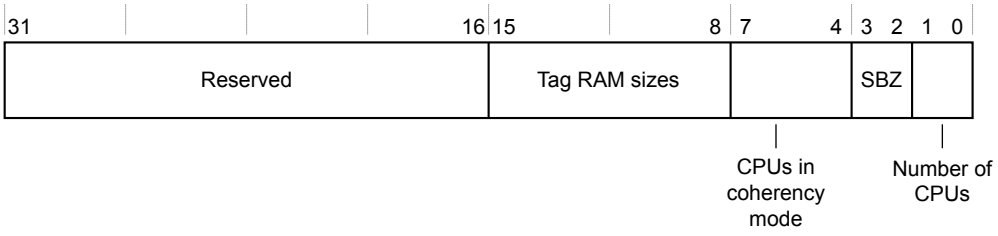

Figure 2-2 SCU Configuration Register bit assignments

The following table shows the SCU Configuration Register bit assignments.

Table 2-3 SCU Configuration Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved, SBZ.
[15:8]	Tag RAM sizes	Bits [15:14] indicate Cortex-A9 processor CPU3 tag RAM size if present. Bits [13:12] indicate Cortex-A9 processor CPU2 tag RAM size if present. Bits [11:10] indicate Cortex-A9 processor CPU1 tag RAM size if present. Bits [9:8] indicate Cortex-A9 processor CPU0 tag RAM size. The encoding is as follows: `0b00` 16KB cache, 64 indexes per tag RAM. `0b01` 32KB cache, 128 indexes per tag RAM. `0b10` 64KB cache, 256 indexes per tag RAM. `0b11` Reserved Non-present CPUs have a Tag RAM size of `0b00`, the same as 16KB.
[7:4]	CPUs SMP	Shows the Cortex-A9 processors that are in *Symmetric Multi-processing* (SMP) or *Asymmetric Multi-processing* (AMP) mode. **0** This Cortex-A9 processor is in AMP mode, not taking part in coherency, or not present. **1** This Cortex-A9 processor is in SMP mode, taking part in coherency. Bit 7 is for CPU3 Bit 6 is for CPU2 Bit 5 is for CPU1 Bit 4 is for CPU0.
[3:2]	-	Reserved, SBZ
[1:0]	CPU number	Number of CPUs present in the Cortex-A9 MPCore processor `0b00` One Cortex-A9 processor, CPU0. `0b01` Two Cortex-A9 processors, CPU0 and CPU1. `0b10` Three Cortex-A9 processors, CPU0, CPU1, and CPU2. `0b11` Four Cortex-A9 processors, CPU0, CPU1, CPU2, and CPU3.

2.2.4 SCU CPU Power Status Register

Characteristics and bit assignments of the SCU CPU Power Status Register.

Purpose Specifies the state of the Cortex-A9 processors with reference to power modes.

Usage constraints

This register is writable in Secure state if the relevant bit in the SAC register is set.

This register is writable in Non-secure state if the relevant bits in the SAC and SNSAC registers are set.

Dormant mode and powered-off mode are controlled by an external power controller. SCU CPU Status Register bits indicate to the external power controller the power domains that can be powered down.

Before entering any other power mode than Normal, the Cortex-A9 processor must set its status field to signal to the power controller the mode it is about to enter. The Cortex-A9 processor then executes a WFI entry instruction. When in WFI state, the **PWRCTLOn** bus is enabled and signals to the power controller what it must do with power domains.

The SCU CPU Power Status Register bits can also be read by a Cortex-A9 processor exiting low-power mode to determine its state before executing its reset setup.

Cortex-A9 processors status fields take **PWRCTLIn** values at reset, except for nonpresent Cortex-A9 processors. For nonpresent Cortex-A9 processors, writing to this field has no effect.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *2.2.1 SCU register summary* on page 2-26.

The following figure shows the SCU CPU Power Status Register bit assignments.

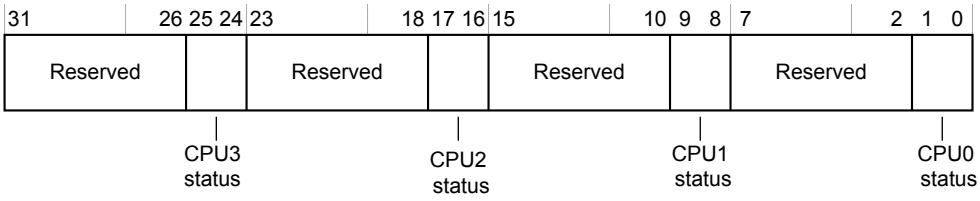

Figure 2-3 SCU CPU Power Status Register bit assignments

The following table shows the SCU CPU Power Status Register bit assignments.

Table 2-4 SCU CPU Power Status Register bit assignments

Bits	Name	Function
[31:26]	-	Reserved, SBZ
[25:24]	CPU3 status	Power status of the Cortex-A9 processor: `0b00` Normal mode. `0b01` Reserved. `0b10` The Cortex-A9 processor is about to enter, or is in, dormant mode. No coherency request is sent to the Cortex-A9 processor. `0b11` The Cortex-A9 processor is about to enter, or is in, powered-off mode, or is nonpresent. No coherency request is sent to the Cortex-A9 processor. The default value is `0b00` when CPU3 processor is present, else `0b11`
[23:18]	-	Reserved, SBZ
[17:16]	CPU2 status	Power status of the Cortex-A9 processor. The default value is `0b00` when CPU2 processor is present, else `0b11`
[15:10]	-	Reserved, SBZ

Table 2-4 SCU CPU Power Status Register bit assignments (continued)

Bits	Name	Function
[9:8]	CPU1 status	Power status of the Cortex-A9 processor. The default value is `0b00` when CPU1 processor is present, else `0b11`
[7:2]	-	Reserved, SBZ
[1:0]	CPU0 status	Power status of the Cortex-A9 processor. The default value is `0b00` when CPU0 processor is present, else `0b11`

2.2.5 SCU Invalidate All Registers in Secure State Register

Characteristics and bit assignments of the SCU Invalidate All Registers in Secure State.

Purpose Invalidates the SCU tag RAMs on a per Cortex-A9 processor and per way basis.

Usage constraints This register:
- Invalidates all lines in the selected ways.
- Is a write-only register that always reads as zero.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *2.2.1 SCU register summary* on page 2-26.

The following figure shows the SCU Invalidate All Register in Secure state bit assignments.

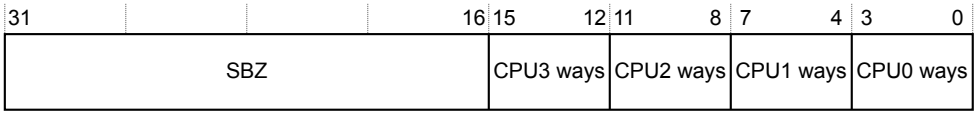

Figure 2-4 SCU Invalidate All Registers in Secure state bit assignments

The following table shows the SCU Invalidate All Register in Secure state bit assignments.

Table 2-5 SCU Invalidate All Registers in Secure state bit assignments

Bits	Name	Function
[31:16]	-	-
[15:12]	CPU3 ways	Specifies the ways that must be invalidated for CPU3. Writing to these bits has no effect if the Cortex-A9 MPCore processor has fewer than four processors.
[11:8]	CPU2 ways	Specifies the ways that must be invalidated for CPU2. Writing to these bits has no effect if the Cortex-A9 MPCore processor has fewer than three processors.
[7:4]	CPU1 ways	Specifies the ways that must be invalidated for CPU1. Writing to these bits has no effect if the Cortex-A9 MPCore processor has fewer than two processors.
[3:0]	CPU0 ways	Specifies the ways that must be invalidated for CPU0.

2.2.6 Filtering Start Address Register

Characteristics and bit assignments of the Filtering Start Address Register.

Purpose Provides the start address for use with master port 1 in a two-master port configuration.

Usage constraints This register is writable:
- In Secure state if the relevant bit in the SAC register is set.
- In Non-secure state if the relevant bits in the SAC and SNSAC registers are set.

Configurations Available in all two-master port configurations. When only one master port is present, these registers are not implemented. Writes have no effect and reads return a value `0x0` for all filtering registers.

Attributes See the register summary in *2.2.1 SCU register summary* on page 2-26.

The following figure shows the Filtering Start Address Register bit assignments.

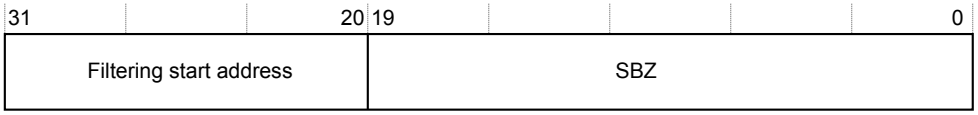

Figure 2-5 Filtering Start Address Register bit assignments

The following table shows the Filtering Start Address Register bit assignments.

Table 2-6 Filtering Start Address Register bit assignments

Bits	Name	Function
[31:20]	Filtering start address	Start address for use with master port 1 in a two-master port configuration when address filtering is enabled. The default value is the value of **FILTERSTART** sampled on exit from reset. The value on the pin gives the upper address bits with 1MB granularity.
[19:0]	-	SBZ

Related references

A.4 Configuration signals on page Appx-A-95.

2.2.7 Filtering End Address Register

Characteristics and bit assignments of the Filtering End Address Register.

Purpose Provides the end address for use with master port 1 in a two-master port configuration.

Usage constraints This register is writable

- In Secure state if the relevant bit in the SAC register is set.
- In Non-secure state if the relevant bits in the SAC and SNSAC registers are set.
- Has an inclusive address as its end address. This means that the topmost megabyte of address space of memory can be included in the filtering address range.

Configurations Available in all two-master product configurations. When only one master port is present, writes have no effect and reads return a value `0x0` for all filtering registers.

Attributes See the register summary in *2.2.1 SCU register summary* on page 2-26.

The following figure shows the Filtering End Address Register bit assignments.

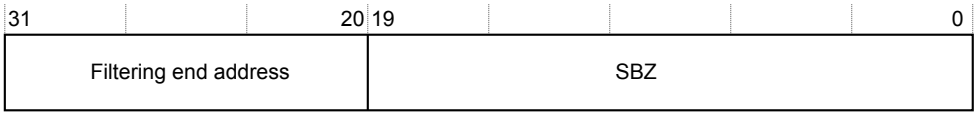

Figure 2-6 Filtering End Address Register bit assignments

The following table shows the Filtering End Address Register bit assignments.

Table 2-7 Filtering End Address Register bit assignments

Bits	Name	Function
[31:20]	Filtering end address	End address for use with master port 1 in a two-master port configuration, when address filtering is enabled. The default value is the value of **FILTEREND** sampled on exit from reset. The value on the pin gives the upper address bits with 1MB granularity.
[19:0]	-	SBZ.

Related references

A.4 Configuration signals on page Appx-A-95.

2.2.8 SCU Access Control Register (SAC)

Characteristics and bit assignments of the SAC.

Purpose Controls access to the following registers on a per Cortex-A9 processor basis:

- *2.2.2 SCU Control Register* on page 2-27.
- *2.2.4 SCU CPU Power Status Register* on page 2-29.
- *2.2.5 SCU Invalidate All Registers in Secure State Register* on page 2-31.
- *2.2.6 Filtering Start Address Register* on page 2-31.
- *2.2.7 Filtering End Address Register* on page 2-32.
- *2.2.9 SCU Non-secure Access Control Register* on page 2-34.

A processor in the Cortex-A9 MPCore multiprocessor can set up the SCU and then write zero to the register. This prevents any Secure or Non-secure access from altering the configuration of the register again. This prevents any more changes to the SCU configuration after booting.

Usage constraints This register is writable:

- In Secure state if the relevant bit in the SAC register is set.
- In Non-secure state if the relevant bits in the SAC and SNSAC are set.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *2.2.1 SCU register summary* on page 2-26.

The following figure shows the SAC register bit assignments.

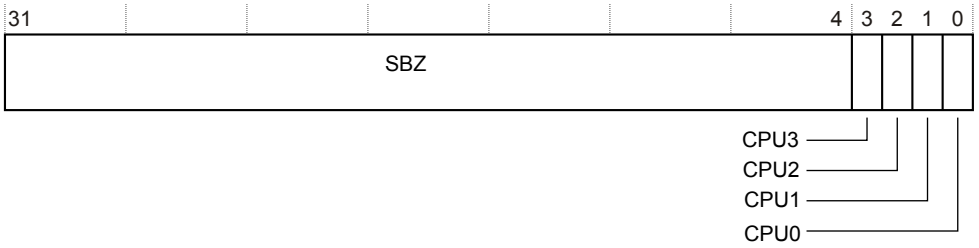

Figure 2-7 SAC register bit assignments

The following table shows the SAC register bit assignments.

Table 2-8 SAC register bit assignments

Bits	Name	Function	
[31:4]	SBZ	-	
[3]	CPU3	**0**	CPU3 cannot access the registers. [d]
		1	CPU3 can access the registers. This is the default.
[2]	CPU2	**0**	CPU2 cannot access the registers.
		1	CPU2 can access the registers. This is the default.
[1]	CPU1	**0**	CPU1 cannot access the registers.
		1	CPU1 can access the registers. This is the default.
[0]	CPU0	**0**	CPU0 cannot access the registers.
		1	CPU0 can access the registers. This is the default.

2.2.9 SCU Non-secure Access Control Register

Characteristics and bit assignments of the SNSAC register.

Purpose Controls Non-secure access to the following registers on a per Cortex-A9 processor basis:

- *2.2.2 SCU Control Register* on page 2-27.
- *2.2.4 SCU CPU Power Status Register* on page 2-29.
- *2.2.6 Filtering Start Address Register* on page 2-31.
- *2.2.7 Filtering End Address Register* on page 2-32.
- *2.2.8 SCU Access Control Register (SAC)* on page 2-33.

In addition, it controls Non-secure access to the global timer, private timers, and watchdog.

Usage constraints This register is writable in Secure state if the relevant bit in the SAC register is set.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *2.2.1 SCU register summary* on page 2-26.

The following figure shows the SNSAC register bit assignments.

[d] The accessible registers are the SAC Register, the SCU Control Register, the SCU CPU Status Register, the SCU Invalidate All Register in Secure State, the filtering registers, and the SCU CPU Power Status register.

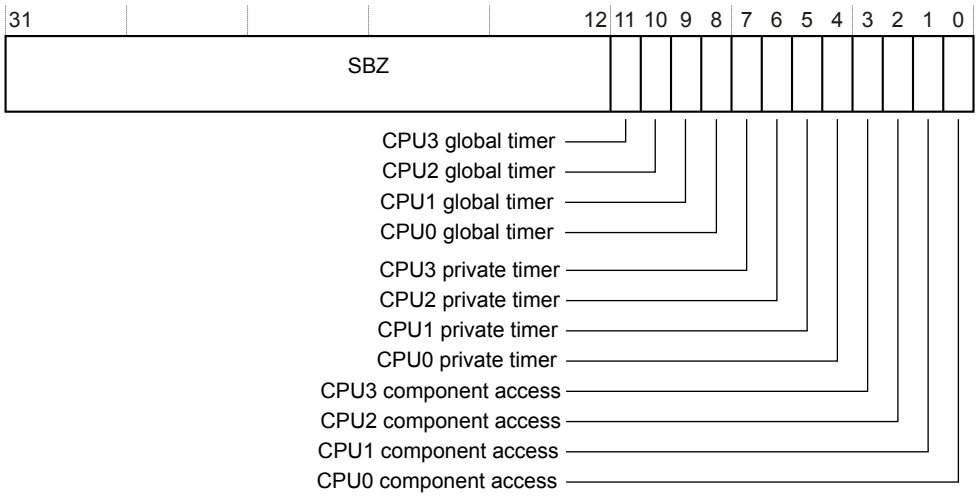

Figure 2-8 SNSAC register bit assignments

The following table shows the SNSAC register bit assignments.

Table 2-9 SNSAC register bit assignments

Bits	Name	Function
[31:12]	SBZ	-
[11]	CPU3 global timer	Non-secure access to the global timer for CPU<n>. • <n> is 3 for bit[11] • <n> is 2 for bit[10] • <n> is 1 for bit[9] • <n> is 0 for bit[8]. **0** Secure accesses only. This is the default value. **1** Secure accesses and Non-secure accesses.
[10]	CPU2 global timer	
[9]	CPU1 global timer	
[8]	CPU0 global timer	
[7]	Private timers for CPU<n>	Non-secure access to the private timer and watchdog for CPU<n>. • <n> is 3 for bit[7] • <n> is 2 for bit[6]] • <n> is 1 for bit[5] • <n> is 0 for bit[4]. **0** Secure accesses only. Non-secure reads return 0. This is the default value. **1** Secure accesses and Non-secure accesses.
[6]		
[5]		
[4]		
[3]	Register access for CPU<n>	Non-secure access to the registers for CPU<n>. • <n> is 3 for bit[3] • <n> is 2 for bit[2]] • <n> is 1 for bit[1] • <n> is 0 for bit[0]. **0** CPU cannot write the registers. **1** CPU can access the registers.
[2]		
[1]		
[0] [e]		

[e] The accessible registers are the SAC Register, the SCU Control Register, the SCU CPU Status Register, the filtering registers, and the SCU CPU Power Status Register.

2.3 AMBA AXI Master Port Interfaces

Description of the AMBA AXI interfaces such as AXI issuing capabilities, AXI transactions and transaction IDs, attribute encodings, address filtering capabilities, and clocking in different interfaces.

This section contains the following subsections:

2.3.1 AXI issuing capabilities

The Cortex-A9 MPCore L2 interface can have two 64-bit wide AXI bus masters. In a two bus master configuration, there is also an option to configure address filtering.

The following table shows the AXI master interface attributes.

Table 2-10 AXI master interface attributes

Attribute	Format
Write Issuing Capability	10 per processor, including: • Eight non-cacheable writes. • Two evictions. Two additional writes can also be performed for eviction traffic from the SCU. Three more write transactions can be issued if the ACP is implemented.
Read Issuing Capability	14 per processor, including: • Four instruction reads. • Six linefill reads. • Four non-cacheable read. Seven more read transactions can be issued if the ACP is implemented.
Combined Issuing Capability	Up to 24 per processor. Plus 2 for SCU evictions. Ten more transactions can be issued, if the ACP is implemented.
Write ID Capability	32
Write Interleave Capability	1
Write ID Width	6
Read ID Capability	32
Read ID Width	6

The AXI protocol and meaning of each AXI signal are not described in this document. For more information, see *AMBA® AXI Protocol v1.0 Specification*.

Note

These numbers are the theoretical maximums for the Cortex-A9 MP processor. A typical system is unlikely to reach these numbers. ARM recommends that you perform profiling to tailor your system resources appropriately for optimum performance.

Related concepts

2.3.5 Address filtering capabilities on page 2-40.

2.3.2 Cortex-A9 MPCore AXI transactions

Cortex-A9 MPCore contains up to four individual Cortex-A9 processors that can generate only a subset of all AXI transactions. As a consequence, only this subset of AXI transactions can appear on the Cortex-A9 MPCore master ports.

However, when the ACP is implemented, ACP traffic can generate transactions not defined in this list.

For more information on AXI transactions, see *ARM® Cortex®-A9 Technical Reference Manual*.

2.3.3 AXI transaction IDs

List of AXI transaction IDs and AXI USER bits.

There are several possible sources for the AXI transactions a Cortex-A9MP processor issues on its AXI master ports.

ARIDMx[5:0] encodings

List of **ARIDMx[5:0]** encodings for read transactions.

As the following table shows, the **ARIDMx[2]** encodings distinguish between transactions originating from Cortex-A9 processors and transactions originating from the ACP:

- **ARIDMx[2]** = 0 the transaction originates from one of the Cortex-A9 processors.
- **ARIDMx[2]** = 1 the transaction originates from the ACP.

Table 2-11 ARID encodings

	Transaction types		
	Cortex-A9 transactions		**ACP transactions**
ARIDMx[2]	**ARIDMx[2]** = 0		**ARIDMx[2]** = 1
ARIDMx[5:3]	Transaction type:		ACP read IDs
	`0b000`	Non-cacheable.	**ARIDMx[5:3]** = **ARIDS[2:0]**
	`0b010`	Data linefill buffer 0.	
	`0b011`	Data linefill buffer 1.	
	`0b100`	Instruction linefill.	
	`0b101`	Instruction linefill.	
	`0b110`	Instruction linefill.	
	`0b111`	Instruction linefill.	
ARIDMx[1:0]	Cortex-A9 processor:		Unused, forced to `0b00`.
	`0b00`	CPU0.	
	`0b01`	CPU1.	
	`0b10`	CPU2.	
	`0b11`	CPU3.	

AWIDMx[5:0] encodings

List of **AWIDMx[5:0]** encodings for write transactions.

As the following table shows, the **AWIDMx[2]** encodings distinguish between transactions originating from Cortex-A9 processors and transactions originating from the ACP:

- **AWIDMx[2]** = 0 the transaction originates from one of the Cortex-A9 processors.
- **AWIDMx[2]** = 1 the transaction originates from the ACP.

Table 2-12 AWIDMx encodings

	Transaction types		
	Cortex-A9 transactions		**ACP transactions**
AWIDMx[2]	**AWIDMx[2]** = 0		**AWIDMx[2]** = 1
AWIDMx[5:3]	`0b000`	Non-cacheable	ACP read IDs
	`0b010`	Eviction	**AWIDMx[5:3]** = **AWIDS[2:0]**
	`0b011`	Eviction	
	`0b100`	Eviction	
	`0b101`	Eviction	
AWIDMx[1:0]	`0b00`	CPU0.	Unused, forced to `0b00`.
	`0b01`	CPU1.	
	`0b10`	CPU2.	
	`0b11`	CPU3.	

2.3.4 AXI USER attributes encodings

List of implementation-specific AXI USER bit encodings on the master ports.

ARUSERMx[6:0] encodings

List of **ARUSERMx[6:0]** encodings for read transactions.

