RealView® Platform Baseboard for ARM11 MPCore

HBI-0159
HBI-0175
HBI-0176

User Guide

ARM®

ARM DUI 0351E

RealView Platform Baseboard for ARM11 MPCore
User Guide

Release Information

Change History

Date	Issue	Confidentiality	Change
October 2007	A	Non-Confidential	New document
May 2008	B	Non-Confidential	Fixes for Errata
March 2009	C	Non-Confidential	Fixes for Errata
July 2010	D	Non-Confidential	Document update
April 2011	E	Non-Confidential	Document update

Proprietary Notice

Confidentiality Status

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

Product Status

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

Web Address

`http://www.arm.com`

Conformance Notices

This section contains conformance notices.

Federal Communications Commission Notice

This device is test equipment and consequently is exempt from part 15 of the FCC Rules under section 15.103 (c).

CE Declaration of Conformity

The system should be powered down when not in use.

The PB11MPCore generates, uses, and can radiate radio frequency energy and may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment causes harmful interference to radio or television reception, which can be determined by turning the equipment off or on, you are encouraged to try to correct the interference by one or more of the following measures:

- ensure attached cables do not lie across the card
- reorient the receiving antenna
- increase the distance between the equipment and the receiver
- connect the equipment into an outlet on a circuit different from that to which the receiver is connected
- consult the dealer or an experienced radio/TV technician for help

Note

It is recommended that wherever possible shielded interface cables be used.

Contents
RealView Platform Baseboard for ARM11 MPCore User Guide

Preface
About this book xviii
Feedback xxvi

Chapter 1 Introduction
1.1 About the PB11MPCore 1-2
1.2 Precautions 1-6

Chapter 2 Getting Started
2.1 Setting up the baseboard 2-2
2.2 Boot Monitor configuration 2-4
2.3 JTAG debugger and USB config support 2-6
2.4 Baseboard configuration switches 2-7

Chapter 3 Hardware Description
3.1 Baseboard architecture 3-2
3.2 Tile interconnections 3-10
3.3 ARM11 MPCore test chip 3-12
3.4 Northbridge 3-15
3.5 Southbridge 3-20

3.6 Ethernet interface 3-30
3.7 USB Interface 3-31
3.8 USB config interface 3-32
3.9 DVI Interface 3-33
3.10 PCI interface 3-34
3.11 Power supply control 3-36
3.12 Clock architecture 3-38
3.13 Resets 3-45
3.14 Interrupts 3-48
3.15 Test, configuration, and debug interfaces 3-54

Chapter 4 Programmer's Reference
4.1 Memory map 4-3
4.2 Configuration and initialization 4-9
4.3 Status and system control registers 4-11
4.4 System Controller (SYSCTRL) 4-43
4.5 Advanced Audio CODEC Interface, AACI 4-46
4.6 Color LCD Controller, CLCDC 4-48
4.7 Single Master Direct Memory Access Controller, SMDMAC 4-50
4.8 Dynamic Memory Controller, DMC 4-51
4.9 Ethernet 4-52
4.10 General Purpose Input/Output, GPIO 4-53
4.11 Generic Interrupt Controller, GIC 4-55
4.12 Keyboard and Mouse Interface, KMI 4-60
4.13 MultiMedia Card Interface, MCI 4-61
4.14 AXI to PCI bridge 4-62
4.15 Real Time Clock, RTC 4-64
4.16 Two-wire serial bus interface, SBCon 4-65
4.17 Smart Card Interface, SCI 4-68
4.18 Synchronous Serial Port, SSP 4-69
4.19 Static Memory Controller, SMC 4-70
4.20 Timers 4-71
4.21 UART 4-72
4.22 USB interface 4-74
4.23 Watchdog 4-76
4.24 CompactFlash interface 4-77

Chapter 5 Processor Sub-System
5.1 ARM11 MPCore test chip overview 5-2
5.2 Clocks 5-4
5.3 Resets and interrupts 5-11
5.4 Power supply control 5-18
5.5 Memory configuration 5-20
5.6 L220 bypass and peripheral decode 5-22
5.7 Debug and configuration using JTAG 5-27

Appendix A Signal Descriptions
A.1 Compact Flash interface A-2
A.2 Audio CODEC interface A-6
A.3 MMC and SD card interface A-7
A.4 Keyboard and mouse interface A-9
A.5 GPIO interface A-10
A.6 UART interface A-11
A.7 Synchronous Serial Port interface A-12
A.8 Smart Card interface A-13
A.9 Ethernet interface A-15
A.10 USB interface A-16
A.11 DVI display interface A-17
A.12 RealView Logic Tile header connectors A-19
A.13 Test and debug connections A-41

Appendix B Specifications
B.1 Electrical Specification B-2
B.2 Timing specifications B-3

Appendix C Memory Expansion Boards
C.1 About memory expansion C-2
C.2 Fitting a memory board C-4
C.3 Connector pinout C-5

Appendix D RealView Logic Tile expansion
D.1 About the RealView Logic Tile D-2
D.2 Header connectors D-3

Appendix E Boot Monitor and platform library
E.1 About the Boot Monitor E-2
E.2 About the platform library E-3
E.3 Using the baseboard Boot Monitor and platform library E-4

Appendix F Boot Monitor Commands
F.1 About Boot Monitor commands F-2
F.2 Boot Monitor command set F-3

Appendix G Loading FPGA Images
G.1 General procedure G-2
G.2 Board files G-3
G.3 The progcards utilities G-5
G.4 Upgrading your hardware G-6

Glossary

List of Tables
RealView Platform Baseboard for ARM11 MPCore User Guide

	Change History	ii
Table 2-1	Boot Monitor startup behavior	2-4
Table 2-2	STDIO redirection	2-5
Table 2-3	Selecting the boot device	2-7
Table 2-4	Selecting fast mode	2-8
Table 2-5	ARM11 MPCore reset behavior	2-8
Table 2-6	Enable Level 2 Cache fix	2-8
Table 2-7	Number of ARM11 MPCore AXI ports	2-9
Table 3-1	FPGA image selection	3-21
Table 3-2	Serial interface device addresses	3-28
Table 3-3	Legacy Mode: INT[15:0]	3-50
Table 3-4	Legacy Mode: nIRQ and nFIQ	3-51
Table 3-5	New Mode with DCC: INT[15:0]	3-51
Table 3-6	New Mode with DCC: nIRQ and nFIQ	3-52
Table 3-7	New mode without DCC: INT[15:0]	3-52
Table 3-8	New Mode without DCC: nIRQ and nFIQ	3-53
Table 4-1	System memory map	4-3
Table 4-2	Memory map for standard peripherals	4-4
Table 4-3	Boot memory	4-9
Table 4-4	Memory chip selects and address range	4-10
Table 4-5	Register map for status and system control registers	4-11

Table 4-6 SYS_ID register bit assignments 4-15
Table 4-7 SYS_OSCx register bit assignments 4-17
Table 4-8 SYS_LOCK register bit assignments 4-18
Table 4-9 Flag registers 4-19
Table 4-10 SYS_RESETCTL register bit assignments 4-21
Table 4-11 SYS_MCI register bit assignment 4-22
Table 4-12 SYS_FLASH register bit assignments 4-23
Table 4-13 SYS_CLCD register bit register assignments 4-24
Table 4-14 SYS_MISC register bit assignment 4-27
Table 4-15 SYS_DMAPSR register bit assignments 4-28
Table 4-16 SYS_DMAPSR register bit coding 4-29
Table 4-17 SYS_PEX_STAT register bit assignments 4-30
Table 4-18 SYS_PCI_STAT register bit assignments 4-31
Table 4-19 SYS_PLD_CTRL1 register bit assignments 4-33
Table 4-20 SYS_PLD_CTRL2 register bit assignments 4-34
Table 4-21 SYS_PLD_INIT register bit assignments 4-36
Table 4-22 SYS_PROCID0 register bit assignments 4-38
Table 4-23 SYS_PROCID1 register bit assignments 4-39
Table 4-24 SYS_VOLTAGE_CTLx registers 4-41
Table 4-25 SYS_TEST_OSCx registers 4-42
Table 4-26 SYSCTRL implementation 4-43
Table 4-27 SYS_CTRL0 register 4-44
Table 4-28 SYS_CTRL1 register 4-45
Table 4-29 AACI implementation 4-46
Table 4-30 Modified AACI PeriphID3 register 4-47
Table 4-31 CLCDC implementation 4-48
Table 4-32 Values for different display resolutions 4-49
Table 4-33 SMDMAC implementation 4-50
Table 4-34 DMC implementation 4-51
Table 4-35 Ethernet implementation 4-52
Table 4-36 GPIO implementation 4-53
Table 4-37 GPIO2 and MCI status signals 4-53
Table 4-38 Generic Interrupt Controller implementation 4-55
Table 4-39 Interrupt signals to controllers 4-56
Table 4-40 KMI implementation 4-60
Table 4-41 MCI implementation 4-61
Table 4-42 AXI to PCI bridge implementation 4-62
Table 4-43 PCI bus memory map 4-62
Table 4-44 RTC implementation 4-64
Table 4-45 Serial bus implementation 4-65
Table 4-46 Serial interface device addresses 4-66
Table 4-47 SBCon 0 serial bus register 4-66
Table 4-49 Serial bus device addresses 4-67
Table 4-48 SBCon 1 serial bus register 4-67
Table 4-50 SCI implementation 4-68
Table 4-51 SSP implementation 4-69
Table 4-52 SMC implementation 4-70

Table 4-53 Timer implementation 4-71
Table 4-54 UART implementation 4-72
Table 4-55 USB implementation 4-74
Table 4-56 USB controller base address 4-75
Table 4-57 Watchdog implementation 4-76
Table 4-58 CompactFlash implementation 4-77
Table 4-59 CF_CTRL register bit assignments 4-78
Table 5-1 PLL frequency limits for different divisors 5-7
Table 5-2 Clock signals 5-10
Table 5-3 Reset handling 5-12
Table 5-4 Reset deassertion with respective clock 5-14
Table 5-5 Test chip interrupt control register 5-15
Table 5-6 Internal interrupt lines 5-16
Table 5-7 Test chip power status register 5-18
Table 5-8 Most Useful settings of the L2BYPASS and MPMASTNUM configuration bits 5-23
Table 5-9 Peripheral decoder mapping 5-24
Table 5-10 Test chip TAP instructions 5-27
Table 5-11 ARM11 MPCore TAP ID register 5-29
Table 5-12 Test chip TAP ID register 5-29
Table A-1 Compact Flash connector pinout A-3
Table A-2 Multimedia Card interface signals A-8
Table A-3 Mouse and keyboard port signal descriptions A-9
Table A-4 Serial plug signal assignment A-11
Table A-5 SSP signal assignment A-12
Table A-6 Smartcard connector signal assignment A-13
Table A-7 Signals on SCI expansion connector A-14
Table A-8 Ethernet signals A-15
Table A-9 DVI connector signals A-18
Table A-10 HDRX signals A-20
Table A-11 HDRY signals A-27
Table A-12 HDRZ signals A-34
Table B-1 Baseboard electrical characteristics B-2
Table B-2 AC Specifications B-3
Table C-1 Static memory connector signals C-6
Table E-1 STDIO redirection E-5
Table E-2 platform library options E-11
Table E-3 NFU commands E-16
Table E-4 NFU MANAGE commands E-17
Table F-1 Standard Boot Monitor command set F-3
Table F-2 MMC, SD, and CompactFlash card sub-menu commands F-4
Table F-3 Boot Monitor Configure commands F-5
Table F-4 Boot Monitor Debug commands F-6
Table F-5 Boot Monitor NOR flash commands F-7

List of Figures
RealView Platform Baseboard for ARM11 MPCore User Guide

	Key to timing diagram conventions	xxi
Figure 1-1	PB11MPCore system architecture	1-3
Figure 3-1	Baseboard layout	3-2
Figure 3-2	Front panel layout	3-4
Figure 3-3	Rear panel layout	3-5
Figure 3-4	PB11MPCore top level block diagram	3-6
Figure 3-5	Main tile and system bus routing	3-11
Figure 3-6	Top-level view of ARM11 MPCore test chip	3-13
Figure 3-7	Northbridge block diagram	3-15
Figure 3-8	Southbridge block diagram	3-20
Figure 3-9	PCI-PCI Express interface	3-34
Figure 3-10	Voltage control and voltage and current monitoring	3-36
Figure 3-11	Clock architecture	3-38
Figure 3-12	Northbridge clock domains	3-41
Figure 3-13	PCI and PCI Express clock routing	3-42
Figure 3-14	ARM11 MPCore test chip clocks	3-44
Figure 3-15	Reset routing	3-45
Figure 3-16	Reset Controller state diagram	3-46
Figure 3-17	Reset timing	3-47
Figure 3-18	External and internal interrupt sources	3-48
Figure 4-1	System memory map for standard peripherals	4-8

Figure 4-2 SYS_ID register 4-15
Figure 4-3 SYS_USERSW register 4-16
Figure 4-4 SYS_LED register 4-16
Figure 4-5 SYS_OSCx register 4-17
Figure 4-6 SYS_LOCK register 4-18
Figure 4-7 100Hz Counter, SYS_100HZ register 4-19
Figure 4-8 SYS_RESETCTL register 4-21
Figure 4-9 SYS_MCI register 4-22
Figure 4-10 SYS_FLASH register 4-23
Figure 4-11 SYS_CLCD register 4-24
Figure 4-12 SYS_CFGSW register 4-25
Figure 4-13 SYS_24MHZ register 4-26
Figure 4-14 SYS_MISC register 4-26
Figure 4-15 SYS_DMAPSR register 4-28
Figure 4-16 SYS_PEX_STAT register 4-29
Figure 4-17 SYS_PCI_STAT register 4-31
Figure 4-18 SYS_PLD_CTRL1 register 4-32
Figure 4-19 SYS_PLD_CTRL2 register 4-34
Figure 4-20 SYS_PLD_INIT register 4-35
Figure 4-21 SYS_PROCID0 register 4-38
Figure 4-22 SYS_PROCID1 register 4-39
Figure 4-23 SYS_OSCRESETx register 4-40
Figure 4-24 AACI ID register 4-47
Figure 4-25 CF_CTRL Register 4-78
Figure 5-1 Top-level view of ARM11 MPCore test chip 5-2
Figure 5-2 Clock domain overview 5-4
Figure 5-3 PLL block diagram 5-6
Figure 5-4 Test chip interrupt control register 5-15
Figure 5-5 Test chip power status register 5-18
Figure 5-6 L220 data RAM organization 5-21
Figure 5-7 L220 bypass mechanism 5-23
Figure 5-8 MPCore and L220 port configurations 5-24
Figure 5-9 TAP signal connections between the four ARM11 MPCore CPUs 5-28
Figure 5-10 TAP ID register 5-28
Figure A-1 Compact Flash connector pin numbering A-2
Figure A-2 Audio connectors A-6
Figure A-3 MMC/SD card socket pin numbering A-7
Figure A-4 MMC card A-8
Figure A-5 KMI connector A-9
Figure A-6 GPIO connector A-10
Figure A-7 Serial connector A-11
Figure A-8 SSP expansion interface A-12
Figure A-9 Smartcard contacts assignment A-13
Figure A-10 SCI expansion A-14
Figure A-11 Ethernet connector A-15
Figure A-12 USB connectors A-16
Figure A-13 DVI connector A-17

Figure A-14 HDRX, HDRY, and HDRZ pin numbering A-19
Figure A-15 JTAG connector A-41
Figure A-16 USB debug connector A-42
Figure A-17 Integrated Logic Analyzer (ILA) connector A-42
Figure B-1 Tile site multiplexed AXI timing B-4
Figure C-1 Static memory board block diagram C-2
Figure C-2 Samtec 120-way connector C-5
Figure D-1 Signals on the RealView Logic Tile expansion connectors D-2
Figure D-2 HDRX, HDRY, and HDRZ (upper) pin numbering D-4

Preface

This preface introduces the *RealView® Platform Baseboard for ARM11 MPCore User Guide*. It contains the following sections:

- *About this book* on page xviii
- *Feedback* on page xxvi.

About this book

This book describes how to set up and use the RealView Platform Baseboard for ARM11 MPCore (PB11MPCore).

Intended audience

This document has been written for experienced hardware and software developers to aid the development of ARM-based products using the PB11MPCore as part of a development system.

Using this book

This book is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an introduction to the PB11MPCore. This chapter shows the physical layout of the board and identifies the main components.

Chapter 2 ***Getting Started***

Read this chapter for a description of how to set up and start using the PB11MPCore. This chapter describes connecting the debug equipment and initializing and configuring the baseboard.

Chapter 3 ***Hardware Description***

Read this chapter for a description of the hardware architecture of the PB11MPCore. This chapter describes the peripherals, clocks, resets, and the debug hardware provided by the baseboard.

Chapter 4 ***Programmer's Reference***

Read this chapter for a description of the PB11MPCore memory map and registers. There is also basic information on the peripherals and controllers present on the baseboard.

Chapter 5 ***Processor Sub-System***

Read this chapter for a description of the hardware architecture of the ARM11 MPCore test chip. This chapter describes the peripherals, clocks, resets, and the debug hardware provided by the test chip.

Appendix A ***Signal Descriptions***

Refer to this appendix for a description of the signals on the connectors.

Appendix B ***Specifications***

Refer to this appendix for electrical and timing specifications.

Appendix C ***Memory Expansion Boards***

Refer to this appendix for details on using the available PISMO memory expansion boards.

Appendix D ***RealView Logic Tile expansion***

Refer to this appendix for details of how to use a RealView Logic Tile with the PB11MPCore.

Appendix E ***Boot Monitor and platform library***

Refer to this appendix for details of how to use the ARM Boot Monitor and platform library.

Appendix F ***Boot Monitor Commands***

Refer to this appendix for details of the user commands accepted by the Boot Monitor command interpreter.

Appendix G ***Loading FPGA Images***

Refer to this appendix for details of how to use the `progcards` utilities to load images from the supplied Versatile CD into the baseboard and Logic Tile FPGAs and PLDs.

Product revision status

The r*n*p*n*v*n* identifier indicates the revision status of products, such as *PrimeCells*, described in this document, where:

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

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

v*n* Identifies a version that does not affect the external functionality of the product.

Typographical conventions

The following typographical conventions are used in this book:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes ARM processor signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
`monospace italic`	Denotes arguments to commands and functions where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.

Other conventions

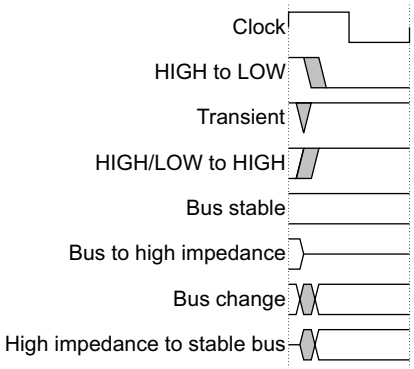
This document uses other conventions. They are described in the following sections:

- *Timing diagrams*
- *Signals* on page xxi
- *Bytes, Halfwords, and Words* on page xxii
- *Bits, bytes, k, and M* on page xxii
- *Register fields* on page xxii.
- *Numbering* on page xxiii.

Timing diagrams

The figure named *Key to timing diagram conventions* on page xxi 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

When a signal is described as being asserted, the level depends on whether the signal is active HIGH or active LOW. Asserted means HIGH for active high signals and LOW for active low signals:

Prefix n Active LOW signals are prefixed by a lowercase n except in the case of AXI, AHB or APB reset signals. These are named **ARESETn**, **HRESETn** and **PRESETn** respectively.

Prefix A Denotes global *Advanced eXtensible Interface* (AXI) signals:

Prefix AR Denotes AXI read address channel 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 AHB signals are prefixed by an upper case H.

Prefix P APB signals are prefixed by an upper case P.

Prefix R Denotes AXI read data channel signals.

Prefix W Denotes AXI write data channel signals.

Bytes, Halfwords, and Words

Byte Eight bits.

Halfword Two bytes (16 bits).

Word Four bytes (32 bits).

Quadword 16 contiguous bytes (128 bits).

Bits, bytes, k, and M

Suffix b Indicates bits.

Suffix B Indicates bytes.

Suffix k When used to indicate an amount of memory means 1024 and is uppercase. When used to indicate a frequency means 1000 and is lowercase.

Suffix M When used to indicate an amount of memory means 1024^2 = 1 048 576 and is uppercase. When used to indicate a frequency means 1 000 000 and is uppercase.

Register fields

All reserved or unused address locations must not be accessed as this can result in unpredictable behavior of the device.

All reserved or unused bits of registers must be written as zero, and ignored on read unless otherwise stated in the relevant text.

All registers bits are reset to logic 0 by a system reset unless otherwise stated in the relevant text.

Unless otherwise stated in the relevant text, all registers support read and write accesses. A write updates the contents of the register and a read returns the contents of the register.

All registers defined in this document can only be accessed using word reads and word writes, unless otherwise stated in the relevant text.

Numbering

The numbering convention is:

Hexadecimal numbers

Hexadecimal numbers are always prefixed with `0x`, and use uppercase alphabetical characters, for example `0xFFDD00CC`.

Binary numbers

Binary numbers are always prefixed with a lowercase b, for example b1001001.

Ranges

Ranges use a standard dash to indicate a range of numbers, for example 12-19.

Bit numbers

Single bit numbers are enclosed in brackets, for example [2].
A bit range is enclosed in brackets with a colon, for example [7:0].

Further reading

This section lists publications from both ARM and third parties that provide additional information on developing code for the ARM family of processors.

ARM periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets, addenda, and the Frequently Asked Questions list.

ARM publications

This manual contains information that is specific to the PB11MPCore. See the following documents for other relevant information:

The following publications provide reference information about the ARM architecture:

- *AMBA® Specification* (ARM IHI 0011)
- *AMBA 3 AXI Protocol* (ARM IHI 0022)
- *ARM Architecture Reference Manual* (ARM DDI 0100)

The following publications provided information about the ARM11 MPCore processor:

- *MPCore Multiprocessor Technical Reference Manual* (ARM DDI 0360)
- *ARM VFP11 Vector Floating-point Coprocessor Technical Reference Manual* (ARM DDI 0274)

The following publications provide information about ARM PrimeCell® controllers used in the ARM11 MPCore test chip:

- *L220 Cache Controller Technical Reference Manual* (ARM DDI 0329)

The following publications provide information about ARM PrimeCell controllers and bus interconnect used in the PB11MPCore Northbridge ASIC:

- *ARM PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual* (ARM DDI 0331)
- *ARM PrimeCell Static Memory Controller (PL354 series) Technical Reference Manual* (ARM DDI 0380)
- *ARM PrimeCell Single Master DMA Controller (PL081) Technical Reference Manual* (ARM DDI 0218)
- *ARM PrimeCell Color LCD Controller (PL111) Technical Reference Manual* (ARM DDI 0293)
- *ARM PrimeCell AXI Configurable Interconnect (PL300) Technical Reference Manual* (ARM DDI 0354)

The following publications are open access documents that provide information about ARM PrimeCell peripherals and controllers used in the Southbridge FPGA:

- *ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual* (ARM DDI 0173)
- *ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual* (ARM DDI 0172)
- *ARM PrimeCell System Controller (SP810) Technical Reference Manual* (ARM DDI 0254)
- *ARM PrimeCell PS2 Keyboard/Mouse Interface (PL050) Technical Reference Manual* (ARM DDI 0143)
- *ARM PrimeCell UART (PL011) Technical Reference Manual* (ARM DDI 0183)
- *ARM PrimeCell Synchronous Serial Port (PL022) Technical Reference Manual* (ARM DDI 0194)
- *ARM PrimeCell Smart Card Interface (PL131) Technical Reference Manual* (ARM DDI 0228)
- *ARM PrimeCell General Purpose Input/Output (PL061) Technical Reference Manual* (ARM DDI 0190)
- *ARM PrimeCell Real Time Clock (PL031) Technical Reference Manual* (ARM DDI 0224)
- *ARM Core MPCore Test Chip (MP003) Errata Notice, Document Revision 2.0 PR145-PRDC-006028v2.0.*

The following publications provide information about related ARM products and toolkits:

- *RealView Logic Tile LT-XC2V4000+ User Guide* (ARM DUI 0186)
- *RealView Logic Tile LT-XC4VLX100+ User Guide* (ARM DUI 0345)
- *RealView® ICE User Guide* (ARM DUI 0155)
- *RealView Debugger User Guide* (ARM DUI 0153)
- *RealView Compilation Tools Compilers and Libraries Guide* (ARM DUI 0205)
- *RealView Compilation Tools Developer Guide* (ARM DUI 0203)
- *RealView Compilation Tools Linker and Utilities Guide* (ARM DUI 0206).

Other Publications

The following publication describes the JTAG ports with which RealView ICE communicates:

- *IEEE Standard Test Access Port and Boundary Scan Achitecture* (IEEE Std. 1149.1).

Feedback

ARM® Limited welcomes feedback both on the PB11MPCore and on the documentation.

Feedback on this product

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

- the product name
- a concise explanation of your comments.

Feedback on this manual

If you have any comments about this document, send email to `errata@arm.com` giving:

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

ARM® Limited also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the *RealView Platform Baseboard for ARM11 MPCore* (PB11MPCore). It contains the following sections:

- *About the PB11MPCore* on page 1-2
- *Precautions* on page 1-6.

1.1 About the PB11MPCore

The *Platform Baseboard for ARM11 MPCore* (PB11MPCore) is a highly integrated software and hardware development system based on the ARM SMP architecture. It is supplied self-powered, in an ATX profile enclosure.

Used standalone the PB11MPCore serves as a fast multiprocessor software development platform with four ARM11 MPCore processors and a memory system running at near ASIC speed.

Used with FPGA-based *RealView Logic Tiles*, stacked on the baseboard, it enables custom *AMBA 3* peripherals, processors, and DSPs to be added to the existing ARM development system.

The PB11MPCore is intended for development of both embedded and operating system based software for ARM processors. The ability to add Logic Tiles enables new hardware to be prototyped and validated, and drivers to be debugged.

The PB11MPCore includes:

- an ARM11 MPCore test chip
- a structured ASIC (Northbridge)
- an FPGA (Southbridge)
- static and dynamic memory
- integrated peripherals
- tile site to enable connection of RealView Logic Tiles
- connectors for external peripherals and JTAG configuration and debug.

Figure 1-1 on page 1-3 shows the top level system architecture of the baseboard.

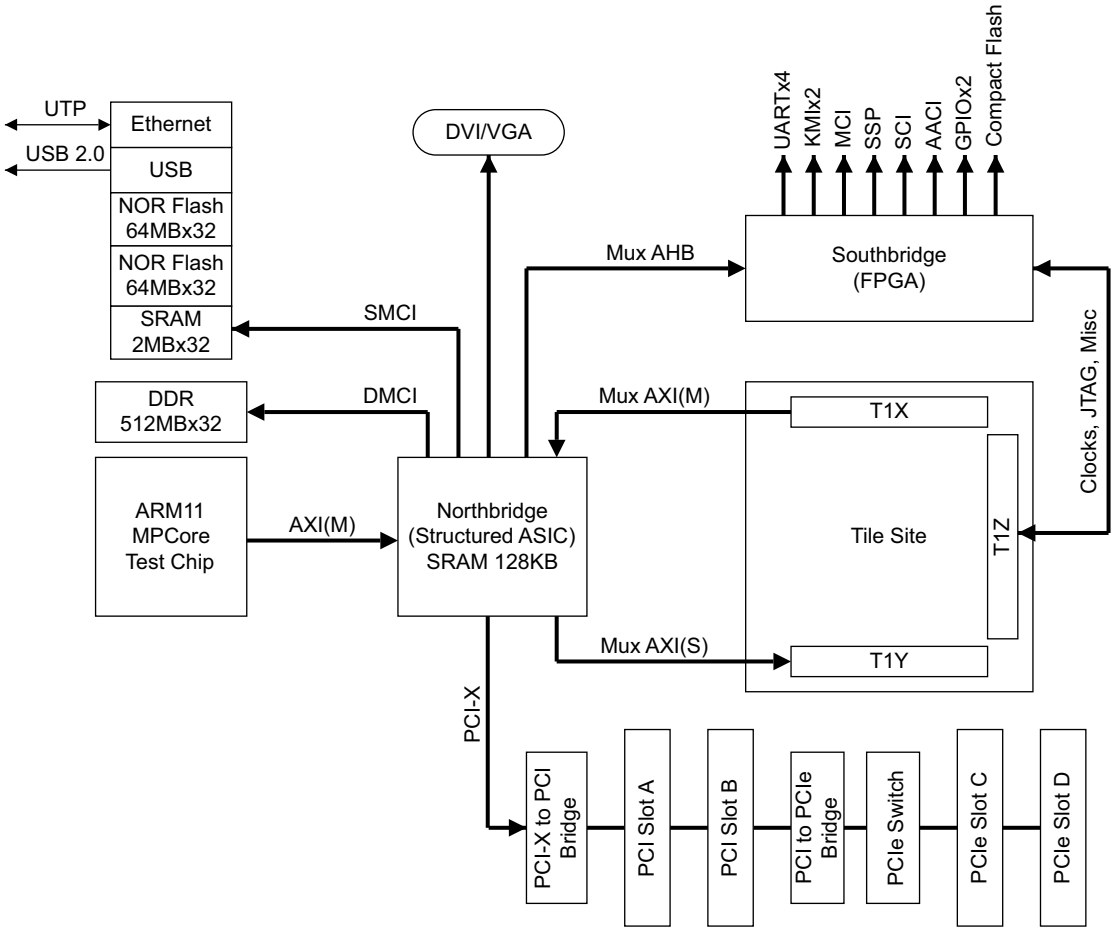


Figure 1-1 PB11MPCore system architecture

The following sections introduce the major system components:

- *ARM11 MPCore test chip* on page 1-4
- *Northbridge* on page 1-4
- *Southbridge* on page 1-4
- *PB11MPcore expansion* on page 1-5

1.1.1 ARM11 MPCore test chip

The ARM11 MPCore test chip implements the following system components and interfaces:

- four ARM11 MPCore processors, each with Vector Floating Point processor
- separate 32KB data and instruction caches
- 1MB unified Level 2 cache
- AXI interface (64-bit) to the Northbridge.

See Chapter 5 *Processor Sub-System* for details.

1.1.2 Northbridge

The Northbridge implements the following system components and interfaces:

- Static Memory Controller (PL354)
- Dynamic Memory Controller (PL340)
- Single Master DMA Controller (PL081)
- Color LCD Controller (PL111)
- AXI Configurable Interconnect (PL300)
- multiplexed AXI interfaces to and from the baseboard tile site
- multiplexed AHB-Lite interface to the baseboard Southbridge
- PCI/PCI-X interface.

See *Northbridge* on page 3-15 for details.

1.1.3 Southbridge

The Southbridge implements the following peripheral components and interfaces:

- Advanced Audio CODEC Interface (PL041)
- Multimedia Card Interface (PL180)
- PS2 Keyboard/Mouse Interface (PL050) x 2
- UART (PL011) x 4
- Watchdog Module (SP805) x 2
- Dual-Timer Module (SP804) x 4
- Synchronous Serial Port (PL022)
- Smart Card Interface (PL131)
- General Purpose Input/Output (PL061) x 3
  - sixteen GPIO ports available for external use
  - eight GPIO ports reserved for internal use.
- Real Time Clock (PL031)

- Generic Interrupt Controller x 4
- multiplexed AHB-Lite interface to the Northbridge
- AHB interface to the baseboard Compact Flash memory.

See *Southbridge* on page 3-7 and the appropriate *Technical Reference Manual* for details of the *PrimeCell* components that are integrated in the Southbridge FPGA.

Note

ARM do not support modifications made to the ARM Southbridge FPGA design. Custom peripheral development is only supported when using an attached Logic Tile such as the LT-XC4VLX1000+ for which specific *Application Notes* are made available on the CD supplied with the PB11MPCore and the ARM website.

1.1.4 PB11MPcore expansion

You can expand the PB11MPCore by adding:

- one or more stacked Logic Tiles containing your custom IP
- one or more PCI or PCI Express® cards
- one or more stacked PISMO static memory expansion boards
- Compact Flash card
- VGA monitor or color LCD display (DVI connection)
- MMC, SD, or SIM cards
- USB devices to the three USB ports (1 OTG and 2 standard host ports)
- serial devices to the synchronous serial port (SSP) and the four UARTs
- a keyboard and mouse
- audio devices to the onboard CODEC (AAC)
- a 10/100Mbps Ethernet network to the onboard Ethernet controller
- custom I/O to the 16-bit GPIO header.

1.2 Precautions

This section contains safety information and advice on how to avoid damage to the baseboard.

1.2.1 Ensuring safety

The baseboard is powered from an ATX power supply unit within the ATX enclosure.

Warning

To avoid a safety hazard, use the baseboard in its enclosure and only provide power via the ATX power connector (J39) using the integral ATX power supply unit.

1.2.2 Preventing damage

The baseboard is intended for use in a laboratory or engineering development environment. If removed from its enclosure, the board will become more sensitive to electrostatic discharges and will generate increased electromagnetic emissions.

Caution

To avoid damage, observe the following precautions:

- never subject the board to high electrostatic potentials
- always wear a grounding strap when touching the board in or away from its enclosure
- avoid touching the component pins or any other metallic element
- always power down the board when connecting Logic Tiles, memory expansion boards, or making external connections.

Caution

Do not use near equipment that is:

- sensitive to electromagnetic emissions (such as medical equipment)
- a transmitter of electromagnetic emissions.

Chapter 2
Getting Started

This chapter describes how to set up and prepare the PB11MPCore for use. It contains the following sections:

- *Setting up the baseboard* on page 2-2
- *Boot Monitor configuration* on page 2-4
- *JTAG debugger and USB config support* on page 2-6
- *Baseboard configuration switches* on page 2-7.

2.1 Setting up the baseboard

The following items are supplied:

- a custom ATX enclosure containing:
  - a PB11MPCore (micro ATX profile) printed-circuit board (HBI-0159)
  - front panel printed-circuit boards (HBI-0175, HBI-0176)
  - an ATX mains operated power supply (110V/230V AC)
- a CD containing sample programs, Boot Monitor code, FPGA and PLD images, and additional release documentation.

Note

For normal standalone operation, it is not necessary to change any of the default switch settings on the baseboard. See *Baseboard configuration switches* on page 2-7.

Before fitting any Logic Tiles, you should test that the system boots normally.

To set up the PB11MPCore as a standalone development system:

1. If you are using one or more expansion Logic Tiles, open the enclosure and stack the tiles on the tile expansion connectors on the baseboard.
   See Appendix D *RealView Logic Tile expansion* and the user guide for your Logic Tile for details.
2. If you are using one or more memory expansion boards, open the enclosure and stack the boards on the PISMO connector on the baseboard.
   See Appendix C *Memory Expansion Boards* for details.
3. If you are using an external display:
   - for analog VGA displays, connect a DVI-A cable from the display to the DVI connector on the back panel of the enclosure
   - for digital CLCD displays, connect a DVI-D cable from the display to the DVI connector on the back panel of the enclosure.

   Note

   A DVI-I cable can be used to connect a digital or analog display.
4. If you are using a debugger, connect to the JTAG connector (JTAG-ICE) on the back panel of the enclosure. See *Rear panel layout* on page 3-5 for the location of the JTAG-ICE connector, and *JTAG debugger and USB config support* on page 2-6 for information on debug support.

5. Apply power to the ATX enclosure, switch on the mains switch on the rear panel. See *Rear panel layout* on page 3-5 for the location of the mains switch.
   Press the power button on the front panel. The power indicator will light. See *Front panel layout* on page 3-4 for the location of the power button and indicator.

6. If you are using the supplied Boot Monitor software to select and run an application, see Appendix E *Boot Monitor and platform library*.

2.2 Boot Monitor configuration

The Boot Monitor application is typically loaded into the NOR flash memory and selected to run at power on. Follow the instructions in *Loading Boot Monitor into NOR flash* on page E-6 for details of loading the boot flash with the image from the supplied CD. How the Boot Monitor runs is determined by the setting of User Switches.

Note

It is not necessary to open the enclosure to configure the Boot Monitor. The User Switches 1 to 3 on the front panel of the ATX enclosure (see *Front panel layout* on page 3-4) control this. The User Switches in switch bank S4 on the PB11MPCore baseboard (see *Baseboard layout* on page 3-2) duplicate the front panel User Switches and must be left in the default all switches OFF position for normal operation.

User Switches 4 to 8 (S4-4 to S4-8) are not used by the Boot Monitor and are available for user applications.

If a different loader program is present at the boot location, the function of the entire User Switch bank becomes implementation dependent.

User Switch 1 (S4-1) determines the Boot Monitor behavior after a reset. The available options are shown in Table 2-1.

Table 2-1 Boot Monitor startup behavior

User Switch 1	Startup Behavior
OFF	A prompt is displayed enabling you to enter Boot Monitor commands.
ON	The Boot Monitor executes a boot script that has been loaded into NOR flash, a Multimedia (MMC) or Secure Digital (SD) card. If a boot script is not present, the Boot Monitor prompt is displayed.

The boot script can execute any Boot Monitor commands. It typically selects and runs an application image that has been stored in either NOR flash memory or on a MMC or SD card. You can store one or more code images in flash memory and use the boot script to start an image at reset. Use the `SET BOOTSCRIPT` command to set the boot script file name from the Boot Monitor (see *Standard Boot Monitor command set* on page F-3).

Output and input of text from STDIO for both applications and Boot Monitor I/O depends on the setting of User Switch 2 (S4-2) and User Switch 3 (S4-3) as listed in Table 2-2.

Table 2-2 STDIO redirection

User Switch 2	User Switch 3	Output	Input	Description
OFF	OFF	UART0 or console	UART0 or console	STDIO autodetects whether to use semihosting I/O or a UART. If a debugger is connected and semihosting is enabled, STDIO is redirected to the debugger console window. Otherwise, STDIO goes to UART0.
OFF	ON	UART0	UART0	STDIO is redirected to UART0. This occurs even under semihosting.
ON	OFF	DVI	Keyboard	STDIO is redirected to the DVI display and keyboard. This occurs even under semihosting.
ON	ON	DVI	UART0	STDIO output is redirected to the DVI display and input is redirected to UART0. This occurs even under semihosting.

User Switches 2 and 3 (S4-2 and S4-3) do not affect file I/O operations performed under semihosting. Semihosting operation requires a debugger and a JTAG interface device. See *Redirecting character output to hardware devices* on page E-7 for more details on I/O.

2.3 JTAG debugger and USB config support

In configuration mode, you can use either a JTAG debugger, such as *RealView ICE* or the custom USB config port to configure the baseboard configuration flash, FPGA, and PLDs.

Note

You cannot program normal flash memory from configuration mode.

2.3.1 JTAG debugger

You can use a JTAG debugger connected to the JTAG connector on the rear panel to:

- connect to the ARM11 MPCore test chip and download programs to memory and debug them
- program new FPGA, and PLD images on the baseboard
- program new FPGA, and PLD images on a Logic Tile fitted to the baseboard.

See *Rear panel layout* on page 3-5 for the location of the JTAG ICE connector, and Appendix G *Loading FPGA Images* for device programming details.

2.3.2 USB config port

The PB11MPCore contains custom logic that interfaces the USB config port to the onboard JTAG signals.You can use a PC connected to the custom USB config port connector on the front panel to:

- program new FPGA, and PLD images on the baseboard
- program new FPGA, and PLD images on a Logic Tile fitted to the baseboard.

See *Front panel layout* on page 3-4 for the location of the USB config connector, and Appendix G *Loading FPGA Images* for device programming details.

Caution

The baseboard is supplied with the FPGA and PLD images already programmed. Information is provided in Appendix G *Loading FPGA Images* however, in case of accidental erasure of the FPGA or PLDs.You are advised not to reprogram the FPGA or PLDs with any images other than those provided by ARM.

2.4 Baseboard configuration switches

Note

For normal operation, it is not necessary to open the ATX enclosure and change the baseboard configuration switch bank (S7) settings. The default setting is all switches OFF.

The location of the configuration switch bank (S7) on the baseboard is shown in Figure 3-1 on page 3-2.

The switch settings determine at power up:

- boot memory configuration
- Flash recovery mode
- Level 2 cache fix is enabled (and Level 2 cache is not bypassed)
- number of master ports available from the ARM11 MPCore.

2.4.1 Boot memory configuration

Use switch S7-1 to change the boot device. The available options are shown in Table 2-3.

Table 2-3 Selecting the boot device

S7-1	Device
OFF	NOR flash (default)
ON	PISMO expansion memory

2.4.2 Fast mode

Use switch S7-2 to increase the Northbridge and Southbridge frequencies from the default values of **ACLK** 70MHz and **HCLK** 35MHz to fast mode **ACLK** 100MHz and **HCLK** 40MHz. By using fast mode you can encounter a potential data corruption issue when using an OS. This is a consequence of MPCore test chip errata, see *ARM Core MPCore Test Chip (MP003) Errata Notice*. The setting is shown in Table 2-4.

Table 2-4 Selecting fast mode

S7-2	Device
OFF	Normal mode, default, Northbridge and Southbridge frequency
ON	Fast mode, faster **ACLK** and **HCLK**

2.4.3 Flash recovery

Use switch S7-5 to select whether all or only processor CPU#[0] in the ARM11 MPCore are released from reset. This function is provided to allow for recovery from corrupted NOR flash. The available options are shown in Table 2-5.

Table 2-5 ARM11 MPCore reset behavior

S7-5	Reset Setting
OFF	CPU#[3:0] are all released from reset (default)
ON	CPU#[0] only is released from reset

2.4.4 Level 2 Cache fix enable

Use switch S7-6 to select whether the Level 2 Cache fix is enabled. The available options are shown in Table 2-6.

Table 2-6 Enable Level 2 Cache fix

S7-6	Reset Setting
OFF	Level 2 Cache fix is enabled (default)
ON	Level 2 Cache fix is disabled

Note

The Level 2 Cache fix introduces an additional five clock cycles of latency to the system, but is required for reliable operation while the cache is in use.

2.4.5 MPMASTNUM selection

Use switch S7-8 to select the number of AXI master ports made available from the ARM11 MPCore. The available options are shown in Table 2-7.

Table 2-7 Number of ARM11 MPCore AXI ports

S7-8	Reset Setting
OFF	Master 0, and Master 1 of the ARM11 MPCore are available (default)
ON	Master 0 of the ARM11 MPCore is available

Note

For details of all the available ARM11MPCore test chip internal and external AXI port options see *L220 bypass module* on page 5-22.

Chapter 3
Hardware Description

This chapter describes the on-board hardware. It contains the following sections:

- *Baseboard architecture* on page 3-2
- *Tile interconnections* on page 3-10
- *ARM11 MPCore test chip* on page 3-12
- *Northbridge* on page 3-15
- *Southbridge* on page 3-20
- *Ethernet interface* on page 3-30
- *USB Interface* on page 3-31.
- *USB config interface* on page 3-32
- *DVI Interface* on page 3-33
- *PCI interface* on page 3-34
- *Power supply control* on page 3-36
- *Clock architecture* on page 3-38
- *Resets* on page 3-45
- *Interrupts* on page 3-48
- *Test, configuration, and debug interfaces* on page 3-54.

3.1 Baseboard architecture

Figure 3-1 shows the layout of the baseboard.

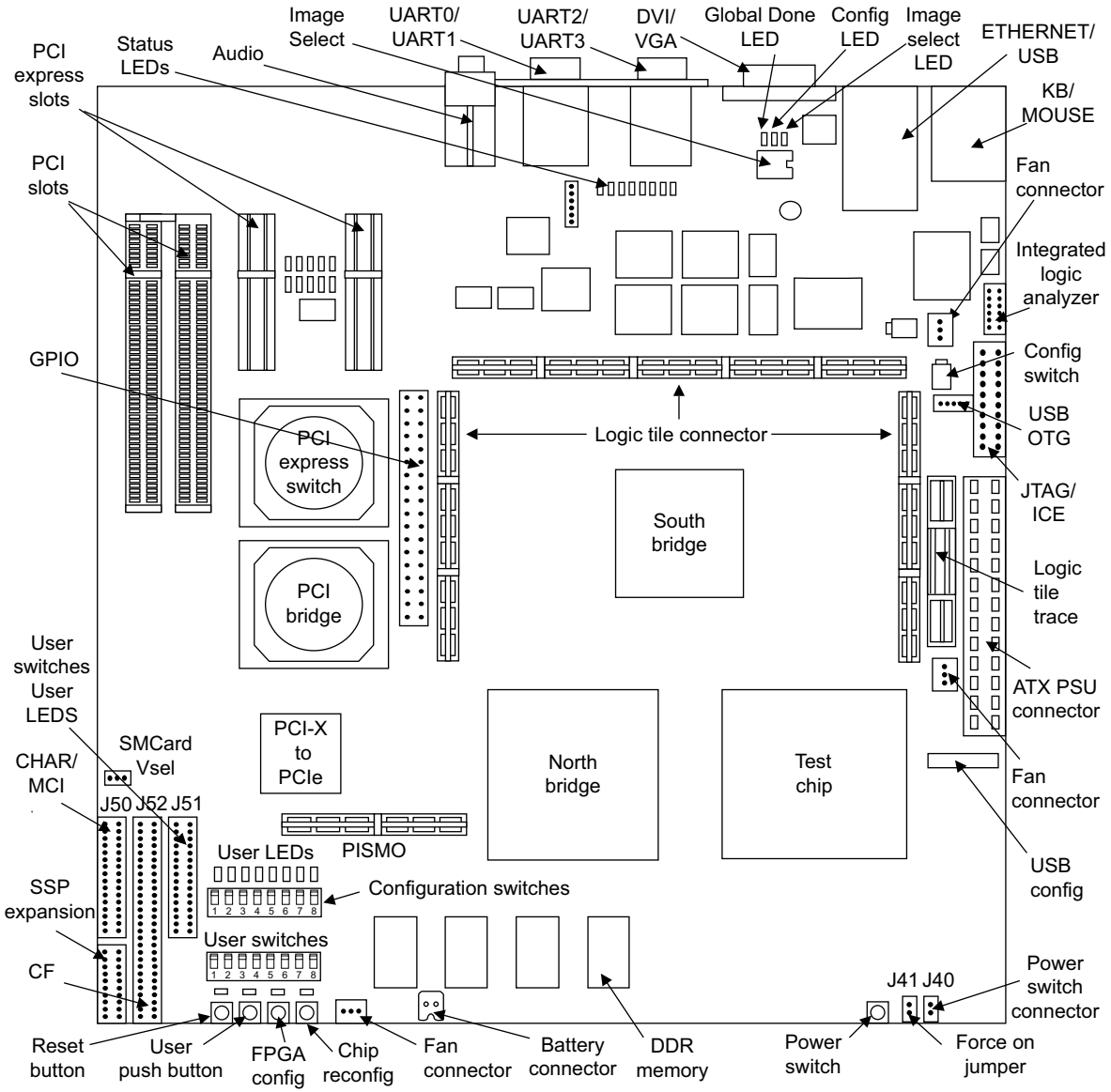


Figure 3-1 Baseboard layout

The major system components and interfaces provided by the baseboard are:

- a tile site to support ARM Logic Tiles
- a ARM11 MPCore test chip
- a Northbridge implementing the major system controllers and interfaces
- a Southbridge implementing most of the system peripherals
- a PLD controlling system configuration
- an 8MB configuration flash to hold the FPGA images
- 512MB of 32-bit wide (DDR) SDRAM
- 8MB of 32-bit wide (Pseudo) SRAM
- 128MB of 32-bit wide NOR flash in two banks of 64MB
- PISMO connector for up to 256MB (4x64MB) of static memory expansion
- PCI and PCI Express expansion buses (2 slots for each interface type)
- USB interface providing 1 OTG and 2 standard USB 2.0 host ports
- Ethernet interface providing 10Base-T and 100Base-TX support
- DVI-I interface providing color LCD display and analog VGA monitor support
- Synchronous Serial Port (SSP) interface
- 4 x RS232 interfaces with full handshake
- PS2 keyboard and mouse interfaces
- audio CODEC interface (AAC)
- Compact Flash, MMC, SD and Smart Card interfaces
- general purpose (User) switches and LEDs
- real-time clock (RTC)
- time of year clock (TOY) with backup battery
- programmable clock generators
- power supplies, voltage control and current monitoring circuitry
- JTAG config and debug, and Integrated Logic Analyzer (ILA) support
- USB config port.

3.1.1 Enclosure front panel

Figure 3-2 shows the front panel of the ATX enclosure that provides:

- power on-off button and indicator
- USB Config port connector (USB CFG)
- Config switch and indicator (FPGA and PLD configuration)
- Soft Reset switch
- Hard Reset switch
- Compact Flash interface connector
- USB OTG interface connector
- MMC and SD memory card interface connector
- User switches 1 to 8
- User LEDs 1 to 8
- SIM Card interface
- 5¼ bay (for customer hardware expansion).

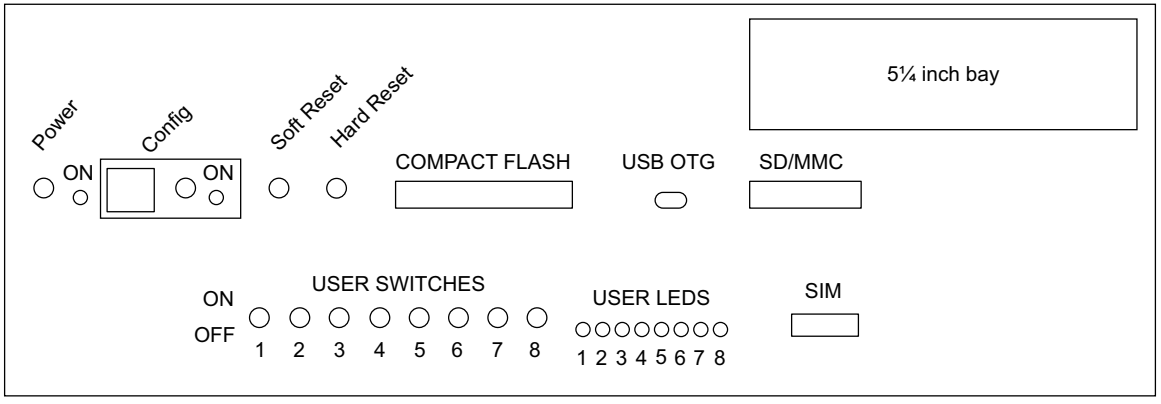


Figure 3-2 Front panel layout

Bypassing the power switch

The power button on the front panel is typically used to power on the system.

If a link is placed across J41 (FORCE ON), the power is forced on and the Power push button has no effect.

A jumper must not be placed across connector J40 (POWER). You can, however, use this connector to attach an external push button if the Power push button is not accessible or the board is mounted in the enclosure.

3.1.2 Enclosure rear panel

Figure 3-3 shows the ATX enclosure rear panel that provides:

- power connector and on-off switch
- keyboard and mouse interface (PS/2)
- Ethernet interface (10Base-T and 100Base-TX)
- 2x USB 2.0 ports
- video interface (DVI – color LCD and analog VGA supported)
- JTAG debugger interface
- 4x RS232 serial ports (includes handshake signals for modem support)
- audio interface (analog mic-in, line-in, and line-out supported)
- 2x PCI and 2x PCI Express slots.

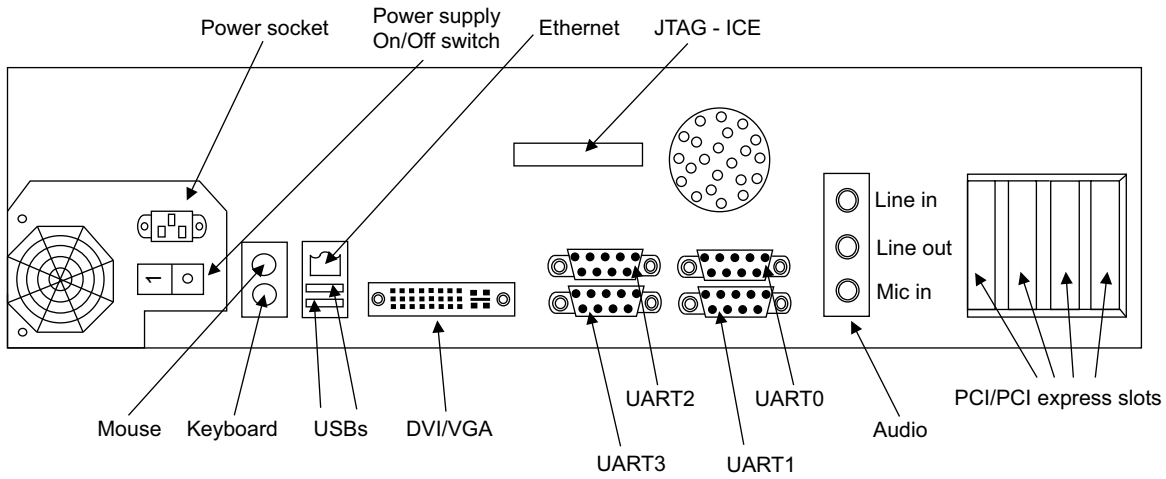


Figure 3-3 Rear panel layout

3.1.3 System architecture

Figure 3-4 on page 3-6 shows the top level system architecture of the baseboard.

