

Cortex[™]-A9 MPCore

Revision: r2p0

Technical Reference Manual



Cortex-A9 MPCore

Technical Reference Manual

Copyright © 2008-2009 ARM. All rights reserved.

Release Information

The following changes have been made to this book.

Change history			
Date	Issue	Confidentiality	Change
04 April 2008	A	Non-Confidential	First release for r0p0
08 July 2008	B	Non-Confidential Restricted Access	First release for r0p1
16 December 2008	C	Non-Confidential Restricted Access	First release for r1p0
2 October 2009	D	Non-Confidential Restricted Access	First release for r2p0
27 November 2009	E	Non-Confidential	Second release for r2p0

Proprietary Notice

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

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

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

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

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

Confidentiality Status

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

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

Cortex-A9 MPCore Technical Reference Manual

Preface

About this manual	xiv
Additional reading	xvii
Feedback	xix

Chapter 1

Introduction

1.1	About the Cortex-A9 MPCore processor	1-2
1.2	Configurable options	1-4
1.3	Private Memory Region	1-5
1.4	Interfaces	1-7
1.5	MPCore considerations	1-8
1.6	Product revisions	1-10

Chapter 2

Snoop Control Unit

2.1	About the SCU	2-2
2.2	SCU registers	2-3
2.3	AMBA AXI Master Port Interfaces	2-15
2.4	Accelerator Coherency Port	2-24
2.5	Event communication with an external agent using WFE/SEV	2-27

Chapter 3	Interrupt Controller	
3.1	About the Interrupt Controller	3-2
3.2	TrustZone support	3-5
3.3	Interrupt distributor programmers model	3-7
3.4	Cortex-A9 processor interface register descriptions	3-25
Chapter 4	Global timer, Private timers, and Watchdog registers	
4.1	About the private timer and watchdog blocks	4-2
4.2	Private timer and watchdog registers	4-3
4.3	About the Global Timer	4-10
4.4	Global timer registers	4-11
Chapter 5	Clocks, Resets, and Power Management	
5.1	Clocks	5-2
5.2	Resets	5-3
5.3	Power management	5-6
Chapter 6	Debug	
6.1	External Debug Interface Signals	6-2
6.2	Cortex-A9 MPCore APB Debug interface and memory map	6-3
Appendix A	Signal Descriptions	
A.1	Clock and clock control signals	A-2
A.2	Resets and reset control signals	A-3
A.3	Interrupts	A-4
A.4	Configuration signals	A-5
A.5	Standby and wait for event signals	A-7
A.6	Power management signals	A-8
A.7	AXI interfaces	A-10
A.8	Performance monitoring signals	A-22
A.9	Exception flags signals	A-23
A.10	Parity error signals	A-24
A.11	MBIST interface	A-25
A.12	Scan test signal	A-26
A.13	External Debug interface	A-27
A.14	PTM interface signals	A-31
Appendix B	Revisions	
	Glossary	

List of Tables

Cortex-A9 MPCore Technical Reference Manual

	Change history	ii
Table 1-1	Configurable options for the Cortex-A9 MPCore processor	1-4
Table 1-2	Permitted access sizes for private memory regions	1-5
Table 1-3	Cortex-A9 MPCore private memory region	1-6
Table 2-1	SCU registers summary	2-3
Table 2-2	SCU Control Register bit assignments	2-4
Table 2-3	SCU Configuration Register bit assignments	2-6
Table 2-4	SCU CPU Power Status Register bit assignments	2-8
Table 2-5	SCU Invalidate All Registers in Secure state bit assignments	2-9
Table 2-6	Filtering Start Address Register bit assignments	2-10
Table 2-7	Filtering End Address Register bit assignments	2-11
Table 2-8	SCU Access Control Register bit assignments	2-12
Table 2-9	SCU Non-secure Access Control Register bit assignments	2-13
Table 2-10	AXI master interface attributes	2-15
Table 2-11	ARID encodings	2-17
Table 2-12	AWIDMx encodings	2-18
Table 2-13	ARUSERMx[6:0] encodings	2-19
Table 2-14	AWUSERMx[8:0] encodings	2-20
Table 3-1	Distributor register summary	3-10
Table 3-2	ICDDCR bit assignments for secure accesses	3-12
Table 3-3	ICDDCR bit assignments for Non-secure accesses	3-13
Table 3-4	ICDICTR bit assignments	3-14
Table 3-5	ICDDIR bit assignments	3-15

Table 3-6	ppi_status Register bit assignments	3-16
Table 3-7	spi_status Register bit assignments	3-17
Table 3-8	periph_id_[3:0] Register bit assignments	3-19
Table 3-9	periph_id_0 Register bit assignments	3-20
Table 3-10	periph_id_1 Register bit assignments	3-20
Table 3-11	periph_id_2 Register bit assignments	3-20
Table 3-12	periph_id_3 Register bit assignments	3-21
Table 3-13	periph_id_[7:4] Register bit assignments	3-21
Table 3-14	periph_id_4 Register bit assignments	3-22
Table 3-15	component_id Register bit assignments	3-22
Table 3-16	component_id_0 Register bit assignments	3-23
Table 3-17	component_id_1 Register bit assignments	3-24
Table 3-18	component_id_2 Register bit assignments	3-24
Table 3-19	component_id_3 Register bit assignments	3-24
Table 3-20	Cortex-A9 processor interface register summary	3-25
Table 3-21	ICPIIR Register bit assignments	3-26
Table 4-1	Timer and watchdog registers	4-3
Table 4-2	Private Timer Control Register bit assignments	4-4
Table 4-3	Watchdog Control Register bit assignments	4-7
Table 4-4	Global timer registers	4-11
Table 4-5	Global Timer Control Register bit assignments	4-12
Table 5-1	Reset combinations in a Cortex-A9 MPCore system	5-3
Table 5-2	Cortex-A9 MPCore power modes	5-6
Table A-1	Cortex-A9 MPCore clocks and clock control signals	A-2
Table A-2	Reset signals	A-3
Table A-3	Reset clock control signals	A-3
Table A-4	Watchdog request reset signal	A-3
Table A-5	Interrupt line signals	A-4
Table A-6	Configuration signals	A-5
Table A-7	Security control signals	A-6
Table A-8	Standby and wait for event signals	A-7
Table A-9	Power control interface signals	A-8
Table A-10	Write address signals for AXI Master0	A-10
Table A-11	Write data signals for AXI Master0	A-12
Table A-12	Write response signals for AXI Master0	A-12
Table A-13	Read address signals for AXI Master0	A-13
Table A-14	PL310 signals on M0	A-14
Table A-15	Read data signals for AXI Master0	A-15
Table A-16	AXI Master0 clock enable signals	A-15
Table A-17	Write address signals for AXI ACP	A-17
Table A-18	Write data signals for AXI ACP	A-18
Table A-19	Write response signals for AXI ACP	A-19
Table A-20	Read address signals for AXI ACP	A-19
Table A-21	Read data signals for AXI ACP	A-21
Table A-22	ACLKENS signal	A-21
Table A-23	Performance monitoring signals	A-22
Table A-24	Exception flags signals	A-23

Table A-25	Error reporting signals	A-24
Table A-26	MBIST interface signals	A-25
Table A-27	MBIST signals with parity support implemented	A-25
Table A-28	MBIST signals without parity support implemented	A-25
Table A-29	Scan test signal	A-26
Table A-30	Authentication interface signals	A-27
Table A-31	APB interface signals	A-28
Table A-32	Cross trigger interface signals	A-29
Table A-33	Miscellaneous debug signals	A-30
Table A-34	PTM interface signals	A-31
Table B-1	Issue A	B-1
Table B-2	Differences between issue A and issue B	B-1
Table B-3	Differences between issue B and issue C	B-2
Table B-4	Differences between issue C and issue D	B-4
Table B-5	Differences between Issue D and Issue E	B-6

List of Figures

Cortex-A9 MPCore Technical Reference Manual

	Key to timing diagram conventions	xvi
Figure 1-1	Example multiprocessor configuration	1-3
Figure 2-1	SCU Control Register bit assignments	2-4
Figure 2-2	SCU Configuration Register bit assignments	2-6
Figure 2-3	SCU CPU Power Status Register bit assignments	2-7
Figure 2-4	SCU Invalidate All Registers in Secure state bit assignments	2-9
Figure 2-5	Filtering Start Address Register bit assignments	2-10
Figure 2-6	Filtering End Address Register bit assignments	2-11
Figure 2-7	SCU Access Control Register bit assignments	2-12
Figure 2-8	SCU Non-secure Access Control Register bit assignments	2-13
Figure 2-9	Input data from a slave to a master with a two-to-three ratio	2-22
Figure 2-10	Input data from a slave to a master with a two-to-five ratio	2-22
Figure 2-11	Output data from master to slave with a three-to-two ratio	2-22
Figure 2-12	Output data from master to slave with a two-to-five ratio	2-23
Figure 2-13	ACLKENS timing example	2-23
Figure 3-1	Distributor register map	3-8
Figure 3-2	Cortex-A9 processor interface register map	3-9
Figure 3-3	ICDDCR bit assignments for Secure accesses	3-12
Figure 3-4	ICDDCR bit assignments for Non-secure accesses	3-12
Figure 3-5	ICDICTR bit assignments	3-13
Figure 3-6	ICDDIR bit assignments	3-15
Figure 3-7	ppi_status Register bit assignments	3-16
Figure 3-8	spi_status Register bit assignments	3-17

Figure 3-9	spi_status Register address map	3-18
Figure 3-10	periph_id_[3:0] Register bit assignments	3-19
Figure 3-11	periph_id_[7:4] Register bit assignments	3-21
Figure 3-12	PrimeCell ID Register bit assignments	3-23
Figure 3-13	ICPIIR Register bit assignments	3-26
Figure 4-1	Private Timer Control Register bit assignments	4-4
Figure 4-2	Private Timer Interrupt Status Register bit assignment	4-5
Figure 4-3	Watchdog Control Register bit assignments	4-7
Figure 4-4	Watchdog Interrupt Status Register bit assignment	4-8
Figure 4-5	Watchdog Reset Status Register bit assignment	4-9
Figure 4-6	Global Timer Control Register bit assignments	4-12
Figure 4-7	Global Timer Interrupt Status Register bit assignment	4-13
Figure 5-1	Three-to-one timing ratio	5-2
Figure 5-2	Cortex-A9 MPCore power domains and clamps	5-11
Figure 6-1	External debug interface signals in CortexA9 MPCore designs	6-2

Preface

This preface introduces the *Cortex-A9 MPCore Technical Reference Manual*. It contains the following sections:

- *About this manual* on page xiv
- *Feedback* on page xix.

About this manual

This book is for the *Cortex-A9 MPCore*.

Note

The Cortex-A9 MPCore consists of between one and four Cortex-A9 processors and a *Snoop Control Unit* (SCU).

Product revision status

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

rn Identifies the major revision of the product.

pn Identifies the minor revision or modification status of the product.

Intended audience

This book is written for hardware and software engineers implementing 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*

Read this for a high-level view of the Cortex-A9 MPCore processor and a description of its features.

Chapter 2 *Snoop Control Unit*

Read this for a description of the Snoop Control Unit of the Cortex-A9 MPCore processor.

Chapter 3 *Interrupt Controller*

Read this for a description of the Cortex-A9 MPCore Interrupt Controller.

Note

The *PrimeCell Generic Interrupt Controller (PL390)* and the Cortex A9 Interrupt Controller share the same programmers model. There are implementation-specific differences.

Chapter 4 *Global timer, Private timers, and Watchdog registers*

Read this for a description of the Cortex-A9 MPCore timer and watchdog registers.

Chapter 5 *Clocks, Resets, and Power Management*

Read this for a description of the clocking modes and the reset signals. This chapter also describes the power management facilities.

Chapter 6 *Debug*

Read this for a description of the Cortex-A9 MPCore debug registers and resources.

Appendix A *Signal Descriptions*

Read this for a description of the Cortex-A9 MPCore input and output signals.

Appendix B *Revisions*

Read this for a description of technical changes between released issues of this book.

Glossary Read this for definitions of terms used in this book.

Conventions

Conventions that this book can use are described in:

- *Typographical*
- *Timing diagrams* on page xvi
- *Signals* on page xvii.

Typographical

The typographical conventions are:

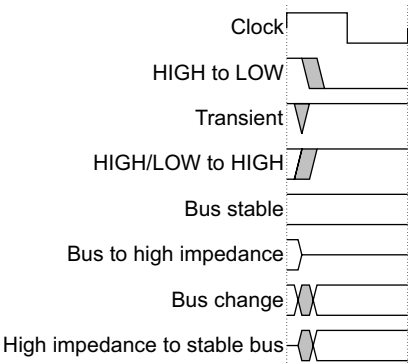
italic Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
<code>monospace</code>	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
<u>monospace</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<i>monospace italic</i>	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 >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: <ul style="list-style-type: none">MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>

Timing diagrams

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

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



Key to timing diagram conventions

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: <ul style="list-style-type: none"> • 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.
Prefix A	Denotes global <i>Advanced eXtensible Interface</i> (AXI) signals:
Prefix AF	Denotes Advanced Trace Bus (ATB) flush control signals.
Prefix AR	Denotes AXI read address channel signals.
Prefix AT	Denotes ATB data flow signals.
Prefix AW	Denotes AXI write address channel signals.
Prefix B	Denotes AXI write response channel signals.
Prefix C	Denotes AXI low-power interface signals.
Prefix H	Denotes <i>Advanced High-performance Bus</i> (AHB) signals.
Prefix P	Denotes <i>Advanced Peripheral Bus</i> (APB) signals.
Prefix R	Denotes AXI read data channel signals.
Prefix W	Denotes AXI write data channel signals.

Additional reading

This section lists publications by ARM and by third parties.

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

ARM publications

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

- *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* (ARM DDI 0406)
- *Cortex-A9 Technical Reference Manual* (ARM DDI 0338)
- *Cortex-A9 Floating-Point Unit Technical Reference Manual* (ARM DDI 0408)

- *Cortex-A9 NEON Media Processing Engine Technical Reference Manual* (ARM DDI 0409)
- *Cortex-A9 MBIST TRM* (ARM DDI 0414)
- *Cortex-A9 Configuration and Sign-Off Guide* (ARM DII 0146)
- *AMBA AXI Protocol v1.0 Specification* (ARM IHI 0022)
- *ARM Generic Interrupt Controller Architecture Specification* (ARM IHI 0048)
- *CoreSight™ PTM™-A9 TRM* (ARM DDI 0401)
- *CoreSight PTM-A9 IM* (ARM DII 0162)
- *CoreSight Program Flow Trace Architecture Specification* (ARM IHI 0035)
- *CoreSight Technology System Design Guide* (ARM DGI 0012)
- *CoreSight v1.0 Architecture Specification* (ARM IHI 0029)
- *ARM Debug Interface v5 Architecture Specification* (ARM IHI 0031)
- *PrimeCell Level 2 Cache Controller (PL310) Technical Reference Manual* (ARM DDI 0246)
- *RealView ICE and RealView Trace User Guide* (ARM DUI 0155)
- *Intelligent Energy Controller Technical Overview* (ARM DTO 0005).

External publications

This section lists relevant documents published by third parties:

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

Feedback

ARM Limited welcomes feedback both on the Cortex-A9 MPCore, and on its documentation.

Feedback on this product

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

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

Feedback on this manual

If you have any comments on this manual, send an e-mail to errata@arm.com. Give:

- the title
- the number
- the relevant page number(s) to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1

Introduction

This chapter describes the Cortex-A9 MPCore processor. It describes the major functional blocks. It contains:

- *About the Cortex-A9 MPCore processor* on page 1-2
- *Configurable options* on page 1-4
- *Private Memory Region* on page 1-5
- *Interfaces* on page 1-7
- *MPCore considerations* on page 1-8
- *Product revisions* on page 1-10.

1.1 About the Cortex-A9 MPCore processor

The Cortex-A9 MPCore processor consists of:

- 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 sit beside the timers and watchdog control registers in the private memory region of the Cortex-A9 MPCore.

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

There are other configuration options that impact 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.

See *Configurable options* on page 1-4.

Figure 1-1 on page 1-3 shows an example multiprocessor configuration.

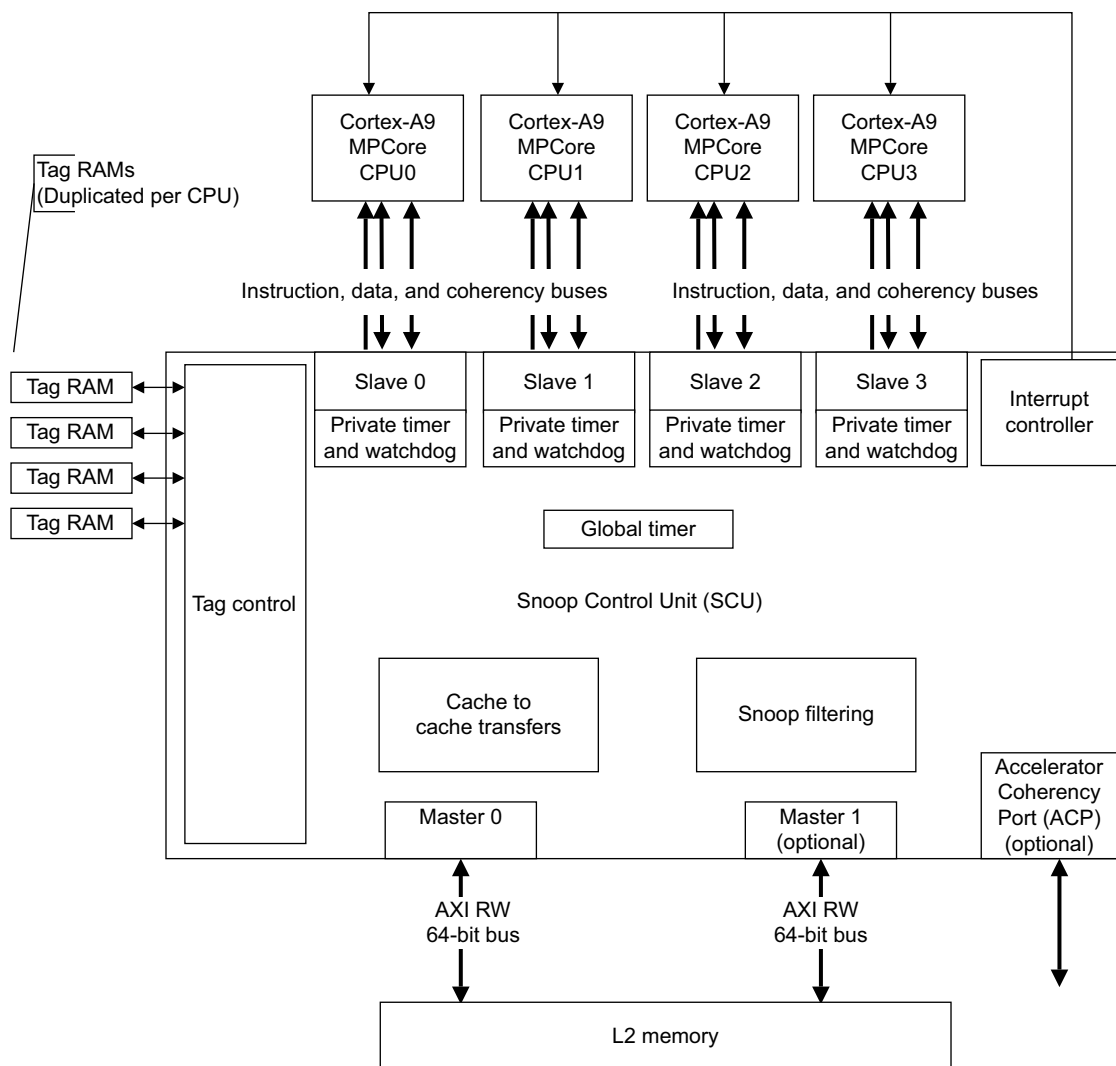


Figure 1-1 Example multiprocessor configuration

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 also available in this configuration.

1.2 Configurable options

Table 1-1 shows the 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 entries or 128 entries
Media Processing Engine with NEON technology per Cortex-A9 processor ^a	Included or not
FPU per Cortex-A9 processor ^b	Included or not
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
<i>Program Trace Macrocell</i> (PTM) interface per Cortex-A9 processor	Included or not
Power off and dormant mode wrappers	Included or not
Support for parity error detection ^c	Included or not
ARM_BIST	Included or not
Master ports	One or two
Accelerator Coherency Port	One, included or not
<i>Shared Peripheral Interrupts</i> (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 *Cortex-A9 TRM* describes the parity error scheme. See *Parity error signals* on page A-24 for a description of the signals.

1.3 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]**. See *Configuration signals* on page A-5 for more information on **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 private memory 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

Table 1-2 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 ^a	Word ^b	Doubleword ^a
Global timer, private timers, and watchdogs	No	No	Yes	No
SCU registers	Yes	No	Yes	No
Cortex-A9 processor interrupt interfaces				
Interrupt distributor				

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

Table 1-3 on page 1-6 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 <i>Snoop Control Unit</i>
0x0100 - 0x01FF	Interrupt controller interfaces	Chapter 3 <i>Interrupt Controller</i>
0x0200 - 0x02FF	Global timer	<i>About the Global Timer</i> on page 4-10
0x0300 - 0x03FF	-	-
0x0400 - 0x04FF	-	-
0x0500 - 0x05FF	-	-
0x0600 - 0x06FF	Private timers and watchdogs	<i>Private timer and watchdog registers</i> on page 4-3
0x0700 - 0x07FF	Reserved	Any access to this region causes a SLVERR abort exception
0x0800 - 0x08FF		
0x0900 - 0x09FF		
0x0A00 - 0x0AFF		
0x0B00 - 0x0FFF		
0x1000 - 0x1FFF	Interrupt Distributor	<i>Interrupt distributor programmers model</i> on page 3-7

1.4 Interfaces

This Cortex-A9 MPCore has the following interfaces:

- *AMBA AXI interfaces*
- *Debug interfaces*
- *Design for Test interface*
- *Interrupts interface.*

1.4.1 AMBA AXI interfaces

The AMBA AXI interfaces include one or two AXI Master port interfaces, and one Accelerator Coherency (ACP) AXI Slave port. See *AMBA AXI Master Port Interfaces* on page 2-15. See also the *AMBA AXI Protocol Specification*.