As the following table shows, the value and the meaning of the **ARUSERMx** encodings depend on the source of the transaction. There are transactions originating from Cortex-A9 processors and transactions originating from the ACP:

- **ARIDMx[2]** = 0 from one of the Cortex-A9 processors.
- **ARIDMx[2]** = 1 from the ACP.

Table 2-13 ARUSERMx[6:0] encodings

	Transaction types	
	Cortex-A9 transactions	**ACP transactions**
	ARIDMx[2] = 0	**ARIDMx[2] = 1**
ARUSERMx[6]	Speculative linefill to L2C-310	ACP USER bits
ARUSERMx[5]	Prefetch hint	**ARUSERMx[6:5]** = `0b00`
ARUSERMx[4:1]	Inner attributes	**ARUSERMx[4:1]** = **ARUSERSx[4:1]**
	`0b0000` Strongly Ordered	
	`0b0001` Device	
	`0b0011` Normal Memory NonCacheable	
	`0b0110` WriteThrough	
	`0b0111` Write Back no Write Allocate	
	`0b1111` Write Back Write Allocate	
ARUSERMx[0]	Shared bit	
	0 Non-coherent request.	
	1 Coherent request.	

AWUSERMx[8:0] encodings

List of **AWUSERMx[8:0]** encodings for write transactions.

As the following table shows, the value and the meaning of the **AWUSERMx** encodings depend on the source of the transaction:

- **AWIDMx[2]** = 0 from one of the Cortex-A9 processors.
- **AWIDMx[2]** = 1 from the ACP.

Table 2-14 AWUSERMx[8:0] encodings

	Transaction types	
	Cortex-A9 transactions	**ACP transactions**
	AWIDMx[2] = 0	**AWIDMx[2] = 1**
AWUSERMx[8]	Early **BRESP** enable	ACP USER bits
AWUSERMx[7]	Full line of write zeros indication	AWUSERMx[8:5] = `0b0000`
AWUSERMx[6]	Clean eviction information	
AWUSERMx[5]	L1 eviction information	

Table 2-14 AWUSERMx[8:0] encodings (continued)

	Transaction types	
	Cortex-A9 transactions	**ACP transactions**
	AWIDMx[2] = 0	**AWIDMx[2] = 1**
AWUSERMx[4:1]	Inner attributes: `0b0000` Strongly Ordered `0b0001` Device `0b0011` Normal Memory NonCacheable `0b0110` WriteThrough `0b0111` Write Back no Write Allocate `0b1111` Write Back Write Allocate	**AWUSERMx[4:0]** = **AWUSERS[4:0]**. Each master agent connected to the ACP can specify its own AXI USER signals. However, to maintain consistency, ARM recommends that the ACP AXI USER signal encodings match those of the Cortex-A9 processors.
AWUSERMx[0]	Shared bit: **0** Non-coherent request. **1** Coherent request.	

2.3.5 Address filtering capabilities

The SCU register bank contains dedicated registers to provide address filtering capabilities. The Filtering Start Address Register, the Filtering End Address Register, and the SCU Control Register are the dedicated registers.

On exit from reset, these registers sample the values present on the **FILTEREN**, **FILTERSTART**, and **FILTEREND** pins. Although the registers are writable, ARM strongly recommends that the software does not modify the values sampled on exit from reset.

When Address Filtering is enabled, SCU Control Register bit [1] = 1, any access that fits in the address range between the Filtering Start Address and the Filtering End Address is issued on the AXI Master port M1. All other accesses outside of this range are directed onto AXI Master port M0.This filtering rule is applied independently of the AXI request type and attributes. When Address Filtering is disabled, accesses can be issued indifferently on AXI Master port M0 or AXI Master port M1, if the AXI ordering rules are respected. However, in this case, locked and exclusive accesses are always issued on AXI Master port M0.

Related references

2.2.6 Filtering Start Address Register on page 2-31.
2.2.7 Filtering End Address Register on page 2-32.
2.2.2 SCU Control Register on page 2-27.

2.3.6 Device accesses filtering

In the r2p0 revision, the SCU also provides the ability to direct all device accesses onto the same AXI Master port, M0.

This feature can be used in systems where slow device traffic is expected. Directing all device traffic on the same AXI Master port M0 ensures that the other AXI Master port M1 remains available for other traffic types, cacheable traffic for example.

Note

The Address Filtering capabilities take precedence over the Force Device to AXI Master port M0 feature. That is, when address filtering is enabled, a device access falling in the Address Filtering range is issued onto AXI Master port M1 even if SCU Control Register bit[1] is set.

Related references

2.2.2 SCU Control Register on page 2-27.

2.3.7 AXI master interface clocking

Description of AXI bus ratios and AXI Timing diagram examples.

The Cortex-A9 MPCore Bus Interface Unit supports the following AXI bus ratios relative to **CLK**:

- Integer ratios through clock enable: 1:1, 2:1, 3:1, …
- Half-integer ratios through clock enable: 1.5, 2.5 and 3.5 ratios.

In all cases, AXI transfers remain synchronous. There is no requirement for an asynchronous AXI interface with integer and half integer ratios. The ratios are configured through external pins, with the following signals that qualify the input and output signals on AXI:

- **INCLKENM0** and **OUTCLKENM0**
- **INCLKENM1** and **OUTCLKENM1**.

Timing diagram for INCLKEN with three-to-two ratio

Figure showing a timing diagram example of read data return from an AXI slave back into the Cortex-A9 MPCore processor, with a three-to-two clock timing ratio.

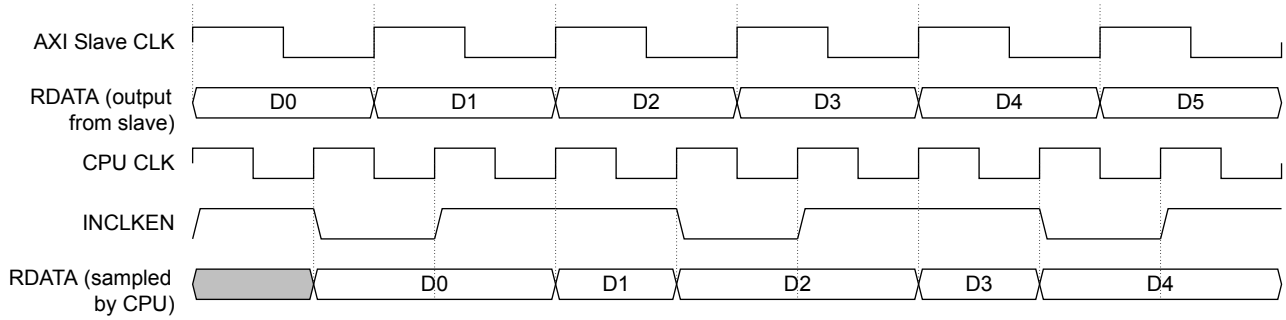

Figure 2-9 Timing diagram for INCLKEN with three-to-two clock ratio between CPU and AXI Slave CLK

Timing diagram for INCLKEN with five-to-two ratio

Figure showing a timing diagram example of read data return from an AXI slave back into the Cortex-A9 MPCore processor, with a five-to-two clock timing ratio.

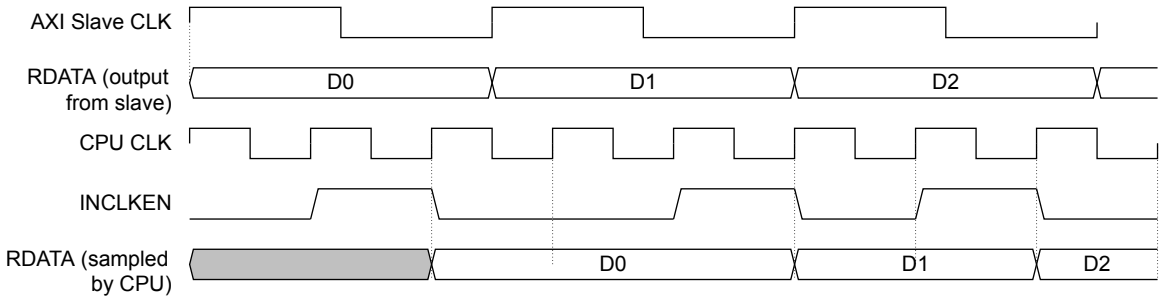

Figure 2-10 Timing diagram for INCLKEN with five-to-two clock ratio between CPU and AXI Slave CLK

Timing diagram for OUTCLKEN with three-to-two ratio

Figure showing a timing diagram example of data write from the Cortex-A9 MPCore processor into an AXI slave, with a three-to-two clock timing ratio.

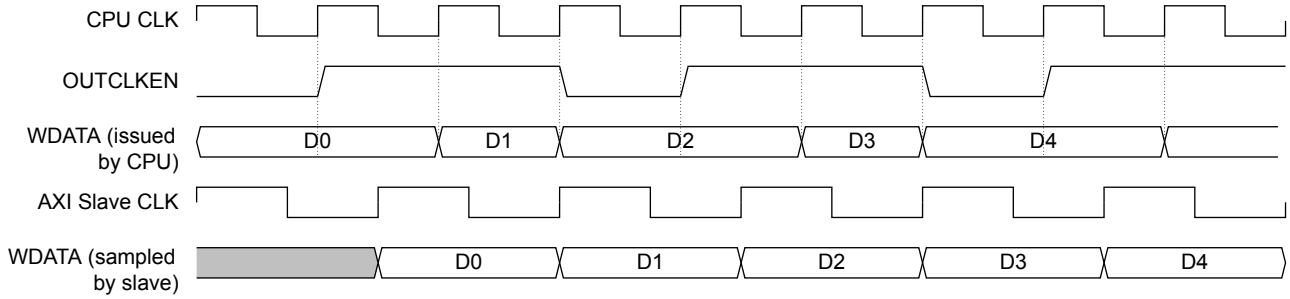

Figure 2-11 Timing diagram for OUTCLKEN with three-to-two clock ratio between CPU and AXI Slave CLK

Timing diagram for OUTCLKEN with five-to-two ratio

Figure showing a timing diagram example of data write from the Cortex-A9 MPCore processor into an AXI slave, with a five-to-two clock timing ratio.

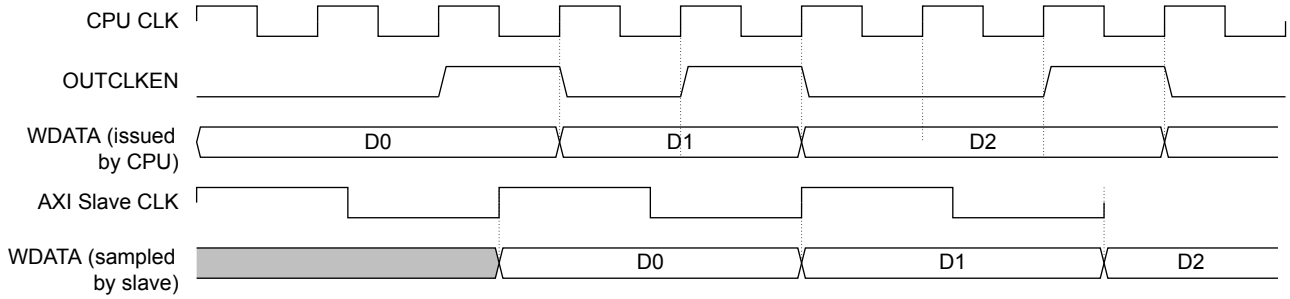

Figure 2-12 Timing diagram for OUTCLKEN with five-to-two clock ratio between CPU and AXI Slave CLK

2.4 Accelerator Coherency Port

The *Accelerator Coherency Port* (ACP) is an optional AXI 64-bit slave port that can be connected to non-cached AXI master peripherals, such as a DMA engine or cryptographic engine.

This AMBA 3 AXI compatible slave interface on the SCU provides an interconnect point for a range of system masters that for overall system performance, power consumption, or reasons of software simplification, are better interfaced directly with the Cortex-A9 MPCore processor.

This section contains the following subsections:

2.4.1 ACP requests

The read and write requests performed on the ACP behave differently depending on whether the request is coherent or not.

ACP requests behavior is as follows:

ACP coherent read requests

An ACP read request is coherent when **ARUSER[0]** = 1 and **ARCACHE[1]** = 1 alongside **ARVALID**.

In this case, the SCU enforces coherency.

When the data is present in one of the Cortex-A9 processors within the Cortex-A9MPCore, the data is read directly from the relevant processor, and returned to the ACP port.

When the data is not present in any of the Cortex-A9 processors, the read request is issued on one of the Cortex-A9 MPCore AXI master ports, along with all its AXI parameters, except for the locked attribute.

ACP non-coherent read requests

An ACP read request is non-coherent when **ARUSER[0]** = 0 or **ARCACHE[1]** = 0 alongside **ARVALID**.

In this case, the SCU does not enforce coherency, and the read request is directly forwarded to one of the available Cortex-A9 MPCore AXI master ports.

ACP coherent write requests

An ACP write request is coherent when **AWUSER[0]** = 1 and **AWCACHE[1]** = 1 alongside **AWVALID**.

In this case, the SCU enforces coherency.

When the data is present in one of the Cortex-A9 processors within the Cortex-A9 MPCore, the data is first cleaned and invalidated from the relevant CPU.

When the data is not present in any of the Cortex-A9 processors, or when it has been cleaned and invalidated, the write request is issued on one of the Cortex-A9 MPCore AXI master ports, along with all corresponding AXI parameters except for the locked attribute.

——— **Note** ———

The transaction can optionally allocate into the L2 cache if the write parameters are set accordingly.

———————————

ACP non-coherent write requests

An ACP write request is non-coherent when **AWUSER[0]** = 0 or **AWCACHE[1]** = 0 alongside **AWVALID**.

In this case, the SCU does not enforce coherency, and the write request is forwarded directly to one of the available Cortex-A9 MPCore AXI master ports.

2.4.2 ACP interface clocking

Unlike the AXI Master port interfaces, the ACP port does not support half clock ratio between the AXI clock and the SCU clock. Only integer clock ratios are supported, with the use of a single **ACLKENS** signal.

The following figure shows a timing example where **ACLKENS** is used with a 3:1 clock ratio between **CLK** and the ACP AXI clock, **ACLK**.

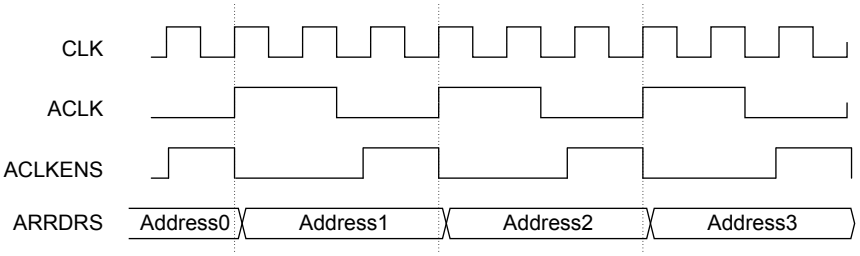

Figure 2-13 ACLKENS timing example

The ACP slave port samples the AXI input requests, and the AXI output values, only on the rising edge of **CLK** when **ACLKENS** is HIGH.

2.4.3 ACP limitations

The ACP is optimized for cache-line length transfers and it supports a wide range of AMBA 3 AXI requests, but it has some limitations that must be considered. These limitations are the ACP performance and ACP functional limitations.

ACP performance limitations

ACP accesses are optimized for transfers that match Cortex-A9 processors coherent requests.

For example:

- A wrapped burst of four doublewords (length = 3, size = 3), with a 64-bit aligned address, and all byte strobes set.
- An incremental burst of four doublewords, with the first address corresponding to the start of a cache line, and all byte strobes set.

For maximum performance use ACP accesses that match this optimized format. ACP accesses that do not match this format cannot benefit from the SCU optimizations, and have significantly lower performance.

ACP functional limitations

The ACP is a full AMBA 3 AXI slave component, with the exception of three types of transfers that are not supported.

The transfers that are not supported are the following:

- Exclusive read and write transactions to coherent memory.
- All locked transactions, except SWP style transactions to non-coherent memory regions. An SWP style transaction is a locked read access to non-coherent memory, followed by a write access to the same address, and with the same attributes.
- Optimized coherent read and write transfers when byte strobes are not all set.

As a consequence, it is not possible to use the LDREX/STREX mechanism through the ACP to gain exclusive access to coherent memory regions, that are marked with **AxUSER[0]** = 1 and **AxCACHE[1]** = 1.

However, the LDREX/STREX mechanism is fully supported through the ACP for non-coherent memory regions, marked with **AxUSER[0]** = 0 or **AxCACHE[1]** = 0.

Related references

2.4.2 ACP interface clocking on page 2-44.

2.5 Event communication with an external agent using WFE/SEV

A peripheral connected on the coherency port or any other external agent can participate in the WFE/SEV event communication of the Cortex-A9 MPCore processor by using the **EVENTI** pin.

When this pin is asserted, it sends an event message to all the Cortex-A9 processors in the cluster. This is similar to executing a SEV instruction on one processor of 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 power-saving mode. The **EVENTI** input pin must remain high at least one **CPUCLK** clock cycle to be visible by the processors.

The external agent can see that at least one of the Cortex-A9 processors in the cluster has executed an SEV instruction by checking the **EVENTO** pin. This pin is set high for one **CLK** clock cycle when any of the Cortex-A9 processors in the cluster executes an SEV instruction.

Chapter 3
Interrupt Controller

This chapter describes the implementation-defined features of the Interrupt Controller.

Interrupt Controller is compliant with the *ARM® Generic Interrupt Controller Architecture Specification 1.0*.

This chapter does not reproduce information already in the *ARM® Generic Interrupt Controller Architecture Specification*.

It contains the following sections:

3.1 About the Interrupt Controller

The Interrupt Controller is a single functional unit that is located in a Cortex-A9 MPCore design. It is responsible for centralizing all interrupt sources before dispatching them to each individual Cortex-A9 processor. There is one interrupt interface per Cortex-A9 processor.

The Interrupt Controller is memory-mapped. The Cortex-A9 processors access it by using a private interface through the SCU.

This section contains the following subsections:

3.1.1 Interrupt Controller Clock frequency

The clock period is configured, during integration, as a multiple of the MPCore clock period. This multiple, N, must be greater than or equal to two. As a consequence, the minimum pulse width of signals driving external interrupt lines is N Cortex-A9 processor clock cycles.

See *Chapter 5 Clocks, Resets, and Power Management* on page 5-75 for a description of **PERIPHCLK** and **PERIPHCLKEN**.

The timers and watchdogs use the same clock as the interrupt controller.

3.1.2 Interrupt Distributor interrupt sources

Interrupt sources for the Interrupt Distributor can be of several types.

The types are the following:

Software Generated Interrupts (SGI)
Each Cortex-A9 processor has private interrupts, ID0-ID15, that can only be triggered by software. These interrupts are aliased so that there is no requirement for a requesting Cortex-A9 processor to determine its own CPU ID when it deals with SGIs. The priority of an SGI depends on the value set by the receiving Cortex-A9 processor in the banked SGI priority registers, not the priority set by the sending Cortex-A9 processor.

Global timer, PPI(0)
The global timer uses ID27.

A legacy nFIQ pin, PPI(1)
In legacy FIQ mode the legacy **nFIQ** pin, on a per Cortex-A9 processor basis, bypasses the interrupt distributor logic and directly drives interrupt requests into the Cortex-A9 processor.

When a Cortex-A9 processor uses the Interrupt Controller, rather than the legacy pin in the legacy mode, by enabling its own Cortex-A9 processor interface, the legacy **nFIQ** pin is treated like other interrupt lines and uses ID28.

Private timer, PPI(2)
Each Cortex-A9 processor has its own private timers that can generate interrupts, using ID29.

Watchdog timers, PPI(3)
Each Cortex-A9 processor has its own watchdog timers that can generate interrupts, using ID30.

A legacy nIRQ pin, PPI(4)
In legacy IRQ mode the legacy **nIRQ** pin, on a per Cortex-A9 processor basis, bypasses the interrupt distributor logic and directly drives interrupt requests into the Cortex-A9 processor.

When a Cortex-A9 processor uses the Interrupt Controller, rather than the legacy pin in the legacy mode, by enabling its own Cortex-A9 processor interface, the legacy **nIRQ** pin is treated like other interrupt lines and uses ID31.

Shared Peripheral Interrupts (SPI)

SPIs are triggered by events generated on associated interrupt input lines. The Interrupt Controller can support up to 224 interrupt input lines. The interrupt input lines can be configured to be edge sensitive (positive edge) or level sensitive (high level). SPIs start at ID32.

A unique ID identifies interrupt sources, except the SGIs that are aliased and identified by CPU source.

All interrupt sources have their own configurable priority.

All interrupt sources, except the SGIs and PPIs, also have their own configurable CPU target list, that is, a list of Cortex-A9 processors where the interrupt is sent when triggered by the Interrupt Distributor.

3.1.3 Interrupt Distributor arbitration

The interrupt distributor centralizes all interrupt sources before dispatching them to each individual Cortex-A9 processor.

The Interrupt Distributor arbitrates in the following priority order:

1. Highest priority interrupts. These have the lowest value in the *Interrupt Priority Register* (ICDIPTR).
2. For interrupts with the same priority value, the Interrupt Distributor arbitrates on the interrupt ID number. It dispatches the smaller ID number first
3. For aliased SGI with the same priority value and the same ID number, the Interrupt Distributor arbitrates on the source CPU number. It dispatches the smaller CPU number first.

The Interrupt Controller hardware ensures that an interrupt targeted at several Cortex-A9 processors can only be taken by one Cortex-A9 processor at a time.

3.1.4 Cortex-A9 MPCore 1-N interrupt model handling

In systems with two or more processors, if an interrupt is received by more than one processor, the Cortex-A9 MPCore ensures that only one of the processors reads the corresponding interrupt ID. This removes the requirement for a lock on the Interrupt Service Routine.