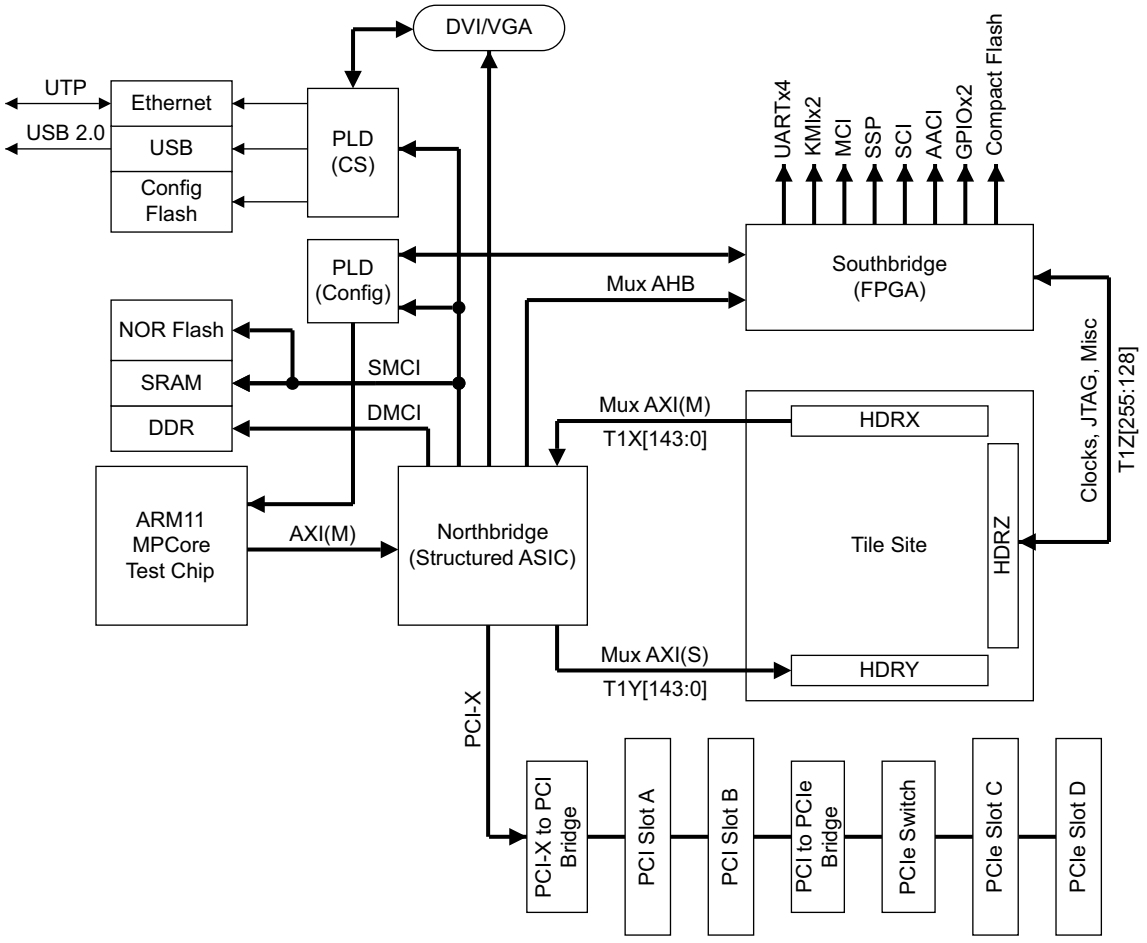


Figure 3-4 PB11MPCore top level block diagram

3.1.4 Northbridge

The Northbridge is a structured ASIC and provides the major system controllers and interfaces, and interfaces to the onboard ARM11 MPCore test chip. This enables the PB11MPCore to operate as a standalone development system for ARM11 MPCore based applications. See *Northbridge* on page 3-15 for details.

The Northbridge also interfaces to the tile site. This enables the PB11MPCore to operate as a standalone AMBA 3 AXI peripheral development system using an attached Logic Tile.

3.1.5 Southbridge

The Southbridge is an FPGA (Xilinx XC4VLX40-10) and provides the majority of the system peripherals. It also implements the system control and configuration functions required by the PB11MPCore. See *Southbridge* on page 3-20 for details.

Note

The majority of the controllers and interfaces implemented in the Northbridge and Southbridge are ARM *PrimeCell*® components.

Caution

The image loaded into the Southbridge FPGA must match the system configuration or both the baseboard and an attached Logic Tile may be damaged. A copy of the configuration FPGA image is supplied on the Versatile Family CD in case of accidental corruption of the preloaded image. See Appendix G *Loading FPGA Images* for details.

3.1.6 PCI bus connectors

Two PCI and two PCI Express bus connectors are provided to allow PCI and PCI Express profile cards to be connected directly to the baseboard. Rear panel access to the card back plates is provide. See *Rear panel layout* on page 3-5 for details.

3.1.7 Displays

The color LCD signals from the CLCD controller in the Northbridge are processed by a DVI transmitter chip and fed to the baseboard DVI-I connector. The color LCD signals are also converted to analog VGA signals by a video DAC and fed to the baseboard DVI-I connector. A serial 2-wire interface implemented in the Southbridge provides the DDC2B interface. See *DVI Interface* on page 3-33 for details.

Note

A DVI-D cable is required to connect a digital display, and a DVI-A cable is required to connect an analog display. Alternatively, a DVI-I cable can be used to connect a digital or analog display.

3.1.8 Logic Tile expansion

Logic Tiles, such as the LT-XC4VLX100+, when connected to the tile site, enable the development of additional AMBA 3 AXI peripherals, or custom logic, for use with ARM cores. See Appendix D *RealView Logic Tile expansion* and *RealView Logic Tile header connectors* on page A-19 for details.

Caution

Due to pin limitations, the PB11MPCore implements multiplexed AMBA 3 AXI interfaces at the HDRX and HDRY tile site headers. The Logic Tile used must implement a similar multiplexing scheme to be compatible with the PB11MPCore external AMBA 3 AXI interfaces.

ARM Application Note AN151 *Example AXI design for a Logic Tile on top of AXI Versatile baseboards* describes the multiplexed AMBA 3 AXI interface requirements and includes example code.

3.1.9 Clock generation

The baseboard contains the following clock sources:

- Six crystal oscillators. These provide the reference frequencies for the Real Time Clock, AACI, USB, Ethernet, and programmable clock sources.
- Seven programmable ICS307M-02 clock sources. These provide both local and global system clocks.
- Four PCI clocks.
- Four PCI Express spread spectrum differential clocks for reduced EMI.

See *Clock architecture* on page 3-38 for details.

3.1.10 Debug and test interfaces

The JTAG connector on the rear panel of the ATX enclosure enables JTAG hardware debugging equipment, such as RealView ICE, to be connected to the PB11MPCore. The JTAG equipment enables you to debug ARM-based applications.

The `progcards_usb` utility can control the JTAG signals via the dedicated USB config port provided on the front panel. This enables downloading of images to the FPGA and PLDs on the PB11MPCore, and a Logic Tile if fitted using a PC.
See *JTAG debugger and USB config support* on page 2-6.

An *Integrated Logic Analyzer* (ILA) connector on the baseboard enables debugging of the PB11MPCore and Logic Tile FPGA designs at the same time as the JTAG connector is being used to debug application code. An example of an ILA debugging device is the Xilinx ChipScope.

A trace connector is present on the baseboard for accessing processor cores implemented on Logic Tiles.

Note

The ARM11 MPCore test chip does not support trace.

3.2 Tile interconnections

The tile site on the baseboard enables the board to be used with Logic Tiles. The tiles are stackable and each tile has three connectors on the top and bottom (HDRX, HDRY, and HDRZ). See *RealView Logic Tile header connectors* on page A-19 for details.

Note

ARM support the use of Logic Tiles at the PB11MPCore tile site. Support is available through *Applications Notes* provided on the *Versatile Family CD* and on the ARM website at: `www.arm.com/documentation/Application_Notes/`

The bus usage on the tiles depend on the type of baseboard and the combination of tiles used. For PB11MPCore, HDRX (Bus T1X) carries the mux AXI master bus signals and HDRY carries the mux AXI slave bus signals (Bus T1Y). HDRZ (Bus T1Z) carries the additional I/O signals required by the Logic Tile (clocks, resets, JTAG etc). See Figure 3-5 on page 3-11 for a simplified diagram of the main tile and system bus routing.

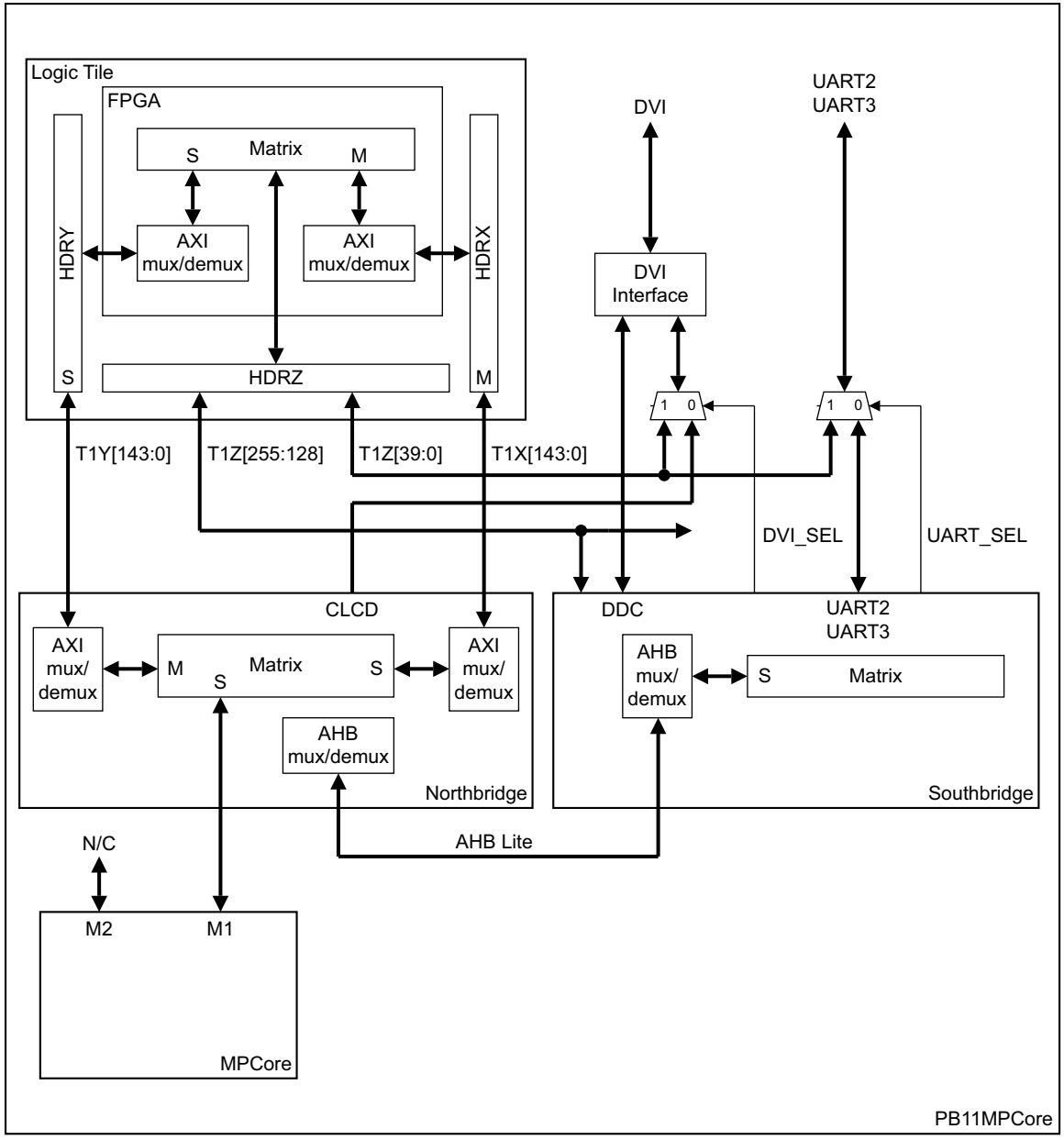


Figure 3-5 Main tile and system bus routing

Note

On the PB11MPCore, CLCD, UART2, and UART3 interfaces can be sourced from the baseboard or logic implemented in the FPGA on an attached Logic Tile. The CLCD source and the UART 2 and UART3 source are controlled independently by signals **DVI_SEL** and **UART_SEL** respectively.

For more details of the interconnect see *RealView Logic Tile header connectors* on page A-19.

3.2.1 AXI bus multiplexing

A bus multiplexing scheme is necessary to reduce the number of pins required on the HDRX and HDRY headers. Refer to *Application Note AN151* for details and example code.

3.3 ARM11 MPCore test chip

The ARM11 MPCore test chip is a proof-of-concept vehicle for the ARM11 MPCore macrocell and is built for two reasons:

- It enables you to functionally validate the ARM11 MPCore macrocell on the silicon process it is built on.
- You can use it as a building block to create prototype systems designed for product and operating system development.

The ARM11 MPCore test chip functionality consists of:

- four ARM11 MPCore CPUs, each CPU may be suspended (clock is stopped)
- L220 level 2 cache controller
- PLL
- boundary scan logic
- input and output pads.

Note

For user specific details on the test chip see Chapter 5 *Processor Sub-System* on page 5-1 and for details on the multiprocessor see *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360).

3.3.1 ARM11 MPCore test chip overview

Figure 3-6 shows the top-level functionality of the test chip.

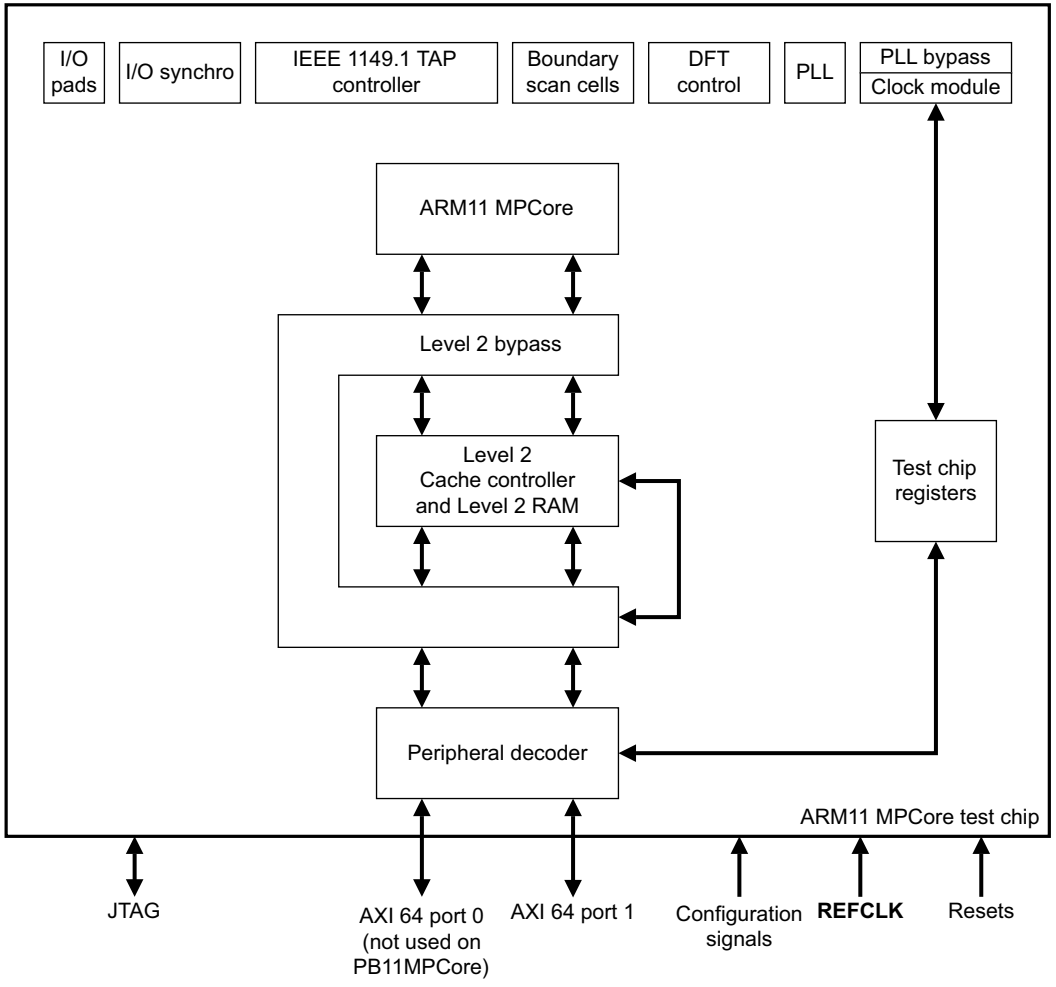


Figure 3-6 Top-level view of ARM11 MPCore test chip

The main features of the test chip are:

- a ARM11 MPCore incorporating:
  - four ARM11 MPCore CPUs with Vector Floating Point (VFP) that implement the ARM architecture v6
  - Level 1 (L1) memory subsystem providing 32KB instruction cache and 32KB data cache per CPU
  - JTAG-based debug.
- L220 Cache Controller incorporating 1MB of Level 2 (L2) unified cache
- peripheral decoder providing dual 64-bit external AXI buses
- on-chip PLL with skew control
- test chip register bank for configuring the ARM11 MPCore test chip
- design for test (DFT):
  - test control via a TEST_MODE test data register
  - L1 and L2 memory systems Memory Built-In Self Test (MBIST)
  - full chip and boundary scan support

See the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360), and the *L220 Cache Controller Technical Reference Manual* (ARM DDI 0329) for further details on these features.

3.4 Northbridge

Figure 3-7 shows in simplified form the architecture of the Northbridge on the PB11MPCore. The Northbridge is implemented as a structured ASIC.

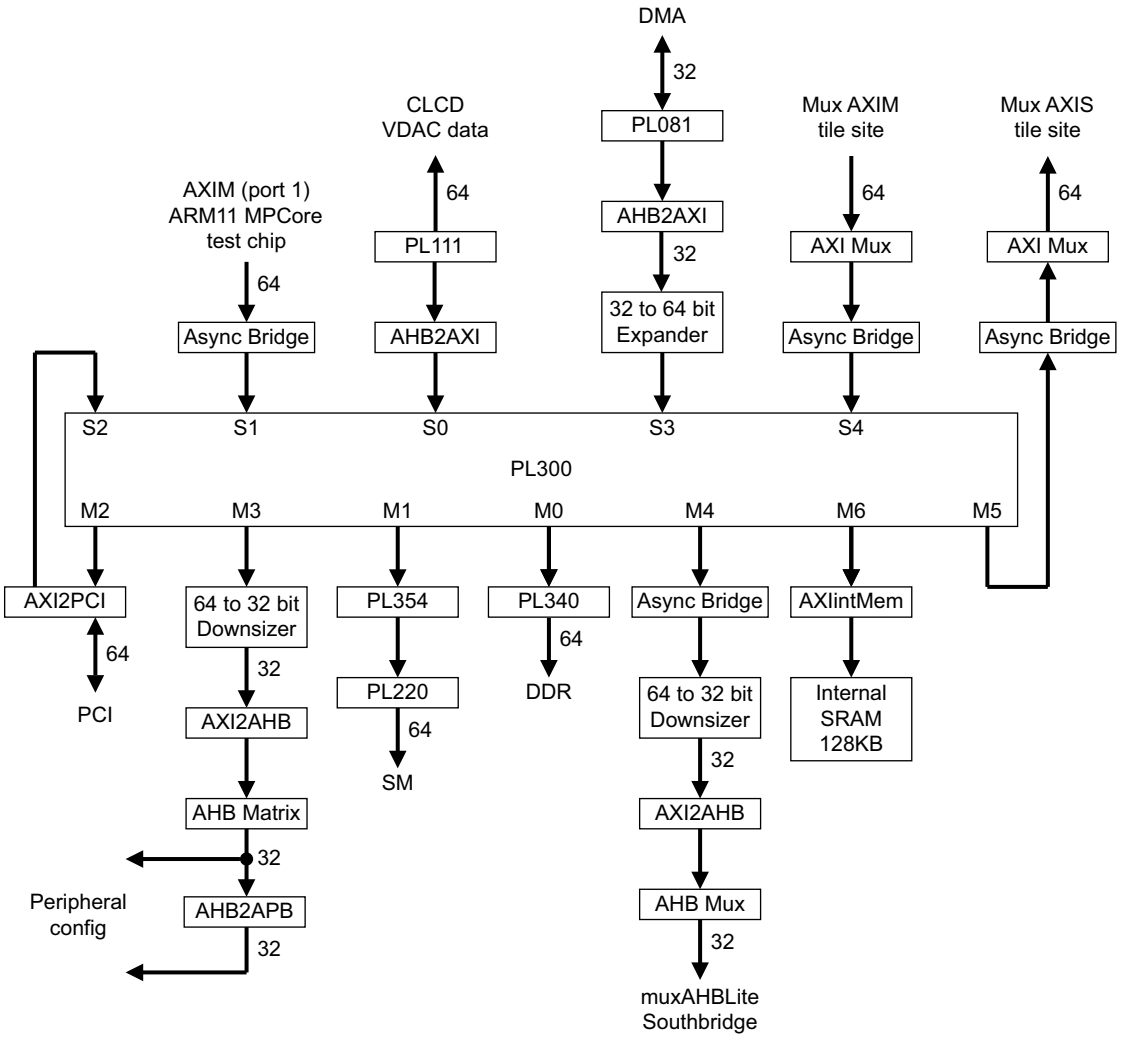


Figure 3-7 Northbridge block diagram

For details on the Northbridge see:

- *ARM11 MPCore test chip interface*
- *CLCD controller*
- *Memory controllers* on page 3-17
- *Multiplexed AHB-Lite interface* on page 3-18
- *Multiplexed AXI interfaces* on page 3-19
- *PCI interface* on page 3-19.

3.4.1 ARM11 MPCore test chip interface

The Northbridge interfaces to the ARM11 MPCore test chip via an asynchronous non-multiplexed 64-bit AXI bus to give the highest possible data transfer rate.

See *AMBA 3 AXI Protocol* (ARM IHI 0022) for details of the AXI interface and Chapter 5 *Processor Sub-System* for details of the ARM11 MPCore test chip interfaces.

3.4.2 CLCD controller

The Northbridge implements the PrimeCell PL111 *Color LCD Controller*, an Advanced Microcontroller Bus Architecture (AMBA) compliant System-on-Chip (SoC) peripheral that is developed, tested, and licensed by ARM.

The PrimeCell PL111 provides all of the necessary control signals to interface directly to a variety of color and monochrome LCD panels. It supports:

- single- and dual-panel mono *Super Twisted Nematic* (STN) displays with 4 or 8-bit interfaces
- single- and dual-panel color STN displays
- *Thin Film Transistor* (TFT) color displays.

Display resolutions are programmable up to 1024x768, and hardware cursor support for single-panel displays is provided.

See *Color LCD Controller, CLCDC* on page 4-48 for details of PB11MPCore usage and the *ARM PrimeCell Color LCD Controller (PL111) Technical Reference Manual* (ARM DDI 0293) for full programming details.

3.4.3 Memory controllers

The Northbridge implements memory controllers for:

- static memory
- dynamic memory
- direct memory access.

Static memory controller, SMC

The Northbridge implements the PrimeCell PL354 *Static Memory Controller*, an Advanced Microcontroller Bus Architecture (AMBA) compliant System-on-Chip (SoC) peripheral that is developed, tested, and licensed by ARM.

The PrimeCell PL354 is part of the PL350 series of area-optimized SRAM and NAND memory controllers with on-chip bus interfaces that conform to the AMBA Advanced eXtensible Interface (AXI) protocol.

In the Northbridge the PrimeCell PL354 supports:

- Pseudo Static Random Access Memory (PSRAM)
- NOR flash devices with an SRAM interface
- Ethernet and USB controllers with an SRAM interface.

See *Static Memory Controller, SMC* on page 4-70 for details of PB11MPCore usage and the *ARM PrimeCell Static Memory Controller (PL350 series) Technical Reference Manual* (ARM DDI 0380) for full programming details.

Dynamic memory controller, DMC

The Northbridge implements the PrimeCell PL340 *Dynamic Memory Controller*, an Advanced Microcontroller Bus Architecture (AMBA) compliant System-on-Chip (SoC) peripheral that is developed, tested, and licensed by ARM.

The PL340 DMC is a high-performance, area-optimized SDRAM memory controller with on-chip bus interfaces that conform to the AMBA Advanced eXtensible Interface (AXI) protocol.

In the Northbridge the PrimeCell PL340 supports:

- Synchronous Dynamic Random Access Memory (DDR SDRAM)
- a shared external memory bus interface.

Note

To reduce the pinout requirements, the Northbridge uses the PL340 DMC in conjunction with the PrimeCell PL220 *External Bus Interface* (EBI) to implement a shared external memory bus interface. See *ARM PrimeCell External Bus Interface (PL220) Technical Reference Manual* (DDI 0249) for details.

See *Dynamic Memory Controller, DMC* on page 4-51 for details of PB11MPCore usage and the *ARM PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual* (ARM DDI 0331) for full programming details.

Single master direct memory access controller, SMDMAC

The Northbridge implements the PrimeCell PL081 *Single Master DMA Controller*, an Advanced Microcontroller Bus Architecture (AMBA) compliant System-on-Chip (SoC) peripheral that is developed, tested, and licensed by ARM.

The PrimeCell PL081 is an AMBA AHB module, and connects to the Advanced High-performance Bus (AHB). Two DMA channels, each supporting a unidirectional transfer are provided. There are 16 peripheral DMA request lines and each peripheral connected to the PL081 can assert either a single DMA request, or a burst DMA request, the DMA burst size is programmable.

In the Northbridge the PrimeCell PL081 supports:

- eight channels of direct memory access (with DMAC flow control only)
- DMA access to the tile site
- two selectable DMA channel to peripheral mappings

See *Single Master Direct Memory Access Controller, SMDMAC* on page 4-50 for details of PB11MPCore usage and the *ARM PrimeCell Single Master DMA Controller (PL081) Technical Reference Manual* (ARM DDI 0218) for full programming details.

3.4.4 Multiplexed AHB-Lite interface

The Northbridge interfaces to the Compact Flash interface and low speed peripherals in the Southbridge via a custom multiplexed AHB-Lite master port.

See *AMBA® Specification* (ARM IHI 0011) for details of the AHB-Lite interface and *Southbridge* on page 3-20 for details of the Southbridge peripherals.

3.4.5 Multiplexed AXI interfaces

The Northbridge interfaces to the tile site via two multiplexed 64-bit AXI buses. Bus multiplexing is required at the tile site to reduce the pin count. Header HDRX carries a master multiplexed AXI bus from the tile site to the baseboard, and header HDRY carries a slave multiplexed AXI bus from the tile site to the baseboard.

See *Application Note AN151*, available on the Versatile Family CD or the ARM website, for details of the multiplexing scheme and *RealView Logic Tile header connectors* on page A-19 for details of the baseboard signal pinout.

3.4.6 PCI interface

The Northbridge implements an AXI to PCI bridge (AXI2PCI) that provides interface functions conforming to the PCI-X Protocol Addendum to the PCI Local Bus Specification Revision 2.0a.

See *AXI to PCI bridge* on page 4-62 for details of PB11MPCore usage and user programming details.

3.5 Southbridge

Figure 3-8 shows the architecture of the Southbridge on the PB11MPCore. The Southbridge is implemented by ARM as a custom FPGA.

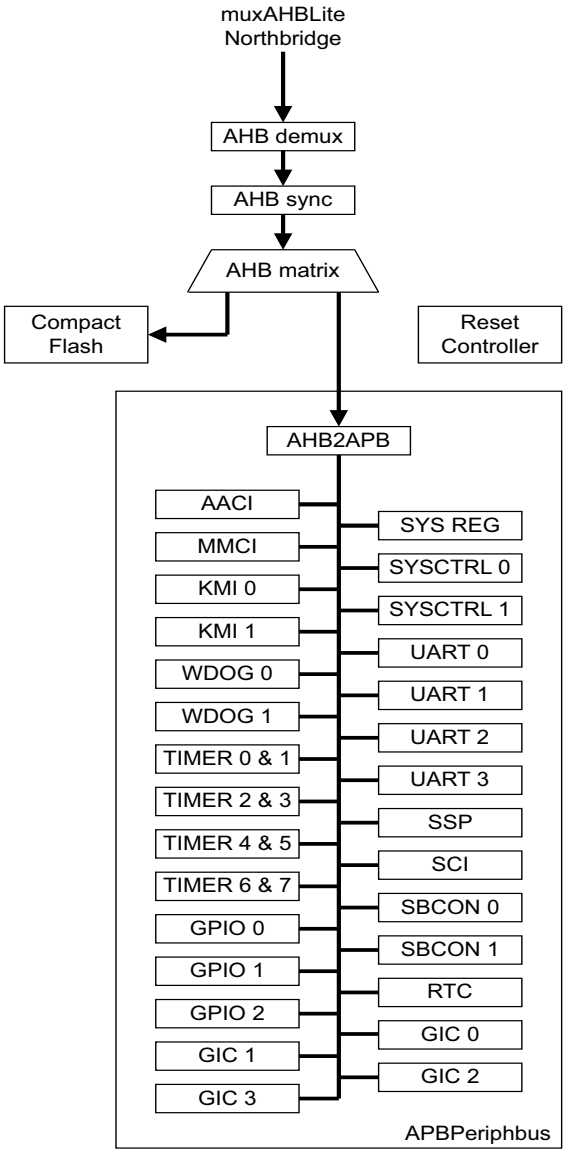


Figure 3-8 Southbridge block diagram

For details on the Southbridge see:

- *FPGA configuration*
- *Reset controller* on page 3-23
- *CompactFlash* on page 3-23
- *APB peripherals* on page 3-23.

3.5.1 FPGA configuration

At power-up the FPGA loads its configuration data from configuration flash memory. Parallel data from the flash memory is streamed by the configuration PLD into the configuration ports of the FPGA.

The image loaded into the FPGA is determined by the image select switch S1 on the baseboard (see *Baseboard layout* on page 3-2 for the location of S1).

The image selection options are listed in Table 3-1.

Table 3-1 FPGA image selection

S1-1	FPGA image	Image Address
OFF	FPGA image 1 (this is the image supplied with the board)	`0x0`
ON	FPGA image 3 (this image is not supplied with the board)	`0x400000`

Note

The configuration flash can hold two FPGA images however, only one FPGA image is provided.

The configuration flash is a separate device and not part of the user NOR flash.

You can use the appropriate `Progcards` utility connected via a JTAG debugger or the USB Config port to reprogram the PLDs, FPGA, and flash if the PB11MPCore is placed in configuration mode. See *Debug and Config port support* on page 3-54.

Caution

You are advised not to program these devices with any images other than those provided by ARM.

Program the configuration PLD as follows:

1. Connect an interface cable to either the JTAG or USB config port.
2. Set the Config switch on the front panel to ON (up position).
3. Power-up the board, the Config LED will light.
4. Run the appropriate `Progcards` utility from:
   `install_directory``\Versatile\PB11MPCore\`*`build`*`\Release\boardfiles\`
5. Choose the required configuration image, see *FPGA image selection* on page 3-21.

Caution

The 1.5V cell battery provides the **VBATT** backup voltage to the external DS1338 time-of-year clock and FPGA encryption key circuitry within the FPGA. Removing the battery erases the encryption key.

Each board is provided with an encryption key that is unique to the board. The standard image supplied with the board is not encrypted. However, encrypted images might be supplied by ARM in the future. If you are using encrypted images and the key is erased, you must return the board to ARM to have the key reloaded.

The battery is expected to last for approximately 10 years from manufacture of the PB11MPCore.

To replace the battery:

1. Power on the PB11MPCore. If the battery is removed while the board is powered down, the encryption key will be erased.
2. Remove the old battery.
3. Insert the new battery. Ensure that the positive terminal is facing upwards in the holder.

3.5.2 Reset controller

Correct initialization of the PB11MPCore and an associated Logic Tile requires a timed reset sequences custom Reset controller in the Southbridge monitors the reset sources and sequences the baseboard and Logic Tile resets during power-up (power-on reset sequence) and during baseboard configuration (system reset sequence).
See *Resets* on page 3-45 for details.

3.5.3 CompactFlash

The CompactFlash interface is a custom *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced High-performance Bus* (AHB).

The host (CF/CF+ card slot) 50 pin connector is positioned on the front panel of the enclosure, see *Front panel layout* on page 3-4 for details.

See *CompactFlash interface* on page 4-77 for details of PB11MPCore usage.
The physical interface provided by the PB11MPCore is shown in *Compact Flash interface* on page A-2.

3.5.4 APB peripherals

The majority of the controllers and interfaces implemented in the Southbridge are standard ARM *PrimeCell*® components:

- *Advanced Audio CODEC Interface, AACI* on page 3-24
- *Multimedia Card Interface, MCI* on page 3-24
- *Keyboard and Mouse Interface, KMI* on page 3-24
- *Watchdog Module* on page 3-25
- *Dual-Timer Module* on page 3-25
- *General Purpose Input/Output, GPIO* on page 3-25
- *Generic Interrupt Controller* on page 3-26
- *Status and System Control Register Block* on page 3-26
- *System Controller* on page 3-26
- *UART* on page 3-26
- *Synchronous Serial Port, SSP* on page 3-27
- *Smart Card Interface, SCI* on page 3-27
- *Two-wire serial bus interface* on page 3-28
- *Real Time Clock, RTC* on page 3-29.

Advanced Audio CODEC Interface, AACI

The ARM PrimeCell *Advanced Audio CODEC Interface (AACI)* PL041 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The PrimeCell AACI provides communication to an off-chip CODEC (LM4549) that supports the *AC-link* protocol.

See *Advanced Audio CODEC Interface, AACI* on page 4-46 for details of PB11MPCore usage and the *ARM PrimeCell Advanced Audio CODEC Interface (PL041) Tecnical Reference Manual* (ARM DDI 0173) for full programming details.
The physical interface provided by the PB11MPCore is shown in *Audio CODEC interface* on page A-6.

Multimedia Card Interface, MCI

The ARM PrimeCell *Multimedia Card Interface (MCI)* PL180 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The PrimeCell MCI provides all functions specific to the multimedia and secure digital memory card such as the clock generation unit, power management control, and command a data transfer. The interface conforms to *Multimedia Card Specification v2.11* and *Secure Digital Memory Card Physical Layer Specification v0.96*.

See *MultiMedia Card Interface, MCI* on page 4-61 for details of PB11MPCore usage and the *ARM PrimeCell Multimedia Card Interface (PL180) Tecnical Reference Manual* (ARM DDI 0172) for full programming details.
The physical interface provided by the PB11MPCore is shown in *MMC and SD card interface* on page A-7.

Keyboard and Mouse Interface, KMI

The ARM PrimeCell *PS2 Keyboard/Mouse Interface (KMI)* PL050 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The PrimeCell KMI can be used to implement a keyboard or mouse interface that is IBM PS2 or AT compatible.

See *Keyboard and Mouse Interface, KMI* on page 4-60 for details of PB11MPCore usage and the *ARM PrimeCell PS2 Keyboard/Mouse Interface (PL050) Tecnical Reference Manual* (ARM DDI 0143) for full programming details.
The physical interface provided by the PB11MPCore is shown in *Keyboard and mouse interface* on page A-9.

Watchdog Module

The *ARM Watchdog Module* SP805 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The Watchdog module consists of a 32-bit down counter with a programmable time-out interval that has the capability to generate an interrupt and a reset signal on timing out. It is intended to be used to apply a reset to a system in the event of a software failure.

See *Watchdog* on page 4-76 for details of PB11MPCore usage and the *ARM Watchdog Module (SP805) Technical Reference Manual* (ARM DDI 0270) for full programming details.

Dual-Timer Module

The *ARM Dual-Timer Module* SP804 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The Dual-Timer module consists of two programmable 32/16-bit down counters that can generate interrupts on reaching zero.

See *Timers* on page 4-71 for details of PB11MPCore usage and the *ARM Dual-Timer Module (SP804) Technical Reference Manual* (ARM DDI 0271) for full programming details.

General Purpose Input/Output, GPIO

The *ARM PrimeCell General Purpose Input/Output (GPIO)* PL061 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The PrimeCell GPIO provides eight programmable inputs or outputs, both software and hardware control modes are supported. An interrupt interface is provided to configure any number of pins as level of transitional interrupt sources.

See *General Purpose Input/Output, GPIO* on page 4-53 for details of PB11MPCore usage and the *ARM PrimeCell General Purpose Input/Output (PL061) Tecnical Reference Manual* (ARM DDI 0190) for full programming details.
The physical interface provided by the PB11MPCore is shown in *GPIO interface* on page A-10.

Generic Interrupt Controller

The Generic Interrupt controller (GIC) is a custom *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*

The custom GIC can accept interrupts from up to 16 sources and generates **nFIQ** and **nIRQ** responses for the system.

See *Generic Interrupt Controller, GIC* on page 4-55 for details of PB11MPCore usage, and *Interrupts* on page 3-48 for routing options.

Status and System Control Register Block

The baseboard status and system control registers enable the ARM11 MPCore to determine its environment and to control the on-board systems.

See *Status and system control registers* on page 4-11 for a description of each register.

System Controller

The ARM *System Controller* SP810 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The System Controller provides an interface to control the operation of subsystems within the Southbridge. It supports the following functionality:

- a system mode control state machine
- crystal and PLL control
- definition of system response to interrupts
- reset status capture and soft reset generation
- Watchdog and Timer module clock enable generation
- remap control
- general purpose peripheral control registers
- system/peripheral clock control and status.

See *System Controller (SYSCTRL)* on page 4-43 for details of PB11MPCore usage and the *PrimeXsys System Controller (SP810) Tecnical Reference Manual* (ARM DDI 0254) for full programming details.

UART

The *ARM PrimeCell UART* PL011 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The PrimeCell UART performs serial-to-parallel conversion of received data and parallel-to-serial conversion of transmitted data. Separate receive and transmit FIFO buffers, a progammable Baud rate generator, hardware or software flow control, and modem support functions are provided.

Note

An IrDA SIR ENDEC interface is implemented but this is not supported by the PB11MPCore.

See *UART* on page 4-72 for details of PB11MPCore usage and the *ARM PrimeCell UART (PL011) Tecnical Reference Manual* (ARM DDI 0183) for full programming details.
The physical interface provided by the PB11MPCore is shown in *UART interface* on page A-11.

Synchronous Serial Port, SSP

The ARM PrimeCell *Synchronous Serial Port (SSP)* PL022 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The PrimeCell SSP is a master or slave interface that enables synchronous serial communication with slave or master peripherals having one of the following:

- a Motorola SPI-compatible interface
- a Texas Instruments synchronous serial interface
- a National Semiconductor Microwire interface.

See *Synchronous Serial Port, SSP* on page 4-69 for details of PB11MPCore usage and the *ARM PrimeCell Synchronous Serial Port (PL022) Tecnical Reference Manual* (ARM DDI 0194) for full programming details.
The physical interface provided by the PB11MPCore is shown in *Synchronous Serial Port interface* on page A-12.

Smart Card Interface, SCI

The ARM PrimeCell *Smart Card Interface (SCI)* PL131 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The PrimeCell Smart Card Interface (SCI) interfaces to an external Smart Card reader. The SCI can autonomously control data transfer to and from the smart card. Transmit and receive data FIFOs are provided to reduce the required interaction between the host system and the peripheral.

See *Smart Card Interface, SCI* on page 4-68 for details of PB11MPCore usage and the *ARM PrimeCell Smart Card Interface (PL131) Tecnical Reference Manual* (ARM DDI 0228) for full programming details.
The physical interface provided by the PB11MPCore is shown in *Smart Card interface* on page A-13.

Two-wire serial bus interface

The FPGA implements a custom two-wire serial bus interface that is used to identify the PISMO memory expansion modules present in the PISMO expansion memory socket, and to read and set the time-of-year (TOY) clock on the baseboard. A second interface is used to identify and control display equipment connected to the DVI connector on the rear panel of the ATX enclosure.

Each device on the serial bus has its own slave address. The unique write and read addresses for the PISMO memory and the TOY slave on the serial bus are listed in Table 3-2. The address for the DVI display slave is device dependant and must be obtained from the display manufacturer.

Table 3-2 Serial interface device addresses

Device	Write address	Read address	Description
PISMO (static memory module)	`0xA2`	`0xA3`	Identifies the type of memory on the board and how it is configured.
TOY (DS1338 RTC)	`0xD0`	`0xD1`	Reads time data and writes control data to the RTC.
DVI (external display)	display dependant	display dependant	Reads the capabilities of the external display connected to the DVI connector on the rear panel of the ATX enclosure. Can control display settings of *E-DDC* displays.

See *Two-wire serial bus interface, SBCon* on page 4-65 for details of PB11MPCore usage and for more information on programming the interface.

Real Time Clock, RTC

The ARM PrimeCell *Real Time Clock (RTC)* PL031 is an *Advanced Microcontroller Bus Architecture (AMBA)* slave block that connects to the *Advanced Peripheral Bus (APB)*.

The PrimeCell Real Time Clock (RTC) can be used to provide a basic alarm function or long time base counter. An interrupt signal is generated after counting for a programmed number of cycles of real time clock input. Counting in one second intervals is achieved by use of a 1Hz clock input to the PrimeCell RTC.

See *Real Time Clock, RTC* on page 4-64 for details of PB11MPCore usage and the *ARM PrimeCell Real Time Clock (PL031) Technical Reference Manual* (ARM DDI 0224) for full programming details.

3.6 Ethernet interface

The Ethernet interface is implemented using a SMCS LAN9118 10/100 Ethernet controller. The LAN9118 incorporates a *Media ACcess* (MAC) Layer, a *PHYsical* (PHY) layer, *Host Bus Interface* (HBI), receive and transmit FIFOs, power management controls, and a serial configuration EEPROM interface. The HBI models an asynchronous SRAM and interfaces directly to the Northbridge static memory bus.

The internal registers of the LAN9118 are mapped onto the Northbridge static memory bus starting at location `0x4E000000`.

When manufactured, an ARM value for the Ethernet MAC address is loaded into a configuration EEPROM connected to the Ethernet controller.

See *Ethernet* on page 4-52 for details of PB11MPCore usage and the *SMC LAN9118 Data Sheet* for full programming details.

3.7 USB Interface

The Nothbridge provides a SMC bus interface to an external Philips ISP1761 USB 2.0 controller. Three USB interfaces are provided on the baseboard,

The internal registers of the controller are memory-mapped starting at `0x4F000000`.

USB port 1 provides an OTG device interface and connects to the mini USB A-Type connector on the front panel of the enclosure.

USB ports 2 and USB port 3 can function in either master or slave mode and connect to the dual A-Type connector on the rear panel of the enclosure (USB port 2 is the top connector).

Note

The USB config interface has a dedicated USB controller and connects to the USB B-Type connector on the front panel of the enclosure. See *USB config interface* on page 3-32 and *Debug and Config port support* on page 3-54 for details.

3.8 USB config interface

An ARM custom PLD design provides access to the internal JTAG signals via a dedicated USB config port. The USB interface is implemented by an external FTDI FT245BL USB 1.0 controller. A USB B-Type interface connector is provided on the front panel of the enclosure for USB config access by the host PC.

3.9 DVI Interface

The PL111 CLCD controller in the Northbridge is interfaced to the *Digital Visual Interface* (DVI) port on the PB11MPCore using on-board components to provide support for both analog and digital displays. See *CLCD controller* on page 3-16 for details.

The digital portion of the interface is provided by a Silicon Image SiI 160 *Transition Minimized Differential Signaling* (T.M.D.S.) transmitter. The device is VESA compliant, operating in One Pixel/Clock Input/Output mode. DVI resolutions up to XVGA (1024x768) are supported on the PB11MPCore.

The analog portion of the interface is provided by a Texas Instruments THS8134B triple high speed *Digital to Analog Converter* (DAC). The DAC converts the 8-bit RGB data from the PL111 CLCD controller to analog VGA signals.

The DDC2B portion of the interface is provided by a custom *Two-wire Serial Interface* (SBCon) implemented the Southbridge. See *Two-wire serial bus interface* on page 3-28 for details.

The physical interface provided by the PB11MPCore is shown in *DVI display interface* on page A-17.

Note

The on-board DVI interface can be controlled either by the PL111 *Color LCD Controller* implemented in the Northbridge, or by logic implemented in the FPGA on an attached Logic Tile. See *Tile interconnections* on page 3-10 and *RealView Logic Tile header connectors* on page A-19 for details.

3.10 PCI interface

The PCI interface comprises:

- AXI to PCI bridge implemented in the Northbridge
- PCI-X to PCI asynchronous bridge (PCI 6520)
- PCI-X to PCI Express asynchronous bridge (PEX 8114)
- PCI Express switch (PEX 8518).

Figure 3-9 shows the PB11MPCore PCI and PCI Express implementation.

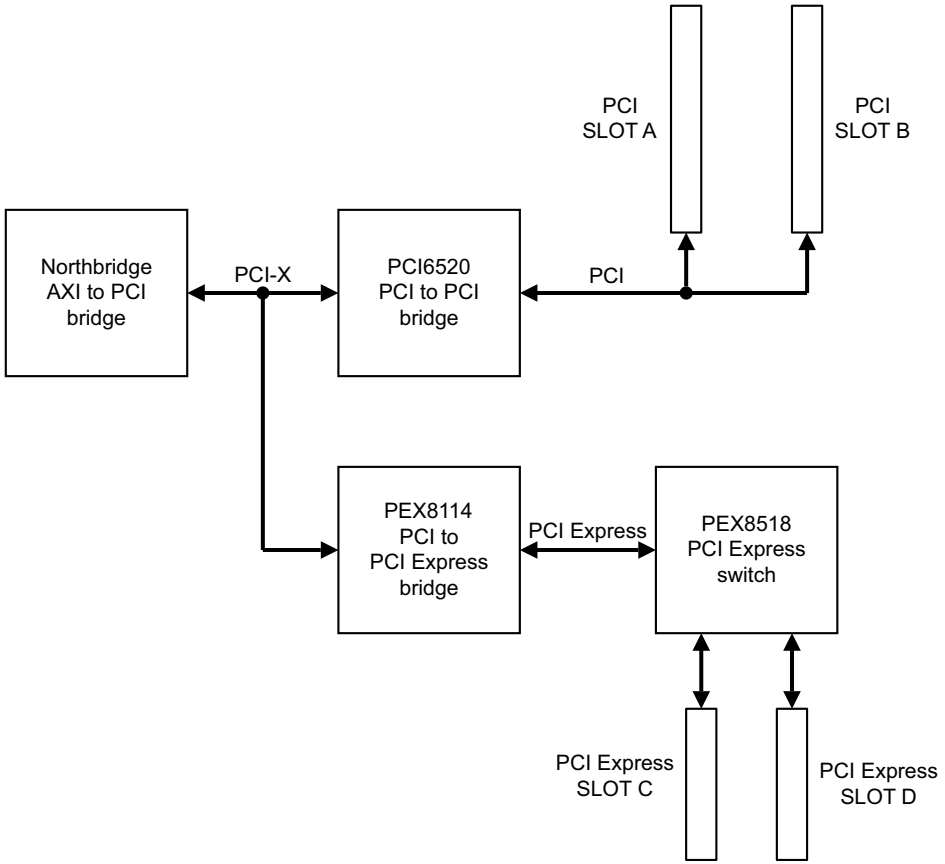


Figure 3-9 PCI-PCI Express interface

The PCI and PCI-X clocks are sourced from the PB11MPCore:

- PCI clock rate is 66MHz (default) or 33MHz
- PCI-X clock rate is 100MHz (embedded in a 2.5GHz signalling rate as defined in the PCI Express specification).

The AXI to PCI bridge provides interface functions conforming to the PCI or PCI-X Specification. The PB11MPCore configures the interface for PCI-X operation.

The PCI-X to PCI asynchronous bridge (PCI6520) interfaces the PCI bridge to the PB11MPCore PCI slots.

The PCI-X to PCI Express asynchronous bridge (PEX8114) and PCI Express switch (PEX8518) interface the PCI bridge to the PB11MPCore PCI Express slots.

See *AXI to PCI bridge* on page 4-62 for details of PB11MPCore usage and the *PCI-to-PCI Bridge Architecture Specification Revision 1.2* and the *PCI Express Base Specification Revision 1.1* for further details.

For details on the bridge and switch components used on the PB11MPCore see the PLX Technology Inc website: `www.plxtech.com`.

3.11 Power supply control

The Southbridge implements control registers and external interfaces to converters to enable you to:

- change the ARM11 MPCore test chip voltages by writing values to the serially-programmed DACs
- read PB11MPCore on-board voltages and currents from an 8-channel 12-bit ADC.

The voltage control and voltage and current monitoring scheme is shown in Figure 3-10.

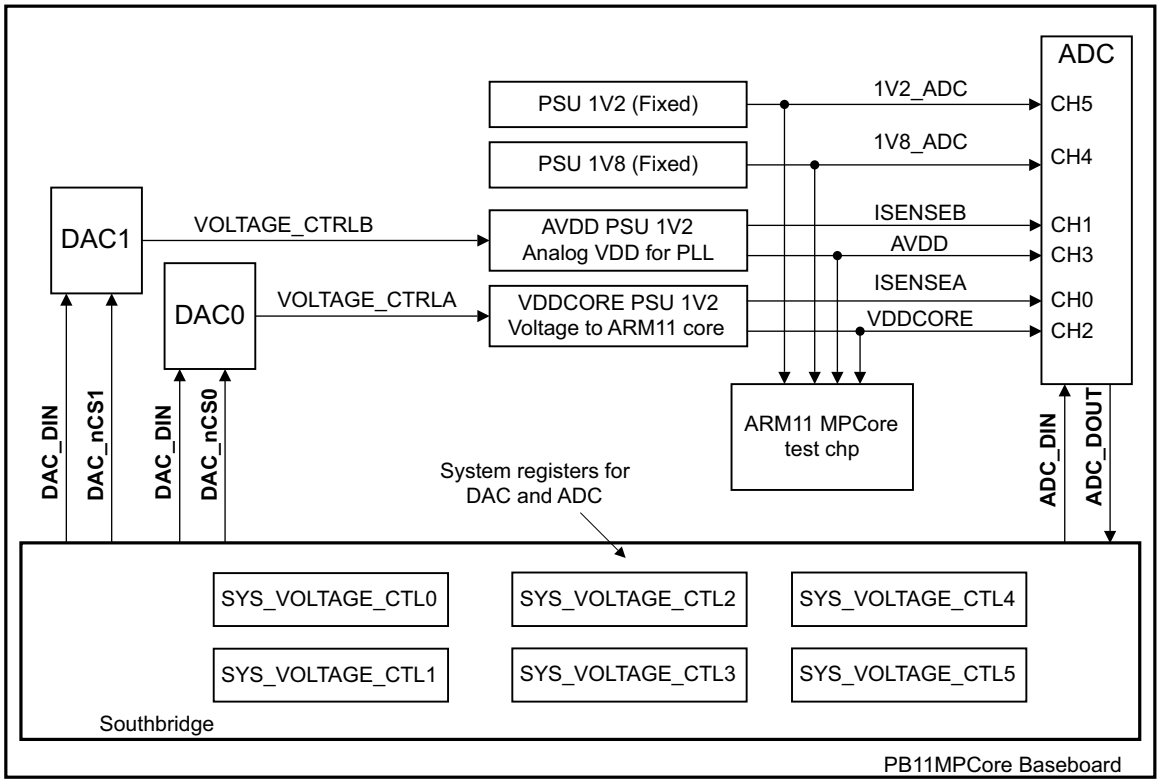


Figure 3-10 Voltage control and voltage and current monitoring

3.11.1 Setting the ARM11 MPCore test chip voltages

The two DACs are continually updated with voltage control values written into the **SYS_VOLTAGE_CTLx** registers in the Southbridge.

There is a voltage adjustment range of ±0.25V for both **VDDCORE** (Core voltage) and **AVDD** (Analog VDD supply voltage to the PLL). The default values loaded into the registers at power-on are:

- `0xAF` for **SYS_VOLTAGE_CTL0** (**VDDCORE**)
- `0x80` for **SYS_VOLTAGE_CTL1** (**AVDD**).

