RealView™ Emulation Baseboard

HBI-0140 Rev C

User Guide

ARM®

Copyright © 2005-2011 ARM Limited. All rights reserved.
ARM DUI 0303E

RealView Emulation Baseboard
User Guide

Copyright © 2005-2011 ARM Limited. All rights reserved.

Release Information

Change history

Description	Issue	Confidentiality	Change
December 2005	A	Non Confidential	New document
July 2006	B	Non Confidential	Second release to fix documentation defects
May 2007	C	Non-Confidential	Third release to fix documentation defects
October 2007	D	Non-Confidential	Fourth release to fix documentation defects
April 2011	E	Non-Confidential	Fifth release to fix documentation defects

Proprietary Notice

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

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

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

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

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 baseboard 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 board
- 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Contents
RealView Emulation Baseboard User Guide

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Chapter 3 Hardware Description

Chapter 4 Programmer's Reference

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

List of Tables
RealView Emulation Baseboard User Guide

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

List of Figures
RealView Emulation Baseboard User Guide

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Preface

This preface introduces the *RealView Emulation Baseboard User Guide*. It contains the following sections:

- *About this document* on page xviii
- *Feedback* on page xxii.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

About this document

This document describes how to set up and use the RealView *Emulation Baseboard* (baseboard).

Intended audience

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

Organization

This document is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an introduction to the baseboard. 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 baseboard. This chapter describes how to connect the add-on boards and how to apply power.

Chapter 3 ***Hardware Description***

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

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

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

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, timing, and mechanical specifications.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Appendix C *LCD Kits*

Refer to this appendix for details of the CLCD display and interface.

Appendix D *PCI Backplane and Enclosure*

Refer to this appendix for details of the PCI backplane board.

Appendix E *PISMO Memory Expansion Boards*

Refer to this appendix for details of the memory expansion boards.

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.

Further reading

This section lists related publications by ARM Limited and other companies.

ARM publications

The following publications provide reference information about the ARM architecture:

- *AMBA™ Specification* (ARM IHI 0011)
- *ARM Architecture Reference Manual* (ARM DDI 0100).

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

- *Multi-ICE™ User Guide* (ARM DUI 0048)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

- *RealView™ ICE and RealView Trace User Guide* (ARM DUI 0155)
- *Trace Debug Tools User Guide* (ARM DUI 0118)
- *ARM MultiTrace® User Guide* (ARM DUI 0150)
- *ARM RealView Logic Tile LT-XC2V4000+ User Guide* (ARM DUI 0186)
- *ARM RealView Core User Guide* (ARM DUI 01273)
- *RealView Debugger v3.0 Essentials Guid*e (ARM DUI 0181)
- *RealView Debugger v3.0 Target Configuration Guid*e (ARM DUI 0182)
- *RealView Debugger v3.0 Trace User Guid*e (ARM DUI 0323)
- *RealView Compilation Tools Compilers and Libraries Guide* (ARM DUI 0205)
- *RealView Compilation Tools Developer Guide* (ARM DUI 0203)
- *RealView Compilation Tools Linker and Utilities Guid*e (ARM DUI 0206).

The following publications provide information about ARM PrimeCell® and other peripheral or controller devices:

- *ARM PrimeCell® Advanced Audio CODEC Interface (PL041) Technical Reference Manual* (ARM DDI 0173)
- *ARM PrimeCell Color LCD Controller (PL111) Technical Reference Manual* (ARM DDI 0161)
- *ARM Dual-Timer Module (SP804) Technical Reference Manual* (ARM DDI 0271)
- *ARM PrimeCell DMA (PL081) Technical Reference Manual* (ARM DDI 0196)
- *ARM Dynamic Memory Controller (PL340) Technical Reference Manual* (ARM DDI 0331)
- *ARM PrimeCell External Bus Interface (PL220) Technical Reference Manual* (ARM DDI 0249)
- *ARM PrimeCell GPIO (PL061) Technical Reference Manual* (ARM DDI 0190)
- *ARM PrimeCell Keyboard Mouse Controller (PL050) Technical Reference Manual* (ARM DDI 0143)
- *ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual* (ARM DDI 0172)
- *ARM PrimeCell Real Time Clock Controller (PL031) Technical Reference Manual* (ARM DDI 0224)
- *ARM PrimeCell Smart Card Interface (PL131) Technical Reference Manual* (ARM DDI 0228)
- *ARM PrimeCell Synchronous Serial Port Controller (PL022) Technical Reference Manual* (ARM DDI 0194)
- *ARM PrimeCell Synchronous Static Memory Controller (PL093) Technical Reference Manual* (ARM DDI 236)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

- *ARM PrimeCell System Controller (SP810) Technical Reference Manual* (ARM DDI 0254)
- *ARM PrimeCell UART (PL011) Technical Reference Manual* (ARM DDI 0183)
- *ARM PrimeCell Watchdog Controller (SP805) Technical Reference Manual* (ARM DDI 0270).

Other publications

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

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

The following data sheets describe some of the integrated circuits or modules used on the baseboard:

- *CODEC with Sample Rate Conversion and 3D Sound* (LM4549) National Semiconductor, Santa Clara, CA.
- *Mobile DiskOnChip Plus 32/64MByte*, M-Systems Inc., Newark, CA.
- *MultiMedia Card Product Manual* SanDisk, Sunnyvale, CA.
- *Serially Programmable Clock Source* (ICS307), ICS, San Jose, CA.
- *Serial Microwire Bus EEPROM* (M93LC46) STMicroelectronics, Amsterdam, The Netherlands.
- *1.8 Volt Intel StrataFlash® Wireless Memory* with 3.0 Volt I/O (28F256L30B90) Intel Corporation, Santa Clara, CA.
- *Three-In-One Fast Ethernet Controller* (LAN91C111) SMSC, Hauppauge, NY.
- *ISP1761 Hi-Speed Universal Serial Bus On-The-Go controller Product data sheet* Philips, The Netherlands.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Feedback

ARM Limited welcomes feedback both on the baseboard and on the documentation.

Feedback on the baseboard

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

- the product name
- an explanation of your comments.

Feedback on this document

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

- the document title
- the document number
- the page number(s) to which your comments refer
- an explanation of your comments.

General suggestions for additions and improvements are also welcome.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Chapter 1
Introduction

This chapter introduces the RealView Emulation Baseboard. It contains the following sections:

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

1.1 About the baseboard

The baseboard is a highly integrated development board that contains:

- a large FPGA (Xilinx Virtex-II XC2V6000)
- static and dynamic memory
- integrated peripherals
- two tile connectors that enable connection of Core Tiles (for example the CT926EJ-S) or Logic Tiles (for example the LT-XC2V8000).

The baseboard typically uses one Core Tile to provide the ARM processor. The FPGA implements the bus infrastructure, memory controllers, and peripheral controllers. There are various FPGA images provided that enable the use of the entire range of Core Tiles. You can modify the FPGA design to prototype custom memory controllers and peripherals.

You can use the baseboard as a basic software development system with a power supply, a Core Tile, and a connection to a JTAG interface unit.

You can expand the baseboard by adding:

- an additional Core Tile to create a multiprocessor system
- one or more Logic Tiles containing custom IP
- a PCI expansion enclosure
- a PISMO static memory expansion board
- VGA monitor or CLCD adaptor and CLCD display
- MMC, SD, or SIM cards
- custom devices to the 16-bit GPIO
- USB devices to the three USB ports
- serial devices to the synchronous serial port and the four UARTs
- a keyboard and mouse
- audio devices to the onboard CODEC
- an Ethernet network to the onboard Ethernet controller.

The expanded system with Core Tiles and Logic Tiles can be used to develop AMBA-compatible (AHB or AMBA3 AXI) peripherals and to test ASIC designs. The fast processor core present in the Core Tile and the peripherals present in the baseboard FPGA or Logic Tile FPGA enable you to develop and text complex systems operating near their target operating frequency.

Figure 1-1 on page 1-3 shows the layout of the baseboard.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

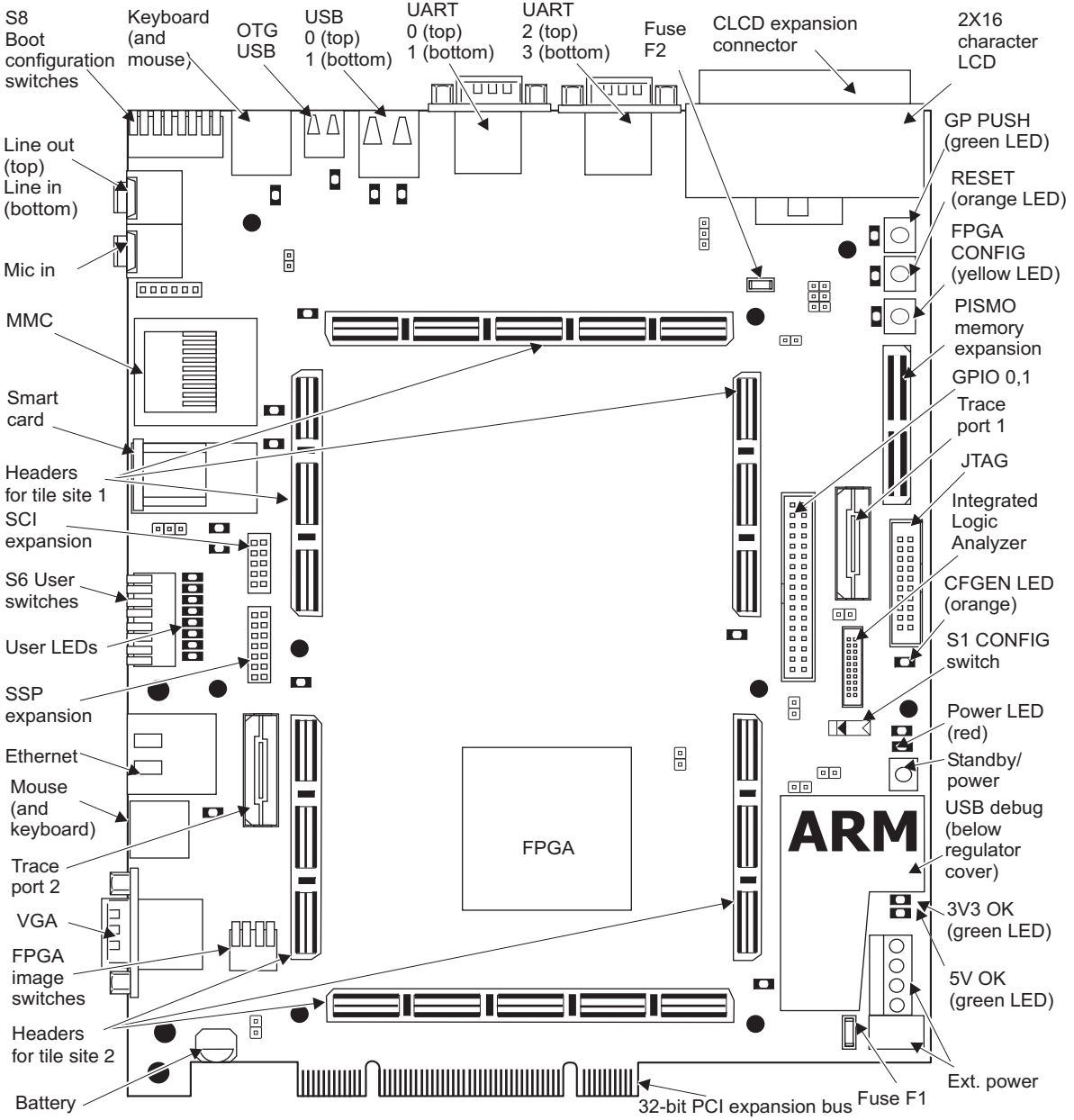

Figure 1-1 baseboard layout

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

1.2 Baseboard architecture

The major components on the baseboard are:

- Two tile sites (supports ARM Core Tiles and Logic Tiles)
- *Field Programmable Gate-Array* (FPGA) that implements a bus matrix, configuration interface, peripheral controllers, and interface logic

  An 8MB configuration flash that holds FPGA images
- 256MB of 32-bit wide DDR SDRAM
- 4MB of 16-bit wide Cellular (Pseudo-static) RAM
- 64MB of 32-bit wide NOR flash
- 64MB of 16-bit wide NAND Disk-on-Chip flash
- up to 320MB (5x64MB) of static memory (flash or RAM) in an optional PISMO expansion board
- PCI expansion connector
- USB interface controller IC and connector
- Ethernet interface controller IC and connector
- connectors for VGA, color LCD display interface board, four UARTs, GPIO, keyboard, mouse, Smart Card, audio, MMC, and SSP
- Electronic switches that select between the controllers located in the FPGA or on one of the tile sites.
- debug and test connectors for JTAG, Integrated Logic Analyzer, and Trace port
- general purpose DIP switches and LEDs
- 2 row by 16 character LCD display
- power supply circuitry
- *Real-Time Clock* (RTC)
- time of year clock with backup battery
- programmable clock generators.