1.4.2 Debug interfaces

The external debug interface of the Cortex-A9 MPCore is compliant with the ARMv7 Debug recommendations.

With the exception of 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 also provides an external Debug APB interface for memory-mapped accesses to debug and performance monitor registers.

See Chapter 6 *Debug*.

1.4.3 Design for Test interface

The *Cortex-A9 MBIST Controller Technical Reference Manual* gives information on the MBIST interface.

1.4.4 Interrupts interface

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

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

See Chapter 3 *Interrupt Controller*.

1.5 MPCore considerations

This section describes multiprocessing considerations. It contains the following sections:

- *About Cortex-A9 MPCore coherency*
- *Registers with multiprocessor uses*
- *Maintenance operations broadcasting.*

1.5.1 About Cortex-A9 MPCore coherency

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

Cache coherency among multiple L1 caches of the Cortex-A9 processors is enforced 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 Normal memory, with the following attributes:

- write-back
- Write Allocate
- Shareable.

———— **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 non-cacheable.

—————

1.5.2 Registers with multiprocessor uses

The following registers, described in the *Cortex-A9 TRM*, have multiprocessor uses.

- Auxiliary Control Register
- Configuration Base Address Register
- Multiprocessor Affinity Register.

1.5.3 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* 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.

1.6 Product revisions

This section summarizes the differences in functionality between the different releases of this processor:

- *Differences in functionality between r0p0 and r0p1.*

1.6.1 Differences in functionality between r0p0 and r0p1

There is no change in the described functionality between r0p0 and r0p1.

The only 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.

1.6.2 Differences in functionality between r1p0 and r0p1

These differences are in addition to the differences described in the *Cortex-A9 TRM*.

In r1p0 there is a global timer. See *About the Global Timer* on page 4-10.

In the Interrupt Controller **INT** becomes **IRQS**. See *SPI Status Registers* on page 3-17.

SCU CPU Power Status Register bits reassigned. See *SCU CPU Power Status Register* on page 2-7.

1.6.3 Differences in functionality between r2p0 and r1p0

These differences are in addition to the differences described in the *Cortex-A9 TRM*.

Conditions for coherent snoop for ACP requests amended. See *ACP requests* on page 2-24.

SCU Control register updated. See *SCU Control Register* on page 2-3.

SCU Secure Access Control Register renamed to SCU Non-secure Access Control Register. See *SCU Non-secure Access Control Register* on page 2-12.

Removal of SCU Invalidate All Registers in Non-secure State Register and functionality. See Table 2-1 on page 2-3.

Chapter 2

Snoop Control Unit

This chapter describes the *Snoop Control Unit* (SCU). It contains the following sections:

- *About the SCU* on page 2-2
- *SCU registers* on page 2-3.
- *AMBA AXI Master Port Interfaces* on page 2-15
- *AXI master interface clocking* on page 2-21
- *Accelerator Coherency Port* on page 2-24
- *Event communication with an external agent using WFE/SEV* on page 2-27.

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.

2.1.1 TrustZone extensions

The SCU implements support for the ARM Architecture security extensions. See *SCU Access Control Register (SAC) Register* on page 2-11 See *SCU Non-secure Access Control Register* on page 2-12.

2.1.2 SCU event monitoring

The individual CPU event monitors can be configured to gather statistics on the operation of the SCU. The *Cortex-A9 TRM* describes event monitoring.

2.2 SCU registers

Table 2-1 shows the SCU registers. Addresses are relative to the base address of the region for the SCU memory map, which 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	Reset value	Banked	Page
0x00	SCU Control Register	Implementation defined	No	page 2-3
0x04	SCU Configuration Register	Implementation defined	No	page 2-5
0x08	SCU CPU Power Status Register	-	No	page 2-7
0x0C	SCU Invalidate All Registers in Secure State	0x0	No	page 2-8
0x40	Filtering Start Address Register	Defined by FILTERSTART input	No	page 2-9
0x44	Filtering End Address Register	Defined by FILTEREND input	No	page 2-10
0x50	<i>SCU Access Control</i> (SAC) Register	b1111	No	page 2-11
0x54	<i>SCU Non-secure Access Control</i> (SNSAC) Register	0x0	No	page 2-12

Note

SCU registers must not be written with NEON STR instructions.

2.2.1 SCU Control Register

The SCU Control Register characteristics are:

- | | |
|----------------|---|
| Purpose | <ul style="list-style-type: none"> • enables speculative linefills to L2 with PL310 • enables Force all Device to port0 • enables IC standby mode • enables SCU standby mode • enables 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 *SCU registers* on page 2-3.

Figure 2-1 shows the SCU Control Register bit assignments.

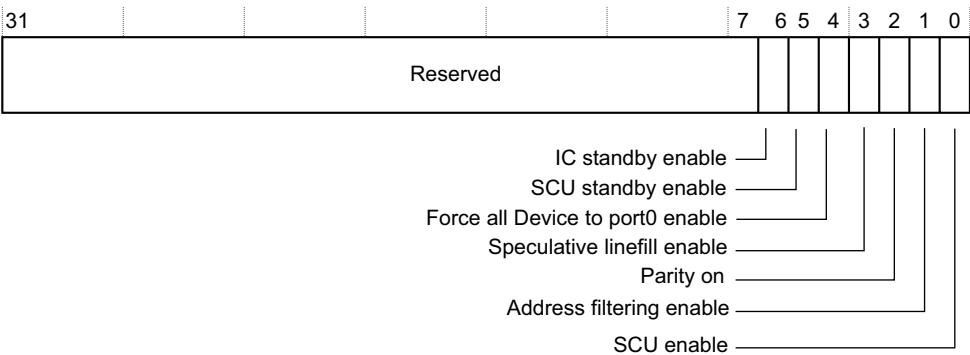


Figure 2-1 SCU Control Register bit assignments

Table 2-2 shows the SCU Control Register bit assignments.

Table 2-2 SCU Control Register bit assignments

Bits	Name	Description
[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.
[5]	SCU standby enable	When set, SCU CLK is turned off when all processors are in WFI mode, there is no pending request on ACP (if implemented), and there is no remaining activity in the SCU. When SCU CLK is off, ARREADY s, AWREADY s and DWREADY s on the ACP are forced HIGH. The clock is turned on when any processor leaves WFI mode, or if there is a new request on ACP.

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

Bits	Name	Description
[4]	Force all Device to port0 enable	When set, all requests from ACP or processors with AxCACHE = DV are forced to be issued on the AXI Master port M0. See <i>Address filtering capabilities</i> on page 2-20.
[3]	Speculative linefills enable	When set, coherent linefill requests are sent speculatively to the PL310 in parallel with the tag lookup. If the tag lookup misses, the confirmed linefill is sent to the PL310 and gets Rdata earlier because the data request was already initiated by the speculative request. This feature works only if the PL310 is present in the design.
[2]	Parity ON	1 = Parity on. 0 = Parity off. This is the default setting. This bit is always zero if support for parity is not implemented.
[1]	Address filtering enable	1 = Addressing filtering on. 0 = Addressing filtering off. The default value is the value of FILTEREN sampled on exit from reset. This bit is always zero if the SCU is implemented in the single master port configuration. See <i>Address filtering capabilities</i> on page 2-20.
[0]	SCU enable	1 = SCU enable. 0 = SCU disable. This is the default setting.

2.2.2 SCU Configuration Register

The SCU Configuration Register characteristics are:

Purpose	<ul style="list-style-type: none"> 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 <i>SCU registers</i> on page 2-3.

Figure 2-2 on page 2-6 shows the format for this register.

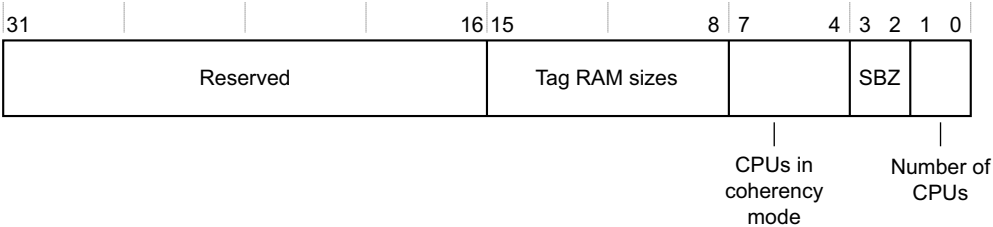


Figure 2-2 SCU Configuration Register bit assignments

Table 2-3 shows the SCU Configuration Register bit assignments.

Table 2-3 SCU Configuration Register bit assignments

Bits	Name	Description
[31:16]	Reserved	<i>Should-Be-Zero</i> (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: b11 = reserved b10 = 64KB cache, 256 indexes per tag RAM b01 = 32KB cache, 128 indexes per tag RAM b00 = 16KB cache, 64 indexes per tag RAM.
[7:4]	CPUs SMP	Defines the Cortex-A9 processors that are in <i>Symmetric Multi-processing</i> (SMP) or <i>Asymmetric Multi-processing</i> (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 b11 = four Cortex-A9 processors, CPU0, CPU1, CPU2, and CPU3 b10 = three Cortex-A9 processors, CPU0, CPU1, and CPU2 b01 = two Cortex-A9 processors, CPU0 and CPU1 b00 = one Cortex-A9 processor, CPU0.

2.2.3 SCU CPU Power Status Register

The SCU CPU Power Status Register characteristics are:

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 SCU the mode it is about to enter so that the SCU can determine if it still can send coherency requests to the Cortex-A9 processor. The Cortex-A9 processor then executes a WFI entry instruction. When in WFI state, the **PWRCTL**On 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 set-up.

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 *SCU registers summary* on page 2-3.

Figure 2-3 shows the SCU CPU Power Status Register bit assignments.

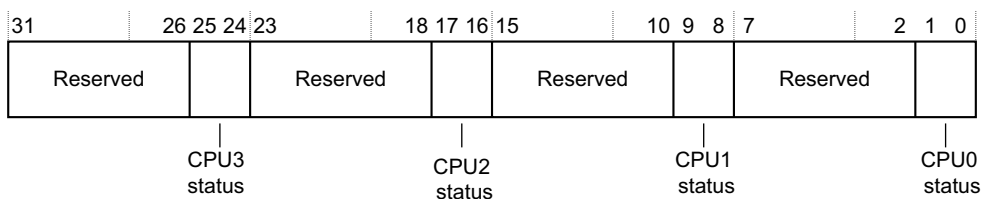


Figure 2-3 SCU CPU Power Status Register bit assignments

Table 2-4 shows the SCU CPU Power Status Register bit assignments.

Table 2-4 SCU CPU Power Status Register bit assignments

Bits	Name	Description
[31:26]	Reserved	SBZ
[25:24]	CPU3 status	Power status of the Cortex-A9 processor: b00: Normal mode. b01: Reserved. b10: the Cortex-A9 processor is about to enter (or is in) dormant mode. No coherency request is sent to the Cortex-A9 processor. b11: 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.
[23:18]	Reserved	SBZ
[17:16]	CPU2 status	Power status of the Cortex-A9 processor.
[15:10]	Reserved	SBZ
[9:8]	CPU1 status	Power status of the Cortex-A9 processor.
[7:2]	Reserved	SBZ
[1:0]	CPU0 status	Power status of the Cortex-A9 processor.

2.2.4 SCU Invalidate All Registers in Secure State Register

The SCU Invalidate All Registers in Secure State characteristics are:

Purpose	Invalidates the SCU tag RAMs on a per Cortex-A9 processor and per way basis.
Usage constraints	This register: <ul style="list-style-type: none">Invalidates all lines in the selected ways.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 <i>SCU registers summary</i> on page 2-3.

Figure 2-4 on page 2-9 shows the format of this register.

31				16	15	12	11	8	7	4	3	0
SBZ					CPU3 ways	CPU2 ways	CPU1 ways	CPU0 ways				

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

Table 2-5 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	Description
[31:16]	-	-
[15:12]	CPU3 ways	Indicates the ways that must be invalidated for Cortex-A9 MPCore CPU3. Writing to these bits has no effect if the Cortex-A9 MPCore processor has fewer than four processors.
[11:8]	CPU2 ways	Indicates the ways that must be invalidated for Cortex-A9 MPCore CPU2. Writing to these bits has no effect if the Cortex-A9 MPCore processor has fewer than three processors.
[7:4]	CPU1 ways	Indicates the ways that must be invalidated for Cortex-A9 MPCore CPU1. Writing to these bits has no effect if the Cortex-A9 MPCore processor has fewer than two processors.
[3:0]	CPU0 ways	Indicates the ways that must be invalidated for Cortex-A9 MPCore CPU0.

2.2.5 Filtering Start Address Register

The Filtering Start Address Register characteristics are:

Purpose	Provides the start address for use with master port 1 in a two-master port configuration.
Usage constraints	This register is writable: <ul style="list-style-type: none"> 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 <i>SCU registers summary</i> on page 2-3.

Figure 2-5 on page 2-10 shows the Filtering Start Address Register bit assignments.

31				20	19							0
Filtering start address					SBZ							

Figure 2-5 Filtering Start Address Register bit assignments

Table 2-6 shows the bit assignments for the Filtering Start Address Register.

Table 2-6 Filtering Start Address Register bit assignments

Bits	Name	Description
[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

See *Configuration signals* on page A-5.

2.2.6 Filtering End Address Register

The Filtering End Address Register characteristics are:

Purpose	Provides the end address for use with master port 1 in a two-master port configuration.
Usage constraints	<p>This register is writable</p> <ul style="list-style-type: none"> • 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 Table 2-1 on page 2-3.

Figure 2-6 on page 2-11 shows the Filtering End Address Register bit assignments.

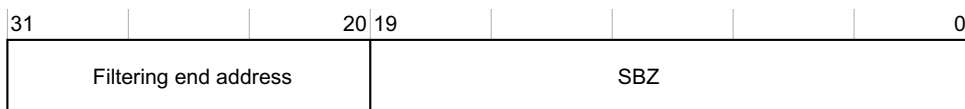
**Figure 2-6 Filtering End Address Register bit assignments**

Table 2-7 shows the bit assignments for the Filtering End Address Register.

Table 2-7 Filtering End Address Register bit assignments

Bits	Name	Description
[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.

See *Configuration signals* on page A-5.

2.2.7 SCU Access Control Register (SAC) Register

The SAC characteristics are:

Purpose	Controls access to the following registers on a per Cortex-A9 processor basis: <ul style="list-style-type: none"> • <i>SCU Control Register</i> on page 2-3 • <i>SCU CPU Power Status Register</i> on page 2-7 • <i>SCU Invalidate All Registers in Secure State Register</i> on page 2-8 • <i>Filtering Start Address Register</i> on page 2-9 • <i>Filtering End Address Register</i> on page 2-10 • <i>SCU Non-secure Access Control Register</i> on page 2-12.
Usage constraints	This register is writable: <ul style="list-style-type: none"> • 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 SCU Non-secure Access Control Registers are set.
Configurations	Available in all Cortex-A9 multiprocessor configurations.
Attributes	See the register summary in <i>SCU registers summary</i> on page 2-3.

Figure 2-7 shows the SCU SAC Register bit assignments.

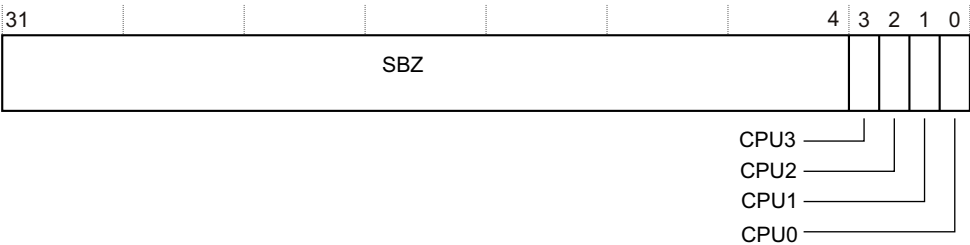


Figure 2-7 SCU Access Control Register bit assignments

Table 2-8 shows the bit assignment for the SCU Access Control Register.

Table 2-8 SCU Access Control Register bit assignments

Bits	Name	Description
[31:4]	SBZ	-
[3]	CPU3	0 = CPU3 cannot access the components ^a 1 = CPU3 can access the components. This is the default.
[2]	CPU2	0 = CPU2 cannot access the components. 1 = CPU2 can access the components. This is the default.
[1]	CPU1	0 = CPU1 cannot access the components. 1 = CPU1 can access the components. This is the default.
[0]	CPU0	0 = CPU0 cannot access the components. 1 = CPU0 can access the components. This is the default.

a. The accessible components 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.

2.2.8 SCU Non-secure Access Control Register

The SCU Non-secure Access Control Register characteristics are:

- Purpose** Controls Non-secure access to the following registers on a per Cortex-A9 processor basis:
- SCU Control Register on page 2-3
 - SCU CPU Power Status Register on page 2-7
 - Filtering Start Address Register on page 2-9

- *Filtering End Address Register* on page 2-10
- *SCU Access Control Register (SAC) Register* on page 2-11.

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 Table 2-1 on page 2-3.

Figure 2-8 shows the bit assignments.

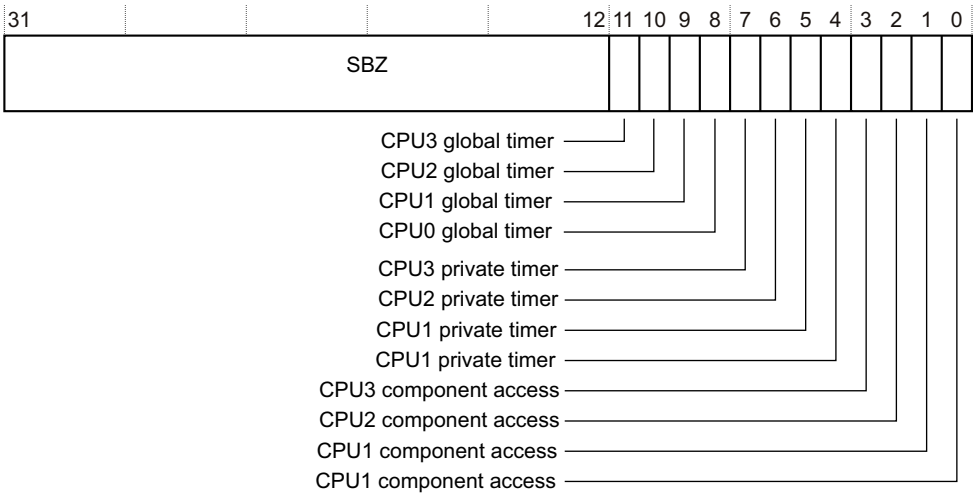


Figure 2-8 SCU Non-secure Access Control Register bit assignments

Table 2-9 shows the bit assignment for the SCU Non-secure Access Control Register.

Table 2-9 SCU Non-secure Access Control Register bit assignments

Bits	Name	Description
[31:12]	SBZ	-

Table 2-9 SCU Non-secure Access Control Register bit assignments (continued)

Bits	Name	Description
[11]	CPU3 global timer	Non-secure access to the global timer for CPU<n>.
[10]	CPU2 global timer	<ul style="list-style-type: none"> • <n> is 3 for bit[11] • <n> is 2 for bit[10]
[9]	CPU1 global timer	<ul style="list-style-type: none"> • <n> is 1 for bit[9]
[8]	CPU0 global timer	<ul style="list-style-type: none"> • <n> is 0 for bit[8]. <p>0 = Secure accesses only. This is the default value. 1 = Secure accesses and Non-Secure accesses.</p>
[7]	Private timers for CPU<n>	Non-secure access to the private timer and watchdog for CPU<n>.
[6]		<ul style="list-style-type: none"> • <n> is 3 for bit[7]
[5]		<ul style="list-style-type: none"> • <n> is 2 for bit[6]
[4]		<ul style="list-style-type: none"> • <n> is 1 for bit[5] • <n> is 0 for bit[4]. <p>0 = Secure accesses only. Non-secure reads return 0. This is the default value. 1 = Secure accesses and Non-secure accesses.</p>
[3]	Component access for CPU<n>	Non-secure access to the components for CPU<n>.
[2]		<ul style="list-style-type: none"> • <n> is 3 for bit[3]
[1]		<ul style="list-style-type: none"> • <n> is 2 for bit[2]
[0]		<ul style="list-style-type: none"> • <n> is 1 for bit[1] • <n> is 0 for bit[0]. <p>0 = CPU cannot write the components^a 1 = CPU can access the components^a.</p>

a. The accessible components 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

The following sections describe the AMBA AXI interfaces:

- *AXI issuing capabilities*
- *Cortex-A9 MPCore supported AXI transactions* on page 2-16
- *AXI transaction IDs* on page 2-16
- *AXI USER attributes encodings* on page 2-18
- *Address filtering capabilities* on page 2-20.