When accessing the ICCIAR register, other processors then read the spurious ID, or another pending ID.

Related references

1.5 Private Memory Region on page 1-17.

3.2 Security extensions support

The Interrupt Controller permits all implemented interrupts to be individually defined as Secure or Non-secure.

You can program Secure interrupts to use either the IRQ or FIQ interrupt mechanism of a Cortex-A9 processor through the FIQen bit in the ICPICR Register. Non-secure interrupts are always signaled using the IRQ mechanism of a Cortex-A9 processor.

This section contains the following subsections:

3.2.1 Priority formats

The Cortex-A9 processor implements a five-bit version of the priority format. In Non-secure state, only 4 bits of the priority format are visible.

See the *ARM® Generic Interrupt Controller Architecture Specification*.

3.2.2 Using CFGSDISABLE

The Interrupt Controller provides the facility to prevent write accesses to critical configuration registers when you assert **CFGSDISABLE**. This signal controls write behavior for the secure control registers in the distributor and Cortex-A9 processor interfaces, and the *Lockable Shared Peripheral Interrupts* (LSPIs) in the Interrupt Controller.

If you use **CFGSDISABLE**, ARM recommends that you assert **CFGSDISABLE** during the system boot process, after the software has configured the registers. Ideally, the system must only deassert **CFGSDISABLE** if a hard reset occurs.

When **CFGSDISABLE** is HIGH, the Interrupt Controller prevents write accesses to the following registers in the:

Distributor

The Secure enable of the ICDDCR.

Secure interrupts defined by LSPI field in the ICDICTR:

- Interrupt Security Registers
- Interrupt Set-Enable Registers
- Interrupt Clear-Enable Registers
- Interrupt Set-Pending Registers
- Interrupt Clear-Pending Registers
- Interrupt Priority Registers
- ICDIPTR
- Interrupt Configuration Register.

Cortex-A9 interrupt interfaces

The ICCICR, except for the EnableNS bit.

After you assert **CFGSDISABLE**, it changes the register bits to read-only and therefore the behavior of these secure interrupts cannot change, even in the presence of rogue code executing in the secure domain.

3.3 Distributor register descriptions

Summary of registers that the distributor provides with information on purpose, usage contraints, configurations, and attributes for each register.

This section contains the following subsections:

3.3.1 Distributor register summary

List of distributor registers.

Registers not described in the following table are RAZ/WI.

This section does not reproduce information about registers already described in the *ARM Generic Interrupt Controller Architecture Specification 1.0*.

The ICDIPR and ICDIPTR registers are byte accessible and word accessible. All other registers in the following table are word accessible.

See *1.5 Private Memory Region* on page 1-17 for the offset of this page from **PERIPHBASE[31:13]**.

Table 3-1 Distributor register summary

Base	Name	Type	Reset	Width	Function
0x000	ICDDCR	RW	0x00000000	32	*3.3.2 Distributor Control Register* on page 3-52
0x004	ICDICTR	RO	Configuration dependent	32	*3.3.3 Interrupt Controller Type Register* on page 3-53
0x008	ICDIIDR	RO	0x0102043B	32	*3.3.4 Distributor Implementer Identification Register* on page 3-55
0x00C - 0x07C	-	-	-	-	Reserved
0x080 - 0x09C	ICDISRn	RW [f]	0x00000000	32	Interrupt Security Registers
0x100	ICDISERn	RW	0x0000FFFF	32	Interrupt Set-Enable Registers
0x104 - 0x11C			0x00000000		
0x180	ICDICERn	RW	0x0000FFFF	32	Interrupt Clear-Enable Registers
0x184 - 0x19C			0x00000000		
0x200 - 0x27C	ICDISPRn	RW	0x00000000	32	Interrupt Set-Pending Registers
0x280 - 0x29C	ICDICPRn	RW	0x00000000	32	Interrupt Clear-Pending Registers
0x300 - 0x31C	ICDABRn	RO	0x00000000	32	Active Bit registers
0x380 - 0x3FC	-	-	-	-	Reserved
0x400 - 0x4FC	ICDIPRn	RW	0x00000000	32	Interrupt Priority Registers

[f] You must access this register in Secure state.

Table 3-1 Distributor register summary (continued)

Base	Name	Type	Reset	Width	Function
0x7FC	-	-	-	-	Reserved
0x800 - 0x8FC	ICDIPTRn	RW	0x0000000	32	*3.3.7 Interrupt Processor Targets Registers* on page 3-56
0xBFC	-	-	-	-	Reserved
0xC00	ICDICFRn	RW	0xAAAAAAAA	32	*3.3.8 Interrupt Configuration Registers* on page 3-56
0xC04			0x7DC00000		
0xC08- 0xC3C			0x55555555 [g]		
0xD00	ICPPISR	-	0x00000000	32	*3.3.9 PPI Status Register* on page 3-56
0xD04- 0xD1C	ICSPISRn	RO	0x00000000	32	*3.3.10 SPI Status Registers* on page 3-57
0xD80- 0xEFC	-	-	-	-	Reserved
0xF00	ICDSGIR	WO	-	32	Software Generated Interrupt Register
0xF04 - 0xFCC	-	-	-	-	Reserved
0xFD0	ICPIDR0	RO	0x4	8	Peripheral ID0 register
0xFD4	ICPIDR1	RO	0x0	8	Peripheral ID1 register
0xFD8	ICPIDR2	RO	0x0	8	Peripheral ID2 register
0xFDC	ICPIDR3	RO	0x0	8	Peripheral ID3 register
0xFE0	ICPIDR4	RO	0x90	8	Peripheral ID4 register
0xFE4	ICPIDR5	RO	0xB3	8	Peripheral ID5 register
0xFE8	ICPIDR6	RO	0x1B	8	Peripheral ID6 register
0xFEC	ICPIDR7	RO	0x0	8	Peripheral ID7 register
0xFF0	ICCIDR0	RO	0xD	8	Component ID0 register
0xFF4	ICCIDR1	RO	0xF0	8	Component ID1 register
0xFF8	ICCIDR2	RO	0x5	8	Component ID2 register
0xFFC	ICCIDR3	RO	0xB1	8	Component ID3 register

3.3.2 Distributor Control Register

Characteristics and bit assignments of the ICDDCR.

Purpose Controls whether the distributor responds to external stimulus changes that occur on **SPI**s and **PPI**s.

Usage constraints This register is banked. The register you access depends on the type of access:

Secure access
Distributor provides access to the Secure Enable and Non-secure Enable bits.

Non-secure access
Distributor provides access to the Non-secure enable bit only.

You cannot modify Enable Secure if **CFGSDISABLE** is set. You can modify Enable Non-secure even if **CFGSDISABLE** is set, through the S or the NS register.

Configurations Available in all Cortex-A9 multiprocessor configurations.

[g] Reset value is 0x55555555 when the corresponding interrupts are present, else 0x00000000

Attributes See the register summary in *3.3.1 Distributor register summary* on page 3-51.

The following figure shows the ICDDCR bit assignments for Secure accesses.

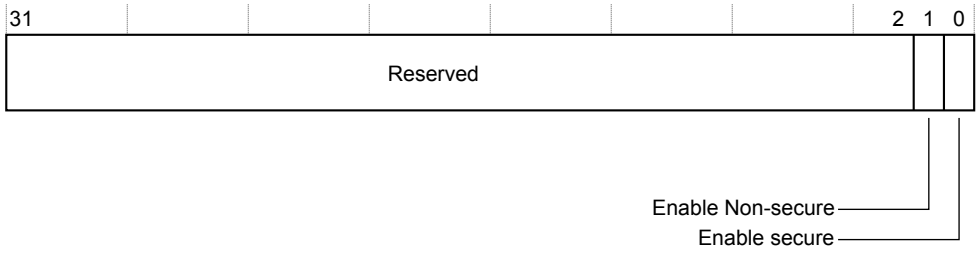

Figure 3-1 ICDDCR bit assignments for Secure accesses

The following table shows the ICDDCR bit assignments for secure accesses.

Table 3-2 ICDDCR bit assignments for secure accesses

Bits	Name	Function
[31:2]	-	Reserved
[1]	Enable Non-secure	**0** Disables all Non-secure interrupt control bits in the distributor from changing state because of any external stimulus change that occurs on the corresponding **SPI** or **PPI** signals. **1** Enables the distributor to update register locations for Non-secure interrupts.
[0]	Enable secure	**0** Disables all Secure interrupt control bits in the distributor from changing state because of any external stimulus change that occurs on the corresponding **SPI** or **PPI** signals. **1** Enables the distributor to update register locations for Secure interrupts.

The following figure shows the ICDDCR bit assignments for Non-secure accesses.

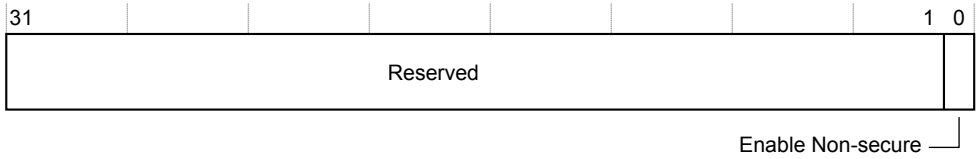

Figure 3-2 ICDDCR bit assignments for Non-secure accesses

The following table shows the ICDDCR bit assignments for Non-secure accesses.

Table 3-3 ICDDCR bit assignments for Non-secure accesses

Bits	Name	Function
[31:1]	-	Reserved
[0]	Enable Non-secure	**0** Disables all Non-secure interrupts control bits in the distributor from changing state because of any external stimulus change that occurs on the corresponding **SPI** or **PPI** signals. **1** Enables the distributor to update register locations for Non-secure interrupts.

3.3.3 Interrupt Controller Type Register

Characteristics and bit assignments of the ICDICTR.

Purpose Provides information about the configuration of the Interrupt Controller.

Usage constraints There are no usage constraints.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *3.3.1 Distributor register summary* on page 3-51.

The following figure shows the ICDICTR bit assignments.

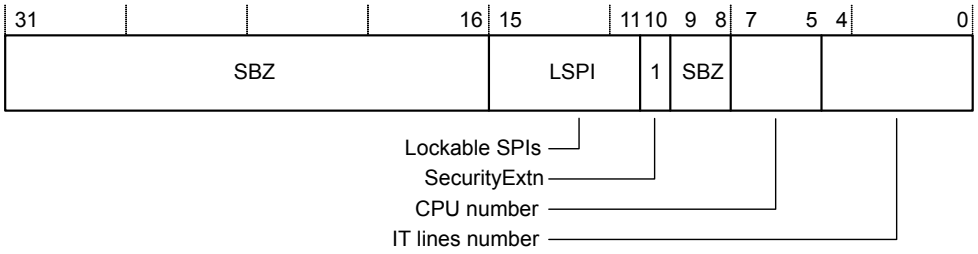

Figure 3-3 ICDICTR bit assignments

The following table shows the ICDICTR bit assignments.

Table 3-4 ICDICTR bit assignments

Bits	Name	Function
[31:16]	-	Reserved
[15:11]	LSPI	Returns the number of *Lockable Shared Peripheral Interrupts* (LSPIs) that the controller contains. The encoding is: `0b11111` 31 LSPIs, that are the interrupts of IDs 32-62. When **CFGSDISABLE** is HIGH, the interrupt controller prevents writes to any register location that controls the operating state of an LSPI.
[10]	SecurityExtn	Returns the number of security domains that the controller contains: **1** The controller contains two security domains. This bit always returns the value one.
[9:8]	-	Reserved

Table 3-4 ICDICTR bit assignments (continued)

Bits	Name	Function
[7:5]	CPU number	The encoding is: `0b000` The Cortex-A9 MPCore configuration contains one Cortex-A9 processor. `0b001` The Cortex-A9 MPCore configuration contains two Cortex-A9 processors. `0b010` The Cortex-A9 MPCore configuration contains three Cortex-A9 processors. `0b011` The Cortex-A9 MPCore configuration contains four Cortex-A9 processors. `0b1xx` Unused values.
[4:0]	IT lines number	The encoding is: `0b00000` The distributor provides 32 interrupts, no external interrupt lines. [h] `0b00001` The distributor provides 64 interrupts, 32 external interrupt lines. `0b00010` The distributor provides 96 interrupts, 64 external interrupt lines. `0b00011` The distributor provides 128 interrupts, 96 external interrupt lines. `0b00100` The distributor provides 160 interrupts, 128 external interrupt lines. `0b00101` The distributor provides 192 interrupts, 160 external interrupt lines. `0b00110` The distributor provides 224 interrupts, 192 external interrupt lines. `0b00111` The distributor provides 256 interrupts, 224 external interrupt lines. All other values not used.

3.3.4 Distributor Implementer Identification Register

Characteristics and bit assignments of the ICDIIDR.

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

Usage constraints There are no usage constraints.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *3.3.1 Distributor register summary* on page 3-51.

The following figure shows the ICDIIDR bit assignments.

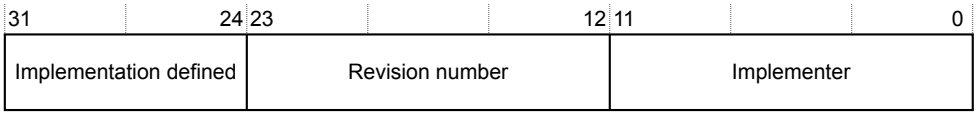

Figure 3-4 ICDIIDR bit assignments

The following table shows the ICDIIDR bit assignments.

Table 3-5 ICDIIDR bit assignments

Bits	Values	Name	Function
[31:24]	`0x01`	Implementation version	Gives implementation version number.
[23:12]	`0x020`	Revision number	Returns the revision number of the controller.
[11:0]	`0x43B`	Implementer	Implementer number.

[h] The distributor always uses interrupts of IDs 0 to 31 to control any SGIs and PPIs that the Interrupt Controller might contain.

3.3.5 Interrupt Set-Enable Registers

Summary of the implementation defined features of the ICDISERn.

In the Cortex-A9 MPCore, SGIs are always enabled. The corresponding bits in the ICDISERn are read as one, write ignored.

3.3.6 Interrupt Clear-Enable Registers

Summary of the implementation defined features of the ICDICERn.

In the Cortex-A9 MPCore, SGIs are always enabled. The corresponding bits in the ICDICERn are read as one, write ignored.

3.3.7 Interrupt Processor Targets Registers

Summary of the implementation defined features of the ICDIPTRn.

For systems that support only one Cortex-A9 processor, all these registers read as zero, and writes are ignored. The single Cortex-A9 processor is always set as the target of any interruption.

For systems that support two or more Cortex-A9 processors, if the Processor Target field is set to 0 for a specific SPI, then this interrupt cannot be set pending through the hardware pins, nor by a write to the Set-Pending register.

3.3.8 Interrupt Configuration Registers

Summary of the implementation defined features of the ICDICFR.

Each bit-pair describes the interrupt configuration for an interrupt. The options for each pair depend on the interrupt type as follows:

SGI The bits are read-only and a bit-pair always reads as `0b10`.

PPI The bits are read-only

PPI[1] and [4]:0b01
Interrupt is active LOW level sensitive.

PPI[0], [2],and[3]:0b11
Interrupt is rising-edge sensitive.

SPI The LSB bit of a bit-pair is read-only and is always `0b1`. You can program the MSB bit of the bit-pair to alter the triggering sensitivity as follows:

0b01 Interrupt is active HIGH level sensitive

0b11 Interrupt is rising-edge sensitive.

There are 31 LSPIs, interrupts 32-62. You can configure and then lock these interrupts against more change using **CFGSDISABLE**. The LSPIs are present only if the SPIs are present.

3.3.9 PPI Status Register

Characteristics and bit assignments of the ICPPISR.

Purpose Enables a Cortex-A9 processor to access the status of the inputs on the distributor:
- PPI(4) is for **nIRQ<n>**
- PPI(3) is for watchdog interrupts
- PPI(2) is for private timer interrupts
- PPI(1) is for nFIQ<n>
- PPI(0) is for the global timer.

Usage constraints A Cortex-A9 processor can only read the status of its own **PPI** and therefore cannot read the status of **PPI** for other Cortex-A9 processors.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *3.3.1 Distributor register summary* on page 3-51.

The following figure shows the ICPPISR bit assignments.

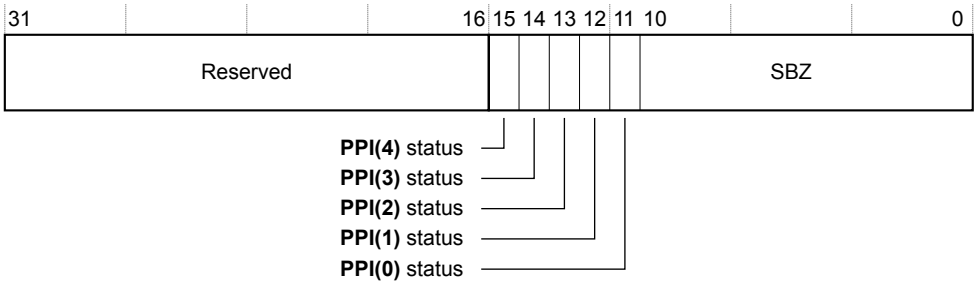

Figure 3-5 ICPPISR bit assignments

The following table shows the ICPPISR bit assignments.

Table 3-6 ICPPISR bit assignments

Bits	Name	Function
[31:16]	-	Reserved
[15:11]	ppi_status	Returns the status of the **PPI(4:0)** inputs on the distributor: • PPI[4] is nIRQ • PPI[3] is the private watchdog • PPI[2] is the private timer • PPI[1] is nFIQ • PPI[0] is the global timer. PPI[1] and PPI[4] are active LOW PPI[0], PPI[2], and PPI[3] are active HIGH. **Note** These bits return the actual status of the **PPI(4:0)** signals. The ICDISPRn and ICDICPRn registers can also provide the **PPI(4:0)** status but because you can write to these registers then they might not contain the actual status of the **PPI(4:0)** signals.
[10:0]	-	SBZ

3.3.10 SPI Status Registers

Characteristics and bit assignments of the ICSPISRn.

Purpose Enables a Cortex-A9 processor to access the status of **IRQS[N:0]** inputs on the distributor.

Usage constraints There are no usage constraints.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *3.3.1 Distributor register summary* on page 3-51.

The following figure shows the ICSPISRn bit assignments.

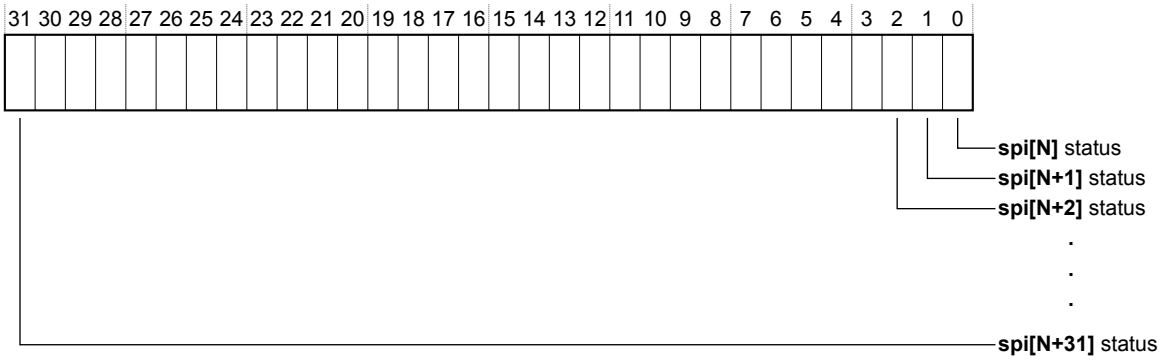

Figure 3-6 ICSPISRn bit assignments

The following table shows the ICSPISRn bit assignments.

Table 3-7 ICSPISRn bit assignments

Bits	Name	Function
[31:0]	spi_status	Returns the status of the **IRQS[N:0]** inputs on the distributor: **Bit [X] = 0** **IRQS[X]** is LOW **Bit [X] = 1** **IRQS[X]** is HIGH. **Note** • The **IRQS** that X refers to depends on its bit position and the base address offset of the spi_status Register as the following figure shows. • These bits return the actual status of the **IRQS** signals. The pending_set and pending_clr Registers can also provide the **IRQS** status but because you can write to these registers then they might not contain the actual status of the **IRQS** signals.

The following figure shows the address map that the distributor provides for the SPIs.

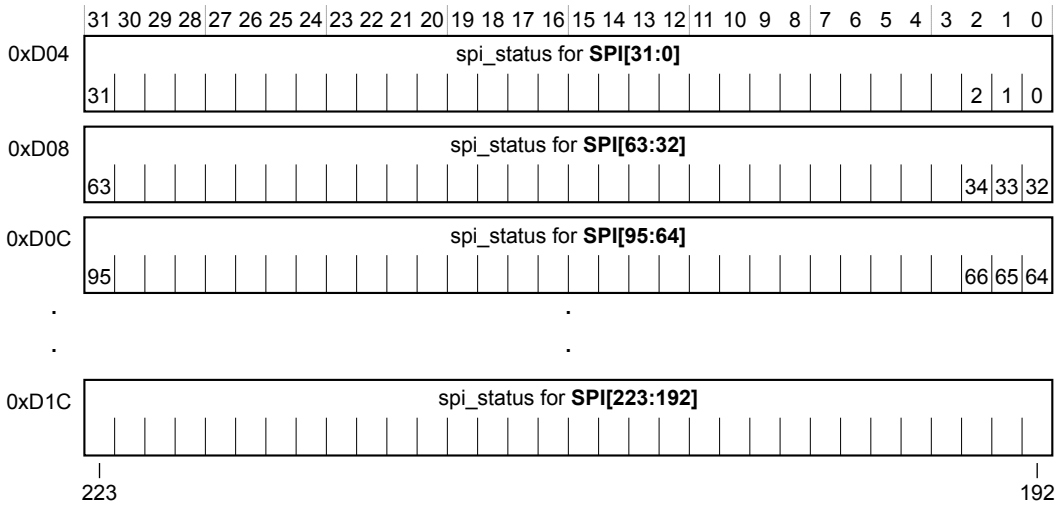

Figure 3-7 ICSPISRn address map

In this figure, the values for the SPIs are read-only. This register contains the values for the SPIs for the corresponding Cortex-A9 processor interface. The distributor provides up to seven registers. If you

configure the Interrupt Controller to use fewer than 224 SPIs, then it reduces the number of registers accordingly. For locations where interrupts are not implemented then the distributor:

- Ignores writes to the corresponding bits.
- Returns 0 when it reads from these bits.