A value of `0xFF` gives maximum negative offset from the default (–0.25V) and a value of `0x0` gives maximum positive offset from the default (+0.25V). See also *Voltage control registers, SYS_VOLTAGE_CTLx* on page 4-41.

Caution

ARM recommends that the maximum variation in supply voltage be less than +/– 10%. Operation outside this range might be unreliable or damage the test chip.

3.11.2 Reading the ARM11 MPCore test chip voltages

The **SYS_VOLTAGE_CTLx** registers in the Southbridge are continually updated with test chip voltage values provided by the ADC. The relationship between the voltage and the LSB of the register field is given by the following formula:

V_{DDx} = SYS_VOLTAGE_CTL[19:8] * 2.5/4096 volts (2.5V full scale)

See also *Voltage control registers, SYS_VOLTAGE_CTLx* on page 4-41.

3.11.3 Reading the ARM11 MPCore test chip currents

The **SYS_VOLTAGE_CTLx** registers in the Southbridge are continually updated with voltage values provided by the ADC that are proportional to the test chip currents. The relationship between the test chip currents and the LSB of the register field is given by the following formulae:

For **VDDCORE**:

I_{DDx} = SYS_VOLTAGE_CTL[19:8] * 1/4096 ampere (1A full scale)

For **AVDD**:

I_{DDx} = SYS_VOLTAGE_CTL[19:8] * 1/81920 ampere (50mA full scale)

See also *Voltage control registers, SYS_VOLTAGE_CTLx* on page 4-41.

3.12 Clock architecture

The main clock routing for the PB11MPCore is shown in Figure 3-11. The PCI clock routing is shown separately in Figure 3-13 on page 3-42.

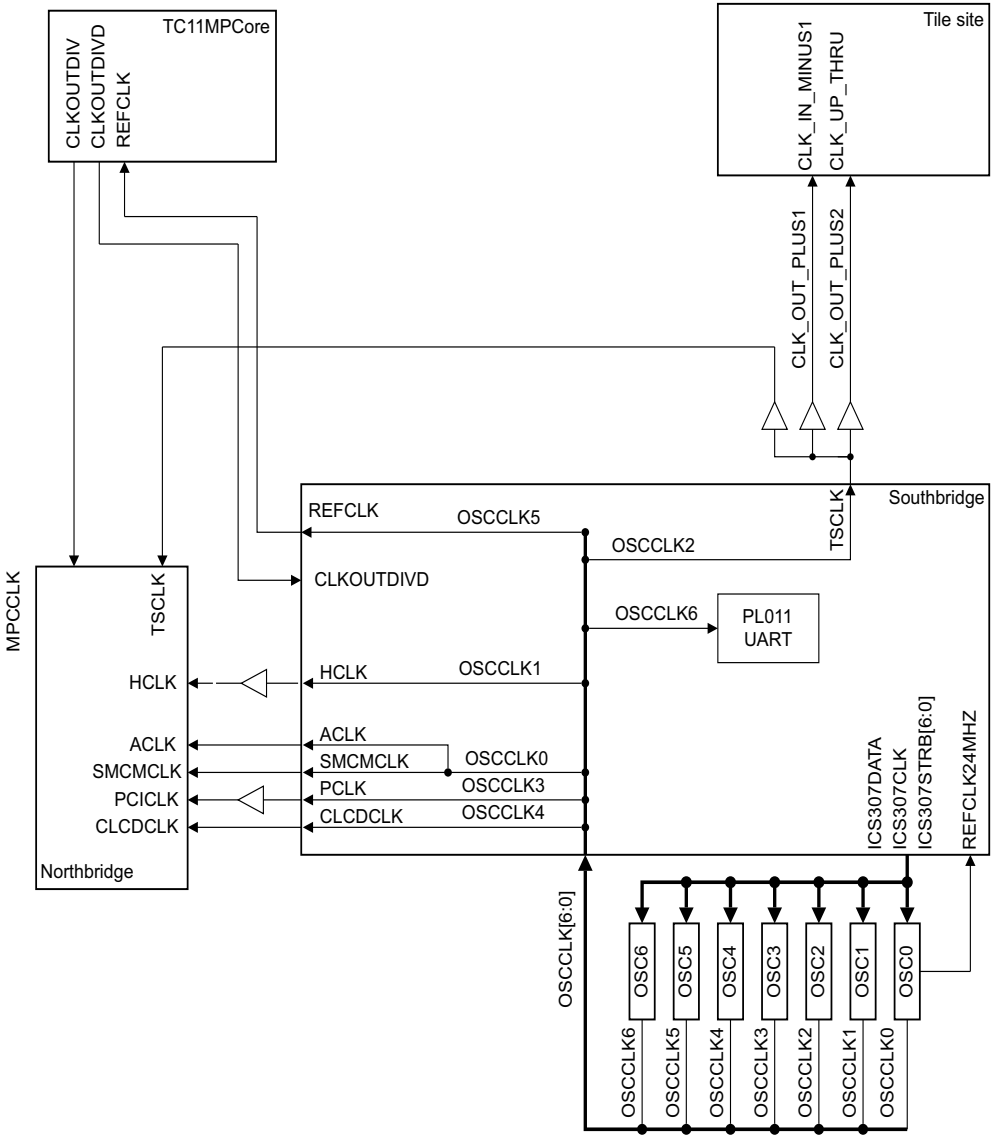


Figure 3-11 Clock architecture

3.12.1 PB11MPCore clocks

This section describes the clocks used by the PB11MPCore.

ICS307 programmable clock generators

Seven programmable (6–200 MHz) clocks, **OSCCLK[6:0]** are supplied to the Southbridge by the programmable MicroClock ICS307 clock generators (OSC0–OSC6):

OSCCLK0 Default frequency: 100MHz.

- generates **ACLK** for the Northbridge:
  - AXI infrastructure
  - PL340 Dynamic Memory Controller
  - PL081 DMA Controller
  - internal memory
  - peripheral configuration.
- generates **SMCCLK** for Northbridge (50MHz).

OSCCLK1 Default frequency: 40MHz.
Generates **HCLK** for the AHB interface between the Northbridge and Southbridge.

OSCCLK2 Default frequency: 25MHz.
Generates **TSCLK** for distribution to the tile site. **TSCLK** is also supplied to the Northbridge async bridges to synchronize AXI signals to and from the tile site.

OSCCLK3 Default frequency: 50MHz.
Generates **PCLK** the reference clock for the PCI unit in the Northbridge, and the external PCI and PCI Express bridges.

OSCCLK4 Default frequency: 25MHz.
Generates **CLCDCLK** the reference clock for the PL111 CLCD controller in the Northbridge, and the external video DAC and DVI transmitter (1024x768 resolution at 60Hz frame rate support).

OSCCLK5 Default frequency: 70MHz.
Generates **REFCLK** for the ARM11 MPCore test chip PLL.
See *Clock domain overview* on page 5-4 for details of the ARM11 MPCore test chip clocks.

OSCCLK6 Default frequency: 24MHz.
Generates **UARTCLK** the reference clock for the PL011 UART in the Southbridge.

The output frequencies of the ICS307s are controlled by divider values loaded into the serial data input pins on the oscillators.

The serial interface logic is implemented in the Southbridge. The only user interface to the clock control logic is through the SYS_OSCx registers. See *Oscillator Registers, SYS_OSCx* on page 4-17 and *Oscillator reset registers, SYS_OSCRESETx* on page 4-40 for programming details.

You can calculate the oscillator output frequency from the formula:

$$OSCCLKx = \frac{48 \times (VDW+8)}{(RDW+2) \times DIVIDE} \text{ MHz}$$

where:

VDW Is the VCO divider word (4 – 511) from SYS_OSCx[8:0]

RDW Is the reference divider word (1 – 127) from SYS_OSCx[15:9]

OD Is the output divider select (2 to 10) selected from SYS_OSCx[18:16]:

- `b000` selects divide by 10
- `b001` selects divide by 2
- `b010` selects divide by 8
- `b011` selects divide by 4
- `b100` selects divide by 5
- `b101` selects divide by 7
- `b110` selects divide by 3
- `b111` selects divide by 6.

For more information on the ICS clock generator and a frequency calculator, see the IDT web site at `www.idt.com`.

A crystal on the board provides a fixed frequency 24MHz reference clock for the programmable oscillators OSC0-OSC6 and to generate other fixed frequency clocks in the design.

Northbridge clocks

The clock domains within the Northbridge are shown in Figure 3-12.

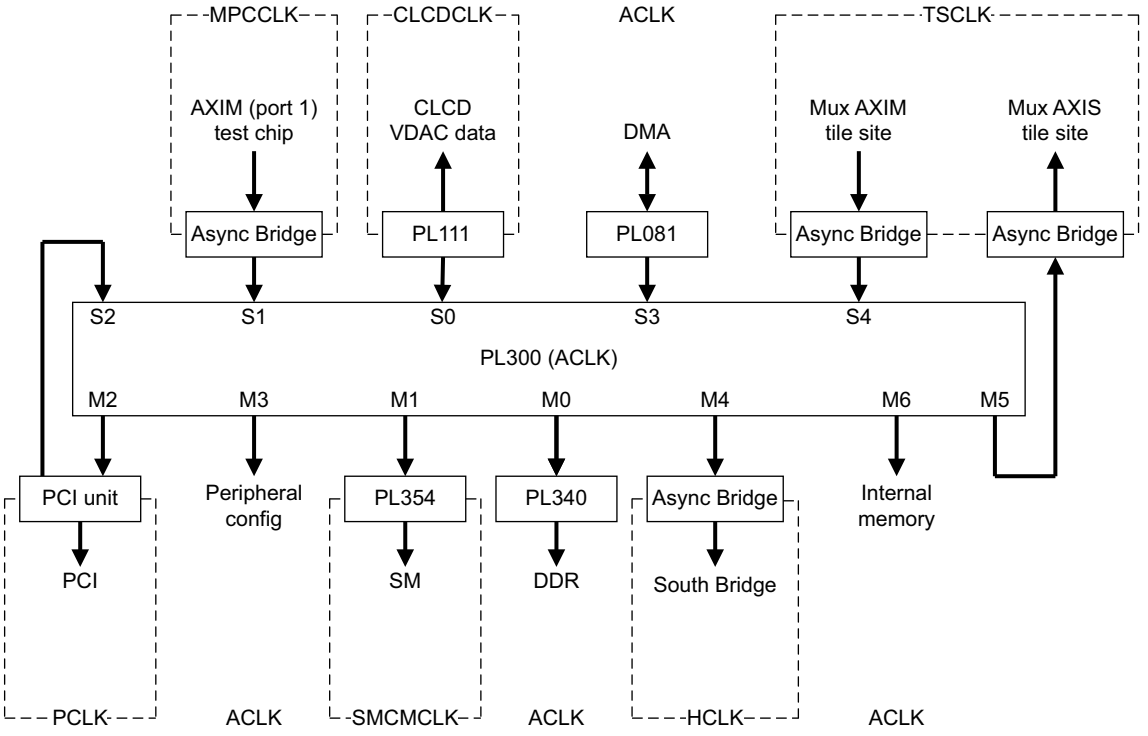


Figure 3-12 Northbridge clock domains

Southbridge clocks

The UART, Smart Card Interface (SCI), and Synchronous Serial Port (SSP) are clocked from a 24MHz reference clock.

The Dual Timer Counter modules and the Watchdog modules are clocked by a 1MHz reference clock.

3.12.2 PCI and PCI Express clocks

The PCI and PCI Express clock routing is shown in Figure 3-13.

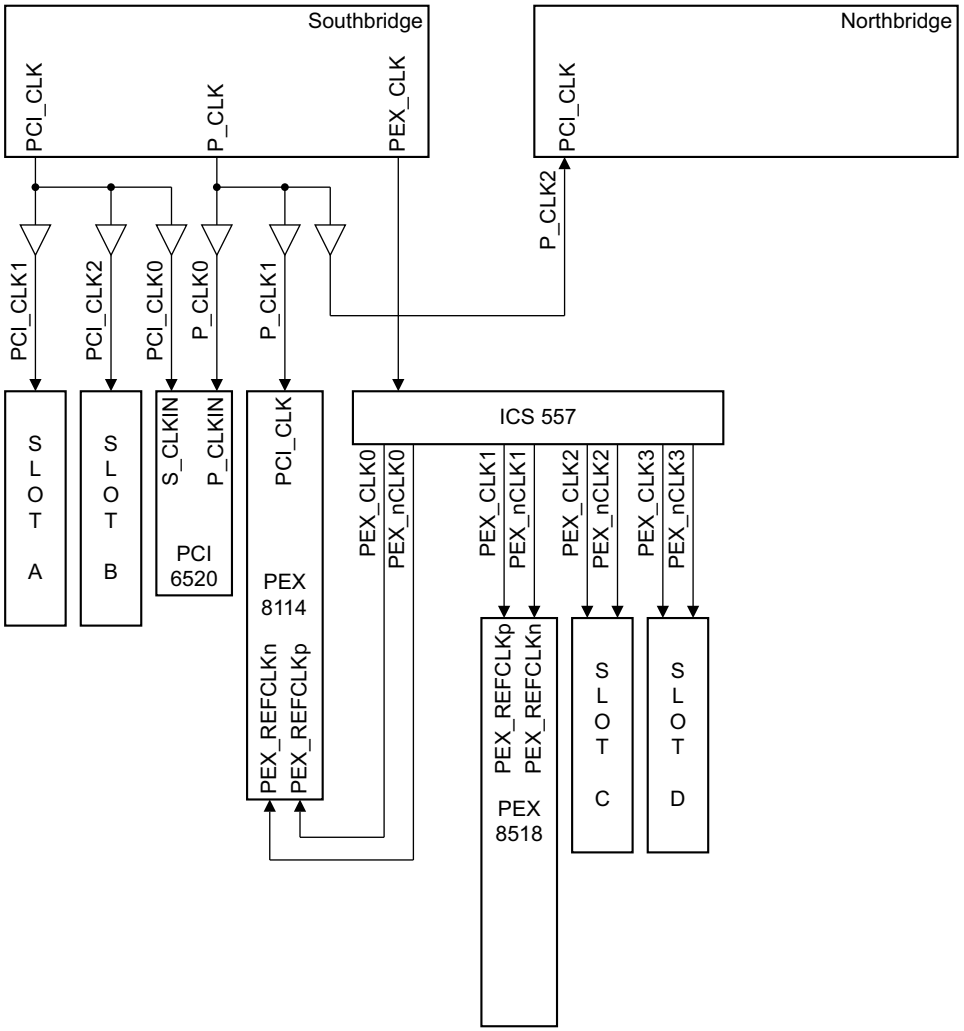


Figure 3-13 PCI and PCI Express clock routing

PCI_CLK Clock source for the PCI slots A and B and the secondary port on the asynchronous PCI to PCI bridge (PLX 6520).

Note

The clock frequency is determined by the lowest frequency card inserted into Slot A or B.

P_CLK Clock source for the PB11MPCore PCI Unit in the Northbridge and the primary port on the asynchronous PCI to PCI bridge (PLX 6520).

PEX_CLK Clock source for the ICS557 clock generator that provides the 100MHz differential clocks to drive the PCI Express slots C and D.

3.12.3 ARM11 MPCore test chip clocks

The ARM11 MPCore test chip includes a PLL and Clock Divider. These components generate the clocks required by the ARM11 MPCore and test chip subsystems and the external clocks **CLKOUTDIV** and **CLKOUTDIVD** used by the Northbridge and Southbridge respectively. Figure 3-14 on page 3-44 shows a simplified diagram of the ARM11 MPCore test chip clock generation scheme and the associated clock domains.

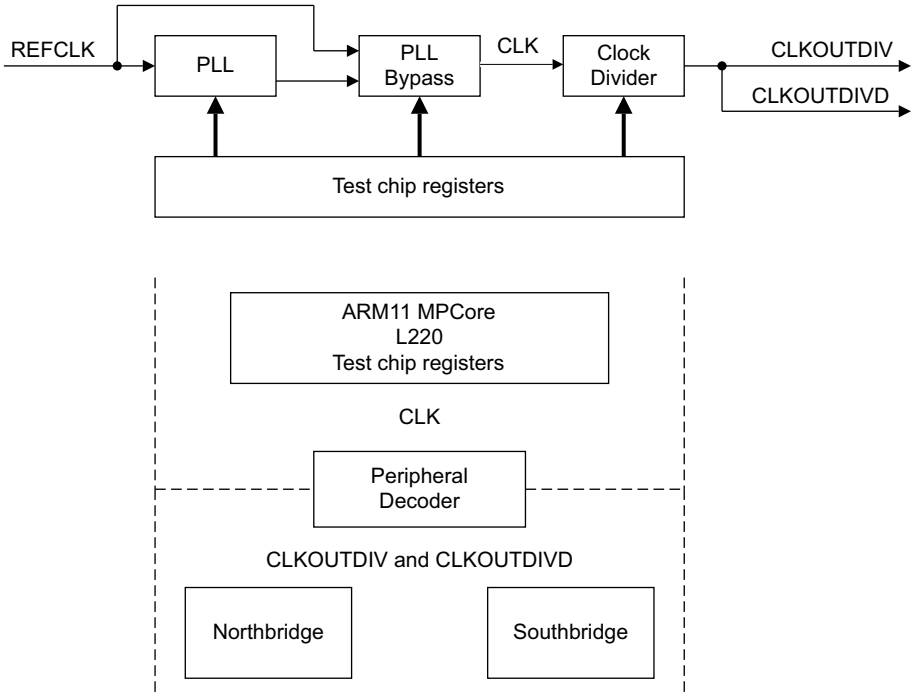


Figure 3-14 ARM11 MPCore test chip clocks

The default clock frequencies are:

CLK ARM11 MPCore, L220: 210MHz.

CLKOUTDIV AXI bus to Northbridge: 70MHz.

CLKOUTDIVD AXI bus to Southbridge: 70MHz.

ARM11 MPCore test chip clock generation is described in more detail on page 5-4 of Chapter 5 *Processor Sub-System*.

3.13 Resets

The resets domains for the PB11MPCore are shown in Figure 3-15.

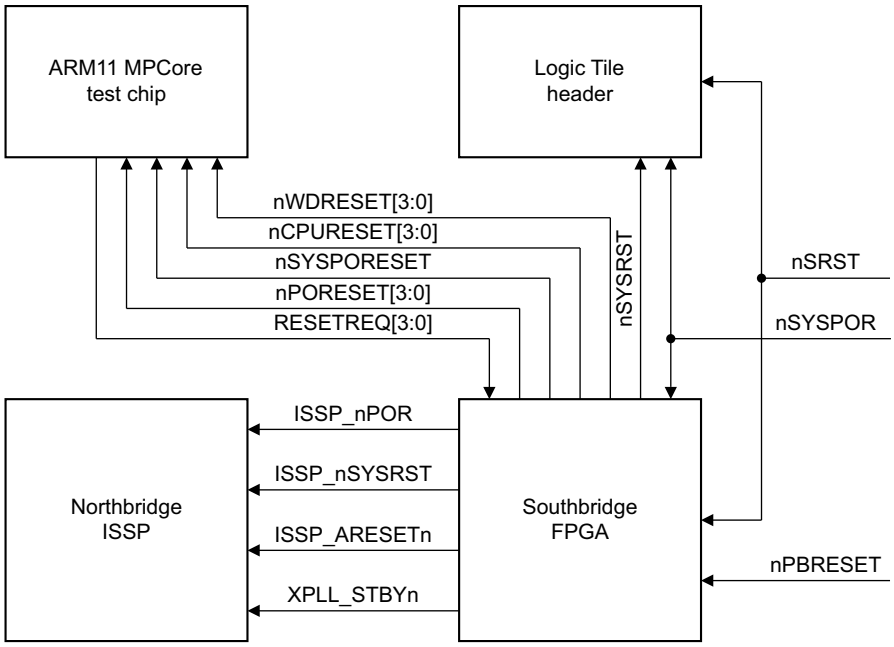


Figure 3-15 Reset routing

The Reset Controller state diagram is shown in Figure 3-16 on page 3-46

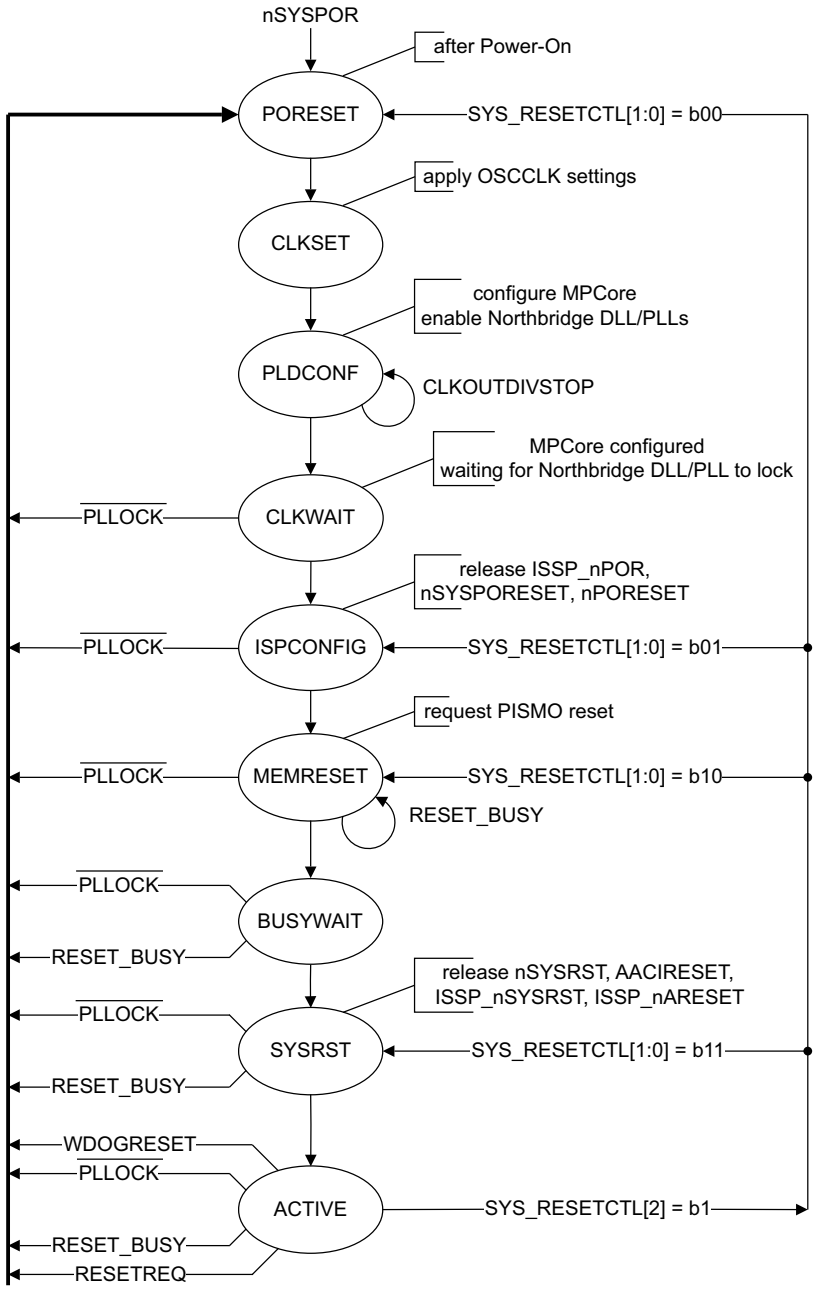


Figure 3-16 Reset Controller state diagram

The reset timing is shown in Figure 3-17.

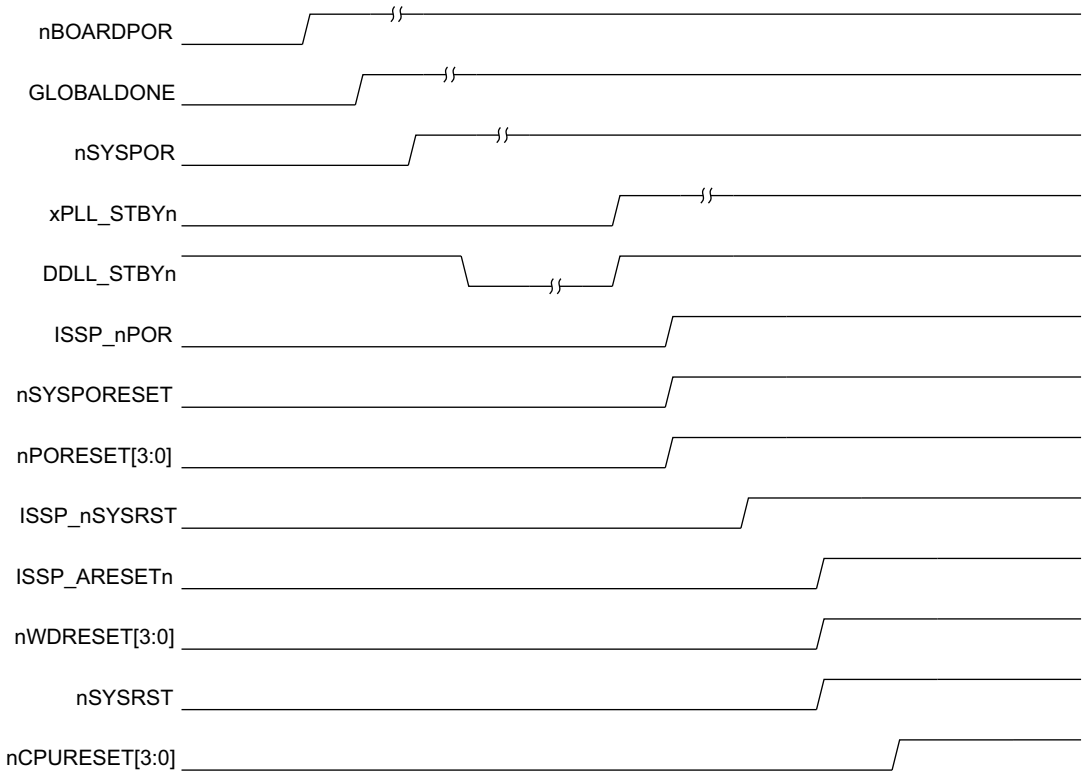


Figure 3-17 Reset timing

Note

If a peripheral implemented on a Logic Tile fitted to the tile site needs to generate a system level reset it can pulse the **nSRST** signal low. This will cause the **nSYSRST** signal to pulse low resetting the system. This is a system level reset, it will not generate a power on reset.

3.14 Interrupts

The PB11MPCore implements four custom *Generic Interrupt Controllers* (GICs) in the Southbridge and a *Distributed Interrupt Controller* is implemented in the ARM11 MPCore multiprocessor in the test chip.

3.14.1 Generic Interrupt Controller, GIC

The custom *Generic Interrupt controller* (GIC) is an AMBA compliant SoC peripheral that is developed and tested by ARM Limited.

The GICs accept interrupts from peripherals in the Northbridge, Southbridge, on-board peripherals, and the tile site. The GICs generate **nFIQ** and **nIRQ** signals to the ARM11 MPCore test chip and the tile site:

GIC0 generates the ARM11 MPCore **nIRQ**

GIC1 generates the ARM11 MPCore **nFIQ**

GIC2 generates the tile site **nIRQ**

GIC3 generates the tile site **nFIQ**

Figure 3-18 shows the interrupt routing.

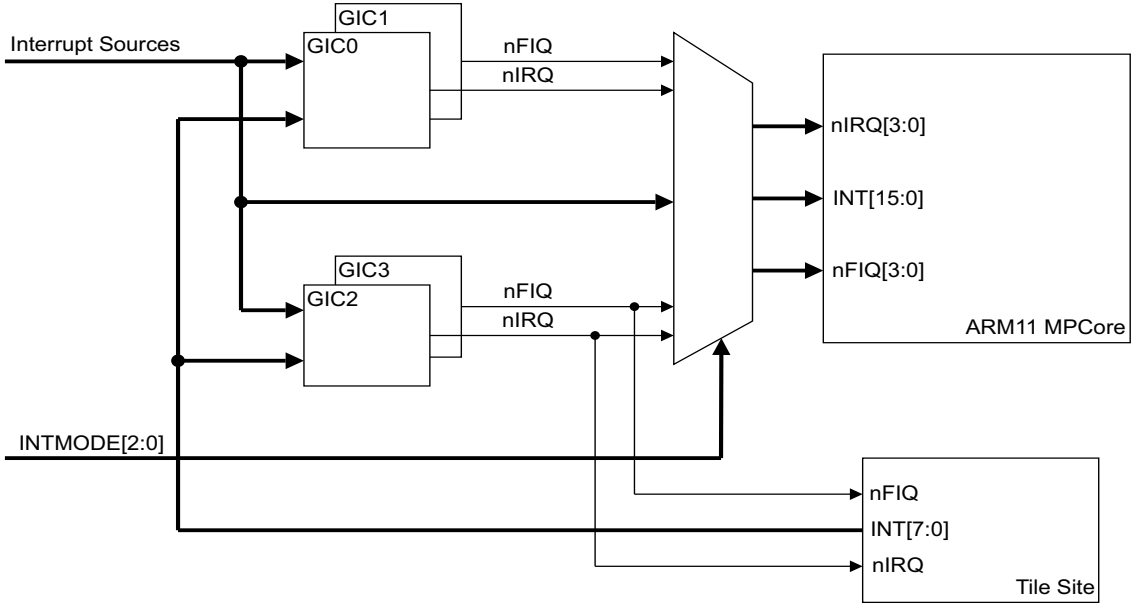


Figure 3-18 External and internal interrupt sources

3.14.2 ARM11 MPCore Distributed Interrupt Controller

The Distributed Interrupt Controller collates interrupts for the individual processors in the ARM11 MPCore from a large number of sources and provides:

- masking of interrupts
- prioritization of the interrupts
- distribution of the interrupts to the target processors
- tracking the status of interrupts
- generation of interrupts by software.

The Distributed Interrupt Controller control registers are accessed through the ARM11 MPCore *Snoop Control Unit* (SCU).

For the PB11MPCore:

The base address is `0x1F001000` and the address range is `0x1F001000` to `0x1F001FFF`.

See the *MPCore Multiprocessor Technical Reference Manual* (DUI 0360) for details.

3.14.3 Distributed Interrupt Controller routing

A total of 32 interrupt lines, **INT[31:0]** are provided by the Distributed Interrupt Controller in the ARM11 MPCore:

- Interrupt lines **INT[31:16]** are used internally by the ARM11 MPCore test chip. See *Interrupt Routing* on page 5-16 for details of their use.
- Interrupt lines **INT[15:0]** are available for external use and routing is controlled by the PB11MPCore.

Three routing options are provided:

- Legacy Mode interrupt routing
- New Mode with DCC interrupt routing
- New Mode without DCC interrupt routing

See *PLD control register 1, SYS_PLD_CTRL1* on page 4-32 for details of the INTMODE[2:0] field that controls the routing mode.

Table 3-3 lists Legacy Mode interrupt routing for ARM11 MPCore **INT[15:0]** (INTMODE = `b000`).

Table 3-3 Legacy Mode: INT[15:0]

ARM11 MPCore Interrupt	Southbridge Direction	Source
INT[0]	Input	reserved, set to zero
INT[1]	Input	reserved, set to zero
INT[2]	Input	reserved, set to zero
INT[3]	Input	reserved, set to zero
INT[4]	Input	reserved, set to zero
INT[5]	Input	reserved, set to zero
INT[6]	Input	reserved, set to zero
INT[7]	Input	reserved, set to zero
INT[8]	Output	GIC0 **nIRQ**
INT[9]	Output	GIC2 **nIRQ**
INT[10]	Output	reserved, set to zero
INT[11]	Output	reserved, set to zero
INT[12]	Output	reserved, set to zero
INT[13]	Output	reserved, set to zero
INT[14]	Output	reserved, set to zero
INT[15]	Output	reserved, set to zero

Table 3-4 lists Legacy Mode interrupt routing for ARM11 MPCore **nIRQ** and **nFIQ** (INTMODE = `b0000`).

Table 3-4 Legacy Mode: nIRQ and nFIQ

ARM11 MPCore nIRQ and nFIQ	ARM11 MPCore processor	Source
nIRQ0	CPU#[0]	GIC0 **nIRQ**
nIRQ1	CPU#[1]	GIC2 **nIRQ**
nIRQ[3:2]	CPU#[3:2]	`b11`
nFIQ{3:0]	CPU#[3:0]	`b1111`

Table 3-5 lists new mode with DCC interrupt routing for ARM11 MPCore **INT[15:0]** (INTMODE = `b001`).

Table 3-5 New Mode with DCC: INT[15:0]

ARM11 MPCore Interrupt	Southbridge Direction	Source
INT[0]	Output	**AACINTR**
INT[1]	Output	**TIMERINT01**
INT[2]	Output	**TIMERINT23**
INT[3]	Output	**USBINT**
INT[4]	Output	**UARTINT0**
INT[5]	Output	**UARTINT1**
INT[6]	Output	**RTCINT**
INT[7]	Output	**KMIINT0**
INT[8]	–	reserved, set to zero
INT[9]	–	reserved, set to zero
INT[10]	–	reserved, set to zero
INT[11]	–	reserved, set to zero
INT[12]	Output	GIC1 **nFIQ**

Table 3-5 New Mode with DCC: INT[15:0] (continued)

ARM11 MPCore Interrupt	Southbridge Direction	Source
INT[13]	Output	GIC3 **nFIQ**
INT[14]	Output	**MCIINTR[0]**
INT[15]	Output	**MCIINTR[1]**

Table 3-6 lists New Mode with DCC interrupt routing for ARM11 MPCore **nIRQ** and **nFIQ** (INTMODE = `b001`).

Table 3-6 New Mode with DCC: nIRQ and nFIQ

ARM11 MPCore nIRQ and nFIQ	ARM11 MPCore CPU	Source
nIRQ[3:0]	CPU#[3:0]	`b1111`
nFIQ[3:0]	CPU#[3:0]	`b1111`

Table 3-7 lists new mode without DCC interrupt routing for ARM11 MPCore **INT[15:0]** (INTMODE = `b010`).

Table 3-7 New mode without DCC: INT[15:0]

ARM11 MPCore Interrupt	Southbridge Direction	Source
INT[0]	Output	**AACINTR**
INT[1]	Output	**TIMERINT01**
INT[2]	Output	**TIMERINT23**
INT[3]	Output	**USBINT**
INT[4]	Output	**UARTINT0**
INT[5]	Output	**UARTINT1**
INT[6]	Output	**RTCINT**
INT[7]	Output	**KMIINT0**
INT[8]	Output	**KMIINT1**

Table 3-7 New mode without DCC: INT[15:0] (continued)

ARM11 MPCore Interrupt	Southbridge Direction	Source
INT[9]	Output	**ETHINTR**
INT[10]	Output	GIC0 **nIRQ**
INT[11]	Output	GIC2 **nIRQ**
INT[12]	Output	GIC1 **nFIQ**
INT[13]	Output	GIC3 **nFIQ**
INT[14]	Output	**MCIINTR[0]**
INT[15]	Output	**MCIINTR[1]**

Table 3-8 lists New Mode without DCC interrupt routing for ARM11 MPCore **nIRQ** and **nFIQ** (INTMODE = `b010`).

Table 3-8 New Mode without DCC: nIRQ and nFIQ

ARM11 MPCore nIRQ and nFIQ	ARM11 MPCore CPU	Source
nIRQ[3:0]	CPU#[3:0]	`b1111`
nFIQ[3:0]	CPU#[3:0]	`b1111`

3.15 Test, configuration, and debug interfaces

The following test and configuration interfaces are located on the PB11MPCore:

- configuration switches, see *Baseboard configuration switches* on page 2-7
- JTAG, see *Debug and Config port support*
- Logic analyzer, see *Integrated logic analyzer (ILA)* on page 3-57
- Boot Monitor, see *Using the baseboard Boot Monitor and platform library* on page E-4.

3.15.1 Debug and Config port support

The PB11MPCore supports debugging and configuration using embedded and external hardware. The debugging interface is controlled by JTAG, the configuration interface is controlled by either JTAG or USB.

JTAG hardware

The RealView Debugger, for example, uses an external RealView ICE interface box to connect to the JTAG connector on the rear panel. See *JTAG connector* on page A-41 for pinout details.

USB config port

The USB config port is implemented by a PLD on the PB11MPCore. An application, `Progcards USB` can control the JTAG config signals from the USB port of the PC when the PC is connected to the USB config port on the front panel by a standard USB cable. See *USB debug connector* on page A-42 for pinout details.

Note

ARM RealView ICE grounds pin 20 of the JTAG connector. On the PB11MPCore, pin 20 is connected to a pull-up resistor and the **nICEDETECT** signal. The USB config port is automatically disabled if a JTAG emulator is connected and **nICEDETECT** is LOW. If you are using third-party debugging hardware, ensure that a ground is present on pin 20 of the JTAG connector.

The PB11MPCore has two scan chains:

Debug The **D_x** signals are used for the test chip and synthesized JTAG TAP controllers in the RealView Logic Tile. This is the normal mode of operation, see *JTAG debug (normal) mode* on page 3-55.

Config The **C_x** signals are used to program the FPGA and PLDs. This chain is available in configuration mode, see *JTAG configuration mode* on page 3-55. See also *Integrated logic analyzer (ILA)* on page 3-57.

JTAG debug (normal) mode

During normal operation and software development, the PB11MPCore operates in debug mode.

The debug mode is selected by default when the Config switch on the front panel is OFF, see Figure 3-2 on page 3-4.

In debug mode:

- The CONFIG LED is off on the front panel (and on each tile in the stack).
- The JTAG signals are routed through the ARM11 MPCore test chip.
- The JTAG scan path is rerouted sequentially to:
  - Logic Tile debug scan chain (**D_x** signals)
  - Southbridge debug scan chain (**D_x** signals)
  - ARM11 MPCore debug unit scan chain.
- A debugger (RealView Debugger for example), connected to the JTAG ICE connector on the rear panel, controls the scan chain.
- An Integrated Logic Analyzer (ILA) connected to the ChipScope connector (J9) on the baseboard can be used to debug the FPGAs on stacked tiles while the debugger is examining code on the ARM11 MPCore. The signals are routed through the Logic Tile configuration scan chain (**C_x** signals).
- the FPGAs in the system load their images from configuration flash.

JTAG configuration mode

This mode is selected when the CONFIG switch on the front panel is ON, see Figure 3-2 on page 3-4.

In configuration mode:

- the CONFIG LED is lit on the front panel (and on each tile in the stack)
- the JTAG scan path is rerouted sequentially to:
  - Logic Tile configuration scan chain (**C_x** signals)
  - Southbridge configuration scan chain (**C_x** signals)
  - Configuration PLD scan chain
  - Chip Select PLD scan chain
  - Northbridge scan chain
  - ARM11 MPCore test chip boundary scan chain.

- a configuration utility, `Progcards` for example, controls the scan chain
- the debug unit in the ARM11 MPCore is not visible and is replaced by the boundary scan chain in the test chip that is used for board-level production testing
- the board can be configured or upgraded in the field using JTAG equipment or the onboard USB config port
- the non-volatile PLD devices can be reprogrammed directly by JTAG
- the volatile FPGA images can be loaded from either JTAG or the configuration flash memory.

Note

The configuration flash memory does not have a JTAG port, it is programmed using JTAG by loading a flash-loader design into the FPGAs and PLDs. The flash-loader then transfers data from the JTAG programming utility to the configuration flash.

Caution

Third party JTAG debug equipment may not support this method of programming configuration flash, where possible, you should use ARM RealView Debugger.

After configuration you must:

1. return the CONFIG switch on the front panel to OFF.
2. power cycle the development system.

Power measurement and TCK

The MPCore test chip can be put into low power state (WFI) when not executing code. Linux, for example, makes use of this feature to significantly reduce the total power consumption. A feature of the test chip requires **TCK** to be low while in WFI state to reduce power. By default, if the JTAG connector is removed **TCK** is pulled high. External links must therefore be placed on the JTAG connector to make power measurements without the JTAG device connected:

- Link pins 9 and 10 on J10 to pull **TCK** low
- Link 18 and 20 for JTAG detect.

The JTAG connector is described in *JTAG* on page A-41. See also *Voltage control registers, SYS_VOLTAGE_CTLx* on page 4-41.

3.15.2 Integrated logic analyzer (ILA)

See *Integrated Logic Analyzer (ILA)* on page A-42 for pinout details. For more details on the integrated logic analyzer, see the ChipScope details on the Xilinx website (www.xilinx.com).

Chapter 4
Programmer's Reference

This chapter describes the memory map and the configuration registers for the peripherals in the baseboard FPGA. It contains the following sections:

- *Memory map* on page 4-3
- *Configuration and initialization* on page 4-9
- *Status and system control registers* on page 4-11
- *System Controller (SYSCTRL)* on page 4-43
- *Advanced Audio CODEC Interface, AACI* on page 4-46
- *Color LCD Controller, CLCDC* on page 4-48
- *Single Master Direct Memory Access Controller, SMDMAC* on page 4-50
- *Dynamic Memory Controller, DMC* on page 4-51
- *Ethernet* on page 4-52
- *General Purpose Input/Output, GPIO* on page 4-53
- *Generic Interrupt Controller, GIC* on page 4-55
- *Keyboard and Mouse Interface, KMI* on page 4-60
- *MultiMedia Card Interface, MCI* on page 4-61
- *AXI to PCI bridge* on page 4-62
- *Real Time Clock, RTC* on page 4-64
- *Two-wire serial bus interface, SBCon* on page 4-65

- *Smart Card Interface, SCI* on page 4-68
- *Synchronous Serial Port, SSP* on page 4-69
- *Static Memory Controller, SMC* on page 4-70
- *System Controller (SYSCTRL)* on page 4-43
- *Timers* on page 4-71
- *USB interface* on page 4-74
- *UART* on page 4-72
- *Watchdog* on page 4-76
- *CompactFlash interface* on page 4-77.

For detailed information on the programming interface for the ARM PrimeCells or other ARM IP, see the appropriate technical reference manual. For the DMA channels, interrupt signals, release versions, and any modifications made to the standard part, see the section of this chapter that describes the peripheral or controller.

Note

The peripherals and controllers implemented in the Southbridge are in the standard images distributed by ARM on the Versatile Family CD.

ARM do not recommend or support replacing the ARM IP peripherals and controllers implemented in the Southbridge with your own custom designs.

Custom IP development should be done using an attached Logic Tile for which ARM support is provided.

4.1 Memory map

The system memory map is divided with sections assigned to the Northbridge, Southbridge, and the Logic Tile site as shown in Table 4-1.

Table 4-1 System memory map

Owner	Address range	Bus type	Memory region size
Northbridge	0x00000000–0x0FFFFFFF	DDR	256MB (DMC mirror)
Southbridge	0x10000000–0x1001FFFF	APB	128KB
Northbridge	0x10020000–0x100DFFFF	AHB	768KB
Northbridge	0x100E0000–0x100FFFFF	APB	128KB
Northbridge	0x10100000–0x17FFFFFF	Reserved	127MB
Southbridge	0x18000000–0x1FFFFFFF	AHB	128MB
Northbridge	0x20000000–0x3FFFFFFF	Reserved	512MB
Northbridge	0x40000000–0x5FFFFFFF	SMC	512MB
Northbridge	0x60000000–0x6FFFFFFF	PCI	256MB
Northbridge	0x70000000–0x8FFFFFFF	DDR	512MB
Northbridge	0x90000000–0xBFFFFFFF	PCI	768MB
Logic Tile site	0xC0000000–0xFFFFFFFF	External	1GB

Note

The 16MB memory region 0x00000000–0x00FFFFFF can be remapped to:

- NOR flash (SMC CS0)
- PISMO expansion memory (SMC CS4)

The other memory regions have fixed decoding and are handled either internally or externally.

The locations for memory, peripherals, and controllers for the Northbridge and the Southbridge are listed in Table 4-2. A pictorial overview of the memory map is also shown in *System memory map for standard peripherals* on page 4-8.

Table 4-2 Memory map for standard peripherals

Peripheral	Address range	Bus type	Region size
Dynamic memory mirror (0x70000000-0x7FFFFFFF) During boot remapping however, the bottom 16MB of this memory region (0x00000000-0x00FFFFFF) can be: • NOR flash: 0x40000000-0x40FFFFFF • static expansion memory (PISMO): 0x50000000-0x50FFFFFF	0x00000000-0x0FFFFFFF	AXI	256MB
System registers	0x10000000-0x10000FFF	APB	4KB
System controller 0	0x10001000-0x10001FFF	APB	4KB
2-Wire Serial Bus Control	0x10002000-0x10002FFF	APB	4KB
Reserved	0x10003000-0x10003FFF	APB	4KB
Advanced Audio CODEC	0x10004000-0x10004FFF	APB	4KB
MultiMedia Card Interface	0x10005000-0x10005FFF	APB	4KB
Keyboard/Mouse Interface 0	0x10006000-0x10006FFF	APB	4KB
Keyboard/Mouse Interface 1	0x10007000-0x10007FFF	APB	4KB
Reserved for future use	0x10008000-0x10008FFF	APB	4KB
UART 0 Interface	0x10009000-0x10009FFF	APB	4KB
UART 1 Interface	0x1000A000-0x1000AFFF	APB	4KB

Table 4-2 Memory map for standard peripherals (continued)

Peripheral	Address range	Bus type	Region size
UART 2 Interface	0x1000B000-0x1000BFFF	APB	4KB
UART 3 Interface	0x1000C000-0x1000CFFF	APB	4KB
Synchronous Serial Port Interface	0x1000D000-0x1000DFFF	APB	4KB
Smart Card Interface	0x1000E000-0x1000EFFF	APB	4KB
Watchdog 0 Interface	0x1000F000-0x1000FFFF	APB	4KB
Watchdog 1 Interface	0x10010000-0x10010FFF	APB	4KB
Timer modules 0 and 1 Interface (Timer 1 starts at 0x10011020)	0x10011000-0x10011FFF	APB	4KB
Timer modules 2 and 3 Interface (Timer 3 starts at 0x10012020)	0x10012000-0x10012FFF	APB	4KB
GPIO Interface 0	0x10013000-0x10013FFF	APB	4KB
GPIO Interface 1	0x10014000-0x10014FFF	APB	4KB
GPIO Interface 2 (miscellaneous onboard I/O)	0x10015000-0x10015FFF	APB	4KB
Serial Bus Control (DVI)	0x10016000-0x10016FFF	APB	4KB
Real Time Clock Interface	0x10017000-0x10017FFF	APB	4KB
Timer Modules 4 and 5 Interface (Timer 5 starts at 0x10018020)	0x10018000-0x10018FFF	APB	4KB
Timer Modules 6 and 7 Interface (Timer 7 starts at 0x10019020)	0x10019000-0x10019FFF	APB	4KB
System Controller 1	0x1001A000-0x1001AFFF	APB	4KB
Reserved for future use (4K x 5)	0x1001B000-0x1001FFFF	APB	20KB
Color LCD Controller configuration	0x10020000-0x1002FFFF	AHB	64KB
DMA Controller configuration	0x10030000-0x1003FFFF	AHB	64KB
Reserved (64K x 2)	0x10040000-0x1005FFFF	AHB	128KB
Internal Northbridge SRAM	0x10060000-0x1007FFFF	AXI	128KB
Reserved (64K x 6)	0x10080000-0x100DFFFF	AHB	384KB

Table 4-2 Memory map for standard peripherals (continued)

Peripheral	Address range	Bus type	Region size
Dynamic Memory Controller configuration	0x100E0000-0x100E0FFF	APB	4KB
Static Memory Controller configuration	0x100E1000-0x100E1FFF	APB	4KB
Reserved	0x100E2000-0x100E2FFF	APB	4KB
APB Registers (PLL configuration)	0x100E3000-0x100E3FFF	APB	4KB
Reserved for future use	0x100E4000-0x100EFFFF	APB	48KB
Reserved for future use (DAP ROM table)	0x100F0000-0x100FFFFF	APB	64KB
Reserved	0x10100000-0x103FFFFF	–	3MB
Reserved for future use	0x10400000-0x16FFFFFF	AHB or AXI	108MB
Reserved for future use	0x17000000-0x17FFFFFF	AXI	16MB
Compact Flash	0x18000000-0x18000FFF	AHB	4KB
Reserved for future use	0x18001000-0x1BFFFFFF	AHB	63.096MB
Reserved for future use	0x1C000000-0x1DFFFFFF	AHB	32MB
Generic Interrupt Controller 1 (GIC1) (**nIRQ** interrupt handling for ARM11 MPCore)	0x1E000000-0x1E00FFFF	AHB	64KB
Generic Interrupt Controller 2 (GIC2) (**nFIQ** interrupt handling for ARM11 MPCore)	0x1E010000-0x1E01FFFF	AHB	64KB
Generic Interrupt Controller 3 (GIC3) (**nIRQ** interrupt handling for Tile Site)	0x1E020000-0x1E02FFFF	AHB	64KB
Generic Interrupt Controller 4 (GIC4) (**nFIQ** interrupt handling for Tile Site)	0x1E030000-0x1E03FFFF	AHB	64KB
Reserved for future use	0x1E040000-0x1EFFFFFF	AHB	15.75MB
Reserved for future use (ARM11 MPCore test chip control 0x1F000000-0x1F004000)	0x1F000000-0x1FFFFFFF	AHB	16MB
Reserved	0x20000000-0x3FFFFFFF	AHB or AXI	512MB

Table 4-2 Memory map for standard peripherals (continued)

Peripheral	Address range	Bus type	Region size
SMC Chip Selects: • CS0 NOR flash 0x40000000–0x43FFFFFF • CS1 NOR flash 0x44000000–0x47FFFFFF • CS2 Cellular RAM 0x48000000–0x4BFFFFFF • CS3 configuration PLD Config flash 0x4C000000–0x4DFFFFFF Ethernet 0x4E000000–0x4EFFFFFF USB 0x4F000000–0x4FFFFFFF • CS4 PISMO (**nCS0**) 0x50000000–0x53FFFFFF • CS5 PISMO (**nCS1**) 0x54000000–0x57FFFFFF • CS6 PISMO (**nCS2**) 0x58000000–0x5BFFFFFF • CS7 PISMO (**nCS3**) 0x5C000000–0x5FFFFFFF	0x40000000-0x5FFFFFFF	SMC	512MB
PCI interface	0x60000000-0x6FFFFFFF	PCI	256MB
Dynamic memory (CS0)	0x70000000-0x7FFFFFFF	DDR	256MB
Dynamic memory (CS1)	0x80000000-0x8FFFFFFF	DDR	256MB
PCI interface	0x90000000-0xBFFFFFFF	PCI	768MB
Logic Tile site expansion. (If a Logic tile is not fitted, the baseboard aborts accesses to this memory region)	0xC0000000-0xFFFFFFFF	External	1GB

Figure 4-1 on page 4-8 shows an overview of the memory map.