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. See *Address filtering capabilities* on page 2-20. Table 2-10 shows the AXI master interface attributes.

Table 2-10 AXI master interface attributes

Attribute	Format
Write Issuing Capability	10 per processor, including: <ul style="list-style-type: none"> • eight non-cacheable writes • two evictions. If the ACP is implemented, three more write transactions can be issued.
Read Issuing Capability	9 per processor, including: <ul style="list-style-type: none"> • four instruction reads • four linefill reads. • one non-cacheable read. If the ACP is implemented, seven more read transactions can be issued.
Combined Issuing Capability	21 per processor. If the ACP is implemented, 10 more transactions can be issued.
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*.

2.3.2 Cortex-A9 MPCore supported AXI transactions

Cortex-A9 MPCore master ports generate only a subset of all possible AXI transactions.

For coherent and noncoherent write-back write-allocate transactions the supported transactions are:

- WRAP4 64-bit for read transactions (linefills)
- INCR4 64-bit for write transactions (evictions)

For noncoherent non-cacheable transactions:

- INCRN (N:1-16) 32-bit read transfers
- INCRN (N:1-8) 64-bit read transfers
- INCR1 8-bit, 16-bit, 32-bit, and 64-bit read transfers
- INCR1 8-bit, 16-bit, 32-bit, and 64-bit write transfers
- INCR1 8-bit, 16-bit, 32-bit, 64-bit exclusive read transfers
- INCR1 8-bit, 16-bit, 32-bit, 64-bit exclusive write transfers
- INCR1 32-bit RW (locked) for swap
- INCR1 8-bit RW (locked) for swap.

The following points apply to AXI transfers:

- Wrap burst are only read transfers, 64-bit, 4 beats
- Incr single can be any size for read or write
- Incr burst (more than one beat) are only 32-bit or 64-bit
- No transfer from the Cortex-A9 processors is marked as FIXED
- Write transfers with all byte strobes low can occur.

2.3.3 AXI transaction IDs

There are several possible sources for the AXI transactions a Cortex-A9MP processor issues on its AXI master ports. This section describes the AXI transaction IDs and AXI USER bits in the following sections:

- *ARIDMx[5:0] encodings* on page 2-17
- *AWIDMx[5:0] encodings* on page 2-17.
- *ARUSERMx[6:0] encodings* on page 2-18
- *AWUSERMx[8:0] encodings* on page 2-19.

ARIDMx[5:0] encodings

This section describes the ARIDMx[5:0] encodings for read transactions. As Table 2-11 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: b000 non-cacheable b010 data linefill buffer 0 b011 data linefill buffer 0 b100 instruction linefill b101 instruction linefill b110 instruction linefill b111 instruction linefill	ACP read IDs ARIDMx[5:3] = ARIDS[2:0]
ARIDMx[1:0]	Cortex-A9 processor: b00 CPU0 b01 CPU1 b10 CPU2 b11 CPU3	Unused

AWIDMx[5:0] encodings

This section describes the AWIDMx[5:0] encodings for write transactions. As Table 2-12 on page 2-18 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.

- **AWIDM_x[2]** = 1 the transaction originates from the ACP.

Table 2-12 AWIDM_x encodings

Transaction types		
	Cortex-A9 transactions	ACP transactions
AWIDM_x[2]	AWIDM_x[2] = 0	AWIDM_x[2] = 1
AWIDM_x[5:3]	b000 non-cacheable b010 eviction b011 eviction b100 eviction b101 eviction	ACP read IDs AWIDM_x[5:3] = AWIDS[2:0]
AWIDM_x[1:0]	b00 CPU0 b01 CPU1 b10 CPU2 b11 CPU3	Unused, forced to b00>

2.3.4 AXI USER attributes encodings

This section describes the implementation-specific AXI USER bit encodings on the master ports in the following sections:

- *ARUSERM_x[6:0] encodings*
- *AWUSERM_x[8:0] encodings* on page 2-19.

ARUSERM_x[6:0] encodings

This section describes the **ARUSERM_x[6:0]** encodings for read transactions. As Table 2-13 on page 2-19 shows, the value and the meaning of the **ARUSERM_x** encodings depend on the source of the transaction. There are transactions originating from Cortex-A9 processors and transactions originating from the ACP:

- **ARIDM_x[2]** = 0 from one of the Cortex-A9 processors.

- **ARIDM_x[2]** = 1 from the ACP.

Table 2-13 ARUSERM_x[6:0] encodings

Transaction types		
	Cortex-A9 transactions ARIDM _x [2] = 0	ACP transactions ARIDM _x [2] = 1
ARUSERM _x [6]	b0 Speculative linefill to PL310	ACP USER bits
ARUSERM _x [5]	Prefetch hint	ARUSERM _x [6:5] = 2'b00
ARUSERM _x [4:1]	Inner attributes b0000 Strongly Ordered b0001 Device b0011 Normal Memory NonCacheable b0110 WriteThrough b0111 Write Back no Write Allocate b1111 Write Back Write Allocate	ARUSERM _x [4:1] = ARUSERS _x [4:1]
ARUSERM _x [0]	Shared bit 1 Coherent request 0 Non-coherent request	

AWUSERM_x[8:0] encodings

This section describes the **AWUSERM_x[8:0]** encodings for write transactions. As Table 2-14 on page 2-20 shows, the value and the meaning of the **AWUSERM_x** encodings depend on the source of the transaction:

- **AWIDM_x[2]** = 0 from one of the Cortex-A9 processors.

- **AWIDM_x[2]** = 1 from the ACP.

Table 2-14 **AWUSERM_x[8:0]** encodings

Transaction types		
	Cortex-A9 transactions AWIDM _x [2] = 0	ACP transactions AWIDM _x [2] = 1
AWUSERM _x [8]	Early BRESP enable	ACP USER bits
AWUSERM _x [7]	Full line of write zeros indication	AWUSERM _x [8:5] = 4'b0000
AWUSERM _x [6]	Clean eviction information	
AWUSERM _x [5]	L1 eviction information	
AWUSERM _x [4:1]	Inner attributes b0000 Strongly Ordered b0001 Device b0011 Normal Memory NonCacheable b0110 WriteThrough b0111 Write Back no Write Allocate b1111 Write Back Write Allocate	AWUSERM _x [4:0] = AWUSERS[4:0] ^a
AWUSERM _x [0]	Shared bit b0 Non-coherent request b1 Coherent request	

a. Each master agent connected to the ACP can determine its corresponding AXI USER signals. To maintain consistency ARM recommends that the ACP AXI USER signal encodings match those of the Cortex-A9 processors.

2.3.5 Address filtering capabilities

The SCU register bank contains dedicated registers to provide address filtering capabilities:

- *Filtering Start Address Register* on page 2-9
- *Filtering End Address Register* on page 2-10
- *SCU Control Register* on page 2-3.

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, provided that the AXI ordering rules are respected. However, in this case, locked and exclusive accesses are always issued on AXI Master port M0.

Device filtering

In the r2p0 revision, the SCU also provides the ability to direct all device accesses onto the same AXI Master port, M0. See *SCU Control Register* on page 2-3.

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.

2.3.6 AXI master interface clocking

The Cortex-A9 MPCore Bus Interface Unit supports the following AXI bus ratios:

- 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.

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

- **INCLKLENM0** and **OUTCLKLENM0**
- **INCLKLENM1** and **OUTCLKLENM1**.

Figure 2-9 on page 2-22 shows input data going from a slave to a master with a timing ratio of two to three.

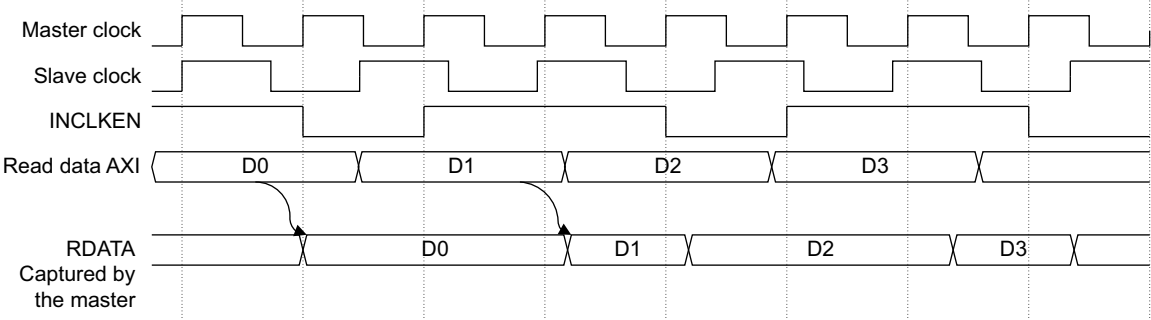


Figure 2-9 Input data from a slave to a master with a two-to-three ratio

Figure 2-10 shows input data going from a slave to a master with a timing ratio of two to five.

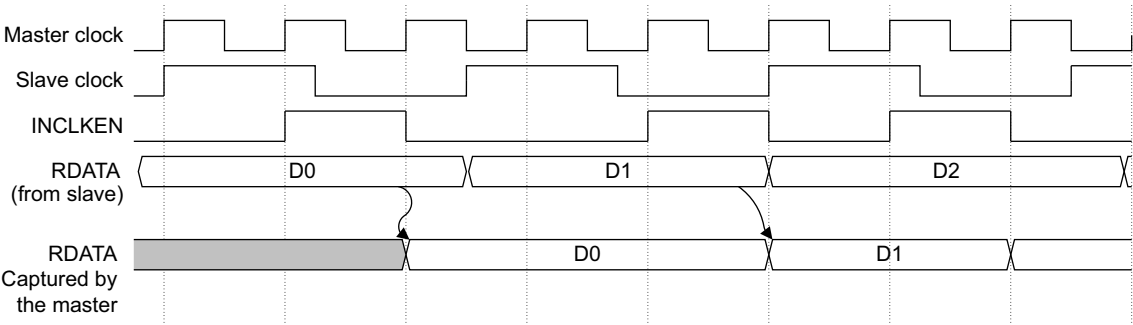


Figure 2-10 Input data from a slave to a master with a two-to-five ratio

Figure 2-11 shows output data going from a master to a slave with a timing ratio of two to three.

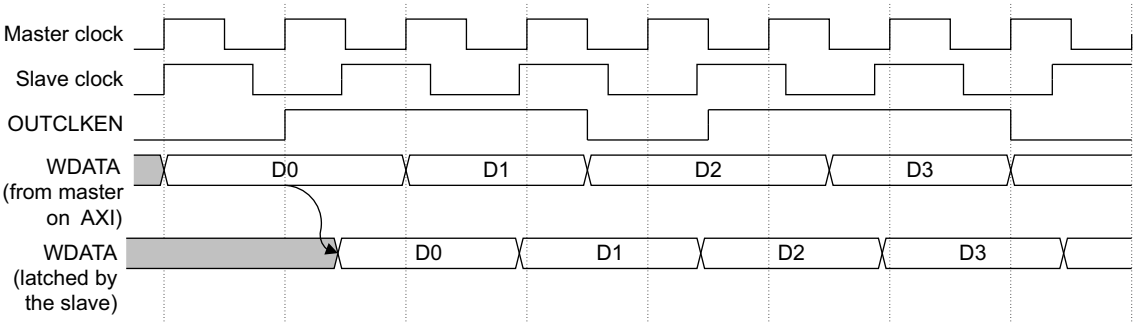


Figure 2-11 Output data from master to slave with a three-to-two ratio

Figure 2-12 shows output data going from a master to a slave with a timing ratio of two to five.

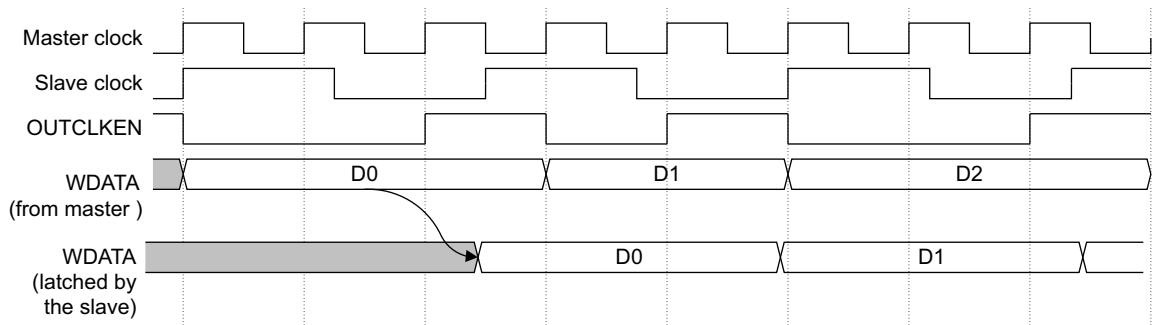


Figure 2-12 Output data from master to slave with a two-to-five ratio

2.3.7 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.

Figure 2-13 shows a timing example where **ACLKENS** is used with a 3:1 clock ratio between **CLK** and the ACP AXI clock, **ACLK**.

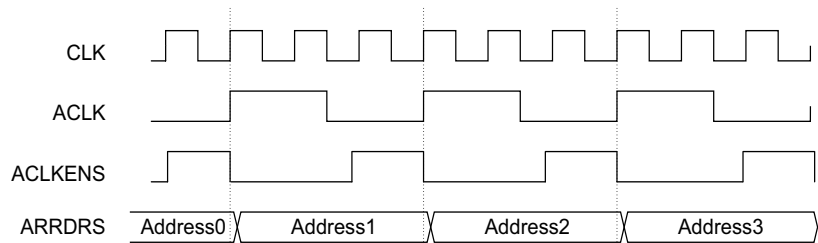


Figure 2-13 ACLKENS timing example

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 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 DMAengine 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. *ACP interface clocking* on page 2-23 describes ACP timing.

The following sections describe the ACP:

- *ACP requests*
- *ACP limitations* on page 2-25.

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, with the exception of 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 once 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 with the exception of the locked attribute.

Note

The transaction may 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]** = 1 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 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. This section describes some ACP limitations. It contains the following sections:

- ACP performance limitations
- ACP functional limitations.

ACP performance limitations

ACP accesses are optimized for transfers which match Cortex-A9 processors coherent requests:

- 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 the following transfers which are not supported:

- Coherent exclusive read and write transfers
- Coherent locked read and write transfers
- 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, which 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**.

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 core of the cluster. This enables the external agent to signal to the cores 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 internal processor has executed an SEV instruction by checking the **EVENTO** pin. This pin is set high for one **CPUCLK** clock cycle when any of the internal Cortex-A9 processor executes an SEV instruction.

Chapter 3

Interrupt Controller

This chapter describes the implementation-defined features of the Interrupt Controller. This chapter does not reproduce information already in the *ARM Generic Interrupt Controller Architecture Specification*. The chapter contains the following sections:

- *About the Interrupt Controller* on page 3-2
- *TrustZone support* on page 3-5
- *Interrupt distributor programmers model* on page 3-7
- *Cortex-A9 processor interface register descriptions* on page 3-25.

Note

The *PrimeCell Generic Interrupt Controller GIC (PL390)* and the Cortex A9 Interrupt Controller share a programmers model. There are implementation-specific differences.

3.1 About the Interrupt Controller

The Interrupt Controller is a single functional unit that is located in a Cortex-A9 multiprocessor design. There is one Cortex-A9 processor 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. Access the Interrupt Controller using **PERIPHBASE[31:13]**.

3.1.1 Interrupt Controller Clock frequency

The clock period is configured, during integration, as a multiple of the SCU 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* for a description of **PERIPHCLK** and **PERIPHCLKEN**.

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

3.1.2 Interrupt types and sources

This section describes interrupt types and interrupt distributor sources in:

- *Interrupt types*
- *Interrupt Distributor interrupt sources* on page 3-3.

Interrupt types

The Cortex-A9 multiprocessor has the following types of interrupts:

Software Generated Interrupt (SGI)

Generated by writing to the Software Generated Interrupt Register (ICDSGIR).

A maximum of 16 SGIs can be generated for each Cortex-A9 processor interface. See Software Generated Interrupt Register on page 4-48.

Private Peripheral Interrupt (PPI)

An interrupt generated by a peripheral that is specific to a single Cortex-A9 processor.

There are 5 PPIs for each Cortex-A9 processor interface. See *Interrupt Distributor interrupt sources* on page 3-3.

Shared Peripheral Interrupt (SPI)

An interrupt generated by a peripheral that the Interrupt Controller can route to any, or all, Cortex-A9 processor interfaces.

The Interrupt Controller supports a maximum of 224 SPIs. The permitted values are 0, 32, 64, 96, 128, 160, 192, or 224.

Lockable Shared Peripheral Interrupts (LSPI)

There are 31 LSPIs, interrupts 32-62. You can configure and then lock these interrupts against further change using **CFGSDISABLE**. The LSPIs are present only if the SPIs are present. See SPI Status Registers on page 4-46.

Interrupt Distributor interrupt sources

The interrupt distributor centralizes all interrupt sources before dispatching the highest priority ones to each individual Cortex-A9 processor.

Hardware ensures that an interrupt targeted at several Cortex-A9 processors can only be taken by one Cortex-A9 processor at a time. See the *ARM Generic Interrupt Controller Architecture Specification*.

All interrupt sources are identified by a unique ID. All interrupt sources have their own configurable priority and list of targeted Cortex-A9 processors, that is, a list of Cortex-A9 processors that the interrupt is sent to when triggered by the interrupt distributor.

Interrupt sources are of the following types:

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.

A legacy nFIQ pin, PPI(0)

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(1)

Each Cortex-A9 processor has its own private timers that can generate interrupts, using ID29.

Watchdog timers, PPI(2)

Each Cortex-A9 processor has its own watchdog timers that can generate interrupts, using ID30.

A legacy nIRQ pin, PPI(3)

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.

Global timer, PPI(4)

The global timer uses ID27.

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 (posedge) or level sensitive (high level). SPIs start at ID32.

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 b10.

PPI The bits are read-only

PPI[0] and [3]:b01

interrupt is active LOW level sensitive.

PPI[1], [2],and[4]:b11

interrupt is rising-edge sensitive.

See Table 3-6 on page 3-16.

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

b01 interrupt is active HIGH level sensitive

b11 interrupt is rising-edge sensitive.

3.2 TrustZone 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 signalled using the IRQ mechanism of a Cortex-A9 processor.

Note

A Non-secure access to a register of a Secure interrupt behaves as RAZ/WI.

3.2.1 Priority formats

The Cortex-A9 processor implements a five-bit version of the priority format in 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 the read and 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 enable_s Register. See *Distributor Control Register* on page 4-17.

Secure interrupts defined by LSPI field in the ic_type_reg:

- Interrupt Security Registers. See *Interrupt security registers* on page 4-21.
- Enable Set Registers.
- Enable Clear Registers.
- Pending Set Registers.
- Pending Clear Registers.

- Priority Level Registers.
- SPI Target Registers.
- Interrupt Configuration Register. See *Interrupt Configuration Registers* on page 4-42.

Cortex-A9 processor interface

The ICPICR Register, except for the EnableNS bit. See *Processor Interface Control Register* on page 4-58.

———— Note ————

When **CFGSDISABLE** is HIGH the Interrupt Controller only permits write access to the EnableNS bit. All other bits are read-only.

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 Interrupt distributor programmers model

The interrupt distributor contains the following register maps:

- *Distributor register map*
- *Cortex-A9 processor interface register map* on page 3-9.

3.3.1 Distributor register map

The distributor has 4KB of address space allocated to it, as Figure 3-1 on page 3-8 shows. Table 1-3 on page 1-6 gives the offset of this page from **PERIPHBASE[31:13]**.

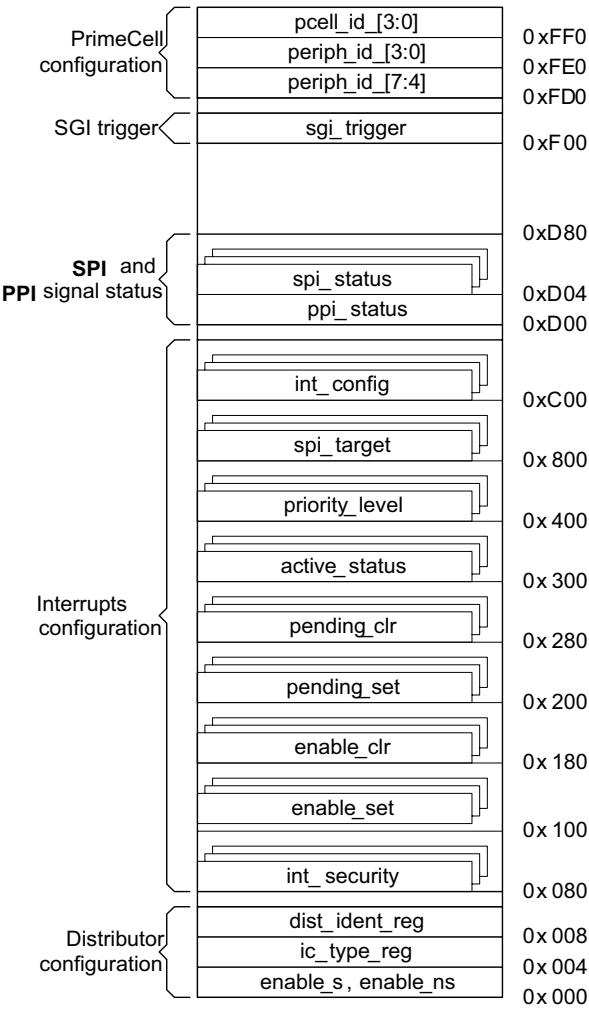


Figure 3-1 Distributor register map

In Figure 3-1, the register map address range contains the following regions:

Distributor configuration

Use these registers to determine the global configuration of the distributor and to control its operating state.