3.4 Interrupt interface register descriptions

List of registers that each Cortex-A9 processor interface provides with information on purpose, usage contraints, configurations, and attributes for each register.

This section contains the following subsections:

3.4.1 Processor interface register summary

List of Cortex-A9 processor interface registers.

This section does not reproduce information about registers already described in the *ARM® Generic Interrupt Controller Architecture Specification*.

Table 3-8 Cortex-A9 processor interface register summary

Base	Name	Type	Reset	Width	Function
0x000	ICCICR	RW	0x00000000	32	CPU Interface Control Register
0x004	ICCPMR	RW	0x00000000	32	Interrupt Priority Mask Register
0x008	ICCBPR	RW	0x2 0x3	32	Binary Point Register
0x00C	ICCIAR	RO	0x000003FF	32	Interrupt Acknowledge Register
0x010	ICCEOIR	WO	-	32	End Of Interrupt Register
0x014	ICCRPR	RO	0x000000FF	32	Running Priority Register
0x018	ICCHPIR	RO	0x000003FF	32	Highest Pending Interrupt Register
0x01C	ICCABPR	RW	0x3	32	Aliased Non-secure Binary Point Register
0xFC	ICCIDR	RO	0x3901243B	32	*3.4.2 CPU Interface Implementer Identification Register* on page 3-60

3.4.2 CPU Interface Implementer Identification Register

Characteristics and bit assignments for the ICCIIDR Register.

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

Usage constraints There are no usage constraints.

Configurations Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in *3.4.1 Processor interface register summary* on page 3-60.

The following figure shows the ICCIIDR bit assignments.

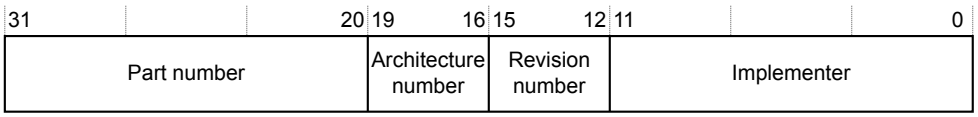

Figure 3-8 ICCIIDR bit assignments

The following table shows the ICCIIDR bit assignments.

Table 3-9 ICCIIDR bit assignments

Bits	Values	Name	Function
[31:20]	`0x390`	Part number	Identifies the peripheral.
[19:16]	0x1	Architecture version	Identifies the architecture version.
[15:12]	0x2	Revision number	Returns the revision number of the Interrupt Controller. The implementer defines the format of this field.
[11:0]	0x43B	Implementer	Returns the JEP106 code of the company that implemented the Cortex-A9 processor interface RTL. It uses the following construct: **[11:8]** The JEP106 continuation code of the implementer. **[7]** 0. **[6:0]** The JEP106 code [6:0] of the implementer.

Chapter 4
Global timer, private timers, and watchdog registers

This chapter describes the timers and watchdog registers.

It contains the following sections:

4.1 About the private timer and watchdog blocks

List of features of the private timer and watchdog blocks.

- A 32-bit counter that generates an interrupt when it reaches zero.
- An eight-bit prescaler value to qualify the clock period.
- Configurable single-shot or auto-reload modes.
- Configurable starting values for the counter.
- The clock for these blocks is PERIPHCLK.

The watchdog can be configured as a timer.

See *Chapter 5 Clocks, Resets, and Power Management* on page 5-75 for a description of **CLK**, **PERIPHCLK**, and **PERIPHCLKEN**.

This section contains the following subsections:

- *4.1.1 Calculating timer intervals* on page 4-63.
- *4.1.2 Security extensions* on page 4-63.

4.1.1 Calculating timer intervals

The timer interval is calculated using the following equation.

$$\left(\frac{(\text{PRESCALER\_value}+1) \times (\text{Load\_value}+1)}{\text{PERIPHCLK}}\right)$$

Figure 4-1 Equation for timer intervals

This equation can be used to calculate the period between two events generated by a timer or watchdog.

4.1.2 Security extensions

The SCU Non-secure Access Control Register section describes how to use timers in Secure or Non-secure state.

Related references

2.2.9 SCU Non-secure Access Control Register on page 2-34.

4.2 Private timer and watchdog registers

Summary of timer and watchdog registers with information on bit assignments for each register.

Addresses are relative to the base address of the timer and watchdog region defined by the private memory map.

All timer and watchdog registers are word-accessible only.

Use **nPERIPHRESET** to reset these registers, except the Watchdog Reset Status Register.

nWDRESET resets the Watchdog Reset Status Register.

This section contains the following subsections:

4.2.1 Private timer and watchdog register summary

List of timer and watchdog registers.

All registers not described in the following table are Reserved.

Table 4-1 Timer and watchdog registers

Offset	Type	Reset Value	Function
0x00	RW	0x00000000	*4.2.2 Private Timer Load Register* on page 4-65
0x04	RW	0x00000000	*4.2.3 Private Timer Counter Register* on page 4-65
0x08	RW	0x00000000	*4.2.4 Private Timer Control Register* on page 4-65
0x0C	RW	0x00000000	*4.2.5 Private Timer Interrupt Status Register* on page 4-66
0x20	RW	0x00000000	*4.2.6 Watchdog Load Register* on page 4-66
0x24	RW	0x00000000	*4.2.7 Watchdog Counter Register* on page 4-66
0x28	RW	0x00000000	*4.2.8 Watchdog Control Register* on page 4-67
0x2C	RW	0x00000000	*4.2.9 Watchdog Interrupt Status Register* on page 4-68
0x30	RW	0x00000000	*4.2.10 Watchdog Reset Status Register* on page 4-69
0x34	WO	-	*4.2.11 Watchdog Disable Register* on page 4-69

Note

The private timers stop counting when the associated processor is in debug state.

4.2.2 Private Timer Load Register

The Timer Load Register contains the value copied to the Timer Counter Register when it decrements down to zero with auto reload mode enabled. Writing to the Timer Load Register means that you also write to the Timer Counter Register.

4.2.3 Private Timer Counter Register

The Timer Counter Register is a decrementing counter.

The Timer Counter Register decrements if the timer is enabled using the timer enable bit in the Timer Control Register. If a Cortex-A9 processor timer is in debug state, the counter only decrements when the Cortex-A9 processor returns to non-debug state.

When the Timer Counter Register reaches zero and auto reload mode is enabled, it reloads the value in the Timer Load Register and then decrements from that value. If auto reload mode is not enabled, the Timer Counter Register decrements down to zero and stops.

When the Timer Counter Register reaches zero, the timer interrupt status event flag is set and the interrupt ID 29 is set as pending in the Interrupt Distributor, if interrupt generation is enabled in the Timer Control Register.

Writing to the Timer Counter Register or Timer Load Register forces the Timer Counter Register to decrement from the newly written value.

4.2.4 Private Timer Control Register

Bit assignments for the Private Timer Control Register.

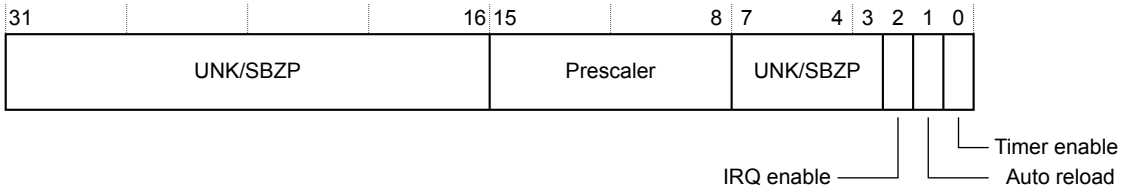

Figure 4-2 Private Timer Control Register bit assignments

The following table shows the Private Timer Control Register bit assignments.

Table 4-2 Private Timer Control Register bit assignments

Bits	Name	Function
[31:16]	-	UNK/SBZP.
[15:8]	Prescaler	The prescaler modifies the clock period for the decrementing event for the Counter Register. See *4.1.1 Calculating timer intervals* on page 4-63 for the equation.
[7:3]	-	UNK/SBZP.
[2]	IRQ Enable	If set, the interrupt ID 29 is set as pending in the Interrupt Distributor when the event flag is set in the Timer Status Register.

Table 4-2 Private Timer Control Register bit assignments (continued)

Bits	Name	Function
[1]	Auto reload	**0** Single shot mode.Counter decrements down to zero, sets the event flag and stops. **1** Auto-reload mode. Each time the Counter Register reaches zero, it is reloaded with the value contained in the Timer Load Register.
[0]	Timer Enable	Timer enable: **0** Timer is disabled and the counter does not decrement. All registers can still be read and written **1** Timer is enabled and the counter decrements normally.

The timer is incremented every prescaler value+1. For example, if the prescaler has a value of five then the global timer is incremented every six clock cycles. **PERIPHCLK** is the reference clock for this.

4.2.5 Private Timer Interrupt Status Register

Bit assignments for the Private Timer Interrupt Status Register.

This is a banked register for all Cortex-A9 processors present.

The event flag is a sticky bit that is automatically set when the Counter Register reaches zero. If the timer interrupt is enabled, Interrupt ID 29 is set as pending in the Interrupt Distributor after the event flag is set. The event flag is cleared when written to 1.

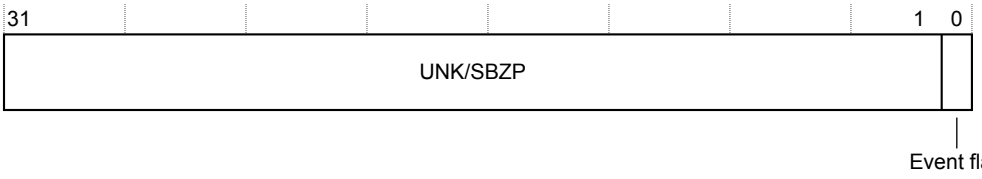

Figure 4-3 Private Timer Interrupt Status Register bit assignment

4.2.6 Watchdog Load Register

The Watchdog Load Register contains the value copied to the Watchdog Counter Register when it decrements down to zero with auto reload mode enabled, in Timer mode.

Writing to the Watchdog Load Register means that you also write to the Watchdog Counter Register.

4.2.7 Watchdog Counter Register

The Watchdog Counter Register is a down counter.

It decrements if the Watchdog is enabled using the Watchdog enable bit in the Watchdog Control Register. If the Cortex-A9 processor associated with the Watchdog is in debug state, the counter does not decrement until the Cortex-A9 processor returns to non-debug state.

When the Watchdog Counter Register reaches zero and auto reload mode is enabled, and in timer mode, it reloads the value in the Watchdog Load Register and then decrements from that value. If auto reload mode is not enabled or the watchdog is not in timer mode, the Watchdog Counter Register decrements down to zero and stops.

When in watchdog mode, the only way to update the Watchdog Counter Register is to write to the Watchdog Load Register. When in timer mode the Watchdog Counter Register is write accessible.

The behavior of the watchdog when the Watchdog Counter Register reaches zero depends on its mode:

Timer mode When the Watchdog Counter Register reaches zero, the watchdog interrupt status event flag is set and the interrupt ID 30 is set as pending in the Interrupt Distributor, if interrupt generation is enabled in the Watchdog Control Register.

Watchdog mode If a software failure prevents the Watchdog Counter Register from being refreshed, the Watchdog Counter Register reaches zero, the Watchdog reset status flag is set, and the associated **WDRESETREQ** reset request output pin is asserted for one **PERIPHCLK** cycle. The external reset source is then responsible for resetting all or part of the Cortex-A9 MPCore design.

4.2.8 Watchdog Control Register

Bit assignments for the Watchdog Control Register.

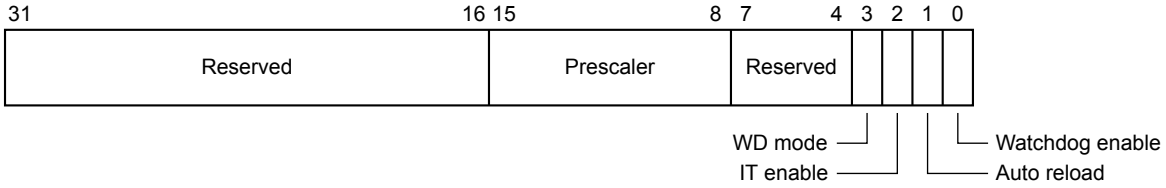

Figure 4-4 Watchdog Control Register bit assignments

The following table shows the Watchdog Control Register bit assignments.

Table 4-3 Watchdog Control Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved.
[15:8]	Prescaler	The prescaler modifies the clock period for the decrementing event for the Counter Register. See *4.1.1 Calculating timer intervals* on page 4-63.
[7:4]	-	Reserved.
[3]	Watchdog mode	**0** Timer mode, default. Writing a zero to this bit has no effect. You must use the Watchdog Disable Register to put the watchdog into timer mode. See *4.2.11 Watchdog Disable Register* on page 4-69. **1** Watchdog mode.
[2]	IT Enable	If set, the interrupt ID 30 is set as pending in the Interrupt Distributor when the event flag is set in the watchdog Status Register. In watchdog mode, this bit is ignored.
[1]	Auto-reload	**0** Single shot mode. Counter decrements down to zero, sets the event flag and stops. **1** Auto-reload mode. Each time the Counter Register reaches zero, it is reloaded with the value contained in the Load Register and then continues decrementing.
[0]	Watchdog Enable	Global watchdog enable **0** Watchdog is disabled and the counter does not decrement. All registers can still be read and /or written. **1** Watchdog is enabled and the counter decrements normally.

4.2.9 Watchdog Interrupt Status Register

Bit assignments for the Watchdog Interrupt Status Register.

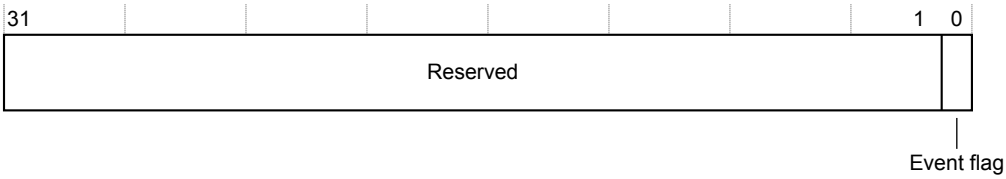

Figure 4-5 Watchdog Interrupt Status Register bit assignment

The event flag is a sticky bit that is automatically set when the Counter Register reaches zero in timer mode. If the watchdog interrupt is enabled, Interrupt ID 30 is set as pending in the Interrupt Distributor after the event flag is set. The event flag is cleared when written with a value of 1. Trying to write a zero to the event flag or a one when it is not set has no effect.

4.2.10 Watchdog Reset Status Register

Bit assignments for the Watchdog Reset Status Register.

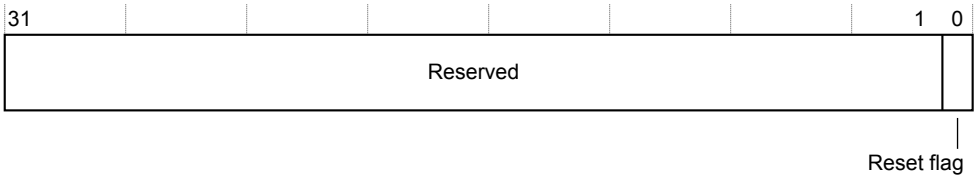

Figure 4-6 Watchdog Reset Status Register bit assignment

The reset flag is a sticky bit that is automatically set, in watchdog mode, when the Counter Register reaches zero and a reset request is sent accordingly.

The reset flag is cleared when written with a value of 1. Trying to write a zero to the reset flag or a one when it is not set has no effect. This flag is not reset by normal Cortex-A9 processor resets but has its own reset line, **nWDRESET**. **nWDRESET** must not be asserted when the Cortex-A9 processor reset assertion is the result of a watchdog reset request with **WDRESETREQ**. This distinction enables software to differentiate between a normal boot sequence, reset flag is zero, and one caused by a previous watchdog time-out, reset flag set to one.

4.2.11 Watchdog Disable Register

Use the Watchdog Disable Register to switch from watchdog to timer mode. The software must write `0x12345678` then `0x87654321` successively to the Watchdog Disable Register so that the watchdog mode bit in the Watchdog Control Register is set to zero.

If one of the values written to the Watchdog Disable Register is incorrect or if any other write occurs in between the two word writes, the watchdog remains in the same mode. To reactivate the Watchdog, the software must write 1 to the watchdog mode bit of the Watchdog Control Register.

Related references

4.2.8 Watchdog Control Register on page 4-67.

Related references

1.6 Interfaces on page 1-19.

A.2 Resets and reset control signals on page Appx-A-92.

4.3 About the Global Timer

List of features of the global timer.

- The global timer is a 64-bit incrementing counter with an auto-incrementing feature. It continues incrementing after sending interrupts.
- The global timer is memory mapped in the private memory region.
- The global timer is accessed at reset in Secure State only.
- The global timer is accessible to all Cortex-A9 processors in the cluster. Each Cortex-A9 processor has a private 64-bit comparator that is used to assert a private interrupt when the global timer has reached the comparator value. All the Cortex-A9 processors in a design use the banked ID, ID27, for this interrupt. ID27 is sent to the Interrupt Controller as a Private Peripheral Interrupt.
- The global timer is clocked by **PERIPHCLK**.

——— **Note** ———

- From r2p0, the comparators for each processor with the global timer fire when the timer value is greater than or equal to. In previous revisions the comparators fired when the timer value was equal to.
- The global timer does not stop counting when any of the processors are in debug state.

———————————

Related references

1.5 Private Memory Region on page 1-17.

2.2.9 SCU Non-secure Access Control Register on page 2-34.

3.1.2 Interrupt Distributor interrupt sources on page 3-48.

4.4 Global timer registers

Summary of global timer registers with information on bit assignments for each register.

This section contains the following subsections:

4.4.1 Global timer register summary

List of global timer registers.

The offset is relative to PERIPH_BASE_ADDR + `0x0200`. Use **nPERIPHRESET** to reset these registers.

Table 4-4 Global timer registers

Offset	Type	Reset value	Banked	Function
`0x00`	R/W	`0x00000000`	No	*4.4.2 Global Timer Counter Registers, 0x00 and 0x04* on page 4-71
`0x04`	R/W	`0x00000000`		
`0x08`	R/W	`0x00000000`	Yes [i]	*4.4.3 Global Timer Control Register* on page 4-72
`0x0C`	R/W	`0x00000000`	Yes	*4.4.4 Global Timer Interrupt Status Register* on page 4-73
`0x10`	R/W	`0x00000000`	Yes	*4.4.5 Comparator Value Registers, 0x10 and 0x14* on page 4-73
`0x14`	R/W	`0x00000000`		
`0x18`	R/W	`0x00000000`	Yes	*4.4.6 Auto-increment Register, 0x18* on page 4-73

4.4.2 Global Timer Counter Registers, 0x00 and 0x04

There are two timer counter registers. They are the lower 32-bit timer counter at offset `0x00` and the upper 32-bit timer counter at offset `0x04`.

You must access these registers with 32-bit accesses. You cannot use `STRD`/`LDRD`.

Modifying the timer enable bit

Modifying the timer enable bit by writing 32-bit timer counter registers.

Procedure

1. Clear the timer enable bit in the Global Timer Control Register
2. Write the lower 32-bit timer counter register.
3. Write the upper 32-bit timer counter register.

[i] Some bits

4. Set the timer enable bit.

Getting the 64-bit Global Timer Counter register value

Getting the 64-bit value of the Global Timer Counter register by reading 32-bit timer counter registers.

Procedure

1. Read the upper 32-bit timer counter register.
2. Read the lower 32-bit timer counter register.
3. Read the upper 32-bit timer counter register again. If the value is different to the 32-bit upper value read previously, go to the previous step. Otherwise the 64-bit timer counter value is correct.

4.4.3 Global Timer Control Register

Bit assignments for the Global Timer Control Register.

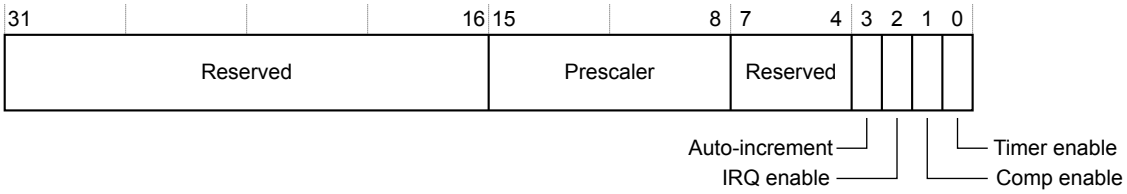

Figure 4-7 Global Timer Control Register bit assignments

The following table shows the Global Timer Control Register bit assignments.

Table 4-5 Global Timer Control Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved
[15:8]	Prescaler	The prescaler modifies the clock period for the decrementing event for the Counter Register. See *4.1.1 Calculating timer intervals* on page 4-63 for the equation.
[7:4]	-	Reserved
[3]	Auto-increment[j]	This bit is banked per Cortex-A9 processor. **0** Single shot mode. When the counter reaches the comparator value, the event flag is set. It is the responsibility of software to update the comparator value to get more events. **1** Auto increment mode. Each time the counter reaches the comparator value, the comparator register is incremented with the auto-increment register, so that more events can be set periodically without any software updates.
[2]	IRQ Enable	This bit is banked per Cortex-A9 processor. If set, the interrupt ID 27 is set as pending in the Interrupt Distributor when the event flag is set in the Timer Status Register.