1.2.1 System architecture

Figure 1-2 on page 1-5 shows the architecture of the baseboard.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

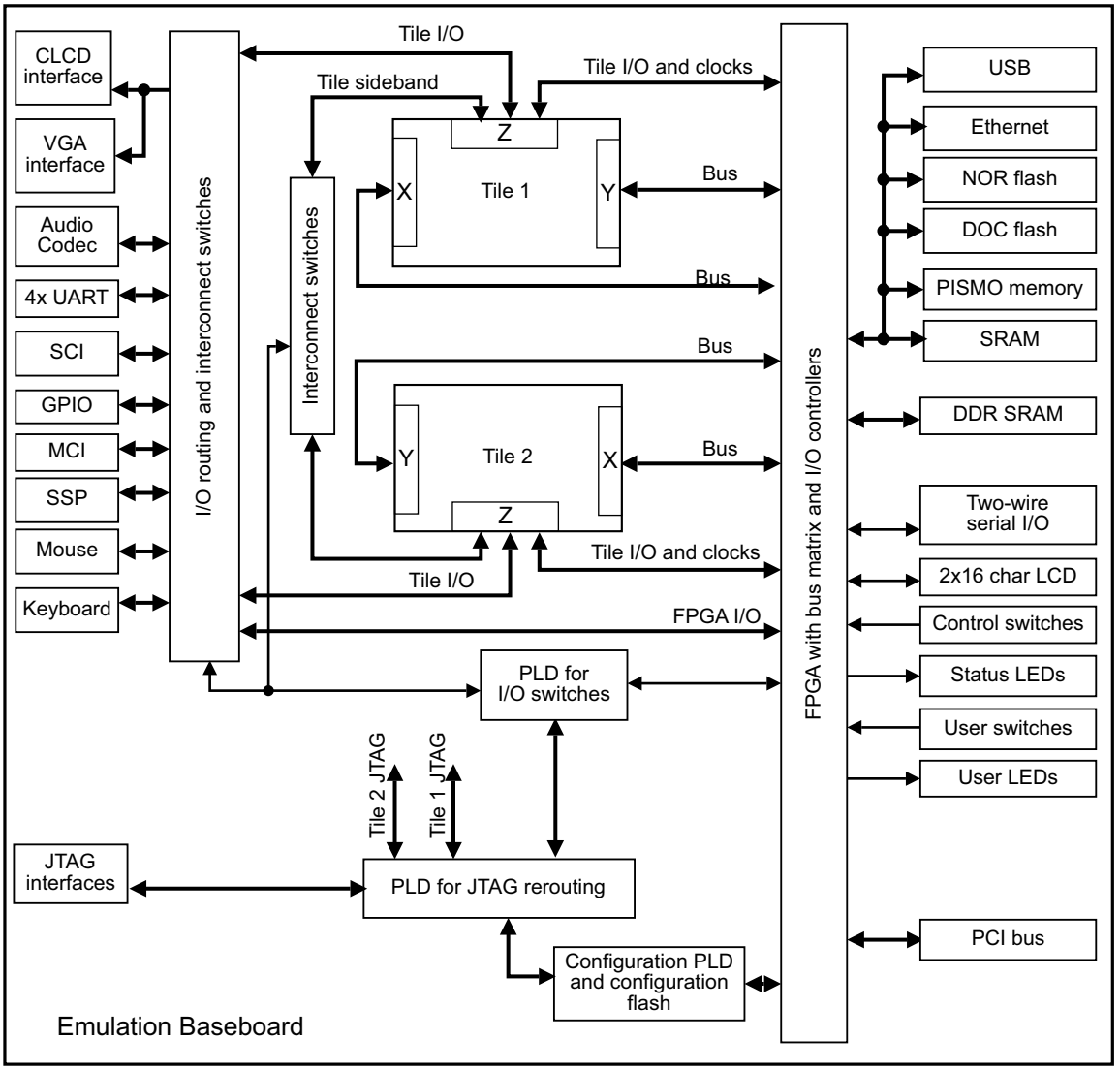

Figure 1-2 baseboard block diagram

1.2.2 Baseboard FPGA

The Xilinx XC2V6000 FPGA provides system control and configuration functions for the baseboard that enable it (together with a Core Tile) to operate as a development system.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The FPGA also implements additional peripherals and controllers, for example the audio CODEC, MultiMedia Card interface, interrupt controllers, Watchdog, and PCI interfaces. Additional peripherals can be added to the baseboard FPGA or to the FPGAs in expansion Logic Tiles. See *FPGA* on page 3-16.

Note

The peripherals and controllers implemented in the baseboard FPGA are in the standard images distributed as application notes by ARM Ltd.

You can replace some or all of the logic in the FPGA with your own designs, however, the details of how to do this are beyond the scope of this user guide. The CD includes FPGA HDL and signal definition files that can be used as a basis for custom designs. If the FPGA is modified to include custom designs, the Xilinx PCI IP block is removed from the design.

The image loaded into the FPGA must match the system configuration. See *Loading FPGA and PLD images* on page 2-20. The application notes present on the CD and available for download from the ARM web site provide the files for specific combinations of boards.

1.2.3 Core Tile expansion

Core Tiles, such as the CT926EJ-S, provide one or more processors for the baseboard system. Depending on the Core Tile used, the baseboard can be configured to use either AXI, AHB, or ARM7TDMI buses (see *Connecting a Core Tile* on page 2-8). There are two tile sites. A Core Tile is typically added to tile site 1 and a second Core Tile or a Logic Tile can be added to tile site 2.

1.2.4 PCI expansion bus

The PCI expansion bus (together with the PCI backplane) allows PCI cards to be used with the baseboard (see Appendix D *PCI Backplane and Enclosure*). The baseboard can also be plugged into a PC that has a 3.3V or 5V 32-bit PCI expansion bus.

1.2.5 Displays

The baseboard FPGA outputs signals for a color LCD display. An external interface board can be connected to the CLCD connector to drive different size displays.

The CLCD signals are also converted on the baseboard to VGA signals and output on a standard VGA connector. The resolution of the VGA signal is configurable. See Appendix C *LCD Kits*.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

There is also a two row by sixteen character display mounted on the baseboard. This display shows board status information and can be used for debugging or as the output from applications.

1.2.6 Logic Tile expansion

Logic Tiles, such as the LT-XC2V6000, enable the development of additional AMBA AXI, AHB, and APB peripherals, or custom logic, for use with ARM cores. You can place standard or custom peripherals in the FPGAs on the baseboard and the Logic Tile. See *Connecting a Logic Tile* on page 2-11 and *Tile headers and signal interconnects* on page 3-3.

1.2.7 Endianess

The baseboard and the example software supports little endian by default. Refer to the application note for your hardware configuration for information on support for other endian options.

1.2.8 Clock generators

The baseboard contains the following clock sources:

- crystal oscillators (these are the reference frequencies for the Real Time Clock, USB, AACI, Ethernet, and programmable oscillators)
- four programmable ICS307 clock sources. Two of these are used as the reference for the CPU system clock to the Core Tile and the CLCD controller clock. The other programmable clocks are routed through the FPGA and can be used as for example external reference clocks for attached tiles
- if fitted, the PCI backplane oscillator
- if fitted, a clock can be imported from one of the three programmable oscillators on a Logic Tile.

See *Clock architecture* on page 3-37.

1.2.9 Debug and test interfaces

The JTAG connector enables JTAG hardware debugging equipment, such as Multi-ICE or RealView ICE, to be connected to the baseboard. The JTAG equipment enables you to debug ARM-based applications.

The `progcards` utility can control the JTAG signals from the on-board USB debug port controller and enable downloading images to the FPGA and PLDs. See *JTAG and USB debug port support* on page 3-83.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

An *Integrated Logic Analyzer* (ILA) connector enables debugging the baseboard 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.

Two trace connectors are present on the baseboard for accessing processor cores implemented on Logic Tiles. Attached Core Tiles might also have their own trace connectors if trace is supported by the processor test chip fitted on the Core Tile.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

1.3 Tile interconnections

The two tile sites on the baseboard enable the board to be used with Core Tiles and Logic Tiles. The tiles are stackable and each tile has three connectors on the top and bottom (HDRX, HDRY, and HDRZ).

The bus usage on the tiles and baseboard depend on the type and combination of tiles. HDRX has the master bus signals (AXI, AHB, or ARM7TDMI bus) for both Core Tiles and Logic Tiles. For Logic Tiles, HDRY has the slave bus signals. (See Figure 1-3.)

Some of the I/O devices on the board can be driven by the FPGA or by I/O controllers implemented in a Logic Tile present on tile site 2.

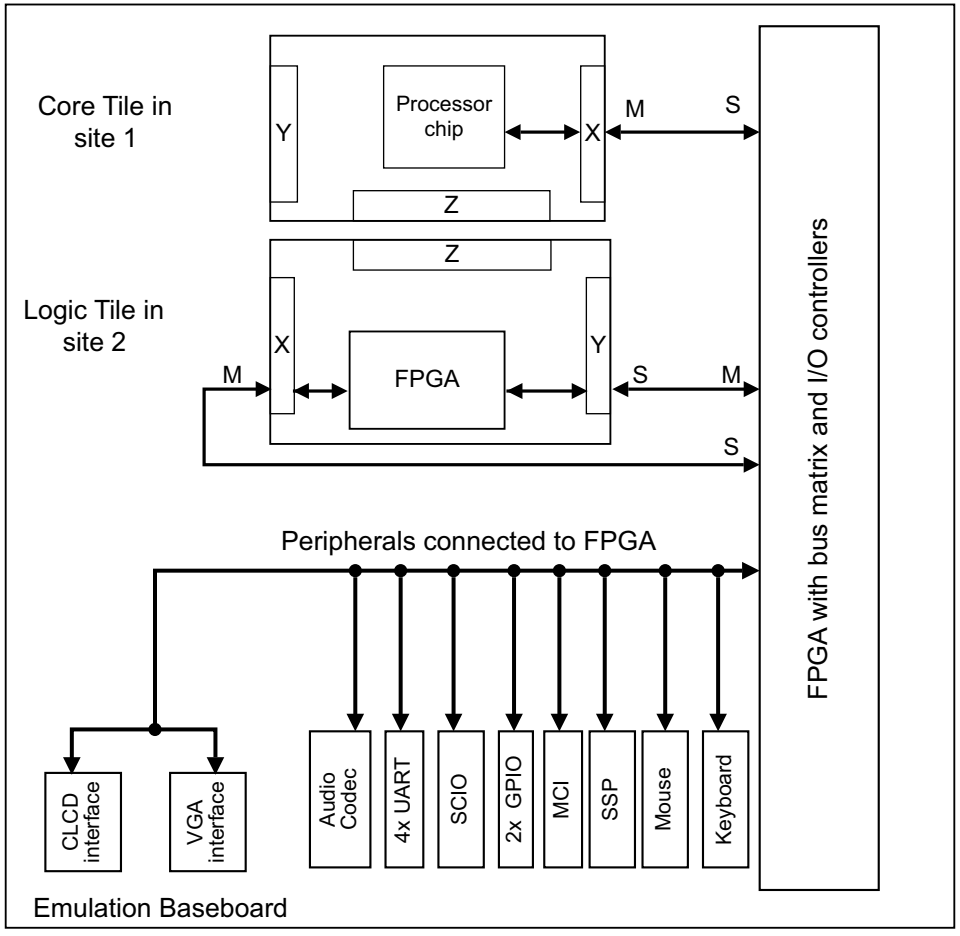

Figure 1-3 Example of bus routing

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

For more detail, see *Tile headers and signal interconnects* on page 3-3.

Note

Some Core Tiles have a second master or slave bus on HDRY. See the documentation for your Core Tile for details of bus usage.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

1.4 Precautions

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

1.4.1 Ensuring safety

The baseboard can be powered from one of the following sources:

- the supplied power supply connected to J28
- a bench power supply connected to the screw terminals on header J27
- an external PCI bus.

Warning

Do not supply more than one power source. If you are using the baseboard with the PCI enclosure for example, do not connect a power source to J27 or J28.

To avoid a safety hazard, only connect *Safety Extra Low Voltage* (SELV) equipment to the connectors on the baseboard.

1.4.2 Preventing damage

The baseboard is intended for use in a laboratory or engineering development environment. If operated without an enclosure, the board is sensitive to electrostatic discharges and generates electromagnetic emissions.

Caution

To avoid damage to the board, observe the following precautions:

- never subject the board to high electrostatic potentials
- always wear a grounding strap when handling the board
- only hold the board by the edges
- avoid touching the component pins or any other metallic element
- do not connect more than one power source to the platform
- always power down the board when connecting Logic Tiles or expansion boards.

Caution

Do not use the board near equipment that is:

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

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Chapter 2
Getting Started

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

Caution

The FPGA image must match attached the Core and Logic Tiles. Do not mount any tiles until you ensure that the correct FPGA image is present (see *Loading FPGA and PLD images* on page 2-20).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.1 Setting up the baseboard

The following items are supplied with the baseboard:

- the baseboard mounted on a metal tray
- an AC power supply that provides a 12VDC output
- a CD containing sample programs, Boot Monitor code, FPGA RTL, FPGA and PLD images, and additional documentation
- USB OTG, Ethernet, and serial cables
- this user guide.

To set up the baseboard as a development system:

1. Set the configuration switches to select the boot memory location and FPGA image. See *Setting the configuration switches* on page 2-4.
2. Connect a Core Tile to tile site 1.

   Caution

   You must ensure that the FPGA image matches the Core Tile you are using. See *Connecting a Core Tile* on page 2-8.

3. If you are using a memory expansion board, connect it to the PISMO expansion socket on the baseboard. See *Connecting expansion memory* on page 2-12 and Appendix E *PISMO Memory Expansion Boards*.
4. If you are using an external display:
   - For VGA displays, connect the cable from the display to the VGA connector on the baseboard.
   - For *Color Liquid Crystal Displays* (CLCD), connect the CLCD adaptor board cable to the baseboard. See Appendix C *LCD Kits*.
5. Apply power to the baseboard. See *Supplying power* on page 2-19.

   Caution

   If you are using the baseboard with the PCI backplane, see Appendix D *PCI Backplane and Enclosure*. If used with a PCI backplane, the power is supplied from the backplane only.

6. If you are using Logic Tiles to implement peripherals, mount the tiles in tile site 2. See *Connecting a Logic Tile* on page 2-11.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

If you are using a second Core Tile to implement a multi-processor system, mount the second Core Tile in tile site 2.

7. Load a new image into the configuration flash on the baseboard. See *Loading FPGA and PLD images* on page 2-20 and *FPGA configuration* on page 3-17.

   If you are using a Logic Tile, the configuration flash on the Logic Tile must also contain the appropriate image for the Logic Tile FPGA. See *FPGA configuration* on page 3-17, the relevant application note, and the documentation for your Logic Tile.

   Caution

   If the wrong images are loaded into the configuration flash on the baseboard and the Logic Tiles, there might be output signal conflicts that result in excessive power consumption or damage to the boards.

8. If you are using a JTAG debugger, connect it to the JTAG port on the board. See *Connecting JTAG debugging equipment* on page 2-13.

9. If you are using a *Trace Port Analyzer* (TPA), connect the Trace Port interface buffer board to the connector on the Core Tile. See *Connecting the Trace Port Analyzer to the baseboard* on page 2-16.

10. Power off and then power the baseboard back on. The power on sequence for the baseboard is:
    - the image selected by the configuration switches is loaded into the FPGA from the configuration flash
    - after FPGA configuration finishes, the GLOBAL_DONE LED is lit.
    - the processor begins executing instructions in memory at address `0x0`.

      Note

      The memory-remap switches control the remapping of non-volatile memory to respond at address `0x0` at power on. The non-volatile memory typically contains the Boot Monitor or an operating system loader.

11. If you are using the supplied Boot Monitor software to select and run an application, see *Using the baseboard Boot Monitor and platform library* on page 2-32.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.2 Setting the configuration switches

Figure 2-1 shows the configuration switches on the baseboard.