Interrupts configuration

These registers hold the operating parameters for each interrupt.

PPI and SPI status

These registers provide the status of the **PPI** and **SPI** input signals.

SGI trigger Use this register to generate a Software Generated Interrupt.

3.3.2 Cortex-A9 processor interface register map

Each Cortex-A9 processor interface has 256 bytes of address space allocated to it, as Figure 3-2 shows.

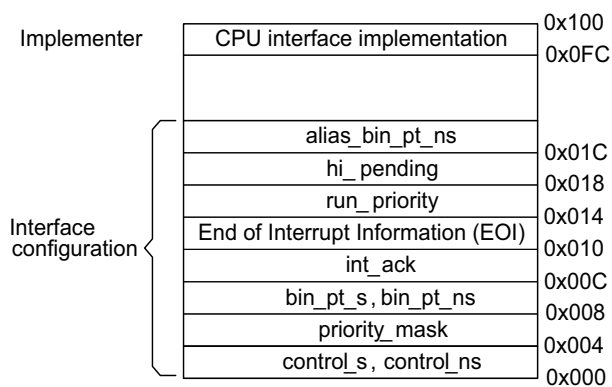


Figure 3-2 Cortex-A9 processor interface register map

In Figure 3-2, the register map address range contains the following regions:

Implementer

These registers enable the identification of the processor interface by software.

Interface configuration

Use these registers to control the operating state of the Cortex-A9 processor interface.

3.3.3 Distributor register descriptions

This section describes the registers that the distributor provides. Table 3-1 on page 3-10 shows the distributor registers.

Registers not described in Table 3-1 on page 3-10 are RAZ/WI. This section does not reproduce information about registers already described in the *ARM Generic Interrupt Controller Architecture Specification*.

The ICDIPR and ICDIPTR registers are byte accessible and word accessible. All other registers in Table 3-1 are word accessible.

See Table 1-3 on page 1-6 for the offset of this page from **PERIPHBASE[31:13]**.

Table 3-1 Distributor register summary

Base	Name	Type	Reset	Width	Description
0x000	ICDDCR ^a	RW	0x00000000	32	<i>Distributor Control Register</i> on page 3-11
0x004	ICDICTR	RO	Configuration dependent	32	<i>Interrupt Controller Type Register</i> on page 3-13
0x008	ICDDIR	RO	0x0102043B	32	<i>Distributor Implementer Identification Register</i> on page 3-14
0x00C - 0x07C	-	-	-	-	Reserved
0x080 - 0x09C	ICDISR	RW ^b	0x00000000	32	Interrupt security registers
0x100 - 0x11C	ICDISER	RW	0x00000000 ^c	32	Enable set registers
0x180 - 0x19C	ICDICER	RW	0x00000000 ^b	32	Enable clear registers
0x200 - 0x21C	IDISPR	RW	0x00000000	32	Pending set registers
0x280 - 0x29C	ICDICPR	RW	0x00000000	32	Pending clear registers
0x300 - 0x31C	ICDABR	RO	0x00000000	32	Active status registers
0x380 - 0x3FC	-	-	-	-	Reserved
0x400 - 0x4FC	ICDIPR	RW	0x00000000	32	Priority level registers
0x7FC	-	-	-	-	Reserved
0x800 - 0x8FC	ICDIPTR	RW ^d	0x00000000	32	SPI Target registers
0xBFC	-	-	-	-	Reserved
0xC00 - 0xC3C	ICDICR	RW	Configuration dependent	32	Interrupt Configuration Registers
0xD00	ppi_status	-	0x00000000	32	<i>PPI Status Register</i> on page 3-15
0xD04 - 0xD1C	spi_status	RO	0x00000000	32	<i>SPI Status Registers</i> on page 3-17
0xD80 - 0xEFC	-	-	-	-	Reserved
0xF00	ICDSGIR	WO	-	32	Software Generated Interrupt Register

Table 3-1 Distributor register summary (continued)

Base	Name	Type	Reset	Width	Description
0xF04 - 0xFCC	-	-	-	-	Reserved
0xFD0 - 0xFEC	periph_id_[4:0]	RO	Configuration dependent	8	<i>Peripheral Identification Registers</i> on page 3-18
0xFF0 - 0xFFC	component_id_[3:0]	RO	-	8	<i>PrimeCell Identification Registers</i> on page 3-22

- You cannot modify enable_s if **CFGSDISABLE** is set. You can modify enable_ns even if **CFGSDISABLE** is set, through the S or the NS register.
- You must access this register in Secure state.
- The reset value for the registers that contain the SGI and PPI interrupts is implementation-dependent.
- The registers that control the SGI and PPI interrupts are read-only and the reset value is configuration-dependent.

3.3.4 Distributor Control Register

The ICDDCR characteristics are:

Purpose Controls whether the distributor responds to external stimulus changes that occur on **SPIs** and **PPIs**.

Usage constraints

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

Non-secure access distributor provides access to the enable_ns Register.

Secure access distributor provides access to the enable_s Register.

Configurations

Available in all Cortex-A9 multiprocessor configurations.

Attributes See the register summary in Table 3-1 on page 3-10.

Figure 3-3 on page 3-12 shows the ICDDCR bit assignments for Secure accesses.

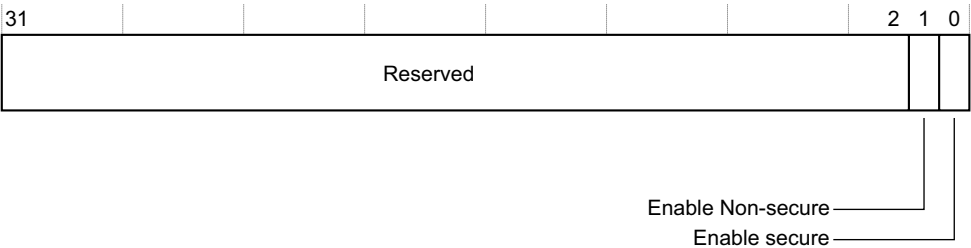


Figure 3-3 ICDDCR bit assignments for Secure accesses

Table 3-2 shows the ICDDCR bit assignments for secure accesses.

Table 3-2 ICDDCR bit assignments for secure accesses

Bits	Name	Description
[31:2]	-	Reserved
[1]	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
[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.

Figure 3-4 shows the ICDDCR bit assignments for Non-secure accesses.

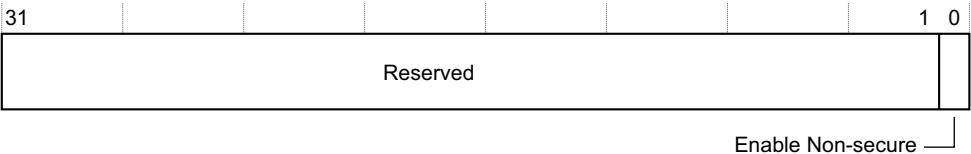


Figure 3-4 ICDDCR bit assignments for Non-secure accesses

Table 3-3 shows the ICDDCR bit assignments for Non-secure accesses.

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

Bits	Name	Description
[31:1]	-	Reserved
[0]	Enable Non-secure	<p>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</p> <p>1 = enables the distributor to update register locations for Non-secure interrupts</p>

3.3.5 Interrupt Controller Type Register

The ICDICTR characteristics are:

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 Table 3-1 on page 3-10.

Figure 3-5 shows the ICDICTR bit assignments.

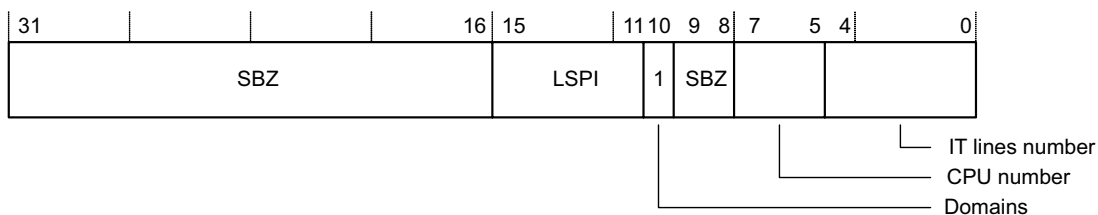


Figure 3-5 ICDICTR bit assignments

Table 3-4 shows the ICDICTR bit assignments.

Table 3-4 ICDICTR bit assignments

Bits	Name	Description
[31:16]	-	Reserved
[15:11]	LSPI	Returns the number of <i>Lockable Shared Peripheral Interrupts</i> (LSPIs) that the controller contains. The encoding is: b11111 = 31 LSPIs, which are the interrupts of IDs 32-62. When CFGSDISABLE is HIGH then the interrupt controller prevents writes to any register locations that control the operating state of an LSPI.
[10]	Domains	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]	-	SBZ
[7:5]	CPU number	The encoding is: b000 the Cortex-A9 design contains one Cortex-A9 processor. b001 the Cortex-A9 design contains two Cortex-A9 processors. b010 the Cortex-A9 design contains three Cortex-A9 processors. b011 the Cortex-A9 design contains four Cortex-A9 processors. b1xx: Reserved for future extensions.
[4:0]	SPI lines number	The encoding is: b00000 = the distributor provides 32 interrupts ^a , no external interrupt lines. b00001 = the distributor provides 64 interrupts, 32 external interrupt lines. b00010 = the distributor provides 96 interrupts, 64 external interrupt lines. b00011 = the distributor provide 128 interrupts, 96 external interrupt lines. b00100 = the distributor provides 160 interrupts, 128 external interrupt lines. b00101 = the distributor provides 192 interrupts, 160 external interrupt lines. b00110 = the distributor provides 224 interrupts, 192 external interrupt lines. b00111 = the distributor provides 256 interrupts, 224 external interrupt lines. All other values are Reserved for future extension.

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

3.3.6 Distributor Implementer Identification Register

The ICDDIR characteristics are:

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 Table 3-1 on page 3-10.

Figure 3-6 shows the ICDDIR bit assignments.

31	24	23		12	11		0
Implementation defined			Revision number			Implementer	

Figure 3-6 ICDDIR bit assignments

Table 3-5 shows the ICDDIR bit assignments.

Table 3-5 ICDDIR bit assignments

Bits	Name	Description
[31:24]	Implementation version	Gives implementation version number.
[23:12]	Revision number	Returns the revision number of the controller.
[11:0]	Implementer	Implementer number.

3.3.7 PPI Status Register

The ppi_status Register characteristics are:

- Purpose** Enables a Cortex-A9 processor to access the status of the private 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 Table 3-1 on page 3-10.

Figure 3-7 shows the ppi_status Register bit assignments.

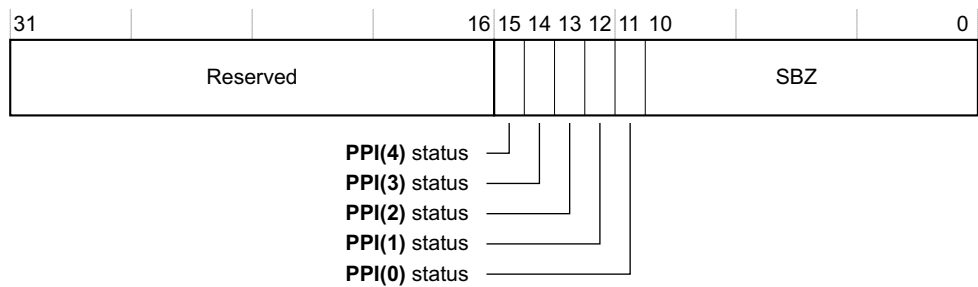


Figure 3-7 ppi_status Register bit assignments

Table 3-6 shows the register bit assignments.

Table 3-6 ppi_status Register bit assignments

Bits	Name	Description
[31:16]	-	Reserved
[15:11]	ppi_status	Returns the status of the PPI(4:0) inputs on the distributor: <ul style="list-style-type: none">• 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. <div>Note</div> These bits return the actual status of the PPI(4:0) signals. The pending_set and pending_clr Registers also can 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.

3.3.8 SPI Status Registers

The spi_status Register characteristics are:

- Purpose** Enables a Cortex-A9 processor to access the status of **IRQS[31:0]** 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 Table 3-1 on page 3-10.

Figure 3-8 shows the spi_status Register bit assignments.

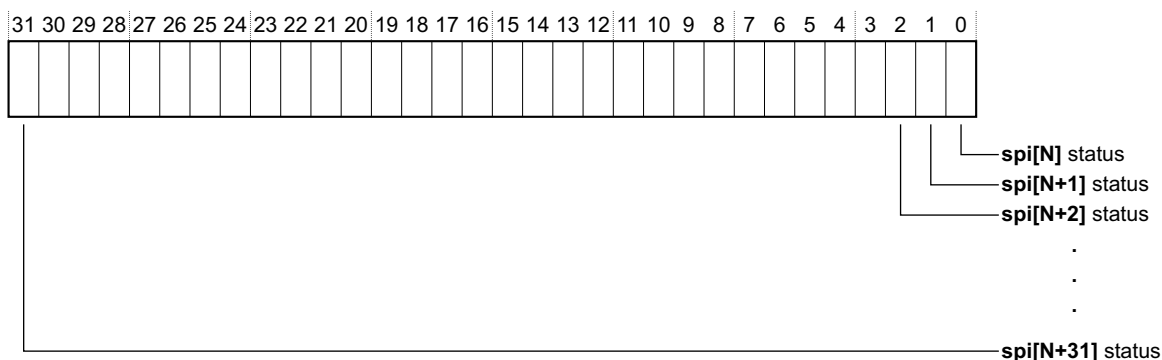


Figure 3-8 spi_status Register bit assignments

Table 3-7 shows the spi_status Register bit assignments.

Table 3-7 spi_status Register bit assignments

Bits	Name	Description
[31:0]	spi_status	<p>Returns the status of the IRQS[31:0] inputs on the distributor:</p> <p>Bit [X] = 0 IRQS[X] is LOW</p> <p>Bit [X] = 1 IRQS[X] is HIGH.</p> <p>———— Note ————</p> <p>The IRQS that X refers to depends on its bit position and the base address offset of the spi_status Register as Figure 4-28 on page 4-48 shows.</p> <p>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.</p>

Figure 3-9 shows the address map that the distributor provides for the SPIs.

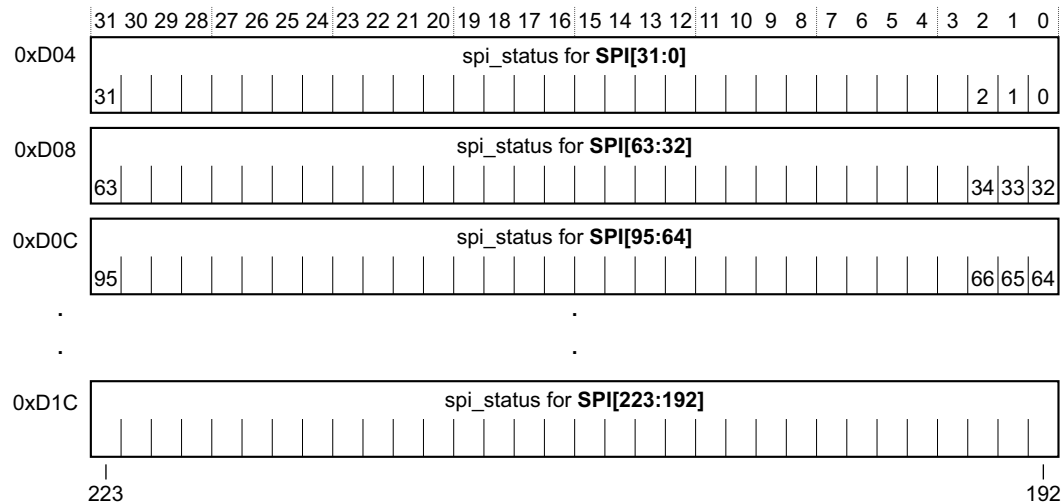


Figure 3-9 spi_status Register address map

In Figure 3-9 the values for the SPIs are read-only. The distributor provides a register at address offset 0xD04. This register contains the values for the SPIs for the corresponding Cortex-A9 processor interface. The distributor provides up to 7 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.

3.3.9 Peripheral Identification Registers

The `periph_id_[4:0]` Registers provide information about the configuration of the peripheral. Table 3-1 on page 3-10 shows the address base offset, reset value, and access type for these registers.

Each register provides eight bits of data but because some fields span across two adjacent `periph_id` registers then the following sections describe them:

- *periph_id_[3:0] register group*
- *periph_id_[7:4] register group* on page 3-21.

periph_id_[3:0] register group

Figure 3-10 on page 3-19 shows the `periph_id_[3:0]` register group bit assignments.

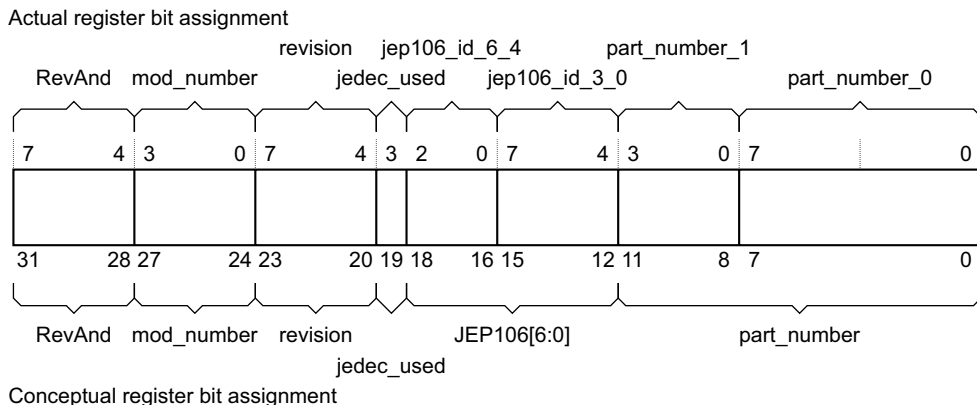
**Figure 3-10 periph_id[3:0] Register bit assignments**

Table 3-8 shows the periph_id[3:0] register group bit assignments.

Table 3-8 periph_id[3:0] Register bit assignments

Bits	Name	Description
[31:28]	RevAnd	These bits read as 0x0.
[27:24]	mod_number	Identifies data that is relevant to the ARM partner.
[23:20]	Architecture number	Identifies the major revision number, <i>n</i> , of the peripheral. The revision number starts from 0 and is revision-dependent.
[19]	jedec_used	Identifies if the IC uses the JEP106 manufacturer's identity code.
[18:12]	JEP106[6:0]	Identifies the designer. This is set to b0111011, to indicate that ARM designed the peripheral.
[11:0]	part_number	Identifies the peripheral.

The following subsections describe the periph_id[3:0] registers:

- *PrimeCell Identification Register 0* on page 3-23
- *PrimeCell Identification Register 1* on page 3-24
- *PrimeCell Identification Register 2* on page 3-24
- *PrimeCell Identification Register 3* on page 3-24.

Peripheral Identification Register 0

The `periph_id_0` Register is hard-coded and the fields in the register control the reset value. Table 3-9 shows the register bit assignments.

Table 3-9 `periph_id_0` Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined
[7:0]	<code>part_number_0</code>	-

Peripheral Identification Register 1

The `periph_id_1` Register is hard-coded and the fields in the register control the reset value. Table 3-10 shows the register bit assignments.

Table 3-10 `periph_id_1` Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined.
[7:4]	<code>jep106_id_3_0</code>	JEP106 identity code [3:0]. See the <i>JEP106, Standard Manufacturer's Identification Code</i> .
[3:0]	<code>part_number_1</code>	-

Peripheral Identification Register 2

The `periph_id_2` Register is hard-coded and the fields in the register control the reset value. Table 3-11 shows the register bit assignments.

Table 3-11 `periph_id_2` Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined.
[7:4]	Architecture number	The architecture number of the IC.
[3]	<code>jedec_used</code>	This indicates that the IC uses a manufacturer's identity code that was allocated by JEDEC according to JEP106.
[2:0]	<code>jep106_id_6_4</code>	JEP106 identity code [6:4]. See the <i>JEP106, Standard Manufacturer's Identification Code</i> .

Peripheral Identification Register 3

The `periph_id_3` Register is hard-coded and the fields in the register control the reset value. Table 3-12 shows the register bit assignments.

Table 3-12 `periph_id_3` Register bit assignments

Bits	Name	Description
[31:8]	-	Undefined.
[7:4]	RevAnd	-
[3:0]	mod_number	The customer can update this field if they modify the RTL of the Interrupt Controller.

`periph_id_[7:4]` register group

Figure 3-11 shows the `periph_id_[7:4]` register group bit assignments.

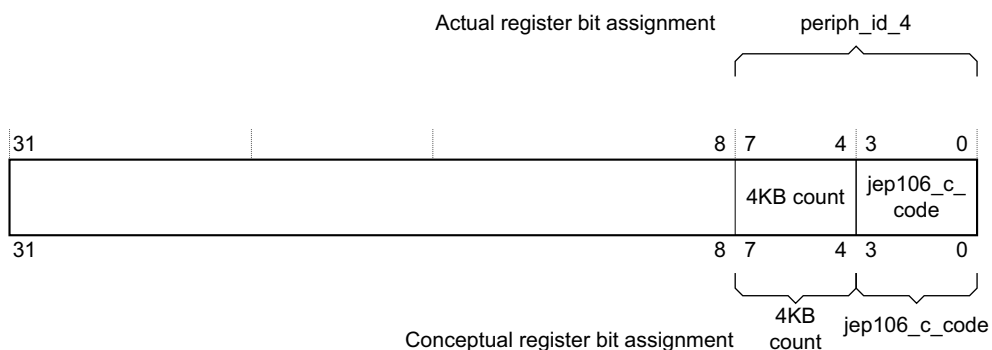


Figure 3-11 `periph_id_[7:4]` Register bit assignments

Table 3-13 shows the `periph_id_[7:4]` register group bit assignments.