[j] When the Auto-increment and Comp enable bits are set, an IRQ is generated every auto-increment register value.

Table 4-5 Global Timer Control Register bit assignments (continued)

Bits	Name	Function
[1]	Comp Enable[j]	This bit is banked per Cortex-A9 processor. If set, it enables the comparison between the 64-bit Timer Counter and the related 64-bit Comparator Register.
[0]	Timer Enable	Timer enable **0** Timer is disabled and the counter does not increment. All registers can still be read and written. **1** Timer is enabled and the counter increments normally.

4.4.4 Global Timer Interrupt Status Register

Bit assignments for the Global Timer Interrupt Status Register.

This is a banked register for all Cortex-A9 processors present.

The event flag is a sticky bit that is automatically set when the Counter Register reaches the Comparator Register value. If the timer interrupt is enabled, Interrupt ID 27 is set as pending in the Interrupt Distributor after the event flag is set. The event flag is cleared when written to 1.

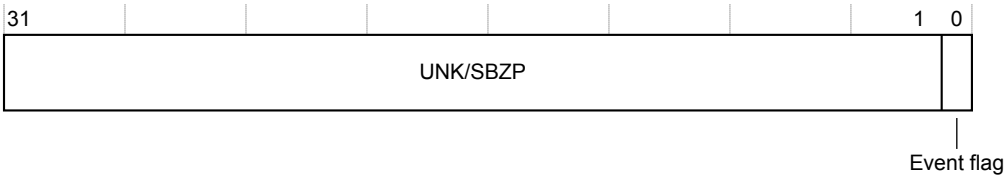

Figure 4-8 Global Timer Interrupt Status Register bit assignment

4.4.5 Comparator Value Registers, 0x10 and 0x14

There are two 32-bit registers, the lower 32-bit comparator value register at offset `0x10` and the upper 32-bit comparator value register at offset `0x14`.

You must access these registers with 32-bit accesses. You cannot use `STRD`/`LDRD`. There is a Comparator Value Register for each Cortex-A9 processor.

Setting the Comp enable bit and the IRQ enable bit

Ensuring that updates to this register do not set the Interrupt Status Register.

Procedure

1. Clear the Comp Enable bit in the Timer Control Register.
2. Write the lower 32-bit Comparator Value Register.
3. Write the upper 32-bit Comparator Value Register.
4. Set the Comp Enable bit and, if necessary, the IRQ enable bit.

4.4.6 Auto-increment Register, 0x18

This 32-bit register gives the increment value of the Comparator Register when the Auto-increment bit is set in the Timer Control Register. Each Cortex-A9 processor present has its own Auto-increment Register.

If the comp enable and auto-increment bits are set when the global counter reaches the Comparator Register value, the comparator is incremented by the auto-increment value, so that a new event can be set periodically.

The global timer is not affected and goes on incrementing.

Chapter 5
Clocks, Resets, and Power Management

This chapter describes the clocks, resets, and power management features of the Cortex-A9 MPCore.

It contains the following sections:

5.1 Clocks

List of functional clock inputs of the processor and an example of the **PERIPHCLK**.

The Cortex-A9 MPCore processor does not have any asynchronous interfaces. Therefore, all the bus interfaces and the interrupt signals must be synchronous with reference to **CLK**.

CLK

This is the main clock of the Cortex-A9 processor.

All Cortex-A9 processors in the Cortex-A9 MPCore processor and the SCU are clocked with a distributed version of CLK.

PERIPHCLK

The Interrupt Controller, global timer, private timers, and watchdogs are clocked with **PERIPHCLK**.

PERIPHCLK must be synchronous with **CLK**, and the **PERIPHCLK** clock period, N, must be configured as a multiple of the **CLK** clock period. This multiple N must be equal to, or greater than two.

PERIPHCLKEN

This is the clock enable signal for the Interrupt Controller and timers. The **PERIPHCLKEN** signal is generated at **CLK** clock speed. **PERIPHCLKEN** HIGH on a CLK rising edge indicates that there is a corresponding **PERIPHCLK** rising edge.

The following figure shows an example with the **PERIPHCLK** clock period N as three.

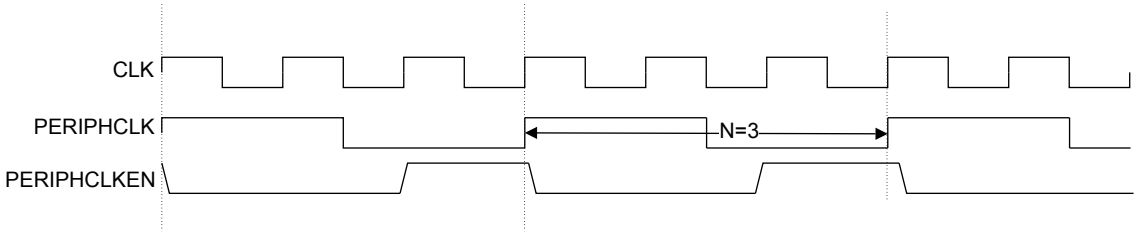

Figure 5-1 Three-to-one timing ratio

Note

From r2p0 onwards, PERIPHCLK can remain inactive in cases when you do not use any of the peripherals in the Private Memory Region.

5.2 Resets

The reset signals present in the Cortex-A9 MPCore processor design enable you to reset different parts of the design independently.

This section contains the following subsections:

5.2.1 Reset combinations

List of different reset combinations that can be expected in a Cortex-A9 MPCore system.

In the following table, [n] refers to the Cortex-A9 processor that initiates a reset.

Table 5-1 Reset combinations in a Cortex-A9 MPCore system

	nSCURESET and nPERIPHRESET	nCPURESET[3:0]	nNEONRESET[3:0]	nDBGRESET[3:0]	nWDRESET[3:0]
Cortex-A9 MPCore Power on reset	0	All 0	All 0	All 0	All 0
Cortex-A9 MPCore Software reset	0	All 0	All 0	All 1	All 0
Per processor Power-on reset	1	[n]=0	[n]=0	[n]=0	[n]=0 or all 1
Per processor Software reset	1	[n]=0	[n]=0	All 1	[n]=0 or all 1
SIMD MPE power on	1	All 1	[n]=0	All 1	All 1
Cortex-A9 MPCore Debug	1	All 1	All 1	All 0	All 1
Per processor Debug	1	All 1	All 1	[n]=0	All 1
Per processor Watchdog flag	1	All 1	All 1	All 1	[n]=0

5.2.2 Cortex-A9 MPCore power-on reset

This power-on or cold reset initializes the whole logic in the Cortex-A9 MPCore processor. You must apply power-on or cold reset to the Cortex-A9 MPCore processor when power is first applied to the system.

In the case of power-on reset, the leading (falling) edge of the reset signals does not have to be synchronous to **CLK** but the rising edge must be. This is achieved by using the **CPUCLKOFF** and **NEONCLKOFF** signals. You must assert the reset signals for at least nine **CLK** cycles to ensure correct reset behavior.

Performing a reset sequence on power-on

Recommended procedure of performing a reset sequence on power-on.

Procedure

1. Apply all resets: **nCPURESET**, **nDBGRESET**, **nWDRESET**, **nSCURESET**, **nPERIPHRESET**, and **nNEONRESET** if the SIMD MPE is present.
2. Apply at least nine CLK cycles, plus at least one cycle in each other clock domain, or more if the documentation for other components requests it. There is no harm in applying more clock cycles than this, and maximum redundancy can be achieved by applying 15 cycles on every clock domain.
3. Assert all **CPUCLKOFF** signals with a value of 0b1 and, if there is an SIMD MPE present, all **NEONCLKOFF**.
4. Stop **CLK** and **PERIPHCLK**.
5. Wait for the equivalent of approximately ten cycles, depending on your implementation. This compensates for clock and reset tree latencies.
6. Release resets.
7. Wait for the equivalent of another approximately ten cycles, again to compensate for clock and reset tree latencies.
8. Deassert all **CPUCLKOFF** and **NEONCLKOFF**. This ensures that all registers in the design see the same **CLK** edge on exit from the reset sequence.
9. Start **CLK** and **PERIPHCLK**.

5.2.3 Cortex-A9 MPCore software reset

This software or warm reset initializes all functional logic in each of the individual Cortex-A9 processor present in the cluster apart from the debug logic.

All breakpoints and watchpoints are retained during this.

ARM recommends that you use the power-on reset sequence, except that **nDBGRESET** must not be asserted during the sequence. This ensures that the debug registers retain their values.

Related tasks

Performing a reset sequence on power-on on page 5-78.

5.2.4 Individual processor power-on reset

This reset initializes the whole logic in a single Cortex-A9 processor, including its debug logic. It is expected to be applied when this individual Cortex-A9 processor exits from power down or dormant state.

This reset only applies to configurations where each individual Cortex-A9 processor is implemented in its own power domain.

Performing a power-on reset sequence on a single processor

Procedure of the power-on reset sequence when performed on a single processor.

Procedure

1. Apply **nCPURESET[n]** and **nDBGRESET[n]**, plus **nNEONRESET[n]** if the SIMD MPE is present. **nWDRESET[n]** reset can also be applied optionally if you want to reset the corresponding Watchdog flag.
2. Wait for at least nine CLK cycles, plus at least one cycle in each other clock domain, or more if the documentation for other components requests it. There is no harm in applying more clock cycles than this, and maximum redundancy can be achieved by for example applying 15 cycles on every clock domain.

3. Assert **CPUCLKOFF[n]** with a value of 0b1 and, if there is a SIMD MPE present, **NEONCLKOFF[n]**.
4. Wait for the equivalent of approximately ten cycles, depending on your implementation. This compensates for clock and reset tree latencies.
5. Release all resets.
6. Wait for the equivalent of another approximately ten cycles, again to compensate for clock and reset tree latencies.
7. Deassert **CPUCLKOFF[n]** and **NEONCLKOFF[n]**. This ensures that all registers in the processor, and in the SIMD MPE, see the same CLK edge on exit from the reset sequence.

5.2.5 Individual processor software reset

This reset initializes all functional logic in a single Cortex-A9 processor apart from its debug logic.

All breakpoints and watchpoints are retained during this individual warm reset.

This reset only applies to configuration where each individual Cortex-A9 processor is implemented in its own power domain

ARM recommends that you use the individual processor power-on reset sequence, except that **nDBGRESET** must not be asserted during the sequence. This ensures the debug registers of the individual processors retain their values.

Related tasks

Performing a power-on reset sequence on a single processor on page 5-78.

5.2.6 Individual processor power-on SIMD MPE reset

This reset initializes all the SIMD logic of the MPE in a single Cortex-A9 processor. It is expected to be applied when the SIMD part of the MPE exits from powerdown state.

This reset only applies to configurations where SIMD MPE logic is implemented in its own dedicated power domain, separated from the rest of the processor logic.

Performing a power-on reset sequence on an individual CPU SIMD MPE

Recommended procedure of performing a reset sequence on power-on for an individual CPU SIMD MPE power-on.

Procedure

1. Apply **nNEONRESET[n]**.
2. Wait for at least nine CLK cycles. There is no harm in applying more clock cycles than this, and maximum redundancy can be achieved by for example applying 15 cycles on every clock domain.
3. Assert **NEONCLKOFF[n]** with a value of `0b1`.
4. Wait for the equivalent of approximately ten cycles, depending on your implementation. This compensates for clock and reset tree latencies.
5. Release **nNEONRESET[n]**.
6. Wait for the equivalent of approximately another ten cycles, again to compensate for clock and reset tree latencies.
7. Deassert **NEONCLKOFF[n]**. This ensures that all registers in the SIMD MPE part of the processor see the same **CLK** edge on exit from the reset sequence.

5.2.7 Cortex-A9 MPCore debug reset

This reset initializes the debug logic in all Cortex-A9 processors present in the cluster.

To perform a Cortex-A9 MPCore debug reset, assert all **nDBGRESET** signals during a few **CLK** cycles. **CPUCLKOFF** and **NEONCLKOFF** must remain deasserted during this reset sequence.

5.2.8 Individual processor debug reset

This reset initializes the debug logic in a single Cortex-A9 processor in the cluster.

To perform a Cortex-A9 individual processor debug reset, assert the corresponding **nDBGRESET[n]** signal during a few CLK cycles. **CPUCLKOFF[n]** and **NEONCLKOFF[n]** must remain deasserted during this reset sequence.

5.2.9 Individual processor watchdog flag reset

This reset clears the watchdog flag associated with a single Cortex-A9 processor. Watchdog functionality is independent from all other processor functionality, so this reset is independent from the all other resets.

5.3 Power management

Description of the different parts of the Cortex-A9MPCore power management system.

This section contains the following subsections:

5.3.1 Individual Cortex-A9 processor power management

Description of power modes in the processor.

About power modes

There are four power management modes available. These modes are run, standby, dormant, and shutdown.

Place holders for clamps are inserted around each Cortex-A9 processor so that implementation of different power domains can be eased. It is the responsibility of software to signal to the Snoop Control Unit and the Distributed Interrupt Controller that a Cortex-A9 processor is shut off so that the Cortex-A9 processor can be seen as non-existent in the cluster. Each Cortex-A9 processor can be in one of the following modes:

Run mode
Everything is clocked and powered-up

Standby mode
The CPU clock is stopped. Only logic required for wake-up is still active.

Dormant mode
Everything is powered off except RAM arrays that are in retention mode.

Shutdown
Everything is powered-off.

The following table shows the individual power modes.

Table 5-2 Cortex-A9 MPCore power modes

Mode	Cortex-A9 processor logic	RAM arrays	Wake-up mechanism
Run Mode	Powered-up Everything clocked	Powered-up	N/A
Standby modes	Powered-up Only wake-up logic clocked	Powered-up	Standard Standby modes wake up events. See *Standby modes* on page 5-82.
Dormant	Powered-off	Retention state/ voltage	External wake-up event to power controller, that can perform a reset of the processor.
Shutdown	Powered-off	Powered-off	External wake-up event to power controller, that can perform a reset of the processor.

Entry to Dormant or powered-off mode must be controlled through an external power controller. The CPU Status Register in the SCU is used with the CPU WFI entry flag to signal to the power controller the power domain that it can cut, using the PWRCTL bus.

Run mode

Run mode is the normal mode of operation, where all the functionality of the Cortex-A9 processor is available.

Standby modes

WFI and WFE Standby modes disable most of the clocks in a processor, while keeping its logic powered up. This reduces the power drawn to the static leakage current, leaving a tiny clock power overhead requirement to enable the device to wake up.

Entry into WFI Standby mode is performed by executing the `WFI` instruction.

The transition from the WFI Standby mode to the Run mode is caused by:

- An **IRQ** interrupt, regardless of the value of the CSPR.I bit.
- An **FIQ** interrupt, regardless of the value of the CSPR.F bit.
- An asynchronous abort, regardless of the value of the CPSR.A bit.
- A debug event, if invasive debug is enabled and the debug event is permitted.
- A CP15 maintenance request broadcast by other processors.

Entry into WFE Standby mode is performed by executing the WFE instruction.

The transition from the WFE Standby mode to the Run mode is caused by:

- An **IRQ** interrupt, unless masked by the CPSR.I bit.
- An **FIQ** interrupt, unless masked by the CPSR.F bit.
- An asynchronous abort, unless masked by the CPSR.A bit.
- A debug event, if invasive debug is enabled and the debug event is permitted.
- The assertion of the **EVENTI** input signal.
- The execution of an SEV instruction on any processor in the multiprocessor system.
- A CP15 maintenance request broadcast by other processors.

The debug request can be generated by an externally generated debug request, using the **EDBGRQ** pin on the Cortex-A9 processor, or from a Debug Halt instruction issued to the Cortex-A9 processor through the APB debug port.

The debug channel remains active throughout a `WFI` instruction.

Note

When a processor in Standby mode receives an SCU coherency request, the clock on its L1 memory system is restored temporarily so that the request can be handled. This mechanism prevents the requirement for a processor about to enter Standby mode from having to flush its L1 data cache by ensuring that its coherent data remain accessible by other processors.

Dormant mode

Dormant mode is designed to enable the Cortex-A9 processor to be powered down, while leaving the caches powered up and maintaining their state.

The RAM blocks that are to remain powered up must be implemented on a separate power domain, and there is a requirement to clamp all the inputs to the RAMs to a known logic level, with the chip enable being held inactive. This clamping is not implemented in gates as part of the default synthesis flow because it would contribute to a tight critical path. Implementations that want to implement Dormant mode must add these clamps around the RAMs, either as explicit gates in the RAM power domain, or as pull-down transistors that clamp the values while the Cortex-A9 processor is powered down. The RAM blocks that must remain powered up during Dormant mode are:

- All Data RAMs associated with the cache.
- All Tag RAMs associated with the cache.

Before entering Dormant mode, the state of the Cortex-A9 processor, excluding the contents of the RAMs that remain powered up in dormant mode, must be saved to external memory. These state saving operations must ensure that the following occur:

- All ARM registers, including CPSR and SPSR registers are saved.
- All system registers are saved.

- All debug-related state must be saved.
- The Cortex-A9 processor must correctly set the CPU Status Register in the SCU so that it enters Dormant Mode.
- A Data Synchronization Barrier instruction is executed to ensure that all state saving has been completed.
- The Cortex-A9 processor then communicates with the power controller that it is ready to enter dormant mode by performing a WFI instruction so that power control output reflects the value of SCU CPU Status Register.

Transition from Dormant mode to Run mode is triggered by the external power controller. The external power controller must assert reset to the Cortex-A9 processor until the power is restored. After power is restored, the Cortex-A9 processor leaves reset, and by interrogating the power control register in SCU, can determine that the saved state must be restored.

Related references

2.2.4 SCU CPU Power Status Register on page 2-29.

Shutdown mode

Shutdown mode has the entire device powered down, and all state, including cache, must be saved externally by software.

The part is returned to the run state by the assertion of reset. This state saving is performed with interrupts disabled, and finishes with a DSB operation. The Cortex-A9 processor then communicates with a power controller that the device is ready to be powered down in the same manner as when entering Dormant Mode.

5.3.2 Communication to the Power Management Controller

Communication between the Cortex-A9 processor and the external Power Management Controller can be performed using the **PWRCTLOn** Cortex-A9 MPCore output signals and Cortex-A9 MPCore input clamp signals.

PWRCTLOn Cortex-A9 MPCore output signals	These signals constrain the external Power Management Controller. The value of **PWRCTLOn** depends on the value of the SCU CPU Status Register. The SCU CPU Status Register value is only copied to **PWRCTLOn** after the Cortex-A9 processor signals that it is ready to enter low-power mode by executing a WFI instruction and subsequent **STANDBYWFI** pin assertion.
Cortex-A9 MPCore input signals	The external Power Management Controller uses **CPUCLAMP[3:0]**, **NEONCLAMP[3:0]**, and **CPURAMCLAMP[4:0]** to isolate Cortex-A9 MPCore power domains from one another before they are turned off. These signals are only meaningful if the Cortex-A9 MPCore processor has been implemented with power clamps designed in.

Related references

2.2.4 SCU CPU Power Status Register on page 2-29.

5.3.3 Cortex-A9 MPCore power domains

The Cortex-A9 MPCore processor can support up to 14 power domains.

The supported power domains are the following:

- Four power domains, one for each of the Cortex-A9 processors, apart from their Data Engines.
- Four power domains, one for each of the Cortex-A9 processor Data Engines.
- Four power domains, one for each of the Cortex-A9 processor caches and TLB RAMs.
- One power domain for SCU duplicated TAG RAMs.
- One power domain for remaining logic, the SCU logic cells, and private peripherals.

The following figure shows the power domains and where placeholders are inserted for power domain isolation.

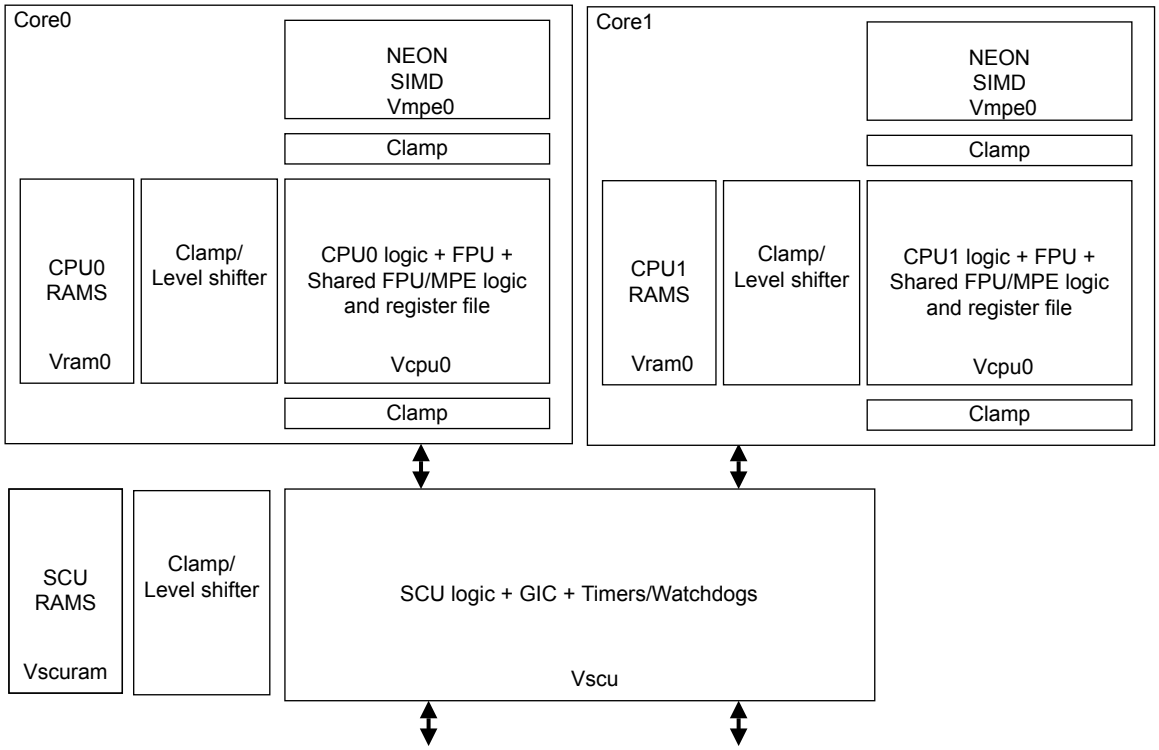

Figure 5-2 Cortex-A9 MPCore power domains and clamps

5.3.4 About multiprocessor bring-up

There are several possible ways to set up the multiprocessing capabilities of the Cortex-A9 MPCore. Examples of this set up for the primary processor and the non-primary processor are provided.