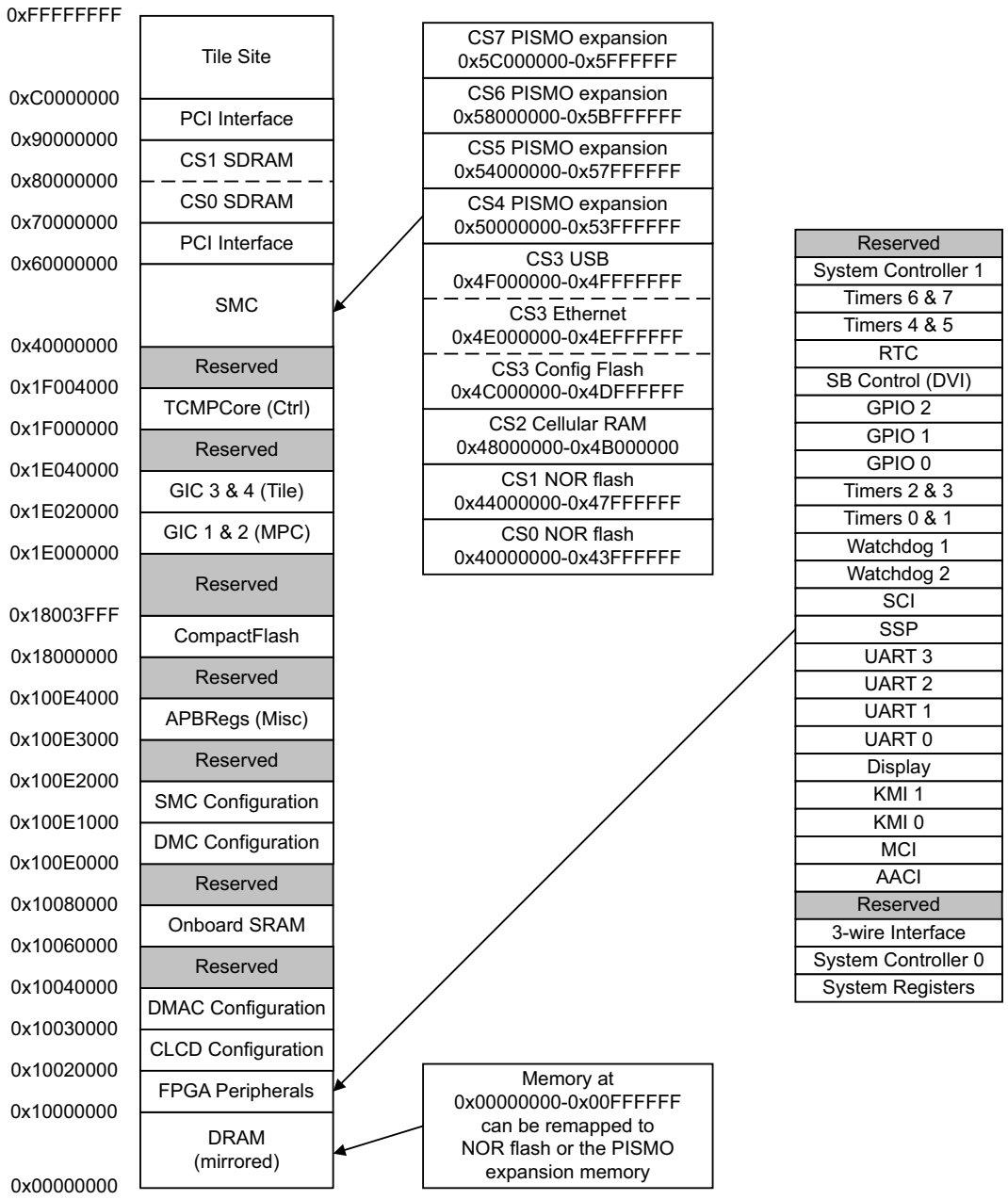


Figure 4-1 System memory map for standard peripherals

4.2 Configuration and initialization

This section describes how the baseboard and external memory and peripherals are configured and initialized at power on. See *Status and system control registers* on page 4-11 for details on configuring the PB11MPCore once the system is out of reset.

4.2.1 Remapping of boot memory

On reset, the ARM11 MPCore begins executing code at address `0x00000000`. During normal operation this address is mirrored volatile DRAM but at reset remapping enables mirrored non-volatile static memory to be accessed. The non-volatile memory to be remapped is determined by the configuration switch S7-1 as listed in Table 4-3.

Table 4-3 Boot memory

S7-1	Chip select	Memory Range	Comment
OFF	SMC CS0	`0x40000000`–`0x40FFFFFF`	Lower 16MB bank of NOR flash is remapped to `0x00000000-0x00FFFFFF`.
ON	SMC CS4	`0x50000000`–`0x50FFFFFF`	Lower 16MB bank of PISMO expansion memory is remapped to `0x00000000-0x00FFFFFF`.

Note

While NOR flash or PISMO expansion memory is remapped, any access to an address in the range `0x01000000` to `0x03FFFFFF` will be read back as zero.

4.2.2 Memory characteristics

Table 4-4 lists the DMC and SMC chip selects, and memory range. Addresses not listed are decoded by the Northbridge or the Southbridge for local peripheral selection or are passed to the Logic Tile site.

The static memory controller chip select CS3 is further decoded by the configuration PLD to select Config Flash, Ethernet or USB.

Table 4-4 Memory chip selects and address range

Chip Select	Address range	Device
DMC CS0	0x00000000-0x0FFFFFFF (0x70000000-0x7FFFFFFF)	DRAM (DRAM mirror)
DMC CS1	0x80000000-0x8FFFFFFF	DRAM
SMC CS0	0x40000000-0x43FFFFFF (0x00000000-0x00FFFFFF)	NOR flash (when remapped: 0x01000000-0x03FFFFFF will be read back as zero)
SMC CS1	0x44000000-0x47FFFFFF	NOR flash
SMC CS2	0x48000000-0x4BFFFFFF	Cellular RAM
SMC CS3 *Additional address decoding is handled by the CS PLD.*	0x4C000000-0x4DFFFFFF	Configuration flash
	0x4E000000-0x4EFFFFFF	Ethernet
	0x4F000000-0x4FFFFFFF	USB
SMC CS4	0x50000000-0x53FFFFFF (0x00000000-0x00FFFFFF)	PISMO expansion memory (**nCS0**) (when remapped: 0x01000000-0x03FFFFFF will be read back as zero)
SMC CS5	0x54000000-0x57FFFFFF	PISMO expansion memory (**nCS1**)
SMC CS6	0x58000000-0x5BFFFFFF	PISMO expansion memory (**nCS2**)
SMC CS7	0x5C000000-0x5FFFFFFF	PISMO expansion memory (**nCS3**)

4.3 Status and system control registers

The baseboard status and system control registers enable the ARM11 MPCore to determine its environment and to control the on-board systems. The register set is listed in Table 4-5.

Note

All registers are 32 bits wide and do not support byte writes. Write operations must be word-wide and bits marked as *reserved* must be preserved using read-modify-write.

The status and system control registers base address is `0x10000000`.

Table 4-5 Register map for status and system control registers

Register	Offset Value	Access[a]	Reset Value	Description
SYS_ID	`0x0000`	Read-only	`0xX159X5XX`	System Identifier. See *ID Register, SYS_ID* on page 4-15.
SYS_USERSW	`0x0004`	Read-only	`0x0000000XX`	Bits [7:0] map to S4 (user switches). See *User Switch Register, SYS_USERSW* on page 4-16.
SYS_LED	`0x0008`	Read/Write	`0x000000000`	Bits [7:0] map to user LEDs. See *LED Register, SYS_LED* on page 4-16.
SYS_OSC[0:4]	`0x000C`–`0x001C`	Read/Write Lockable	0: `0x00012C5C` 1: `0x00002CC0` 2: `0x00002C75` 3: `0x00020211` 4: `0x00002C75`	Settings for the ICS307 programmable oscillators: OSC0 - OSC4. See *Oscillator Registers, SYS_OSCx* on page 4-17.
SYS_LOCK	`0x0020`	Read/Write	`0x00010000`	Write `0xA05F` to unlock lockable registers. See *Lock Register, SYS_LOCK* on page 4-17.
SYS_100HZ	`0x0024`	Read-only	`0x00000000`	100Hz counter. See *100Hz Counter, SYS_100HZ* on page 4-19.
Reserved	`0x0028`-`0x002C`	–	–	This region is reserved for use by the Emulation Baseboard (EB).
SYS_FLAGS	`0x0030`	Read	`0x00000000`	General-purpose flags (reset by any reset). See *Flag Registers, SYS_FLAGSx and SYS_NVFLAGSx* on page 4-19.
SYS_FLAGSSET	`0x0030`	Write	`0x00000000`	Set bits in general-purpose flags.

Table 4-5 Register map for status and system control registers (continued)

Register	Offset Value	Access[a]	Reset Value	Description
SYS_FLAGSCLR	0x0034	Write-only	–	Clear bits in general-purpose flags.
SYS_NVFLAGS	0x0038	Read	0xXXXXXXXX	General-purpose nonvolatile flags (reset only on power up).
SYS_NVFLAGSSET	0x0038	Write	0x00000000	Set bits in general-purpose nonvolatile flags.
SYS_NVFLAGSCLR	0x003C	Write-only	–	Clear bits in general-purpose nonvolatile flags.
SYS_RESETCTL	0x0040	Read/Write Lockable	0x00000000	Controls the software reset level to be applied to system components. See *Reset Control Register, SYS_RESETCTL* on page 4-21.
Reserved	0x0044	–	–	This region is reserved for use by the PB11MPCore.
SYS_MCI	0x0048	Read-only	0x0000000x	MCI status and control register. See *MCI Register, SYS_MCI* on page 4-22.
SYS_FLASH	0x004C	Read/Write	0x00000000	Controls write protection of flash devices. See *Flash Control Register, SYS_FLASH* on page 4-23.
SYS_CLCD	0x0050	Read/Write	0x00001F00	Controls LCD power and multiplexing. See *CLCD Control Register, SYS_CLCD* on page 4-24.
Reserved	0x0054	Read/Write	–	This region is reserved for use by the Emulation Baseboard (EB).
SYS_CFGSW	0x0058	Read-only	0x000000XX	Read register returns the current switch settings of switch S7. See *Configuration select switch, SYS_CFGSW* on page 4-25.
SYS_24MHZ	0x005C	Read-only	0x00000000	32-bit counter clocked at 24MHz. See *24MHz Counter, SYS_24MHZ* on page 4-26.
SYS_MISC	0x0060	Read-only	0x00000000	Miscellaneous control flags. See *Miscellaneous flags, SYS_MISC* on page 4-26.

Table 4-5 Register map for status and system control registers (continued)

Register	Offset Value	Access[a]	Reset Value	Description
SYS_DMAPSR	0x0064	Read/Write	0x00000000	Selection control for remapping DMA from external peripherals to DMA. See *DMA peripheral map register, SYS_DMAPSR* on page 4-28.
SYS_PEX_STAT	0x0068	Read-only	0x0000XXXX	PCI Express status register. See *PCI Express status register, SYS_PEX_STAT* on page 4-29.
SYS_PCI_STAT	0x006C	Read/Write Lockable	0x00000337	PCI status register. See *PCI status register, SYS_PCI_STAT* on page 4-31.
Reserved	0x0070	–	–	This region is reserved for use by the PB11MPCore.
SYS_PLD_CTRL1	0x0074	Read/Write Lockable	0xXX0000XX	This register sets the Config PLD write data register fields that configure the ARM11 MPCore test chip. See *PLD control register 1, SYS_PLD_CTRL1* on page 4-32 for details.
SYS_PLD_CTRL2	0x0078	Read/Write Lockable	0x000FE00F	This register reads the Config PLD read data register fields that provide status information from the ARM11 MPCore test chip. See *PLD control register 2, SYS_PLD_CTRL2* on page 4-34
SYS_PLD_INIT	0x007C	Read/Write Lockable	0x11210002	This register defines the ARM11 MPCore *Test chip PLL control register* and *Test chip clock divider register* hardware initialization values. See *PLD initialization register, SYS_PLD_INIT* on page 4-35
Reserved	0x0080	–	–	This region is reserved for use by the PB11MPCore.
SYS_PROCID0	0x0084	Read-only	0x06000000	Read returns a description for the test chip present on the platform baseboard (PB). See *Processor ID register 0, SYS_PROCID0* on page 4-38.
SYS_PROCID1	0x0088	Read-only	0xFF000000	Indicates that there is no Core Tile fitted.

Table 4-5 Register map for status and system control registers (continued)

Register	Offset Value	Access[a]	Reset Value	Description
SYS_OSCRESET[0:4]	0x008C–0x009C	Read/Write	0x00000000	Value to load into the SYS_OSC[0:4] registers on a manual reset. At power-on reset, the SYS_OSCRESET[0:4] registers are loaded with the same default value as used for SYS_OSC[0:4].
SYS_VOLTAGE_CTL[0:7]	0x00A0–0x00BC	Read/Write Lockable	–	Monitoring and control of ARM11 MPCore voltages and currents.
SYS_TEST_OSC[0:4]	0x00C0–0x00D0	Read-only	–	32-bit counter clocked from ICS307 oscillators. See *Oscillator test registers, SYS_TEST_OSCx* on page 4-41.
SYS_OSC[5:6]	0x00D4–0x00D8	Read/Write Lockable	0x00000000	Settings for the ICS307 programmable oscillators: OSC5 - OSC6. See *Oscillator Registers, SYS_OSCx* on page 4-17.
SYS_OSCRESET[5:6]	0x00DC–0x00E0	Read/Write Lockable	0x00000000	Value to load into the SYS_OSC[5:6] registers on a manual reset. At power-on reset, the SYS_OSCRESET[5:6] registers are loaded with the same default value as used for SYS_OSC[5:6].
SYS_TEST_OSC[5:6]	0x00E4–0x00E8	Read-only	–	32-bit counter clocked from ICS307 oscillators. See *Oscillator test registers, SYS_TEST_OSCx* on page 4-41.

a. If Access is lockable, the register can only be written if SYS_LOCK is unlocked (see *Lock Register, SYS_LOCK* on page 4-17).

4.3.1 ID Register, SYS_ID

The SYS_ID register at `0x10000000` is a read-only register that identifies the board and FPGA.

Figure 4-2 shows the register bit assignment.

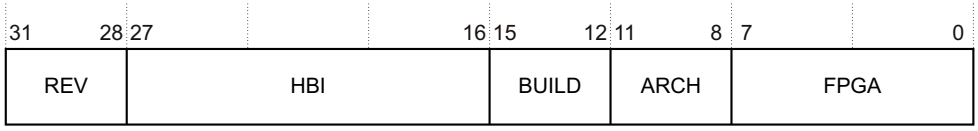


Figure 4-2 SYS_ID register

The function of the register bits are shown in Table 4-6. The register value depends on the image loaded into the FPGA.

Table 4-6 SYS_ID register bit assignments

Bits	Access	Name	Reset	Description
[31:28]	Read-only	REV	`0x0`	Board revision: `0x0` = Rev A `0x1` = Rev B `0x2` = Rev C
[27:16]	Read-only	HBI	`0x159`	HBI board number (`0x09F`)
[15:12]	Read-only	BUILD	`0x0`	Build variant of board (from BOM) `0xF`: all builds
[11:8]	Read-only	ARCH	`0x5`	Bus architecture `0x4` = AHB `0x5` = AXI
[7:0]	Read-only	FPGA	`0x00`	FPGA build

4.3.2 User Switch Register, SYS_USERSW

The SYS_USERSW register at `0x10000004` reads the front panel User Switches and general purpose (user) switch S4. A value of 1 indicates that the switch is on.

Figure 4-3 shows the register bit assignment.

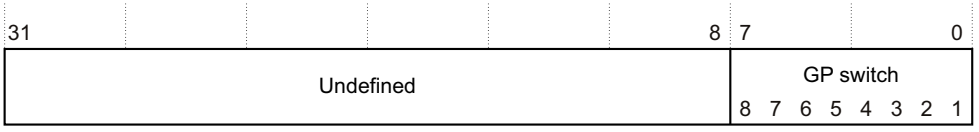


Figure 4-3 SYS_USERSW register

4.3.3 LED Register, SYS_LED

The SYS_LED register at `0x10000008` controls the user LEDs. Set the corresponding register bit to 1 to light the LED. At reset all LEDs are turned off.

Figure 4-4 shows the register bit assignment.

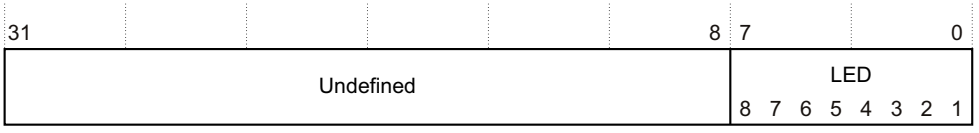


Figure 4-4 SYS_LED register

4.3.4 Oscillator Registers, SYS_OSCx

The oscillator registers, SYS_OSC0 through to SYS_OSC4, at `0x1000000C`–`0x1000001C` and SYS_OSC5 and SYS_OSC6, at `0x100000D4` and `0x100000D8` are read/write registers that control the frequency of the clocks generated by the ICS307 programmable oscillators OSC0 through to OSC6.

Figure 4-5 shows the registers bit assignment.

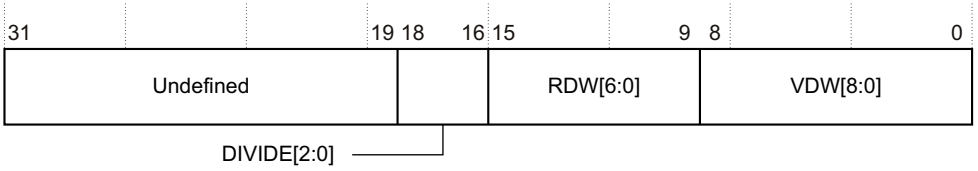


Figure 4-5 SYS_OSCx register

The function of the register bits are shown in Table 4-7. For more detail on bit values, see *ICS307 programmable clock generators* on page 3-39 and *Clock frequency restrictions* on page B-3.

Table 4-7 SYS_OSCx register bit assignments

Bits	Access	Name	Reset	Description
[31:20]	Write ignored, read as zero	–	`0x000`	Undefined
[19]	Write ignored Read as zero	–	`b0`	Undefined
[18:16]	Read/Write	DIVIDE[2:0]	`bxxx`	Output divider select
[15:9]	Read/Write	RDW[6:0]	`bxxxxxxx`	Reference divider value
[8:0]	Read/Write	VDW[8:0]	`bxxxxxxxxx`	VCO divider value

Note

Before writing to a SYS_OSCx register, you must unlock it by writing the value `0x0000A05F` to the SYS_LOCK register. After writing the SYS_OSC register, you should relock it by writing any value other than `0x0000A05F` to the SYS_LOCK register.

4.3.5 Lock Register, SYS_LOCK

The SYS_LOCK register at `0x10000020` locks or unlocks access to all lockable registers.

Lockable registers cannot be modified while they are locked. This mechanism prevents the registers from being overwritten accidently. The registers are locked by default after a reset.

Figure 4-6 shows the register bit assignment.

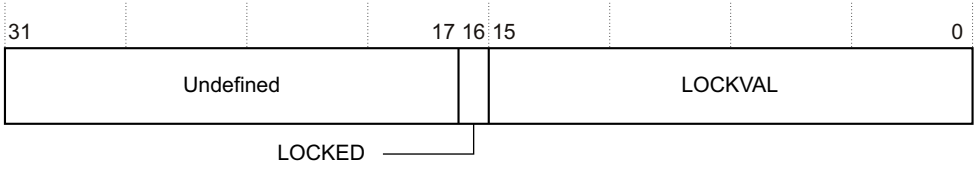


Figure 4-6 SYS_LOCK register

The function of the register bits are shown in Table 4-8.

Table 4-8 SYS_LOCK register bit assignments

Bits	Access	Name	Reset	Description
[31:20]	Write ignored, read as zero	–	`0x000`	Undefined
[19:17]	Write ignored, read as zero	–	`b000`	Undefined
[16]	Read-only	LOCKED	b1	This bit indicates if the lockable registers are locked or unlocked: `b0` = unlocked `b1` = locked.
[15:0]	Read/Write	LOCKVAL	`0x0000`	Write the value `0xA05F` to unlock the lockable registers. Write any other value to this register to lock them.

4.3.6 100Hz Counter, SYS_100HZ

The SYS_100HZ register at `0x10000024` is a 32-bit counter incremented at 100Hz. The 100Hz reference is derived from the on-board 32.768kHz crystal oscillator. The register is set to zero by a reset and when read returns the count since the last reset.

Figure 4-7 shows the register bit assignment.

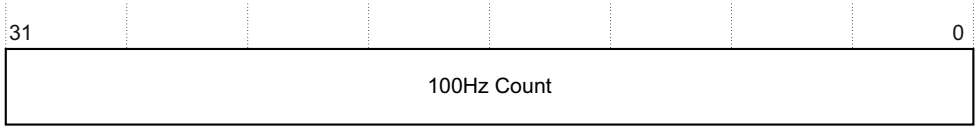


Figure 4-7 100Hz Counter, SYS_100HZ register

4.3.7 Flag Registers, SYS_FLAGSx and SYS_NVFLAGSx

The registers shown in Table 4-9 provide two 32-bit register locations containing general-purpose flags. You can assign any meaning to the flags.

Table 4-9 Flag registers

Register	Address	Access	Reset by	Description
SYS_FLAGS	`0x10000030`	Read	Reset	Flag register
SYS_FLAGSSET	`0x10000030`	Write	Reset	Flag Set register
SYS_FLAGSCLR	`0x10000034`	Write	Reset	Flag Clear register
SYS_NVFLAGS	`0x10000038`	Read	POR	Nonvolatile Flag register
SYS_NVFLAGSSET	`0x10000038`	Write	POR	Nonvolatile Flag Set register
SYS_NVFLAGSCLR	`0x1000003C`	Write	POR	Nonvolatile Flag Clear register

The board provides two distinct types of flag register:

- The SYS_FLAGS Register is cleared by a normal reset, such as a reset caused by pressing the reset button.
- The SYS_NVFLAGS Register retains its contents after a normal reset and is only cleared by a *Power-On Reset* (POR).

Flag and Nonvolatile Flag Registers

The SYS_FLAGS and SYS_NVFLAGS registers contain the current state of the flags.

Flag and Nonvolatile Flag Set Registers

The SYS_FLAGSSET and SYS_NVFLAGSSET registers are used to set bits in the SYS_FLAGS and SYS_NVFLAGS registers:

- write 1 to SET the associated flag
- write 0 to leave the associated flag unchanged.

Flag and Nonvolatile Flag Clear Registers

Use the SYS_FLAGSCLR and SYS_NVFLAGSCLR registers to clear bits in the SYS_FLAGS and SYS_NVFLAGS registers:

- write 1 to CLEAR the associated flag
- write 0 to leave the associated flag unchanged.

4.3.8 Reset Control Register, SYS_RESETCTL

The SYS_RESETCTL register at `0x10000040` allows a software reset to be applied to selected system components.

Figure 4-8 shows the register bit assignment.

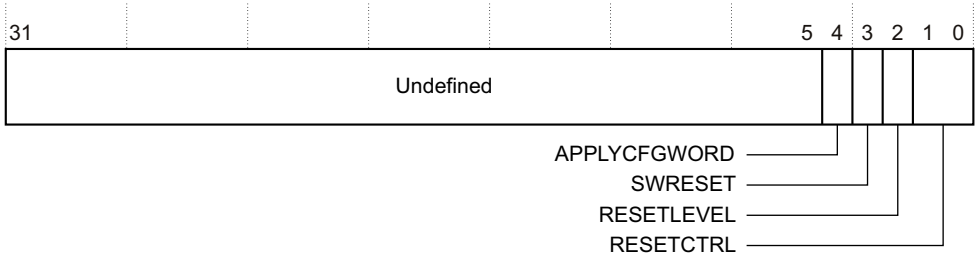


Figure 4-8 SYS_RESETCTL register

The function of the register bits are shown in Table 4-10.

Table 4-10 SYS_RESETCTL register bit assignments

Bits	Access	Name	Reset	Description
[31:5]	Write ignored, read as zero	–	`0x0000000`	Undefined
[4]	Write only	APPLYCFGWORD	`b0`	Updates OSCCLK registers and applies a PORESET.
[3]	Write only	SWRESET	`b0`	Software equivalent of PBRESET button.
[2]	Write only	RESETLEVEL	`b0`	Forces a software reset at the depth set by RESETCTRL.
[1:0]	Read/Write	RESETCTRL	`b00`	Sets the depth of a software reset `b00` = PORESET `b01` = ISPCONFIG `b10` = MEMRESET `b11` = SYSRST

See *Reset Controller state diagram* on page 3-46 for details of the reset entry points.

4.3.9 MCI Register, SYS_MCI

The SYS_MCI register at `0x10000048` provides status information on the Multimedia card socket.

Figure 4-9 shows the register bit assignment.

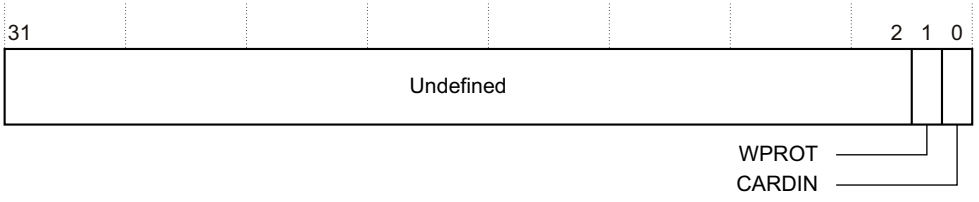


Figure 4-9 SYS_MCI register

The function of the register bits are shown in Table 4-11.

Table 4-11 SYS_MCI register bit assignment

Bits	Access	Name	Reset	Description
[31:4]	Write ignored, read as zero	–	`0x0000000`	Undefined
[3:2]	Write ignored, read as zero	–	`b00`	Undefined
[1]	Read-only	WPROT	`b0`	Status of the Write Protect bit (WPROT)
[0]	Read-only	CARDIN	`b0`	Card Detect: `b0` = no card detected `b1` = card detected

4.3.10 Flash Control Register, SYS_FLASH

The SYS_FLASH register at `0x1000004C` controls write protection of static memory devices.

Figure 4-10 shows the register bit assignment.

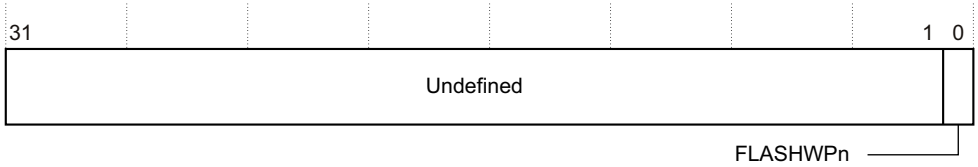


Figure 4-10 SYS_FLASH register

The function of the register bits are listed in Table 4-12.

Table 4-12 SYS_FLASH register bit assignments

Bits	Access	Name	Reset	Description
[31:4]	Write ignored, read as zero	–	`0x0000000`	Undefined
[3:1]	Write ignored, read as zero	–	`b000`	Undefined
[0]	Read/Write	FLASHWPn	`b0`	Controls writing to Flash (power-on reset state is `b0`) `b0` = writing to Flash is disabled `b1` = writing to Flash is enabled

4.3.11 CLCD Control Register, SYS_CLCD

The SYS_CLCD register at `0x10000050` returns CLCD adaptor board status and enables external serial communication via the baseboard *Synchronous Serial Port* SSP expansion connector J27.

Note

The CLCD adaptor board status signals are provided for legacy compatibility only.

Figure 4-11 shows the register bit assignment.

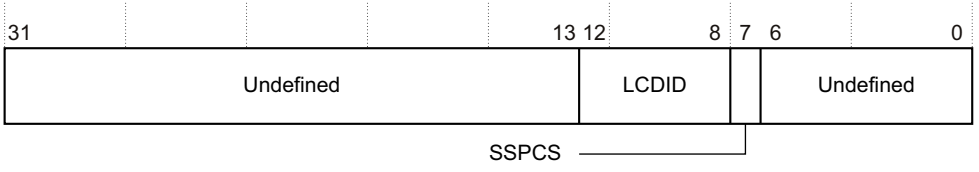


Figure 4-11 SYS_CLCD register

The function of the register bits are listed in Table 4-13.

Table 4-13 SYS_CLCD register bit register assignments

Bits	Access	Name	Reset	Description
[31:16]	Write ignored, read as zero	–	`0x0000`	Undefined
[15:13]	Write ignored, read as zero	–	`b000`	Undefined
[12:8]	Read-only	LCDID	`b11111`	Returns the setting of the ID links on the CLCD adaptor board **Note** Retained for legacy compatibility only.

Table 4-13 SYS_CLCD register bit register assignments (continued)

Bits	Access	Name	Reset	Description
[7]	Read/Write	SSPCS	b0	Enables external serial communication via the baseboard *Synchronous Serial Port* SSP expansion connector J27. b1 = **SSPnCS** at J27 is active b0 = **SSPnCS** at J27 is not active See *Synchronous Serial Port, SSP* on page 3-27.
[6:4]	Write ignored, read as zero	–	b000	Undefined
[3:0]	Write ignored, read as zero	–	0x0	Undefined

4.3.12 Configuration select switch, SYS_CFGSW

This SYS_CFGSW register at 0x10000058 is a read-only register that reads the configuration select switch bank S7 settings. A value of 1 indicates that a switch is on.

Figure 4-12 shows the register bit assignment.

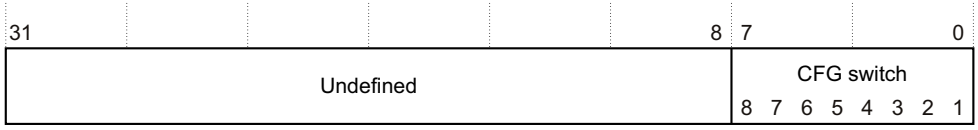


Figure 4-12 SYS_CFGSW register

4.3.13 24MHz Counter, SYS_24MHZ

The SYS_24MHZ register at `0x1000005C` is a read-only register that provides a 32-bit count value. The count increments at 24MHz frequency from the 24MHz crystal reference output **REFCLK24MHZ** from OSC0. The register is set to zero by a reset.

Figure 4-13 shows the register bit assignment.

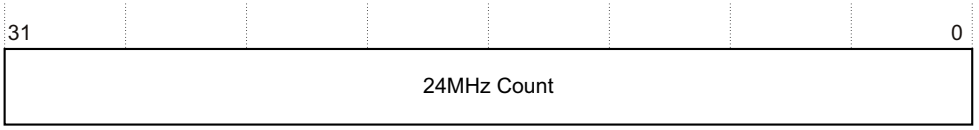


Figure 4-13 SYS_24MHZ register

4.3.14 Miscellaneous flags, SYS_MISC

The SYS_MISC register at `0x10000060` returns the value of the tile site tile detect signal and other miscellaneous flags related to communication. See Table 4-14 on page 4-27 for details.

Figure 4-14 shows the register bit assignment.

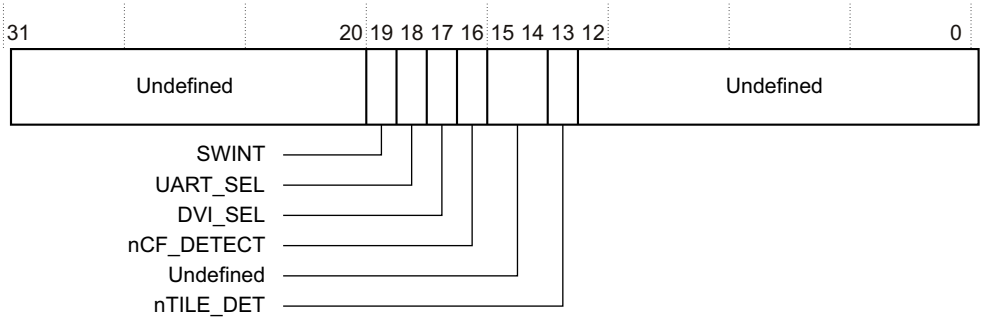


Figure 4-14 SYS_MISC register

The function of the register bits are listed in Table 4-14 on page 4-27.

Table 4-14 SYS_MISC register bit assignment

Bits	Access	Name	Reset	Description
[31:20]	Write ignored, read as zero	–	0x0000	Undefined
[19]	Read/Write	SW_INT	b0	Software interrupt
[18]	Read/Write	UART_SEL	b0	Selects the source for the UART 2 and UART 3 connectors: b0: baseboard interface b1: tile site interface
[17]	Read/Write	DVI_SEL	b0	Selects the source for the DVI connector: b0: baseboard interface b1: tile site interface
[16]	Read-only	nCF_DETECT	b1	b0: CompactFlash card detected b1: No CompactFlash card detected
[15:14]	Write ignored, read as zero	–	b000	Undefined
[13]	Read-only	nTILE_DET	b0	Tile Detect b0 = tile present b1 = tile not present
[12]	Write ignored, read as zero	–	b0	Undefined
[11:0]	Write ignored, read as zero	–	0x000	Undefined

4.3.15 DMA peripheral map register, SYS_DMAPSR

The DMA peripheral map register, SYS_DMAPSR at `0x10000064` permits the mapping of DMA channels to external interfaces. The register is set to zero by a reset. The DMA mapping is disabled by default.

Figure 4-15 shows the register bit assignment.

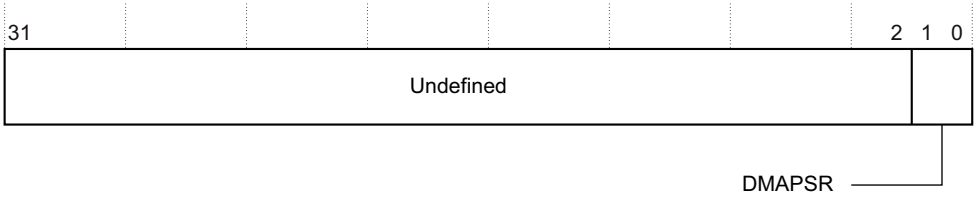


Figure 4-15 SYS_DMAPSR register

The function of the register bits are listed in Table 4-15.

Table 4-15 SYS_DMAPSR register bit assignments

Bit	Access	Name	Reset	Description
[31:4]	Write ignored, read as zero	–	`0x0000000`	Undefined
[3:2]	Write ignored, read as zero	–	`b00`	Undefined
[1:0]	Read/Write	DMAPSR	`b00`	Selects the peripheral group to be mapped to DMA. See Table 4-16 on page 4-29 for the bit encoding.

Table 4-16 lists the bit encoding. See *Single Master Direct Memory Access Controller, SMDMAC* on page 4-50 for more information on the DMA logic.

Table 4-16 SYS_DMAPSR register bit coding

DMAPSR = `b00`	**DMAPSR** = `b01`	**DMAPSR** = `b1X`	**DMA Request and Response**
Peripheral	**Peripheral**	**Peripheral**	**DMACSREQ DMACBREQ DMACCLR**
reserved	reserved	reserved	[15:8]
SCI TX	UART0 TX	reserved	[7]
SCI RX	UART0 RX	reserved	[6]
AACI RX	UART1 TX	reserved	[5]
AACI TX	UART1 RX	reserved	[4]
MCI	UART2 TX	reserved	[3]
T1DMAC[0]	UART2 RX	reserved	[2]
USB[1]	SSP TX	reserved	[1]
USB[0]	SSP RX	reserved	[0]

4.3.16 PCI Express status register, SYS_PEX_STAT

The PCI express status register, SYS_PEX_STAT at `0x10000068` monitors the lane status of the PCI-X to PCI Express bridge (PEX8114) and PCI Express switch (PEX 8518) components.

Figure 4-16 shows the register bit assignment.

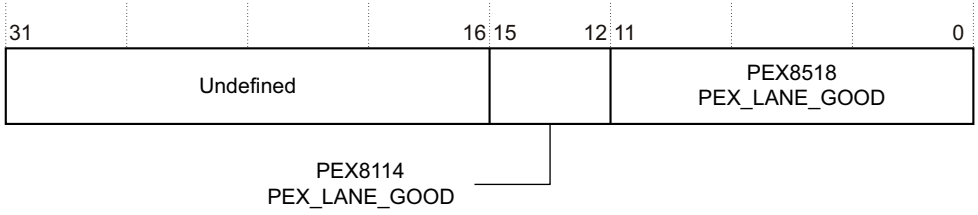


Figure 4-16 SYS_PEX_STAT register

The function of the register bits are listed in Table 4-17.

Table 4-17 SYS_PEX_STAT register bit assignments

Bits	Access	Name	Reset	Description
[31:16]	Write ignored, read as zero	–	0x0000	Undefined
[15:12]	Read-only	PEX8114 PEX_LANE_GOOD	0xX	PCI-X to PCI Express bridge lane status
[11:0]	Read-only	PEX8518 PEX_LANE_GOOD	0xXXX	PCI Express switch lane status

Note

The **PEX_LANE_GOOD** signals indicate the presence and link-up state of each PCI Express lane. These determine which lanes are active, and provide the status of the final negotiated link width as follows:

- if the **PEX_LANE_GOODx** signal is continuously active, the link is *trained* to its programmed width
- if the **PEX_LANE_GOODx** signal alternates to and from the active state, the link is *trained* with fewer lanes than the programmed width.

Please refer to the PEX 8114 and PEX 8518 technical documentation at the PLX Technology Inc website: www.plxtech.com for further details.

4.3.17 PCI status register, SYS_PCI_STAT

The PCI status register, SYS_PCI_STAT at `0x1000006C` monitors the status of the PCI-X bridge (PCI6520) component.

Caution

ARM do not recommend changing the register default values as this may effect the reliability of PCI interface.

Figure 4-17 shows the register bit assignment.

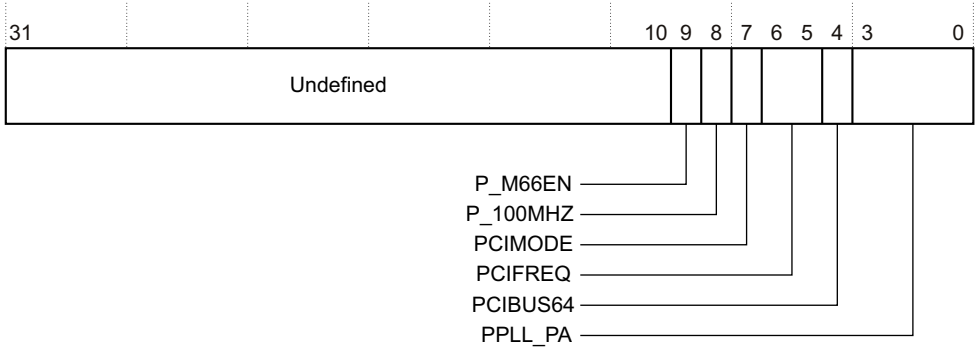


Figure 4-17 SYS_PCI_STAT register

The function of the register bits are listed in Table 4-18.

Table 4-18 SYS_PCI_STAT register bit assignments

Bits	Access	Name	Reset	Description
[31:12]	Write ignored, read as zero	–	`0x00000`	Undefined
[11:10]	Write ignored, read as zero	–	`b00`	Undefined
[9]	Read/Write	P_M66EN	`b1`	66MHz PCI-X enable
[8]	Read/Write	P_100MHZ	`b1`	Clock control

Table 4-18 SYS_PCI_STAT register bit assignments (continued)

Bits	Access	Name	Reset	Description
[7]	Read/Write	PCIMODE	b0	PCI mode: b0 = PCI-X b1 = PCI
[6:5]	Read/Write	PCIFREQ	b01	PCI frequency: b00 = 33MHz b01 = 66MHz b10 = 100MHz b11 = 133MHz
[4]	Read/Write	PCIBUS64	b1	64-bit PCI bus enable
[3:0]	Read/Write	PPLL_PA	0x7	PLL range: 0x1 = PPLL_RANGE300_600MHz 0x3 = PPLL_RANGE150_300MHz 0x4 = PPLL_RANGE100_200MHz 0x5 = PPLL_RANGE75_150MHz 0x7 = PPLL_RANGE50_100MHz

4.3.18 PLD control register 1, SYS_PLD_CTRL1

The SYS_PLD_CTRL1 register at 0x10000074 sets the Config PLD write data register fields that configure the ARM11 MPCore test chip.

Figure 4-18 shows the register bit assignment.

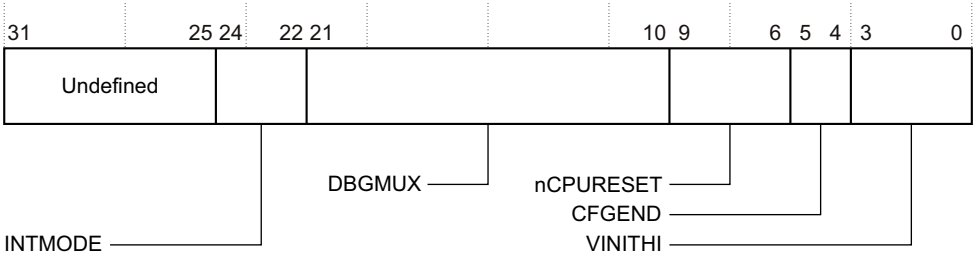


Figure 4-18 SYS_PLD_CTRL1 register

The function of the register bits are listed in Table 4-19 on page 4-33.

Table 4-19 SYS_PLD_CTRL1 register bit assignments

Bits	Access	Name	Reset	Description
[31:28]	Write ignored, read as zero	–	0x0	Undefined
[27:25]	Write ignored, read as zero	–	b000	Undefined
[24:22]	Read/Write Lockable	INTMODE	b000	Interrupt mode: • bx00: Legacy mode • bx01: New mode with DCC • bx1x: New mode with no DCC • b1xx: FIQ[3:0] enable (independent of INTMODE[1:0]) See *Generic Interrupt Controller, GIC* on page 4-55 for details.
[21:10]	Read/Write Lockable	DBGMUX	0x000	Sets the user defined debug cross-trigger mode. The required ARM11 MPCore **EDBGRQ[3:0]** and **DBGACK[3:0]** signals are routed using a debug matrix. **Note** The debug matrix is not implemented in all releases of this product, please see the accompanying *Release Notes* for implementation details.
[9:6]	Read/Write Lockable	nCPURESET	b1111	Individual MPCore CPU resets. nCPURESET[x] resets CPU[x} See *Resets* on page 5-11 for details.
[5:4]	Read/Write Lockable	CFGEND	b00	The U and EE bits reset values in the CP15 Control Register and E Bit reset value in CPSR/SPSR depend on the value of CFGEND[1:0]. See the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360).
[3:0]	Read/Write Lockable	VINITHI	b0000	When HIGH, indicates high-Vecs mode for the respective MPCore CPU. See *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360).

4.3.19 PLD control register 2, SYS_PLD_CTRL2

The SYS_PLD_CTRL2 register at `0x10000078` reads the PLD read data register fields that provide status on the ARM11 MPCore test chip and the PLD.

Figure 4-19 shows the register bit assignment.

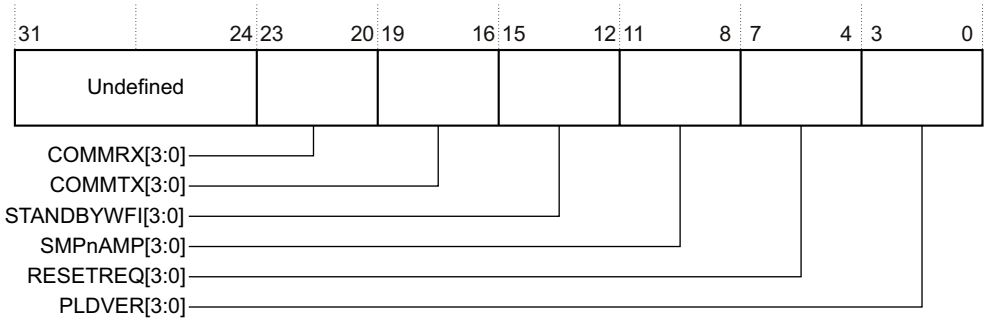


Figure 4-19 SYS_PLD_CTRL2 register

The function of the register bits are listed in Table 4-20.

Table 4-20 SYS_PLD_CTRL2 register bit assignments

Bits	Access	Name	Reset	Description
[31:24]	Read as zero, write ignored.	–	`0x00`	Undefined
[23:20]	Read-only	COMMRX	bxxxx	Comms channels receive
[19:16]	Read-only	COMMTX	bxxxx	Comms channels transmit
[15:12]	Read-only	STANDBYWFI	bxxxx	Individual WFI indicators. Indicates if an ARM11 MPCore CPU is in WFI state. See the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360).

Table 4-20 SYS_PLD_CTRL2 register bit assignments (continued)

Bits	Access	Name	Reset	Description
[11:8]	Read -only	SMPnAMP	bxxxx	Individual AMP or SMP mode indicators, one from each ARM11 MPCore CPU. • 0: Indicates AMP mode (processor is not part of coherency) • 1: Indicates SMP mode (processor is part of coherency) See the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360).
[7:4]	Read-only	RESETREQ	bxxxx	Individual watchdog reset requests, one from each ARM11 MPCore CPU
[3:0]	Read-only	PLDVER	bxxxx	PLD build version

4.3.20 PLD initialization register, SYS_PLD_INIT

The SYS_PLD_INIT register at `0x1000007C` sets the ARM11 MPCore test chip *Test chip PLL control register* and *Test chip clock divider register* hardware initialization values. You cannot access these registers directly.

Figure 4-20 shows the register bit assignment.

Note

The values written are transferred to the config PLD via a serial stream during a hard reset (**nSYSPOR**) and during a soft reset (**nSYSRST**).

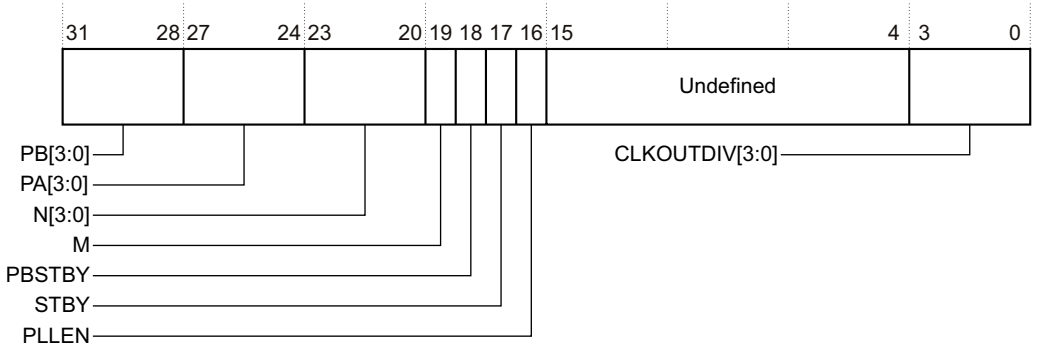


Figure 4-20 SYS_PLD_INIT register

The function of the register bits are listed in Table 4-21.

Table 4-21 SYS_PLD_INIT register bit assignments

Bits	Access	Name	Reset	Description
[31:28]	Read/Write Lockable	PB divider	b0010	PB divider ratio that the PLL uses: 0000: pb-value is 1 0001: pb-value is 2 … 1111: pb-value is 16.
[27:24]	Read/Write Lockable	PA divider	b0010	PA divider ratio that the PLL uses: 0000: pa-value is 1 0001: pa-value is 2 … 1111: pa-value is 16.
[23:20]	Read/Write Lockable	N divider	b0011	N divider ratio that the PLL uses: 0000: n-value is 1 0001: n-value is 2 … 1111: n-value is 16.
[19]	Read/Write Lockable	M divider	b0	M divider ratio that the PLL uses: • 0: m-value is 1 • 1: m-value is 2.
[18]	Read/Write Lockable	PBSTBY	b0	• 0: **CLKB** from PLL is active • 1: **CLKB** from PLL is 0.
[17]	Read/Write Lockable	STBY	b0	• 0: **CLKA** from PLL is active • 1: **CLKA** from PLL is 0.

Table 4-21 SYS_PLD_INIT register bit assignments (continued)

Bits	Access	Name	Reset	Description
[16]	Read/Write Lockable	PLLEN	b1	• 0: if *committed* and bit 0 of the PLL control test data register is 0 (its reset value) **REFCLK** will be selected as **CLKOUT** on the next rising edge of **REFCLK** • 1: if *committed*, and bit 0 of the PLL control test data register is 0 (its reset value) the PLL will be selected as **CLKOUT** on the next rising edge of **REFCLK**. PLL enable is *committed* when **STANDBYWFI[3:0]** from the mpcore_tc_module == b1111 or the input signal **CONFIGINIT** == b1.
[15:4]	Read as zero, write ignored.	-	0x000	Undefined
[3:0]	Read/Write Lockable	CLKOUTDIV	b0011	The clock divider ratio to drive the AXI bus clock and the FPGA: 0000: **CLKOUTDIV** is equal to **CLKIN** 0001: **CLKOUTDIV** is **CLKIN** divided by 2 0010: **CLKOUTDIV** is **CLKIN** divided by 3 0011: **CLKOUTDIV** is **CLKIN** divided by 4 0100: **CLKOUTDIV** is **CLKIN** divided by 5 0101: **CLKOUTDIV** is **CLKIN** divided by 6 0110: **CLKOUTDIV** is **CLKIN** divided by 7 0111: **CLKOUTDIV** is **CLKIN** divided by 8 1000: **CLKOUTDIV** is **CLKIN** divided by 1 1001: **CLKOUTDIV** is **CLKIN** divided by 10 1010: **CLKOUTDIV** is **CLKIN** divided by 3 1011: **CLKOUTDIV** is **CLKIN** divided by 12 1100: **CLKOUTDIV** is **CLKIN** divided by 5 1101: **CLKOUTDIV** is **CLKIN** divided by 14 1110: **CLKOUTDIV** is **CLKIN** divided by 7 1111: **CLKOUTDIV** is **CLKIN** divided by 16

4.3.21 Processor ID register 0, SYS_PROCID0

This register at `0x10000084` indicates the test chip fitted to the baseboard.

Figure 4-21 shows the register bit assignment.

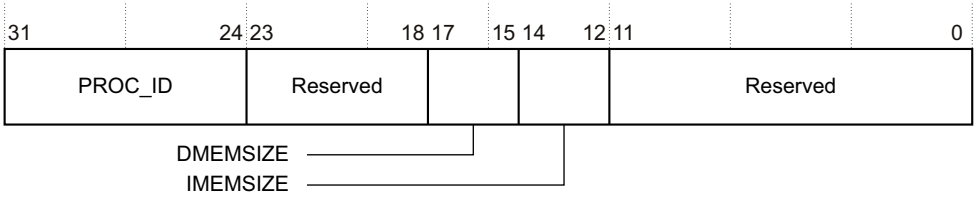


Figure 4-21 SYS_PROCID0 register

The function of the register bits are listed in Table 4-22.

Table 4-22 SYS_PROCID0 register bit assignments

Bits	Access	Name	Reset	Description
[31:24]	Read-only	PROC_ID	`0x06`	Returns the test chip processor type: `0x00` = ARM7TDMI `0x02` = ARM9xx `0x04` = ARM1136 `0x06` = ARM11 MPCore `0x08` = ARM1156 `0x0A` = ARM1176 `0x0C` = A9 `0x0E` = A8 `0x10` = R4 `0xFF` = not fitted
[23:20]	Read-only	–	`0x0`	Reserved
[19:18]	Read-only	–	`b00`	Reserved
[17:15]	Read-only	DMEMSIZE	`b100`	Returns the size of the data cache memory in the processor, default is 32KB for ARM11 MPCore.
[14:12]	Read-only	IMEMSIZE	`b100`	Returns the size of the instruction cache memory in the processor, default is 32KB for ARM11 MPCore
[11:0]	Read-only	–	`0x000`	Reserved

4.3.22 Processor ID register 1, SYS_PROCID1

This register at 0x10000088 identifies the tile fitted to the baseboard.

Figure 4-22 shows the register bit assignment.

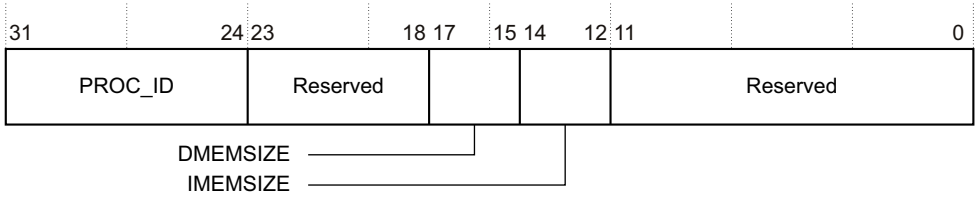


Figure 4-22 SYS_PROCID1 register

The function of the register bits are listed in Table 4-23

Table 4-23 SYS_PROCID1 register bit assignments

Bits	Access	Name	Reset	Description
[31:24]	Read-only	PROC_ID	0xFF	Returns processor type on the tile site: 0x00 = ARM7TDMI 0x02 = ARM9xx 0x04 = ARM1136 0x06 = ARM11 MPCore 0x08 = ARM1156 0x0A = ARM1176 0x0C = A9 0x0E = A8 0x10 = R4 0xFF = not fitted (default for a Logic Tile)
[23:20]	Read-only	–	0x0	Reserved
[19:18]	Read-only	–	b00	Reserved
[17:15]	Read-only	DMEMSIZE	b000	Returns the size of the data cache memory for a processor on the tile site. Default = b000 for a Logic Tile.
[14:12]	Read-only	IMEMSIZE	b000	Returns the size of the instruction cache memory for a processor on the tile site, Default = b000 for a Logic Tile.
[11:0]	Read-only	–	0x000	Reserved

4.3.23 Oscillator reset registers, SYS_OSCRESETx

The oscillator reset registers, SYS_OSCRESET0 through to SYS_OSCRESET4, at `0x1000008C`–`0x1000009C`, and SYS_OSCRESET5 and SYS_OSCRESET6, at `0x1000000DC` and `0x1000000E0` are read/write registers that control the frequency of the clocks generated by clock generators OSC0 through to OSC6 when a manual reset is generated.

Figure 4-23 shows the registers bit assignment.