Table 3-13 `periph_id_[7:4]` Register bit assignments

Bits	Name	Description
[31:8]	-	-
[7:4]	4KB count	Identifies the address space that the registers occupy.
[3:0]	jep106_c_code	Identifies the JEP106 continuation code.

The following subsection describe the `periph_id_[7:4]` register:

- *Peripheral Identification Register 4.*

Peripheral Identification Register 4

The `periph_id_4` Register is hard-coded and the fields in the register control the reset value. Table 3-14 shows the register bit assignments.

Table 3-14 `periph_id_4` Register bit assignments

Bits	Name	Description
[31:8]	-	Undefined.
[7:4]	4KB count	The number of 4KB address blocks you require, to access the registers, expressed in powers of 2.
[3:0]	jep106_c_code	The JEP106 continuation code value represents how many 0x7F continuation characters occur in the manufacturer’s identity code. See <i>JEP106, Standard Manufacturer’s Identification Code</i> .

3.3.10 PrimeCell Identification Registers

The `component_id_[3:0]` Registers are four eight-bit wide registers, that can conceptually be treated as a single register that holds a 32-bit PrimeCell ID value. You can use the register for automatic BIOS configuration. The `component_id` Register is set to 0xB105F00D.

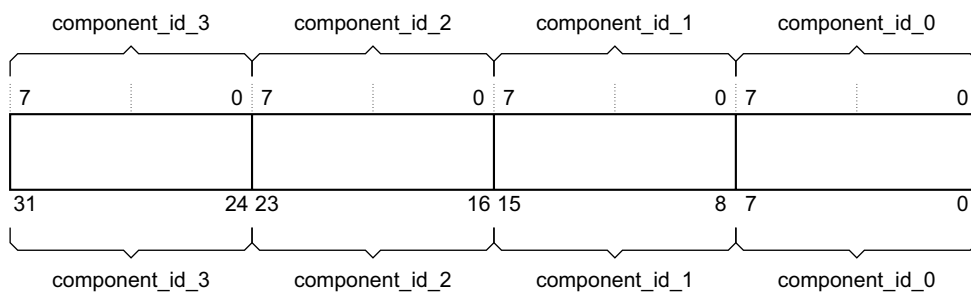
Table 3-15 lists the register bit assignments.

Table 3-15 `component_id` Register bit assignments

Bits	Reset value	Register	Bits	Description
-	-	<code>component_id_3</code>	[31:8]	Read undefined
[31:24]	0xB1	<code>component_id_3</code>	[7:0]	-
-	-	<code>component_id_2</code>	[31:8]	Read undefined
[23:16]	0x05	<code>component_id_2</code>	[7:0]	-
-	-	<code>component_id_1</code>	[31:8]	Read undefined
[15:8]	0xF0	<code>component_id_1</code>	[7:0]	-
-	-	<code>component_id_0</code>	[31:8]	Read undefined
[7:0]	0x0D	<code>component_id_0</code>	[7:0]	-

Figure 3-12 shows the register bit assignments.

Actual register bit assignment



Conceptual register bit assignment

Figure 3-12 PrimeCell ID Register bit assignments

The following subsections describe the component_id Registers:

- *PrimeCell Identification Register 0*
- *PrimeCell Identification Register 1* on page 3-24
- *PrimeCell Identification Register 2* on page 3-24
- *PrimeCell Identification Register 3* on page 3-24.

PrimeCell Identification Register 0

The component_id_0 Register is hard-coded and the fields in the register control the reset value. Table 3-16 lists the register bit assignments.

Table 3-16 component_id_0 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined
[7:0]	component_id_0	-

PrimeCell Identification Register 1

The component_id_1 Register is hard-coded and the fields in the register control the reset value. Table 3-17 lists the register bit assignments.

Table 3-17 component_id_1 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined
[7:0]	component_id_1	-

PrimeCell Identification Register 2

The component_id_2 Register is hard-coded and the fields in the register control the reset value. Table 3-18 lists the register bit assignments.

Table 3-18 component_id_2 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined
[7:0]	component_id_2	-

PrimeCell Identification Register 3

The component_id_3 Register is hard-coded and the fields in the register control the reset value. Table 3-19 lists the register bit assignments.

Table 3-19 component_id_3 Register bit assignments

Bits	Name	Description
[31:8]	-	Reserved, read undefined
[7:0]	component_id_3	-

3.4 Cortex-A9 processor interface register descriptions

This section shows the registers that each Cortex-A9 processor interface provides. Table 3-20 shows the Cortex-A9 processor interface registers. This section does not reproduce information about registers already described in the *ARM Generic Interrupt Controller Architecture Specification*.

Table 3-20 Cortex-A9 processor interface register summary

Base	Name	Type	Reset	Width	Description
0x000	ICPICR	RW	0x00000000	32	<i>Processor Interface Control Register</i>
0x004	ICCIPMR	RW	0x00000000	32	<i>Priority Mask Register</i>
0x008	ICCBPR	RW	0x2 0x3	32	<i>Binary Point Register</i>
0x00C	ICCIAR	RO	0x000003FF	32	<i>Interrupt Acknowledge Register</i>
0x010	ICCEOIR	WO	-	32	<i>End Of Interrupt Register</i>
0x014	ICCRPR	RO	0x000000FF	32	<i>Running Priority Register</i>
0x018	ICCHPIR	RO	0x000003FF	32	<i>Highest Pending Interrupt Register</i>
0x01C ^a	ICCABPR	RW	0x3	32	<i>Aliased Non-secure Binary Point Register</i>
0xFC	ICPIIR	RO	0x3901243B	32	<i>Processor Interface Implementer Identification Register</i>

a. This address location is only accessible when the Cortex-A9 processor performs a Secure access.

3.4.1 Processor Interface Implementer Identification Register

The ICPIIR Register characteristics are:

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 Table 3-20.

Figure 3-13 on page 3-26 shows the ICPIIR Register bit assignments.

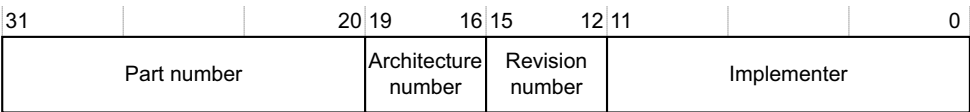


Figure 3-13 ICPIIR Register bit assignments

Table 3-21 shows the ICPIIR Register bit assignments.ss

Table 3-21 ICPIIR Register bit assignments

Bits	Name	Description
[31:20]	Part number	Identifies the peripheral.
[19:16]	Architecture version	Identifies the architecture version.
[15:12]	Revision number	Returns the revision number of the Interrupt Controller. The implementer defines the format of this field.
[11:0]	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:

- *About the private timer and watchdog blocks* on page 4-2
- *Private timer and watchdog registers* on page 4-3
- *About the Global Timer* on page 4-10
- *Global timer registers* on page 4-11.

4.1 About the private timer and watchdog blocks

The private timer and watchdog blocks have the following features:

- a 32-bit counter that generates an interrupt when it reaches zero
- configurable single-shot or auto-reload modes
- configurable starting values for the counter
- the clock for these blocks is the same clock as the Interrupt Controller clock.

The watchdog can be configured as a timer. See Chapter 5 *Clocks, Resets, and Power Management* for a description of **CLK**, **PERIPHCLK**, and **PERIPHCLKEN**.

4.1.1 Calculating timer intervals

The timer interval is calculated using the following equation:

$$\left(\frac{(\text{PRESCALER_value}+1) \times (\text{Load_value}+1)}{\text{PERIPHCLK}} \right)$$

This equation can be used to calculate the period between two events out of the timers and the watchdog time-out time.

4.1.2 Security extensions

See *SCU Non-secure Access Control Register* on page 2-12 for information about using timers in Secure or Non-secure state.

4.2 Private timer and watchdog registers

Addresses are relative to the base address of the timer and watchdog region defined by the private memory map. See *Interfaces* on page 1-7. 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. See *Resets and reset control signals* on page A-3.

Table 4-1 shows the timer and watchdog registers. All registers not described in Table 4-1 are Reserved.

Table 4-1 Timer and watchdog registers

Offset	Type	Reset Value	Description
0x00	RW	0x00000000	<i>Private Timer Load Register</i>
0x04	RW	0x00000000	<i>Private Timer Counter Register</i> on page 4-4
0x08	RW	0x00000000	<i>Private Timer Control Register</i> on page 4-4
0x0C	RW	0x00000000	<i>Private Timer Interrupt Status Register</i> on page 4-5
0x20	RW	0x00000000	<i>Watchdog Load Register</i> on page 4-6
0x24	RW	0x00000000	<i>Watchdog Counter Register</i> on page 4-6
0x28	RW	0x00000000	<i>Watchdog Control Register</i> on page 4-6
0x2C	RW	0x00000000	<i>Watchdog Interrupt Status Register</i> on page 4-7
0x30	RW	0x00000000	<i>Watchdog Reset Status Register</i> on page 4-9
0x34	WO	-	<i>Watchdog Disable Register</i> on page 4-9

Note

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

4.2.1 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.

4.2.2 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 the Cortex-A9 processor belonging to the timer is in debug state, the counter does not decrement until 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.

4.2.3 Private Timer Control Register

Figure 4-1 shows the Private Timer Control Register bit assignments.

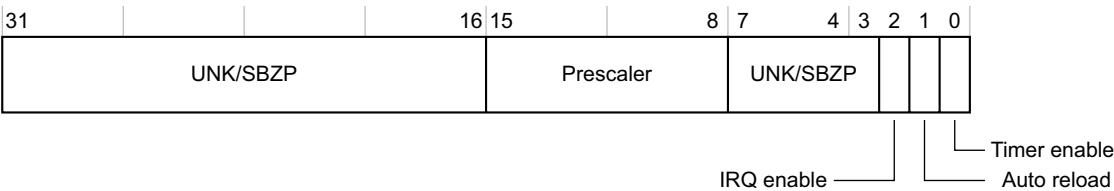


Figure 4-1 Private Timer Control Register bit assignments

Table 4-2 shows the Private Timer Control Register bit assignments.

Table 4-2 Private Timer Control Register bit assignments

Bits	Name	Description
[31:16]	-	UNK/SBZP.
[15:8]	Prescaler	The prescaler modifies the clock period for the decrementing event for the Counter Register. See <i>Calculating timer intervals</i> on page 4-2 for the equation.
[7:3]	-	UNK/SBZP.

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

Bits	Name	Description
[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.
[1]	Auto reload	1'b0 = Single shot mode. Counter decrements down to zero, sets the event flag and stops. 1'b1 = 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 1'b0 = Timer is disabled and the counter does not decrement. All registers can still be read and written 1'b1 = Timer is enabled and the counter decrements normally.

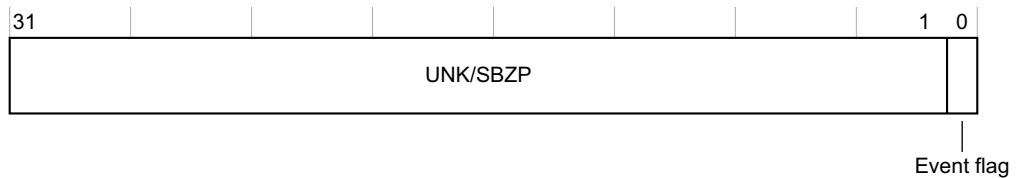
The timer is incremented every prescaler value+1. For example, if the prescaler has a value of five then the global timer is incremented every six clock cycles.

4.2.4 Private Timer Interrupt Status Register

Figure 4-2 shows the Private Timer Interrupt Status Register bit assignment.

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 4-2 Private Timer Interrupt Status Register bit assignment**

4.2.5 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.

4.2.6 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 belonging to 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 current 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. The external reset source is then responsible for resetting all or part of the Cortex-A9 MPCore design.

4.2.7 Watchdog Control Register

Figure 4-3 on page 4-7 shows the Watchdog Control Register bit assignments.

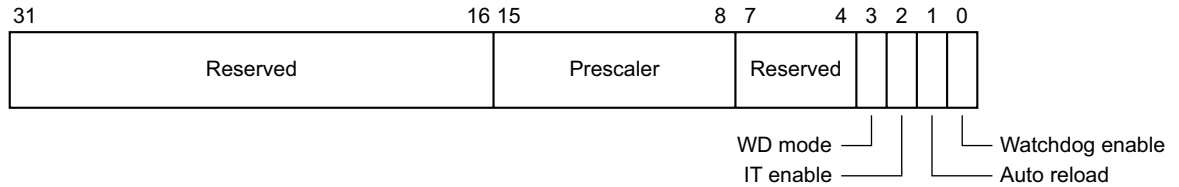
**Figure 4-3 Watchdog Control Register bit assignments**

Table 4-3 shows the Watchdog Control Register bit assignments.

Table 4-3 Watchdog Control Register bit assignments

Bits	Name	Description
[31:16]	-	Reserved.
[15:8]	Prescaler	The prescaler modifies the clock period for the decrementing event for the Counter Register. See <i>Calculating timer intervals</i> on page 4-2.
[7:4]	-	Reserved.
[3]	Watchdog mode	1'b0 = 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 <i>Watchdog Disable Register</i> on page 4-9. 1'b1 = 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	1'b0 = Single shot mode. Counter decrements down to zero, sets the event flag and stops. 1'b1 = 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 1'b0 = Watchdog is disabled and the counter does not decrement. All registers can still be read and /or written 1'b1 = Watchdog is enabled and the counter decrements normally.

4.2.8 Watchdog Interrupt Status Register

Figure 4-4 on page 4-8 shows the Watchdog Interrupt Status Register bit assignments.

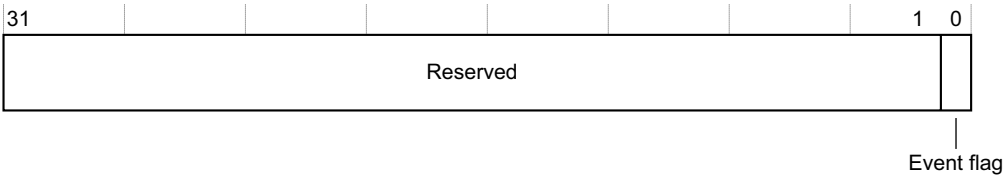


Figure 4-4 Watchdog Interrupt Status Register bit assignment

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.

4.2.9 Watchdog Reset Status Register

Figure 4-5 shows the Watchdog Reset Status Register bit assignment.

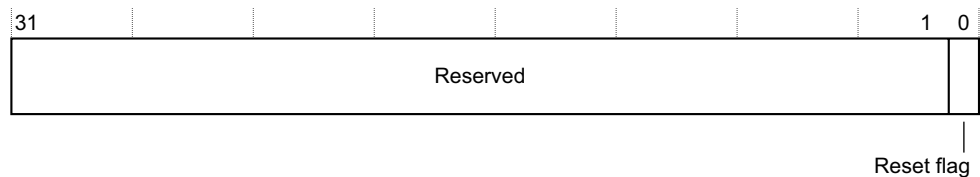


Figure 4-5 Watchdog Reset Status Register bit assignment

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

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 that 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.

4.2.10 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 its current state. To reactivate the Watchdog, the software must write 1 to the watchdog mode bit of the Watchdog Control Register. See *Watchdog Control Register* on page 4-6.

4.3 About the Global Timer

The global timer has the following features:

- The global timer is a 64-bit incrementing counter with an auto-incrementing feature.
- The global timer is memory mapped in the same address space as the private timers. See *Private Memory Region* on page 1-5.
- The global timer is accessed at reset in Secure State only. See *SCU Non-secure Access Control Register* on page 2-12.
- The global timer is accessible to all Cortex-A9 processors. Each Cortex-A9 processor has a 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. See *Interrupt Distributor interrupt sources* on page 3-3.
- The global timer is clocked by **PERIPHCLK**.

Note

The global timer does not stop counting when any of the CPUs are in debug state.

4.4 Global timer registers

Table 4-4 shows the 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	Description
0x00	R/W	0x00000000	<i>Global Timer Counter Registers, 0x00 and 0x04</i>
0x04	R/W	0x00000000	
0x08	R/W	0x00000000	<i>Global Timer Control Register on page 4-12</i>
0x0C	R/W	0x00000000	<i>Global Timer Interrupt Status Register on page 4-13</i>
0x10	R/W	0x00000000	<i>Comparator Value Registers, 0x10 and 0x14 on page 4-13</i>
0x14	R/W	0x00000000	
0x18	R/W	0x00000000	<i>Auto-increment Register, 0x18 on page 4-13</i>

4.4.1 Global Timer Counter Registers, 0x00 and 0x04

There are two timer counter registers. They are the lower 32-bit timer counter and the upper 32-bit timer counter.

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

To modify the register proceed as follows:

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
4. Set the timer enable bit.

To get the value from the Global Timer Counter register proceed as follows:

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 back to step 2. Otherwise the 64-bit timer counter value is correct.

4.4.2 Global Timer Control Register

Figure 4-6 shows the Global Timer Control Register bit assignments.

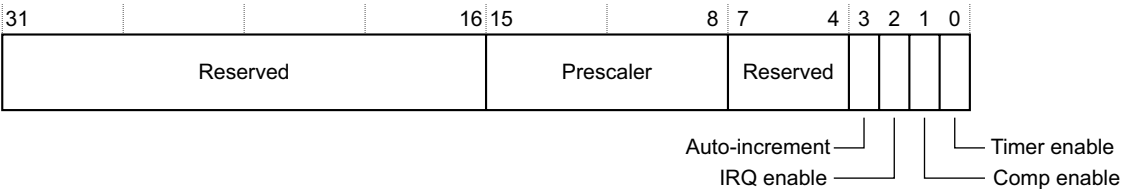


Figure 4-6 Global Timer Control Register bit assignments

Table 4-5 shows the Global Timer Control Register bit assignments.

Table 4-5 Global Timer Control Register bit assignments

Bits	Name	Description
[31:16]	-	Reserved
[15:8]	Prescaler	The prescaler modifies the clock period for the decrementing event for the Counter Register. See <i>Calculating timer intervals</i> on page 4-2 for the equation.
[7:4]	-	Reserved
[3]	Auto-increment ^a	This bit is banked per Cortex-A9 processor. 1'b0: single shot mode. When the counter reaches the comparator value, sets the event flag. It is the responsibility of software to update the comparator value to get further events. 1'b1: auto increment mode. Each time the counter reaches the comparator value, the comparator register is incremented with the auto-increment register, so that further 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.
[1]	Comp Enable ^a	This bit is banked per Cortex-A9 processor. If set, it allows the comparison between the 64-bit Timer Counter and the related 64-bit Comparator Register.
[0]	Timer Enable	Timer enable 1'b0 = Timer is disabled and the counter does not increment. All registers can still be read and written 1'b1 = Timer is enabled and the counter increments normally.

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

4.4.3 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 4-7 shows the Global Timer Interrupt Status Register bit assignment.

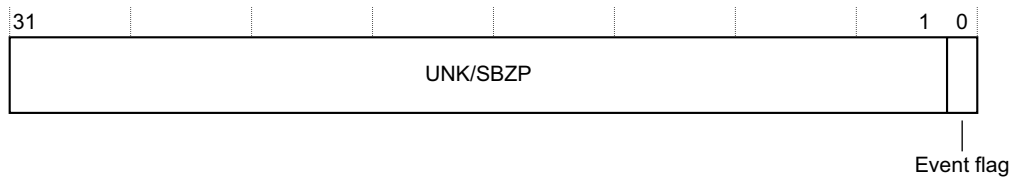


Figure 4-7 Global Timer Interrupt Status Register bit assignment

4.4.4 Comparator Value Registers, 0x10 and 0x14

There are two 32-bit registers, the lower 32-bit comparator value register and the upper 32-bit comparator value register.

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.

To ensure that updates to this register do not set the Interrupt Status Register proceed as follows:

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.

4.4.5 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:

- *Clocks* on page 5-2
- *Resets* on page 5-3
- *Power management* on page 5-6.

5.1 Clocks

The Cortex-A9 MPCore processor has these functional clock inputs:

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 watchdog 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.

Figure 5-1 shows an example with the **PERIPHCLK** clock period N as three.