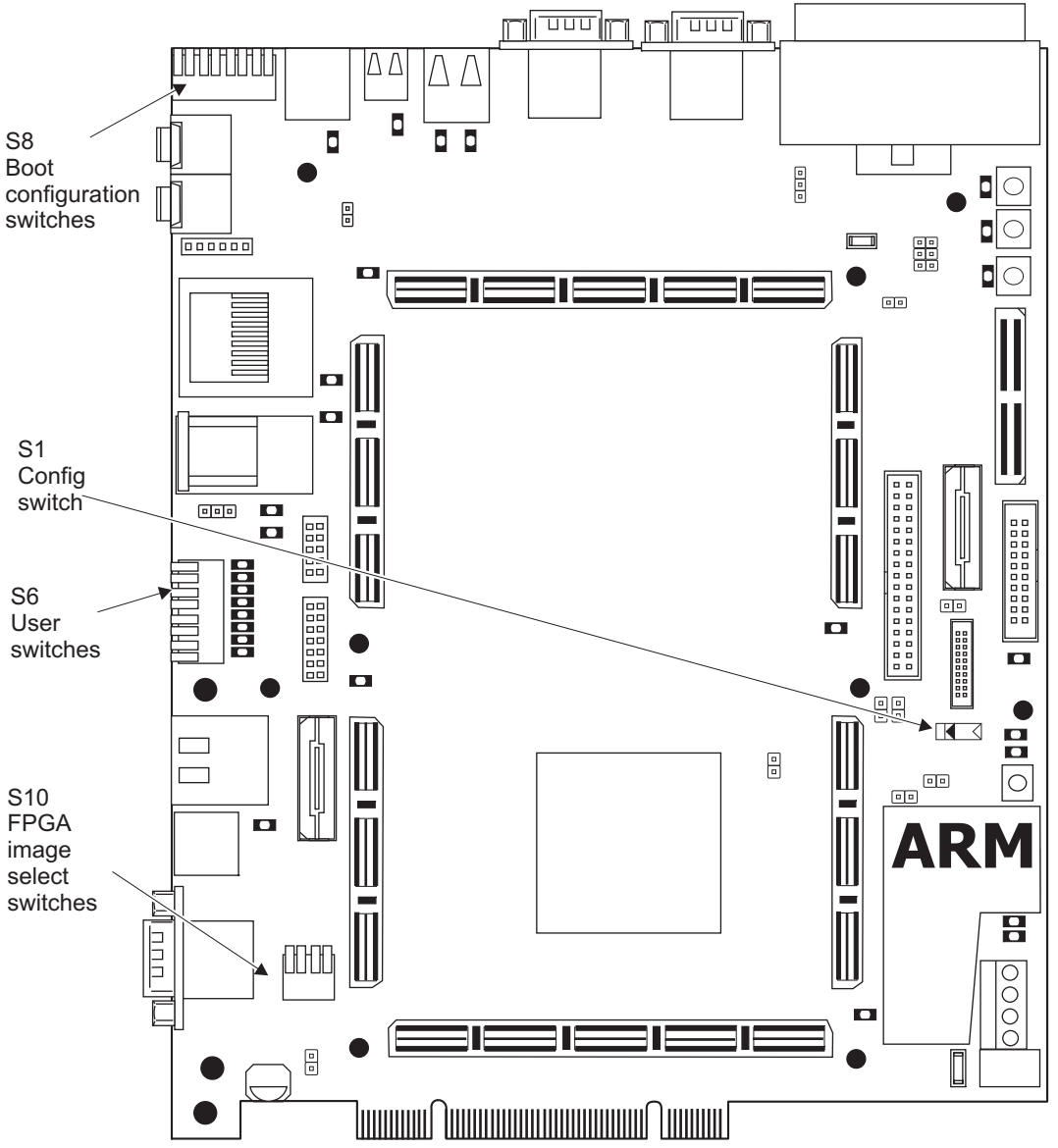

Figure 2-1 Location of configuration switches

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

The function of switches S6 and S8 is determined by the image loaded into the FPGA. The descriptions in this section refer to the standard images supplied with the product.

Switch S8 controls how the baseboard configures itself and the action to take after reset. User switch S6 can be read by an application running on the baseboard. S6 is also used to select options for the Boot Monitor program at power on. S10 selects the image from the configuration flash memories that is loaded into the FPGAs.

2.2.1 Boot memory and clock speed configuration switch S8

The configuration switches S8-1 to S8-4 determine boot memory type.

Note

If the switch lever is down, the switch is ON. The default is OFF, switch lever up.

Table 2-1 Selecting the boot device

S8-4	S8-3	S8-2	S8-1	Memory device used as boot memory
OFF	OFF	OFF	OFF	NOR Flash remapped to address `0x0`.
OFF	OFF	OFF	ON	Flash remapped to address `0x0`.
OFF	OFF	ON	OFF	SRAM remapped to address `0x0`.
OFF	OFF	ON	ON	Reserved.
OFF	ON	OFF	OFF	PISMO expansion memory selected by **nCS1** is remapped to address `0x0`.
OFF	ON	OFF	ON	PISMO expansion memory selected by **nCS2** is remapped to address `0x0`.
OFF	ON	ON	OFF	PISMO expansion memory selected by **nCS3** is remapped to address `0x0`.
OFF	ON	ON	ON	PISMO expansion memory selected by **nCS0** is remapped to address `0x0`.
ON	X	X	X	Routing switches S8[3:1] are ignored and all accesses are placed on the tile site 2 master bus for slaves located in an attached Logic Tile.

The Boot Monitor remaps SDRAM to address `0x0`. If you are not using the Boot Monitor, your OS loader or application must reset the memory remapping (see *System Controller* on page 4-88).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The default values for configuration switches S8-1 to S8-8 are listed in Table 2-2. For more information on configuration switch S8, see *Test, configuration, and debug interfaces* on page 3-83 and the Application Note for the FPGA image and Core Tile that you are using.

Table 2-2 Default switch positions

Switch	Default	Function in default position
S8-1	OFF	Selects as boot memory (see Table 2-1 on page 2-5)
S8-2	OFF	
S8-3	OFF	
S8-4	OFF	
S8-5	OFF	Sets the default bus frequency: • ON selects 10MHz system clock (typically the ACLK and HCLK). • OFF selects the maximum frequency for the attached system (typically this is 25MHz for AHB systems or 30MHz for AXI systems).
S8-6	OFF	Reserved to override default system clock frequency. See the Application Note for your product configuration for more details. The default selection typically bypasses the PLL and sets the bus to core ratio to 1:1.
S8-7	OFF	
S8-8	OFF	

2.2.2 FPGA image select switch

The default values for configuration switches S10-1 to S10-4 are listed in Table 2-3. For more information on FPGA configuration, see *FPGA configuration* on page 3-17.

Table 2-3 Default positions for FPGA select switch S10

Switch	Default	Function in default position
S10-1	OFF	Tile site 1 FPGA image 1 selected, (image 2 selected if ON).
S10-2	OFF	Tile site 2 FPGA image 1 selected, (image 2 selected if ON).

S10-3	OFF	Selects baseboard FPGA image in config flash.
		`00` image 1 at address `0x000000`
		`01` image 2 at address `0x400000`

S10-4	OFF	Reserved.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

For information on other configuration links and the function of the status LEDs, see *Test, configuration, and debug interfaces* on page 3-83.

2.2.3 Boot Monitor configuration switch

The Boot Monitor application is typically loaded into the NOR or DOC flash memory and selected to run at power on. Follow the instructions in *Loading Boot Monitor into NOR flash* on page 2-39 or *Loading Boot Monitor into Disk-on-Chip* on page 2-41 for details of loading the boot flash with the image from the CD.

Table 2-4 Default positions for FPGA select switch S10

Switch	Default	Function in default position
S6-1	OFF	A prompt is displayed enabling you to enter Boot Monitor commands after the system is started.
S6-2	OFF	STDIO autodetects whether to use UART0 or the debugger console.
S6-3	OFF	
S6-4 to S6-8	OFF	Reserved for use by the application

For more detail on Boot Monitor configuration options, see *Boot Monitor configuration switches* on page 2-32.

2.2.4 Config switch

The config switch selects between normal debug mode (switch OFF) and configuration mode (switch ON).

The baseboard is placed into configuration mode to load images into the FPGA and PLDs. For more details on debug and configuration modes, see *Connecting JTAG debugging equipment* on page 2-13 and *Upgrading your hardware* on page 2-24.

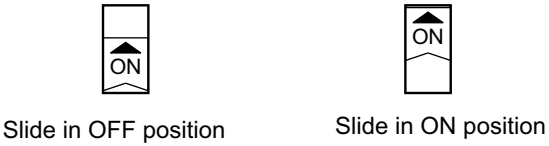

Figure 2-2 CONFIG slide switch

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.3 Connecting a Core Tile

The baseboard does not include a processor. A processor must be added by:

- adding a Core Tile (this is the typical way of adding a processor)
- adding a Logic Tile and implementing a processor in the Logic Tile FPGA
- implementing an SMM in the FPGA on the baseboard.

Caution

The FPGA image must match the type and number of Core Tiles that are present on the baseboard.

Because the AHB and AXI buses use different pins, an incorrect FPGA configuration might result in damaged boards. To check the configuration:

1. Set the CONFIG switch S1 to the OFF position.
2. Power on the board.
3. Check that the text displayed on the character LCD matches the Core Tile to add.

The FPGA images are included with the application notes on the supplied CD:

- use AN152 for ARM11MPCore Core Tile
- use AN148 for ARM7TDMI, ARM926EJ-S and ARM1136JF-S Core Tiles.

See *Loading FPGA and PLD images* on page 2-20, *FPGA configuration* on page 3-17, and the relevant application notes for more details.

One or two Core Tiles can be connected to the baseboard:

- Figure 2-3 on page 2-9 shows a single Core Tile attached to tile site 1. Tile site 1 is typically used when there is only one tile.
- Figure 2-4 on page 2-9 shows two Core Tiles attached to tile site 1 and 2.

Press down firmly on the Core Tile to ensure that the connectors mate properly.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

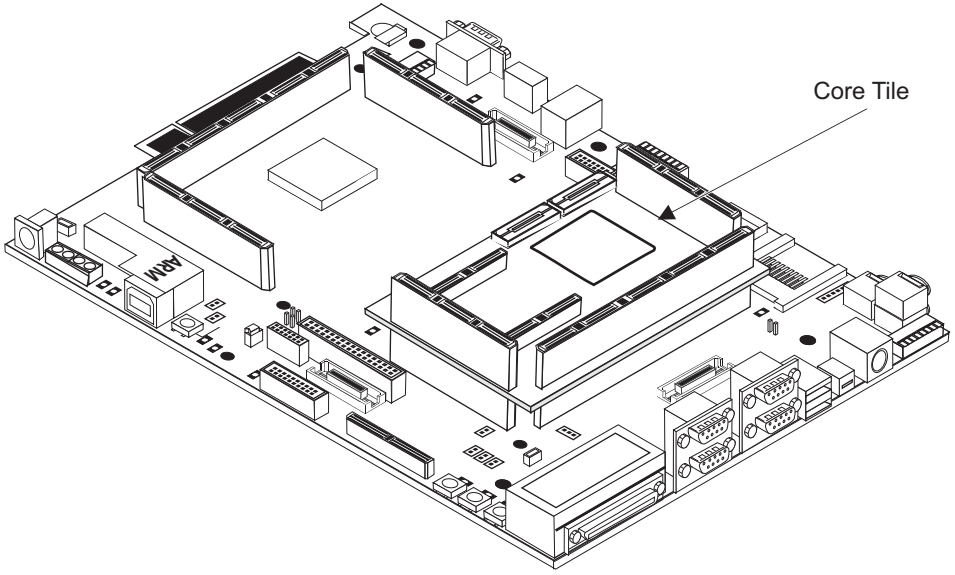

Figure 2-3 Core Tile mounted on tile site 1

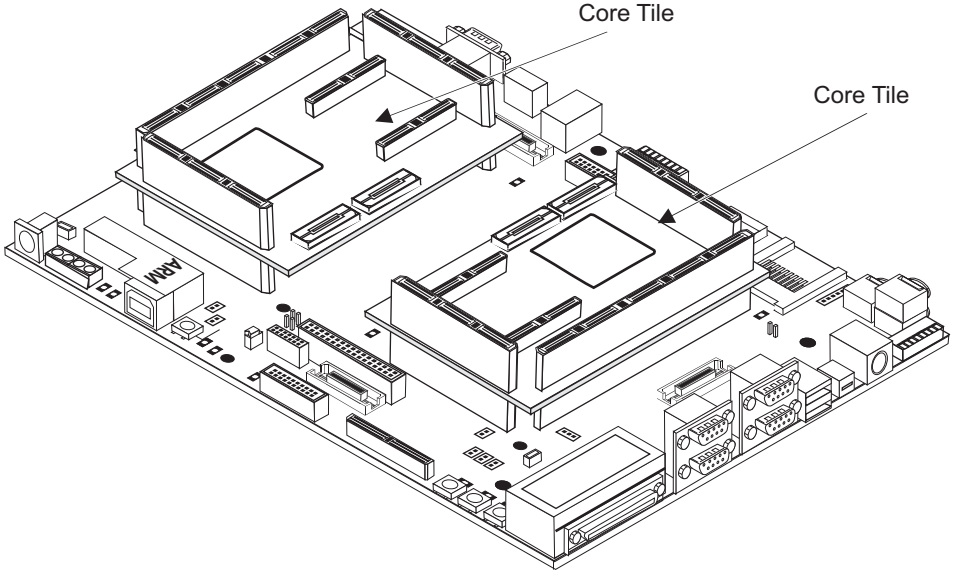

Figure 2-4 Multiprocessor system with two Core Tiles

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.3.1 Tile status messages displayed on the LCD

The firmware loaded by default into the baseboard FPGA detects the tiles present at power-on and outputs the status to the character LCD as listed in Table 2-5.

Caution

The default FPGA image is for the CT926EJ-S. See *Loading FPGA and PLD images* on page 2-20 for details on loading the FPGA image for other Core Tiles.

Table 2-5 Tile status messages displayed on character LCD

Tile site 1	Tile site 2	Message displayed
No tile fitted	No tile fitted	TS1 Fit CT926EJS TS2 Optional
No tile fitted	Core Tile or Logic Tile present	TS1 Fit CT926EJS TS2 present
CT926EJ-S Core Tile present	No tile fitted	TS1 OK CT926EJS TS2 Optional
CT926EJ-S Core Tile present	Core Tile or Logic Tile present	TS1 OK CT926EJS TS2 Optional

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.4 Connecting a Logic Tile

The Logic Tiles, such as the LT-XC2V6000, enable developing AMBA AHB, AXI, and APB peripherals, or custom logic, for use with ARM cores.

Press down firmly on the Logic Tile to ensure that the connectors mate properly.

Note

The design in the Logic Tile FPGA must implement logic to handle the AHB or AXI bus signals from the baseboard. See the application notes supplied on the CD (and on the ARM web site) for more details on using Logic Tiles.

Figure 2-5 shows a Logic Tile mounted on the baseboard.