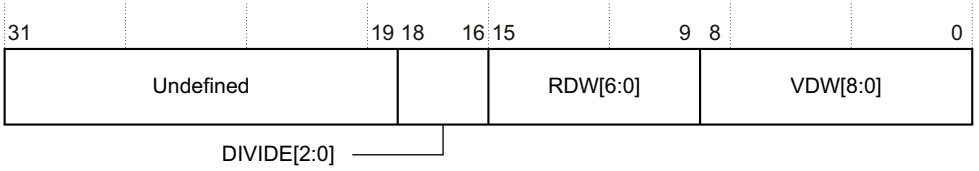


Figure 4-23 SYS_OSCRESETx register

Note

Before writing to a SYS_OSCRESETx register you must unlock it by writing the value `0x0000A05F` to the SYS_LOCK register (see *Lock Register, SYS_LOCK* on page 4-17). After writing the SYS_OSCRESETx register, you should relock it by writing any value other than `0x0000A05F` to the SYS_LOCK register.

For more detail on bit values, see *ICS307 programmable clock generators* on page 3-39 and *Oscillator Registers, SYS_OSCx* on page 4-17.

Note

At power-on reset (**nSYSPOR**), the SYS_OSCRESETx registers are loaded with the same default values used for the SYS_OSCx registers.

The values of the SYS_OSCRESETx values can be changed after powering on the baseboard. Pushing the reset push button loads the values of the SYS_OSCRESETx registers into the SYS_OSCx registers and loads the programmable oscillators with the new values.

4.3.24 Voltage control registers, SYS_VOLTAGE_CTLx

These registers are used to:

- set the voltages applied to the test chip
- read the voltages applied to the test chip
- read the current drawn by the test chip.

Note

The currents drawn by the test chip are measured by a voltage drop across a current sense resistor (VDDCORE_DIFFx).

The function of the registers bits are listed in Table 4-24.

Table 4-24 SYS_VOLTAGE_CTLx registers

Register	Address	Bits [31:20] (Read-only)	Bits [19:8] (Read-only)	Bits [7:0] (Read/Write)
SYS_VOLTAGE_CTL0	0x100000A0	Reserved, read as zero	Read VDDCORE	Set VDDCORE
SYS_VOLTAGE_CTL1	0x100000A4	Reserved, read as zero	Read AVDD	Set AVDD
SYS_VOLTAGE_CTL2	0x100000A8	Reserved, read as zero	Read VDDCORE current	Reserved
SYS_VOLTAGE_CTL3	0x100000AC	Reserved, read as zero	Read AVDD current	Reserved
SYS_VOLTAGE_CTL4	0x100000B0	Reserved, read as zero	Read 1V8_ADC	Reserved
SYS_VOLTAGE_CTL5	0x100000B4	Reserved, read as zero	Read 1V2_ADC	Reserved
SYS_VOLTAGE_CTL6	0x100000B8	Reserved, read as zero	Reserved	Reserved
SYS_VOLTAGE_CTL7	0x100000BC	Reserved, read as zero	Reserved	Reserved

The default value loaded into the **SYS_VOLTAGE_CTL0** register at power-on is 0xAF. The default value loaded into the **SYS_VOLTAGE_CTL1** register at power-on is 0x80. See *Setting the ARM11 MPCore test chip voltages* on page 3-37 for more details on changing the test chip voltages. See also *Power measurement and TCK* on page 3-56.

4.3.25 Oscillator test registers, SYS_TEST_OSCx

The oscillator test registers, SYS_TEST_OSC0 through to SYS_TEST_OSC4, at 0x100000C0–0x100000D0, and SYS_TEST_OSC5 and SYS_TEST_OSC6, at 0x100000E4 and 0x100000E8, provide 32-bit count values. The count increments at the frequency of the corresponding ICS307 programmable oscillator (OSC0 through to OSC6). The registers are set to zero by a reset.

Table 4-25 lists the SYS_TEST_OSCx registers address locations.

Table 4-25 SYS_TEST_OSCx registers

Register	Address	Access	Description
SYS_TEST_OSC0	0x100000C0	Read-only	Counter clocked from OSC0
SYS_TEST_OSC1	0x100000C4	Read-only	Counter clocked from OSC1
SYS_TEST_OSC2	0x100000C8	Read-only	Counter clocked from OSC2
SYS_TEST_OSC3	0x100000CC	Read-only	Counter clocked from OSC3
SYS_TEST_OSC4	0x100000D0	Read-only	Counter clocked from OSC4
SYS_TEST_OSC5	0x100000E4	Read-only	Counter clocked from OSC5
SYS_TEST_OSC6	0x100000E8	Read-only	Counter clocked from OSC6

4.4 System Controller (SYSCTRL)

The SP810 System Controller (SYSCTRL) is an AMBA compliant SoC peripheral that is developed and tested by ARM Limited.

Table 4-26 SYSCTRL implementation

Property	Value
Location	Southbridge
Memory base address	• SYSCTRL 0: `0x10001000` • SYSCTRL 1: `0x1001A000`
Interrupt	–
DMA	–
Release version	ARM SYSCTRL SP810 r0p0
Reference documentation	*PrimeXsys System Controller (SP810) Technical Reference Manual* DDI 0254B See also *System Controller* on page 3-26.

SYSCTRL 0 Controls the remap signal, watchdog 0 clock enable, and timer enables for timers 0,1,2 and 3.

SYSCTRL 1 Controls watchdog 1 clock enable, and timer enables for timers 4,5,6 and 7.

4.4.1 PrimeCell modifications

The PrimeXsys System Controller (SP810) used in the Southbridge implements only the SYS_CTRL and peripheral ID registers. These registers support the following functionality:

- Identification of the build version of the System Controller
- Watchdog and timer module clock enable generation
- memory remap control.

The function of the bits in the SYS_CTRL0 register at address `0x10001000` is listed in *SYS_CTRL0 register* on page 4-44.

Note

TIMCLK is 1MHz. **REFCLK** is 32.768kHz.

Table 4-27 SYS_CTRL0 register

Bits	Function
[31:24]	Reserved. Use read-modify-write to preserve value.
[23]	Watchdog0 enable override. If 0, the enable output is derived from the **REFCLK** source. If 1, the enable output is forced HIGH.
[22]	Timer 3 enable override. If 0, the enable output is derived from the Timing Reference Select signal. If 1, the enable output is forced HIGH.
[21]	Timer 3 enable/ Timer Reference Select. If 0, the timing reference is **REFCLK**. If 1, the timing reference is **TIMCLK**.
[20]	Timer 2 enable override. If 0, the enable output is derived from the Timing Reference Select signal. If 1, the enable output is forced HIGH.
[19]	Timer 2 enable/ Timer Reference Select. If 0, the timing reference is **REFCLK**. If 1, the timing reference is **TIMCLK**.
[18]	Timer 1 enable override. If 0, the enable output is derived from the Timing Reference Select signal. If 1, the enable output is forced HIGH.
[17]	Timer 1 enable/ Timer Reference Select. If 0, the timing reference is **REFCLK**. If 1, the timing reference is **TIMCLK**.
[16]	Timer 0 enable override. If 0, the enable output is derived from the Timing Reference Select signal. If 1, the enable output is forced HIGH.
[15]	Timer 0 enable/ Timer Reference Select. If 0, the timing reference is **REFCLK**. If 1, the timing reference is **TIMCLK**.
[14:10]	Reserved. Use read-modify-write to preserve value.
[9]	Remap status. This read-only bit returns the remap status.
[8]	Remap clear request. Set this bit to disable memory remapping and return to normal mapping with dynamic memory selected for memory accesses to the region `0x00000000-0x00FFFFFF`.
[7:0]	Reserved. Use read-modify-write to preserve value.

The function of the bits in the SYS_CTRL1 register at address `0x1001A000` is listed in Table 4-28

Table 4-28 SYS_CTRL1 register

Bits	Function
[31:24]	Reserved. Use read-modify-write to preserve value.
[23]	Watchdog1 enable override. If 0, the enable output is derived from the **REFCLK** source. If 1, the enable output is forced HIGH.
[22]	Timer7 enable override. If 0, the enable output is derived from the Timing Reference Select signal. If 1, the enable output is forced HIGH.
[21]	Timer7 enable/ Timer Reference Select. If 0, the timing reference is **REFCLK**. If 1, the timing reference is **TIMCLK**.
[20]	Timer6 enable override. If 0, the enable output is derived from the Timing Reference Select signal. If 1, the enable output is forced HIGH.
[19]	Timer6 enable/ Timer Reference Select. If 0, the timing reference is **REFCLK**. If 1, the timing reference is **TIMCLK**.
[18]	Timer5 enable override. If 0, the enable output is derived from the Timing Reference Select signal. If 1, the enable output is forced HIGH.
[17]	Timer5 enable/ Timer Reference Select. If 0, the timing reference is **REFCLK**. If 1, the timing reference is **TIMCLK**.
[16]	Timer4 enable override. If 0, the enable output is derived from the Timing Reference Select signal. If 1, the enable output is forced HIGH.
[15]	Timer4 enable/ Timer Reference Select. If 0, the timing reference is **REFCLK**. If 1, the timing reference is **TIMCLK**.
[14:10]	Reserved. Use read-modify-write to preserve value.
[9]	Reserved. Use read-modify-write to preserve value.
[8]	Reserved. Use read-modify-write to preserve value.
[7:0]	Reserved. Use read-modify-write to preserve value.

4.5 Advanced Audio CODEC Interface, AACI

The PL041 PrimeCell *Advanced Audio CODEC Interface* (AACI) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-29 AACI implementation

Property	Value
Location	Southbridge (the CODEC is an external component, LM4549)
Memory base address	`0x10004000`
Interrupt	51
DMA	• AACI TX: channel 4 • AACI RX: channel 5 **Note** You must set **DMAPSR** = `b00` in the SYS_DMAPSR register to select this peripheral for DMA access.
Release version	ARM AACI PL041 r1p0 (modified to one channel and 256 FIFO depth in compact mode and 512 FIFO depth in non-compact mode)
Platform Library support	No support provided.
Reference documentation	*ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual* DDI 0173 and *National Semiconductor LM4549 Data Sheet*. See also *Modified AACI PeriphID3 register* on page 4-47.

4.5.1 PrimeCell Modifications

The AACI PrimeCell in the Southbridge has a different FIFO depth than the standard PL041. Therefore, the AACIPeriphID3 register contains the values listed in Table 4-30.

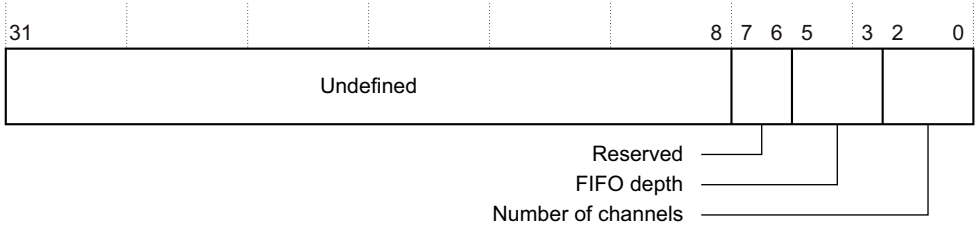


Figure 4-24 AACI ID register

Table 4-30 Modified AACI PeriphID3 register

Bit	Access	Description
[31:8]	Write as zeros, read is undefined	Undefined
[7:6]	Read-modify-write to preserve value	Reserved
[5:3]	Read-only	FIFO depth in compact mode b000 8 b001 16 b010 32 b011 64 b100 128 b101 256 b110 512 (default) b111 1024
[2:0]	Read-only	number of channels b000 4 b001 1 (default) b010 2 b011 3 b100 4 b101 5 b110 6 b111 7

4.6 Color LCD Controller, CLCDC

The PL111 PrimeCell *Color LCD Controller* (CLCDC) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-31 CLCDC implementation

Property	Value
Location	Northbridge
Memory base address	`0x10020000` **Note** There is also a LCD system control register at `0x10000050`. See *CLCD Control Register, SYS_CLCD* on page 4-24.
Interrupt	55
DMA	–
Release version	ARM CLCDC PL111 (version r0p0)
Reference documentation	*ARM PrimeCell Color LCD Controller (PL111) Technical Reference Manual* DDI 0293.

The following locations are reserved, and must not be used during normal operation:

- locations at offsets `0x030` to `0x1FE` are reserved for possible future extensions
- locations at offsets `0x400` to `0x7FF` are reserved for test purposes.

Note

Different display resolutions require different data and synchronization timing. **OSCCLK4** (25MHz default) is assigned as **CLCDCLK** for the LCD controller. Default display resolution is 1024x768 at 60Hz frame rate.
See *PrimeCell Color LCD Controller (PL111) Technical Reference Manual* (DDI 0293) for details of the LCD timing registers.

4.6.1 Display resolutions and display memory organization

Use registers CLCD_TIM0, CLCD_TIM1, CLCD_TIM2, and SYS_OSC4 to define the display timings. Table 4-32 lists the register and clock values for different display resolutions.

Table 4-32 Values for different display resolutions

Display resolution	CLCDCLK frequency and SYS_OSC4 register value	CLCD_TIM0 register at 0x10020000	CLCD_TIM1 register at 0x10020004	CLCD_TIM2 register at 0x10020008
QVGA(240x320) (portrait) on VGA	25MHz, 0x2C77	0xC7A7BF38	0x595B613F	0x04eF1800
QVGA (320x240) (landscape) on VGA	25MHz, 0x2C77	0x9F7FBF4C	0x818360eF	0x053F1800
QCIF (176x220) (portrait) on VGA	25MHz, 0x2C77	0xe7C7BF28	0x8B8D60DB	0x04AF1800
VGA (640x480) on VGA	25MHz, 0x2C77	0x3F1F3F9C	0x090B61DF	0x067F1800
SVGA (800x600) on SVGA	36MHz, 0x2CAC	0x1313A4C4	0x0505F657	0x071F1800
XVGA (1024x768) on XVGA	64MHz, 0x35CF6	0x972F67FC	0x17030EFF	0x07FF3800
Epson 2.2in panel QCIF (176x220)	16MHz, 0x2C48	0x02010228	0x010004DB	0x04AF3800
Sanyo 3.8in panel QVGA (320x240)	10MHz, 0x2C2A	0x0505054C	0x050514eF	0x053F1800

The mapping of the 32 bits of pixel data in memory to the RGB display signals depends on the resolution and display mode.

For details on setting the red, green, and blue brightness for direct (non-palettized) 24-bit and 16-bit color modes see the *CLCD Technical Reference Manual*. Selftest example code, that displays 24-bit and 16-bit VGA images, is provided on the accompanying CD.

Note

For resolutions based on one to sixteen bits per pixel, multiple pixels are encoded into each 32-bit word.

All monochrome modes, and color modes using 8 or fewer bits per pixel, use the palette to encode the color value from the data bits, see the *CLCD Technical Reference Manual* for details.

4.7 Single Master Direct Memory Access Controller, SMDMAC

The PL081 PrimeCell *Single Master DMA Controller* (SMDMAC) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-33 SMDMAC implementation

Property	Value
Location	Northbridge
Memory base address	`0x10030000` for SMDMAC configuration `0x10000064` for SMDMAC mapping register SYS_DMAPSR
Interrupt	56
DMA	–
Release version	ARM DMAC PL081 r1p2
Reference documentation	*ARM PrimeCell Single Master DMA Controller (PL081) Technical Reference Manual*. See also *Single master direct memory access controller, SMDMAC* on page 3-18.

Eight peripheral DMA interfaces are provided in two selectable DMA mappings, a third mapping option has been reserved for future use. Only DMAC flow control is supported.

4.7.1 DMAC flow control

In DMAC flow control mode, the SMDMAC is programmed with the amount of data that is to be transferred and requires only request signals from the peripheral to show that its buffer is ready for access. Each channel requires three signals, **DMASREQ**, **DMABREQ**, and **DMACLR**, see *ARM PrimeCell Single Master DMA Controller (PL081) Technical Reference Manual* (DDI 0218) for details.

4.7.2 DMA channel allocation

The DMA channel allocation is selected by the SYS_DMAPSR register in the System Register block, see *DMA peripheral map register, SYS_DMAPSR* on page 4-28 for details of the DMAPSR field bit encoding.

4.8 Dynamic Memory Controller, DMC

The PL340 PrimeCell *Dynamic Memory Controller* (DMC) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-34 DMC implementation

Property	Value
Location	Northbridge
Memory base address	`0x100E0000`
Interrupt	–
DMA	–
Release version	ARM DMC PL340 r0p0
Reference documentation	*ARM PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual* See also *Remapping of boot memory* on page 4-9.

The DMC controls the 512MB of DDR 100MHz dynamic memory that is available on the baseboard. Sample programs that configure and use dynamic memory can be found on the CD that accompanies the baseboard.

4.8.1 Register values

For detailed register descriptions, see the *ARM PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual*.

Caution

Refer to the application note for your baseboard and Logic Tile combination for values for your system.

The application note package contains a Boot Monitor and platform library that initializes the DDR registers after power up (the `platform.a` library contains memory setup routines), see *Using the baseboard Boot Monitor and platform library* on page E-4 for details on using these utilities.

4.9 Ethernet

The Ethernet interface is implemented in an external SMCS LAN9118 10/100 Ethernet single-chip MAC and PHY. The internal registers of the LAN9118 are memory-mapped onto the static memory bus and occupy locations starting at base address `0x4E000000`.

Table 4-35 Ethernet implementation

Property	Value
Location	Board (LAN9118 chip)
Memory base address	`0x4E000000` (mapped onto the SMC bus)
Interrupt	60
DMA	None, use memory to memory DMA to access the FIFO buffers in the LAN9118 Host Bus Interface.
Release version	Custom interface to external controller.
Reference documentation	*LAN9118 Data Sheet* (see also *Ethernet interface* on page 3-30).

Refer to the LAN9118 application note or to the self test program supplied on the CD for additional information.

When manufactured, an ARM value for the Ethernet MAC address and the register base address are loaded into the EEPROM. The register base address is 0. The MAC address is unique, but can be reprogrammed if required.

4.10 General Purpose Input/Output, GPIO

The PL061 PrimeCell *General Purpose Input/Output* (GPIO) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited. Use the GPIO to generate or detect low frequency signals (less than 1MHz).

Table 4-36 GPIO implementation

Property	Value
Location	Southbridge
Memory base address	• GPIO 0: `0x10013000` • GPIO 1: `0x10014000` • GPIO 2: `0x10015000`
Interrupt	• GPIO 0: 38 • GPIO 1: 39 • GPIO 2: 40
DMA	NA
Release version	ARM GPIO PL061 r1p0
Reference documentation	*ARM PrimeCell General Purpose Input/Output (PL061) Technical Reference Manual* (see also *General Purpose Input/Output, GPIO* on page 3-25)

4.10.1 Onboard I/O control

GPIO2 is dedicated to the USB, push button, and MCI status signals as listed in Table 4-37.

Table 4-37 GPIO2 and MCI status signals

GPIO2 bit	Description
[7:5]	Reserved
[4]	USB Host Controller suspend/wakeup (pin 119 of ISP1761 USB Controller)
[3]	USB Device Controller suspend/wakeup (pin 120 of ISP1761 USB Controller)

Table 4-37 GPIO2 and MCI status signals (continued)

GPIO2 bit	Description
[2]	General-Purpose push button
[1]	MCI Write-protect status
[0]	MCI Card-present status

4.11 Generic Interrupt Controller, GIC

A custom Generic Interrupt Controller (GIC) is implemented in the Southbridge.

Table 4-38 Generic Interrupt Controller implementation

Property	Value
Location	Southbridge
Memory base address	`0x1E000000` GIC0 generates ARM11 MPCore **nIRQ** `0x1E010000` GIC1 generates ARM11 MPCore **nFIQ** `0x1E020000` GIC2 generates tile site **nIRQ** `0x1E030000` GIC3 generates tile site **nFIQ**
Interrupt	nFIQ and nIRQ signals are output to the ARM11 MPCore and the tile site in response to an interrupt from a peripheral.
DMA	–
Release version	ARM Custom IP
Reference documentation	*MPCore Distributed Interrupt Controller* in the *MPCore Multiprocessor Technical Reference Manual* (DDI 0360).

Four GICs are implemented in the Southbridge:

GIC0 generates ARM11 MPCore **nIRQ**

GIC1 generates ARM11 MPCore **nFIQ**

GIC2 generates tile site **nIRQ**

GIC3 generates tile site **nFIQ**

The GICs accept interrupts from peripherals in the Northbridge, Southbridge, on-board peripherals, and the tile site. The GICs generate **nFIQ** and **nIRQ** signals to the ARM11 MPCore and the tile site. See *Interrupts* on page 5-14 for details.

4.11.1 Interrupt signals

The device interrupts are listed in Table 4-39.

Note

Refer to the application note for your product for details on how interrupts are handled for your system and the interrupts signals present on the connectors.

Table 4-39 Interrupt signals to controllers

Bit	Interrupt source	Description
[95:90]	reserved	–
[89]	P_nINT[7]	PCI Express-to-PCI-X bridge (PEX8114) P_nINTD
[88]	P_nINT[6]	PCI Express-to-PCI-X bridge (PEX8114) P_nINTC
[87]	P_nINT[5]	PCI Express-to-PCI-X bridge (PEX8114) P_nINTB
[86]	P_nINT[4]	PCI Express-to-PCI-X bridge (PEX8114) P_nINTA
[85]	P_nINT[3]	PCI SLOT A P_nINTA or SLOT B P_nINTB
[84]	P_nINT[2]	PCI SLOT A P_nINT|D or SLOT B P_nINTA
[83]	P_nINT[1]	PCI SLOT A P_nINTC or SLOT B P_nINTD
[82]	P_nINT[0]	PCI SLOT A P_nINTB or SLOT B P_nINTC
[81]	P_NMI	Southbridge PCI bridge NMI
[80]	PCI_INTR	Southbridge PCI bridge INT
[79]	Reserved	–
[78]	Reserved	–
[77]	Reserved	–
[76]	Reserved	–
[75]	Reserved	–
[74]	Timer 6,7	Timers 6 and 7
[73]	Timer 4,5	Timers 4 and 5
[72]	Watchdog1	Watchdog 1 alarm

Table 4-39 Interrupt signals to controllers (continued)

Bit	Interrupt source	Description
[71]	T1_INT7	Interrupts from tile site
[70]	T1_INT6	
[69]	T1_INT5	
[68]	T1_INT4	
[67]	T1_INT3	
[66]	T1_INT2	
[65]	T1_INT1	
[64]	T1_INT0	
[63]	Reserved	–
[62]	Reserved	–
[61]	USB	Interrupt from USB controller IC
[60]	Ethernet	Interrupt from Ethernet controller IC
[59]	Reserved	–
[58]	PISMO	PISMO interrupt from memory expansion board
[57]	Reserved	–
[56]	DMAC	DMA controller
[55]	CLCD	CLCD display
[54]	Reserved	Reserved for Character LCD display (not currently supported)
[53]	KMI1	Keyboard/Mouse Interface
[52]	KMI0	Keyboard/Mouse Interface
[51]	AACI	CODEC controller interrupt
[50]	MCIb	Multimedia Card Interface interrupt b
[49]	MCIa	Multimedia Card Interface interrupt a
[48]	SCI	Smart Card interface
[47]	UART3	UART3

Table 4-39 Interrupt signals to controllers (continued)

Bit	Interrupt source	Description
[46]	UART2	UART2
[45]	UART1	UART1
[44]	UART0	UART0
[43]	SSP	Synchronous serial port
[42]	RTC	Real time clock
[41]	reserved	–
[40]	GPIO2	GPIO controller
[39]	GPIO1	GPIO controller
[38]	GPIO0	GPIO controller
[37]	Timer 2,3	Timers 2 and 3
[36]	Timer 0,1	Timers 0 and 1
[35]	reserved	Reserved for COMMTX
[34]	reserved	Reserved for COMMRX
[33]	Software interrupt	Software interrupt.
[32]	Watchdog0	Watchdog timer 0
[31:16]	GIC	reserved for GIC software interrupts
[15:0]	GIC	Software interrupts

4.11.2 Handling interrupts

This section describes interrupt handling and clearing in general.

For examples of interrupt detection and handling, see the platform library code supplied on the CD.

All interrupts are routed to all four GICs.

The sequence to determine and clear an interrupt is:

1. If required, stack the workspace. If interrupt pre-emption is used, also stack R14 and SPSR.
2. Determine interrupt ID by reading the Interrupt Ack Register of the interface.
3. If interrupt pre-emption is required, re-enable interrupts by setting CPSR bit 7.

   Warning

   A reentrant interrupt handler must save the IRQ state, switch processor modes, and save the state for the new processor mode before branching to a nested subroutine or C function.
   See *6.7.3 Reentrant interrupt handlers* in the *RealView Compilation Tools Developer Guide* for details.

4. Jump to the interrupt service routine. For hardware-triggered interrupts, the service routine must clear the interrupt in the peripheral by setting the appropriate bit in the peripheral interrupt-control register.
5. Write the interrupt number to the End of Interrupt Register.
6. Restore the workspace.
7. Return from the interrupt.

Note

The peripheral might contain its own interrupt mask and clear registers that must be configured before an interrupt is enabled.

4.12 Keyboard and Mouse Interface, KMI

The PL050 PrimeCell PS2 *Keyboard/Mouse Interface* (KMI) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited. Two KMIs are present on the baseboard:

KMI0 is used for keyboard input

KMI1 is used for mouse input.

Table 4-40 KMI implementation

Property	Value
Location	Southbridge
Memory base address	`0x10006000` KMI 0 (keyboard) `0x10007000` KMI 1 (mouse)
Interrupt	52 KMI 0 53 KMI 1
DMA	–
Release version	ARM KMI PL050 r1p0
Reference documentation	*ARM PrimeCell PS2 Keyboard/Mouse Interface (PL050) Technical Reference Manual*

4.13 MultiMedia Card Interface, MCI

The PL180 PrimeCell *MultiMedia Card Interface* (MCI) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited. The interface supports both Multimedia Cards and Secure Digital cards.

Table 4-41 MCI implementation

Property	Value
Location	Southbridge
Memory base address	`0x10005000`
Interrupt	49 for MCI A 50 for MCI B
DMA	3
Release version	ARM MCI PL180 r1p0
Reference documentation	*ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual*

The interrupts for the MCI card are managed by the GPIO 2 PrimeCell. See *General Purpose Input/Output, GPIO* on page 4-53.

4.14 AXI to PCI bridge

The AXI to PCI bridge is implemented in the Northbridge.

Table 4-42 AXI to PCI bridge implementation

Property	Value
Location	Northbridge
Memory base address	`0x90000000` `0x60000000` (reserved for PCI expansion)
Interrupt	82 P_nINT[0] 83 P_nINT[1] 84 P_nINT[2] 85 P_nINT[3]
DMA	None. Memory to memory transfers can be set up in the DMAC.
Release version	NEC (AXI2PCI)
Reference documentation	–

The windows that provide access to the PCI expansion bus are listed in Table 4-43.

Table 4-43 PCI bus memory map

Usage	Address
AXI2PCI	`0x90040000`
PCI I/O window	`0x90050000` to `0x9005FFFF`
PCI Memory window	`0xA0000000` to `0xBFFFFFFF`

The AXI to PCI bridge enables you to use the PB11MPCore with third-party PCI or PCI-Express expansion cards. PB11MPCore functions as a PCI host, that is, it generates clocks to the PCI or PCI-Express card.

The Northbridge AXI to PCI bridge recognizes accesses to addresses `0x90000000` to `0xBFFFFFFF` within the memory map as being intended for a target within PCI address space. There is also an additional region from `0x60000000` to `0x6FFFFFFF` that is reserved for PCI expansion if required.

Note

Only one PCI bus may use the I/O window at a time, as it is 4KB aligned.

PCI bridge initialization and configuration routines are included as part of the selftest suite on the Versatile Family CD.

4.15 Real Time Clock, RTC

The PL031 PrimeCell *Real Time Clock Controller* (RTC) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

A counter in the RTC is incremented every second. The RTC can therefore be used as a basic alarm function or long time-base counter.

The current value of the clock can be read at any time or the RTC can be programmed to generate an interrupt after counting for a programmed number of seconds. The interrupt can be masked by writing to the interrupt match set or clear register.

Table 4-44 RTC implementation

Property	Value
Location	Southbridge
Memory base address	`0x10017000`
Interrupt	42
DMA	–
Release version	ARM RTC PL031 r1p0
Reference documentation	*ARM PrimeCell Real Time Clock (PL031) Technical Reference Manual*

Note

There is also a separate Time-of-Year (TOY) implemented in an external RTC chip (DS1338) on the baseboard. The external RTC can be accessed by the serial bus interface (see *Two-wire serial bus interface, SBCon* on page 4-65). For details on the programming interface to the Time-of-Year RTC, see the data sheet for the Maxim DS1338 integrated circuit (`www.maxim-ic.com`).

4.16 Two-wire serial bus interface, SBCon

Two custom two-wire serial bus interfaces (SBCon 0 and SBCon 1) are implemented in the Southbridge.

SBCon 0 provides access to:

- the serial EEPROM on PISMO static memory expansion boards
- the Maxim DS1338 RTC on the baseboard.

SBCon 1 provides access to:

- the *Digital Data Channel* (DDC) of the external display connected to the DVI connector on the rear panel.

Table 4-45 Serial bus implementation

Property	Value
Location	Southbridge
Memory base address	SBCon 0: `0x10002000` SBCon 1: `0x10016000`
Interrupt	–
DMA	–
Release version	Custom logic
Reference documentation	• *Two-wire serial bus interface* on page 3-28 • Appendix C *Memory Expansion Boards* • *PISMO Specification Version 1.0* • data sheet for the Maxim DS1338 RTC (`www.maxim-ic.com`) • *VESA DDC Specification Version 3.0*

The registered device addresses are listed in Table 4-46

Table 4-46 Serial interface device addresses

Device	Write address	Read address	Description
PISMO (static memory module)	0xA2	0xA3	Identifies the type of memory on the board and how it is configured.
TOY (DS1338 RTC)	0xD0	0xD1	Reads time data and writes control data to the RTC.
DVI (external display)	display dependant	display dependant	Reads external display capabilities at the DVI connector. Can control display settings of *E-DDC* displays.

The registers listed in Table 4-47 and Table 4-48 on page 4-67 control the serial bus interfaces.

Table 4-47 SBCon 0 serial bus register

Address	Name	Access	Description
0x10002000	SB_CONTROL	Read	Read serial control bits: Bit [0] is **SCL** Bit [1] is **SDA**
0x10002000	SB_CONTROLS	Write	Set serial control bits: Bit [0] is **SCL** Bit [1] is **SDA**
0x10002004	SB_CONTROLC	Write	Clear serial control bits: Bit [0] is **SCL** Bit [1] is **SDA**

Table 4-48 SBCon 1 serial bus register

Address	Name	Access	Description
0x10016000	SB_CONTROL	Read	Read serial control bits: Bit [0] is **SCL** Bit [1] is **SDA**
0x10016000	SB_CONTROLS	Write	Set serial control bits: Bit [0] is **SCL** Bit [1] is **SDA**
0x10016004	SB_CONTROLC	Write	Clear serial control bits: Bit [0] is **SCL** Bit [1] is **SDA**

Note

SDA is an open-collector signal that is used for sending and receiving data. Set the output (sending) value HIGH before reading the current value.

Software must manipulate the **SCL** and **SDA** bits directly to access the data in the three devices. See the `\firmware\examples` directory on the CD for example code for reading the EEPROM that is on the memory expansion board.

Table 4-49 Serial bus device addresses

Device	Write address	Read address
Static expansion EEPROM (PISMO)	0xA2	0xA3
Time-of-year clock (TOY)	0xD0	0xD1
External display	display dependant	display dependant

4.17 Smart Card Interface, SCI

The PL131 PrimeCell *Smart Card Interface* (SCI) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-50 SCI implementation

Property	Value
Location	Southbridge
Memory base address	`0x1000E0000`
Interrupt	48
DMA	7 SCI transmit 6 SCI receive **Note** You must set **DMAPSR** = `b00` in the SYS_DMAPSR register to select this peripheral for DMA access.
Release version	ARM SCI PL131 r1p0
Platform Library support	No support provided
Reference documentation	*ARM SCI PrimeCell (PL131) Technical Reference Manual* ARM DDI 0228

See the self-test software that is supplied on the CD accompanying the PB11MPCore for an example of detecting a SIM card response to a reset.

4.18 Synchronous Serial Port, SSP

The PL022 PrimeCell *Synchronous Serial Port* (SSP) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-51 SSP implementation

Property	Value
Location	Southbridge
Memory base address	`0x1000D000`
Interrupt	43
DMA	1 SSP transmit 0 SSP receive **Note** You must set **DMAPSR** = `b01` in the SYS_DMAPSR register to select this peripheral for DMA access.
Release version	ARM SSP PL022 r1p0
Platform Library support	No support provided
Reference documentation	*ARM PrimeCell Synchronous Serial Port Controller (PL022) Technical Reference Manual* ARM DDI 0194

4.19 Static Memory Controller, SMC

The PrimeCell *Static Memory Controller* (SMC) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-52 SMC implementation

Property	Value
Location	Northbridge
Memory base address	`0x100E1000`
Interrupt	–
DMA	–
Release version	ARM SMC PL354 r0p0
Reference documentation	*ARM PrimeCell Static Memory Controller (PL350 series) Technical Reference Manual*, *Configuration and initialization* on page 4-9

4.20 Timers

The SP804 Dual-Timer module is an AMBA compliant SoC peripheral that is developed and tested by ARM Limited.

The module is an AMBA slave module and connects to the *Advanced Peripheral Bus* (APB). The Dual-Timer module consists of two programmable 32/16-bit down counters that can generate interrupts on reaching zero.

Table 4-53 Timer implementation

Property	Value
Location	Southbridge
Memory base address	• Timer 0-1: `0x10011000` • Timer 2-3: `0x10012000` • Timer 4-5: `0x10018000` • Timer 6-7: `0x10019000`
Interrupt	• Timer 0-1: 36 • Timer 2-3: 37 • Timer 4-5: 73 • Timer 6-7: 74
DMA	–
Release version	ARM Dual-Timer SP804 r1p2
Platform Library support	**timer_enable** Enables a timer with a given period and mode **timer_disable** Disables the defined timer **timer_interrupt_clear** Clears the timer interrupt
Reference documentation	*ARM Timer Module (SP804) Technical Reference Manual* ARM DDI 0271

At reset, the timers are clocked by a 32.768kHz reference from an external oscillator module. Use the System Controller to change the timer reference from 32.768kHz to 1MHz.

4.21 UART

The PL011 PrimeCell UART is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited. The 24MHz reference clock to the UARTs is from the crystal oscillator that is part of OSC0.

Table 4-54 UART implementation

Property	Value
Location	Southbridge
Memory base address	• UART 0: `0x10009000` • UART 1: `0x1000A000` • UART 2: `0x1000B000` • UART 3: `0x1000C000`
Interrupt	• UART 0: 44 • UART 1: 45 • UART 2: 46 • UART 3: 47
DMA	• UART 0 TX: 7 • UART 0 RX: 6 • UART 1 TX: 5 • UART 1 RX: 4 • UART 2 TX: 3 • UART 2 RX: 2 **Note** You must set **DMAPSR** = `b01` in the SYS_DMAPSR register to select this peripheral for DMA access.
Release version	ARM UART PL011 r1p3
Platform Library support	**_platform_uart_entry** Handles all channel operations for the UART channels, reading characters, writing characters, and opening the channel.
Reference documentation	*PrimeCell UART (PL011) Technical Reference Manual* ARM DDI 0183

4.21.1 Variations from the 16C550 UART

The PrimeCell UART varies from the industry-standard 16C550 UART device as follows:

- receive FIFO trigger levels are 1/8, 1/4, 1/2, 3/4, and 7/8
- the internal register map address space, and the bit function of each register differ
- the deltas of the modem status signals are not available
- 1.5 stop bits not available (1 or 2 stop bits only are supported)
- no independent receive clock.

Note

Because of changes in the programmer's model, the PrimeCell UART (PL011) is not backwards compatible with the previous PrimeCell UART (PL010).

4.22 USB interface

The USB interface is provided by a Philips ISP1761 controller that provides a standard USB host controller and an *On-The-Go* (OTG) dual role device controller. The USB host has two downstream ports. The OTG can function as either a host or slave device.

Table 4-55 USB implementation

Property	Value
Location	Board (an ISP1761 chip)
Memory base address	`0x4F000000` (mapped onto the SMC bus)
Interrupt	61
DMA	There are two DMA channels available for the USB controller. These are selectable as 0 or 1. See *Single Master Direct Memory Access Controller, SMDMAC* on page 4-50. **Note** You must set **DMAPSR** = `b00` in the SYS_DMAPSR register to select this peripheral for DMA access.
Release version	Custom interface to external controller
Reference documentation	*ISP1761 Hi-Speed Universal Serial Bus On-The-Go controller Product data sheet* (see also *USB Interface* on page 3-31 and test program supplied on the CD)

The ISP1761 has the following features:

- fully compliant to the USB Rev. 2.0 specification
- fully compliant to the USB On-The-Go specification
- includes high-performance USB peripheral controller with integrated Serial Interface Engine, FIFO memory, and transceiver
- configurable number of downstream and upstream hosts or functions
- USB host is USB 2.0 compliant and supports up to *Full speed* (12Mb/s)
- programmable interrupts and DMA
- FIFO and 63KB on-chip RAM for USB.

The ISP1761 register base addresses are shown in Table 4-56.

Table 4-56 USB controller base address

Address	Description
0x4F000000	Host controller EHCI registers
0x4F00200	Peripheral controller registers
0x4F00300	Host controller configuration registers
0x4F000370	OTG controller registers
0x4F000400	Host controller buffer memory (63KB)

Note

The suspend/wakeup signals for the device and host controllers are connected to GPIO2 (see *General Purpose Input/Output, GPIO* on page 4-53).

4.23 Watchdog

The SP805 Watchdog module is an AMBA compliant SoC peripheral developed and tested by ARM Limited.

The module is an AMBA slave module and connects to the *Advanced Peripheral Bus* (APB). The Watchdog module consists of a 32-bit down counter with a programmable timeout interval that has the capability to generate an interrupt and a reset signal on timing out. It is intended to be used to apply a reset to a system in the event of a software failure.

Table 4-57 Watchdog implementation

Property	Value
Location	Southbridge
Memory base address	• Watchdog 0: `0x1000F000` • Watchdog 1: `0x10010000`
Interrupt	• Watchdog 0: 32 • Watchdog 1: 72
DMA	–
Release version	ARM WDOG SP805 r2p0
Platform Library support	No support provided.
Reference documentation	*ARM Watchdog Controller (SP805) Technical Reference Manual*

Note

The Watchdog counter is disabled if the core is in debug state.

4.24 CompactFlash interface

The CompactFlash interface is a custom AMBA AHB compliant peripheral developed by ARM Limited.

The module is an AMBA slave module and connects to the *Advanced High-performance Bus* (AHB). The interface supports:

- True IDE Mode (16-bit)
- I/O Mode (data and task file register read and write access only).

Table 4-58 CompactFlash implementation

Property	Value
Location	Southbridge
Memory base address	• Access: `0x18000000` • Access: `0x18000100` (alternative status and control registers) • Control: `0x18000300` • Expansion: `0x18000304`
Interrupt	59
DMA	–
Release version	Custom logic
Platform Library support	`yes`
Reference documentation	*CF+ and CompactFlash Specification Revision 4.1*

4.24.1 CompactFlash Control Register, CF_CTRL

The CompactFlash control register at `0x18000300` provides control and status information for the inserted CF card.

Figure 4-25 on page 4-78 shows the register bit allocations.

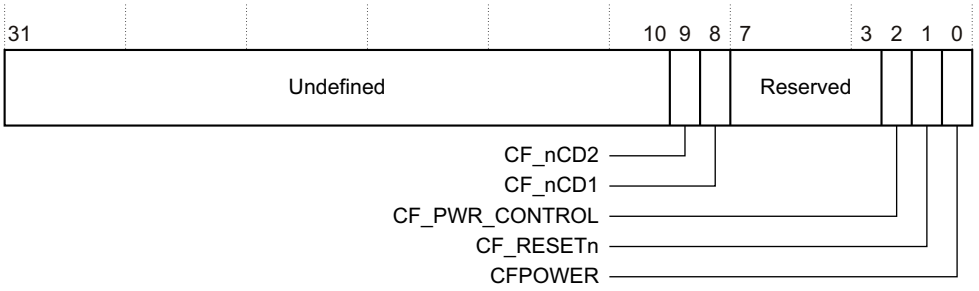


Figure 4-25 CF_CTRL Register

The function of the register bits are shown in Table 4-59.

Table 4-59 CF_CTRL register bit assignments

Bits	Access	Name	Reset	Description
[31:12]	Write ignored, read as zero	–	0x00000	Undefined
[11;10]	Write ignored, read as zero	–	b00	Undefined
[9]	Read only	CF_nCD2	b1	Card Detection: b00: card inserted bx1: card not inserted b1x: card not inserted
[8]	Read only	CF_nCD1	b1	
[7:3]	Write ignored, read as zero	–	b00000	Reserved
[2]	Read/Write	CF_PWR_CONTROL	b0	Power Control: b0: determined by CFPOWER (bit 0) b1: determined by chip detect (CF card)
[1]	Read/Write	CF_RESETn	b0	Card Reset (active low)
[0]	Read/Write	CFPOWER	b0	Card Power: b0: no power applied to card b1: 3V3 applied to card

Chapter 5
Processor Sub-System

This chapter provides details of the ARM11 MPCore test chip which affect the use of the PB11MPCore baseboard.

It contains the following sections:

- *ARM11 MPCore test chip overview* on page 5-2
- *Clocks* on page 5-4
- *Resets and interrupts* on page 5-11
- *Power supply control* on page 5-18
- *Memory configuration* on page 5-20
- *L220 bypass and peripheral decode* on page 5-22
- *Debug and configuration using JTAG* on page 5-27.

Note

Refer to the *ARM11 MPCore Processor Technical Reference Manual* and the *L220 Cache Controller Technical Reference Manual* for further details on the major test chip components.

5.1 ARM11 MPCore test chip overview

Figure 5-1 shows the top-level functionality of the test chip.

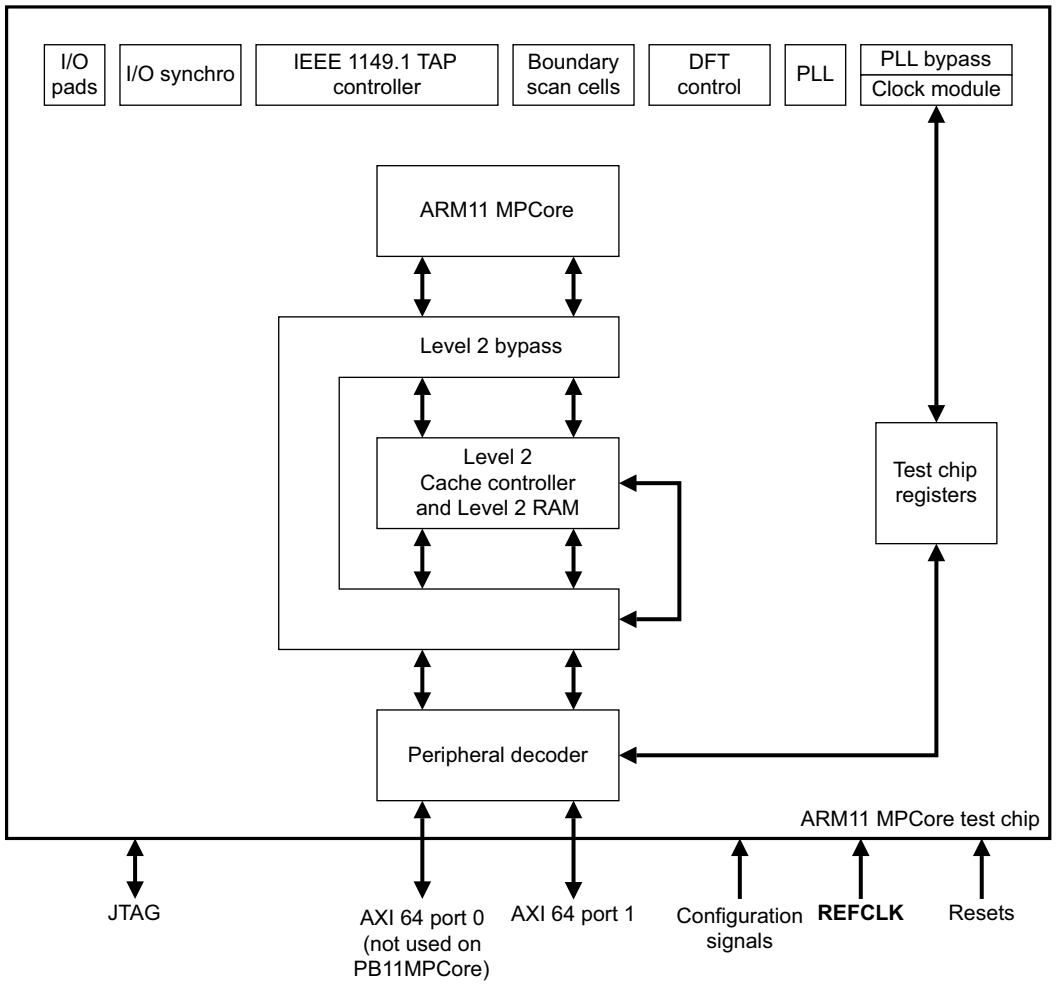


Figure 5-1 Top-level view of ARM11 MPCore test chip

The main features of the test chip are:

- an ARM11 MPCore incorporating:
  - four ARM11 MPCore CPUs with Vector Floating Point (VFP) that implement the ARM architecture v6
  - Level 1 (L1) memory subsystem providing 32KB instruction cache and 32KB data cache per CPU
  - JTAG-based debug.
- L220 Cache Controller incorporating 1MB of Level 2 (L2) unified cache
- peripheral decoder providing dual 64-bit external AXI buses
- on-chip PLL with skew control
- test chip register bank for configuring the ARM11 MPCore test chip
- design for test (DFT):
  - test control via a TEST_MODE test data register
  - L1 and L2 memory systems Memory Built-In Self Test (MBIST)
  - full chip and boundary scan support

See the *ARM11 MPCore Processor Technical Reference Manual* (DDI 0360) and the *L220 Cache Controller Technical Reference Manual* (DDI 329) for further details on these features.

5.2 Clocks

This section describes the clocking requirements for the ARM11 MPCore test chip and contains the following subsections:

- *Clocking overview*
- *PLL* on page 5-5
- *PLL bypass module* on page 5-8
- *Clock signals overview* on page 5-10.

5.2.1 Clocking overview

Figure 5-2 shows the clock domains within the ARM11 MPCore test chip.

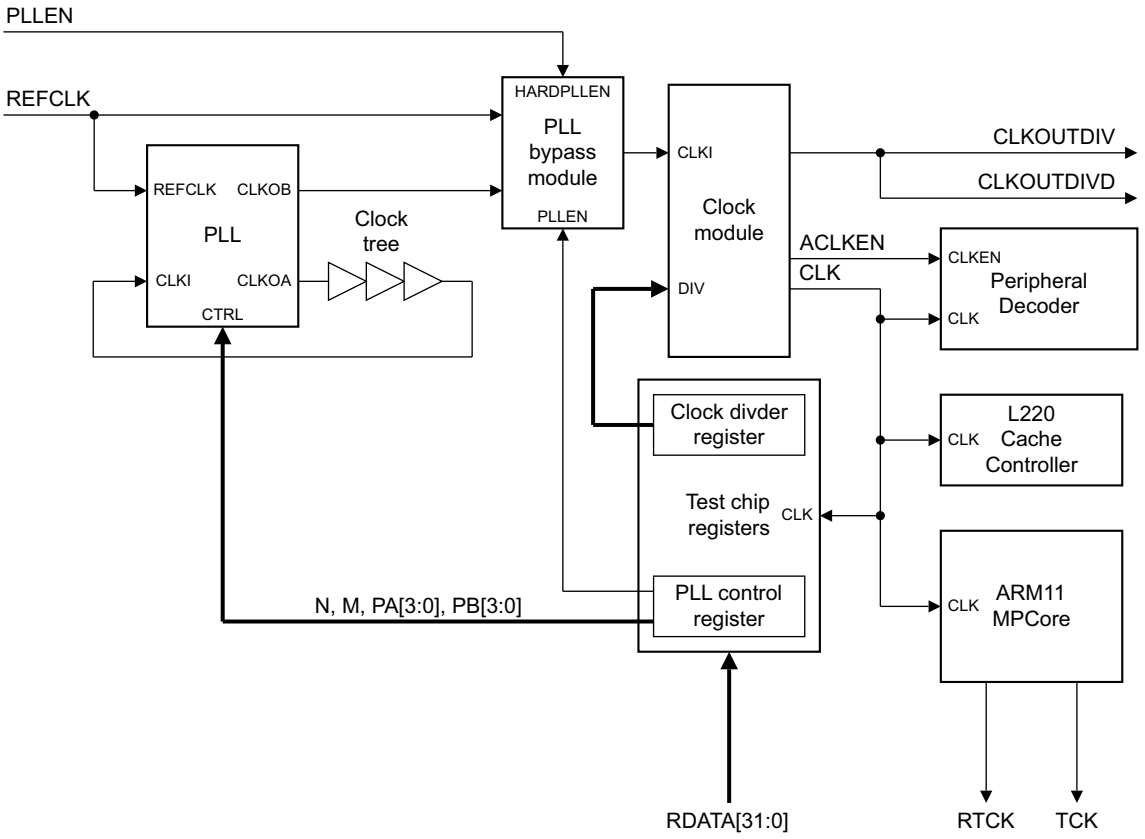


Figure 5-2 Clock domain overview

The ARM11 MPCore test chip clock domain consists of the PLL, the PLL bypass module, and the Clock module. As well as generating the main clock, **CLK** the PLL bypass module and the Clock module enable the bypassing of the PLL so that the test chip input clock, **REFCLK** can be used to generate the internal and external clocks. The PLL bypass module ensures that the Clock module can switch from the PLL output clock, **CLKOB** to the test chip input clock, **REFCLK** without glitches.

Peripheral decoder

The Peripheral decoder is required to interface externally at a lower AXI bus frequency. A clock enable signal, **ACLKEN** combined with the main clock, **CLK** are used to drive this interface. **CLK** and **ACLKEN** are generated by the test chip Clock module.

L220 Cache Controller

Clocking of the L220 RAM arrays is synchronized to the main clock, **CLK** by the L220.

ARM11 MPCore

The ARM11 MPCore has a single clock domain and is clocked by the main clock, **CLK**. generated by the test chip Clock module.

The debug clock signals **TCK** and **RTCK** are generated in the ARM11 MPCore. **TCK** is synchronized to the main clock, **CLK** in the ARM11 MPCore.

ARM11 MPCore test chip registers

The test chip registers are clocked by the main clock, **CLK** generated by the test chip Clock module.

5.2.2 PLL

The ARM11 MPCore test chip contains an on-board PLL with skew control. Figure 5-3 on page 5-6 shows a block diagram of the PLL.

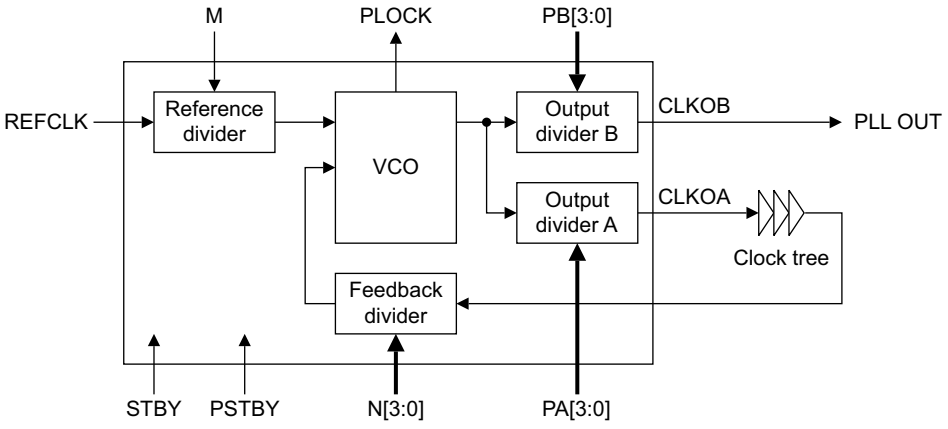


Figure 5-3 PLL block diagram

The following equations explain the relationships between **REFCLK**, **CLKOA**, **CLKOB**, **Fvco**, PA, PB, M, and N:

$$\frac{\textbf{REFCLK}}{M} = \frac{\textbf{Fvco}}{PA \times N} \quad \text{therefore,} \quad \textbf{Fvco} = \frac{\textbf{REFCLK} \times PA \times N}{M}$$

$$\textbf{CLKOA} = \frac{\textbf{Fvco}}{PA} \quad \text{therefore,} \quad \textbf{CLKOA} = \frac{\textbf{REFCLK} \times N}{M}$$

$$\textbf{CLKOB} = \frac{\textbf{Fvco}}{PB} \quad \text{therefore,} \quad \textbf{CLKOB} = \frac{\textbf{REFCLK} \times PA \times N}{M \times PB}$$

Referring to the *PLL block diagram* on page 5-6:

- **N[3:0]** can have a value between 1 and 15
- **M** can have a value of 1, or 2
- **PA[3:0]** can have a value between 1 and 15
- **PB[3:0]** can have a value between 1 and 15.