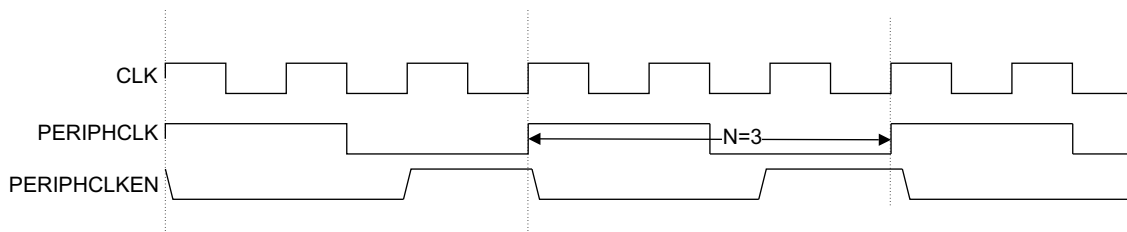


Figure 5-1 Three-to-one timing ratio

5.2 Resets

The reset signals present in the Cortex-A9 MPCore processor design enable you to reset different parts of the design independently. Table 5-1 shows the different reset combinations that can be expected in a Cortex-A9 MPCore system. [n] refers to the Cortex-A9 processor that initiates a reset

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

	Cortex-A9 MPCore		Individual CPUs			Cortex-A9 MPCore debug	Individual CPUs	
	power on	soft ware	power on	soft ware	power on SIMD MPE		debug	watch dog flag
nSCURESET and nPERIPHRESET	0	0	1	1	1	1	1	1
nCPURESET[3:0]	All 0	All 0	[n]=0	[n]=0	All 1	All 1	All 1	All 1
nNEONRESET[3:0]	All 0	All 0	[n]=0	[n]=0	[n]=0	All 1	All 1	All 1
nDBGRESET[3:0]	All 0	All 1	[n]=0	All 1	All 1	All 0	[n]=0	All 1
nWDRESET[3:0]	All 0	All 0	[n]=0 or all 1	[n]=0 or all 1	All 1	All 1	All 1	[n] = 0

The following sections describe the reset combinations:

- *Cortex-A9 MPCore power-on reset*
- *Cortex-A9 MPCore software reset* on page 5-4
- *Individual CPU power-on reset* on page 5-4
- *Individual CPU software reset* on page 5-4
- *Individual CPU power-on SIMD MPE reset* on page 5-4
- *Cortex-A9 MPCore debug reset* on page 5-4
- *Individual CPU debug reset* on page 5-5
- *Individual CPU watchdog flag reset* on page 5-5.

5.2.1 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.

5.2.2 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.

5.2.3 Individual CPU 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 powerdown or dormant state.

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

5.2.4 Individual CPU 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.

5.2.5 Individual CPU 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.

5.2.6 Cortex-A9 MPCore debug reset

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

5.2.7 Individual CPU debug reset

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

5.2.8 Individual CPU 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

This section describes Cortex-A9MPCore power management. It contains the following sections:

- *Individual Cortex-A9 processor power management*
- *Communication to the Power Management Controller* on page 5-10
- *IEM Support* on page 5-10
- *Cortex-A9 MPCore power domains* on page 5-10
- *Multiprocessor bring-up* on page 5-11.

5.3.1 Individual Cortex-A9 processor power management

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
- WFI 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.

Table 5-2 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

Table 5-2 Cortex-A9 MPCore power modes (continued)

Mode	Cortex-A9 processor logic	RAM arrays	Wake-up mechanism
WFI/WFE	Powered-up Only wake-up logic clocked	Powered-up	Wake-up on interrupts (external or timer/WD). L1 memory system only wake-up in case of SCU coherency request.
Dormant	Powered-off	Retention state/voltage	External wake-up event to power controller, which can perform a reset of the processor.
Shutdown	Powered-off	Powered-off	External wake-up event to power controller, which 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 in conjunction with CPU WFI entry flag to signal to the power controller the power domain that it can cut, using the PWRCTL bus. See *SCU CPU Power Status Register* on page 2-7.

Run mode

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

Wait for interrupt (WFI/WFE)

Wait for Interrupt mode disables all the clocks of a Cortex-A9 processor, while keeping its logic powered up. This reduces the power drawn to the static leakage current.

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

- an interrupt, masked or unmasked
- an imprecise data abort, regardless of the value of the CPSRA bit
- a debug request, regardless of whether debug is enabled
- a cp15 maintenance request from another processor
- a reset.

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

- an interrupt, unless masked
- a debug request, regardless of whether debug is enabled
- a cp15 maintenance request from another processor
- a previous exception return on the same processor
- a reset

- the assertion of the **EVENTI** input signal
- the execution of an SEV instruction on any processor in a multiprocessor system.

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

Entry into WFI Mode is performed by executing the Wait For Interrupt instruction.

Entry into WFE Mode is performed by executing the Wait For Event instruction.

To ensure that the memory system is not affected by the entry into the Standby state, the software must perform a Data Synchronization Barrier operation. This ensures that all explicit memory accesses occurring in program order before the WFI has completed. This prevents any possible deadlocks that could be caused in a system where memory access triggers or enables an interrupt that the CPU is waiting for.

On WFI/WFE the hardware ensures that:

- Any other memory accesses that have been started at the time that the WFI instruction is executed complete as normal. This ensures that the Level 2 memory system does not see any disruption caused by the WFI.
- The debug bus remains active throughout a WFI.

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 of 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. See *SCU CPU Power Status Register* on page 2-7.
- 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 (see *SCU CPU Power Status Register* on page 2-7).
- On entry into Dormant mode, the Reset signal to the Cortex-A9 processor must be asserted by the external power control mechanism.

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.

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 (see *SCU CPU Power Status Register* on page 2-7). 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 **DEBUGCLAMP**, **CPUCLAMP[3:0]**, **DECLAMP[3:0]**, and **RAMCLAMP[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.

5.3.3 IEM Support

The IEM infrastructure is intended to be supported at the system level to enable you to choose the level to use in the SOC to separate different power domains.

There are placeholders between Cortex-A9 MPCore logic and RAM arrays so that implementation of level-shifters for these parts can be on a different power domain.

5.3.4 Cortex-A9 MPCore power domains

The Cortex-A9 MPCore processor can support up to fourteen power domains:

- 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.

Figure 5-2 shows the power domains and where placeholders are inserted for power domain isolation.

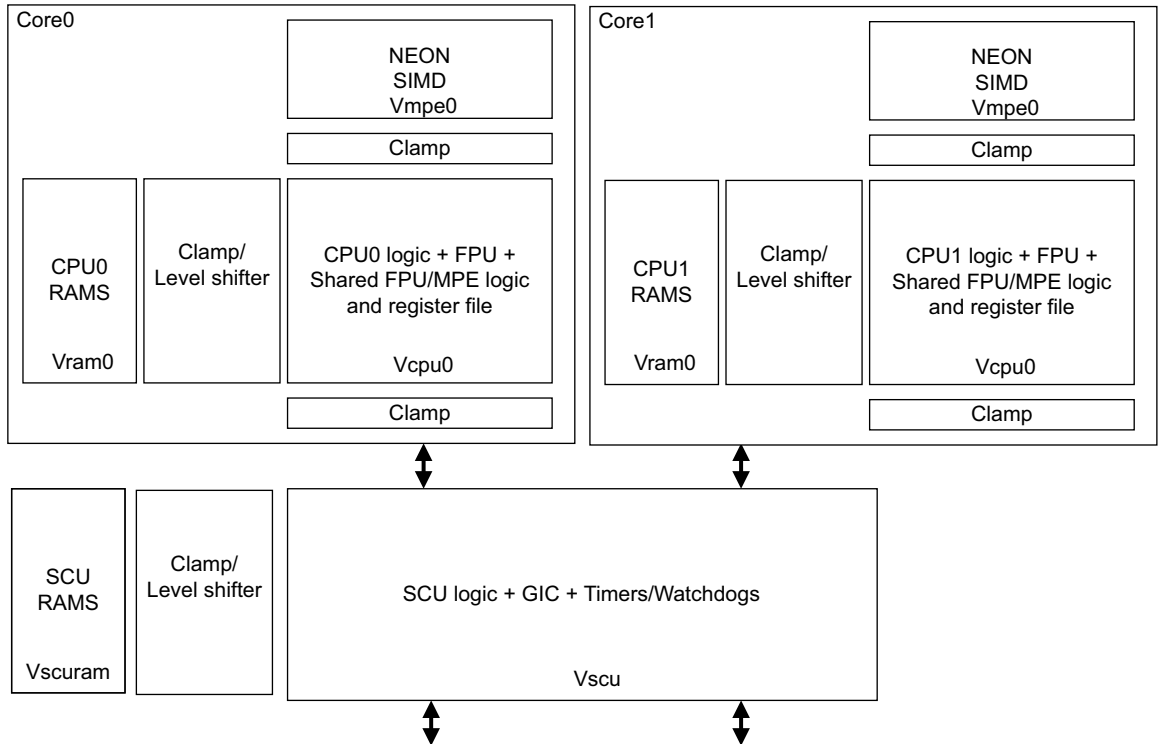


Figure 5-2 Cortex-A9 MPCore power domains and clamps

5.3.5 Multiprocessor bring-up

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.

For the lead processor:

1. Invalidate the data cache
2. Invalidate the SCU duplicate tags for all processors.
3. Invalidate the PL310, if used.
4. Enable the SCU
5. Enable the data cache
6. Enable the P310
7. Set SMP mode with ACTLR.SMP.

For non-lead processors:

1. Invalidate the data cache
2. Enable the data cache
3. Set SMP with ACTLR.SMP.

Chapter 6

Debug

This chapter describes some of the debug and trace considerations in Cortex-A9 MPCore designs. It contains the following sections:

- *External Debug Interface Signals* on page 6-2
- *Cortex-A9 MPCore APB Debug interface and memory map* on page 6-3.

6.1 External Debug Interface Signals

In the MPCore implementation, the debug interface of each individual Cortex-A9 processor is exposed externally, 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*.

Figure 6-1 shows the CortexA9 MP external debug interface signals.

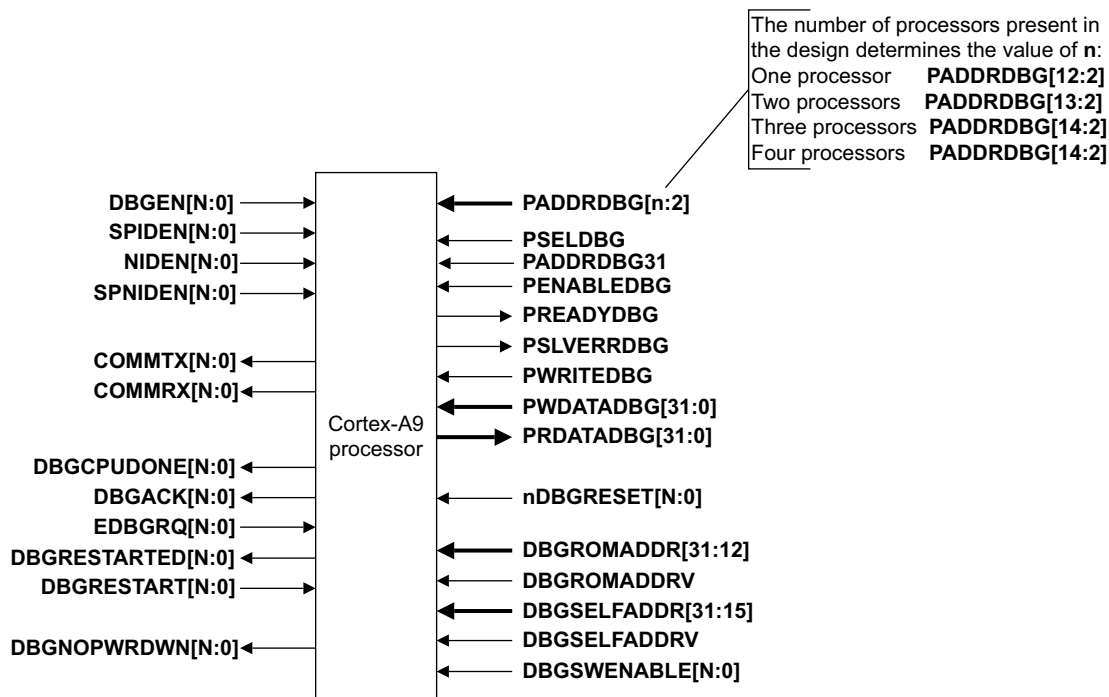


Figure 6-1 External debug interface signals in CortexA9 MPCore designs

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. See *Cortex-A9 MPCore APB Debug interface and memory map* on page 6-3.

The CortexA9 MPCore external debug interface does not implement:

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

6.2 Cortex-A9 MPCore APB Debug interface and memory map

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 Cortex-A9 TRM for detailed memory mapping of this 8KB memory region.

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. The following sections describe the uses of **PADDRDBG** in the following Cortex-A9 MPCore configurations:

- *A single Cortex-A9 processor configuration*
- *Two Cortex-A9 processors configuration*
- *Three Cortex-A9 processors configuration on page 6-4*
- *Four Cortex-A9 processors configuration on page 6-4.*

6.2.1 A single Cortex-A9 processor configuration

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.

6.2.2 Two Cortex-A9 processors configuration

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

6.2.3 Three Cortex-A9 processors configuration

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.

6.2.4 Four Cortex-A9 processors configuration

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 signaling signal names such as **TEINIT[N:0]**, the value of N is one less than the number of processors in your design. The appendix contains the following sections:

- *Clock and clock control signals* on page A-2
- *Resets and reset control signals* on page A-3
- *Interrupts* on page A-4
- *Configuration signals* on page A-5
- *Standby and wait for event signals* on page A-7
- *Power management signals* on page A-8
- *AXI interfaces* on page A-10
- *Performance monitoring signals* on page A-22
- *Exception flags signals* on page A-23
- *Parity error signals* on page A-24
- *MBIST interface* on page A-25
- *Scan test signal* on page A-26
- *External Debug interface* on page A-27
- *PTM interface signals* on page A-31.

A.1 Clock and clock control signals

Table A-1 shows the 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

See Chapter 5 *Clocks, Resets, and Power Management*.

A.2 Resets and reset control signals

Table A-2 shows the 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]^a	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. Only if an MPE is present

Table A-3 shows the clock control signals that are used to cut the clocks during reset sequences. **NEONCLKOFF** is only present when there is a Data Engine in your design. See Chapter 5 *Clocks, Resets, and Power Management*.

Table A-3 Reset clock control signals

Name	I/O	Source	Description
CPUCLKOFF	I	Reset controller	Individual Cortex-A9 Processor CPU clock enable, active-LOW. 0 = clock is enabled 1 = clock is stopped.
NEONCLKOFF	I		MPE SIMD logic clock control: 0 = do not cut MPE SIMD logic clock 1 = cut MPE SIMD logic clock.

Table A-4 shows 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

See Chapter 4 *Global timer, Private timers, and Watchdog registers*.

A.3 Interrupts

Table A-5 shows the interrupt line signals.

Table A-5 Interrupt line signals

Name	I/O	Source	Description
IRQS[x:0] ^a	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 <i>Interrupt Controller</i> .
nIRQ[N:0] ^a	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. The nIRQ input must be asserted until the processor acknowledges the interrupt.
nFIQ[N:0] ^a	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. The nFIQ input must be 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.

a. The minimum pulse width of signals driving external interrupt lines is one **PERIPHCLK** cycle.

A.4 Configuration signals

Table A-6 shows the configuration signals.

Table A-6 Configuration signals

Name	I/O	Source or destination	Description
CFGEND[N:0]	I	System configuration	<p>Individual Cortex-A9 processor endianness configuration.</p> <p>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.</p> <p>0 = EE bit is LOW</p> <p>1 = EE bit is HIGH</p> <p>This pin is only sampled during reset of the processor.</p>
CFGNMFI[N:0]	I		<p>Individual Cortex-A9 processor configuration of fast interrupts to be nonmaskable:</p> <p>0 = clear the NMFI bit in the CP15 c1 Control Register</p> <p>1 = set the NMFI bit in the CP15 c1 Control Register.</p> <p>This pin is only sampled during reset of the processor.</p>
CLUSTERID[3:0]	I		Value read in Cluster ID register field, bits[11:8] of the MPIDR.
FILTEREN	I		<p>For use with configurations with two master ports.</p> <p>Enables filtering of address ranges at reset. See <i>SCU Control Register</i> on page 2-3 for information on setting this signal.</p>
FILTERSTART[31:20]	I		<p>For use with configurations with two master ports.</p> <p>Specifies the start address for address filtering at reset. See <i>Filtering Start Address Register</i> on page 2-9.</p>
FILTEREND[31:20]	I		<p>For use with configurations with two master ports.</p> <p>Specifies the end address for address filtering. See <i>Filtering End Address Register</i> on page 2-10.</p>
PERIPHBASE[31:13]	I		<p>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.</p>

Table A-6 Configuration signals (continued)

Name	I/O	Source or destination	Description
SMPnAMP[N:0]	O	System integrity controller	Signals AMP or SMP mode for each Cortex-A9 processor. 0 = Asymmetric 1 = Symmetric.
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 SCTL.R.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 SCTL.R.V.

Table A-7 shows the 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 <i>Using CFGSDISABLE</i> on page 3-5.
CP15SDISABLE[N:0]	I		Individual Cortex-A9 Processor write access disable for some system control processor registers.

A.5 Standby and wait for event signals

Table A-8 shows the standby and wait for event 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 wake-up from WFE state.
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 state. 0 = processor not in standby state 1 = processor in standby state.
STANDBYWFI[N:0]	O		Indicates that a Cortex-A9 processor is in Standby WFI state. 0 = processor not in wait for event state 1 = processor in wait for event state.

See *Individual Cortex-A9 processor power management* on page 5-6.

A.6 Power management signals

Table A-9 shows 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]^a	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[1:0].
PWRCTLI1[1:0]	I		Reset value for CPU0 status field (bits [9:8]) of SCU CPU Power Status Register. [9:8].
PWRCTLI2[1:0]	I		Reset value for CPU0 status field (bits [17:16]) of SCU CPU Power Status Register. [17:16].
PWRCTLI3[1:0]	I		Reset value for CPU0 status field (bits [25:24]) of SCU CPU Power Status Register. [25:24].
PWRCTLO0[1:0]	O		b0x CPU0 must be powered on b10 CPU0 can enter dormant mode b11 CPU0 can enter powered-off mode.
PWRCTLO1[1:0]	O		b0x CPU1 must be powered on b10 CPU1 can enter dormant mode b11 CPU1 can enter powered-off mode. This signal exists only if CPU1 is present.

Table A-9 Power control interface signals (continued)

Name	I/O	Source or Destination	Description
PWRCTLO2[1:0]	O	Power controller	b0x CPU2 must be powered on b10 CPU2 can enter dormant mode b11 CPU2 can enter powered-off mode. This signal exists only if CPU2 is present.
PWRCTLO3[1:0]	O		b0x CPU3 must be powered on b10 CPU3 can enter dormant mode b11 CPU3 can enter powered-off mode This signal exists only if CPU3 is present.
SCUIDLE	O	PL310 or power controller	Indicates that the SCU is in IDLE state. This signal can be used to control clock or power-gating control signals of the modules connected to the SCU. In the case of the PL310, the SCUIDLE output of the Cortex-A9 MPCore can be connected to the STOPCLK input of the PL310.

a. Only if an MPE is present

See *SCU CPU Power Status Register* on page 2-7. See also *Communication to the Power Management Controller* on page 5-10.

A.7 AXI interfaces

In Cortex-A9 designs there can be two AXI master ports and an Accelerator Coherence Port, an AXI slave. The following sections describe the AXI interfaces:

- *AXI Master0 signals*
- *AXI Master1 signals* on page A-15
- *AXI ACP signals* on page A-16.

A.7.1 AXI Master0 signals

The following sections describe the AXI Master0 interface signals:

- *Write address signals for AXI Master0*
- *Write data channel signals* on page A-12
- *Write response channel signals* on page A-12
- *Speculative read interface signals for M0* on page A-14
- *Read data channel signals* on page A-13
- *Read data channel signals* on page A-15
- *AXI Master0 Clock enable signals* on page A-15.

Write address signals for AXI Master0

Table A-10 shows the 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	Pl310 or other system AXI devices	Address.
AWBURSTM0[1:0]	O		Burst type: b01 = INCR incrementing burst b10 = WRAP Wrapping burst. In the case of writes from the ACP the burst type could be STATIC (burst = 00) and this value can be forwarded to AXI ports. Other values are Reserved.
AWCACHM0[3:0]	O		Cache type giving additional information about cacheable characteristics.
AWIDM0[5:0]	O		Request ID See <i>AWIDMx[5:0] encodings</i> on page 2-17.

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

Name	I/O	Source or Destination	Description
AWLENM0[3:0]	O	PL310 or other system AXI devices	The number of data transfers that can occur within each burst. Each burst can be 1-16 transfers long: b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers b0011 = 4 data transfers.
AWLOCKM0[1:0]	O		Lock type: b00 = normal access b01 = exclusive access b10 = locked access.
AWPROTM0[2:0]	O		Protection Type.
AWREADYM0	I		Address ready.
AWSIZEM0[1:0]	O		Burst size: b000 = 8-bit transfer b001 = 16-bit transfer b010 = 32-bit transfer b011 = 64-bit transfer.
AWUSERM0[8:0]	O		[8] early BRESP . Used with the PL310. [7] full line of zeros. Used with the PL310. [6] clean eviction. [5] level 1 eviction. [4:1] inner attributes. See <i>AWUSERMx[8:0] encodings</i> on page 2-19. [0] shared.
AWVALIDM0	O		Address valid.