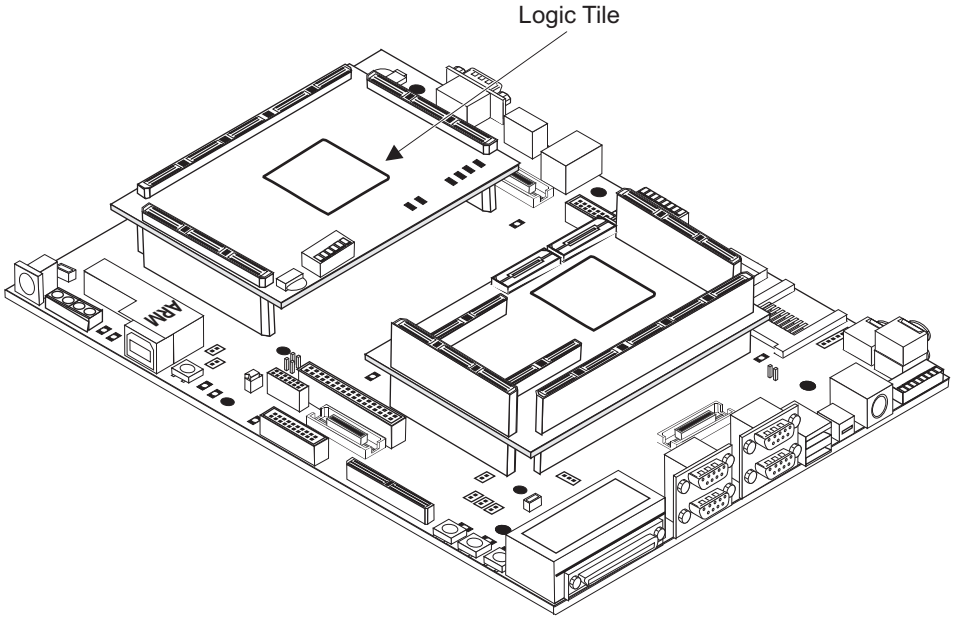

Figure 2-5 Logic Tile mounted in tile site 2

Use the JTAG interface on the baseboard to program the configuration flash in the Logic Tile or to directly load the Logic Tile FPGA image. For more information on JTAG signals, see *JTAG and USB debug port support* on page 3-83.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.5 Connecting expansion memory

To install a static memory PISMO expansion board:

1. Ensure that the baseboard is powered down.
2. Align the memory expansion board with the connectors on the baseboard as shown in Figure 2-6.
3. Press the module into the connector.

See Appendix E *PISMO Memory Expansion Boards* for more details on memory expansion boards.

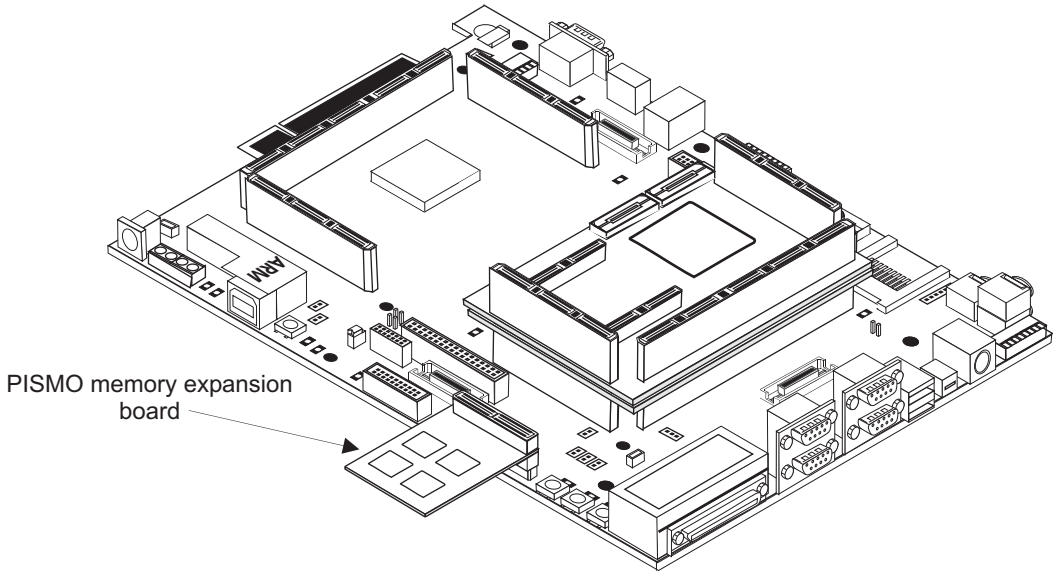

Figure 2-6 Adding PISMO memory

Note

If the configuration switches are set to boot from the PISMO memory and more than one PISMO board is fitted, the memory width is determined by the bottom-most PISMO board.

The PISMO connectors on the Core Tile are not used in the standard FPGA images. The Core Tile PISMO connectors can, however, be used if a custom image is made for the baseboard FPGA.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.6 Connecting JTAG debugging equipment

You can use JTAG debugging equipment and the JTAG connector to:

- Connect a debugger to the system processors and download programs to memory and debug them.
- Program new configuration images into the configuration flash, FPGA, and PLDs on the board. (You cannot program the normal flash from configuration mode.)

  The USB debug port can also be used to download device images. See *Procedure for progcards_usb.exe* on page 2-28.

The setup for using a JTAG interface with the baseboard is shown in Figure 2-8 on page 2-14.

Note

The USB debug port is used to load FPGA and PLD images and cannot be used to debug applications. For more details, see *JTAG and USB debug port support* on page 3-83.

To use the JTAG interface for debugging, the CONFIG slide switch must be in the OFF position as shown in Figure 2-7.

Note

Some early versions of the Emulator Baseboard have a jumper instead of a switch. Placing the jumper across the connectors is equivalent to the switch being in the ON position.

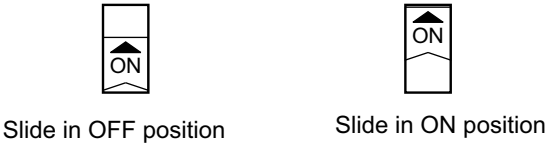

Figure 2-7 CONFIG slide switch

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

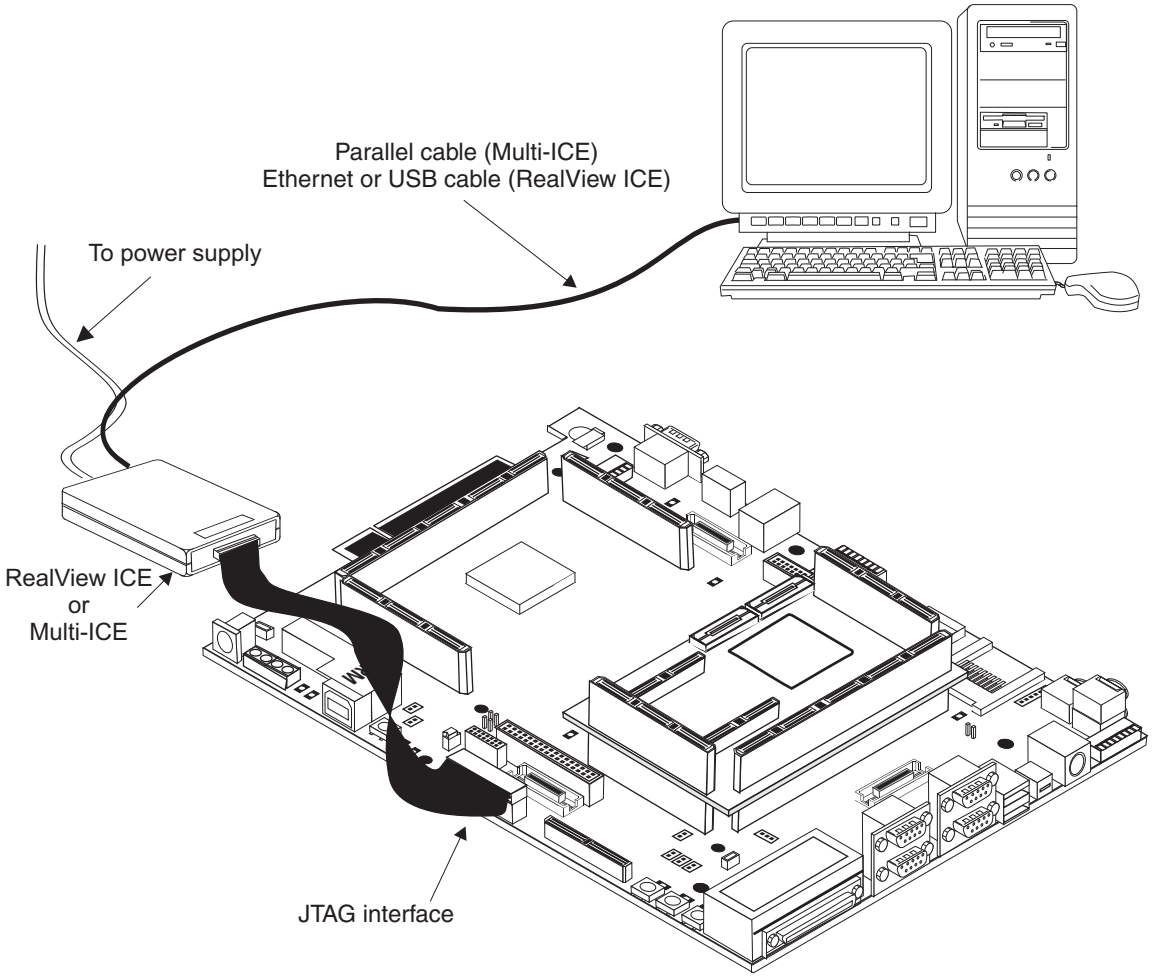

Figure 2-8 JTAG connection for debugger

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.7 Connecting Trace

If the Core Tile test chip contains an *Embedded Trace Macrocell* (ETM), a Trace connector is provided on the Core Tile. Use the JTAG connector on the baseboard to provide the JTAG signals that are required for controlling the ETM.

Note

If the test chip on the Core Tile does not contain an ETM, there are no trace connectors present on the Core Tile.

Trace using multiplexed trace packets (such as RealView Trace and Multi-Trace) require only one Mictor connector (Trace Port A). The Core Tile supports both multiplexed (one trace connector) and demultiplexed mode (two trace connectors). The second connector is present to support trace tools that use demultiplexed trace packets.

Figure 2-9 on page 2-17 illustrates a trace debugging setup with RealView Trace and RealView ICE. Figure 2-10 on page 2-18 illustrates a trace debugging setup with Multi-Trace and Multi-ICE.

Note

If you are using Core Tiles, use the Trace connector present on the Core Tile.

The baseboard has two trace connectors that are connected to the tile headers. Use these trace connections if a processor is instantiated in Logic Tiles mounted on the tile sites (Soft Macrocell Model of a CPU core). The trace size supported by the connectors is medium (16-bit packets).

For more details on using Trace, see *About trace* and the documentation supplied with your Trace hardware.

2.7.1 About trace

The components used for trace capture are:

ETM The Embedded Trace Macrocell is part of the test chip and monitors the ARM core buses and outputs compressed information through the trace port to a trace connector. The on-chip ETM contains trigger and filter logic to control what is traced.

Trace connector and adaptor board

The trace connector enables you to connect a TPA to the processor. The connector is a high-density AMP Mictor connector. The pinout for this connector is provided in Test and debug connections on page A-33.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The adaptor board buffers the high-speed signals between the Trace connector and the Trace Port Analyzer.

JTAG unit This is a protocol converter that converts debug commands from the debugger into JTAG messages for the ETM.

Trace Port Analyzer

The TPA is an external device (such as RealView Trace) that connects to the trace connector (through the adaptor board) and stores information sent from the ETM.

Note

Some processors contain *Embedded Trace Buffers* (ETB) and do not require a TPA. This processors use a different system to record and analyze the trace output. Refer to the documentation for the processor for details of using ETB instead of the TPA.

Debugger and Trace software

The debugger and trace software controls the JTAG, ETM, and Trace Port Analyzer. The trace software reconstructs program flow from the information captured in the Trace Port Analyzer.

Note

The trace and debug components must match the debugger you are using:

ARM eXtended Debugger (AXD)

AXD is a component of the *ARM Developer Suite* (ADS). Use AXD with Multi-ICE, Trace Debug Toolkit, and Multi-Trace.

ARM RealView Debugger (RVD)

RVD is a component of *RealView Development Suite* (RVDS). Use RVD with RealView ICE and RealView Trace or with Multi-ICE and Multi-Trace.

2.7.2 Connecting the Trace Port Analyzer to the baseboard

For RealView Trace, connect the TPA to the adaptor board and plug the adaptor into the Core Tile as shown in Figure 2-9 on page 2-17. RealView Trace requires a RealView ICE JTAG unit connected to the baseboard. The Ethernet and power supply cables connect to the RealView ICE unit.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

For Multi-Trace, connect the *Trace Port Analyzer* (TPA) to the buffer board and plug the adaptor into the Core Tile as shown in Figure 2-10 on page 2-18. Multi-Trace requires a Multi-ICE JTAG unit.

Note

The high-density cable from the RealView ICE box requires a buffer board to connect to the JTAG connector on the baseboard.

The low-density cable can be used to connect the RealView ICE box directly to the baseboard JTAG connector, but this interface operates at lower speed.