In this description of multiprocessor bring-up:

- All operations within a step on a single processor can occur in any order.
- All operations on one step on a single processor must occur before any operations in a subsequent step occur on that processor.
- All operations on a non-lead processor must not occur before the equivalent step number on the lead processor.

No other ordering applies.

Note

L2C-310 cache controller setup is fully independent. You can perform setup at any time, before or after the Cortex-A9 MPCore multiprocessing bring-up.

Performing bring-up on the primary processor

Bring-up on process for the primary processor.

Procedure

1. Invalidate:
   - The SCU duplicate tags for all processors.
   - The data cache.
2. Enable the SCU.
3. Enable the data cache, set the SMP mode with ACTLR.SMP=1.

Performing bring-up on the non-primary processor

Bring-up on process for the non-primary processor.

Procedure

1. Invalidate the data cache.
2. Wait for the SCU to be enabled by the primary processor.
3. Enable the data cache, set the SMP mode with ACTLR.SMP=1.

Chapter 6
Debug

This chapter describes some of the debug and trace considerations in Cortex-A9 MPCore designs.

It contains the following sections:

6.1 External Debug Interface signals

In the Cortex-A9 MPCore implementation, the debug interface of each individual Cortex-A9 processor is exported to the MPCore boundary, so that each individual Cortex-A9 can be debugged independently.

Multi-processing debug capabilities, such as cross-triggering, can be configured externally to the Cortex-A9 MPCore. See the *CoreSight™ v1.0 Architecture Specification* and *ARM® Debug Interface v5 Architecture Specification*.

The following figure shows the Cortex-A9 MPCore external debug interface signals.

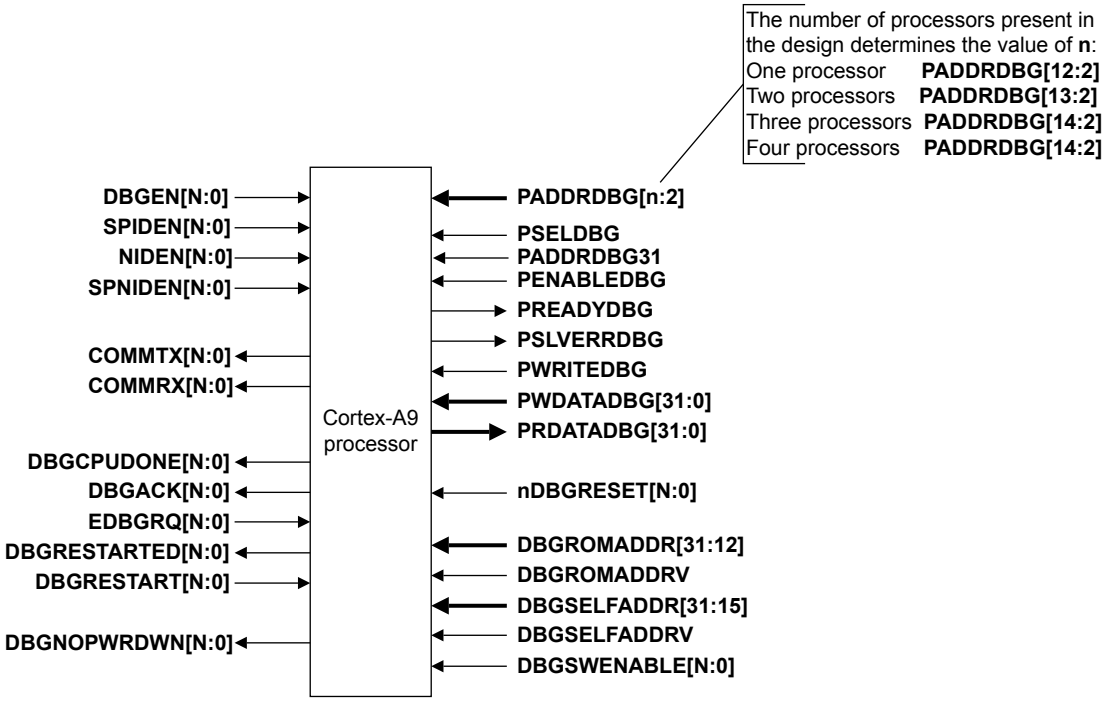

Figure 6-1 External debug interface signals in CortexA9 MPCore designs

A few signals on the Cortex-A9 MPCore debug interface are common to all Cortex-A9 processors in the cluster. This is the case for the APB debug interface.

The CortexA9 MPCore external debug interface does not implement:

- **DBGTRIGGER**.
- **DBGPWRDUP**.
- **DBGOSLOCKINIT**.

Related references

6.2 Cortex-A9 MPCore APB Debug interface and memory map on page 6-88.

6.2 Cortex-A9 MPCore APB Debug interface and memory map

The Cortex-A9 MPCore has a single Debug APB interface to access the individual Cortex-A9 processors in the cluster. Because it contains between one and four individual Cortex-A9 processors, the Cortex-A9 MPCore appears as an 8KB, 16KB, 24KB, or 32KB CoreSight memory region, accessed when **PSELDBG** is asserted.

Each Cortex-A9 processor contains two 4KB CoreSight components, for the debug and performance monitor resources, mapped in a contiguous 8KB memory region. See the *ARM® Cortex®-A9 Technical Reference Manual* for detailed memory mapping of this 8KB memory region.

This section contains the following subsections:

6.2.1 PADDRDBG values

The value of **PADDRDBG** differs in the Cortex-A9 MPCore configurations. It can have a value of [12:0], [13:0], or [14:0] depending on the number of processors.

6.2.2 Configuration for a single Cortex-A9 processor

In this configuration, **PADDRDBG** is [12:0]. **PADDRDBG[12]** is used to select the debug or performance monitor area of the processor.

- Use **PADDRDBG[12]** = 0 to access the debug area of the Cortex-A9 processor.
- Use **PADDRDBG[12]** = 1 to access the performance monitor area of the Cortex-A9 processor.

6.2.3 Configuration for two Cortex-A9 processors

In this configuration, **PADDRDBG** is [13:0]. **PADDRDBG[13]** is used to select which of the processors is accessed.

- Use **PADDRDBG[13]** = 0 to access CPU0 resources.
- Use **PADDRDBG[13]** = 1 to access CPU1 resources.

PADDRDBG[12] is used to select the debug or performance monitor area of the processor:

- Use **PADDRDBG[12]** = 0 to access the debug area of the selected Cortex-A9 processor.
- Use **PADDRDBG[12]** = 1 to access the performance monitor area of the selected Cortex-A9 processor.

6.2.4 Configuration for three Cortex-A9 processors

In this configuration, **PADDRDBG** is [14:0]. **PADDRDBG[14:13]** is used to select which of the processors is accessed.

- Use **PADDRDBG[14:13]** = 00 to access CPU0 resources.
- Use **PADDRDBG[14:13]** = 01 to access CPU1 resources.
- Use **PADDRDBG[14:13]** = 10 to access CPU2 resources.

PADDRDBG[12] is used to select the debug or performance monitor area of the processor

- Use **PADDRDBG[12]** = 0 to access the debug area of the selected Cortex-A9 processor.
- Use **PADDRDBG[12]** = 1 to access the performance monitor area of the selected Cortex-A9 processor.

Note

In this configuration, the external CoreSight system must ensure that the Cortex-A9 MPCore is never accessed with **PADDRDBG[14:13]** = 11. When **PADDRDBG[14:13]** = 11, **PSELDBG** must not be asserted.

6.2.5 Configuration for four Cortex-A9 processors

In this configuration, **PADDRDBG** is [14:0]. **PADDRDBG[14:13]** is used to select which of the processors is accessed.

- Use **PADDRDBG[14:13]** = 00 to access CPU0 resources.
- Use **PADDRDBG[14:13]** = 01 to access CPU1 resources.
- Use **PADDRDBG[14:13]** = 10 to access CPU2 resources.
- Use **PADDRDBG[14:13]** = 11 to access CPU3 resources.

PADDRDBG[12] is used to select the debug or performance monitor area of the processor

- Use **PADDRDBG[12]** = 0 to access the debug area of the selected Cortex-A9 processor.
- Use **PADDRDBG[12]** = 1 to access the performance monitor area of the selected Cortex-A9 processor.

Appendix A
Signal Descriptions

This appendix describes the Cortex-A9 MPCore signals.

In signal names such as **TEINIT[N:0]**, the value of N is one less than the number of processors in your design.

It contains the following sections:

A.1 Clock and clock control signals

List of clock and clock control signals.

Table A-1 Cortex-A9 MPCore clocks and clock control signals

Name	I/O	Source	Description
CLK	I	Clock controller	Global clock
MAXCLKLATENCY[2:0]	I	Implementation-specific static value	Control dynamic clock gating delays. These pins are sampled during reset of the processor.
PERIPHCLK	I	Clock controller	Clock for the timer and Interrupt Controller
PERIPHCLKEN	I	Clock controller	Clock enable for the timer and Interrupt Controller

Related references

Chapter 5 Clocks, Resets, and Power Management on page 5-75.

A.2 Resets and reset control signals

List of reset, reset clock control, and watchdog request reset signals.

This section contains the following subsections:

A.2.1 Reset signals

Description of reset signals.

Table A-2 Reset signals

Name	I/O	Source	Description
nCPURESET[N:0]	I	Reset controller or clock controller	Individual Cortex-A9 processor resets
nDBGRESET[N:0]	I		Processor debug logic resets
nNEONRESET[N:0] Only if an MPE is present	I		Cortex-A9 MPE SIMD logic resets
nPERIPHRESET	I		Timer and interrupt controller reset
nSCURESET	I		SCU global reset
nWDRESET[N:0]	I		Processor watchdog resets

A.2.2 Reset clock control signals

The reset clock control signals are used to cut the clocks during reset sequences. **NEONCLCKOFF[N:0]** is only present when there is a Data Engine in your design.

Table A-3 Reset clock control signals

Name	I/O	Source	Description
CPUCLKOFF[N:0]	I	Reset controller	Individual Cortex-A9 processor CPU clock enable, active-LOW: **0** Clock is enabled. **1** Clock is stopped.
NEONCLKOFF[N:0]	I		MPE SIMD logic clock control: **0** Do not cut MPE SIMD logic clock. **1** Cut MPE SIMD logic clock.

Related references

Chapter 5 Clocks, Resets, and Power Management on page 5-75.

A.2.3 Watchdog request reset signal

WDRESETREQ[N:0] is the watchdog request reset signal.

Table A-4 Watchdog request reset signal

Name	I/O	Destination	Description
WDRESETREQ[N:0]	O	System exception controller	Processor watchdog reset requests

Related references

Chapter 4 Global timer, private timers, and watchdog registers on page 4-62.

A.3 Interrupts

List of interrupt line signals.

Table A-5 Interrupt line signals

Name	I/O	Source	Description
IRQS[x:0]	I	Interrupt sources	Interrupt distributor interrupt lines. x can be 31, 63,…, up to 223 by increments of 32. If there are no interrupt lines this pin is removed. See *Chapter 3 Interrupt Controller* on page 3-47.
nIRQ[N:0]	I		Individual Cortex-A9 processor legacy 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.
nFIQ[N:0]	I		Individual Cortex-A9 processor private FIQ request input lines. 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.
nIRQOUT[N:0]	O	Power controller	Active-LOW output of individual processor nIRQ from the Interrupt Controller. For use when processors are powered off and interrupts are handled by the Interrupt Controller under the control of an external power controller.
nFIQOUT[N:0]	O		Active-LOW output of individual processor nFIQ from the Interrupt Controller. For use when processors are powered off and interrupts are handled by the Interrupt Controller under the control of an external power controller.

Note

For **IRQS[x:0]**, **nIRQ[N:0]**, and **nFIQ[N:0]**, the minimum pulse width of signals driving external interrupt lines is one **PERIPHCLK** cycle.

A.4 Configuration signals

List of configuration signals.

Table A-6 Configuration signals

Name	I/O	Source or destination	Description
CFGEND[N:0]	I	System configuration	Individual Cortex-A9 processor endianness configuration. Forces the EE bit in the CP15 c1 Control Register (SCTLR) to 1 at reset so that the Cortex-A9 processor boots with big-endian data handling. **0** EE bit is LOW. **1** EE bit is HIGH. This pin is only sampled during reset of the processor.
CFGNMFI[N:0]	I		Individual Cortex-A9 processor configuration of fast interrupts to be non-maskable: **0** Clear the NMFI bit in the CP15 c1 Control Register. **1** Set the NMFI bit in the CP15 c1 Control Register. This pin is only sampled during reset of the processor.
CLUSTERID[3:0]	I		Value read in Cluster ID register field, bits[11:8] of the MPIDR.
FILTEREN	I		For use with configurations with two master ports. Enables filtering of address ranges at reset. See *2.2.2 SCU Control Register* on page 2-27 for information on setting this signal.
FILTERSTART[31:20]	I		For use with configurations with two master ports. Specifies the start address for address filtering at reset. See *2.2.6 Filtering Start Address Register* on page 2-31.
FILTEREND[31:20]	I		For use with configurations with two master ports. Specifies the end address for address filtering. See *2.2.7 Filtering End Address Register* on page 2-32.
PERIPHBASE[31:13]	I		Specifies the base address for Timers, Watchdogs, Interrupt Controller, and SCU registers. Only accessible with memory-mapped accesses. This value can be retrieved by a Cortex-A9 processor using the CP15 c15 Configuration Base Address Register.
SMPnAMP[N:0]	O	System integrity controller	Signals AMP or SMP mode for each Cortex-A9 processor. **0** Asymmetric. **1** Symmetric.

Table A-6 Configuration signals (continued)

Name	I/O	Source or destination	Description
TEINIT[N:0]	I	System configuration	Individual Cortex-A9 processor out-of-reset default exception handling state. When set to: **0** ARM. **1** Thumb. This pin is only sampled during reset of the processor. It sets the initial value of SCTLR.TE.
VINITHI[N:0]	I		Individual Cortex-A9 processor control of the location of the exception vectors at reset: **0** Exception vectors start at address `0x00000000`. **1** Exception vectors start at address `0xFFFF0000`. This pin is only sampled during reset of the processor. It sets the initial value of SCTLR.V.

A.5 Security control signals

List of security control signals.

Table A-7 Security control signals

Name	I/O	Source or destination	Description
CFGSDISABLE	I	Security controller	Disables write access to some system control processor registers: **0** Not enabled. **1** Enabled. See *3.2.2 Using CFGSDISABLE* on page 3-50.
CP15SDISABLE[N:0]	I		Individual Cortex-A9 processor write access disable for some system control processor registers.

A.6 WFE and WFI Standby signals

List of WFI and WFE Standby mode signals.

Table A-8 Standby and wait for event signals

Name	I/O	Source or Destination	Description
EVENTI	I	External coherent agent	Event input for Cortex-A9 processor to wake up from WFE Standby mode.
EVENTO	O		Event output. This signal is active when one SEV instruction is executed.
STANDBYWFE[N:0]	O	Power controller	Indicates if a Cortex-A9 processor is in WFE Standby mode. **0** Processor not in WFE Standby mode. **1** Processor in WFE Standby mode.
STANDBYWFI[N:0]	O		Indicates that a Cortex-A9 processor is in WFI Standby mode. **0** Processor not in WFI Standby mode. **1** Processor in WFI Standby mode.

Related concepts

5.3.1 Individual Cortex-A9 processor power management on page 5-81.

A.7 Power management signals

List of power control interface signals.

Table A-9 Power control interface signals

Name	I/O	Source or Destination	Description
CPUCLAMP[N:0]	I	Power controller	Interrupt interface clamps control signals: **CPUCLAMP[3]** CPU3 interface. **CPUCLAMP[2]** CPU2 interface. **CPUCLAMP[1]** CPU1 interface. **CPUCLAMP[0]** CPU0 interface.
CPURAMCLAMP[N:0]	I		Enables the clamp cells in Dormant mode.
SCURAMCLAMP	I		Enables the SCU clamp cells in Dormant mode.
NEONCLAMP[N:0] [k]	I		Activates the Cortex-A9 MPE SIMD logic clamps: **0** Clamps not active. **1** Clamps active.
PWRCTLI0[1:0]	I		Reset value for CPU0 status field, bits [1:0] of SCU CPU Power Status Register.
PWRCTLI1[1:0]	I		Reset value for CPU1 status field, bits [9:8] of SCU CPU Power Status Register.
PWRCTLI2[1:0]	I		Reset value for CPU2 status field, bits [17:16] of SCU CPU Power Status Register.
PWRCTLI3[1:0]	I		Reset value for CPU3 status field, bits [25:24] of SCU CPU Power Status Register.
PWRCTLO0[1:0]	O		`0b0x` CPU0 must be powered on. `0b10` CPU0 can enter dormant mode. `0b11` CPU0 can enter powered-off mode.
PWRCTLO1[1:0]	O		`0b0x` CPU1 must be powered on. `0b10` CPU1 can enter dormant mode. `0b11` CPU1 can enter powered-off mode. This signal exists only if CPU1 is present.

[k] Only if an MPE is present.

Table A-9 Power control interface signals (continued)

Name	I/O	Source or Destination	Description
PWRCTLO2[1:0]	O	Power controller	`0b0x` CPU2 must be powered on. `0b10` CPU2 can enter dormant mode. `0b11` CPU2 can enter powered-off mode. This signal exists only if CPU2 is present.
PWRCTLO3[1:0]	O		`0b0x` CPU3 must be powered on. `0b10` CPU3 can enter dormant mode. `0b11` CPU3 can enter powered-off mode. This signal exists only if CPU3 is present.
SCUIDLE	O	L2C-310 or power controller	In the case of the L2C-310, the **SCUIDLE** output of the Cortex-A9 MPCore can be connected to the **STOPCLK** input of the L2C-310.

Related references

2.2.4 SCU CPU Power Status Register on page 2-29.
5.3.2 Communication to the Power Management Controller on page 5-83.

A.8 AXI interfaces

In Cortex-A9 designs there can be two AXI master ports and an Accelerator Coherence Port, an AXI slave.

This section contains the following subsections:

A.8.1 AXI Master0 signals

List of AXI Master0 interface signals with source and destination information for each signal.

Write address signals for AXI Master0

List of write address signals for AXI Master0.

Table A-10 Write address signals for AXI Master0

Name	I/O	Source or Destination	Description
AWADDRM0[31:0]	O	L2C-310 or other system AXI devices	Address.
AWBURSTM0[1:0]	O		Burst type Cortex-A9 processors can only issue INCR (BURST = 01) incrementing bursts. In the case of writes from the ACP, the burst type can also be FIXED (BURST = 00) or WRAP (BURST = 10) and these values can be forwarded onto the AXI Master0 port. Other values are Reserved.
AWCACHEM0[3:0]	O		Cache type giving additional information about cacheable characteristics set by the memory type and Outer cache policy.
AWIDM0[5:0]	O		Request ID See *AWIDMx[5:0] encodings* on page 2-38.

Table A-10 Write address signals for AXI Master0 (continued)

Name	I/O	Source or Destination	Description
AWLENM0[3:0]	O	L2C-310 or other system AXI devices	The number of data transfers that can occur within each burst.
AWLOCKM0[1:0]	O		Lock type.
AWPROTM0[2:0]	O		Protection Type.
AWREADYM0	I		Address ready.
AWSIZEM0[1:0]	O		Burst size: `0b00` 8-bit transfer. `0b01` 16-bit transfer. `0b10` 32-bit transfer. `0b11` 64-bit transfer.
AWUSERM0[8:0]	O		[8] early **BRESP**. Used with the L2C-310. [7] full line of zeros. Used with the L2C-310. [6] clean eviction. [5] level 1 eviction. [4:1] Memory type and inner cache policy. See *AWUSERMx[8:0] encodings* on page 2-39. [0] shared.
AWVALIDM0	O		Address valid.

Write data channel signals

List of write data signals for AXI Master0.

Table A-11 Write data signals for AXI Master0

Name	I/O	Source or destination	Description
WDATAM0[63:0]	O	L2C-310 or other system AXI devices	Write data
WIDM0[5:0]	O		Write ID
WLASTM0	O		Write last indication
WREADYM0	I		Write ready
WSTRBM0[7:0]	O		Write byte lane strobe
WVALIDM0	O		Write valid

Write response channel signals

List of write response signals for AXI Master0.

Table A-12 Write response signals for AXI Master0

Name	I/O	Source or destination	Description
BIDM0[5:0]	I	L2C-310 or other system AXI devices	Response ID
BREADYM0	O		Response ready
BRESPM0[1:0]	I		Write response
BVALIDM0	I		Response valid

Read address signals

List of read address signals for AXI Master0.

Table A-13 Read address signals for AXI Master0

Name	I/O	Source or destination	Description
ARADDRM0[31:0]	O	L2C-310 or other system AXI devices	Address
ARBURSTM0[1:0]	O		Burst type: Cortex-A9 processors can only issue one of the two following AXI burst types: `0b01` INCR incrementing burst `0b10` WRAP Wrapping burst. In the case of reads from the ACP, the burst type can also be FIXED (BURST = 00) and this value can be forwarded onto the AXI Master0 port. Other values are Reserved.
ARCACHEM0[3:0]	O		Cache type giving additional information about cacheable characteristics.
ARIDM0[5:0]	O		Request ID See *ARIDMx[5:0] encodings* on page 2-37.
ARLENM0[3:0]	O		Burst length that gives the exact number of transfers.
ARLOCKM0[1:0]	O		Lock type.
ARPROTM0[2:0]	O		Protection Type
ARREADYM0	I		Address ready.