The frequency limits for **CLKOA** and **CLKOB** when PA = PB = N are listed in Table 5-1.

Table 5-1 PLL frequency limits for different divisors

PA, PB, N	CLKOA, CLKOB
1	300 - 540MHz
2	150 - 270MHz
3	100 - 180MHz
4	75 - 135MHz
5	60 - 108MHz
6	50 - 90MHz
7	43 - 77MHz
8	38 - 67MHz
9	34 - 54MHz
10	28 - 49MHz
11	25 - 45MHz

Table 5-1 PLL frequency limits for different divisors (continued)

PA, PB, N	CLKOA, CLKOB
12	24 - 40MHz
13	22 - 38MHz
14	20 - 36MHz
15	19 - 33MHz

5.2.3 PLL bypass module

The PLL bypass module enables you to select between **REFCLK** and **CLKOB** as output. These signals are considered to be unrelated and asynchronous. Logic is included in the test chip to prevent glitches when the selected clock changes.

Note

Any pulse, HIGH or LOW, with a duration shorter than the corresponding pulse of the input clock, before or after the selected clock switches, is considered to be a glitch.

To ensure glitch free clock switching:

- the selection of the previous clock must go from HIGH to LOW before the selection of the new clock goes from LOW to HIGH
- A guard band time must be provided between deselecting and selecting the clocks to guarantee there is no overlap. This is because the clocks have different frequencies and synchronizations. The guard band time is set by using a timer. The lowest frequency, **REFCLK**, drives the timer, and the timer counts four cycles whenever the selection changes.

Note

REFCLK is considered to be the lowest clock frequency, so it must always be lower than **CLKOB**. You must take this into account when you program the PLL divider values.

Either of the following can bypass the PLL:

- a hardware PLL enable, **PLLEN**, an external primary input pin
- bit 0, of the test chip PLL control register, a software PLL enable.

You can initialize the *Test chip PLL control register* before you release the main reset by using the test chip **CONFIGINIT** feature. This is the initialization method used by the Southbridge on the PB11MPCore. See *Clocks* on page 5-4 for further information on this hardware initialization feature.

5.2.4 Clock signals overview

The clock related signals are summarized in Table 5-2.

Table 5-2 Clock signals

Signal	Direction	Description	Toggling rate	Pad type
CLKOUTDIV	Output	This is the AXI bus clock.	18.75-133MHz	9mA output pad
CLKOUTDIVD	Output	Reflects exactly the **CLKOUTDIV** output. This is a second pad to drive out the AXI bus clock.	18.75-133MHz	9mA output pad
CLKOUTDIVSTOPPED	Output	When HIGH, indicates that the **CLKOUTDIV** output becomes LOW.	18.75-133MHz	9mA output pad
CONFIGINIT	Input	Clock enable according to **REFCLK**. The PLL and clock divider control registers use this clock enable.	18.75-133MHz	Standard input pad
nCONFIGRST	Input	Resets the PLL and clock divider control registers.	Not applicable	Schmidt trigger input pad
PLLEN	Input	Enables or disables the PLL. When LOW, **REFCLK** drives the entire test chip.	Static input	Standard input pad
PLOCK	Output	Indicates whether the PLL is locked. 0: PLL is not locked. 1: PLL is locked	18.75-133MHz	9mA output pad
REFCLK	Input	Global clock. This clock is a reference clock for the PLL.	18.75-133MHz	Schmidt trigger input pad

5.3 Resets and interrupts

This section describes the reset and interrupt signals that enter the ARM11 MPCore test chip pins and contains the following sections:

- *Resets*
- *Interrupts* on page 5-14

5.3.1 Resets

This section describes all the reset signals that enter the test chip:

nTRST This is the test logic reset. It also acts as the reset to the TAP controller.

nSYSPORESET

This is the main system reset on the ARM11 MPCore test chip. It resets:

- The peripheral decoder.
- The test chip-specific registers, apart from the test chip PLL control register, and the test chip clock divider register.
- The L220.
- All the debug registers that can be reset, and all the watch and breakpoint registers in the ARM11 MPCore. It also drives the ARM11 MPCore snoop control unit (SCU) **nSCURESET** signal.
- **MBISTRESETN** of the ARM11 MPCore MBIST controller.
- **MBISTRESETN** of the L220 MBIST controller.

nCONFIGRST

This signal resets:

- the test chip PLL control register
- the test chip clock divider register
- the PLL bypass module
- the clock module.

Note

The PB11MPCore uses the **CONFIGINIT** feature and activates this signal before all other reset signals. See *ARM11 MPCore test chip clocks* on page 3-43 for more information.

nWDRESET[3:0]

Individual watchdog resets inputs.

nCPURESET[3:0]

These signals individually reset each ARM11 MPCore processor by waking up processors that are in WFI state.

nPORESET[3:0]

Drives the ARM11 MPCore **nPORESET[3:0]** inputs.

RESETREQ[3:0]

This is an output to request individual watchdog resets.

Table 5-3 describes how the resets are handled.

Table 5-3 Reset handling

nTRST	nCONFIGRST	nSYSPORESET	nPORESET [3:0]	nWDRESET [3:0]	nCPURESET [3:0]	Description
X	0	X	X	X	X	Resets: • test chip PLL control register • test chip clock divider register • PLL bypass module • clock module.
X	X	0	All 0	All 0	All 0	Power-on reset. Resets: • all processors • Snoop Control Unit (SCU) • interrupt distributor • test chip logic.
X	X	1	[n]=0	[n]=0	[n]=0	Individual processor power-on reset. Resets all processor logic, including debug logic.

Table 5-3 Reset handling (continued)

nTRST	nCONFIGRST	nSYSPORESET	nPORESET [3:0]	nWDRESET [3:0]	nCPURESET [3:0]	Description
X	X	1	All 1	[n]=0	[n]=0	Individual processor soft reset. Resets all processor logic, excluding debug logic.
X	X	1	All 1	All 1	[n]=0	Individual processor reset from watchdog request, processor logic reset, excluding debug and watchdog reset flag.
0	X	X	X	X	X	Resets the TAP logic embedded in the ARM11 MPCore. Generates and synchronizes: • **TDI** • **TMS** • **TCK** • **RTCK** • test chip TAP controller.
X	X	1	All 1	All 1	All 1	No reset, normal run mode.

Note

You can assert all reset signals asynchronously, but you must deassert them synchronously with respect to their clock domains.

When in test mode, **TESTMODE** is tied to Vdd. When the test mode selected enables the SCAN, all resets must be asynchronous. Table 5-4 lists the reset deassertions with the respective clock.

Table 5-4 Reset deassertion with respective clock

Reset signal	Deasserted synchronously with primary input:	Deasserted synchronously with Clock module output:
nTRST	**TCK**	**CLK** when **ACLKEN** is HIGH
nCONFIGRST	**REFCLK**	**CLK**
nSYSPORESET	N/A	**CLK** when **ACLKEN** is HIGH
nWDRESET[3:0]	N/A	**CLK** when **ACLKEN** is HIGH
nCPURESET[3:0]	N/A	**CLK** when **ACLKEN** is HIGH
nPORESET[3:0]	N/A	**CLK** when **ACLKEN** is HIGH

5.3.2 Interrupts

This section describes all the interrupt signals that enter the test chip:

The ARM11 MPCore test chip has the following interrupt lines:

nIRQ[3:0] CPU legacy IRQ request input lines. Each bit is connected directly to each corresponding CPU:

nIRQ [0] to CPU#[0]
nIRQ [1] to CPU#[1]
nIRQ [2] to CPU#[2]
nIRQ [3] to CPU#[3]

nFIQ[3:0] CPU private FIQ request input lines. Each bit is connected directly to each corresponding CPU:

nFIQ [0] to CPU#[0]
nFIQ [1] to CPU#[1]
nFIQ [2] to CPU#[2]
nFIQ [3] to CPU#[3]

You can configure each *Fast Interrupt Request* (FIQ) as a *Non Maskable Interrupt* (NMI) by using a specific test chip register, the test chip interrupt control register. Figure 5-4 on page 5-15 shows the test chip interrupt control register bit assignments.

This register is located at address: 0x1F003004.

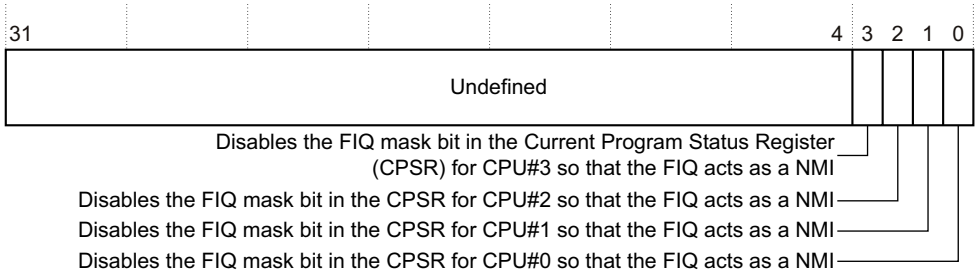


Figure 5-4 Test chip interrupt control register

Table 5-5 describes the test chip interrupt control register.

Table 5-5 Test chip interrupt control register

Bits	Read/write	Reset value	Description
[31:4]	Read as zero, write ignored	0	Undefined
[3]	RW	0	Disable the FIQ mask bit in the *Current Program Status Register* (CPSR) for CPU#[3] so that the FIQ acts as a NMI
[2]	RW	0	Disable the FIQ mask bit in the CPSR for CPU#[2] so that the FIQ acts as a NMI
[1]	RW	0	Disable the FIQ mask bit in the CPSR for CPU#[1] so that the FIQ acts as a NMI
[0]	RW	0	Disable the FIQ mask bit in the CPSR for CPU#[0] so that the FIQ acts as a NMI

INT[15:0] Distributed Interrupt Controller hardware interrupts. Each bit is connected directly to the corresponding Distributed Interrupt Controller hardware interrupt line.

Dual flip flops, clocked by the output clock of the clock module, **CLKOUT**, synchronize the **nIRQ[3:0]**, **nFIQ[3:0]**, and **INT[15:0]** signals.

Interrupt Routing

A total of 32 interrupt lines, **INT[31:0]** are provided by the *Distributed Interrupt Controller* in the MPCore multiprocessor. Refer to the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360) for further details on interrupt handling within the ARM11 MPCore.

Only **INT [15:0]** are available at the input pins of the ARM11 MPCore test chip for connection to external interrupts, **INT [31:16]** are used internally. Table 5-6 lists the internal connections of **INT[31:16]** in the test chip.

Table 5-6 Internal interrupt lines

MPCore interrupt line number	Description
31	Interrupt from L220: Decode error received on master ports from L3
30	Interrupt from L220: Slave error received on master ports from L3
29	Interrupt from L220: Event counter overflow/increment
28	Interrupt from MPCore (SCU): **PMUIRQ [11]**
27	Interrupt from MPCore (SCU): **PMUIRQ [10]**
26	Interrupt from MPCore (SCU): **PMUIRQ [9]**
25	Interrupt from MPCore (SCU): **PMUIRQ [8]**
24	Interrupt from MPCore (SCU): **PMUIRQ [7]**
23	Interrupt from MPCore (SCU): **PMUIRQ [6]**
22	Interrupt from MPCore (SCU): **PMUIRQ [5]**
21	Interrupt from MPCore (SCU): **PMUIRQ [4]**
20	Interrupt from MPCore (CPU3): **PMUIRQ [3]**
19	Interrupt from MPCore (CPU2): **PMUIRQ [2]**
18	Interrupt from MPCore (CPU1): **PMUIRQ [1]**
17	Interrupt from MPCore (CPU0): **PMUIRQ [0]**
16	Tied internally to Gnd

Interrupt routing to the **INT [15:0]** input pins of the test chip is controlled by values written to the INTMODE[1:0] field in the SYS_PLD_CTRL1 register that allows the selection of three routing modes. Refer to *ARM11 MPCore Distributed Interrupt Controller* on page 3-49 for details.

5.4 Power supply control

This section describes the ARM11 MPCore test chip power management.

The whole test chip has a single power domain. No *Intelligent Energy Management*, IEM or adaptive shutdown mechanism is provided. Software can put each ARM11 MPCore CPU in WFI mode. In this case, clock gating ensures that the minimum amount of logic is toggling. Some ARM11 MPCore outputs that provide power information are stored in a specific test chip register. Figure 5-5 shows the test chip power status register bit assignments.

This register is located at address: `0x1F00300C`.

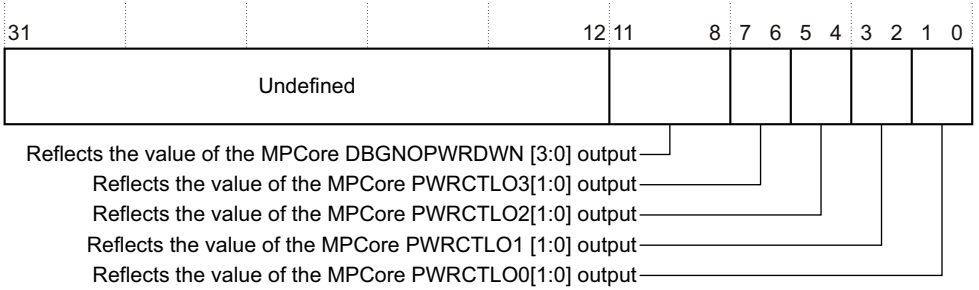


Figure 5-5 Test chip power status register

Table 5-7 lists the test chip power status register bit assignments.

Table 5-7 Test chip power status register

Bits	Read/write	Reset value	Description
[31:12]	Read as zero, write ignored.	0	Undefined.
[11:8]	RO	b0000	Reflects the value of the ARM11 MPCore **DBGNOPWRDWN [3:0]** output. See the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360) for more information.
[7:6]	RO	b00	Reflects the value of the ARM11 MPCore **PWRCTLO3[1:0]** output. See the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360) for more information.

Table 5-7 Test chip power status register (continued)

Bits	Read/write	Reset value	Description
[5:4]	RO	00	Reflects the value of the ARM11 MPCore **PWRCTLO2[1:0]** output. See the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360) for more information.
[3:2]	RO	00	Reflects the value of the ARM11 MPCore **PWRCTLO1 1:0]** output. See the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360) for more information.
[1:0]	RO	00	Reflects the value of the ARM 11 MPCore **PWRCTLO0[1:0]** output. See the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360) for more information.

5.5 Memory configuration

The ARM11 MPCore test chip implements *Level 1* (L1) and *Level 2* (L2) memory subsystems.

The ARM11 MPCore L1 memory subsystem has 32KB of instruction cache and 32KB of data cache per CPU.

The L220 cache controller memory subsystem has 1MB of L2 unified cache.

Note

Software can change the L2 cache size using L2 control registers. See *Peripheral decoder* on page 5-24 for details of the L220 register base address allocation and the *L220 Cache Controller Technical Reference Manual* (ARM DDI 0329) for details of the L2 control registers.

5.5.1 ARM11 MPCore

The *Level 1* (L1) memory subsystem has 32KB of instruction cache and 32KB of data cache per CPU.

The ARM11 MPCore control register block requires 8KB of memory space:
base address is `0x1F000000`, the address range is `0x1F000000` to `0x1F001FFF`.

5.5.2 L220 cache controller

The L2 memory consists of 1MB of L2 unified cache. 1MB is physically implemented, and software can change the L2 cache size using L2 control registers.

L2CC control register block requires 4KB memory space:
base address is `0x1F002000`, the address range is `0x1F002000` to `0x1F002FFF`.

The fixed configuration settings for the L220 are:

- the L220 does not use parity
- the TrustZone feature is not required, so all transactions are considered to be secure
- *Intelligent Energy Manager* (IEM) support is not provided
- L220 and ARM11 MPCore are synchronous, and L2 cache operates at the core frequency
- one master AXI port (master port 1) is used
- the way size is fixed at 128K.

Note

You can bypass the L2 cache using the L220 Control Register. This is the default state. In this case, you have an additional latency of the pipeline length of the L2 cache.

For details on the L220 Level 2 cache implementation see the *L220 Cache Controller Technical Reference Manual* (ARM DDI 0329).

RAM organization

Data RAM requires byte-enable access for data RAM without parity. Figure 5-6 shows the L220 data RAM organization.

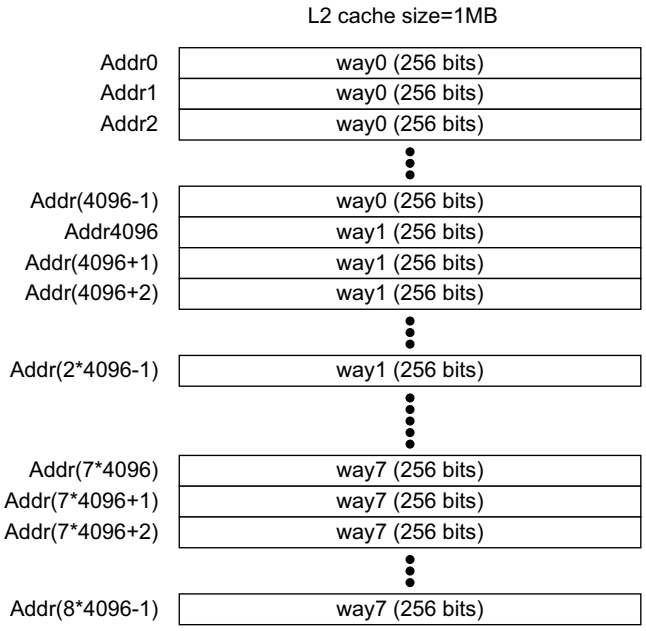


Figure 5-6 L220 data RAM organization

5.6 L220 bypass and peripheral decode

This section describes the L220 bypass mechanism and the built-in peripheral decoder. It contains the following sections:

- *L220 bypass module*
- *Peripheral decoder* on page 5-24

5.6.1 L220 bypass module

You can use the L220 bypass module shown in Figure 5-7 on page 5-23 to bypass the L220.

It is possible to have one or two AXI ports active at the L220 and ARM11 MPCore levels.

In the PB11MPCore the **L2MASTNUM** bit is hard-coded to provide a single master port (AXI port 1) at the L220 level.

Two master ports are used at the ARM11 MPCore level. See *MPMASTNUM selection* on page 2-9 for details.

Note

In a one master configuration, the default master for the L220 is master 1. For ARM11 MPCore, the default master is master 0. Therefore, inside the L220 bypass module, the two AXI bus connections are not only bypassed, but are also exchanged between ports 0 and 1.

Figure 5-7 on page 5-23 shows the L220 bypass mechanism.

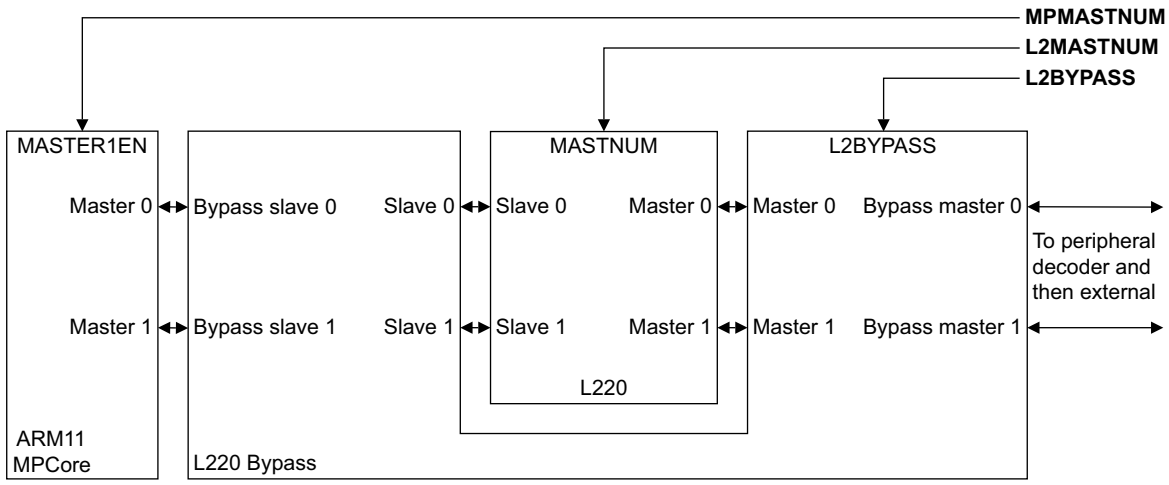


Figure 5-7 L220 bypass mechanism

Table 5-8 lists the most useful settings for the **L2BYPASS** and **MPMASTNUM** configuration bits.

Table 5-8 Most Useful settings of the L2BYPASS and MPMASTNUM configuration bits

L2BYPASS	MPMASTNUM	Description
0	1	The two MPCore masters are used. Master 1 of L220 is used, and AXI port 1 is used externally.
1	0	Master 0 of MPCore is used. Because AXI port 1 is used externally, the L220 bypass module exchanges the connection between port 0 and port 1.
0	0	Master 0 of MPCore is used. Master 1 of L220 is used, and AXI port 1 is used externally.

Figure 5-8 on page 5-24 shows the possible ARM11 MPCore and L220 port combinations using **MPMASTNUM**. The positioning of the Peripheral Decoder, is also shown. See *Peripheral decoder* on page 5-24 for further details.

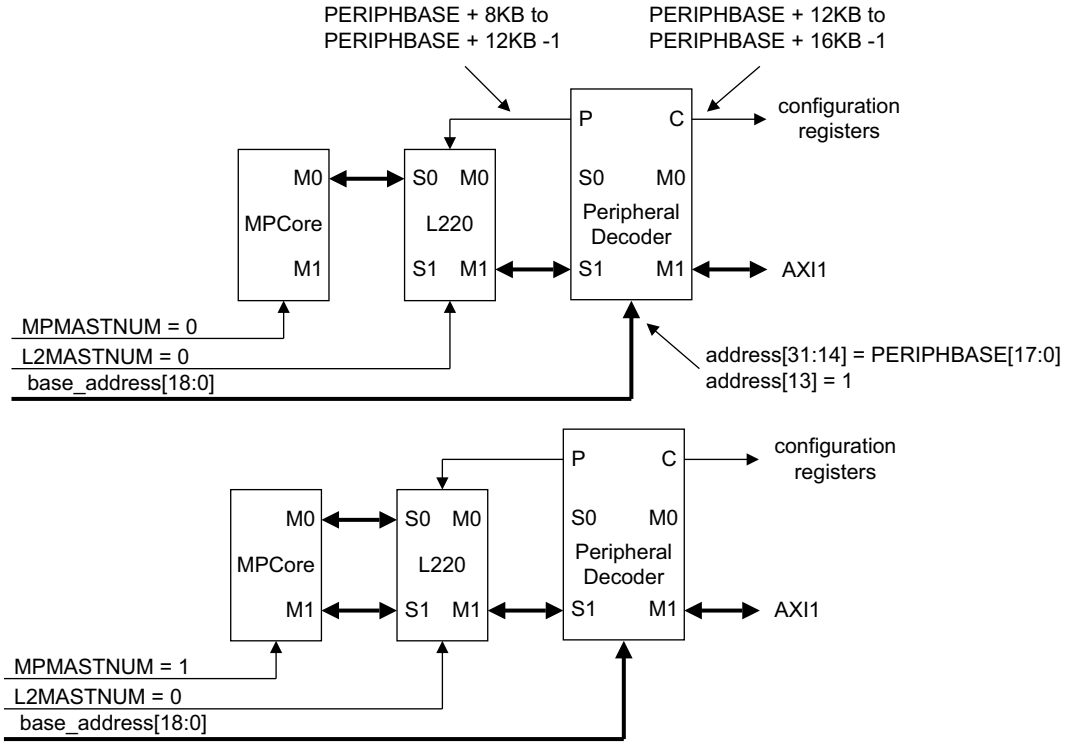


Figure 5-8 MPCore and L220 port configurations

5.6.2 Peripheral decoder

As shown in Figure 5-8, the ARM11 MPCore test chip integrates some AXI decoding logic for selecting the L220 peripheral port and the test chip configuration registers. The Peripheral decoder locally decodes the regions listed in Table 5-9.

Table 5-9 Peripheral decoder mapping

Address space	Peripheral accessed
[PERIPHBASE + 8KB - PERIPHBASE + 12KB -1]	L220 control register space
[PERIPHBASE + 12KB - PERIPHBASE + 16KB -1]	Test chip-specific registers space

The PERIPHBASE value is hard-coded using an external input bus to the test chip and together with address bit 13, defines the base address of the 16KB memory space that is allocated for ARM11 MPCore test chip control.

The hard-coding sets the ARM11 MPCore peripheral decoder base address to `0x1F002000`.

Note

The ARM11 MPCore *Snoop Control Unit* (SCU) decodes the region: [PERIPHBASE - PERIPHBASE + 8KB -1] to control the global registers of the ARM11 MPCore.

Refer to the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360) for further details.

Because accesses to the peripheral ports are rare, they have priority. When a peripheral access arrives from the slave port, the decoder waits for all accesses to **M0** and **M1** to complete before it acknowledges the peripheral access. This access then goes to either the **P** or the **C** port.

Note

During this time, no other accesses are accepted. If there are other accesses to a peripheral port from the same slave port, they are routed without wait.

The routing possibilities of the decoder are:

- idle
- normal
- **S0** to peripheral
- **S1** to peripheral.

In normal mode, accesses from **S0** are routed to **M0**, and accesses from **S1** are routed to **M1**.

Note

The master inputs and outputs are registered, so one cycle of delay is added for each channel.

In ‘**S0** to peripheral’ mode, all accesses from **S1** are blocked, and in ‘**S1** to peripheral’ mode, all accesses from **S0** are blocked. If an access from **S1** targets the peripheral region, it must wait for all transfers from **S0** to be completed. The state machine then goes into ‘idle’ mode, and **S1** goes into ‘peripheral’ mode.

Note

- The state machine always spends one cycle in ‘idle’ mode before it switches to another routing mode, irrespective of the current mode.
- Between the decoder and the downsizer in the Peripheral decoder, all the AXI channels are registered, so one delay cycle is added.

5.7 Debug and configuration using JTAG

This section describes the Debug and JTAG hardware of the test chip. It contains the following sections:

- *TAP controller*
- *Debug*
- *Configuration using JTAG* on page 5-28
- *TAP ID registers* on page 5-28

5.7.1 TAP controller

An IEEE 1149.1 TAP controller and boundary scan chain is instantiated at the top-level of the test chip to enabler test chip configuration using JTAG. This TAP controller uses the same IEEE 1149.1 pins that the RealView ICE communication uses during debug so that the number of pins required is kept to a minimum.

BYPASS, EXTEST, and SAMPLE/PRELOAD instructions are the minimum functionality required to perform configuration of the test chip using JTAG.

The test chip provides a specific ID code that can be obtained using the IDCODE instruction. Table 5-10 lists the test chip TAP instructions.

Table 5-10 Test chip TAP instructions

TAP controller instruction	Instruction encoding
BYPASS	b11111
EXTEST	b00000
IDCODE	b00001
SAMPLE/PRELOAD	b00010

5.7.2 Debug

When the Config switch on the front panel is OFF, you can debug the ARM11 MPCore using a DBGTAP debugger such as RealView ICE. There are four DBGTAP controllers in the ARM11 MPCore that have common **TCK**, **TMS**, and **nTRST** signals. The **TDI** and **TDO** paths are daisy-chained as shown in Figure 5-9 on page 5-28.

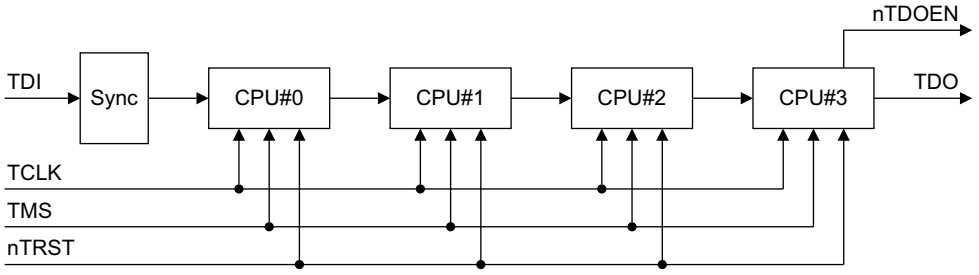


Figure 5-9 TAP signal connections between the four ARM11 MPCore CPUs

5.7.3 Configuration using JTAG

When the Config switch on the front panel is ON, the test chip can be configured at its input and output pins. All inputs and outputs (except for the 5-pin JTAG interface, and **RTCK**) have boundary scan cells on them. The boundary scan chain order follows that of the pads.

5.7.4 TAP ID registers

The TAP ID code is a 32-bit number divided into multiple fields. Figure 5-10 shows the TAP ID register bit assignments.

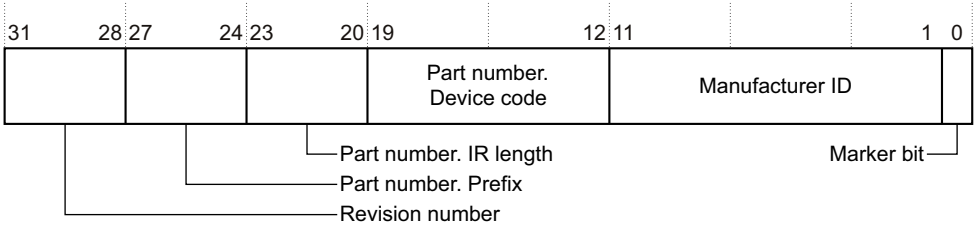


Figure 5-10 TAP ID register

You can access the *ARM11 MPCore TAP ID code* when the PB11MPCore is in Debug mode.

The ARM11 MPCore TAP ID code is:

```
0x07B37477
```

Table 5-11 lists how the register divides the number.

Table 5-11 ARM11 MPCore TAP ID register

Bit field	Use	Description	Value
[31:28]	Revision number	Revision number for the ARM11 MPCore, not necessarily the same as the test chip revision number.	0x0
[27:12]	Part number	[27:24] prefix. This is a fixed pattern.	0x7
		[23:20] IR length. Identifies the length of the instruction register within the TAP.	0xB
		[19:12] device code. This pattern identifies the ARM11 MPCore.	0x37
[11:1]	Manufacturer ID	Holds the officially registered pattern.	b0100 0111 011
[0]	Marker bit	Always set to 1, as required by the JTAG specification.	b1

See the *ARM11 MPCore Processor Technical Reference Manual* (ARM DDI 0360) for details of the remaining Debug registers in the ARM11 MPCore.

You can access the *test chip TAP ID code* when the PB11MPCore is in JTAG Configuration mode.

The test chip TAP ID is:

0x17536021

Table 5-12 lists how the register divides the number.

Table 5-12 Test chip TAP ID register

Bit field	Use	Description	Value
[31:28]	Revision number	Revision number for the test chip, not necessarily the same as the core revision number.	0x1
[27:12]	Part number	[27:24] prefix. This is a fixed pattern.	0x7
		[23:20] IR length. Identifies the length of the instruction register within the TAP.	0x5

Table 5-12 Test chip TAP ID register (continued)

Bit field	Use	Description	Value
		[19:12] device code. This pattern identifies the test chip.	0x36
[11:1]	Manufacturer ID	Holds the officially registered pattern.	b0000 0010 000
[0]	Marker bit	Always set to 1, as required by the JTAG specification.	b1

Appendix A
Signal Descriptions

This appendix provides a summary of signals present on the PB11MPCore connectors. It contains the following sections:

- *Compact Flash interface* on page A-2
- *Audio CODEC interface* on page A-6
- *MMC and SD card interface* on page A-7
- *Keyboard and mouse interface* on page A-9
- *GPIO interface* on page A-10
- *UART interface* on page A-11
- *Synchronous Serial Port interface* on page A-12
- *Smart Card interface* on page A-13
- *Ethernet interface* on page A-15
- *USB interface* on page A-16
- *DVI display interface* on page A-17
- *RealView Logic Tile header connectors* on page A-19
- *Test and debug connections* on page A-41.

A.1 Compact Flash interface

The PB11MPCore provides a Compact Flash connector on the front panel that enables you to connect a CompactFlash Storage Card to the Compact Flash interface. See *Front panel layout* on page 3-4 for the connector location.

Figure A-1 shows the connector pin numbering.

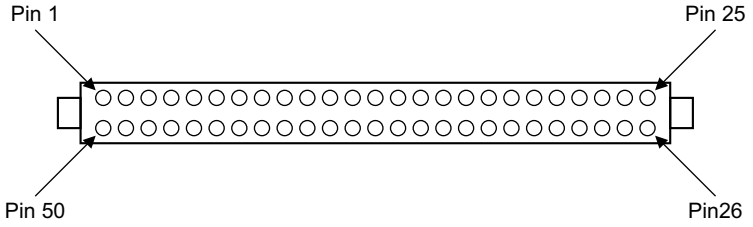


Figure A-1 Compact Flash connector pin numbering

The Compact Flash connector pinout specification supports three basic CompactFlash Storage Card modes:

- PC Card ATA using Memory Mode
- PC Card ATA using I/O Mode
- True IDE Mode

Some pin functions change dependant on the mode, and some also change dependant on the interface protocol used. For example, when PC Card I/O Mode is used, pin 34 has the following three protocol dependant functions:

~IORD When Ultra DMA Protocol is not active:
This is an I/O Read strobe generated by the host.

~HDMARDY When Ultra DMA Protocol DMA Read is active:
This signal indicates that the host is ready to receive Ultra DMA data-in bursts.

HSTROBE When Ultra DMA Protocol DMA Write is active:
This signal is the data out strobe generated by the host.

Note

The PB-A8 CompactFlash interface provides limited support:

- True IDE Mode (16-bit)
- PC Card I/O Mode (data and task file register read and write access only).

PC Card Memory Mode is not supported.

Table A-1 lists the CompactFlash connector pinout for each supported mode. Refer to the *CF+ & CF Specification Rev 4.1* for signal descriptions.

Table A-1 Compact Flash connector pinout

PC Card Memory Mode			PC Card I/O Mode			True IDE Mode		
Pin	**Signal**	**Type**	**Pin**	**Signal**	**Type**	**Pin**	**Signal**	**Type**
1	**GND**		1	**GND**		1	**GND**	
2	**D03**	I/O	2	**D03**	I/O	2	**D03**	I/O
3	**D04**	I/O	3	**D04**	I/O	3	**D04**	I/O
4	**D05**	I/O	4	**D05**	I/O	4	**D05**	I/O
5	**D06**	I/O	5	**D06**	I/O	5	**D06**	I/O
6	**D07**	I/O	6	**D07**	I/O	6	**D07**	I/O
7	**~CE1**	I	7	**~CE1**	I	7	**~CE0**	I
8	**A10**	I	8	**A10**	I	8	**A10**	I
9	**~OE**	I	9	**~OE**	I	9	**~ATA_SEL**	I
10	**A09**	I	10	**A09**	I	10	**A09**	I
11	**A08**	I	11	**A08**	I	11	**A08**	I
12	**A07**	I	12	**A07**	I	12	**A07**	I
13	**VCC**		13	**VCC**		13	**VCC**	
14	**A06**	I	14	**A06**	I	14	**A06**	I
15	**A05**	I	15	**A05**	I	15	**A05**	I

Table A-1 Compact Flash connector pinout (continued)

PC Card Memory Mode			PC Card I/O Mode			True IDE Mode		
Pin	**Signal**	**Type**	**Pin**	**Signal**	**Type**	**Pin**	**Signal**	**Type**
16	**A04**	I	16	**A04**	I	16	**A04**	I
17	**A03**	I	17	**A03**	I	17	**A03**	I
18	**A02**	I	18	**A02**	I	18	**A02**	I
19	**A01**	I	19	**A01**	I	19	**A01**	I
20	**A00**	I	20	**A00**	I	20	**A00**	I
21	**D00**	I/O	21	**D00**	I/O	21	**D00**	I/O
22	**D01**	I/O	22	**D01**	I/O	22	**D01**	I/O
23	**D02**	I/O	23	**D02**	I/O	23	**D02**	I/O
24	**WP**	O	24	**~IOIS16**	O	24	**~IOCS16**	O
25	**~CD2**	O	25	**~CD2**	O	25	**~CD2**	O
26	**~CD1**	O	26	**~CD1**	O	26	**~CD1**	O
27	**D11**	I/O	27	**D11**	I/O	27	**D11**	I/O
28	**D12**	I/O	28	**D12**	I/O	28	**D12**	I/O
29	**D13**	I/O	29	**D13**	I/O	29	**D13**	I/O
30	**D14**	I/O	30	**D14**	I/O	30	**D14**	I/O
31	**D15**	I/O	31	**D15**	I/O	31	**D15**	I/O
32	**~CE2**	I	32	**~CE2**	I	32	**~CS1**	I
33	**~VS1**	O	33	**~VS1**	O	33	**~VS1**	O
34	**~IORD HSTROBE ~HDMARDY**	I	34	**~IORD HSTROBE ~HDMARDY**	I	34	**~IORD HSTROBE ~HDMARDY**	I
35	**~IOWR STOP**	I	35	**~IOWR STOP**	I	35	**~IOWR STOP**	I
36	**~WE**	I	36	**~WE**	I	36	**~WE**	I
37	**READY**	O	37	**~IREQ**	O	37	**INTRQ**	O

Table A-1 Compact Flash connector pinout (continued)

PC Card Memory Mode			PC Card I/O Mode			True IDE Mode		
Pin	**Signal**	**Type**	**Pin**	**Signal**	**Type**	**Pin**	**Signal**	**Type**
38	**VCC**		38	**VCC**		38	**VCC**	
39	**~CSEL**	I	39	**~CSEL**	I	39	**~CSEL**	I
40	**~VS2**	O	40	**~VS2**	O	40	**~VS2**	O
41	**RESET**	I	41	**RESET**	I	41	**RESET**	I
42	**~WAIT** **~DDMARDY** **DSTROBE**	O	42	**~WAIT** **~DDMARDY** **DSTROBE**	O	42	**IORDY** **~DDMARDY** **DSTROBE**	O
43	**~INPACK** **~DMARQ**	O	43	**~INPACK** **~DMARQ**	O	43	**DMARQ**	O
44	**~REG** **~DMACK**	I	44	**~REG** **DMACK**	I	44	**~DMACK**	I
45	**BVD2**	O	45	**~SPKR**	O	45	**~DSAP**	I/O
46	**BVD1**	O	46	**~STSCHG**	O	46	**~PDIAG**	I/O
47	**D08**	I/O	47	**D08**	I/O	47	**D08**	I/O
48	**D09**	I/O	48	**D09**	I/O	48	**D09**	I/O
49	**D10**	I/O	49	**D10**	I/O	49	**D10**	I/O
50	**GND**		50	**GND**		50	**GND**	

A.2 Audio CODEC interface

The PB11MPCore provides three stacked 3.5mm jack connectors on the rear panel that enable you to connect to the analog microphone and auxiliary line level input and output on the CODEC. Figure A-2 shows the pinouts of the sockets. The *Rear panel layout* on page 3-5 shows the connector location.

Note

A link (LK3) on the baseboard enables bias voltage to be applied to the microphone.

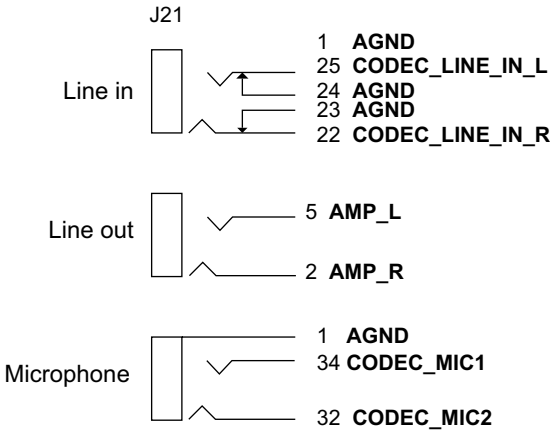


Figure A-2 Audio connectors

A.3 MMC and SD card interface

The MMC/SD card socket shown in Figure A-3 is positioned on the front panel of the enclosure, see *Front panel layout* on page 3-4.

Caution

The MMC or SD card must be inserted into the front panel socket with the contacts facing down.

Figure A-3 shows the pin numbering and signal assignment. In addition, the socket contains switches that are operated by card insertion that provide signaling on the **CARDINx** and **MCI_WPROT** signals.

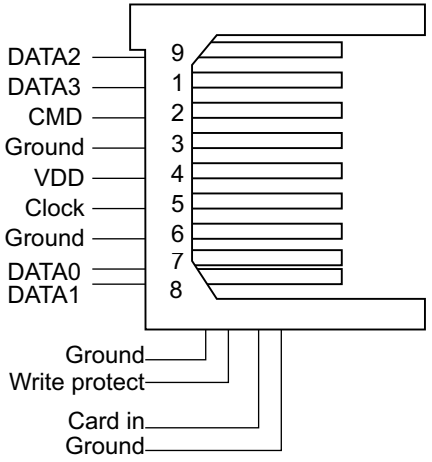


Figure A-3 MMC/SD card socket pin numbering

The MMC card uses seven pins, and the SD card uses all nine pins. The additional pins are located as shown in Figure A-3 with pin 9 next to pin 1 and pins 7 and 8 spaced more closely together than the other pins. Figure A-4 on page A-8 shows an MMC card, with the contacts face up.

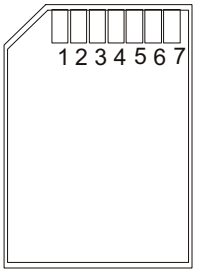


Figure A-4 MMC card

Table A-2 lists the signal assignments.

Table A-2 Multimedia Card interface signals

Pin	Signal	Function SD widebus mode	Function MCI
1	**MCIxDATA3**	Data	Chip select
2	**MCIxCMD**	Command/response	Data in
3	**GND**	Ground	Ground
4	**MCIVDDx**	Supply voltage	Supply voltage
5	**MCICLKx**	Clock	Clock
6	**GND**	Ground	Ground
7	**MCIxDATA0**	Data 0	Data out
8	**MCIxDATA1**	Data 1	NC
9	**MCIxDATA2**	Data 2	NC
10 (DET A)	**CARDINx**	Card insertion detect	Card insertion detect
11 (DET B)	**WPROTx**	Write protect status	Write protect status

A.4 Keyboard and mouse interface

The pinout of the KMI connectors J30A and J30B is shown in Figure A-5.

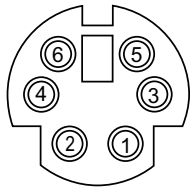


Figure A-5 KMI connector

Table A-3 shows signals on the KMI connectors.

Table A-3 Mouse and keyboard port signal descriptions

Pin	Keyboard (KMI0, J30B)		Mouse (KMI1, J30A)	
	Signal	Function	Signal	Function
1	**KDATA_FILT**	Keyboard data (filtered)	**MDATA_FILT**	Mouse Data (filtered)
2	NC	Not connected	NC	Not connected
3	**GNDKBD_FILT**	Ground (filtered)	**GNDMSE_FILT**	Ground (filtered)
4	**5VF2**	5V (filtered)	**5VF1**	5V (filtered)
5	**KCLK_FILT**	Keyboard clock (filtered)	**MCLK_FILT**	Mouse clock (filtered)
6	NC	Not connected	NC	Not connected

A.5 GPIO interface

Three eight-bit *General Purpose Input/Output* (GPIO) controllers are implemented in the Southbridge. GPIO ports 0 and 1 are available for use as general purpose external I/O on header J33 on the baseboard. GPIO port 3 is used internally by the PB11MPCore. See *Baseboard layout* on page 3-2 for the location of the GPIO header. The signals are shown in Figure A-6.

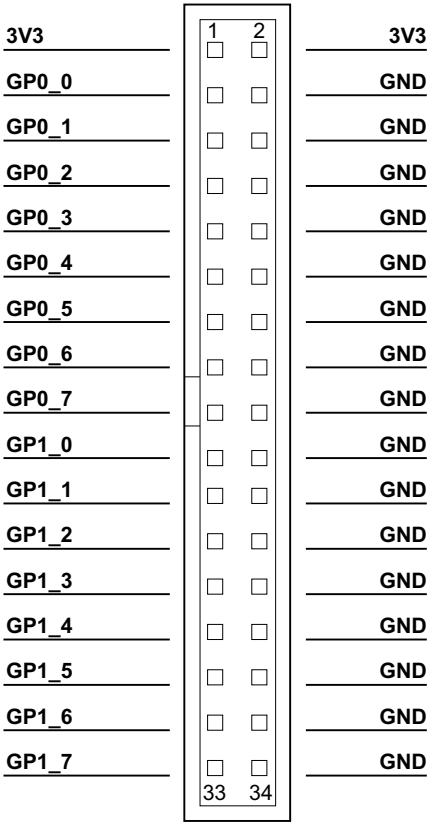


Figure A-6 GPIO connector

Note

Each data pin has an on-board 10KΩ pullup resistor to 3.3V.

A.6 UART interface

The PB11MPCore provides four serial transceivers on the rear panel of the enclosure.

Figure A-7 shows the pin numbering for the 9-pin D-type male connector used on the PB11MPCore and Table A-4 shows the signal assignment for the connectors.

The pinout shown in Figure A-7 is configured as a *Data Communications Equipment* (DCE) device.

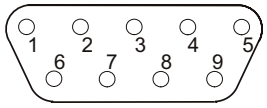


Figure A-7 Serial connector

Table A-4 Serial plug signal assignment

Pin	UART0 J24A (top)	UART1 J24B (bottom)	UART2 J25A (top)	UART3 J25B (bottom)
1	**SER0_DCD**	NC	NC	NC
2	**SER0_RX**	**SER1_RX**	**SER2_RX**	**SER3_RX**
3	**SER0_TX**	**SER1_TX**	**SER2_TX**	**SER3_TX**
4	**SER0_DTR**	**SER1_DTR**[a]	**SER2_DTR**[a]	**SER3_DTR**[a]
5	**SER0_GND**	**SER1_GND**	**SER2_GND**	**SER3_GND**
6	**SER0_DSR**	**SER1_DSR**	**SER2_DSR**	**SER3_DSR**
7	**SER0_RTS**	**SER1_RTS**	**SER2_RTS**	**SER3_RTS**
8	**SER0_CTS**	**SER1_CTS**	**SER2_CTS**	**SER3_CTS**
9	**SER0_RI**	NC	NC	NC

a. The signals **SER1_DTR**, **SER2_DTR**, and **SER3_DTR** are connected to the corresponding **SER1_DSR**, **SER2_DSR**, and **SER3_DSR** signals. These signals cannot be set or read under program control.

A.7 Synchronous Serial Port interface

Figure A-8 shows the signals on the expansion SSP interface connector J28. See *Baseboard layout* on page 3-2 for the location of the connector.

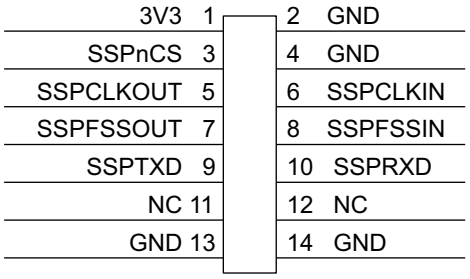


Figure A-8 SSP expansion interface

The signals associated with the SSP are shown in Table A-5.

Table A-5 SSP signal assignment

Signal name	Description
SSPCLKOUT	Clock output from controller
SSPCLKIN	Clock input to controller
SSPFSSOUT	Frame sync output
SSPFSSIN	Frame sync input
SSPTXD	Data output
SSPRXD	Data input
SSPnCS	Chip select

A.8 Smart Card interface

The PB11MPCore provides a SIM socket on the front panel of the enclosure. See *Front panel layout* on page 3-4 for the location of the SIM socket.

Caution

The SIM card must be inserted into the SIM socket with the contacts facing down and the chamfered edge facing out.

Figure A-9 shows the signal assignment of a smartcard. Pins 7 and 8 are not connected and are omitted on some cards.

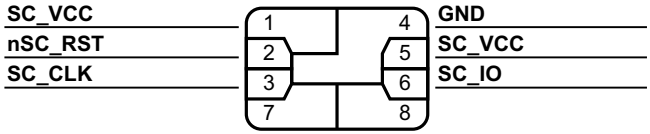


Figure A-9 Smartcard contacts assignment

The signals associated with the SCI are shown in Table A-6.

Table A-6 Smartcard connector signal assignment

Pin	Signal	Description
1	**SC_VCC**	Card power (1.8V, 3.3V, or 5V)
2	**SC_RST**	Reset to card
3	**SC_CLK**	Clock to or from card
4	GND	Ground
5	SC_VCC	Programming voltage
6	**SC_IO**	Serial data to or from the card
7	–	Reserved for future use
8	–	Reserved for future use

Figure A-10 shows the pinout of the SCI Expansion connector. The connector (J3) is located on the front panel PCB (HBI 0176) directly behind the USER LEDs. This can be used to connect to an external smart card device.

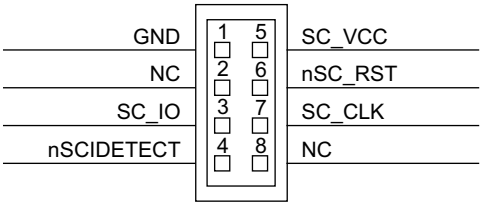


Figure A-10 SCI expansion

Table A-7 lists the signals on the SCI expansion connector.

Table A-7 Signals on SCI expansion connector

Pin	Signal	Description
1	**GND**	Ground
2	**NC**	Not connected
3	**SCI_IO**	SIM data
4	**nSCIDETECT**	Card detect for SIM
5	**SC_VCC**	SIM power
6	**nSC_RST**	Active LOW reset to SIM
7	**SC_CLK**	SIM clock
8	NC	Not connected

A.9 Ethernet interface

The RJ45 Ethernet connector (J14a) is shown in Figure A-11. It is part of the combined RJ45 and Dual USB Type A connector J14, positioned on the rear panel. See Figure A-12 on page A-16 for details of the combined Dual USB Type A connector (J14b).

LEDA (green) and LEDB (yellow) are connected to the LAN9118 controller. The function of the LEDs is determined by registers in the controller. Typical usage is to monitor transmit activity and packet detection.

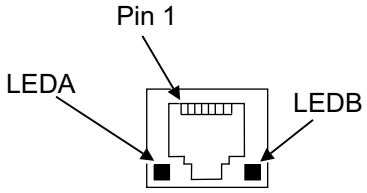


Figure A-11 Ethernet connector

The signals on the Ethernet cable are shown in Table A-8.

Table A-8 Ethernet signals

Pin	Signal
1	**Transmit +**
2	**Transmit -**
3	**Receive +**
4	NC
5	NC
6	**Receive -**
7	NC
8	NC

A.10 USB interface

USB1 provides an OTG interface and connects through the OTG connector J15 positioned on the front panel of the enclosure.

USB2 and USB3 (J14b) provide USB host interfaces and connect through the combined RJ45 and Dual USB Type A connector J14, positioned on the rear panel of the enclosure. See Figure A-11 on page A-15 for details of the combined RJ45 connector (J14a).

Note

For a full description of the USB signals refer to the datasheet for the NXP ISP1761 On-The-Go controller.

Figure A-12 shows the USB connectors and signals.

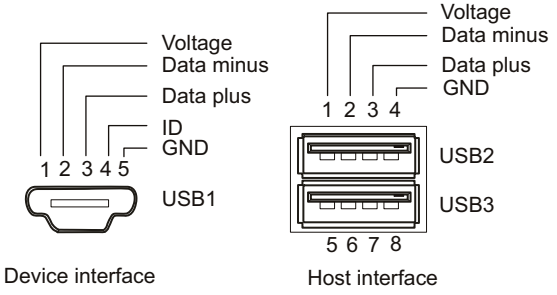


Figure A-12 USB connectors

A.11 DVI display interface

The DVI combined connector (J16) is positioned on the rear panel of the enclosure, see *Rear panel layout* on page 3-5. The connector signals are listed in Table A-9 on page A-18. The digital CLCD data from the Northbridge is passed to a T.M.D.S. transmitter to provide the DVI digital data and to a triple video DAC to provide the analogue RGB signals. The DDC2B interface is provided by a custom *Two-wire Interface* (SBCon) implemented in the Southbridge.