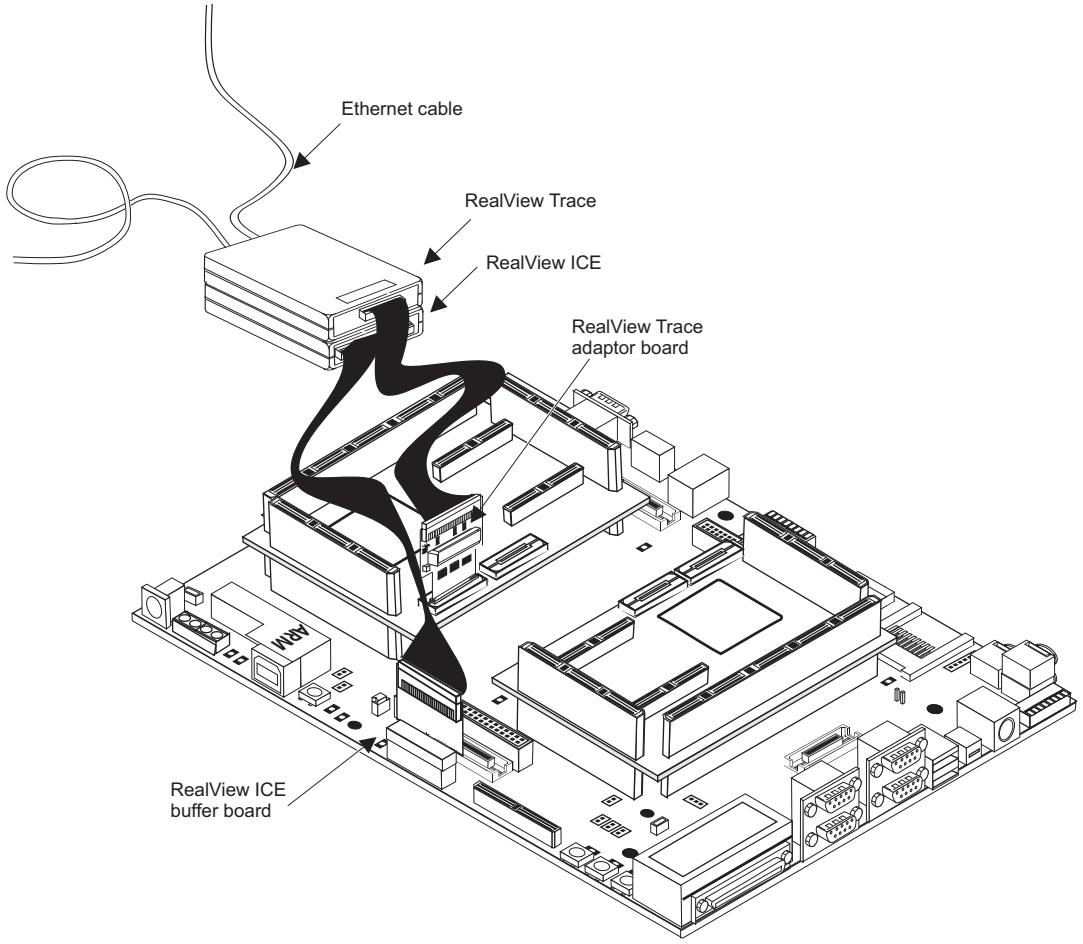

Figure 2-9 Trace connection with RealView Trace

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

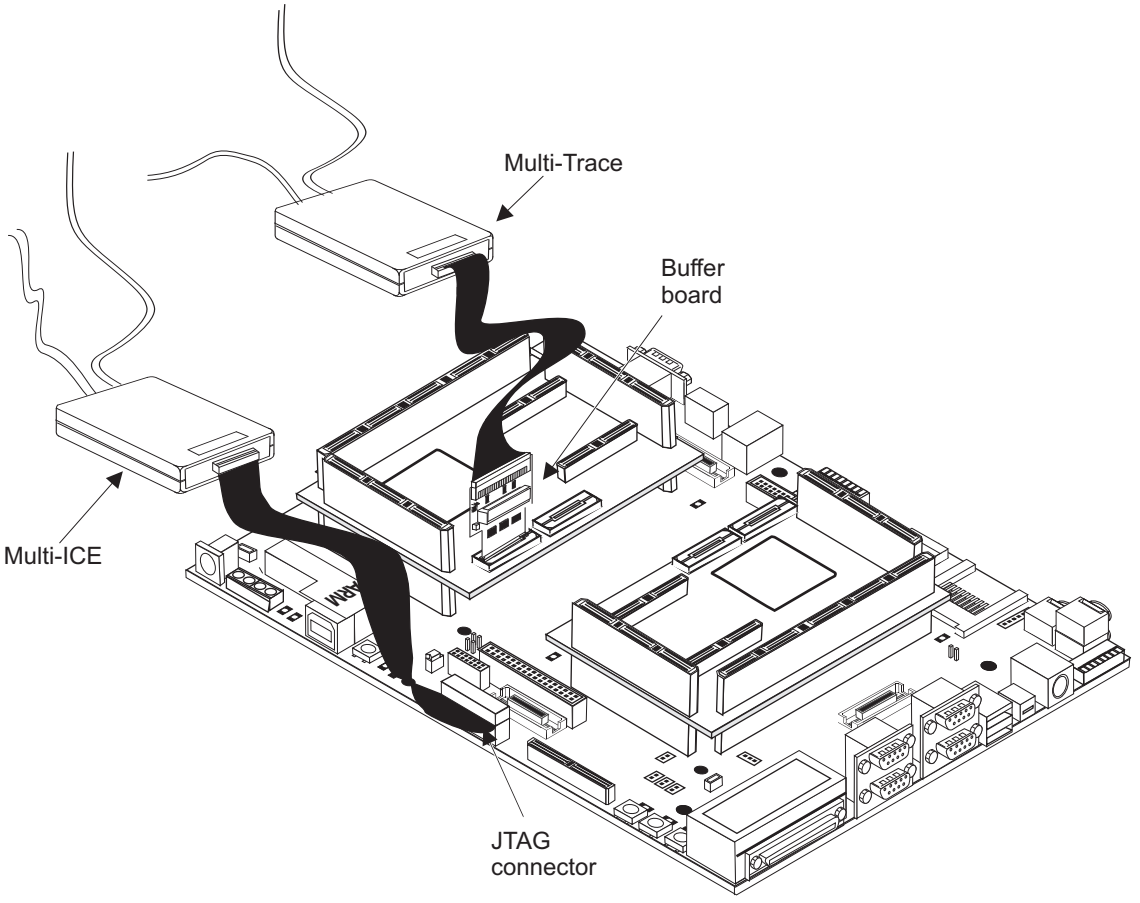

Figure 2-10 Trace connection with Multi-Trace

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.8 Supplying power

If the baseboard is not inserted into a PCI enclosure, you must connect the supplied 12VDC brick power supply to power socket J28 or an external bench power supply to the screw-terminal connector J27. See Figure 2-11.

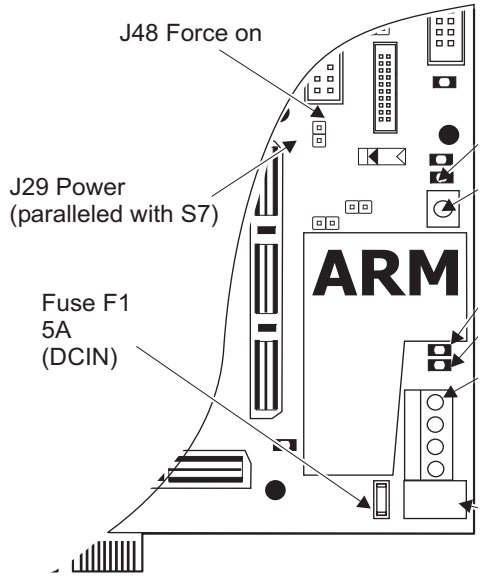

Figure 2-11 Power connectors

Note

If you are using the supplied brick power supply connected to J28, the Standby/power push button toggles the power on and off.

If you are using an external power supply connected to J27, or you are powering the board from the PCI backplane, the Standby/power switch is not used and power is controlled by shutting down the external power source.

Caution

You can use only one power source for the system. Use only one of the PCI connector, J27, or J28.

Do not, for example, use the PCI connector and J27 at the same time.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.9 Loading FPGA and PLD 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. 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 Core Tile as described in *Setting up the baseboard* on page 2-2.
3. Connect RealView ICE (or Multi-ICE) to the JTAG connector or use a USB cable to connect to the USB debug port as described in *Connecting JTAG debugging equipment* on page 2-13.

   Note

   If you are using Multi-ICE, you must start Multi-ICE server and load an appropriate configuration file.

   Progcards for USB or RealView ICE always uses auto-configuration.
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 2-21).

   Caution

   The baseboard is shipped with a test image for a CT926EJ-S Core Tile in the FPGA configuration flash. You must load a new FPGA image into the configuration flash before you can use the baseboard. Refer to the application note for your system configuration for specific instructions. If you change the type of Core Tile, you must load new images and a new Boot Monitor for that tile.

   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 2-39.
6. Run the `progcards` utility and load the image files into the FPGAs (see *Upgrading your hardware* on page 2-24). The board file selects the image files to load. Board files have a `.brd` extension.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.9.1 Board files

The CD includes both the `progcards` programming utilities and the images to load into the FPGAs and PLDs. See the Application Note on the CD for the combination of Core Tile that you are using.

Naming conventions for board files

The file name of a board file identifies the PCB, Core 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`.

coretile The name of the Core Tile that is used with this configuration. For example, `ctmpcore_li` identifies the CTMPCore Core Tile 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Naming conventions for image files

The images 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 for PLD programming and `.bit` is for FPGA programming file

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. The JTAG routing PLD is pre-programed at manufacture and does not appear in the programming scan chain.

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 Core Tile or Logic Tile.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.9.2 The progcards utilities

The progcards utilities are the primary method for programming FPGAs, configuration flash memory, and non-volatile PLDs. This utility is used during board manufacture and to carry out field upgrades.

progcards reads a description of the board JTAG scan chain and 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 2-1 Board file for baseboard with a MPCore tile

```
[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
```


Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.9.3 Upgrading your hardware

Use one of the progcards utilities and the board description (`*.brd`) files to load configuration images to the FPGA. There are three versions of the utility:

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_multiice.exe

`progcards_multiice.exe` uses Multi-ICE and the JTAG interface. It is a TAPOp program that runs on a PC host in a DOS window and communicates with the Multi-ICE server.

See *Procedure for progcards_multiice.exe* on page 2-26 for detailed instructions.

progcards_usb.exe

`progcards_usb.exe` uses the built-in 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 2-28 for detailed instructions.

Note

The latest version of the RVI firmware can be downloaded from the Technical Support area of the ARM web site.

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 box header on the platform baseboard. See Figure 2-8 on page 2-14.
3. Move the CONFIG switch S1 to the ON position (see *Connecting JTAG debugging equipment* on page 2-13). The orange CONFIG LED lights.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4. Turn on the power.
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 `progcards_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. The output is similar to the example transcript is shown in Example 2-2 on page 2-28.
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 baseboard power off and move the CONFIG switch to the OFF position.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

13. Set the configuration switches to match the boot option you are using (see *Setting the configuration switches* on page 2-4).

14. Power on the baseboard and use the Boot Monitor to load your application (see *Using the baseboard Boot Monitor and platform library* on page 2-32).

    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.

Procedure for progcards_multiice.exe

1. If it is not already installed, install the Multi-ICE server (revision 2.2.6 or later).

2. Connect the 20-way JTAG cable from the JTAG interface (Multi-ICE) to the box header on the platform baseboard. See Figure 2-8 on page 2-14.

3. Move the CONFIG switch S1 to the ON position (see *Connecting JTAG debugging equipment* on page 2-13). The orange CONFIG LED lights.

4. Turn on the power.

5. Start the Multi-ICE server. See the documentation supplied with the JTAG hardware for details on using the interface.

6. Select **File → Auto-Configure**. The display should be updated to show a number of FPGA and PLD devices in the scan chain.

   Note

   If auto-configuration fails, you must manually load an appropriate configuration. Select **File → Load → Configuration** and then choose the file in the `multi-ice` subdirectory that matches your hardware configuration.

   The configuration files automatically set the transfer timing (TCK) to 1MHz, therefore manually setting the transfer timing is not necessary.

7. Start a command window by selecting **Run** from the **Start** menu and entering `command` in the text box.

8. Open a command prompt window and change directory to the directory that contains board description (`*.brd`) files for the design to be programmed.

9. The command to start `progcards_multiice` depends on your hardware setup:

   - If you are using Multi-ICE and the server is running on a different machine than you are using to enter the commands, invoke `progcards_multiice` by entering:

     `C:\`*`Boardfiles`*`> progcards_multiice `*`server_name`*

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

- If the Multi-ICE server is running on the same machine, there is no need to supply a server name as the local host is the default. Invoke `progcards_multiice` by entering:

  ```
  C:\Boardfiles> progcards_multiice
  ```

10. `progcards_multiice` searches for board description (`*.brd`) files that match the JTAG scan chain shown on the Multi-ICE server window. All board descriptions matching the first part of the chain are presented as a menu and you can select the file to use.

    `progcards_multiice` runs through the steps required to completely reprogram the boards. An example transcript is shown in Example 2-2 on page 2-28.

11. To bypass programming a Logic Tile or the baseboard select the `Skip` option from the menu. `progcards_multiice` then looks for board description (`*.brd`) 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 baseboard power off and move the CONFIG switch to the OFF position.

13. Set the configuration switches to match the boot option you are using (see *Setting the configuration switches* on page 2-4).

14. Power on the baseboard and use the Boot Monitor to load your application (see *Using the baseboard Boot Monitor and platform library* on page 2-32).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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.

Example 2-2 Example transcript for Multi-ICE and progcards_muiltiice.exe

```
C:\BoardFiles> progcards_multiice
ARM Development Card Logic Programmer
Version 2.53
Attempting to connect to Multi-ICE server
Multi-ICE reports 4 TAP controllers
Several possible boards detected at TAP position 0:-
0: Quit progcards
1: Skip (eb)
2: EB (HBI-01409C) CTMPCore FPGA flash image 0 build28, Character LCD Mux build 3
Make a choice: 2
Step 1: PLD/SVF download of an1151/an151_eb_140c_xc2v6000_ct926umc_le_build3_mux_xc2c128_build2.svf
Progress: 100.00%
Step 2: FPGA download of an1151/an151_eb_140c_xc2v6000_via_build1.bit
Progress: 100.00%, Throughput: 23.19k/s, Frame: 1458
Step 3: Intel flash download of an1151/an151_eb_140c_ctmpcore_li_build7.bit
Progress: 100.00%, Throughput: 31.68k/s
Step 4: Intel flash verify against an1151/an151_eb_140c_xc2v6000_ctmpcore_li_build7.bit
Progress: 100.00%, Throughput: 29.46k/s, Errors: 0%
Step 5: PLD/SVF download of an1151/an151_eb_140c_mux_xc2c128_build1.svf
Progress: 100.00%
```

Procedure for progcards_usb.exe

1. If it is not already installed, install the USB Debug Direct Control software.

   Note

   Windows USB Debug drivers must be installed before using the `progcards_usb` utility. Information on installing the USB drivers can be found in `\boardfiles\USB_Debug_driver\readme.txt`.

2. Connect the USB cable from the host PC to the USB debug port (near the screw terminals) on the platform baseboard. See on page 2-30.
3. Move the CONFIG switch to the ON position (see *Connecting JTAG debugging equipment* on page 2-13).
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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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. The output is similar to the example transcript is shown in Example 2-2 on page 2-28.

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 platform 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 *Setting the configuration switches* on page 2-4).

12. Power on the board and use the Boot Monitor to load your application (see *Using the baseboard Boot Monitor and platform library* on page 2-32).