Write data channel signals

Table A-11 shows the 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	Pl310 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

Table A-12 shows the 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	Pl310 or other system AXI devices	Response ID
BREADYM0	O		Response ready
BRESPM0[1:0]	I		Write response
BVALIDM0	I		Response valid

Read data channel signals

Table A-13 shows the 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	PI310 or other system AXI devices	Address
ARBURSTM0[1:0]	O		Burst type: b01 = INCR incrementing burst b10 = WRAP Wrapping burst. Other values are Reserved.
ARCACHEM0[3:0]	O		Cache type giving additional information about cacheable characteristics.
ARIDM0[5:0]	O		Request ID See <i>ARIDMx[5:0]</i> encodings on page 2-17.
ARLENM0[3:0]	O		Burst length that gives the exact number of transfers: b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers b0011 = 4 data transfers.
ARLOCKM0[1:0]	O		Lock type: b00 = normal access b01 = exclusive access b10 = locked access.
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
ARSIZE0[1:0]	O	PL310 or other system AXI devices	Burst size: b000 = 8-bit transfer b001 = 16-bit transfer b010 = 32-bit transfer b011 = 64-bit transfer.
ARUSER0[6:0]	O		Sideband information: [6] Speculative linefill (Used with PL310) [5] prefetch hint (Used with PL310) [4:1] inner attributes b0000 = Strongly-ordered b0001 = Device b0011 = Normal Memory Non-Cacheable b0110 = Write-Through b0111 = Write-Back no Write Allocate b1111 = Write-Back Write Allocate. [0] shared bit. See <i>ARUSERMx[6:0] encodings</i> on page 2-18.
ARVALID0	O		Address valid.

Speculative read interface signals for M0

Table A-14 shows the interface signals on M0 for speculative read accesses between Cortex-A9MPCore and PL310.

Table A-14 PL310 signals on M0

Name	I/O	Source	Description
SRENDM0[3:0]	I	PL310	Speculative linefill confirmations from PL310.
SRIDM0[23:0]	I		Speculative confirmed IDs from PL310

Read data channel signals

Table A-15 shows the read data signals for AXI Master0.

Table A-15 Read data signals for AXI Master0

Name	I/O	Source or destination	Description
RVALIDM0	I	PI310 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

Table A-16 shows the 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: <ul style="list-style-type: none"> integer ratios of the system clock half integer ratios of the system clock. See <i>Interfaces</i> on page 1-7.
OUTCLKENM0	I		Clock enable for the AXI bus that enables the AXI interface to operate at either: <ul style="list-style-type: none"> integer ratios of the system clock half integer ratios of the system clock. See <i>Interfaces</i> on page 1-7.

A.7.2 AXI Master1 signals

In designs which 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 as well as to the Speculative Read Interface signals **SREND** and **SRID**.

A.7.3 AXI ACP signals

The following sections describe the AXI ACP interface signals:

- *Write address signals for AXI ACP* on page A-17
- *Write data channel signals* on page A-18
- *Write response channel signals* on page A-19
- *Read data channel signals* on page A-19
- *Read data channel signals* on page A-21
- *ACLKENS* on page A-21.

Write address signals for AXI ACP

Table A-17 shows the 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: b01 = INCR incrementing burst b10 = WRAP Wrapping burst. Other values are Reserved.
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. Each burst can be 1-16 transfers long: b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers b0011 = 4 data transfers.
AWLOCKS[0]	I		Lock type: b00 = normal access b01 = exclusive access. Bit [1] is unused. Tie off LOW.
AWPROTS[2:0]	I		Protection Type.
AWREADYS	O		Address ready.

Table A-17 Write address signals for AXI ACP (continued)

Name	I/O	Source or destination	Description
AWSIZES[1:0]	I	External AXI master	Burst size: b000 = 8-bit transfer b001 = 16-bit transfer b010 = 32-bit transfer b011 = 64-bit transfer.
AWUSERS[4:0]	I		Sideband information: [4:1] inner attributes: b0000 = Strongly-ordered b0001 = Device b0011 = Normal Memory Non-Cacheable b0110 = Write-Through b0111 = Write-Back no Write Allocate b1111 = Write-Back Write Allocate [0] shared. See <i>AXI USER attributes encodings</i> on page 2-18.
AWVALIDS	I		Address valid.

Write data channel signals

Table A-18 shows the 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

Table A-19 shows the 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 data channel signals

Table A-20 shows the 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: b01 = INCR incrementing burst b10 = WRAP Wrapping burst. The burst type can be STATIC (burst = 00) and this value can be forwarded to AXI ports. Other values are Reserved.
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. Each burst can be 1-16 transfers long: b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers b0011 = 4 data transfers.

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: b00 = normal access b01 = exclusive access b10 = locked access.
ARPROTS[2:0]	I		Protection Type
ARREADY	O		Address ready
ARSIZE[1:0]	I		Burst size: b000 = 8-bit transfer b001 = 16-bit transfer b010 = 32-bit transfer b011 = 64-bit transfer.
ARUSERS[4:0]	I		Sideband information: [4:1] inner attribute bits: b0000 = Strongly-ordered b0001 = Device b0011 = Normal Memory Non-Cacheable b0110 = Write-Through b0111 = Write-Back no Write Allocate b1111 = Write-Back Write Allocate. [0] shared bit. See <i>AXI USER attributes encodings</i> on page 2-18.
ARVALID	I		Address valid.

Read data channel signals

Table A-21 shows the 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
RREADY	I		Read ready

ACLKENS

Table A-22 shows the **ACLKEN** slave signal.

Table A-22 ACLKENS signal

Name	I/O	Source or destination	Description
ACLKEN	I	Clock controller	Bus clock enable. See <i>ACP interface clocking</i> on page 2-23.

A.8 Performance monitoring signals

Table A-23 shows the 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	PerformanceMonitoring Unit (PMU) or External Performance Monitoring Unit	Performance Monitoring Unit event bus for CPUTn. 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.9 Exception flags signals

Table A-24 shows the **DEFLAGS** and **SCUEVABORT** signals.

Table A-24 Exception flags signals

Name	I/O	Destination	Description
DEFLAGS_n[6:0]	O	System integrity controller	<p>Data Engine output flags. Only implemented if the Cortex-A9 processor includes a Data Engine.</p> <p>If the DE is NEON SIMD unit:</p> <ul style="list-style-type: none"> • 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]. <p>If the DE is FPU:</p> <ul style="list-style-type: none"> • 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 eviction.

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

A.10 Parity error signals

Table A-25 shows 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.
PARITYFAILSCU[N:0]	O		Parity output pin from the SCU tag RAMs. ORed output from each Cortex-A9 processor present in the design.

A.11 MBIST interface

Table A-26 shows the 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.

The size of some MBIST signals depends on whether the implementation has parity support or not. Table A-27 shows these signals with parity support implemented.

Table A-27 MBIST signals with parity support implemented

Name	I/O	Source or destination	Description
MBISTBE[32:0]	I	MBIST controller	MBIST write enable.
MBISTINDATA[71:0]	I		MBIST data in.
MBISTOUTDATA[287:0]	O		MBIST data out.

Table A-28 shows these signals without parity support implemented.

Table A-28 MBIST signals without parity support implemented

Name	I/O	Source or destination	Description
MBISTBE[25:0]	I	MBIST controller	MBIST write enable.
MBISTINDATA[63:0]	I		MBIST data in.
MBISTOUTDATA[255:0]	O		MBIST data out.

See *Cortex-A9 MBIST Controller TRM*.

A.12 Scan test signal

Table A-29 shows the 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.13 External Debug interface

The following sections describe the external debug interface signals:

- *Authentication interface*
- *APB interface signals* on page A-28
- *Cross trigger interface signals* on page A-29
- *Miscellaneous debug interface signals* on page A-29.

A.13.1 Authentication interface

Table A-30 shows the 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.13.2 APB interface signals

Table A-31 shows the 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.
PWDATADBGB[31:0]	I		Write data bus.
PWRITEDBG	I		APB read and write signal.
PRDATADBGB[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.13.3 Cross trigger interface signals

Table A-32 shows the 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.13.4 Miscellaneous debug interface signals

Table A-33 on page A-30 shows the 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 this signal off 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 this signal off to zero.
DBGSELFADDRV	I		Valid signal for DBGSELFADDR . If the offset cannot be determined tie this signal LOW.

A.14 PTM interface signals

Table A-34 shows the 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
WPTFIFOEMPTY_n	O	PTM device	There are no speculative waypoints in the PTM interface FIFO.
WPTCOMMIT_n[1:0]	O		Number of waypoints committed this cycle. It is valid to indicate a valid waypoint and commit it in the same cycle.
WPTCONTEXTID_n[31:0]	O		Context ID for the waypoint. This signal must be true regardless of the condition code of the waypoint.
WPTENABLE_n	I		Enable waypoint. When set, enables the Cortex-A9 processor to output waypoints.
WPTEXCEPTIONTYPE_n[3:0]	O		Exception type: b0001 = Halting Debug b0010 = Secure Monitor b0100 = Imprecise Data Abort b0101 = T2EE trap b1000 = Reset b1001 = UNDEF b1010 = SVC b1011 = Prefetch abort/Software Breakpoint b1100 = Precise data abort/software watchpoint b1110 = IRQ b1111 = FIQ.
WPTFLUSH_n	O		Flush signal from core exception FIFO. All as yet uncommitted waypoints are flushed.
WPTLINK_n	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 current 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
WPTTRACEPROHIBITED_n	O	PTM device	<p>Trace is prohibited for the current waypoint target.</p> <p>Indicates entry to prohibited region. No more waypoints are traced until trace can resume.</p> <p>Indication that PTM clocks can be stopped.</p> <p>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.</p> <p>Only one WPTVALID cycle can be seen with WPTTRACEPROHIBITED set.</p> <p>Trace stops with this waypoint and the next waypoint seen is an Isync packet.</p>
WPTTYPE_n[2:0]	O		<p>Waypoint Type.</p> <p>b000 = Direct Branch</p> <p>b001 = Indirect Branch</p> <p>b010 = Exception</p> <p>b011 = DMB</p> <p>b100 = Debug entry/Trace prohibited</p> <p>b101 = Debug exit (require addresses of first instruction)</p> <p>b110 = Invalid</p> <p>b111 = Invalid.</p> <p>Must only take valid states when WPTVALID is HIGH.</p> <p>Debug Entry must be followed by Debug Exit.</p> <p>———— Note ————</p> <p>Debug exit does not reflect the execution of an instruction.</p>
WPTVALID_n	O		Waypoint is confirmed as valid.

Appendix B

Revisions

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

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 <i>Interrupt Controller</i> .
Parity error option added	Table 1-1 on page 1-4.
Clarify the role of the SCU with reference to data coherency and the non-support of instruction cache coherency	<i>About the SCU</i> on page 2-2.
Added information about exclusive accesses and address filtering	<i>Address filtering</i> on page 2-2.

Table B-2 Differences between issue A and issue B (continued)

Change	Location
SSAC description corrected	<i>SCU Non-secure Access Control Register</i> on page 2-12.
SSAC bit assignments corrected	Table 2-9 on page 2-13.
Change STI, Software Triggered Interrupt, to SGI, Software Generated Interrupt	Throughout Chapter 3 <i>Interrupt Controller</i> .
INTID descriptions extended and clarified	Throughout Chapter 3 <i>Interrupt Controller</i> .
Reset information added	Timer and watchdog registers on page 5-3.
AXI transaction IDs section extended	AXI transaction IDs on page 6-3.
AXI USER encodings section added	AXI USER encodings on page 6-5.
EVENTI information extended and EVENTO information added	WFE/SEV synchronization on page 6-9.
CLUSTERID[3:0] description corrected	<i>Configuration signals</i> on page A-5.
DBGEN[3:0] description added	Table A-30 on page A-27.

Differences between issue B and issue C

Table B-3 Differences between issue B and issue C

Change	Location
Design changes listed	<i>Product revisions</i> on page 1-10.
New entries in the Private Memory map	Table 2-2 on page 2-3.
Timers and watchdogs renamed Private timers and watchdogs	Table 2-2 on page 2-3.
TLB size added as a configurable option	Table 1-1 on page 1-4.
Timing diagrams added	Figure 1-4 on page 1-25, Figure 1-5 on page 1-25, Figure 1-6 on page 1-26, and Figure 1-7 on page 1-26.
CPUCLKOFF and DECLKOFF added to Power-on reset	<i>Cortex-A9 MPCore reset</i> on page 1-28. <i>Configuration signals</i> on page A-5.
Change in SCU Power Status Register layout	<i>SCU CPU Power Status Register</i> on page 2-7
Additional PPI. There are five PPIs per Cortex-A9 processor interface	<i>Interrupt types and sources</i> on page 3-2.

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

Change	Location
PPI(4) added to the PPI Status Register	<i>PPI Status Register</i> on page 3-15.
INT renamed IRQS	<i>SPI Status Registers</i> on page 3-17. <i>Interrupts</i> on page A-4.
Chapter 5 renamed. It was “Private timers and Watchdog Registers”.	Chapter 4 <i>Global timer, Private timers, and Watchdog registers</i> .
L2 interface chapter included in Chapter 1	
nIRQOUT[N:0] and nFIQOUT[N:0] added	<i>Interrupts</i> on page A-4.
MAXCLKLATENCY[2:0] added	<i>Configuration signals</i> on page A-5.
BISTCLAMP removed	<i>Power management signals</i> on page A-8.
AXI descriptions corrected and extended	<i>AXI interfaces</i> on page A-10.
AXI Master1 descriptions removed.	
AWLOCKS[1:0] corrected to AWLOCKS[0] .	Table A-17 on page A-17.
ARIDS[5:0] corrected to ARIDS[2:0] .	Table A-20 on page A-19.
Performance monitoring signals extended and new signals added.	<i>Performance monitoring signals</i> on page A-22.
SCUEVABORT moved to Performance Monitoring from Parity error signals section.	<i>Performance monitoring signals</i> on page A-22.
SCANMODE removed	<i>Scan test signal</i> on page A-26.
PRDATADBГ corrected to PRDATADBГ[31:0]	Table A-31 on page A-28.
WPTT32nT16n changed to WPT32LINKn	Table A-34 on page A-31.

Differences between issue C and issue D.

Table B-4 Differences between issue C and issue D

Change	Location
ARM Architecture Reference Manual promoted to head of ARM publications	<i>ARM publications</i> on page xvii
Global timer re-positioned. Other timers re-named private timers.	Figure 1-1 on page 1-3
<i>Compliance</i> content moved and extended	<i>About Cortex-A9 MPCore coherency</i> on page 1-8
Features list removed	
Configurable options includes Preload Engine options and ARM_BIST	<i>Configurable options</i> on page 1-4
<i>Interfaces</i> section extended	<i>Interfaces</i> on page 1-7
<i>Private Memory Region</i> chapter removed	
<i>Private Memory Region</i> content re-arranged. Table added	<i>Private Memory Region</i> on page 1-5
SLVERR changed to DECERR	Table 1-2 on page 1-5
<i>Interfaces</i> section extended	<i>Interfaces</i> on page 1-7
<i>MPCore Considerations</i> section added	<i>MPCore considerations</i> on page 1-8
ACP behavior description moved and extended	<i>Accelerator Coherency Port</i> on page 2-24
Design changes list extended	<i>Product revisions</i> on page 1-10.
<i>Snoop Control Unit</i> chapter updated and extended to include detailed interface descriptions	Chapter 2 <i>Snoop Control Unit</i>
SCU Register updates	Table 2-1 on page 2-3
Interfaces	<i>SCU Control Register</i> on page 2-3
Removal of content that repeats GIC Architecture content	Chapter 3 <i>Interrupt Controller</i>
Re-organization of remaining <i>Interrupt Controller</i> content	
Timer control register descriptions extended	<i>Private Timer Control Register</i> on page 4-4
Section added	<i>Global Timer Control Register</i> on page 4-12
Resets descriptions revised and extended	<i>Resets</i> on page 5-3
Signals lists updated	Source or destination column added to all signal lists

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

Change	Location
nNEONRESET[N:0] replaces nDERESET[N:0]	Table A-2 on page A-3
NEONCLCKOFF replaces DECLCKOFF	Table A-6 on page A-5
CPUCLCKOFF[N:0] replaces CPUCLOCKOFF[N:0]	
CP15 c15 Configuration Base Address Register replaces System Control Configbase Register	
NEONCLAMP replaces DECLAMP	Table A-9 on page A-8
Power control signal descriptions corrected and clarified.	
SCUIDLE signal added	
Duplicated AXI user encodings removed	Table A-10 on page A-10
ARUSERM0[6:0] corrected	Table A-13 on page A-13
Speculative read interface signals section added	<i>Speculative read interface signals for M0</i> on page A-14
[4:0] in AWUSERS[4:0] corrected to [4:1]	Table A-17 on page A-17
NEON SIMD unit replaces MPE	Table A-23 on page A-22
PMUEVENT size become 57 bits	
DEFLAGS and SCUEVABORT have a separate table	<i>Exception flags signals</i> on page A-23
PARITYSCU[3:0] becomes PARITYFAILSCU[N:0]	Table A-25 on page A-24
MBISTBE[31:0] becomes MBISTBE[32:0]	Table A-27 on page A-25
Description of DBGSWENABLE[N:0] amended	Table A-33 on page A-30
DBGSELFADDR bits corrected to [31:15]	
WPTCOMMITn bits corrected to [1:0]	Table A-34 on page A-31
WPTENABLE corrected to WPTENABLEn	
WPT32LINKn corrected to WPTT32LINKn	
Statement about WPTTARGETTBIT removed	

Table B-5 Differences between Issue D and Issue E

Change	Location
No technical change	-

Glossary

This glossary describes some of the terms used in ARM manuals. Where terms can have several meanings, the meaning presented here is intended.

Abort

A mechanism that indicates to a core that the value associated with a memory access is invalid. An abort can be caused by the external or internal memory system as a result of attempting to access invalid instruction or data memory. An abort is classified as either a Prefetch or Data Abort, and an internal or External Abort.

See also Data Abort, External Abort and Prefetch Abort.

Advanced eXtensible Interface (AXI)

A bus protocol that supports separate address/control and data phases, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels to enable low-cost DMA, ability to issue multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure. The AXI protocol also includes optional extensions to cover signaling for low-power operation.

AXI is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.

Advanced High-performance Bus (AHB)

A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol. The full AMBA AHB protocol specification includes a number of features that are not commonly required for master and slave IP developments and ARM Limited recommends only a subset of the protocol is usually used. This subset is defined as the AMBA AHB-Lite protocol.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

Advanced Microcontroller Bus Architecture (AMBA)

A family of protocol specifications that describe a strategy for the interconnect. AMBA is the ARM open standard for on-chip buses. It is an on-chip bus specification that details a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules.

Advanced Peripheral Bus (APB)

A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to reduce system power consumption.

AHB

See Advanced High-performance Bus.

AHB Access Port (AHB-AP)

An optional component of the DAP that provides an AHB interface to a SoC.

AHB-AP

See AHB Access Port.

Aligned

A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.

AMBA

See Advanced Microcontroller Bus Architecture.

Advanced Trace Bus (ATB)

A bus used by trace devices to share CoreSight capture resources.

APB

See Advanced Peripheral Bus.

Architecture

The organization of hardware and/or software that characterizes a processor and its attached components, and enables devices with similar characteristics to be grouped together when describing their behavior, for example, Harvard architecture, instruction set architecture, ARMv6 architecture.

ARM instruction A word that specifies an operation for an ARM processor to perform. ARM instructions must be word-aligned.

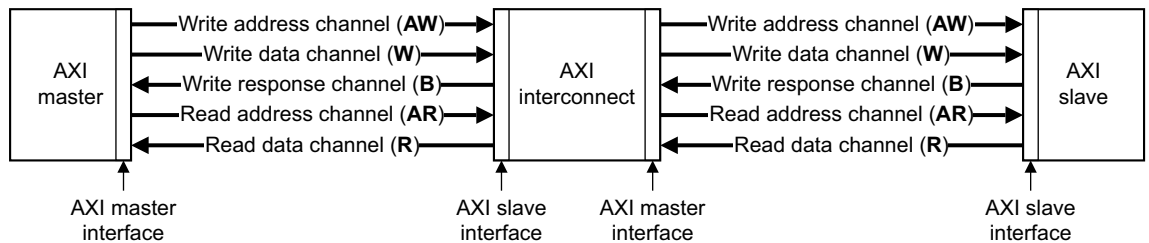
ARM state A processor that is executing ARM (32-bit) word-aligned instructions is operating in ARM state.

AXI See Advanced eXtensible Interface.

AXI channel order and interfaces

The block diagram shows:

- the order that AXI channel signals are described in
- the master and slave interface conventions for AXI components.



AXI terminology The following AXI terms are general. They apply to both masters and slaves:

Active read transaction

A transaction for which the read address has transferred, but the last read data has not yet transferred.

Active transfer

A transfer for which the **xVALID**¹ handshake has asserted, but for which **xREADY** has not yet asserted.

Active write transaction

A transaction for which the write address or leading write data has transferred, but the write response has not yet transferred.

Completed transfer

A transfer for which the **xVALID/xREADY** handshake is complete.

1. The letter **x** in the signal name denotes an AXI channel as follows:

AW	Write address channel.
W	Write data channel.
B	Write response channel.
AR	Read address channel.
R	Read data channel.

Payload	The non-handshake signals in a transfer.
Transaction	An entire burst of transfers, comprising an address, one or more data transfers and a response transfer (writes only).
Transmit	An initiator driving the payload and asserting the relevant xVALID signal.
Transfer	A single exchange of information. That is, with one xVALID/xREADY handshake.

The following AXI terms are master interface attributes. To obtain optimum performance, they must be specified for all components with an AXI master interface:

Combined issuing capability

The maximum number of active transactions that a master interface can generate. This is specified instead of write or read issuing capability for master interfaces that use a combined storage for active write and read transactions.

Read ID capability

The maximum number of different **ARID** values that a master interface can generate for all active read transactions at any one time.

Read ID width

The number of bits in the **ARID** bus.

Read issuing capability

The maximum number of active read transactions that a master interface can generate.

Write ID capability

The maximum number of different **AWID** values that a master interface can generate for all active write transactions at any one time.

Write ID width

The number of bits in the **AWID** and **WID** buses.