Note

The mechanical interconnect includes 29 signal contacts, that are divided into two sections. The first section is organized as three rows of eight contacts. The second section contains five signals that are designed specifically for analog video signals. Hoizontal Sync, Vertical Sync, R, G, and B are all required for analog implementations.

A fused (1A anti-surge) +5V supply (pin 14) is provided by the PB11MPCore.

The combined DVI connector is shown in Figure A-13.

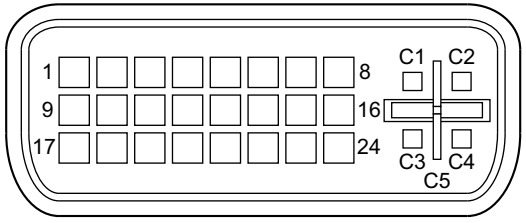


Figure A-13 DVI connector

The DVI connector signals are listed in Table A-9.

Table A-9 DVI connector signals

Pin	Signal	Pin	Signal	Pin	Signal
1	T.M.D.S. Data2-	9	T.M.D.S. Data1-	17	T.M.D.S. Data0-
2	T.M.D.S. Data2+	10	T.M.D.S. Data1+	18	T.M.D.S. Data0+
3	T.M.D.S. Data2/4 Shield	11	T.M.D.S. Data1/3 Shield	19	T.M.D.S. Data0/5 Shield
4	T.M.D.S. Data4-	12	T.M.D.S. Data3-	20	T.M.D.S. Data5-
5	T.M.D.S. Data4+	13	T.M.D.S. Data3+	21	T.M.D.S. Data5+
6	DDC Clock	14	+5V Power	22	T.M.D.S. Clock Shield
7	DDC Data	15	Ground (for +5V)	23	T.M.D.S. Clock+
8	Analog Vertical sync	16	Hot Plug Detect	24	T.M.D.S. Clock-
C1	Analog Red	C2	Analog Green	C3	Analog Blue
C4	Analog Horizontal Sync	C5	Analog Ground (analog R, G, and B return)		

Note

The PB11MPCore implements a single link, T.M.D.S. Data 3, 4, and 5 connections are not used.

A.12 RealView Logic Tile header connectors

These headers allow the connection of a RealView Logic Tile to the PB11MPCore. Figure A-14 shows the pin numbers and power-blade usage of the HDRX, HDRY, and HDRZ headers.

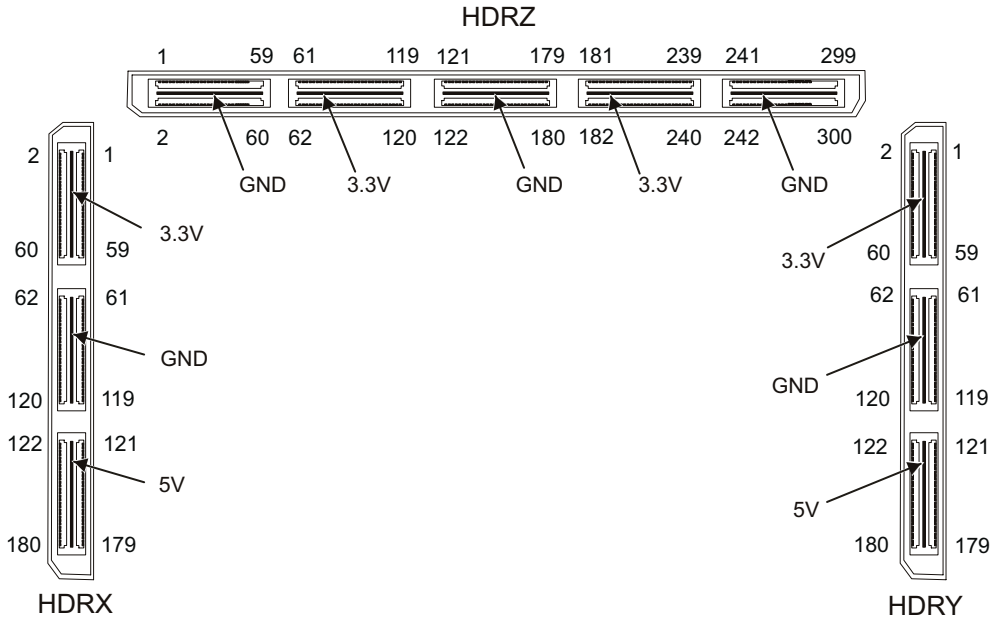


Figure A-14 HDRX, HDRY, and HDRZ pin numbering

Warning

The I/O voltage on a RealView Logic Tile (**VCCO1** and **VCC02**) can be changed by removing resistors on the tile and supplying the I/O voltage from either the tile above or the tile below in a tile stack. However, all signals from the PB11MPCore to a RealView Logic Tile use 3.3V I/O levels and all signals from a RealView Logic Tile to the PB11MPCore must use 3.3V I/O levels.

The 5V supply on the headers is to power voltage converters that might be present on the Logic Tile.

Tables *HDRX signals* on page A-20, *HDRY signals* on page A-27, and *HDRZ signals* on page A-34 list the signals on each header pin.

Note

The designation used for a multiplexed signal is X / Y, where signal X is present when **CLKOUTDIV** is HIGH and signal Y is present when **CLKOUTDIV** is LOW. See *Application Note AN151* for details of the AXI multiplexing scheme.

A.12.1 HDRX signals

Table A-10 describes the signals on the HDRX header pins. For a description of the signals refer to *AMBA 3 AXI Protocol* (ARM IHI 0022).

Table A-10 HDRX signals

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
ARADDR12 / ARADDR28	**X89**	2	1	**X90**	**ARADDR13 / ARADDR29**
ARADDR11 / ARADDR27	**X88**	4	3	**X91**	**ARADDR14 / ARADDR30**
ARADDR10 / ARADDR26	**X87**	6	5	**X92**	**ARADDR15 / ARADDR31**
ARADDR9 / ARADDR25	**X86**	8	7	**X93**	**ARID0 / ARID2**
ARADDR8 / ARADDR24	**X85**	10	9	**X94**	**ARID1 / ARID3**
ARADDR7 / ARADDR23	**X84**	12	11	**X95**	**ARLEN0 / ARLEN2**
ARADDR6 / ARADDR22	**X83**	14	13	**X96**	**ARLEN1 / ARLEN3**
ARADDR5 / ARADDR21	**X82**	16	15	**X97**	**ARSIZE0 / ARSIZE1**
ARADDR4 / ARADDR20	**X81**	18	17	**X98**	**ARID4 / ARPROT2**
ARADDR3 / ARADDR19	**X80**	20	19	**X99**	**ARPROT0 / ARPROT1**
ARADDR2 / ARADDR18	**X79**	22	21	**X100**	**ARBURST0 / ARBURST1**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
ARADDR1 / ARADDR17	**X78**	24	23	**X101**	**ARLOCK0 / ARLOCK1**
ARADDR0 / ARADDR16	**X77**	26	25	**X102**	**ARCACHE0 / ARCACHE2**
BREADY	**X76**	28	27	**X103**	**ARCACHE1 / ARCACHE3**
BVALID	**X75**	30	29	**X104**	**ARVALID/ ARID5**
BRESP0 / BRESP1	**X74**	32	31	**X105**	**ARREADY**
BID4 / BID5	**X73**	34	33	**X106**	**RDATA0 / RDATA32**
BID1 / BID3	**X72**	36	35	**X107**	**RDATA1 / RDATA33**
BID0 / BID2	**X71**	38	37	**X108**	**RDATA2 / RDATA34**
AWREADY	**X70**	40	39	**X109**	**RDATA3 / RDATA35**
AWVALID / AWID5	**X69**	42	41	**X110**	**RDATA4 / RDATA36**
AWCACHE1 / AWCACHE3	**X68**	44	43	**X111**	**RDATA5 / RDATA37**
AWCACHE0 / AWCACHE2	**X67**	46	45	**X112**	**RDATA6 / RDATA38**
AWLOCK0 / AWLOCK1	**X66**	48	47	**X113**	**RDATA7 / RDATA39**
AWBURST0 / AWBURST1	**X65**	50	49	**X114**	**RDATA8 / RDATA40**
AWPROT0 / AWPROT1	**X64**	52	51	**X115**	**RDATA9 / RDATA41**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
nRST_X	**X63**	54	53	**X116**	**RDATA10 / RDATA42**
AWID4 / AWPROT2	**X62**	56	55	**X117**	**RDATA11 / RDATA43**
AWSIZE0 / AWSIZE1	**X61**	58	57	**X118**	**RDATA12 / RDATA44**
AWLEN1 / AWLEN3	**X60**	60	59	**X119**	**RDATA13 / RDATA45**
AWLEN0 / AWLEN2	**X59**	62	61	**X120**	**RDATA14 / RDATA46**
AWID1 / AWID3	**X58**	64	63	**X121**	**RDATA15 / RDATA47**
AWID0 / AWID2	**X57**	66	65	**X122**	**RDATA16 / RDATA48**
AWADDR15 / AWADDR31	**X56**	68	67	**X123**	**RDATA17 / RDATA49**
AWADDR14 / AWADDR30	**X55**	70	69	**X124**	**RDATA18 / RDATA50**
AWADDR13 / AWADDR29	**X54**	72	71	**X125**	**RDATA19 / RDATA51**
AWADDR12 / AWADDR28	**X53**	74	73	**X126**	**RDATA20 / RDATA52**
AWADDR11 / AWADDR27	**X52**	76	75	**X127**	**RDATA21 / RDATA53**
AWADDR10 / AWADDR26	**X51**	78	77	**X128**	**RDATA22 / RDATA54**
AWADDR9 / AWADDR25	**X50**	80	79	**X129**	**RDATA23 / RDATA55**
AWADDR8 / AWADDR24	**X49**	82	81	**X130**	**RDATA24 / RDATA56**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
AWADDR7 / AWADDR23	**X48**	84	83	**X131**	**RDATA25 / RDATA57**
AWADDR6 / AWADDR22	**X47**	86	85	**X132**	**RDATA26 / RDATA58**
AWADDR5 / AWADDR21	**X46**	88	87	**X133**	**RDATA27 / RDATA59**
AWADDR4 / AWADDR20	**X45**	90	89	**X134**	**RDATA28 / RDATA60**
AWADDR3 / AWADDR19	**X44**	92	91	**X135**	**RDATA29 / RDATA61**
AWADDR2 / AWADDR18	**X43**	94	93	**X136**	**RDATA30 / RDATA62**
AWADDR1 / AWADDR17	**X42**	96	95	**X137**	**RDATA31 / RDATA63**
AWADDR0 / AWADDR16	**X41**	98	97	**X138**	**RID0 / RID2**
WREADY	**X40**	100	99	**X139**	**RID1 / RID3**
WVALID / WID5	**X39**	102	101	**X140**	**RRESP0 / RRESP1**
WLAST / WID4	**X38**	104	103	**X141**	**RLAST / RID4**
WSTRB3 / WSTRB7	**X37**	106	105	**X142**	**RVALID / RID5**
WSTRB2 / WSTRB6	**X36**	108	107	**X143**	**RREADY**
WSTRB1 / WSTRB5	**X35**	110	109	**X144**	**X144**
WSTRB0 / WSTRB4	**X34**	112	111	**X145**	**X145**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
WID1 / WID3	**X33**	114	113	**X146**	**X146**
WID0 / WID2	**X32**	116	115	**X147**	**X147**
WDATA31 / WDATA63	**X31**	118	117	**X148**	**X148**
WDATA30 / WDATA62	**X30**	120	119	**X149**	**X149**
WDATA29 / WDATA61	**X29**	122	121	**X150**	**X150**
WDATA28 / WDATA60	**X28**	124	123	**X151**	**X151**
WDATA27 / WDATA59	**X27**	126	125	**X152**	**X152**
WDATA26 / WDATA58	**X26**	128	127	**X153**	**X153**
WDATA25 / WDATA57	**X25**	130	129	**X154**	**X154**
WDATA24 / WDATA56	**X24**	132	131	**X155**	**X155**
WDATA23 / WDATA55	**X23**	134	133	**X156**	**X156**
WDATA22 / WDATA54	**X22**	136	135	**X157**	**X157**
WDATA21 / WDATA53	**X21**	138	137	**X158**	**X158**
WDATA20 / WDATA52	**X20**	140	139	**X159**	**X159**
WDATA19 / WDATA51	**X19**	142	141	**X160**	**X160**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
WDATA18 / WDATA50	**X18**	144	143	**X161**	**X161**
WDATA17 / WDATA49	**X17**	146	145	**X162**	**X162**
WDATA16 / WDATA48	**X16**	148	147	**X163**	**X163**
WDATA15 / WDATA47	**X15**	150	149	**X164**	**X164**
WDATA14 / WDATA46	**X14**	152	151	**X165**	**X165**
WDATA13 / WDATA45	**X13**	154	153	**X166**	**X166**
WDATA12 / WDATA44	**X12**	156	155	**X167**	**X167**
WDATA11 / WDATA43	**X11**	158	157	**X168**	**X168**
WDATA10 / WDATA42	**X10**	160	159	**X169**	**X169**
WDATA9 / WDATA41	**X9**	162	161	**X170**	**X170**
WDATA8 / WDATA40	**X8**	164	163	**X171**	**X171**
WDATA7 / WDATA39	**X7**	166	165	**X172**	**X172**
WDATA6 / WDATA38	**X6**	168	167	**X173**	**X173**
WDATA5 / WDATA37	**X5**	170	169	**X174**	**X174**
WDATA4 / WDATA36	**X4**	172	171	**X175**	**X175**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
WDATA3 / WDATA35	**X3**	174	173	**X176**	**X176**
WDATA2 / WDATA34	**X2**	176	175	**X177**	**X177**
WDATA1 / WDATA33	**X1**	178	177	**X178**	**X178**
WDATA0 / WDATA32	**X0**	180	179	**X179**	**X179**

A.12.2 HDRY signals

Table A-11 lists the signals on the HDRY header pins. For a description of the signals refer to *AMBA 3 AXI Protocol* (ARM IHI 0022).

Table A-11 HDRY signals

Baseboard signals	Y Bus	Even pins	Odd pins	Baseboard signals	Y Bus
ARADDR13 / ARADDR29	**Y90**	2	1	**Y89**	**ARADDR12 / ARADDR28**
ARADDR14 / ARADDR30	**Y91**	4	3	**Y88**	**ARADDR11 / ARADDR27**
ARADDR15 / ARADDR31	**Y92**	6	5	**Y87**	**ARADDR10 / ARADDR26**
ARID0 / ARID2	**Y93**	8	7	**Y86**	**ARADDR9 / ARADDR25**
ARID1 / ARID3	**Y94**	10	9	**Y85**	**ARADDR8 / ARADDR24**
ARLEN0 / ARLEN2	**Y95**	12	11	**Y84**	**ARADDR7 / ARADDR23**
ARLEN1 / ARLEN3	**Y96**	14	13	**Y83**	**ARADDR6 / ARADDR22**
ARSIZE0 / ARSIZE1	**Y97**	16	15	**Y82**	**ARADDR5 / ARADDR21**
ARID4 / ARPROT2	**Y98**	18	17	**Y81**	**ARADDR4 / ARADDR20**
ARPROT0 / ARPROT1	**Y99**	20	19	**Y80**	**ARADDR3 / ARADDR19**
ARBURST0 / ARBURST1	**Y100**	22	21	**Y79**	**ARADDR2 / ARADDR18**
ARLOCK0 / ARLOCK1	**Y101**	24	23	**Y78**	**ARADDR1 / ARADDR17**
ARCACHE0 / ARCACHE2	**Y102**	26	25	**Y77**	**ARADDR0 / ARADDR16**
ARCACHE1 / ARCACHE3	**Y103**	28	27	**Y76**	**BREADY**

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Baseboard signals	Y Bus
ARVALID/b0[a]	**Y104**	30	29	**Y75**	**BVALID**
ARREADY	**Y105**	32	31	**Y74**	**BRESP0 / BRESP1**
RDATA0 / RDATA32	**Y106**	34	33	**Y73**	**BID4 /** (not connected)
RDATA1 / RDATA33	**Y107**	36	35	**Y72**	**BID1 / BID3**
RDATA2 / RDATA34	**Y108**	38	37	**Y71**	**BID0 / BID2**
RDATA3 / RDATA35	**Y109**	40	39	**Y70**	**AWREADY**
RDATA4 / RDATA36	**Y110**	42	41	**Y69**	**AWVALID /**b0[b]
RDATA5 / RDATA37	**Y111**	44	43	**Y68**	**AWCACHE1 / AWCACHE3**
RDATA6 / RDATA38	**Y112**	46	45	**Y67**	**AWCACHE0 / AWCACHE2**
RDATA7 / RDATA39	**Y113**	48	47	**Y66**	**AWLOCK0 / AWLOCK1**
RDATA8 / RDATA40	**Y114**	50	49	**Y65**	**AWBURST0 / AWBURST1**
RDATA9 / RDATA41	**Y115**	52	51	**Y64**	**AWPROT0 / AWPROT1**
RDATA10 / RDATA42	**Y116**	54	53	**Y63**	**ARM_nRESET**
RDATA11 / RDATA43	**Y117**	56	55	**Y62**	**AWID4 / AWPROT2**
RDATA12 / RDATA44	**Y118**	58	57	**Y61**	**AWSIZE0 / AWSIZE1**
RDATA13 / RDATA45	**Y119**	60	59	**Y60**	**AWLEN1 / AWLEN3**

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Baseboard signals	Y Bus
RDATA14 / RDATA46	**Y120**	62	61	**Y59**	**AWLEN0 / AWLEN2**
RDATA15 / RDATA47	**Y121**	64	63	**Y58**	**AWID1 / AWID3**
RDATA16 / RDATA48	**Y122**	66	65	**Y57**	**AWID0 / AWID2**
RDATA17 / RDATA49	**Y123**	68	67	**Y56**	**AWADDR15 / AWADDR31**
RDATA18 / RDATA50	**Y124**	70	69	**Y55**	**AWADDR14 / AWADDR30**
RDATA19 / RDATA51	**Y125**	72	71	**Y54**	**AWADDR13 / AWADDR29**
RDATA20 / RDATA52	**Y126**	74	73	**Y53**	**AWADDR12 / AWADDR28**
RDATA21 / RDATA53	**Y127**	76	75	**Y52**	**AWADDR11 / AWADDR27**
RDATA22 / RDATA54	**Y128**	78	77	**Y51**	**AWADDR10 / AWADDR26**
RDATA23 / RDATA55	**Y129**	80	79	**Y50**	**AWADDR9 / AWADDR25**
RDATA24 / RDATA56	**Y130**	82	81	**Y49**	**AWADDR8 / AWADDR24**
RDATA25 / RDATA57	**Y131**	84	83	**Y48**	**AWADDR7 / AWADDR23**
RDATA26 / RDATA58	**Y132**	86	85	**Y47**	**AWADDR6 / AWADDR22**
RDATA27 / RDATA59	**Y133**	88	87	**Y46**	**AWADDR5 / AWADDR21**
RDATA28 / RDATA60	**Y134**	90	89	**Y45**	**AWADDR4 / AWADDR20**

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Baseboard signals	Y Bus
RDATA29 / RDATA61	**Y135**	92	91	**Y44**	**AWADDR3 / AWADDR19**
RDATA30 / RDATA62	**Y136**	94	93	**Y43**	**AWADDR2 / AWADDR18**
RDATA31 / RDATA63	**Y137**	96	95	**Y42**	**AWADDR1 / AWADDR17**
RID0 / RID2	**Y138**	98	97	**Y41**	**AWADDR0 / AWADDR16**
RID1 / RID3	**Y139**	100	99	**Y40**	**WREADY**
RRESP0 / RRESP1	**Y140**	102	101	**Y39**	**WVALID** /b0[c]
RLAST / RID4	**Y141**	104	103	**Y38**	**WLAST / WID4**
RVALID / (not connected)	**Y142**	106	105	**Y37**	**WSTRB3 / WSTRB7**
RREADY	**Y143**	108	107	**Y36**	**WSTRB2 / WSTRB6**
Y144	**Y144**	110	109	**Y35**	**WSTRB1 / WSTRB5**
Y145	**Y145**	112	111	**Y34**	**WSTRB0 / WSTRB4**
Y146	**Y146**	114	113	**Y33**	**WID1 / WID3**
Y147	**Y147**	116	115	**Y32**	**WID0 / WID2**
Y148	**Y148**	118	117	**Y31**	**WDATA31 / WDATA63**
Y149	**Y149**	120	119	**Y30**	**WDATA30 / WDATA62**

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Baseboard signals	Y Bus
Y150	**Y150**	122	121	**Y29**	**WDATA29 / WDATA61**
Y151	**Y151**	124	123	**Y28**	**WDATA28 / WDATA60**
Y152	**Y152**	126	125	**Y27**	**WDATA27 / WDATA59**
Y153	**Y153**	128	127	**Y26**	**WDATA26 / WDATA58**
Y154	**Y154**	130	129	**Y25**	**WDATA25 / WDATA57**
Y155	**Y155**	132	131	**Y24**	**WDATA24 / WDATA56**
Y156	**Y156**	134	133	**Y23**	**WDATA23 / WDATA55**
Y157	**Y157**	136	135	**Y22**	**WDATA22 / WDATA54**
Y158	**Y158**	138	137	**Y21**	**WDATA21 / WDATA53**
Y159	**Y159**	140	139	**Y20**	**WDATA20 / WDATA52**
Y160	**Y160**	142	141	**Y19**	**WDATA19 / WDATA51**
Y161	**Y161**	144	143	**Y18**	**WDATA18 / WDATA50**
Y162	**Y162**	146	145	**Y17**	**WDATA17 / WDATA49**
Y163	**Y163**	148	147	**Y16**	**WDATA16 / WDATA48**
Y164	**Y164**	150	149	**Y15**	**WDATA15 / WDATA47**

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Baseboard signals	Y Bus
Y165	**Y165**	152	151	**Y14**	**WDATA14 / WDATA46**
Y166	**Y166**	154	153	**Y13**	**WDATA13 / WDATA45**
Y167	**Y167**	156	155	**Y12**	**WDATA12 / WDATA44**
Y168	**Y168**	158	157	**Y11**	**WDATA11 / WDATA43**
Y169	**Y169**	160	159	**Y10**	**WDATA10 / WDATA42**
Y160	**Y170**	162	161	**Y9**	**WDATA9 / WDATA41**
Y171	**Y171**	164	163	**Y8**	**WDATA8 / WDATA40**
Y172	**Y172**	166	165	**Y7**	**WDATA7 / WDATA39**
Y173	**Y173**	168	167	**Y6**	**WDATA6 / WDATA38**
Y174	**Y174**	170	169	**Y5**	**WDATA5 / WDATA37**
Y175	**Y175**	172	171	**Y4**	**WDATA4 / WDATA36**
Y176	**Y176**	174	173	**Y3**	**WDATA3 / WDATA35**
Y177	**Y177**	176	175	**Y2**	**WDATA2 / WDATA34**
Y178	**Y178**	178	177	**Y1**	**WDATA1 / WDATA33**
Y179	**Y179**	180	179	**Y0**	**WDATA0 / WDATA32**

a. These are AXI **ARVALID/ARID5** signals. b0 indicates that for the **ARID5** phase of the controlling clock signal, **ARID5** is always set to logic level zero.
b. These are AXI **AWVALID/AWID5** signals. b0 indicates that for the **AWID5** phase of the controlling clock signal, **AWID5** is always set to logic level zero.
c. These are AXI **WVALID/WID5** signals, b0 indicates that for the WID5 phase of the controlling clock signal, **AWID5** is always set to logic level zero.

A.12.3 HDRZ

Table A-12 lists the signals on the HDRZ header pins. Refer to the user guide for the fitted Logic Tile for the Logic Tile specific HDRZ upper (U) and HDRZ lower (L) signal listing.

Table A-12 HDRZ signals

Baseboard signals	Even pins	Odd pins	Baseboard signals
Z255	2	1	**Z128**
Z254	4	3	**Z129**
Z253	6	5	**Z130**
Z252	8	7	**Z131**
Z251	10	9	**Z132**
Z250	12	11	**Z133**
Z249	14	13	**Z134**
Z248	16	15	**Z135**
Z247	18	17	**Z136**
Z246	20	19	**Z137**
Z245	22	21	**Z138**
Z244	24	23	**Z139**
Z243	26	25	**Z140**
Z242	28	27	**Z141**
Z241	30	29	**Z142**
Z240	32	31	**Z143**
Z249	34	33	**Z144**
Z248	36	35	**Z145**
Z237	38	37	**Z146**
Z236	40	39	**Z147**
Z235	42	41	**Z148**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
Z234	44	43	**Z149**
MBTYPE	46	45	**Z150**
MBTYPE	48	47	**Z151**
Z231	50	49	**Z152**
Z230	52	51	**Z153**
Z229	54	53	**Z154**
Z228	56	55	**Z155**
Z227	58	57	**Z156**
Z226	60	59	**Z157**
Z225	62	61	**Z158**
Z224	64	63	**Z159**
Z223	66	65	**Z160**
Z222	68	67	**Z161**
Z221	70	69	**Z162**
Z220	72	71	**Z163**
Z219	74	73	**Z164**
DMACCLR[1]	76	75	**Z165**
Z217	78	77	**Z166**
Z216	80	79	**Z167**
DMACBREQ[1]	82	81	**Z168**
DMACSREQ[1]	84	83	**Z169**
Z213	86	85	**Z170**
DMACCLR[0]	88	87	**Z171**
Z211	90	89	**Z172**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
Z210	92	91	**Z173**
DMACBREQ[0]	94	93	**Z174**
DMACSREQ[0]	96	95	**Z175**
INT[7]	98	97	**Z176**
INT[6]	100	99	**Z177**
INT[5]	102	101	**Z178**
INT[4]	104	103	**Z179**
INT[3]	106	105	**Z180**
INT[2]	108	107	**Z181**
INT[1]	110	109	**Z182**
INT[0]	112	111	**Z183**
Z199	114	113	**Z184**
Z198	116	115	**Z185**
Z197	118	117	**Z186**
Z196	120	119	**Z187**
Z195	122	121	**Z188**
Z194	124	123	**Z189**
Z193	126	125	**Z190**
Z192	128	127	**Z191**
CLK_POS_DN_IN	130	129	**D_nSRST**
CLK_NEG_DN_IN	132	131	**D_nTRST**
CLK_POS_UP_OUT	134	133	**D_TDO_IN**
CLK_NEG_UP	136	135	**D_TDI**
GND (**CLK_UP_THRU**)	138	137	**D_TCK_OUT**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
CLK_OUT_PLUS1	140	139	**D_TMS_OUT**
CLK_OUT_PLUS2	142	141	**D_RTCK**
CLK_IN_PLUS2	144	143	**C_nSRST**
CLK_IN_PLUS1	146	145	**C_nTRST**
Z182 (**CLK_DN_THRU**)	148	147	**C_TDO_IN**
CLK_GLOBAL	150	149	**C_TDI**
FPGA_IMAGE	152	151	**C_TCK_OUT**
nSYSPOR	154	153	**C_TMS_OUT**
nSYSRST	156	155	**nTILE_DET**
nRTCKEN	158	157	**nCFGEN**
SPARE12 (reserved)	160	159	**GLOBAL_DONE**
SPARE10 (reserved)	162	161	**SPARE11** (reserved)
SPARE8 (reserved)	164	163	**SPARE9** (reserved)
SPARE6 (reserved)	166	165	**SPARE7** (reserved)
SPARE4 (reserved)	168	167	**SPARE5** (reserved)
SPARE2 (reserved)	170	169	**SPARE3** (reserved)
SPARE0 (reserved)	172	171	**SPARE1** (reserved)
Z64	174	173	**Z63**
Z65	176	175	**Z62**
Z66	178	177	**Z61**
Z67	180	179	**Z60**
Z68	182	181	**Z59**
Z79	184	183	**Z58**
Z70	186	185	**Z57**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
Z71	188	187	**Z56**
Z72	190	189	**Z55**
Z73	192	191	**Z54**
Z74	194	193	**Z53**
Z75	196	195	**Z52**
Z76	198	197	**Z51**
Z77	200	199	**Z50**
Z78	202	201	**Z49**
Z79	204	203	**Z48**
Z80	206	205	**Z47**
Z81	208	207	**Z46**
Z82	210	209	**Z45**
Z83	212	211	**Z44**
Z84	214	213	**Z43**
Z85	216	215	**Z42**
Z86	218	217	**Z41**
Z87	220	219	**Z40**
Z88	222	221	**UART3RXD**
Z89	224	223	**nUART3CTS**
Z90	226	225	**UART3TXD**
Z91	228	227	**nUART3RTS**
Z92	230	229	**UART2RXD**
Z93	232	231	**nUART2CTS**
Z94	234	233	**UART2TXD**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
Z95	236	235	**nUART2RTS**
Z96	238	237	**CLCP**
Z97	240	239	**Z30**
Z98	242	241	**CLPOWER**
Z99	244	243	**CLLP**
Z100	246	245	**Z27**
Z101	248	247	**CLFP**
Z102	250	249	**CLCP**
Z103	252	251	**CLAC**
Z104	254	253	**CLD23**
Z105	256	255	**CLD22**
Z106	258	257	**CLD21**
Z107	260	259	**CLD20**
Z108	262	261	**CLD19**
Z109	264	263	**CLD18**
Z110	266	265	**CLD17**
Z111	268	267	**CLD16**
Z112	270	269	**CLD15**
Z113	272	271	**CLD14**
Z114	274	273	**CLD13**
Z115	276	275	**CLD12**
Z116	278	277	**CLD11**
Z117	280	279	**CLD10**
Z118	282	281	**CLD9**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
Z119	284	283	**CLD8**
Z120	286	285	**CLD7**
Z121	288	287	**CLD6**
Z122	290	289	**CLD5**
Z123	292	291	**CLD4**
Z124	294	293	**CLD3**
Z125	296	295	**CLD2**
Z126	298	297	**CLD1**
Z127	300	299	**CLD0**

A.13 Test and debug connections

The PB11MPCore provides connectors to aid diagnostics.

This section contains the following subsections:

- *JTAG*
- *USB config port* on page A-42
- *Integrated Logic Analyzer (ILA)* on page A-42.

A.13.1 JTAG

Figure A-15 shows the pinout of the JTAG connector J10 that is located on the rear panel of the ATX enclosure. All JTAG active HIGH input signals have pull-up resistors.

See *Rear panel layout* on page 3-5 to locate the connector and see *Test, configuration, and debug interfaces* on page 3-54 for a description of how the JTAG config and debug interfaces are implemented on the PB11MPCore.

Note

The term JTAG equipment refers to any hardware that can drive the JTAG signals to devices in the scan chain. Typically this is RealView ICE, although hardware from other suppliers can also be used to debug ARM processors.

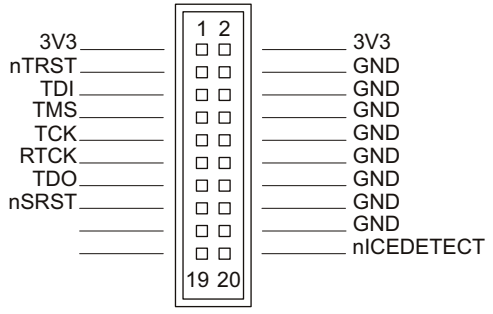


Figure A-15 JTAG connector

A.13.2 USB config port

Figure A-16 shows the signals on the USB config connector J12. **USBDP** and **USBDM** are the positive and negative USB data signals.

See *Front panel layout* on page 3-4 to locate the connector and see *Test, configuration, and debug interfaces* on page 3-54 for a description of how the USB debug interface is implemented on the PB11MPCore.

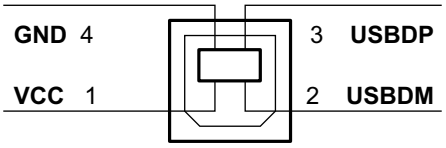


Figure A-16 USB debug connector

A.13.3 Integrated Logic Analyzer (ILA)

Figure A-17 shows the signals on the ILA connector J9. You may use an ILA and JTAG to debug FPGA designs and software at the same time.

See *Baseboard layout* on page 3-2 to locate the connector, and for more information, see the documentation supplied with your analyzer. (The ChipScope product is described on the Xilinx web site at `www.xilinx.com`.)

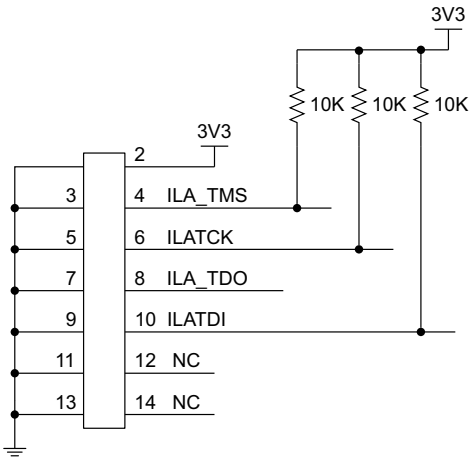


Figure A-17 Integrated Logic Analyzer (ILA) connector

Appendix B
Specifications

This appendix contains the specification for the baseboard. It contains the following sections:

- *Electrical Specification* on page B-2
- *Timing specifications* on page B-3.

B.1 Electrical Specification

This section provides details of the voltage and current characteristics for the baseboard.

B.1.1 Bus interface characteristics

Table B-1 lists the baseboard electrical characteristics for normal operation.

Table B-1 Baseboard electrical characteristics

Symbol	Description	Min.	Max.	Unit
3V3	3V3 from power connector J39 or tile site interface (VIO)	3.1	3.5	V
5V	5V from power connector J39	4.75	5.25	V
12V	12V from power connector J39	11.4	12.6	V
-12V	-12V from power connector J39	-11.4	-12.6	V
V_{IH}	High-level input voltage at tile site interface	2.0	3.6	V
V_{IL}	Low-level input voltage at tile site interface	0	0.8	V
V_{OH}	High-level output voltage at tile site interface	2.4	–	V
V_{OL}	Low-level output voltage at tile site interface	–	0.4	V
C_{IN}	Capacitance on any pin	–	20	pF

B.2 Timing specifications

This section provides details of the baseboard maximum clock frequencies and the tile site AXI bus timings when used stand-alone or with a Logic Tile fitted.

B.2.1 Clock frequency restrictions

The maximum tile site clock (**TSCLK**) frequency that can be used for reliable operation depends on the type of Logic Tile fitted. The default frequency setting is 25MHz and assumes that a LTXC2V8000 is to be used.

Caution

The ICS307 programmable oscillators OSC0, OSC1, OSC2, OSC3, OSC4, OSC5 and OSC6 can be programmed to deliver very high frequency clock signals (200MHZ). The settings for VCO divider, output divider, and output select values are interrelated and must be set correctly. Some combinations of settings do not result in stable operation. For more information on the ICS clock generator and a frequency calculator, see the IDT web site: `www.idt.com`.

B.2.2 AXI bus timings

Table B-2 lists the tile site multiplexed AXI bus timings.

Table B-2 AC Specifications

Parameter	Symbol	Min	Max	Units	Notes
Clock Cycle	**tTScyc**	30	–	ns	Cmax=15pF
Output valid time after clock edge	**tTSov**	–	6.771	ns	
Output hold time after clock edge	**tTSoh**	0.881	–	ns	
Input setup time to clock edge	**tTSis**	–	4.223	ns	
Input hold time after clock edge	**tTSih**	0.975	–	ns	

Figure B-1 on page B-4 shows the tile site multiplexed AXI timing diagram.

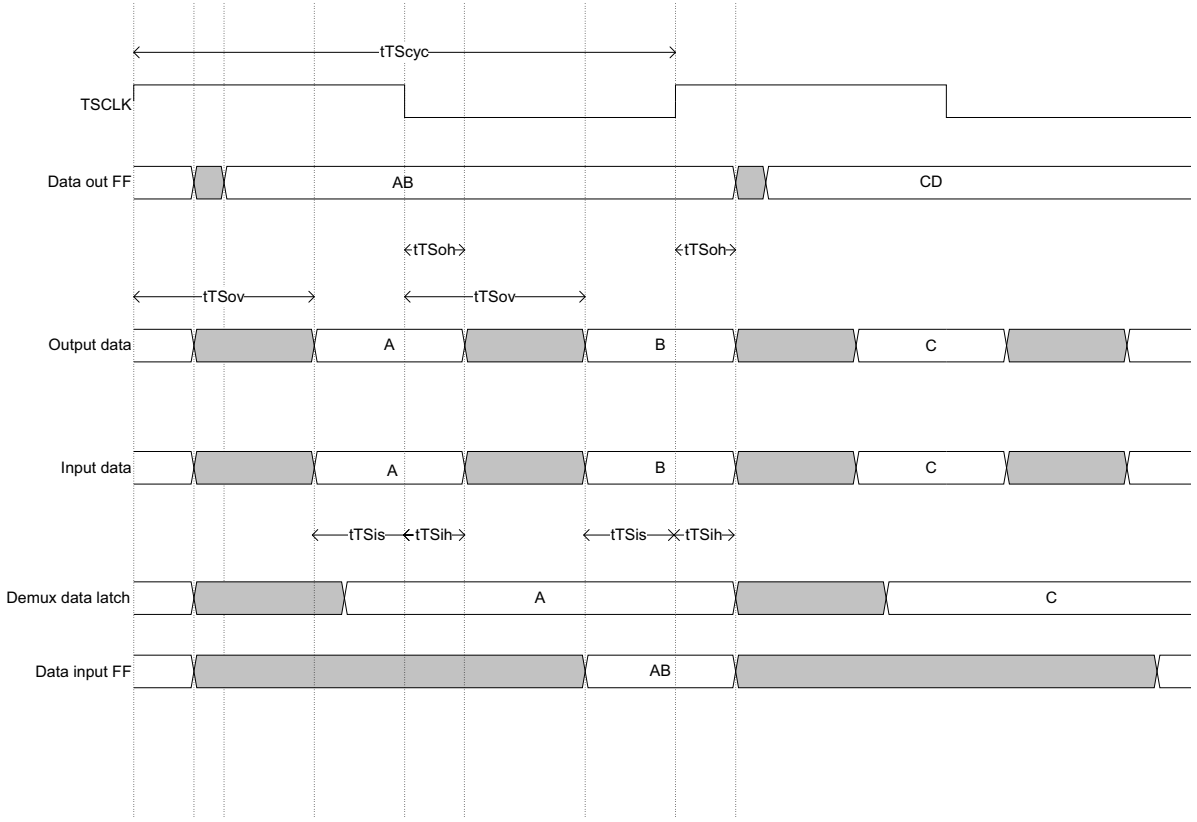


Figure B-1 Tile site multiplexed AXI timing

Appendix C
Memory Expansion Boards

This appendix describes expansion memory modules for the PB11MPCore. It contains the following sections:

- *About memory expansion* on page C-2
- *Fitting a memory board* on page C-4
- *Connector pinout* on page C-5.

C.1 About memory expansion

Only static memory expansion is supported by the PB11MPCore. You can stack static memory expansion boards on the PB11MPCore using the PISMO™ connector provided. There are five chip select signals available on the PISMO connector, each of these can select 64MB of SRAM. The distribution of the chip select signals is determined by the expansion memory boards themselves.

The block diagram for a typical static memory board is shown in Figure C-1.

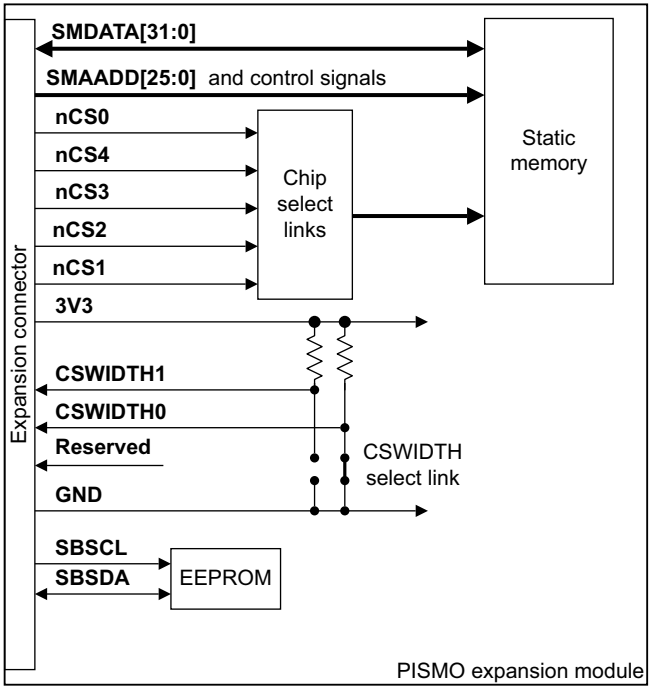


Figure C-1 Static memory board block diagram

Note

Figure C-1 on page C-2 is only a typical example of a PISMO static memory expansion board, different expansion boards may have different features.

See the documentation provided with your memory board for details on specific signals and link options.

The PB11MPCore supports static memory modules compliant to the *Platform Independent Storage MOdule* (PISMO) specification Version 1.0.
General information and specifications are available from the PISMO website: `www.pismoworld.org`.

C.1.1 Operation without expansion memory

You can operate the PB11MPCore without a memory expansion board connected because it has 8MB of (Pseudo) SRAM, 512MB of SDRAM, and 64MB of NOR flash permanently fitted.

C.1.2 Memory board configuration

The PISMO memory module includes a configuration EEPROM that can be read from the PB11MPCore to identify the type of memory on the board and how it is configured. This information can be used by the application or operating system to initialize the memory space. For details of the EEPROM data structure and information elements refer to the *PISMO Specification Version 1.0*.

C.2 Fitting a memory board

To install a memory expansion board:

1. Ensure that the PB11MPCore is powered down.
2. Align the memory expansion board with the PISMO connector on the PB11MPCore. See *Baseboard layout* on page 3-2 for the location of the PISMO connector.
3. Press the module into the connector.
4. Stack additional modules as required (up to five modules may be stacked) using the mating connectors on the PISMO memory modules.

C.3 Connector pinout

This section describes the connector present on the expansion memory board.

Caution

The pinout and naming in Table C-1 on page C-6 are valid for the *PISMO Version 1.0* specification.

The PB11MPCore memory expansion port is not compatible with the *PISMO2 Version 1.0* specification.

C.3.1 Expansion connector

The static memory expansion board uses a 120-way Samtec connector as shown in Figure C-2. The connector pinout is shown in Table C-1 on page C-6.

Note

The numbering of pins on the connector is for the connector as viewed from below.

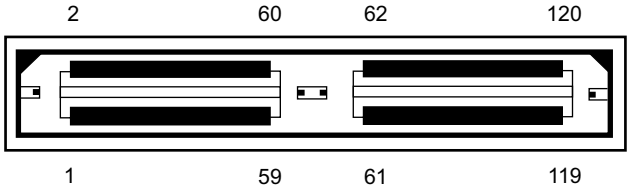


Figure C-2 Samtec 120-way connector

Table C-1 Static memory connector signals

Pin No.	Signal	Pin No.	Signal
1	**DATA[0]**	2	**3V3**
3	**DATA[1]**	4	**3V3**
5	**DATA[2]**	6	**3V3**
7	**DATA[3]**	8	**3V3**
9	**DATA[4]**	10	**VDDIO**[a]
11	**DATA[5]**	12	**VDDIO**[a]
13	**DATA[6]**	14	**VDDIO**[a]
15	**DATA[7]**	16	**VDDIO**[a]
17	**DATA[8]**	18	**1V8**
19	**DATA[9]**	20	**1V8**
21	**DATA[10]**	22	**1V8**
23	**DATA[11]**	24	**1V8**
25	**DATA[12]**	26	NC
27	**DATA[13]**	28	Reserved, not driven
29	**DATA[14]**	30	Reserved, not driven
31	**DATA[15]**	32	Reserved, not driven
33	**DATA[16]**	34	**5V**
35	**DATA[17]**	36	**5V**
37	**DATA[18]**	38	**5V**
39	**DATA[19]**	40	**5V**
41	**DATA[20]**	42	Reserved, not driven
43	**DATA[21]**	44	Reserved, not driven
45	**DATA[22]**	46	Reserved, not driven
47	**DATA[23]**	48	Reserved, not driven

Table C-1 Static memory connector signals (continued)

Pin No.	Signal	Pin No.	Signal
49	**DATA[24]**	50	Reserved, not driven
51	**DATA[25]**	52	Reserved, not driven
53	**DATA[26]**	54	Reserved, not driven
55	**DATA[27]**	56	Reserved, not driven
57	**DATA[28]**	58	Reserved, not driven
59	**DATA[29]**	60	Reserved, not driven
61	**DATA[30]**	62	**SBSCL**, E2PROM serial interface clock (3.3V signal level)
63	**DATA[31]**	64	**SBSDA**, E2PROM serial interface data (3.3V signal level)
65	**ADDR[0]**	66	**nRESET**
67	**ADDR[1]**	68	**nBOARDPOR**, asserted on hardware power cycle
69	**ADDR[2]**	70	**nFLWP**, flash write protect. Drive HIGH to write to flash.
71	**ADDR[3]**	72	**nEARLYRESET**, Reset signal. Differs from **nRESET** in that it is not delayed by **nWAIT**.
73	**ADDR[4]**	74	**nWAIT**, Wait mode input from external memory controller. Pulled HIGH if not used.
75	**ADDR[5]**	76	**nBURSTWAIT**, Synchronous burst wait input. This is used by the external device to delay a synchronous burst transfer if LOW. Pulled HIGH if not used.
77	**ADDR[6]**	78	**CANCELWAIT**, If HIGH, this signal enables the system to recover from an externally waited transfer that has taken longer than expected to finish. Pulled LOW if not used.

Table C-1 Static memory connector signals (continued)

Pin No.	Signal	Pin No.	Signal
79	**ADDR[7]**	80	**nCS[4]**
81	**ADDR[8]**	82	**nCS[3]**
83	**ADDR[9]**	84	**nCS[2]**
85	**ADDR[10]**	86	**nCS[1]**
87	**ADDR[11]**	88	Reserved, not driven
89	**ADDR[12]**	90	Reserved, not driven
91	**ADDR[13]**	92	Reserved, not driven
93	**ADDR[14]**	94	Reserved, not driven
95	**ADDR[15]**	96	**nCS[0]**
97	**ADDR[16]**	98	**nBUSY**, Indicates that memory is not ready to be released from reset. If LOW, this signal holds **nRESET** active.
99	**ADDR[17]**	100	**nIRQ**
101	**ADDR[18]**	102	**nWEN**
103	**ADDR[19]**	104	**nOEN**
105	**ADDR[20]**	106	**nBLS[3]**, Byte Lane Select for bits [31:24]
107	**ADDR[21]**	108	**nBLS[2]**, Byte Lane Select for bits [23:16]
109	**ADDR[22]**	110	**nBLS[1]**, Byte Lane Select for bits [15:8]
111	**ADDR[23]**	111	**nBLS[0]**, Byte Lane Select for bits [7:0]
113	**ADDR[24]**	114	**CSWIDTH[0]**, Indicates bus width for fitted part. Not routed through stackable boards.

Table C-1 Static memory connector signals (continued)

Pin No.	Signal	Pin No.	Signal
115	**ADDR[25]**	116	**CSWIDTH[1]**, Indicates bus width for fitted part. Not routed through stackable boards.
117	**ADDRVALID**, Indicates that the address output is stable during synchronous burst transfers	118	**CLK[1]**
119	**BAA**, Burst Address Advance. Used to advance the address count in the memory device	120	**CLK[0]**

a. VDDIO is the data voltage to host. This is not routed through stackable boards

Appendix D
RealView Logic Tile expansion

This appendix describes the signals present on the RealView Logic Tile expansion headers and give the steps required to install a RealView Logic Tile on the baseboard. It contains the following sections:

- *About the RealView Logic Tile* on page D-2
- *Header connectors* on page D-3.

D.1 About the RealView Logic Tile

RealView Logic Tiles, such as the LT-XC4VLX100+, enable developing AMBA 3 AXI, and AMBA APB peripherals, or custom logic, for use with ARM cores.

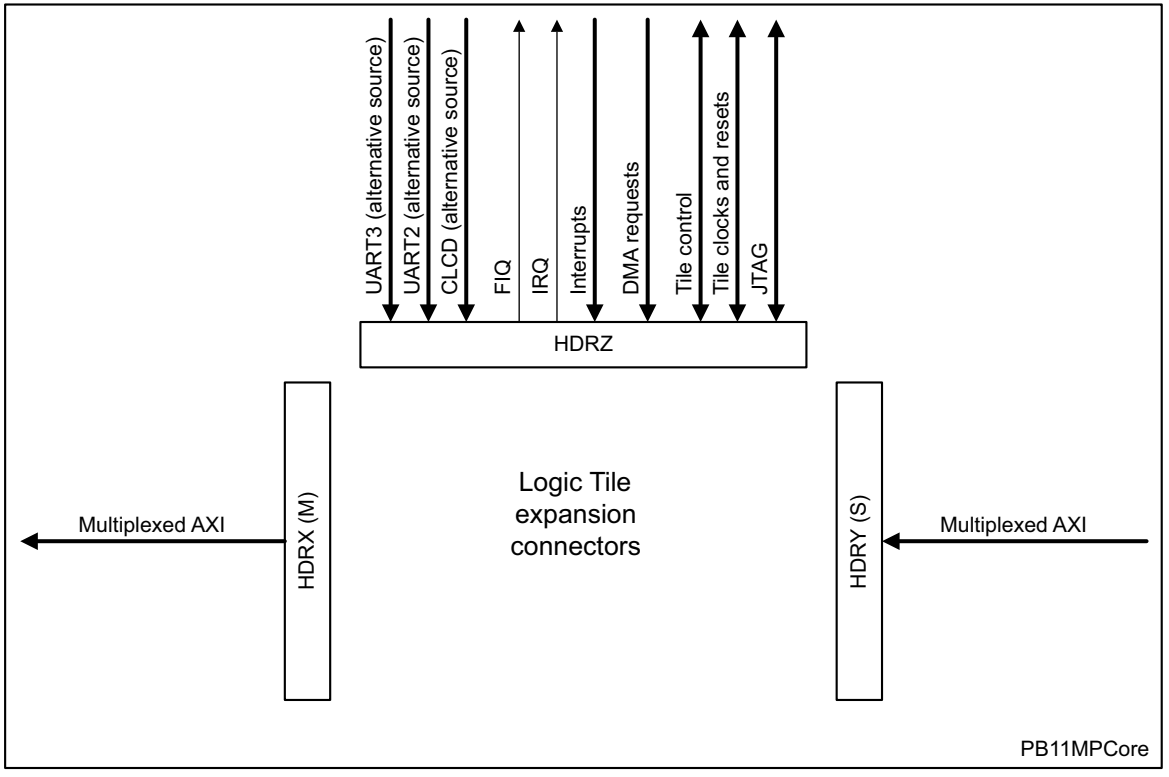


Figure D-1 Signals on the RealView Logic Tile expansion connectors

Note

If you connect a RealView Logic Tile, the design in the tile FPGA must implement logic to handle the Multiplexed AXI bus signals (see *AXI buses used by the Northbridge and RealView Logic Tiles* on page D-5).

D.2 Header connectors

This section gives a brief overview of the RealView Logic Tile header connectors and the associated signal groups. For information on a specific tile site interface, see the documentation for your RealView Logic Tile.

There are three headers on the top and bottom of the tile. The HDRX and HDRY headers are 180-way and the HDRZ connectors are 300-way.

Warning

There is a limit to the number of RealView Logic Tiles which can be stacked on a RealView baseboard. Please contact ARM support for the current recommended limits for the PB11MPCore.

When stacking tiles ensure that the power source can maintain the required voltage at the top tile when supplying maximum current to the system.

Caution

The FPGA signals on a RealView Logic Tile and the PB11MPCore are fully programmable. Ensure that there are no clashes between the signals on the tiles or with the signals from the PB11MPCore.

The FPGA can be damaged if several pins configured as outputs are connected together and attempt to output different logic levels.

Figure D-2 on page D-4 shows the pin numbers and power-blade usage of the HDRX, HDRY, and HDRZ headers on the upper side of the tile. See *RealView Logic Tile header connectors* on page A-19 for details of the signals on the PB11MPCore header connectors.