    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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

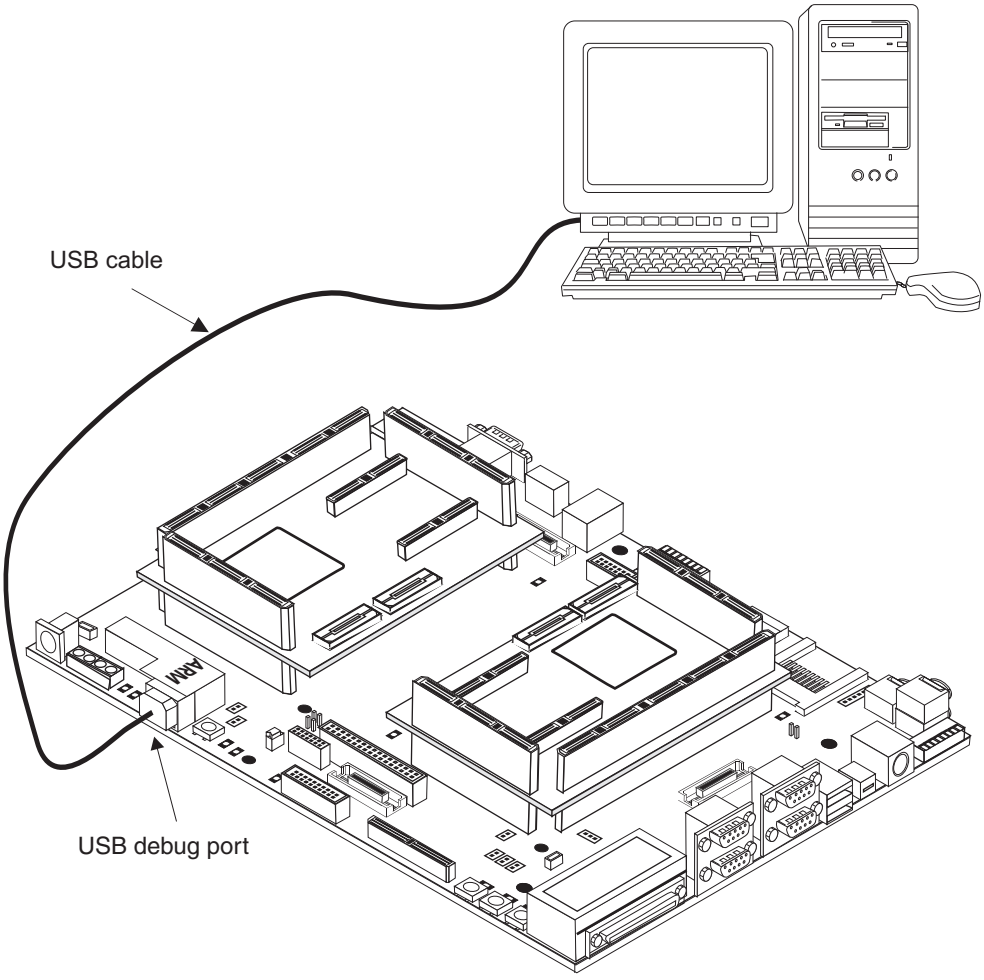

Figure 2-12 USB debug port connection for progcards_usb

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.9.4 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_multiice.exe` ensure that:
   a. the power LED on the Multi-ICE unit is on.
   b. the Multi-ICE server is revision 2.2.6 or later
   c. the scan chain displayed matches the board combination in configuration mode
   d. the **JTAG Bit Transfer Timing** (**TCK**) is set to 1MHz or less
3. If you are using `progcards_rvi.exe`, see the documentation supplied with `progcards_rvi`.
4. If you are using `progcards_usb.exe` ensure that:
   a. the USB cable is plugged into the USB debug port and not the USB user port
   b. the USB Debug drivers are installed on your machine.
5. Ensure that any Core Tiles and Logic Tiles are correctly stacked on the platform baseboard. If necessary, push them firmly to make a proper connection.
6. 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.10 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 thebaseboard.

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 S6-2 and S6-3
- if Boot Monitor configuration switch S6-1 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.

2.10.1 Boot Monitor configuration switches

The Boot Monitor application is typically loaded into the NOR or DOC flash memory and selected to run at power on. Follow the instructions in *Loading Boot Monitor into NOR flash* on page 2-39 or *Loading Boot Monitor into Disk-on-Chip* on page 2-41 for details of loading the boot flash with the image from the CD.

The setting of S6-1 determines how the Boot Monitor starts after a reset:

S6-1 OFF A prompt is displayed enabling you to enter Boot Monitor commands.

S6-1 ON The Boot Monitor executes a boot script that has been loaded into flash. (The boot script must be in the DOC flash). 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 or DOC flash memory. 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 Table 2-7 on page 2-34).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Output and input of text from STDIO for both applications and Boot Monitor I/O depends on the setting of S6-2 and S6-3 as listed in Table 2-6.

Table 2-6 STDIO redirection

S6-2	S6-3	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.

S6-2 and S6-3 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 2-43 for more details on I/O.

Note

Switch S6-2 and S6-4 to S6-8 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 switch S6 is implementation dependent.

2.10.2 Running the Boot Monitor

To run Boot Monitor and have it display a prompt to a terminal connected to UART0, set switch S6-1 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 or a different version of the monitor is required for a different Core Tile, see *Rebuilding the Boot Monitor or platform library* on page 2-44 for information on loading the monitor.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Boot Monitor commands

The command interpreter accepts user commands from the debugger console window or an attached terminal and carries out actions to complete the 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 []. 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).

Note

The commands available depend on the Core Tile fitted. For example, the ARM7TDMI Core Tile does not contain a cache and the `ENABLE CACHES` and `DISABLE CACHES` commands are not present.

Additional Boot Monitor commands might be available depending on the Core Tile fitted. Type `HELP` at the Boot Monitor prompt to display a full list of commands that are available for the specific baseboard and Core Tile configuration.

Table 2-7 lists the commands for the Boot Monitor.

Table 2-7 Boot Monitor commands

Command	Action
`@ script_file`	Runs a script file.
`ALIAS alias commands`	Create an alias command *alias* for the string of commands contained in *commands*.
`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 2-8 on page 2-36 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 2-7 Boot Monitor commands (continued)

Command	Action
COPY *file1 file2*	Copy *file1* to *file2*. For example, to copy the leds code from the PC to the Disk-on-Chip 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 Disk-on-Chip by inputting text. Press Ctrl-Z to end the file.
DEBUG	Enter the debug subsystem. Commands listed in Table 2-9 on page 2-37 can now be executed.
DELETE *filename*	Delete *file* from Disk-on-Chip.
DIRECTORY [*directory*]	List the files in a Disk-on-Chip directory. Files only accessible from semihosting cannot be listed.
DISABLE CACHES	Disable both the I and D caches.
DISPLAY BOOTSCRIPT	Display the current boot script.
ECHO *text*	Echo *text* to the current output device.
ENABLE CACHES	Enable both the I and D caches.
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 2-10 on page 2-38 for flash commands.
HELP	List the Boot Monitor commands.
LOAD *name*	Load the Disk-on-Chip image *name* into memory and run it.
QUIT	Alias for EXIT. Exit the Boot Monitor.
RENAME *old_name new_name*	Rename Disk-on-Chip file named *old_name* to *new_name*.
RUN *image_name*	Load the Disk-on-Chip image *image_name* into memory and run it.
SET BOOTSCRIPT *script_file*	Specify *script_file* as the boot script. If the run boot script switch S6-1 is ON, *script_file* will be run at system reset.
TYPE *filename*	Display the Disk-on-Chip file *filename*.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 2-8 lists the commands for the Configure subsystem.

Table 2-8 Boot Monitor Configure commands

Command	Action
`DISPLAY CLOCKS`	Display system clocks.
`DISPLAY DATE`	Display date.
`DISPLAY HARDWARE`	Display hardware information (for example, the FPGA revisions).
`DISPLAY TIME`	Display time.
`EXIT`	Exit the configure commands and return to executing standard Boot Monitor commands.
`HELP [`*`command`*`]`	List the configure commands. If a command is specified, help for that command is output.
`QUIT`	Alias for `EXIT`. Exit the Configure commands and return to standard Boot Monitor commands.
`SET CLOCK 0` *`frequency`*	Set frequency for clock 0. **Caution** Clock 0 is the reference for the test chip present on the attached Core Tile. Clock 0 can only be directly modified if the PLL on the test chip is bypassed. See the application note for your Core Tile for more information on setting the system clock.
`SET CLOCK 1` *`frequency`*	Set frequency for clock 1. **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 and the settings of switch S8 (see *Setting the configuration switches* on page 2-4).
`SET CLOCK 2` *`frequency`*	Set frequency for clock 2.
`SET CLOCK 3` *`frequency`*	Set frequency for clock 3.
`SET CLOCK 4` *`frequency`*	Set frequency for clock 4.
`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`*

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 2-9 lists the commands for the Debug subsystem.

Table 2-9 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.
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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 2-10 lists the commands for the NOR Flash subsystem.

Table 2-10 Boot Monitor NOR flash commands

Command	Action
`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 if it is stored in NOR flash. See *Loading Boot Monitor into NOR flash* on page 2-39.
`EXIT`	Exit the flash commands and return to executing standard Boot Monitor commands.
`HELP`	List the flash commands. Entering `HELP` followed by a command displays help for that command.
`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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 2-10 Boot Monitor NOR flash commands (continued)

Command	Action
`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_EB.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.

2.10.3 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.

Note

The Boot Monitor is normally located in Disk-on-Chip flash instead of NOR flash (see *Loading Boot Monitor into Disk-on-Chip* on page 2-41). You can, however, load the Boot Monitor into NOR flash instead of Disk-on-Chip flash if this is required for a specific application.

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

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2. Set all S8 switches to OFF. (Setting switches S8[4:1] to OFF selects booting from NOR flash.)

   Set all S6 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 AXD, from the Command Line Interface enter:
     `Debug > Obey EB_DDR_Init_axd.li`
   - For RVD, select **Tools → Include Commands From File**
     Select **EB_DDR_Init_rvd.li**

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_EB.axf`
   b. at the Boot Monitor prompt enter:
      `>FLASH`
      `Flash> LIST IMAGES`
      lists images currently in flash
      `Flash> ERASE IMAGE Boot_Monitor_EB.axf`
      erases the current version of boot monitor
      `Flash> WRITE IMAGE` *`path`*`\Boot_Monitor_EB.axf`
      where path is the *`path`* to the version of the Boot Monitor that is being loaded.

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.

To load the Boot Monitor into Disk-on-Chip instead of into NOR flash, see *Loading Boot Monitor into Disk-on-Chip* on page 2-41.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.10.4 Loading Boot Monitor into Disk-on-Chip

The Disk-on-Chip interface to the NAND flash emulates a disk drive. Use the Disk-on-Chip utility `doc_configure.axf` to format the NAND flash and load the Boot Monitor as the boot file as follows:

1. Power off the board
2. Set all S6 and S8 switches to OFF.
3. Connect a debug cable to the JTAG box header.
4. Power on the board.
5. Connect the debugger to the target
   - For AXD, from the Command Line Interface
     `Debug > Obey EB_DDR_Init_axd.li`
   - For RVD, select **Tools → Include Commands From File**
     Select **EB_DDR_Init_rvd.li**

   DRAM is now initialized and the memory is remapped.
6. From the debugger, load and execute the file `doc_configure.axf`

   (See *Using the Disk-on-Chip configure utility program* on page 2-42 for more details on the configure utility.)
7. If formatting is required, use the utility to format the Disk-on-Chip. At the prompt enter:

   ```
   Config> FORMAT
   ```

   After a short delay, a message displays indicating that formatting is complete.

   Caution

   Formatting the Disk-on-Chip erases all files present on the NAND flash. The Disk-on-Chip is formatted at manufacture. Only reformat if the flash has become corrupted.

   The `FORMAT` command creates two partitions: A binary partition that holds the boot files (SPL & Boot Monitor) and a single FAT partition. When the board is booted the platform mounts the FAT partition and allocates `B:` as the drive letter.

   File accesses routines such as `fopen()` must specify `B:` to use the Disk on Chip. Otherwise the routines use semihosting attempt to access the file if the boot monitor is being run from AXD or RVD. If the boot monitor is being run standalone, the file access fails.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

The FAT file system on the Disk on Chip device only supports 8.3 format file names.

8. Load the initial program loader files for the Disk-on-Chip. At the Boot Monitor prompt enter:

   ```
   Config> WRITE IPL path doc_ipl.axf
   ```

   After a short delay, a message displays indicating that the loader has been successfully programmed to the Disk-on-Chip.

9. Load the secondary program loader files for the Disk-on-Chip. At the prompt enter:

   ```
   Config> WRITE SPL path doc_spl.axf
   ```

   After a short delay, a message displays indicating that the secondary loader has been successfully programmed to the Disk-on-Chip.

10. Load the Boot Monitor as the boot program. At the prompt enter:

    ```
    Config> WRITE BOOT path\Boot_Monitor_EB.axf
    ```

    After a short delay, a message displays indicating that the Boot Monitor has been successfully programmed to the Disk-on-Chip.

11. Loading the images into the Disk-on-Chip NAND flash takes a few minutes to complete. Wait until the prompt is displayed again before proceeding.

12. Verify that the boot file has been copied to the Disk-on-Chip boot region by entering:

    ```
    Config> LIST
    ```

13. Turn the board off.

14. Set switch S8-1 to ON to select DOC as boot memory.

15. Turn the board on.

2.10.5 Using the Disk-on-Chip configure utility program

The command interpreter in `configure.axf` accepts user commands from the debugger console window and carries out actions to complete the commands on the Disk-on-Chip subsystem.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 2-11 lists the commands for the Configure utility.

Table 2-11 Configure utility commands

Command	Action
`FORMAT`	Format the Disk-on-Chip. All files will be deleted.
`LIST`	List all the boot images on the Disk-on-Chip.
`WRITE BOOT` *`filename`*	Load *`filename`* and place the image in the Boot Monitor area of the Disk-on-Chip. For example: `WRITE BOOT Boot_Monitor_EB.axf`
`WRITE IPL` *`filename`*	Load *`filename`* and place the image in the Initial Program Loader area of the Disk-on-Chip. For example: `WRITE IPL doc_ipl.axf`
`WRITE SPL` *`filename`*	Load *`filename`* and place the image in the Secondary Program Loader area of the Disk-on-Chip. For example: `WRITE SPL doc_ipl.axf`
`EXIT`	Exit the Configure utility.
`HELP`	List the Configure utility commands. Entering `HELP` followed by a command displays help for that command.
`QUIT`	Alias for `EXIT`. Exit the Configure utility.

2.10.6 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`
- `:CHARLCD`.

The STDIO calls are redirected within `retarget.c`. Redirection depends on the setting of switch S6, see *Boot Monitor configuration switch* on page 2-7.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.10.7 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 NAND flash:
     `> CREATE myscript.txt ; put any startup code here RUN file_name`
   - If your image is in NOR flash, enter the flash subsystem before running:
     `> 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.
3. Verify the file was entered correctly by typing:
   `>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:
   `>SET BOOTSCRIPT myscript.txt`
5. Set S6-1 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.

2.10.8 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

After rebuilding the Boot Monitor, load it into either NOR flash or the NAND flash Disk-on-Chip, see *Loading Boot Monitor into NOR flash* on page 2-39 and *Loading Boot Monitor into Disk-on-Chip* on page 2-41.

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 2-46).

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`.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.10.9 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 2-44 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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 2-12.

Table 2-12 Platform library options

Option	Description
`_platform_option_no_cache`	Stops the cache from being enabled by default
`_platform_option_no_diskonchip`	Disables Disk on Chip support and stops the driver code from being loaded
`_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 the example below. 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Example 2-3 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 2-43 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 2-49 or on *Running an image from Disk-on-Chip* on page 2-51 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.10.10 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 the Disk on Chip or Flash 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 Disk on Chip and character I/O devices using the mechanisms described in *Redirecting character output to hardware devices* on page 2-43. For example, open the character LCD by using the special file name `:CHARLCD`.

There are no specific tools requirements. Images built with ADS or RVDS 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 Core Tile and run.

2.10.11 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 2-46 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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`*

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

2.10.12 Running an image from Disk-on-Chip

To run an image from the NAND flash Disk-on-Chip:

1. Build and link the application as described in *Loading and running an application from NOR flash* on page 2-49.

   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 must be programmed into Disk-on-Chip using the Boot Monitor.