Table A-13 Read address signals for AXI Master0 (continued)

Name	I/O	Source or destination	Description
ARSIZEM0[1:0]	O	L2C-310 or other system AXI devices	Burst size: `0b00` 8-bit transfer. `0b01` 16-bit transfer. `0b10` 32-bit transfer. `0b11` 64-bit transfer.
ARUSERM0[6:0]	O		Sideband information: [6] Speculative linefill, used with L2C-310 [5] prefetch hint, used with L2C-310 [4:1] inner attributes: `0b0000` Strongly-ordered. `0b0001` Device. `0b0011` Normal Memory Non-Cacheable. `0b0110` Write-Through. `0b0111` Write-Back no Write Allocate. `0b1111` Write-Back Write Allocate. [0] shared bit. See *ARUSERMx[6:0] encodings* on page 2-39.
ARVALIDM0	O		Address valid.

Speculative read interface signals for M0

List of the interface signals on M0 for speculative read accesses between Cortex-A9MPCore and L2C-310.

Table A-14 L2C-310 signals on M0

Name	I/O	Source	Description
SRENDM0[3:0]	I	L2C-310	Speculative linefill confirmations from L2C-310.
SRIDM0[23:0]	I		Speculative confirmed IDs from L2C-310

Read data channel signals

List of read data signals for AXI Master0.

Table A-15 Read data signals for AXI Master0

Name	I/O	Source or destination	Description
RVALIDM0	I	L2C-310 or other system AXI devices	Read valid
RDATAM0[63:0]	I		Read data
RRESPM0[1:0]	I		Read response
RLASTM0	I		Read Last indication
RIDM0[5:0]	I		Read ID
RREADYM0	O		Read ready

AXI Master0 Clock enable signals

List of AXI Master0 clock enable signals.

Table A-16 AXI Master0 clock enable signals

Name	I/O	Source	Description
INCLKENM0	I	Clock controller	Clock enable for the AXI bus that enables the AXI interface to operate at either: • Integer ratios of the system clock. • Half integer ratios of the system clock. See *1.6 Interfaces* on page 1-19.
OUTCLKENM0	I		Clock enable for the AXI bus that enables the AXI interface to operate at either: • Integer ratios of the system clock. • Half integer ratios of the system clock. See *1.6 Interfaces* on page 1-19.

A.8.2 AXI Master1 signals

In designs that implement the AXI Master1 interface, the AXI Master1 interface signals are identical to the AXI Master0 interface signals, except that AXI Master1 signals end in **M1**.This applies to all M0 AXI signals in addition to the Speculative Read Interface signals **SREND** and **SRID**.

A.8.3 AXI ACP signals

List of AXI ACP interface signals with source and destination information for each signal.

Write address signals for AXI ACP

List of AXI write address signals for AXI ACP.

Table A-17 Write address signals for AXI ACP

Name	I/O	Source or destination	Description
AWADDRS[31:0]	I	External AXI master	Address.
AWBURSTS[1:0]	I		Burst type.
AWCACHES[3:0]	I		Cache type giving additional information about cacheable characteristics.
AWIDS[2:0]	I		Request ID
AWLENS[3:0]	I		The number of data transfers that can occur within each burst.
AWLOCKS[0]	I		Lock type: **0** Normal access. **1** Exclusive access. Bit [1] is unused. Tie off LOW.
AWPROTS[2:0]	I		Protection Type.
AWREADYS	O		Address ready.
AWSIZES[1:0]	I	External AXI master	Burst size: `0b00` 8-bit transfer. `0b01` 16-bit transfer. `0b10` 32-bit transfer. `0b11` 64-bit transfer.
AWUSERS[4:0]	I		Sideband information: [4:1] inner attributes: `0b0000` Strongly-ordered. `0b0001` Device. `0b0011` Normal Memory Non-Cacheable. `0b0110` Write-Through. `0b0111` Write-Back no Write Allocate. `0b1111` Write-Back Write Allocate. [0] shared. See *2.3.4 AXI USER attributes encodings* on page 2-38.
AWVALIDS	I		Address valid.

Write data channel signals

List of AXI write data signals for AXI ACP.

Table A-18 Write data signals for AXI ACP

Name	I/O	Source or destination	Description
WDATAS[63:0]	I	External AXI master	Write data
WIDS[2:0]	I		Write ID
WLASTS	I		Write last indication
WREADYS	O		Write ready
WSTRBS[7:0]	I		Write byte lane strobe
WVALIDS	I		Write valid

Write response channel signals

List of AXI write response signals for AXI ACP.

Table A-19 Write response signals for AXI ACP

Name	I/O	Source or destination	Description
BIDS[2:0]	O	External AXI master	Response ID
BREADYS	I		Response ready
BRESPS[1:0]	O		Write response
BVALIDS	O		Response valid

Read address channel signals

List of AXI read address signals for AXI ACP.

Table A-20 Read address signals for AXI ACP

Name	I/O	Source or destination	Description
ARADDRS[31:0]	I	External AXI master	Address.
ARBURSTS[1:0]	I		Burst type.
ARCACHES[3:0]	I		Cache type giving additional information about cacheable characteristics.
ARIDS[2:0]	I		Request ID
ARLENS[3:0]	I		The number of data transfers that can occur within each burst.

Table A-20 Read address signals for AXI ACP (continued)

Name	I/O	Source or destination	Description
ARLOCKS[1:0]	I	External AXI master	Lock type.
ARPROTS[2:0]	I		Protection Type
ARREADYS	O		Address ready
ARSIZES[1:0]	I		Burst size: `0b00` 8-bit transfer. `0b01` 16-bit transfer. `0b10` 32-bit transfer. `0b11` 64-bit transfer.
ARUSERS[4:0]	I		Sideband information: [4:1] Inner attribute bits: `0b0000` Strongly-ordered. `0b0001` Device. `0b0011` Normal Memory Non-Cacheable. `0b0110` Write-Through. `0b0111` Write-Back no Write Allocate. `0b1111` Write-Back Write Allocate. [0] shared bit. See *2.3.4 AXI USER attributes encodings* on page 2-38.
ARVALIDS	I		Address valid.

Read data channel signals

List of AXI read data signals for AXI ACP.

Table A-21 Read data signals for AXI ACP

Name	I/O	Source or destination	Description
RVALIDS	O	External AXI master	Read valid
RDATAS[63:0]	O		Read data
RRESPS[1:0]	O		Read response
RLASTS	O		Read Last indication
RIDS[2:0]	O		Read ID
RREADYS	I		Read ready

Clock enable slave signal

ACLKENS signal is the clock enable slave signal.

Table A-22 ACLKENS signal

Name	I/O	Source or destination	Description
ACLKENS	I	Clock controller	Bus clock enable. See *2.4.2 ACP interface clocking* on page 2-44.

A.9 Performance monitoring signals

List of performance monitoring signals. There are as many **PMUEVENT** buses as there are Cortex-A9 processors in the design.

Table A-23 Performance monitoring signals

Name	I/O	Destination	Description
PMUEVENTn[57:0]	O	*Performance Monitoring Unit* (PMU) or External Performance Monitoring Unit	Performance Monitoring Unit event bus for CPUn. The *Cortex®-A9 Technical Reference Manual* describes the signals and events.
PMUIRQ[N:0]	O	System Integrity Controller or External Performance Monitoring unit	Interrupt requests by system metrics, one per Cortex-A9 processor.
PMUSECURE[N:0]	O	External Performance Monitoring unit	Gives the security status of the Cortex-A9 processor: **0** In Non-secure state. **1** In Secure state. This signal does not provide input to the CoreSight Trace delivery infrastructure.
PMUPRIV[N:0]	O		Gives the status of the Cortex-A9 processor: **0** In user mode. **1** In privileged mode. This signal does not provide input to CoreSight Trace delivery infrastructure.

A.10 Exception flags signals

DEFLAGS and **SCUEVABORT** signals are the Exception flag signals.

Table A-24 Exception flags signals

Name	I/O	Destination	Description
DEFLAGSn[6:0]	O	System integrity controller	Data Engine output flags. Only implemented if the Cortex-A9 processor includes a Data Engine. If the DE is NEON SIMD unit: • Bit[6] gives the value of FPSCR[27] • Bit[5] gives the value of FPSCR[7] • Bits[4:0] give the value of FPSCR[4;0]. If the DE is FPU: • Bit[6] is zero. • Bit[5] gives the value of FPSCR[7] • Bits[4:0] give the value of FPSCR[4;0].
SCUEVABORT	O		Indicates an external abort has occurred during a coherency writeback. **SCUEVABORT** is a pulse signal that is asserted for one CLK clock cycle.

For additional information on the FPSCR, see the *ARM® Cortex®-A9 Floating-Point Unit (FPU) Technical Reference Manual* and the *ARM® Cortex®-A9 NEON™ Media Processing Engine Technical Reference Manual*.

A.11 Parity error signals

List of parity error reporting signals. These signals are present only if parity is defined.

The number of sets of **PARITYFAIL** signals corresponds to the number of Cortex-A9 processors present in the design.

Table A-25 Error reporting signals

Name	I/O	Destination	Description
PARITYFAILn[7:0]	O	System integrity controller	Parity output pin from the RAM array for Cortex-A9 processor n. Indicates a parity fail: **0** No parity fail. **1** Parity fail. Bit [7] BTAC parity error. Bit [6] GHB parity error. Bit [5] Instruction tag RAM parity error. Bit [4] Instruction data RAM parity error. Bit [3] Main TLB parity error. Bit [2] D outer RAM parity error. Bit [1] Data tag RAM parity error. Bit [0] Data data RAM parity error. **PARITYFAILn** are pulse signals that are asserted for one CLK clock cycle.
PARITYFAILSCU[N:0]	O		Parity output pin from the SCU tag RAMs. ORed output from each Cortex-A9 processor present in the design. **PARITYSCU** are pulse signals that are asserted for one CLK clock cycle.

A.12 MBIST interface

The MBIST interface signals can be of two types: signals with parity support implemented and signals without parity support implemented.

This section contains the following subsections:

A.12.1 MBIST interface signals

List of MBIST interface signals.

Table A-26 MBIST interface signals

Name	I/O	Source	Description
MBISTADDR[10:0]	I	MBIST controller	MBIST address.
MBISTARRAY[19:0]	I		MBIST arrays used for testing RAMs.
MBISTENABLE	I		Activates MBIST mode.
MBISTWRITEEN	I		Global write enable.
MBISTREADEN	I		Global read enable.

A.12.2 MBIST interface signals with parity support

List of MBIST signals with parity support implemented. The size of some MBIST signals depends on whether the implementation has parity support or not.

Table A-27 MBIST signals with parity support implemented

Name	I/O	Source or destination	Description
MBISTDE[63:0]	I	MBIST controller	MBIST write enable.
MBISTINDATA[71:0]	I		MBIST data in.
MBISTOUTDATA[287:0]	O		MBIST data out.

A.12.3 MBIST interface signals without parity

List of MBIST signals without parity support implemented. The size of some MBIST signals depends on whether the implementation has parity support or not.

Table A-28 MBIST signals without parity support implemented

Name	I/O	Source or destination	Description
MBISTDE[63:0]	I	MBIST controller	MBIST write enable.
MBISTINDATA[63:0]	I		MBIST data in.
MBISTOUTDATA[255:0]	O		MBIST data out.

See *ARM® Cortex®-A9 MBIST Controller Technical Reference Manual*.

A.13 Scan test signal

List of scan test signal.

Table A-29 Scan test signal

Name	I/O	Destination	Description	
SE	I	DFT controller	Scan enable:	
			0	Not enabled.
			1	Enabled.

A.14 External Debug interface

List of external debug interface signals with source and destination information.

This section contains the following subsections:

A.14.1 Authentication interface

List of authentication interface signals. The value of **N** is one less than the number of processors in your design.

Table A-30 Authentication interface signals

Name	I/O	Source	Description
DBGEN[N:0]	I	Security controller	Invasive debug enable: **0** Not enabled. **1** Enabled.
NIDEN[N:0]	I		Noninvasive debug enable: **0** Not enabled. **1** Enabled.
SPIDEN[N:0]	I		Secure privileged invasive debug enable: **0** Not enabled. **1** Enabled.
SPNIDEN[N:0]	I		Secure privileged noninvasive debug enable: **0** Not enabled. **1** Enabled.

A.14.2 APB interface signals

List of APB interface signals.

Table A-31 APB interface signals

Name	I/O	Source or destination	Description
PADDRDBG[x:2]	I	CoreSight APB device	Programming address. The width of x:2 depends on the configuration: **[12:2]** A uniprocessor or multiprocessor configuration with a single Cortex-A9 processor. **[13:2]** A multiprocessor configuration with two Cortex-A9 processors. **[14:2]** A multiprocessor configuration with three or four Cortex-A9 processors.
PADDRDBG31	I		APB address bus bit [31]: **0** Not an external debugger access. **1** External debugger access.
PENABLEDBG	I		Indicates a second and subsequent cycle of a transfer.
PSELDBG	I		Selects the external debug interface: **0** Debug registers not selected. **1** Debug registers selected.
PWDATADBG[31:0]	I		Write data bus.
PWRITEDBG	I		APB read and write signal.
PRDATADBG[31:0]	O		Read data bus
PREADYDBG	O		Used to extend a transfer by inserting wait states APB slave ready. An APB slave can assert **PREADY** to extend a transfer.
PSLVERRDBG	O		APB slave transfer error: **0** No transfer error. **1** Transfer error.

A.14.3 Cross trigger interface signals

List of CTI signals. The value of **N** is one less than the number of processors in your design.

Table A-32 Cross trigger interface signals

Name	I/O	Source or destination	Description
EDBGRQ[N:0]	I	External debugger or CoreSight interconnect	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**.
DBGACK[N:0]	O		Debug acknowledge signal
DBGCPUDONE[N:0]	O		Debug acknowledge signal **0** Not enabled. **1** Enabled.
DBGRESTART[N:0]	I		Causes the core to exit from Debug state. It must be held HIGH until **DBGRESTARTED** is deasserted. **0** Not enabled. **1** Enabled.
DBGRESTARTED[N:0]	O		Used with **DBGRESTART** to move between Debug state and Normal state. **0** Not enabled. **1** Enabled.

A.14.4 Miscellaneous debug interface signals

List of miscellaneous debug interface signals. The value of **N** is one less than the number of processors in your design.

Table A-33 Miscellaneous debug signals

Name	I/O	Source or destination	Description
COMMRX[N:0]	O	External debugger or CoreSight Interconnect	Comms Channels Receive. Receive portion of Data Transfer Register full flag: **0** Empty. **1** Full.
COMMTX[N:0]	O		Comms Channels Transmit. Transmit portion of Data Transfer Register full flag: **0** Empty. **1** Full.
DBGNOPWRDWN[N:0]	O		Debugger has requested a Cortex-A9 processor is not powered down.
DBGSWENABLE[N:0]	I		When LOW only the external debug agent can modify debug registers. **0** Not enabled. **1** Enabled. Access by the software through the extended cp14 interface is permitted. External cp14 and external debug accesses are permitted.
DBGROMADDR[31:12]	I	CoreSight System configuration	Specifies bits [31:12] of the ROM table physical address. If the address cannot be determined tie off this signal to zero.
DBGROMADDRV	I		Valid signal for **DBGROMADDR**. If the address cannot be determined tie this signal LOW.
DBGSELFADDR[31:15]	I		Specifies bits [31:15] 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 off this signal to zero.
DBGSELFADDRV	I		Valid signal for **DBGSELFADDR**. If the offset cannot be determined tie this signal LOW.

A.15 PTM interface signals

List of PTM interface signals. There can be as many PTM interface signal buses as there are Cortex-A9 processors in the design.

Table A-34 PTM interface signals

Name	I/O	Source or destination	Description
WPTFIFOEMPTYn	O	PTM device	There are no speculative waypoints in the PTM interface FIFO.
WPTCOMMITn[1:0]	O		Number of waypoints committed this cycle. It is valid to indicate a valid waypoint and commit it in the same cycle.
WPTCONTEXTIDn[31:0]	O		Context ID for the waypoint. This signal must be true regardless of the condition code of the waypoint.
WPTENABLEn	I		Enable waypoint. When set, enables the Cortex-A9 processor to output waypoints.
WPTEXCEPTIONTYPEn[3:0]	O		Exception type: `0b0001` Halting Debug. `0b0010` Secure Monitor. `0b0100` Imprecise Data Abort. `0b0101` T2EE trap. `0b1000` Reset. `0b1001` UNDEF. `0b1010` SVC. `0b1011` Prefetch abort/Software Breakpoint. `0b1100` Precise data abort/software watchpoint. `0b1110` IRQ. `0b1111` FIQ.
WPTFLUSHn	O		Flush signal from core exception FIFO. All as yet uncommitted waypoints are flushed.
WPTLINKn	O		The waypoint is a branch and updates the link register. Only HIGH if **WPTTYPE[2:0]** is a direct branch or an indirect branch.

Table A-34 PTM interface signals (continued)

Name	I/O	Source or destination	Description
WPTnSECUREn	O	PTM device	Instructions following the waypoint are executed in Non-secure state. An instruction is in Non-secure state if the NS bit is set and the processor is not in secure monitor mode.
WPTPCn [31:0]	O		Waypoint last executed address indicator. This is the base LR in the case of an exception. Must be 0 for a reset exception, when it must not be traced.Equal to 0 if the waypoint is reset exception.
WPTT32LINKn	O		Indicates the size of the last executed address when in Thumb state: **0** 16-bit instruction. **1** 32-bit instruction.
WPTTAKENn	O		The waypoint passed its condition codes. The address is still used, irrespective of the value of this signal. Must be set for all waypoints except branch.
WPTTARGETJBITn	O		J bit for waypoint destination. This signal is LOW if **WPTTRACEPROHIBITED** is asserted.
WPTTARGETPCn[31:0]	O		Waypoint target address: • Bit [1] must be zero if T-bit is zero. • Bit [0] must be zero if J-bit is zero. The value is zero if **WPTTYPE** is either prohibit or debug.
WPTTARGETTBITn	O		T bit for waypoint destination This signal is LOW if **WPTTRACEPROHIBITED** is asserted.

Table A-34 PTM interface signals (continued)

Name	I/O	Source or destination	Description
WPTTRACEPROHIBITEDn	O	PTM device	Trace is prohibited for the waypoint target. Indicates entry to prohibited region. No more waypoints are traced until trace can resume. Indication that PTM clocks can be stopped. This signal must be permanently asserted if **NIDEN** and **DBGEN** are both LOW, after the in-flight waypoints have exited the core. Either an exception or a serial branch is required to ensure that changes to the inputs have been sampled. Only one **WPTVALID** cycle can be seen with **WPTTRACEPROHIBITED** set. Trace stops with this waypoint and the next waypoint seen is an Isync packet.
WPTTYPEn[2:0]	O		Waypoint Type. `0b000` Direct Branch. `0b001` Indirect Branch. `0b010` Exception. `0b011` DMB. `0b100` Debug entry/Trace prohibited. `0b101` Debug exit, requires addresses of first instruction. `0b110` Invalid. `0b111` Invalid. Must only take valid states when **WPTVALID** is HIGH. Debug Entry must be followed by Debug Exit. **Note** Debug exit does not reflect the execution of an instruction.
WPTVALIDn	O		Waypoint is confirmed as valid.

Appendix B
Revisions

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

It contains the following sections:

- *B.1 Revisions* on page Appx-B-123.

B.1 Revisions

Changes between each issue of this book.

Table B-1 Issue A

Change	Location
First release	-

Table B-2 Differences between issue A and issue B

Change	Location
Clarify the relationship between the GIC (PL390) and the Cortex-A9 Interrupt Controller	*Chapter 3 Interrupt Controller* on page 3-47.
Parity error option added	*1.3 Configurable options* on page 1-15.
Clarify the role of the SCU with reference to data coherency and the non-support of instruction cache coherency	*2.1 About the SCU* on page 2-25.
Added information about exclusive accesses and address filtering	*2.2.2 SCU Control Register* on page 2-27
SSAC description corrected	*2.2.9 SCU Non-secure Access Control Register* on page 2-34.
SSAC bit assignments corrected	*2.2.9 SCU Non-secure Access Control Register* on page 2-34.
Change STI, Software Triggered Interrupt, to SGI, Software Generated Interrupt	Throughout *Chapter 3 Interrupt Controller* on page 3-47.
INTID descriptions extended and clarified	Throughout *Chapter 3 Interrupt Controller* on page 3-47.
Reset information added	*4.2.1 Private timer and watchdog register summary* on page 4-64
AXI transaction IDs section extended	*2.3.3 AXI transaction IDs* on page 2-37
AXI USER encodings section added	*2.3.4 AXI USER attributes encodings* on page 2-38
EVENTI information extended and **EVENTO** information added	*2.5 Event communication with an external agent using WFE/SEV* on page 2-46
CLUSTERID[3:0] description corrected	*A.4 Configuration signals* on page Appx-A-95.
DBGEN[3:0] description added	*A.14.1 Authentication interface* on page Appx-A-115.

Differences between issue B and issue C

Table B-3 Differences between issue B and issue C

Change	Location
Design changes listed	*1.9 Product revisions* on page 1-23.
New entries in the Private Memory map	*4.2 Private timer and watchdog registers* on page 4-64
Timers and watchdogs renamed Private timers and watchdogs	*4.2 Private timer and watchdog registers* on page 4-64
TLB size added as a configurable option	*1.3 Configurable options* on page 1-15.
Timing diagrams added	*2.3.7 AXI master interface clocking* on page 2-41
CPUCLKOFF and **DECLKOFF** added to Power-on reset	*5.2.2 Cortex-A9 MPCore power-on reset* on page 5-77 and *A.4 Configuration signals* on page Appx-A-95.