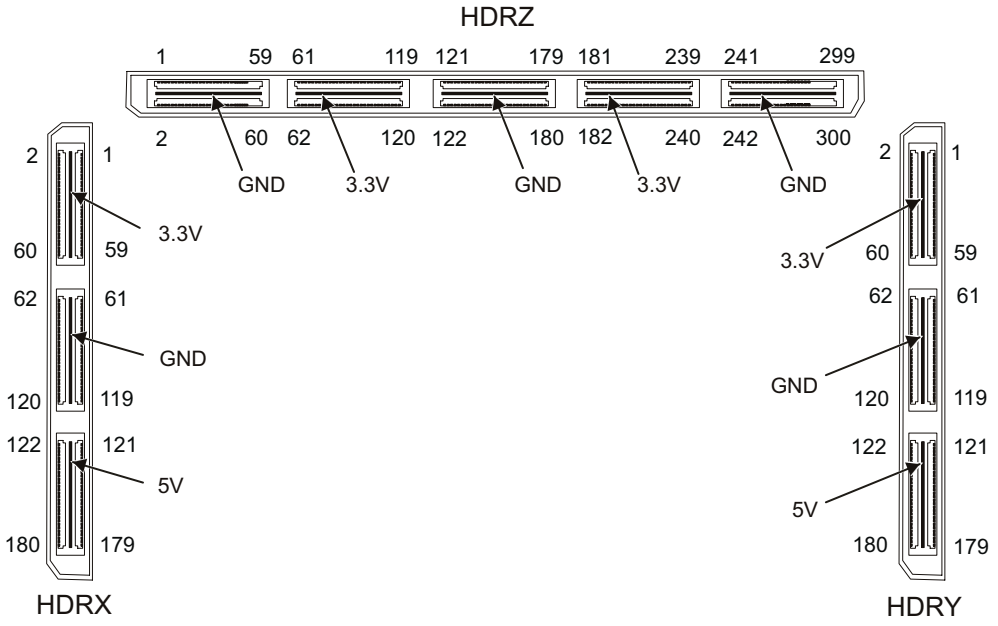


Figure D-2 HDRX, HDRY, and HDRZ (upper) pin numbering

D.2.1 Variable I/O levels

All HDRX, HDRY, and HDRZ connector signals on the PB11MPCore are fixed at 3.3V I/O signalling level.

Caution

The RealView Logic Tile mounted on the PB11MPCore must use the default 3.3V signal levels.

D.2.2 RealView Logic Tile clocks

For use with the PB11MPCore, the only tile clocks used are **CLK_IN_MINUS1** and **CLK_UP_THRU**. These are both driven by **OSCCLK2** from the baseboard.

D.2.3 JTAG

There is no JTAG connector on the RealView Logic Tile, you must use the JTAG debug or USB config connector on the ATX enclosure to debug and configure the Logic Tile.

If multiple RealView Logic Tiles are stacked on the baseboard, the signals are routed upwards to the top tile and then back down to the baseboard.

Use the JTAG interface to program the configuration flash in the RealView Logic Tile or to directly load the RealView Logic Tile FPGA image. For more information on the JTAG signals, see *Test, configuration, and debug interfaces* on page 3-54.

D.2.4 AXI buses used by the Northbridge and RealView Logic Tiles

Multiplexed AXI buses AXI M and AXI S are connected between the Northbridge and the RealView Logic Tile stack. The user-implemented system in the Logic Tile must co-operate with the system implemented within the Northbridge when using these buses:

AXI M The AXI M bus can only be connected to Mux AXI slaves in the Logic Tile stack.

AXI S The Mux AXI S bus can only be connected to Mux AXI masters in the Logic Tile stack.

AXI M

The Northbridge does not contain any slaves attached to the AXI M bus. The PB11MPCore memory map assigns the top 1GB of address space (`0xC0000000–0xFFFFFFFF`) to this bus, so a RealView Logic Tile can contain user-supplied slaves that occupy any of this space. The RealView Logic Tile FPGA must give a response to all transfers that are generated on the Mux AXI M bus, even those to addresses in the range `0x00000000–0xBFFFFFFF`. The Northbridge never generates these addresses on the AXI M bus. A separate tile master might, however, generate accesses to this region.

In a system without a fully-decoded address map, there can be addresses at which there are no slaves to respond to a transaction. In such a system, the tile site must provide a suitable error response to flag the access as illegal and also to prevent the system from locking up by trying to access a nonexistent slave.

When the tile site cannot successfully decode a slave access, it must route the access to a default slave that returns the **DECERR** response.

An implementation option is to have the default slave also record the details of decode errors for later determination of how the errors occurred. In this way, the default slave can significantly simplify the debugging process.

The AXI protocol requires that all data transfers for a transaction are completed, even if an error condition occurs. Therefore any component giving a **DECERR** response must meet this requirement.

See *AXI bus timings* on page B-3 for details of the PB11MPCore tile site multiplexed AXI bus timings.

D.2.5 Reset

A user design in a RealView Logic Tile can reset the baseboard by driving the **nSRST** signal LOW. This has the same effect as pressing the Soft Reset push-button and forces the reset controller to the level specified by the SYS_RESETCTRL register RESETCTRL field. See *Reset Control Register, SYS_RESETCTL* on page 4-21 for details.

nSRST is synchronized by the reset controller and can be driven from any clock source. It must, however, be driven active for a minimum of 84ns (two cycles of 24MHz) to ensure that it is sampled by the reset controller. In order to avoid a deadlock condition, the user design must stop driving the **nSRST** signal after **nSYSRST** is asserted.

nSRST is active low and open-drain. It is shared with the JTAG interface and must not be driven to HIGH state. A resistor on the baseboard pulls the signal HIGH.

The RealView Logic Tile also uses the **nPORESET** signal to generate a local **D_nTRST** pulse.

The **GLOBAL_DONE** signal is held LOW until the FPGA on the RealView Logic Tile has finished configuration. The system is held in reset until this signal goes HIGH.

Appendix E
Boot Monitor and platform library

This appendix describes using the Boot Monitor and platform library provided with the PB11MPCore. It contains the following sections:

- *About the Boot Monitor* on page E-2
- *About the platform library* on page E-3
- *Using the baseboard Boot Monitor and platform library* on page E-4.

E.1 About the Boot Monitor

This is the standard ARM application that runs when the system is booted. It is built with the ARM platform library.

Note

Any application that is built using the ARM platform library (or handles its own initialization) can replace the Boot Monitor.

The Boot Monitor supports the following functions:

- general file operations
- MMC and SD card utilities
- board configuration
- programming images into flash memory
- loading and running another application
- a semihosting server that handles standard ARM semihosting SVC calls.

E.2 About the platform library

The ARM platform library handles the system initialization and re-targets the standard C library. To achieve this, it provides a basic I/O subsystem that supports simple device drivers.

Included with the platform library there is a simple terminal driver, UART, PS/2 keyboard and LCD drivers and support for semihosting I/O.

E.3 Using the baseboard Boot Monitor and platform library

The baseboard Boot Monitor is a collection of tools and utilities designed as an aid to developing applications on the baseboard.

When the Boot Monitor starts on reset, the following actions are performed:

- the memory controllers are initialized
- a stack is set up in memory
- Boot Monitor code is copied into DRAM
- Boot memory remapping is reset
- C library I/O routines are remapped and redirected depending on the settings of User Switches 2 and 3.
- if Boot Monitor configuration switch User Switch 1 is ON, the current boot script, if any, is run.

Caution

The firmware must match the system configuration or the results might be unpredictable. Refer to the application note for your configuration for details of software or firmware that is specific to the combination of baseboard and tiles that your are using.

E.3.1 Boot Monitor configuration switches

The Boot Monitor application is typically loaded into the NOR flash memory and selected to run at power on. Follow the instructions in *Loading Boot Monitor into NOR flash* on page E-6 for details of loading the boot flash with the image from the Versatile CD.

Caution

The User Switches on the front panel of the ATX enclosure will not operate correctly if switch bank S4 on the PB11MPCore baseboard is not in the default state of all switches OFF.

The setting of User Switch 1 determines how the Boot Monitor starts after a reset:

User Switch 1 OFF A prompt is displayed enabling you to enter Boot Monitor commands.

User Switch 1 ON The Boot Monitor executes a boot script that has been loaded into a Multimedia (MMC) or Secure Digital (SD) card. If a boot script is not present, the Boot Monitor prompt is displayed.

The boot script can execute any Boot Monitor commands. It typically selects and runs an application image that has been stored in either NOR flash memory or on the MMC or SD card. You can store one or more code images in flash memory and use the boot script to start an image at reset. Use the `SET BOOTSCRIPT` command to set the boot script file name from the Boot Monitor (see *Standard Boot Monitor command set* on page F-3).

Output and input of text from STDIO for both applications and Boot Monitor I/O depends on the setting of User Switches 2 and 3 as listed in Table E-1.

Table E-1 STDIO redirection

User Switch 2	User Switch 3	Output	Input	Description
OFF	OFF	UART0 or console	UART0 or console	STDIO autodetects whether to use semihosting I/O or a UART. If a debugger is connected and semihosting is enabled, STDIO is redirected to the debugger console window. Otherwise, STDIO goes to UART0.
OFF	ON	UART0	UART0	STDIO is redirected to UART0. This occurs even under semihosting.
ON	OFF	LCD	Keyboard	STDIO is redirected to the LCD and keyboard. This occurs even under semihosting.
ON	ON	LCD	UART0	STDIO output is redirected to the LCD and input is redirected to the keyboard. This occurs even under semihosting.

User Switches 2 and 3 do not affect file I/O operations performed under semihosting. Semihosting operation requires a debugger and a JTAG interface device. See *Redirecting character output to hardware devices* on page E-7 for more details on I/O.

Note

User Switches 4 to 8 are not used by the Boot Monitor and are always available for user applications.

If a different loader program is present at the boot location, the function of the entire User Switch bank is implementation dependent.

E.3.2 Running the Boot Monitor

To run Boot Monitor and have it display a prompt to a terminal connected to UART0, set User Switches 1, 2, and 3 to OFF and reset the system. Standard input and output functions use UART0 by default. The default setting for UART0 is 38400 baud, 8 data bits, no parity, 1 stop bit. There is no hardware or software flow control. Use these values to configure a terminal application on your PC to communicate with the Boot Monitor.

Note

If the Boot Monitor has been accidently deleted from flash memory, see *Rebuilding the Boot Monitor or platform library* on page E-9 for information on loading the monitor.

Boot Monitor commands

See Appendix F *Boot Monitor Commands* for details of the available commands in Version 4 of the Boot Monitor.

Commands are accepted in uppercase or lowercase. The Boot Monitor accepts abbreviations of commands if the meaning is not ambiguous. For example, for `QUIT`, you can type `QUIT`, `QUI`, `QU`, `Q`, `quit`, `qui`, `qu`, or `q`.

Optional parameters for commands are indicated by []. For example `DIRECTORY [directory]` indicates that the `DIRECTORY` command can take an optional parameter to specify which directory to list the contents of. If no parameter is supplied, the default is used (in this case, the current directory).

Type `HELP` at the Boot Monitor prompt to display a full list of the available commands.

E.3.3 Loading Boot Monitor into NOR flash

If the flash becomes corrupt and the board no longer runs the Boot Monitor, the Boot Monitor must be reprogrammed into flash.

Because the debugger does not initialize DRAM, the Boot Monitor image cannot be loaded and run directly. Use the `*.brd` files and the `progcards` utility to setup the board:

1. Power off the board
2. Set all User Switches to OFF
3. Connect a cable from the JTAG device (RealView ICE for example) to the JTAG box header.
4. Power on the board.

5. Connect to the target:
   - For RVD, select **Tools → Include Commands From File**
     Select **PB11MPCore_DDR_Init_rvd_DLL.inc**
6. DRAM is now initialized and the memory is remapped. To load Boot Monitor into flash:
   a. From the debugger, load and execute the file `Boot_Monitor.axf`
   b. at the Boot Monitor prompt enter:
      ```
      >FLASH
      Flash> WRITE IMAGE path\Boot_Monitor_platform.axf
      ```
      where path is the *path* to the version of the Boot Monitor that is being loaded and *platform* is the name of the target system.
7. Loading the image into flash takes a few minutes to complete. Wait until the prompt is displayed again before proceeding.
8. Turn the board off and then on.

Boot Monitor starts automatically.

E.3.4 Redirecting character output to hardware devices

The redirection of character I/O is carried out within the Boot Monitor platform library routines in `retarget.c` and `boot.s`. During startup, the platform library executes a *Software Interrupt* instruction (SWI). If the image is being executed without a debugger (or the debugger is not capturing semihosting calls) the value returned by this SWI is –1, otherwise the value returned is positive. The platform library uses the return value to determine the hardware device used for outputting from the C library I/O functions (Redirection is through a SWI to the debugger console or directly to a hardware device).

Supported devices for character output are:

- `:UART-0` (default destination if debugger is not capturing semihosting calls)
- `:UART-1`
- `:UART-2`
- `:UART-3`.

The STDIO calls are redirected within `retarget.c`. Redirection depends on the setting of User Switches 2 and 3, see *Boot Monitor configuration switches* on page E-4 for details.

E.3.5 Using a boot script to run an image automatically

Use a boot script to run an image automatically after power-on:

1. Create a boot script from the Boot Monitor by typing:
   - if your image is in MMC or SD Card:
     ```
     M:\> CREATE myscript.txt
     ; put any startup code here RUN path\file_name
     ```
   - if your image is in CompactFlash:
     ```
     K:\> CREATE myscript.txt
     ; put any startup code here RUN path\file_name
     ```
   - if your image is in NOR flash:
     ```
     M:\> CREATE myscript.txt
     ; put any startup code here FLASH RUN file_name.
     ```
2. Press **Ctrl-Z** to indicate the end of the boot script and return to the Boot Monitor prompt.

   Note

   RVD does not support **Ctrl-Z** when creating a boot script using the debugger. You must instead type `+++` on a new line to indicate the end of a boot script.
3. Verify the file was entered correctly by typing:
   ```
   M:\>TYPE myscript.txt
   ```
   The contents of the file is displayed to the currently selected output device.
4. Specify the boot script to use at reset from the Boot Monitor by typing:
   ```
   M:\>SET BOOTSCRIPT myscript.txt
   ```
5. Set User Switch 1 ON to instruct the Boot Monitor to run the boot script at power on.
6. Reset the platform. The Boot Monitor runs and executes the boot script `myscript.txt`. In this case, it relocates the image *file_name* and executes it.

E.3.6 Rebuilding the Boot Monitor or platform library

All firmware components are built using either RVDS running under either Windows or Unix/Linux:

- For Windows, ARM RVDS can be used to rebuild the library.

  Use a CodeWarrior project file to rebuild the library. The RVDS `make` utility is not supported.

- GNUmake is available for UNIX and Linux.

  If you are using GNUmake to rebuild the Boot Monitor, set your default directory to *`install_directory`*`/Firmware/Boot_Monitor` and type `make` from a command shell.

  To rebuild the platform library component, set your default directory to *`install_directory`*`/Firmware/platform` and type `make` from a command shell.

After rebuilding the Boot Monitor, load it into NOR flash, see *Loading Boot Monitor into NOR flash* on page E-6.

After rebuilding the platform library, you can link `platform.a` from the target build subdirectories with your application (see *Building an application with the platform library* on page E-10).

Build options

You can specify the following build options after the `make` command:

- `BIG_ENDIAN=1/0`, defining image endianness (Default 0, little endian)
- `THUMB=1/0`, defining image state (Default 0, ARM)
- `DEBUG=1/0`, defining optimization level (Default 0, optimized code)
- `VFP=1/0`, defines VFP support (Default 0, no VFP support).

Note

The Boot Monitor must be built as a simple image. Scatter loading is not supported.

The build options define the subdirectory in the `Builds` directory that contains the compile and link output:

```
<Debug>_<State>_<Endianness>_Endian + further component specific options
```

For example, `Release_ARM_Little_Endian` or `Debug_Thumb_Big_Endian`.

The makefile creates a directory called Builds if it is not already present. The Builds directory contains subdirectories for the specified make options (for example, Debug_ARM_Little_Endian). To delete the objects and images for all targets and delete the Builds directory, type make clean all.

E.3.7 Building an application with the platform library

The platform library on the CD provides all required initialization code to bring the baseboard up from reset. The library is used by the Boot Monitor, but it can be used by an application independently of the other code in the Boot Monitor.

Note

It is not necessary to build your application with the platform library. You can instead let the boot monitor run at power on and remap the memory. After the remap has finished, you can load a typical application built with RVDS that is linked at address 0x8000 and uses semihosting.

The platform library supports:

- remapping of boot memory
- DRAM initialization
- UARTs
- Time-of-Year clock
- output to the character LCD display
- C library system calls.

To build an image that uses the I/O and memory control features present in the platform library:

1. Write the application as normal. There must be a main() routine in the application.

   Linking an application with the platform requires that the application is built using RVCT V2.1 or greater. If you are using RVCT V2.0 or ADS it is not possible to link the application with the platform library, however an application can still utilize the hardware on the board through semihosting.

2. Link the application against the Boot Monitor platform library file platform.a. The file platform.a is in one of the target build subdirectories (*install_dir*\software\firmware\Platform\Builds*target_build*).

   Choose the Builds subdirectory that matches your application. For example, Release_ARM_Little_Endian for ARM code. If the subdirectory does not exist, see *Rebuilding the Boot Monitor or platform library* on page E-9 for details on rebuilding platform.a.

Note

If you are not using the `platform.a` library, you must provide your own initialization and I/O routines.

You can also build the platform library functionality directly into your application without building the platform code as a separate library. This might be useful, for example, if you are using an IDE to develop your application.

See the `filelist.txt` file in the `software` directory for more details on software included on the CD. The `selftest` directory, for example, contains source files that can be used as a starting point for your own application.

3. If required, select the link time selection for the platform library options.

   Platform selection uses special symbols in your application:

   From C `#pragma import(_platform_option_xxxxx)`

   From assembler `IMPORT _platform_option_xxxxx`

   The platform options are listed in Table E-2.

Table E-2 platform library options

Option	Description
`_platform_option_no_cache`	Stops the cache from being enabled by default
`_platform_option_no_lcd_kbd`	Disables LCD & Keyboard support and stops the driver code from being loaded
`_platform_option_no_mmu`	Stops the MMU from being initialized by default.

4. Scatter loading is supported for applications using the platform library, however the scatter file must follow Example E-1 on page E-12. The execution regions INIT and SDRAM must be present, the execution regions ITCM and DTCM are optional, if they are present, the relevant *Tightly Coupled Memory* (TCM) is enabled and the base address is set to the address supplied in the scatter file. The `sys_vectors.o` object must be located at address zero. Additional execution regions, such as one for the SSRAM, can be added. An example scatter loading application can be found in the `examples` directory.

Example E-1 Scatter loading

```
LR_ROOT 0x8000
{
    INIT +0 FIXED
    {
        sys_boot.o (!!!_platform_area_boot, +FIRST)
        *(+RO)
    }
     SDRAM +0
    {
        *(+RW,+ZI)
    }
    ITCM 0x0
    {
        sys_vectors.o(_platform_area_vectors, +FIRST)
        'application code'.o(+RO)
    }
    DTCM 0x08000000
    {
        'application code'.o(+RW,+ZI)    }
}
```

5. To run the image from RAM, load the image with a debugger and execute as normal. The image uses the procedure described in *Redirecting character output to hardware devices* on page E-7 to redirect standard I/O either to the debugger or to be handled by the application itself.

   See *Loading and running an application from NOR flash* on page E-13 for more information on running from flash.

Note

If the platform library encounters a fatal error, all the user LEDs flash at a one-second interval and an error message is output on UART-0.

E.3.8 Building an application that uses semihosting

The boot monitor handles semihosting SWIs the same as a debugger handles SWIs. This enables an image that is not linked against with the platform library to be loaded into flash memory and have the boot monitor manage the I/O.

All I/O using `stdio()`, for example `printf()`, uses the same devices as the boot monitor console (either UART-0 or the Keyboard/LCD depending on the Boot Monitor configuration switch settings). File functions access the flash and character I/O devices using the mechanisms described in *Redirecting character output to hardware devices* on page E-7.

There are no specific tools requirements. Images built with RealView tools should run if they are built to use semihosting. This means that an image built using the tool kit defaults can be loaded onto the baseboard and run.

E.3.9 Loading and running an application from NOR flash

To run an image from NOR flash:

1. Build the application as described in *Building an application with the platform library* on page E-10 and specify a link address suitable for flash. There are the following options for selecting the address:

   Load region in flash

   The image is linked such that its load region, though not necessarily its execution region, is in flash. The load region specified when the image was linked is used as the location in flash and the FLASH_ADDRESS option is ignored. If the blocks in flash are not free, the command fails. Use the `FLASH RUN` command to run the image.

   Load region not in flash and image location not specified

   The image is programmed into the first available contiguous set of blocks in flash that is large enough to hold the image. Use the `FLASH LOAD` and then the `FLASH RUN` commands to load and run the image.

   Load region not in flash, but image stored at a specified flash address

   Use the FLASH_ADDRESS option to specify the location of the image in flash. If the option is not used, the image is programmed into the first available contiguous set of blocks in flash that is large enough to hold the image. Use the `FLASH LOAD` or `FLASH RUN` commands to load and run the image.

   Note

   Images with multiple load regions are not supported.

   If the image is loaded into flash, but the `FLASH RUN` command relocates code to DRAM for execution, the execution address must not be in the top 4MBytes of DRAM since this is used by the Boot Monitor.

2. The image must be programmed into flash using the Boot Monitor. Flash support is implemented in the Boot Monitor image.

   Run the Boot Monitor image from the debugger and enter the flash subsystem, type `FLASH` at the prompt:

   ```
   >FLASH
   flash>
   ```

3. The command used to program the image depends on the type of image:

   - The entry point and load address for ELF images are taken from the image itself. To program the ELF image into flash, use the following command line:

     `flash> WRITE IMAGE elf_file_name NAME name FLASH_ADDRESS address`

   - The entry point and load address for ELF images are taken from the command line options. To program a binary image into flash, use the following command line:

     `flash> WRITE BINARY image_file_name NAME name FLASH_ADDRESS address1 LOAD_ADDRESS address2 ENTRY_POINT address3`

   Note

   name is a short name for the image. If the `NAME` option is not used at the command prompt, *name* will be derived from the file name.

4. The image is now in flash and can be run by the Boot Monitor. At the prompt, type:

   `flash> RUN name`

E.3.10 Running an image from MMC or SD card or CompactFlash

To run an image from the MMC or SD card or CompactFlash:

1. Build and link the application as described in *Loading and running an application from NOR flash* on page E-13.

   Note

   Images with multiple load regions are not supported.

   The image must have an execution region in RAM or DRAM.

   The execution address must not be in the top 4MBytes of DRAM because this area is used by the Boot Monitor.

2. The image can be programmed into MMC or SD card using the Boot Monitor.

Connect a debugger and use semihosting to load the file:

For MMC or SD card:

```
M:\>COPY C:\software\elf_file_name file_name
```

For CompactFlash:

```
K:\>COPY C:\software\elf_file_name file_name
```

3. To run the image manually, from the debugger, or terminal connected to UART0 type:

   For MMC or SD card:

   ```
   M:\>RUN file_name
   ```

   For CompactFlash:

   ```
   K:\>RUN file_name
   ```

E.3.11 Using the Network Flash Utility

The *Network Flash Utility* (NFU) uses the TFTP protocol to access files over the Ethernet network. You can use this utility to program files into flash.

To connect to a server and program a file to flash:

1. Start the NFU utility from the debugger console:
   a. Set the Boot Monitor configuration switches to force the console to use either UART-0 or the LCD and keyboard. (See *Boot Monitor configuration switches* on page E-4 for details.)

      Note

      The debugger console cannot be used because the semihosted console I/O is blocking.

   b. Start the NFU utility.

      Note

      It typically takes several seconds for NFU start. Do not enter any commands until the prompt is displayed.

2. Use the DHCP protocol to get an IP address by entering:

   ```
   manage dhcpc start
   ```

3. Use the map command to map a drive letter to the TFTP server. For example to access a file on a TFTP server with the IP address 192.168.0.1, use:

   ```
   manage map n: 192.168.0.1
   ```

4. After the drive letter has been mapped, use the normal Boot Monitor command on the remote file by specifying the drive letter. For example, to write a file to NOR flash, enter:

   ```
   flash write image n:/hello.axf
   ```

NFU commands

The NFU supports a subset of the standard Boot Monitor commands and adds a new `MANAGE` sub-menu.

The NFU commands are listed in Table E-3.

Table E-3 NFU commands

Command	Action
`CD` *`directory path`*	Change directory
`CONVERT BINARY` *`binary_file`* `LOAD_ADDRESS` *`address`* `[ENTRY_POINT` *`address`*`]`	Provides information to the system that is required by the RUN command in order to execute a binary file. A new file with name *`binary_file`* is produced, but with an `.exe` file extension.
`COPY` *`file1 file2`*	Copy *`file1`* to *`file2`*. For example, to copy the `leds` code from the PC to the flash enter: `COPY C:\software\projects\examples\rvds2.0\leds.axf leds.axf` **Note** Remote file access requires semihosting. Use a debugger connection to provide semihosting.
`CREATE` *`filename`*	Create a new file in the flash by inputting text. Press Ctrl-Z to end the file.
`DELETE` *`filename`*	Delete *`file`* from flash.
`DIRECTORY [`*`directory`*`]`	List the files in a directory. Files that only accessible from semihosting cannot be listed.
`EXIT`	Exit the NFU. The processor is held in a tight loop until it is interrupted by a JTAG debugger.
`FLASH`	Enter the flash file system for the NOR flash on the baseboard. See Table F-5 on page F-7 for NOR flash commands.
`HELP`	Lists the NFU commands.

Table E-3 NFU commands (continued)

Command	Action
M:	Change drive (allocated to MMC or SD card).
K:	Change drive (allocated to CompactFlash card).
MANAGE	Enter the network management sub-menu. See Table E-4 for MANAGE commands.
MKDIR *directory path*	Create a new directory.
QUIT	Alias for EXIT. Exit the Boot Monitor.
RENAME *old_name new_name*	Rename flash file named *old_name* to *new_name*.
RMDIR *directory path*	Remove a directory.
SDCARD	Enter the SD card subsystem.
TYPE *filename*	Display the flash file *filename*.

The MANAGE sub-menu listed in Table E-4 contains the network management commands.

Entering MANAGE on the command line means that all future commands (until EXIT is entered) will be commands from the MANAGE sub-menu.

A single command can be executed by entering MANAGE followed by the command. For example, MANAGE DHCP START gets a IP address from the DHCP server. The next command entered must be from an NFU command.

Table E-4 NFU MANAGE commands

Command	Action
ARP [-a]	Display *Address Resolution Protocol* host table.
ARP -s *hostname*	Add static entry to ARP table.
ARP -d *hostname*	Delete static entry from ARP table.
DHCPC START *ifname*	Use *Dynamic Host Configuration Protocol* (DHCP) to start a connection with the network interface *ifname*.
DHCPC RELEASE *ifname*	Use DHCP to release the connection with the network interface ifname.
DHCPC SIZEOF	Returns information on the size of the DHCP packet.

Table E-4 NFU MANAGE commands (continued)

Command	Action
`DHCP INFORM ifname ip_address`	Uses the DHCP protocol send information to the server located at *ip_address* with the network interface *ifname*.
`EXIT`	Exit the `MANAGE` sub-menu. The commands listed in Table E-3 on page E-16 can be entered at the NFU prompt.
`HELP`	Lists the NFU `MANAGE` commands.
`IFCONFIG`	Displays the IP settings that are used for communications with the server.
`IFCONFIG` [*ifname* [*ip_address*]]	Displays the current IP address if *ip_address* is not suppled. Otherwise, the current IP address for the interface *ifname* is set to *ip_address*.
`IFCONFIG` [*ifname* [*option*]]	Configures the IP interface *ifname*. The value for *option* can be: `netmask maskvalue` set the netmask `dstaddr address` set the destination IP address `mtu n` set the maximum transfer unit `up` activate the interface `down` shutdown the interface
`MAP drive address`	Maps the IP address specified in *address* to *drive*.
`NETSTAT [-option]`	Displays active network connections. The value for *option* can be: `a` display all connections `m` display all multicast connections `i` display interface information `im` display interface information for multicast `r` display routing table `s` display statistics `b` display buffer usage
`PING ip_address`	Send ICMP ECHO_REQUEST packets to the network host. The data in the packet is returned by the host. Reception of the return packet indicates that the TCP/IP connection is functioning.
`QUIT`	Alias for `EXIT`. Exit the `MANAGE` sub-menu.

Table E-4 NFU MANAGE commands (continued)

Command	Action
`ROUTE ADD` *`type target`* `[NETMASK` *`mask`*`] gateway`	Adds a static route to the network address specified by *`target`*. The gateway address is specified by *`gateway`*. *`type`* can be either `-net` or `-host`. If `NETMASK` is used, *`mask`* is the netmask for the target network address.
`ROUTE DEL` *`target`*	Deletes the static route to the network address specified by *`target`*.
`SHOW DNS`	Displays *Domain Name System* (DNS) configuration details received from DHCP.

Using a script file with NFU

When NFU starts, it attempts to run the `NETSTART.BAT` file in the flash. If the script does not exist, a prompt is displayed on the console. For example, to map a drive and write a file to flash, create the following script file:

```
manage dhcp start
manage map n: 192.168.0.1
flash write image n:/hello.axf
```

After the file is executed, you can enter additional NFU commands. To run the file, reset the board and use the `RUN` command from Boot Monitor.

Appendix F
Boot Monitor Commands

This appendix describes Version 4 of the Boot Monitor command set. It contains the following section:

- *About Boot Monitor commands* on page F-2
- *Boot Monitor command set* on page F-3.

F.1 About Boot Monitor commands

The Boot Monitor is the standard ARM application that runs when the system is booted.

The Boot Monitor accepts user commands from the debugger console window or an attached terminal. A command interpreter carries out the necessary actions to complete the user commands.

Note

Commands are accepted in uppercase or lowercase. The Boot Monitor accepts abbreviations of commands if the meaning is not ambiguous. For example, for `QUIT`, you can type `QUIT`, `QUI`, `QU`, `Q`, `quit`, `qui`, `qu`, or `q`.

Optional parameters for commands are indicated by `[param]`. For example `DIRECTORY [directory]` indicates that the `DIRECTORY` command can take an optional parameter to specify which directory to list the contents of. If no parameter is supplied, the default is used (in this case, the current directory).

Type `HELP` at the Boot Monitor prompt to display a full list of the available commands.

F.2 Boot Monitor command set

Table F-1 lists the commands set for the Standard Boot Monitor.

Table F-1 Standard Boot Monitor command set

Command	Action
`@ script_file`	Runs a script file.
`ALIAS alias commands`	Create an alias command *alias* for the string of commands contained in *commands*.
`CD directory path`	Change directory.
`CLEAR BOOTSCRIPT`	Clear the current boot script. The Boot Monitor will prompt for input on reset even if the S6-1 is set to ON to indicate that a boot script should be run.
`CONFIGURE`	Enter Configure subsystem. Commands listed in Table F-2 on page F-4 can now be executed.
`CONVERT BINARY binary_file LOAD_ADDRESS address [ENTRY_POINT address]`	Provides information to the system that is required by the RUN command in order to execute a binary file. A new file with name *binary_file* is produced, but with an `.exe` file extension.
`COPY file1 file2`	Copy *file1* to *file2*. For example, to copy the `leds` code from the PC to the MMC or SD card enter: `COPY C:\software\projects\examples\rvds2.0\leds.axf leds.axf` **Note** Remote file access requires semihosting. Use a debugger connection to provide semihosting.
`CREATE filename`	Create a new file by inputting text. Press Ctrl-Z to end the file.
`DEBUG`	Enter the debug subsystem. Commands listed in Table F-4 on page F-6 can now be executed.
`DELETE filename`	Delete *file* from MMC or SD card or CompactFlash card
`DIRECTORY [directory]`	List the files in a MMC or SD card or CompactFlash card directory. Files only accessible from semihosting cannot be listed.
`DISPLAY BOOTSCRIPT`	Display the current boot script.
`ECHO text`	Echo *text* to the current output device.
`EXIT`	Exit the Boot Monitor. The processor is held in a tight loop until it is interrupted by a JTAG debugger.
`FLASH`	Enter the flash file system for the NOR flash on the baseboard. See Table F-5 on page F-7 for flash commands.

Table F-1 Standard Boot Monitor command set (continued)

Command	Action
HELP [*command*]	List the Boot Monitor commands. Entering HELP followed by a command displays help for that command.
LOAD *name*	Load the image *name* into memory and run it.
M:	Change drive (allocated to MMC or SD card).
K:	Change drive (allocated to CompactFlash card).
MKDIR *directory path*	Create a new directory.
QUIT	Alias for EXIT. Exit the Boot Monitor.
RENAME *old_name new_name*	Rename file named *old_name* to *new_name*.
RMDIR *directory path*	Remove a directory.
RUN *image_name*	Load the image *image_name* into memory and run it.
SDCARD	Enter the SD card subsystem.
SET BOOTSCRIPT *script_file*	Specify *script_file* as the boot script. If the run boot script switch S4-1 is ON, *script_file* will be run at system reset.
TYPE *filename*	Display the file *filename*.

The Boot Monitor (version 4) includes a set of MMC or SD card commands (SDCARD). The SDCARD commands are provided in a separate sub-menu.

Table F-2 lists the commands for the SDCARD sub-menu.

Table F-2 MMC, SD, and CompactFlash card sub-menu commands

Command	Action
EXIT	Exit the SDCARD commands and return to executing standard Boot Monitor commands.
FORMAT [QUICK] [VOLUME *label*]	Formats MMC or SD card for use (does not apply to CompactFlash card). QUICK – Performs a quick format by just overwriting the FAT. VOLUME *label* – Sets the disk label.
HELP [*command*]	List the SDCARD sub-menu commands. Entering HELP followed by a command displays help for that command.
IDENTIFY	Displays CompactFlash card information.

Table F-2 MMC, SD, and CompactFlash card sub-menu commands (continued)

Command	Action
INFORM *drive* [CSD|CID|SCR|SDS]	Displays Card Information. *drive* – Specify drive letter (required for PB926EJ-S only) CSD – Returns information from CSD register CID – Returns information from CID register SCR – Returns information from SCR register SDS – Issues SD_STATUS command and returns results
INITIALISE [*socket*]	If the card has been changed, call INITIALISE to intialize the card and the file system before using any commands (otherwise the behavior is unpredictable and the card can be corrupted). Option [*socket*] selects different socket: M – selects MCI 0 K – selects CompactFlash
QUIT	Alias for EXIT. Exit the SDCARD commands and return to executing standard Boot Monitor commands.

Table F-3 lists the commands for the Configure sub-menu.

Table F-3 Boot Monitor Configure commands

Command	Action
DISABLE DATA CACHE	Disables the D cache.
DISABLE I CACHE	Disables the I cache.
DISABLE MMU	Disables the MMU.
DISPLAY CLOCKS	Display system clocks.
DISPLAY DATE	Display date.
DISPLAY HARDWARE	Display hardware information (for example, the FPGA revisions).
DISPLAY TIME	Display time.
ENABLE DATA CACHE	Enables the D cache.
ENABLE I CACHE	Enable the I cache.
ENABLE MMU	Enable the MMU.
EXIT	Exit the configure commands and return to executing standard Boot Monitor commands.

Table F-3 Boot Monitor Configure commands (continued)

Command	Action
HELP [*command*]	List the configure commands. Entering HELP followed by a command displays help for that command.
QUIT	Alias for EXIT. Exit the Configure commands and return to standard Boot Monitor commands.
SET BAUD *port rate*	Set the baud *rate* for the UART *port* specified. Valid port numbers are 0, 1, 2, 3.
SET CLOCK n FREQUENCY *frequency*	Set the *frequency* in MHz of the requested CLOCK n. **Note** The Boot Monitor does not set any of the clocks on startup. The clock values are determined by the default values in the FPGA image.
SET DATE *dd/mm/yy*	Set date. The date can also be entered as *dd-mm-yy*.
SET TIME *hh:mm:ss*	Set time. The time can also be entered as *hh-mm-ss*.

Table F-4 lists the commands for the Debug sub-menu.

Table F-4 Boot Monitor Debug commands

Command	Action
DEPOSIT *address value* [*size*]	Load memory specified by *address* with *value*. The *size* parameter is optional. If used, it can be BYTE, HALFWORD, or WORD. The default is WORD.
DISABLE MESSAGES	Disable debug messages.
ENABLE MESSAGES	Enable debug messages.
EXAMINE *address*	Examine memory at *address*.
EXIT	Exit the debug commands and return to executing standard Boot Monitor commands.
GO *address*	Run the code starting at *address*.
HELP [*command*]	List the debug commands. Entering HELP followed by a command displays help for that command.
QUIT	Alias for EXIT. Exit the Debug commands and return to standard Boot Monitor commands.

Table F-4 Boot Monitor Debug commands (continued)

Command	Action
MODIFY *address value mask* [*size*]	Performs read-modify-write at memory specified by *address*. The current value at the location is ORed with the result of ANDing *value* and *mask*. The *size* parameter is optional. If used, it can be BYTE, HALFWORD, or WORD. The default is WORD.
START TIMER	Start a timer.
STOP TIMER	Stop the timer started with the START TIMER command and display the elapsed time.

Table F-5 lists the commands for the NOR Flash subsystem.

Table F-5 Boot Monitor NOR flash commands

Command	Action
CREATE BOOTSCRIPT [NAME *name*] [FLASH_ADDRESS *address*]	Creates a boot script that is stored in flash. The image will be identified in flash by the default name BOOTSCRIPT. This can be overridden by using the optional NAME argument. You can specify where in flash the bootscript is written by using the optional FLASH_ADDRESS argument.
DISPLAY BOOTSCRIPT [*name*]	Displays bootscript that is stored in flash. If the optional *name* is not specified then the default name BOOTSCRIPT is used.
DISPLAY IMAGE *name*	Displays details of image *name*.
ERASE IMAGE *name*	Erase an image or binary file from flash.
ERASE RANGE *start* [*end*]	Erase an area of NOR flash from the *start* address to the *end* address. **Note** It is only possible to erase entire blocks of flash. Therefore the entire block of flash that contains *start*, the block that contains *end* and all intervening blocks are erased. This might mean that data before *start* or after *end* will be erased if they are not on block boundaries. If the optional *end* parameter is not specified, only the single block of flash that contains *start* is erased. **Caution** This command can erase the Boot Monitor image. See *Loading Boot Monitor into NOR flash* on page E-6.
EXIT	Exit the NOR flash commands and return to executing standard Boot Monitor commands.
HELP [*command*]	List the flash commands. Entering HELP followed by a command displays help for that command.

Table F-5 Boot Monitor NOR flash commands (continued)

Command	Action
`LIST AREAS`	List areas in flash. An area is one or more contiguous blocks that have the same size and use the same programming algorithm.
`LIST IMAGES`	List images in flash.
`LOAD` *name*	Load the image *image_name* into memory.
`QUIT`	Alias for `EXIT`. Exit the NOR flash commands and return to standard Boot Monitor commands.
`RESERVE SPACE` *address size*	Reserve space in NOR flash. This space will not be used by the Boot Monitor. *address* is the start of the area and *size* is the size of the reserved area.
`RUN` *name*	Load the image *name* from flash and run it.
`UNRESERVE SPACE` *address*	Free the space starting at *address* in NOR flash. This space can be used by the Boot Monitor.
`WRITE BINARY` *file* `[NAME` *new_name*`]` `[FLASH_ADDRESS` *address*`]` `[LOAD_ADDRESS` *address*`]` `[ENTRY_POINT` *address*`]`	Write a binary file to flash. By default, the image is identified by its file name. Use `NAME` *new_name* to specify a name instead of using the default name. Use `FLASH_ADDRESS` *address* to specify where in flash the image is to be located. The optional `LOAD_ADDRESS` and `ENTRY_POINT` arguments enable you to specify the load address and the entry point. If an entry point is not specified, the load address is used as the entry point. **Note** Remote file access requires semihosting. Use a debugger connection to provide semihosting.
`WRITE IMAGE` *file* `[NAME` *new_name*`]` `[FLASH_ADDRESS` *address*`]`	Write an ELF image file to flash. By default, the image is identified by its file name. For example, t:\images\boot_monitor.axf is identified as boot_monitor. Use `NAME` *new_name* to specify a name instead of using the default name. Use `FLASH_ADDRESS` *address* to specify where in flash the image is to be located. If the image is linked to run from flash, the link address is used and *address* is ignored. **Note** Remote file access requires semihosting. Use a debugger connection to provide semihosting.

Appendix G
Loading FPGA Images

This section describes the format of the board files and how to use the `progcards` utilities to load images from the supplied CD into the baseboard and Logic Tile FPGAs and PLDs. It contains the following sections:

- *General procedure* on page G-2
- *Board files* on page G-3
- *The progcards utilities* on page G-5
- *Upgrading your hardware* on page G-6.

G.1 General procedure

The general procedure to load an image is:

1. Follow the instructions in the *Versatile CD Installation Guide* to install the software and data files on your hard disk.
2. Set up the baseboard and Logic Tile as described in *Setting up the baseboard* on page 2-2.
3. Connect RealView ICE to the JTAG connector or use a USB cable to connect a PC to the USB config port. See *JTAG debugger and USB config support* on page 2-6.
4. Place the baseboard in configuration mode (Config switch ON) and power on the development system.
5. Locate the board file (`.brd`) that matches your configuration.
   See *Board files* on page G-3.

   Caution

   The baseboard is shipped with multiplexed AXI connectivity in place for the tile site but without an image for any specific Logic Tile loaded into the configuration flash. You must load a Logic Tile FPGA image into the configuration flash before you can use the baseboard with a Logic Tile. Refer to the application note for your system configuration for specific instructions. If you change the type of Logic Tile, you must also load a new specific image.

   The image file for your configuration also includes a Boot Monitor image for NOR flash. You can also load the Boot Monitor code separately without reloading the FPGA and PLD images, see *Loading Boot Monitor into NOR flash* on page E-6.
6. Run the `progcards` utility and load the image files into the FPGAs.
   See *Upgrading your hardware* on page G-6).
   The board file selects the image files to load. Board files have a `.brd` extension.

G.2 Board files

The CD includes both the `progcards` programming utilities and the images to load into the FPGAs and PLDs. See the *Application Notes* on the CD for the combination of baseboard and Logic Tile that you are using.

G.2.1 Naming conventions for board files

The file name of a board file identifies the PCB, Logic Tile, all programmable devices, and revision. Using the board file eliminates the need to load several individual image files. All file names are in lower case with an underscore character separating the fields:

appnote_boardname_number_core_endian_build_devicelist.brd

where:

appnote The *Application Note* related to this board file.

boardname_number The name and number identify a specific development board. For example, `eb_140c` is for the baseboard that uses PCB board HBI-140C.

Note

The full board number is `HBI-XXXX`R, this is abbreviated to just the significant digits and revision, for example, `HBI-0140C` becomes `140C`.

core The name of the core that is used with this configuration. For example, `mpcore_li` identifies the ARM11 MPCore operating in little-endian mode.

Caution

Ensure that you use the board file that matches your system configuration.

`build`*n* The build number. The number *n* increments from 0

devicelist The name of the programmable devices that are specified in this file.

G.2.2 Naming conventions for image files

The image files contain the image for a single FPGA, PLD, or programmable memory device. The file name of an FPGA or PLD image file identifies the PCB, device and revision. All file names are in lower case with an underscore character separating the fields:

*boardname_number_devicename_device*_build*n*.*extension*

where:

boardname_number The name and number identify a specific development board. For example, `eb_140c` is for the baseboard that uses PCB board HBI-140C.

devicename_device

The name of the programmable devices that match this file. For example, `cfg_xc2c128` identifies the Xilinx XC2C128 configuration PLD.

`build`*n* The build number. The number *n* increments from 0

extension The type of device used by this file. For example `.svf` is used for PLD programming and `.bit` is used for FPGA programming.

Caution

The baseboard is supplied with the PLD images already programmed. The information in this section is provided, however, in case of accidental erasure of the PLDs.You are advised not to reprogram the PLDs with any images other than those provided by ARM Limited.

Using the board file to control image file loading minimizes the risk of incorrectly programming the board. The board files contains a list of correct image files and eliminates the requirement to select individual image files for the programmable devices on the baseboard or attached Logic Tile.

G.3 The progcards utilities

The progcards utilities are the primary method for programming FPGAs, configuration flash memory, and non-volatile PLDs. These utilities are used during board manufacture and to carry out field upgrades.

progcards reads a description of the board JTAG scan chain and a list of operations from a board description (*.brd) file. The file describes which bitstream and configuration files (*.bit, *.svf) must be downloaded to devices on the board.

The upgrade package for a board contains the new files and all previously released versions. This enables you to return to the original configuration.

Example G-1 Board file

```
[General]
Name = Emulation Board HPI-0140 B build 7 MPCore build working interrupts
Priority = 1
Board = Emulation_Board
[ScanChain]
TAPs = 3
TAP0 = XC2V6000
TAP1 = XC2C128
TAP2 = XC2C128
[Program]
SequenceLength = 4
Step1File = an151_eb_140b_cfg_xc2c128_build1.svf
Step1Method = PLD
Step1Tap = 1
Step2Method = Virtex2
Step2Tap = 0
Step2File = an151_eb_140b_xc2v6000_via_build1.bit
Step3Method  = IntelFlash
Step3Tap     = 0
Step3Address = 400000
Step3File    = an151_eb_140b_ctmpcore_li_build7.bit
Step4File = an151_eb_140b_mux_xc2c128_build1.svf
Step4Method = PLD
Step4Tap = 2
```

G.4 Upgrading your hardware

Use one of the progcards utilities and the board description (`*.brd`) files to load configuration images to the FPGA:

progcards_rvi.exe

`progcards_rvi.exe` uses RealView ICE and the JTAG interface. It runs on a PC host in a DOS window and communicates with the RealView ICE interface box.

See *Procedure for progcards_rvi.exe* for detailed instructions.

progcards_usb.exe

`progcards_usb.exe` uses the built-in USB Config port (USB to JTAG interface logic) on the baseboard. A standard USB cable connects the baseboard to the PC running `progcards_usb.exe`.

See *Procedure for progcards_usb.exe* on page G-8 for detailed instructions.

Note

The latest version of the RVI firmware can be downloaded from the Technical Support area of the ARM web site.

G.4.1 Procedure for progcards_rvi.exe

1. Ensure that the RealView ICE firmware is version 1.4.1 (or later) and has the additional patch required for running `progcards_rvi`. The patch can be downloaded from the downloads page on the Technical Support section of the ARM web site. See the `readme` file supplied with `progcards_rvi` for information on updating the firmware.
2. Connect the 20-way JTAG cable from the RealView ICE JTAG interface to the JTAG connector on the rear panel of the ATX enclosure. See Figure 3-3 on page 3-5.
3. Move the Config switch on the front panel of the ATX enclosure to the ON position. The orange Config indicator lights.
4. Turn on the power, the orange Config indicator on the front panel of the ATX enclosure lights.
5. Start a command window by selecting **Run** from the **Start** menu and entering `command` in the text box.

6. Change directory to the directory that contains board description (`*.brd`) files for the design to be programmed.
7. Start the programming utility by entering `progcards_rvi` at the command prompt.
8. A menu is displayed asking which interface box you want to connect to. Select the interface that is connected to the baseboard.

   Note

   See the documentation supplied with `progcards_rvi` for information on connecting directly to a specified interface that is connected to either the network or the local USB connection on the PC.

9. RVI attempts to autoconfigure. If auto-configuration fails, see the documentation supplied with `procards_rvi`.
10. `progcards_rvi` searches for board description files that match the JTAG scan chain. All board descriptions matching the first part of the chain are presented as a menu and you can select the file to use.

    `progcards_rvi` runs through the steps required to completely reprogram the boards.
11. To bypass programming a Logic Tile or the baseboard select the `Skip` option from the menu. `progcards_rvi` then looks for board description files that match the next segment of scan chain and so on.

    Typically one menu is displayed for the Logic Tile and one menu is displayed for the baseboard. If only one board description matches your hardware, it is automatically selected and no menu is displayed.

    Caution

    Ensure that the image file you are loading matches your system configuration. If the incorrect files are loaded, the baseboard and tiles might not function or might be unreliable.

12. After downloading the image completes, turn the power off and move the Config switch on the front panel of the ATX enclosure to the OFF position.
13. Set the configuration switches to match the boot option you are using.
    See *Baseboard configuration switches* on page 2-7.
14. Power on and use the Boot Monitor to load your application.
    See Appendix E *Boot Monitor and platform library*.

    If you are not using the Boot Monitor, use a JTAG debugger to load and run an application. See the documentation supplied with your debugger for details.

G.4.2 Procedure for progcards_usb.exe

1. If it is not already installed, install the USB Config Direct Control software.

   Note

   Windows USB Config drivers must be installed before using the `progcards_usb` utility. Information on installing the USB drivers can be found in `\boardfiles\USB_Config_driver\readme.txt`.

2. Connect the USB cable from the host PC to the USB Config port on the front panel of the ATX enclosure. See Figure 3-2 on page 3-4.
3. Move the Config switch on the front panel of the ATX enclosure to the ON position.
4. Turn on the power. The orange Config LED lights.
5. Start a command window by selecting **Run** from the **start** menu and entering `command` in the text box.
6. Change directory to the directory that contains board description (`*.brd`) files for the design to be programmed.
7. Start the programming utility by entering `progcards_usb` at the command prompt.

   Note

   If `progcards_usb.exe` is not in the current working directory, set your `PATH` environment variable to point to the directory that contains `progcards_usb.exe`.

8. `Progcards_usb` searches for board description files that match the scan chain. All board descriptions matching the first part of the chain are presented as a menu and you can select the file to use.

   `progcards_usb` runs through the steps required to completely reprogram the boards.

9. To bypass programming a Logic Tile or the baseboard select the `Skip` option from the menu. `progcards_usb` then looks for board description files that match the next segment of scan chain and so on.

   Typically one menu is displayed for the Logic Tile and one menu is displayed for the baseboard. If only one board description matches your hardware, it is automatically selected and no menu is displayed.

10. After downloading the image completes, turn the power off and move the Config switch to the OFF position.

11. Set the configuration switches to match the boot option you are using. See *Baseboard configuration switches* on page 2-7.
12. Power on the board and use the Boot Monitor to load your application. See Appendix E *Boot Monitor and platform library*.

    If you are not using the Boot Monitor, use a JTAG debugger to load and run an application. See the documentation supplied with your debugger for details.

G.4.3 Troubleshooting

If the upgrade process fails at any time, or the `progcards` utility reports an error, then check the following:

1. The Config switch is ON and the orange Config LED is on.
2. If you are using `progcards_rvi.exe`, see the documentation supplied with `progcards_rvi`.
3. If you are using `progcards_usb.exe` ensure that:
   a. the USB cable is plugged into the USB Config port on the front panel of the ATX enclosure and not one of the USB user ports on the rear panel.
   b. the USB Config drivers are installed on your machine.
4. Ensure that any Logic Tiles are correctly stacked on the baseboard. If necessary, push them firmly to ensure a proper connection.
5. Look for any `readme.txt` files that might be present in the directory that contains the board description (`*.brd`) files. Follow the instructions provided for new or updated software or engineering changes.

Glossary

This glossary lists abbreviations used in the User Guide.

ASIC	Application Specific Integrated Circuit.
ADC	Analog to Digital Converter. A device that converts an analog signal into digital data.
AHB	Advanced High-performance Bus. An ARM open standard bus protocol.
AMBA 3	Advanced Microcontroller Bus Architecture version 3.
AXI	AMBA 3 Advanced eXtensible Interface. An ARM open standard bus protocol.
BIST	Built In Self Test
DAC	Digital to Analog Converter. A device that converts digital data into analog level signals.
PB11MPCore	RealView Platform Baseboard for ARM11 MPCore. A hardware platform used for system prototyping and debugging of ARM multi-processors.
FPGA	Field Programmable Gate Array.
ICE	In Circuit Emulator. A interface device for configuring and debugging processor cores.
I/O	Input/Output.
JTAG	Joint Test Action Group. The committee which defined the IEEE test access port and boundary-scan standard.

LED Light Emitting Diode.

Multi-ICE Equipment for running and controlling a JTAG interface for system debug. This is legacy JTAG equipment. Does not support the ARM1176JZF-S processor.

PCI Peripheral Component Interconnect. A circuit board level bus interconnect.

PISMO Memory specification for plug in memory modules.

PLD Programmable Logic Device.

PLL Phase-Locked Loop. A type of programmable oscillator that tracks the input frequency and can generate arbitrary multiples or divisions of the input frequency.

RAM Random Access Memory.

RVI RealView ICE. Equipment for running and controlling a JTAG interface for system debug. This is current JTAG equipment. Supports the ARM1176JZF-S processor.

TCM Tightly Coupled Memory, a fast memory block that connects directly to a dedicated I/O port on the processor.

USB Universal Serial Bus. Hardware interface for connecting peripheral devices.