   Connect a debugger and use semihosting to load the file into NAND flash:

   `>COPY C:\software\elf_file_name file_name`

3. To run the image manually, from the debugger, or terminal connected to UART0, type:

   `>RUN file_name`

2.10.13 Using the Network Flash Utility

The *Network Flash Utility* (NFU) is supplied with V2.0 of the firmware. This utility uses the TFTP protocol to access files over the Ethernet network. You can copy files to Disk on Chip or program them 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 on setting the Boot Monitor configuration switches to force the console to use either UART-0 or the LCD and keyboard. (See *Boot Monitor configuration switch* on page 2-7 for details.)

      Note

      The debugger console cannot be used because the semihosted console I/O is blocking.

   b. Start the NFU utility.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

It typically takes several seconds for NFU start. Do not enter any commands until the prompt is displayed.

2. Use the DHCP protocol 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 DOC, enter:

   ```
   flash write image n:/hello.axf
   ```

NFU commands

The NFU supports most of the standard Boot Monitor commands and adds a new `MANAGE` submenu.

The NFU commands are listed in Table 2-13.

Table 2-13 NFU commands

Command	Action
`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 Disk-on-Chip 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 Disk-on-Chip by inputting text. Press Ctrl-Z to end the file.
`DELETE` *`filename`*	Delete *`file`* from Disk-on-Chip.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 2-13 NFU commands (continued)

Command	Action
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 2-10 on page 2-38 for flash commands.
HELP	Lists the NFU commands.
MANAGE	Enter the network management submenu. See Table 2-14 for MANAGE commands.
QUIT	Alias for EXIT. Exit the Boot Monitor.
RENAME *old_name new_name*	Rename Disk-on-Chip file named *old_name* to *new_name*.
TYPE *filename*	Display the Disk-on-Chip file *filename*.

The MANAGE submenu listed in Table 2-14 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 submenu.

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 2-14 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 2-14 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 submenu. The commands listed in Table 2-13 on page 2-52 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 submenu.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 2-14 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, if attempts to run the NETSTART.BAT file on the Disk on Chip. 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 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Chapter 3
Hardware Description

This chapter describes the baseboard hardware. It contains the following sections:

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.1 Tile headers and signal interconnects

The baseboard has two tile sites that enable adding one or more Core Tiles or Logic Tiles. The signals from the tile sites connect to the FPGA and peripherals on the baseboard. There are also interconnections between the two tile sites.

Note

If a tile site is not fitted with a Core Tile, the function of the signals on HDRX and HDRY can be modified by user for any purpose. Such a change would, of course, require a custom design to be implemented in the baseboard FPGA.

Some of the signals on the tile sites use electronic switches to control the routing (see *Header interconnect switches* on page 3-11).

3.1.1 Buses

The bus usage on the tiles and baseboard depends on the type and combination of tiles. HDRX has the master bus signals for both Core Tiles and Logic Tiles. For Logic Tiles, HDRY has the slave bus signals.

The signals on the header connectors depend on the bus present:

AXI For a Logic Tile, HDRX implements the master AXI bus and HDRY implements an AXI slave bus. For a Core Tile, HDRX implements the AXI master bus. A master or slave bus might be implemented on HDRY.

AXI buses have too many signals to fit on HDRX or HDRY. Multiplexors in the Core Tile reduce the number of signals required on the header connectors as shown in Figure 3-2 on page 3-5. Table 3-1 on page 3-6 lists the timing specifications for the multiplexor. Some AXI signals are extremely time critical and are not passed through the multiplexors.

AHB For a Logic Tile, HDRX implements a master AHB bus and HDRY implements a AHB slave bus. See Figure 3-1 on page 3-4. For a Core Tile, HDRX implements an master bus. A master or slave bus might be implemented on HDRY.

ARM7 HDRX on the tile site connected to the Core Tile implements an ARM7TDMI or ARM7-S bus. For a Logic Tile, HDRX implements an AHB master bus and HDRY implements am AHB slave bus.

The organization is the same as shown for the AHB bus in Figure 3-1 on page 3-4, but the control signals and the data bus interconnects on the Core Tile are different. The baseboard FPGA provides an AHB wrapper for the ARM7 bus variants.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Core Tiles with AHB buses have interconnect switches that control the routing of the read and write data buses from the tile header to the processor pins. The baseboard, however, has only one data bus on HDRX so the HWDATA and HRDATA signals are always multiplexed as HDATA. HDRY has enough signals to enable the Logic Tile to use separate HWDATA and HRDATA buses, but the example implementation also uses multiplexed HDATA on HDRY to simplify implementing the interfaces.

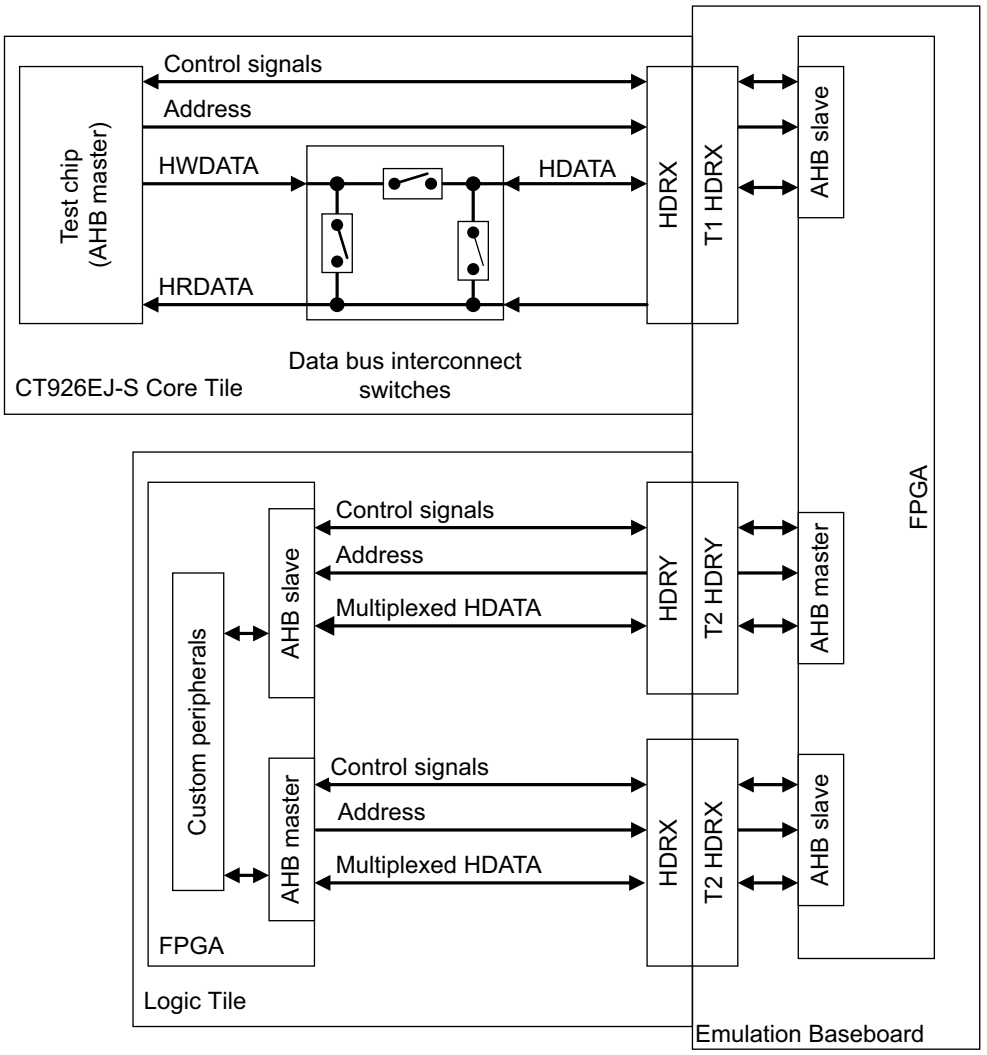

Figure 3-1 Example of an AHB bus interface

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The Core Tiles with AXI buses have multiplexors that reduce the number of signals used.

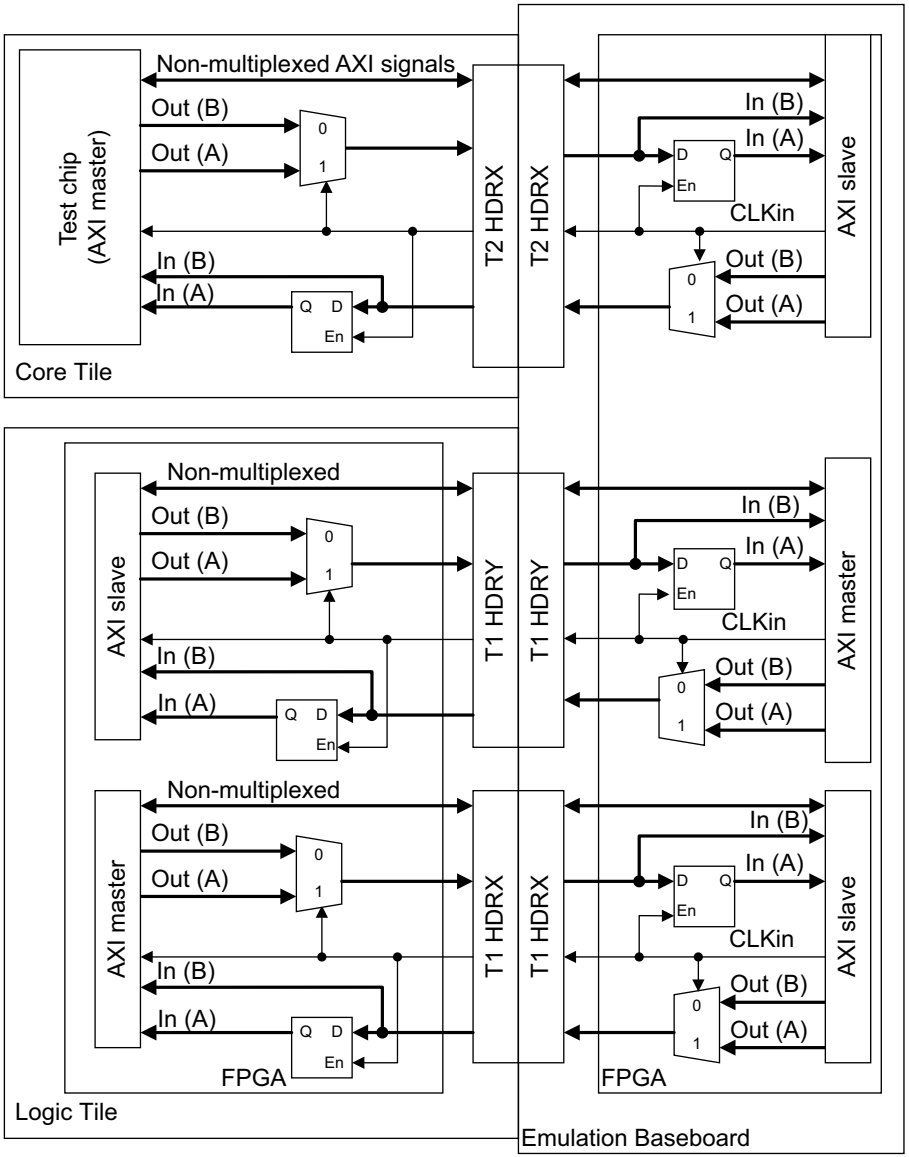

Figure 3-2 Example of a multiplexed AXI bus interface

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

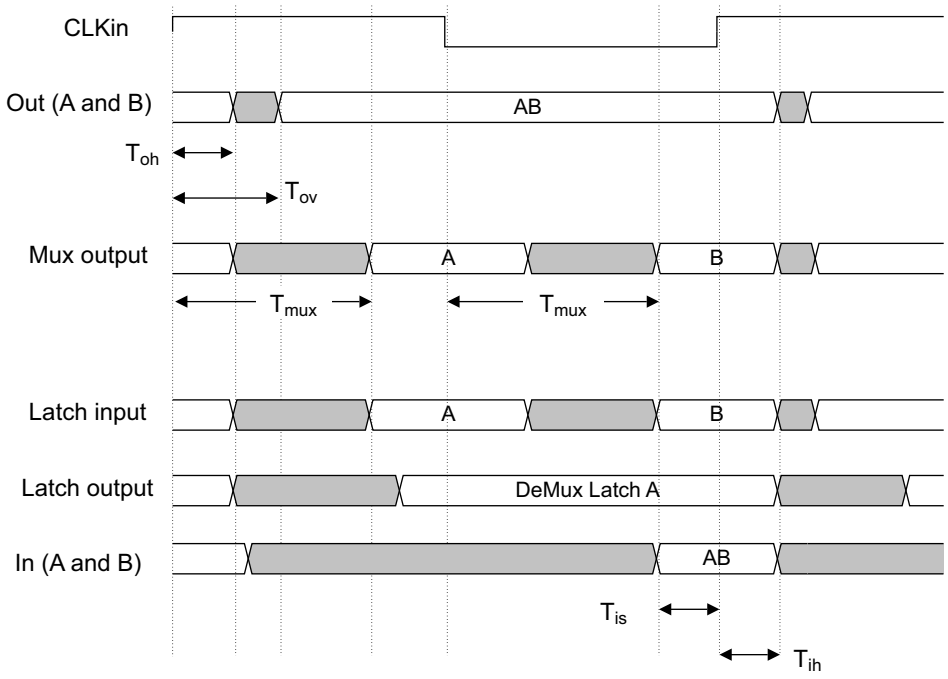

Figure 3-3 AXI multiplex timing

Table 3-1 lists the timing specification (with 50MHz clock assumed) for the AXI multiplexor and demultiplexor. **CLKin** is the clock driven to the test chip from the board. All I/O timing must be with respect to this clock.

Table 3-1 AXI multiplexor timing

Signal	Time	Description
Toh (min.)	0ns	Output hold
Tov (max)	2ns	Output valid
Tis (max)	2ns	Input setup
Tih (max)	0ns	Input hold
Tmux (max)	6ns	Multiplexor and board delay

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Bus signals on headers

The tile buses support AHB, AXI, custom, and peripheral buses as shown in Table 3-2.

Table 3-2 Bus usage on tile sites

Tile	Header	Description
Core Tile	HDRX	Master bus. For Core Tiles that have AHB buses, the HRDATA and HWDATA are tri-state multiplexed onto a single bidirectional HDATA bus. For Core Tiles that have AXI buses, the signals are time-domain multiplexed to reduce the width of the output and input buses.
	HDRY	HDRY is used for control signals for Core Tiles that have AHB buses. HDRY might implement a second master or a slave for Core Tiles that use multiple AXI buses.
	HDRZ	Sideband signals such as I/O signals, JTAG, clocks, and custom interconnection signals. The use of these signals depends on the specific Core Tile.
Logic Tile	HDRX	Master bus For AHB, the two read and write data buses are tri-state multiplexed onto a single bidirectional HDATA bus. For AXI, the output and input buses are each time-domain multiplexed to reduce the bus width.
	HDRY	Slave bus For AHB, the two read and write data buses are tri-state multiplexed onto a single bidirectional HDATA bus. For AXI, the output and input buses are each time-domain multiplexed to reduce the bus width.
	HDRZ	Sideband signals and I/O. The use of these signals depends on the specific Core Tile and the design implemented in the Logic Tile FPGA.