Table B-3 Differences between issue B and issue C (continued)

Change	Location
Correction to Tag RAM sizes values	*2.2.3 SCU Configuration Register* on page 2-28
Change in SCU Power Status Register layout	*2.2.4 SCU CPU Power Status Register* on page 2-29
Additional PPI. There are five PPIs per Cortex-A9 processor interface	*3.1.2 Interrupt Distributor interrupt sources* on page 3-48
PPI(4) added to the PPI Status Register	*3.3.9 PPI Status Register* on page 3-56.
INT renamed **IRQS**	*3.3.10 SPI Status Registers* on page 3-57. *A.3 Interrupts* on page Appx-A-94.
Chapter 5 renamed. It was “Private timers and Watchdog Registers”.	*Chapter 4 Global timer, private timers, and watchdog registers* on page 4-62.
L2 interface chapter included in Chapter 1	
nIRQOUT[N:0] and **nFIQOUT[N:0]** added	*A.3 Interrupts* on page Appx-A-94.
MAXCLKLATENCY[2:0] added	*A.4 Configuration signals* on page Appx-A-95.
BISTCLAMP removed	*A.7 Power management signals* on page Appx-A-99.
AXI descriptions corrected and extended	*A.8 AXI interfaces* on page Appx-A-101.
AXI Master1 descriptions removed.	
AWLOCKS[1:0] corrected to **AWLOCKS[0]**.	*Write address signals for AXI ACP* on page Appx-A-105.
ARIDS[5:0] corrected to **ARIDS[2:0]**.	*Read address channel signals* on page Appx-A-107.
Performance monitoring signals extended and new signals added.	*A.9 Performance monitoring signals* on page Appx-A-110.
SCUEVABORT moved to Performance Monitoring from Parity error signals section.	*A.9 Performance monitoring signals* on page Appx-A-110.
SCANMODE removed	*A.13 Scan test signal* on page Appx-A-114.
PRDATADBG corrected to **PRDATADBG[31:0]**	*A.14.2 APB interface signals* on page Appx-A-115.
WPTT32nT16n changed to WPT32LINKn	*A.15 PTM interface signals* on page Appx-A-119.

Differences between issue C and issue D.

Table B-4 Differences between issue C and issue D

Change	Location
Global timer re-positioned. Other timers re-named private timers.	*1.1.2 Example configuration* on page 1-12
Table 1-1 AXI master interface attributes moved	*2.3.1 AXI issuing capabilities* on page 2-36
Table 1-2 ARID encodings moved	*2.3.3 AXI transaction IDs* on page 2-37
Table 1-3 AWIDMx encodings moved	*2.3.3 AXI transaction IDs* on page 2-37
Compliance content moved and extended	*1.7.1 About Cortex-A9 MPCore coherency* on page 1-20
Features list removed	-
Configurable options includes Preload Engine options and ARM_BIST	*1.3 Configurable options* on page 1-15

Table B-4 Differences between issue C and issue D (continued)

Change	Location
Interfaces section extended	*1.6 Interfaces* on page 1-19
Private Memory Region chapter removed	-
Private Memory Region content re-arranged. Table added	*1.5 Private Memory Region* on page 1-17
SLVERR changed to DECERR	*1.5 Private Memory Region* on page 1-17
Interfaces section extended	*1.6 Interfaces* on page 1-19
MPCore Considerations section added	*1.7 MPCore considerations* on page 1-20
Table 1-4 **ARUSERMx[6:0]**moved	*ARUSERMx[6:0] encodings* on page 2-39
Table 1-5 **AWUSERMx[8:0]** encodings moved	*AWUSERMx[8:0] encodings* on page 2-39
Table 1-6 Core mode and APROT values removed	-
Figure 1-2 moved	*6.1 External Debug Interface signals* on page 6-87
Figure 1-3 Three-to-one timing ratio moved	*5.1 Clocks* on page 5-76
Figure 1-4 moved	*Timing diagram for INCLKEN with three-to-two ratio* on page 2-41
Figure 1-5 moved	*Timing diagram for INCLKEN with five-to-two ratio* on page 2-41
Figure 1-6 moved	*Timing diagram for OUTCLKEN with three-to-two ratio* on page 2-41
Figure 1-7 moved	*Timing diagram for OUTCLKEN with five-to-two ratio* on page 2-42
Figure 1-8 moved	*Timing diagram for OUTCLKEN with five-to-two ratio* on page 2-42
Figure 1-9 moved and renamed	*5.3.3 Cortex-A9 MPCore power domains* on page 5-83
Table 1-7 Configurable options moved	*1.3 Configurable options* on page 1-15
Table 1-8 PADDRDBG width replaced and extended	*6.2 Cortex-A9 MPCore APB Debug interface and memory map* on page 6-88
Table 1-9 Cortex-A9 MPCore reset signals moved	*5.2.1 Reset combinations* on page 5-77
Table 1-10 Cortex-A9 MPCore power modes moved	*5.3.1 Individual Cortex-A9 processor power management* on page 5-81
Table 2-1 Cortex-A9 MPCore memory region moved	*1.5 Private Memory Region* on page 1-17
ACP behavior description moved and extended	*2.4 Accelerator Coherency Port* on page 2-43
Design changes list extended	*1.9 Product revisions* on page 1-23.
Snoop Control Unit chapter updated and extended to include detailed interface descriptions	*Chapter 2 Snoop Control Unit* on page 2-24
SCU Register updates	*2.2.1 SCU register summary* on page 2-26
Interfaces	*2.2.2 SCU Control Register* on page 2-27
Table 3-1 SCU registers summary moved and corrected	*2.2.1 SCU register summary* on page 2-26
Table 3-2 moved and retitled	*2.2.2 SCU Control Register* on page 2-27
Figure 3-1 SCU Control Register format moved and retitled	*2.2.2 SCU Control Register* on page 2-27
Table 3-3 moved and retitled	*2.2.3 SCU Configuration Register* on page 2-28
Figure 3-2 moved and retitled	*2.2.3 SCU Configuration Register* on page 2-28

Table B-4 Differences between issue C and issue D (continued)

Change	Location
Table 3-4 moved and retitled	*2.2.4 SCU CPU Power Status Register* on page 2-29
Figure 3-3 moved and retitled	*2.2.4 SCU CPU Power Status Register* on page 2-29
Table 3-5 moved and retitled	*2.2.5 SCU Invalidate All Registers in Secure State Register* on page 2-31
Figure 3-4 SCU Invalidate All Registers in Non-secure state format removed	-
Table 3-5 removed	-
Figure 3-5 SCU Invalidate All Registers in Secure state format moved	*2.2.5 SCU Invalidate All Registers in Secure State Register* on page 2-31
Table 3-6 moved	*2.2.5 SCU Invalidate All Registers in Secure State Register* on page 2-31
Figure 3-6 moved	*2.2.6 Filtering Start Address Register* on page 2-31
Table 3-7 moved	*2.2.6 Filtering Start Address Register* on page 2-31
Figure 3-7 moved	*2.2.7 Filtering End Address Register* on page 2-32
Table 3-8 moved	*2.2.7 Filtering End Address Register* on page 2-32
Figure 3-8 moved	*2.2.8 SCU Access Control Register (SAC)* on page 2-33
Table 3-9	*2.2.8 SCU Access Control Register (SAC)* on page 2-33
Figure 3-9 renamed and moved	*2.2.9 SCU Non-secure Access Control Register* on page 2-34
Table 3-10	*2.2.9 SCU Non-secure Access Control Register* on page 2-34
Removal of content that repeats GIC Architecture content	*Chapter 3 Interrupt Controller* on page 3-47
Re-organization of remaining *Interrupt Controller* content	
4.2 TrustZone® support renamed and specification content removed	*3.2 Security extensions support* on page 3-50
4.3 About the Interrupt Distributor removed	-
4.4 Interrupt Distributor interrupt sources removed	-
4.5 Cortex-A9 processor interfaces removed	-
Interrupt security registers removed	-
Enable set registers removed	-
Enable clear registers removed	-
Pending set registers removed	-
Pending clear registers removed	-
Active status registers removed	-
Interrupt Priority Registers removed	-
Interrupt Processor Targets Registers removed	-
Interrupt Configuration Registers removed	-
Software Generated Interrupt Register removed	-

Table B-4 Differences between issue C and issue D (continued)

Change	Location
CPU Interface Control Register removed	-
Interrupt Priority Mask Register removed	-
Binary Point Register removed	-
Interrupt Acknowledge Register removed	-
End Of Interrupt Register removed	-
Running Priority Register removed	-
Highest Pending Interrupt Register removed	-
Chapter 5 Timer and Watchdog Registers updated and corrected	*Chapter 4 Global timer, private timers, and watchdog registers* on page 4-62
5.1 About the timer and watchdog blocks renamed	*4.1 About the private timer and watchdog blocks* on page 4-63
Table 5-1 moved	*4.2.1 Private timer and watchdog register summary* on page 4-64
5.2 Timer and watchdog registers moved and renamed	*4.2.1 Private timer and watchdog register summary* on page 4-64
Note about private timer behavior added below Table 4-1	*4.2.1 Private timer and watchdog register summary* on page 4-64
Corrections to Timer Control Register section	*4.2.4 Private Timer Control Register* on page 4-65
Corrections to Timer Interrupt Status Register	*4.2.5 Private Timer Interrupt Status Register* on page 4-66
Clarification of behavior in relation to Interrupt ID 29	*4.2.5 Private Timer Interrupt Status Register* on page 4-66
Comparator Value Registers, 0x10 and 0x14 moved and corrected	*4.4.5 Comparator Value Registers, 0x10 and 0x14* on page 4-73
Auto-increment Register, 0x18 moved and corrected	*4.4.6 Auto-increment Register, 0x18* on page 4-73
5.3 About the Global Timer moved and corrected	*4.3 About the Global Timer* on page 4-70
Global Timer Control Register section added	*4.4.3 Global Timer Control Register* on page 4-72
Global Timer Interrupt Status Register added	*4.4.4 Global Timer Interrupt Status Register* on page 4-73
Resets descriptions revised and extended	*5.2 Resets* on page 5-77
Signals lists updated	Source or destination column added to all signal lists
nNEONRESET[N:0] replaces **nDERESET[N:0]**	*A.2 Resets and reset control signals* on page Appx-A-92
NEONCLCKOFF replaces DECLCKOFF	*A.4 Configuration signals* on page Appx-A-95
CPUCLCKOFF[N:0] replaces CPUCLOCKOFF[N:0]	
CP15 c15 Configuration Base Address Register replaces System Control Config base Register	
NEONCLAMP replaces DECLAMP	*A.7 Power management signals* on page Appx-A-99
Power control signal descriptions corrected and clarified.	
SCUIDLE signal added	
Duplicated AXI user encodings removed	*A.8.1 AXI Master0 signals* on page Appx-A-101
ARUSERM0[6:0] corrected	*Read address signals* on page Appx-A-103
Speculative read interface signals section added	*Speculative read interface signals for M0* on page Appx-A-104
[4:0] in **AWUSERS[4:0]** corrected to [4:1]	*Write address signals for AXI ACP* on page Appx-A-105

Table B-4 Differences between issue C and issue D (continued)

Change	Location
NEON SIMD unit replaces MPE	*A.9 Performance monitoring signals* on page Appx-A-110
PMUEVENT size becomes 57 bits	
DEFLAGS and **SCUEVABORT** have a separate table	*A.10 Exception flags signals* on page Appx-A-111
PARITYSCU[3:0] becomes PARITYFAILSCU[N:0]	*A.11 Parity error signals* on page Appx-A-112
MBISTBE[31:0] becomes MBISTBE[32:0]	*A.12.2 MBIST interface signals with parity support* on page Appx-A-113
Description of **DBGSWENABLE[N:0]** amended	*A.14.4 Miscellaneous debug interface signals* on page Appx-A-117
DBGSELFADDR bits corrected to [31:15]	
WPTCOMMITn bits corrected to [1:0]	*A.15 PTM interface signals* on page Appx-A-119
WPTENABLE corrected to **WPTENABLEn**	
WPT32LINKn corrected toWPTT32LINKn	
Statement about **WPTTARGETTBIT** removed	

Table B-5 Differences between D and F

Document title corrected to *AMBA® Level 2 Cache Controller (L2C-310) Technical Reference Manual*	Additional reading
PL310 corrected to L2C-310 throughout	-
Symmetric configurations corrected to uniform configurations	*1.1 About the Cortex-A9 MPCore processor* on page 1-12
Tag RAMs renamed to Cache line directory	*1.1.2 Example configuration* on page 1-12
Coherency description reworded for clarity	*1.7.1 About Cortex-A9 MPCore coherency* on page 1-20
SCU control register corrections	*2.2.2 SCU Control Register* on page 2-27
Values corrected	*2.3.1 AXI issuing capabilities* on page 2-36
Note about theoretical maximums added	*2.3.1 AXI issuing capabilities* on page 2-36
Corrections to INCR values	*2.3.2 Cortex-A9 MPCore AXI transactions* on page 2-37
Note about transactions added	
Data linefill buffer corrected	*2.3.3 AXI transaction IDs* on page 2-37
Clarification about ratios added	*2.3.7 AXI master interface clocking* on page 2-41
Removed incorrect cross references	*Chapter 3 Interrupt Controller* on page 3-47
Register names aligned with GIC Architecture names	*Chapter 3 Interrupt Controller* on page 3-47
Access description corrected	*3.1 About the Interrupt Controller* on page 3-48
Corrected information about interrupt sources	*3.1.2 Interrupt Distributor interrupt sources* on page 3-48
Paragraph about single processor designs moved	*3.3.7 Interrupt Processor Targets Registers* on page 3-56
Second line corrected	*3.3.1 Distributor register summary* on page 3-51
Interrupt Configuration Registers section added	*3.3.8 Interrupt Configuration Registers* on page 3-56
Values column added to Table 3-5	*3.3.4 Distributor Implementer Identification Register* on page 3-55

Table B-5 Differences between D and F (continued)

Inputs clarified.	*3.3.9 PPI Status Register* on page 3-56
Address offset sentence below Figure 3-6 removed	*3.3.10 SPI Status Registers* on page 3-57
PrimeCell Identification Registers section removed	-
Description of prescaler added to features list	*4.1 About the private timer and watchdog blocks* on page 4-63
PERIPHCLK added as reference clock	*4.2.4 Private Timer Control Register* on page 4-65
Global timer behavior feature added	*4.3 About the Global Timer* on page 4-70
Comparator register offsets added	*4.4.5 Comparator Value Registers, 0x10 and 0x14* on page 4-73
No asynchronous interfaces information added	*5.1 Clocks* on page 5-76
Reset descriptions expanded and clarified	*5.2 Resets* on page 5-77
IEM section removed	*5.3 Power management* on page 5-81
Rewritten and extended	*Standby modes* on page 5-82
WFI replaced by Standby	*5.3 Power management* on page 5-81
Lead processor replaced by primary processor	*5.3.4 About multiprocessor bring-up* on page 5-84
Missing **[N:0]** added to signal names	*A.2 Resets and reset control signals* on page Appx-A-92
Signal descriptions corrected	*A.6 WFE and WFI Standby signals* on page Appx-A-98
	A.6 WFE and WFI Standby signals on page Appx-A-98
STATIC replaced by FIXED	*Write address signals for AXI Master0* on page Appx-A-101
AWBURSTM0[1:0] description expanded	
AWCACHEM0[3:0] description expanded	
AWLENM0[3:0] corrected, repeated AXI information removed	
AWLOCKM0[1:0] corrected, repeated AXI information removed	
AWUSERM0[8:0] description corrected	
ARBURSTM0[1:0] corrected and expanded	*Read address signals* on page Appx-A-103
ARLENM0[3:0] corrected, repeated AXI information removed	*Read address signals* on page Appx-A-103
ARLOCKM0[1:0] corrected, repeated AXI information removed	*Read address signals* on page Appx-A-103
AWBURSTS[1:0] corrected, repeated AXI information removed	*Write address signals for AXI ACP* on page Appx-A-105
AWLENS[3:0] description expanded and corrected	*Write address signals for AXI ACP* on page Appx-A-105
STATIC replaced by FIXED	*Read address channel signals* on page Appx-A-107
ARBURSTS[1:0] corrected, repeated AXI information removed	
ARLENS[3:0] corrected, repeated AXI information removed	
ARLOCKS[1:0] corrected, repeated AXI information removed	
SCUEVABORT description corrected	*A.10 Exception flags signals* on page Appx-A-111

Table B-6 Differences between issue F and issue G

Change	Location	Affects
Correct section title for Read address signals	*Read address signals* on page Appx-A-103	All releases
ACP interface clocking moved to Accelerator Coherency Port Section	*2.4.2 ACP interface clocking* on page 2-44-	All releases
Correct description of Read address channel	*Read address channel signals* on page Appx-A-107	All releases
Correct description of Clock enable slave signal	*Clock enable slave signal* on page Appx-A-108	All releases
Power management standby modes section updated	*Standby modes* on page 5-82	All releases
Update interrupt controller behavior	*3.1.4 Cortex-A9 MPCore 1-N interrupt model handling* on page 3-49	All releases
Update AXI master interface timing diagrams	*2.3.7 AXI master interface clocking* on page 2-41	All releases
Updated SCU register summary table to include security state	*2.2.1 SCU register summary* on page 2-26	All releases
Updated information about Tag RAM sizes	*2.2.3 SCU Configuration Register* on page 2-28	All releases
Updated description of SCU Invalidate All Registers in Secure State	*2.2.5 SCU Invalidate All Registers in Secure State Register* on page 2-31	All releases
Updated description of SAC and SNSAC	*2.2.8 SCU Access Control Register (SAC)* on page 2-33 and *2.2.9 SCU Non-secure Access Control Register* on page 2-34	All releases
Updated description of ACP functional limitations	*ACP functional limitations* on page 2-44	All releases
Updated interrupt controller description	*3.1 About the Interrupt Controller* on page 3-48 *3.1.2 Interrupt Distributor interrupt sources* on page 3-48 *3.1.3 Interrupt Distributor arbitration* on page 3-49	All releases
Updated ICDDCR usage constraints	*3.3.2 Distributor Control Register* on page 3-52	All releases
Updated SCU CPU Power Status Register reset value	*2.2.4 SCU CPU Power Status Register* on page 2-29	All releases

Table B-7 Differences between issue G and issue H

Change	Location	Affects
Updated configuration options for TLB, BTAC and GHB sizes, and the number of entries in the Instruction micro TLB	*1.3 Configurable options* on page 1-15	r4p0
Updated Interrupt Priority Registers in Distributor register summary	*3.3.1 Distributor register summary* on page 3-51	All releases

Table B-8 Differences between issue H and issue I

Change	Location	Affects
Revision number changes only.	-	r4p1

Table B-9 Differences between issue I and issue 10

Change	Location	Affects
Binary number format changed.	Throughout	r4p1
MPE comment made into footnote.	*1.3 Configurable options* on page 1-15	r4p1
FPU comment made into footnote.	*1.3 Configurable options* on page 1-15	r4p1
Support for parity error detection comment made into footnote.	*1.3 Configurable options* on page 1-15	r4p1
PERIPHBASE[31:13] Reserved Description updated	*1.5 Private Memory Region* on page 1-17	r4p1
SCU enable parameter descriptions corrected.	*2.2.2 SCU Control Register* on page 2-27	r4p1
CPU3 comment made into footnote.	*2.2.8 SCU Access Control Register (SAC)* on page 2-33	r4p1
SNSAC register bit assignment labelling corrected	*2.2.8 SCU Access Control Register (SAC)* on page 2-33	r4p1
SNSAC register bit assignments CPU0 description, footnote added.	*2.2.8 SCU Access Control Register (SAC)* on page 2-33	r4p1
ARUSERMx[0] parameter description corrected	*2.3.4 AXI USER attributes encodings* on page 2-38	r4p1
ICDISRn Distributor Register Summary comment made into footnote.	*3.3.1 Distributor register summary* on page 3-51	r4p1
ICDICFRn Distributor Register Summary comment made footnote.	*3.3.1 Distributor register summary* on page 3-51	r4p1
Usage constraints description added.	*3.3.2 Distributor Control Register* on page 3-52	r4p1
ICDICTR IT lines number encoding `0b00000` comment made into footnote	*3.3.2 Distributor Control Register* on page 3-52	r4p1
Processor Interface Register summary `0x01C` footnote added.	*3.4.1 Processor interface register summary* on page 3-60	r4p1
Column added to show the registers that are banked.	*4.4.1 Global timer register summary* on page 4-71	r4p1
Auto Increment and Comp Enable bits fotnotes added.	*4.4.2 Global Timer Counter Registers, 0x00 and 0x04* on page 4-71	r4p1
Stop **CLK** and **PERIPHCLK** step added.	*5.2.2 Cortex-A9 MPCore power-on reset* on page 5-77	r4p1
Start **CLK** and **PERIPHCLK** step added.	*5.2.2 Cortex-A9 MPCore power-on reset* on page 5-77	r4p1
nIRQ[N:0] description updated.	*A.3 Interrupts* on page Appx-A-94	r4p1
nFIQ[N:0] description updated.	*A.3 Interrupts* on page Appx-A-94	r4p1
PWRCTL11[1:0], **PWRCTL12[N:0]** and **PWRCTL13[N:0]** descriptions corrected.	*A.7 Power management signals* on page Appx-A-99	r4p1
NEONCLAMP[N:0] comment made into footnote.	*A.7 Power management signals* on page Appx-A-99	r4p1
ABURSTM0[1:0] signal description corrected.	*A.8.1 AXI Master0 signals* on page Appx-A-101	r4p1
MBISTBE[32:0] signal corrected to **MBISTDE[63:0]**	*A.12.2 MBIST interface signals with parity support* on page Appx-A-113	r4p1
MBISTBE[25:0] signal corrected to **MBISTDE[63:0]**	*A.12.3 MBIST interface signals without parity* on page Appx-A-113	r4p1