Write interleave capability

The number of active write transactions for which the master interface is capable of transmitting data. This is counted from the earliest transaction.

Write issuing capability

The maximum number of active write transactions that a master interface can generate.

The following AXI terms are slave interface attributes. To obtain optimum performance, they must be specified for all components with an AXI slave interface

Combined acceptance capability

The maximum number of active transactions that a slave interface can accept. This is specified instead of write or read acceptance capability for slave interfaces that use a combined storage for active write and read transactions.

Read acceptance capability

The maximum number of active read transactions that a slave interface can accept.

Read data reordering depth

The number of active read transactions for which a slave interface can transmit data. This is counted from the earliest transaction.

Write acceptance capability

The maximum number of active write transactions that a slave interface can accept.

Write interleave depth

The number of active write transactions for which the slave interface can receive data. This is counted from the earliest transaction.

Beat

Alternative word for an individual transfer within a burst. For example, an INCR4 burst comprises four beats.

See also Burst.

Block address

An address that comprises a tag, an index, and a word field. The tag bits identify the way that contains the matching cache entry for a cache hit. The index bits identify the set being addressed. The word field contains the word address that can be used to identify specific words, halfwords, or bytes within the cache entry.

See also Cache terminology diagram on the last page of this glossary.

Burst

A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AHB buses are controlled using the **HBURST** signals to specify if transfers are single, four-beat, eight-beat, or 16-beat bursts, and to specify how the addresses are incremented.

See also Beat.

Byte	An 8-bit data item.
Byte-invariant	<p>In a byte-invariant system, the address of each byte of memory remains unchanged when switching between little-endian and big-endian operation. When a data item larger than a byte is loaded from or stored to memory, the bytes making up that data item are arranged into the correct order depending on the endianness of the memory access. The ARM architecture supports byte-invariant systems in ARMv6 and later versions. When byte-invariant support is selected, unaligned halfword and word memory accesses are also supported. Multi-word accesses are expected to be word-aligned.</p> <p><i>See also</i> Word-invariant.</p>
Byte lane strobe	An AHB signal, HBSTRB , that is used for unaligned or mixed-endian data accesses to determine which byte lanes are active in a transfer. One bit of HBSTRB corresponds to eight bits of the data bus.
Cache	<p>A block of on-chip or off-chip fast access memory locations, situated between the processor and main memory, used for storing and retrieving copies of often used instructions and/or data. This is done to greatly reduce the average speed of memory accesses and so to increase processor performance.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
Cache contention	When the number of frequently-used memory cache lines that use a particular cache set exceeds the set-associativity of the cache. In this case, main memory activity increases and performance decreases.
Cache hit	A memory access that can be processed at high speed because the instruction or data that it addresses is already held in the cache.
Cache line	<p>The basic unit of storage in a cache. It is always a power of two words in size (usually four or eight words), and is required to be aligned to a suitable memory boundary.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
Cache line index	<p>The number associated with each cache line in a cache way. Within each cache way, the cache lines are numbered from 0 to (set associativity) -1.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
Cache lockdown	To fix a line in cache memory so that it cannot be overwritten. Cache lockdown enables critical instructions and/or data to be loaded into the cache so that the cache lines containing them are not subsequently reallocated. This ensures that all subsequent accesses to the instructions/data concerned are cache hits, and therefore complete as quickly as possible.
Cache miss	A memory access that cannot be processed at high speed because the instruction/data it addresses is not in the cache and a main memory access is required.

Cache set	<p>A cache set is a group of cache lines (or blocks). A set contains all the ways that can be addressed with the same index. The number of cache sets is always a power of two.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
Cache way	<p>A group of cache lines (or blocks). It is 2 to the power of the number of index bits in size.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
Clean	<p>A cache line that has not been modified while it is in the cache is said to be clean. To clean a cache is to write dirty cache entries into main memory. If a cache line is clean, it is not written on a cache miss because the next level of memory contains the same data as the cache.</p> <p><i>See also</i> Dirty.</p>
Coherency	<i>See</i> Memory coherency.
Cold reset	<p>Also known as power-on reset. Starting the processor by turning power on. Turning power off and then back on again clears main memory and many internal settings. Some program failures can lock up the processor and require a cold reset to enable the system to be used again. In other cases, only a warm reset is required.</p> <p><i>See also</i> Warm reset.</p>
Communications channel	<p>The hardware used for communicating between the software running on the processor, and an external host, using the debug interface. When this communication is for debug purposes, it is called the Debug Comms Channel. In an ARMv6 compliant core, the communications channel includes the Data Transfer Register, some bits of the Data Status and Control Register, and the external debug interface controller, such as the DBGTap controller in the case of the JTAG interface.</p>
Condition field	A four-bit field in an instruction that specifies a condition under which the instruction can execute.
Conditional execution	If the condition code flags indicate that the corresponding condition is true when the instruction starts executing, it executes normally. Otherwise, the instruction does nothing.
Context	<p>The environment that each process operates in for a multitasking operating system. In ARM processors, this is limited to mean the Physical Address range that it can access in memory and the associated memory access permissions.</p> <p><i>See also</i> Fast context switch.</p>

Control bits	The bottom eight bits of a Program Status Register (PSR). The control bits change when an exception arises and can be altered by software only when the processor is in a privileged mode.
Coprocessor	A processor that supplements the main processor. It carries out additional functions that the main processor cannot perform. Usually used for floating-point math calculations, signal processing, or memory management.
Copy back	<i>See</i> Write-back.
Core	A core is that part of a processor that contains the ALU, the datapath, the general-purpose registers, the Program Counter, and the instruction decode and control circuitry.
Core module	In the context of an ARM Integrator, a core module is an add-on development board that contains an ARM processor and local memory. Core modules can run standalone, or can be stacked onto Integrator motherboards.
Core reset	<i>See</i> Warm reset.
CPSR	<i>See</i> Current Program Status Register
Current Program Status Register (CPSR)	The register that holds the current operating processor status.
Data Abort	<p>An indication from a memory system to a core that it must halt execution of an attempted illegal memory access. A Data Abort is attempting to access invalid data memory.</p> <p><i>See also</i> Abort, External Abort, and Prefetch Abort.</p>
Data cache	A block of on-chip fast access memory locations, situated between the processor and main memory, used for storing and retrieving copies of often used data. This is done to greatly reduce the average speed of memory accesses and so to increase processor performance.
DBGTAP	<i>See</i> Debug Test Access Port.
Debugger	A debugging system that includes a program, used to detect, locate, and correct software faults, together with custom hardware that supports software debugging.
Dirty	<p>A cache line in a write-back cache that has been modified while it is in the cache is said to be dirty. A cache line is marked as dirty by setting the dirty bit. If a cache line is dirty, it must be written to memory on a cache miss because the next level of memory contains data that has not been updated. The process of writing dirty data to main memory is called cache cleaning.</p> <p><i>See also</i> Clean.</p>

DNM	<i>See</i> Do Not Modify.
Do Not Modify (DNM)	In Do Not Modify fields, the value must not be altered by software. DNM fields read as Unpredictable values, and must only be written with the same value read from the same field on the same processor. DNM fields are sometimes followed by RAZ or RAO in parentheses to show which way the bits must read for future compatibility, but programmers must not rely on this behavior.
Doubleword	A 64-bit data item. The contents are taken as being an unsigned integer unless otherwise stated.
Doubleword-aligned	A data item having a memory address that is divisible by eight.
Exception	A fault or error event that is considered serious enough to require that program execution is interrupted. Examples include attempting to perform an invalid memory access, external interrupts, and undefined instructions. When an exception occurs, normal program flow is interrupted and execution is resumed at the corresponding exception vector. This contains the first instruction of the interrupt handler to deal with the exception.
Exception service routine	<i>See</i> Interrupt handler.
Exception vector	<i>See</i> Interrupt vector.
External Abort	An indication from an external memory system to a core that it must halt execution of an attempted illegal memory access. An External Abort is caused by the external memory system as a result of attempting to access invalid memory. <i>See also</i> Abort, Data Abort and Prefetch Abort.
Flat address mapping	A system of organizing memory in which each Physical Address contained within the memory space is the same as its corresponding Virtual Address.
Front of queue pointer	Pointer to the next entry to be written to in the write buffer.
Halfword	A 16-bit data item.
Halt mode	One of two mutually exclusive debug modes. In halt mode all processor execution halts when a breakpoint or watchpoint is encountered. All processor state, coprocessor state, memory and input/output locations can be examined and altered by the JTAG interface. <i>See also</i> Monitor debug-mode.

High vectors	Alternative locations for exception vectors. The high vector address range is near the top of the address space, rather than at the bottom.
Host	A computer that provides data and other services to another computer. Especially, a computer providing debugging services to a target being debugged.
IEEE 754 standard	<i>IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std. 754-1985.</i> The standard that defines data types, correct operation, exception types and handling, and error bounds for floating-point systems. Most processors are built in compliance with the standard in either hardware or a combination of hardware and software.
IEM	<i>See</i> Intelligent Energy Manager.
IGN	<i>See</i> Ignore.
Ignore (IGN)	Must ignore memory writes.
Illegal instruction	An instruction that is architecturally Undefined.
IMB	<i>See</i> Instruction Memory Barrier.
Implementation-defined	Means that the behavior is not architecturally defined, but must be defined and documented by individual implementations.
Implementation-specific	Means that the behavior is not architecturally defined, and does not have to be documented by individual implementations. Used when there are a number of implementation options available and the option chosen does not affect software compatibility.
Imprecise tracing	<p>A filtering configuration where instruction or data tracing can start or finish earlier or later than expected. Most cases cause tracing to start or finish later than expected.</p> <p>For example, if TraceEnable is configured to use a counter so that tracing begins after the fourth write to a location in memory, the instruction that caused the fourth write is not traced, although subsequent instructions are. This is because the use of a counter in the TraceEnable configuration always results in imprecise tracing.</p>
Index	<i>See</i> Cache index.
Index register	A register specified in some load or store instructions. The value of this register is used as an offset to be added to or subtracted from the base register value to form the virtual address, which is sent to memory. Some addressing modes optionally enable the index register value to be shifted prior to the addition or subtraction.

Instruction cache	A block of on-chip fast access memory locations, situated between the processor and main memory, used for storing and retrieving copies of often used instructions. This is done to greatly reduce the average speed of memory accesses and so to increase processor performance.
Instruction cycle count	The number of cycles for which an instruction occupies the Execute stage of the pipeline.
Instruction Memory Barrier (IMB)	An operation to ensure that the prefetch buffer is flushed of all out-of-date instructions.
Intelligent Energy Manager (IEM)	A technology that enables dynamic voltage scaling and clock frequency variation to be used to reduce power consumption in a device.
Internal scan chain	A series of registers connected together to form a path through a device, used during production testing to import test patterns into internal nodes of the device and export the resulting values.
Interrupt handler	A program that control of the processor is passed to when an interrupt occurs.
Interrupt vector	One of a number of fixed addresses in low memory, or in high memory if high vectors are configured, that contains the first instruction of the corresponding interrupt handler.
Invalidate	To mark a cache line as being not valid by clearing the valid bit. This must be done whenever the line does not contain a valid cache entry. For example, after a cache flush all lines are invalid.
Joint Test Action Group (JTAG)	The name of the organization that developed standard IEEE 1149.1. This standard defines a boundary-scan architecture used for in-circuit testing of integrated circuit devices. It is commonly known by the initials JTAG.
JTAG	<i>See</i> Joint Test Action Group.
Line	<i>See</i> Cache line.
Load/store architecture	A processor architecture where data-processing operations only operate on register contents, not directly on memory contents.
Load Store Unit (LSU)	The part of a processor that handles load and store transfers.
LSU	<i>See</i> Load Store Unit.

Macrocell	A complex logic block with a defined interface and behavior. A typical VLSI system comprises several macrocells (such as a processor, an ETM, and a memory block) plus application-specific logic.
Memory bank	One of two or more parallel divisions of interleaved memory, usually one word wide, that enable reads and writes of multiple words at a time, rather than single words. All memory banks are addressed simultaneously and a bank enable or chip select signal determines which of the banks is accessed for each transfer. Accesses to sequential word addresses cause accesses to sequential banks. This enables the delays associated with accessing a bank to occur during the access to its adjacent bank, speeding up memory transfers.
Memory coherency	A memory is coherent if the value read by a data read or instruction fetch is the value that was most recently written to that location. Memory coherency is made difficult when there are multiple possible physical locations that are involved, such as a system that has main memory, a write buffer and a cache.
Memory Management Unit (MMU)	Hardware that controls caches and access permissions to blocks of memory, and translates virtual addresses to physical addresses.
Microprocessor	<i>See</i> Processor.
Miss	<i>See</i> Cache miss.
MMU	<i>See</i> Memory Management Unit.
Modified Virtual Address (MVA)	A Virtual Address produced by the ARM processor can be changed by the current Process ID to provide a <i>Modified Virtual Address</i> (MVA) for the MMUs and caches. <i>See also</i> Fast Context Switch Extension.
Monitor debug-mode	One of two mutually exclusive debug modes. In Monitor debug-mode the processor enables a software abort handler provided by the debug monitor or operating system debug task. When a breakpoint or watchpoint is encountered, this enables vital system interrupts to continue to be serviced while normal program execution is suspended. <i>See also</i> Halt mode.
Multi-ICE	A JTAG-based tool for debugging embedded systems.
MVA	<i>See</i> Modified Virtual Address.
PA	<i>See</i> Physical Address.
Penalty	The number of cycles in which no useful Execute stage pipeline activity can occur because an instruction flow is different from that assumed or predicted.

Power-on reset	See Cold reset.
Prefetching	In pipelined processors, the process of fetching instructions from memory to fill up the pipeline before the preceding instructions have finished executing. Prefetching an instruction does not mean that the instruction must be executed.
Prefetch Abort	An indication from a memory system to a core that it must halt execution of an attempted illegal memory access. A Prefetch Abort can be caused by the external or internal memory system as a result of attempting to access invalid instruction memory. See also Data Abort, External Abort and Abort.
Processor	A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.
Physical Address (PA)	The MMU performs a translation on <i>Modified Virtual Addresses</i> (MVA) to produce the <i>Physical Address</i> (PA) which is given to AHB to perform an external access. The PA is also stored in the data cache to avoid the necessity for address translation when data is cast out of the cache. See also Fast Context Switch Extension.
Read	Reads are defined as memory operations that have the semantics of a load. That is, the ARM instructions LDM, LDRD, LDC, LDR, LDRT, LDRSH, LDRH, LDRSB, LDRB, LDRBT, LDREX, RFE, STREX, SWP, and SWPB, and the Thumb instructions LDM, LDR, LDRSH, LDRH, LDRSB, LDRB, and POP. Java instructions that are accelerated by hardware can cause a number of reads to occur, according to the state of the Java stack and the implementation of the Java hardware acceleration.
RealView ICE	A system for debugging embedded processor cores using a JTAG interface.
Region	A partition of instruction or data memory space.
Remapping	Changing the address of physical memory or devices after the application has started executing. This is typically done to enable RAM to replace ROM when the initialization has been completed.
Reserved	A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.

Saved Program Status Register (SPSR)

The register that holds the CPSR of the task immediately before the exception occurred that caused the switch to the current mode.

SBO *See* Should Be One.

SBZ *See* Should Be Zero.

SBZP *See* Should Be Zero or Preserved.

Scan chain A scan chain is made up of serially-connected devices that implement boundary scan technology using a standard JTAG TAP interface. Each device contains at least one TAP controller containing shift registers that form the chain connected between **TDI** and **TDO**, through which test data is shifted. Processors can contain several shift registers to enable you to access selected parts of the device.

SCREG The currently selected scan chain number in an ARM TAP controller.

Set *See* Cache set.

Set-associative cache

In a set-associative cache, lines can only be placed in the cache in locations that correspond to the modulo division of the memory address by the number of sets. If there are n ways in a cache, the cache is termed n -way set-associative. The set-associativity can be any number greater than or equal to 1 and is not restricted to being a power of two.

Should Be One (SBO)

Should be written as 1 (or all 1s for bit fields) by software. Writing a 0 produces Unpredictable results.

Should Be Zero (SBZ)

Should be written as 0 (or all 0s for bit fields) by software. Writing a 1 produces Unpredictable results.

Should Be Zero or Preserved (SBZP)

Should be written as 0 (or all 0s for bit fields) by software, or preserved by writing the same value back that has been previously read from the same field on the same processor.

SPSR *See* Saved Program Status Register

Synchronization primitive

The memory synchronization primitive instructions are those instructions that are used to ensure memory synchronization. That is, the LDREX, STREX, SWP, and SWPB instructions.

Tag	<p>The upper portion of a block address used to identify a cache line within a cache. The block address from the CPU is compared with each tag in a set in parallel to determine if the corresponding line is in the cache. If it is, it is said to be a cache hit and the line can be fetched from cache. If the block address does not correspond to any of the tags, it is said to be a cache miss and the line must be fetched from the next level of memory.</p> <p><i>See also</i> Cache terminology diagram on the last page of this glossary.</p>
TAP	<i>See</i> Test access port.
Test Access Port (TAP)	<p>The collection of four mandatory and one optional terminals that form the input/output and control interface to a JTAG boundary-scan architecture. The mandatory terminals are TDI, TDO, TMS, and TCK. The optional terminal is TRST. This signal is required in ARM cores because it is used to reset the debug logic.</p>
Thumb instruction	A halfword that specifies an operation for an ARM processor in Thumb state to perform. Thumb instructions must be halfword-aligned.
Thumb state	A processor that is executing Thumb (16-bit) halfword aligned instructions is operating in Thumb state.
TLB	<i>See</i> Translation Look-aside Buffer.
Translation Lookaside Buffer (TLB)	A cache of recently used translation table entries that avoid the overhead of translation table walking on every memory access. Part of the Memory Management Unit.
Translation table	A table, held in memory, that contains data that defines the properties of memory areas of various fixed sizes.
Translation table walk	The process of doing a full translation table lookup. It is performed automatically by hardware.
Trap	An exceptional condition in a VFP coprocessor that has the respective exception enable bit set in the FPSCR register. The user trap handler is executed.
Undefined	Indicates an instruction that generates an Undefined instruction trap. See the <i>ARM Architecture Reference Manual</i> for more details on ARM exceptions.
UNP	<i>See</i> Unpredictable.
Unpredictable	For reads, the data returned when reading from this location is unpredictable. It can have any value. For writes, writing to this location causes unpredictable behavior, or an unpredictable change in device configuration. Unpredictable instructions must not halt or hang the processor, or any part of the system.
VA	<i>See</i> Virtual Address.

Virtual Address (VA)

The MMU uses its translation tables to translate a Virtual Address into a Physical Address. The processor executes code at the Virtual Address, which might be located elsewhere in physical memory.

Warm reset

Also known as a core reset. Initializes the majority of the processor excluding the debug controller and debug logic. This type of reset is useful if you are using the debugging features of a processor.

Watchpoint

A watchpoint is a mechanism provided by debuggers to halt program execution when the data contained by a particular memory address is changed. Watchpoints are inserted by the programmer to enable inspection of register contents, memory locations, and variable values when memory is written to test that the program is operating correctly. Watchpoints are removed after the program is successfully tested. *See also* Breakpoint.

Way

See Cache way.

WB

See Write-back.

Word

A 32-bit data item.

Word-invariant

In a word-invariant system, the address of each byte of memory changes when switching between little-endian and big-endian operation, in such a way that the byte with address A in one endianness has address A EOR 3 in the other endianness. As a result, each aligned word of memory always consists of the same four bytes of memory in the same order, regardless of endianness. The change of endianness occurs because of the change to the byte addresses, not because the bytes are rearranged. The ARM architecture supports word-invariant systems in ARMv3 and later versions. When word-invariant support is selected, the behavior of load or store instructions that are given unaligned addresses is instruction-specific, and is in general not the expected behavior for an unaligned access. It is recommended that word-invariant systems use the endianness that produces the required byte addresses at all times, apart possibly from very early in their reset handlers before they have set up the endianness, and that this early part of the reset handler uses only aligned word memory accesses.

See also Byte-invariant.

Write

Writes are defined as operations that have the semantics of a store. That is, the ARM instructions SRS, STM, STRD, STC, STRT, STRH, STRB, STRBT, STREX, SWP, and SWPB, and the Thumb instructions STM, STR, STRH, STRB, and PUSH. Java instructions that are accelerated by hardware can cause a number of writes to occur, according to the state of the Java stack and the implementation of the Java hardware acceleration.

Write-back (WB)

In a write-back cache, data is only written to main memory when it is forced out of the cache on line replacement following a cache miss. Otherwise, writes by the processor only update the cache. (Also known as copyback).

Write buffer A block of high-speed memory, arranged as a FIFO buffer, between the data cache and main memory, whose purpose is to optimize stores to main memory.

Write completion The memory system indicates to the processor that a write has been completed at a point in the transaction where the memory system is able to guarantee that the effect of the write is visible to all processors in the system. This is not the case if the write is associated with a memory synchronization primitive, or is to a Device or Strongly-ordered region. In these cases the memory system might only indicate completion of the write when the access has affected the state of the target, unless it is impossible to distinguish between having the effect of the write visible and having the state of target updated.

This stricter requirement for some types of memory ensures that any side-effects of the memory access can be guaranteed by the processor to have taken place. You can use this to prevent the starting of a subsequent operation in the program order until the side-effects are visible.

Write-through (WT) In a write-through cache, data is written to main memory at the same time as the cache is updated.

WT *See* Write-through.

Cache terminology diagram

The diagram illustrates the following cache terminology:

- block address
- cache line
- cache set
- cache way
- index
- tag.