Figure 3-4 on page 3-8 shows a basic system consisting of one Core Tile. The FPGA on the baseboard implements interface devices and provides a bus interface to the Core Tile.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

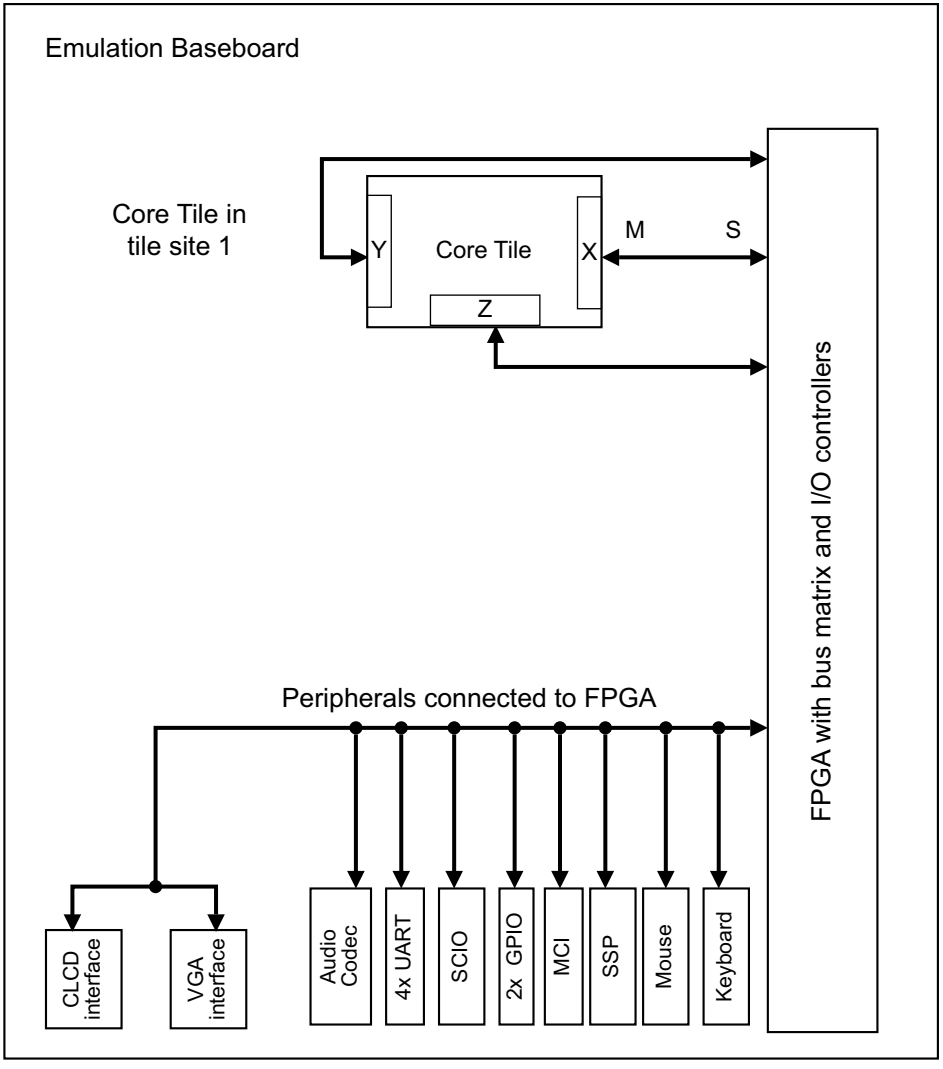

Figure 3-4 Core Tile in site 1

Figure 3-5 on page 3-9 shows a system consisting of one Core Tile and a Logic Tile. The FPGA provides a bus interface to the Core Tile.

The peripherals can be implemented either in the FPGA on the Logic Tile or the FPGA on the baseboard (see *Header interconnect switches* on page 3-11).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

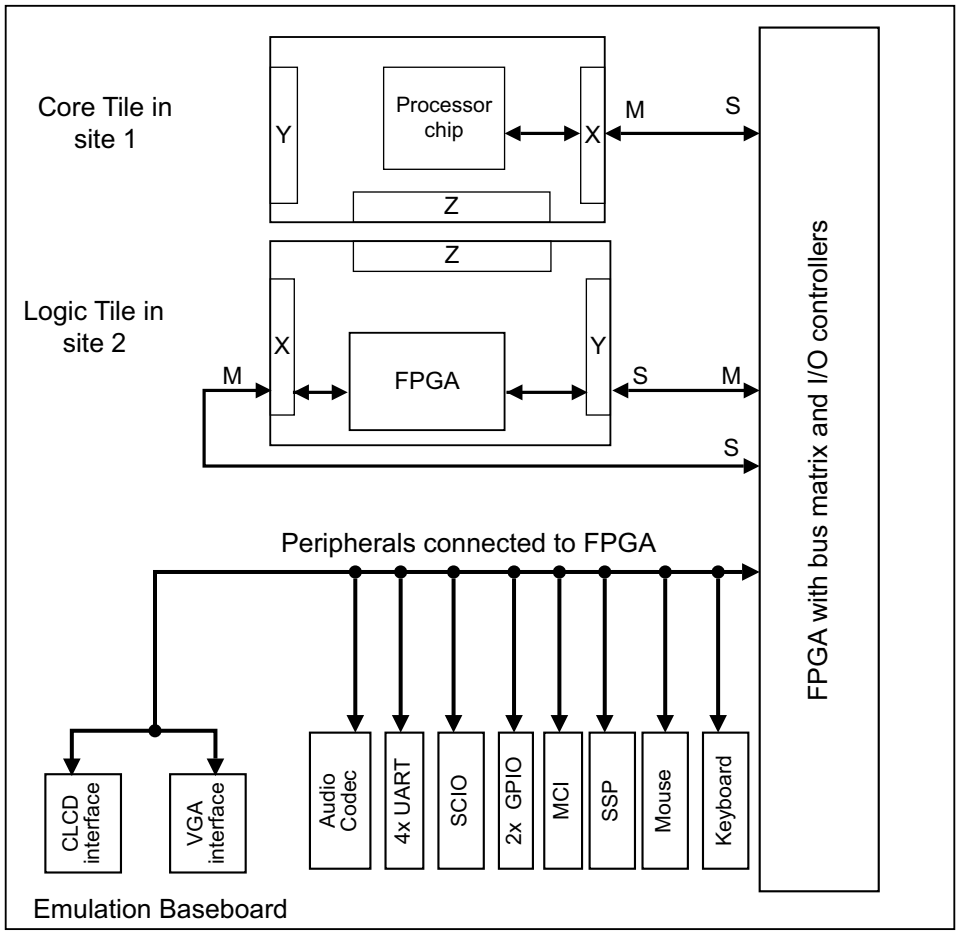

Figure 3-5 Core tile in site 1 and Logic Tile in site 2

Figure 3-6 on page 3-10 shows a dual Core Tile system. The FPGA implements interface devices and provides the bus interfaces to the Core Tiles.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

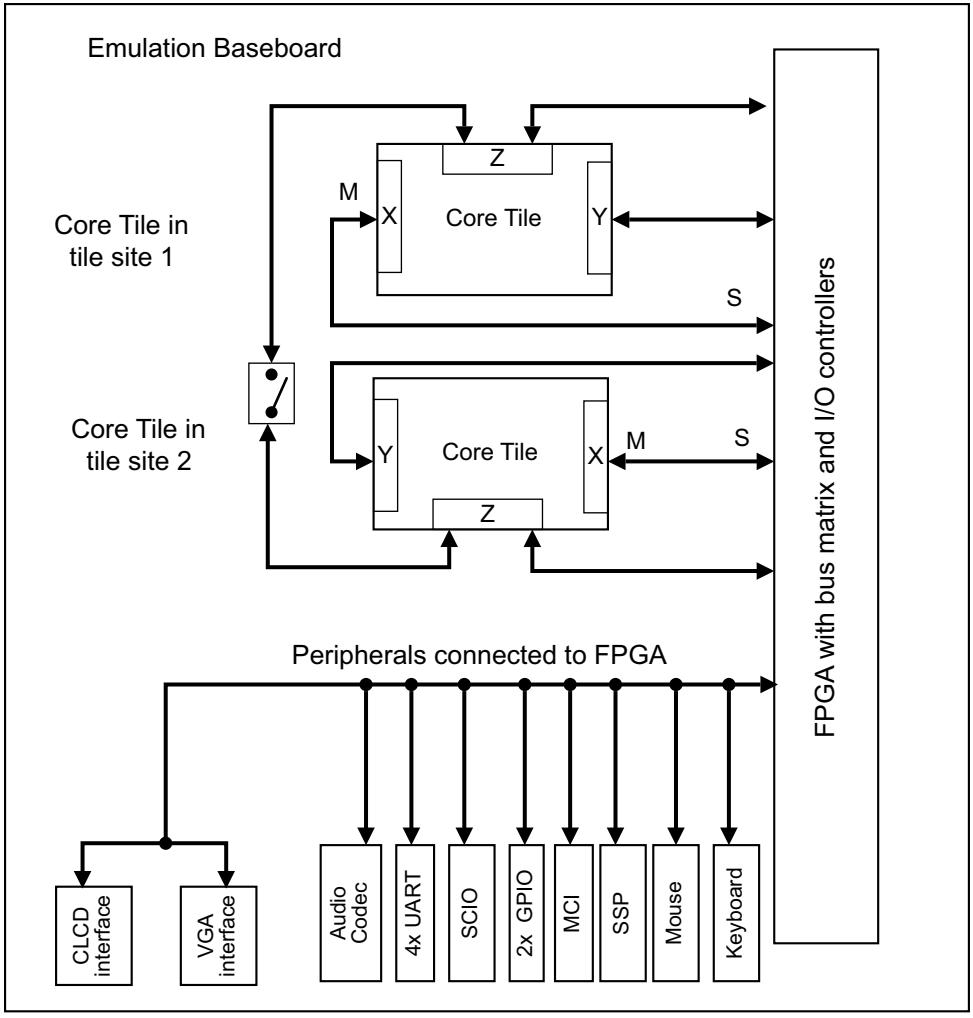

Figure 3-6 Dual Core Tile system

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.1.2 Header interconnect switches

There are electronic switches that route some of the header Z signals:

Sideband signals

These header Z (HDRZ) signals can be directly connected between the two tile sites and allow communication between tiles. They do not connect to other logic on the baseboard.

CLCD signals

The CLCD interface logic and connectors are normally connected to the FPGA. The CLCD can, however, be controlled from either of the tile sites. If neither tile is connected to the CLCD output, the HDRZ signals can be connected together to function as sideband signals.

Peripheral signals

The UARTs, GPIO, SSP, MCI, KMIs, and AACI peripherals are normally connected to the FPGA. These peripherals can, however, be controlled from either of the tile sites. If neither tile is connected to the CLCD output, the header Z signals can be connected together to function as sideband signals.

Unlike the CLCD signals, these peripheral signals are always connected to the FPGA. If a tile drives these signals, the FPGA output signals must be tri-stated.

The controls for the electronic switches come from the control multiplexor PLD. A serial interface implemented in both the PLD and the FPGA copies the value of the SYS_IOSEL register value from the FPGA to the PLD.

See *Peripheral I/O select, SYS_IOSEL* on page 4-26 for the software interface to the control signals.

Sideband signals

Table 4-22 on page 4-28 lists HDRZ signals that can be directly connected together. These signals are only used for direct communication between tiles and do not connect to other logic on the baseboard.

The function of the signal depends on the tiles present and the nature of any custom design implemented in the tile FPGAs.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

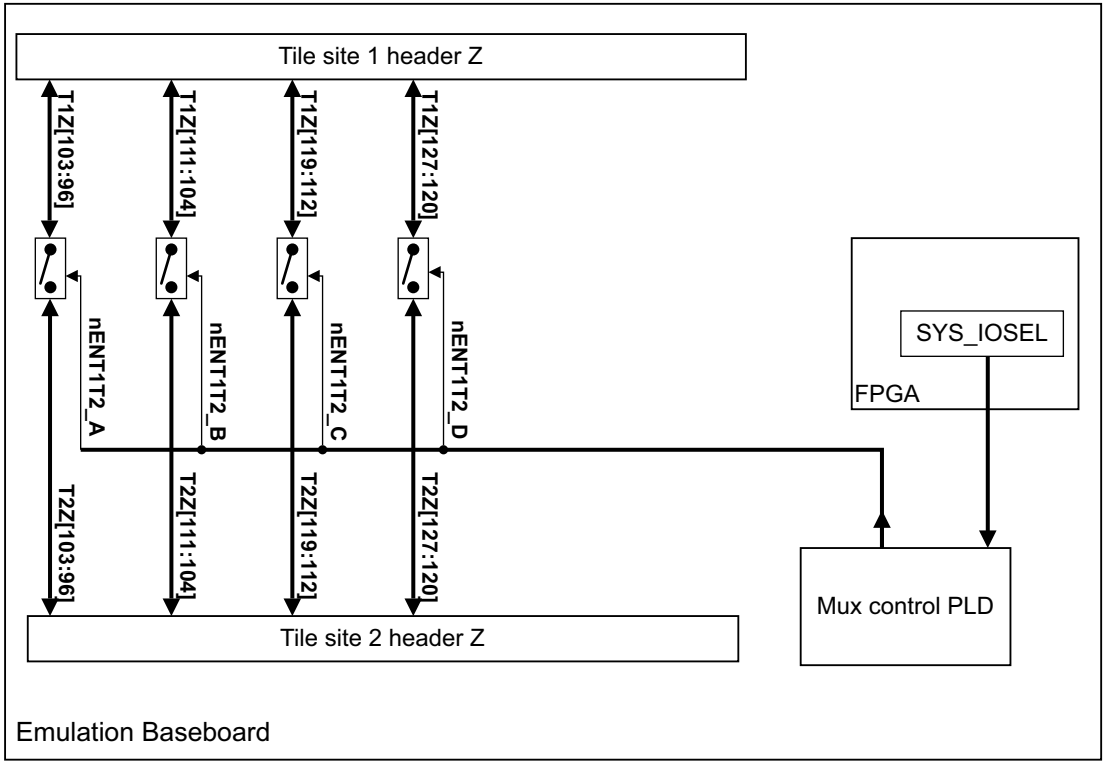

Figure 3-7 Header Z routing switches

CLCDC routing switches

Figure 3-8 on page 3-13 shows the routing logic for the CLCD signals. The signals listed in Table 4-22 on page 4-28 control the CLCD signal routing. These switches can be used when a tile (typically a Logic Tile in tile site 2) implements the *Color LCD Controller* (CLCDC). If a tile is not used as the CLCDC, the header Z signals can be interconnected to enable direct communication between the two tiles.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

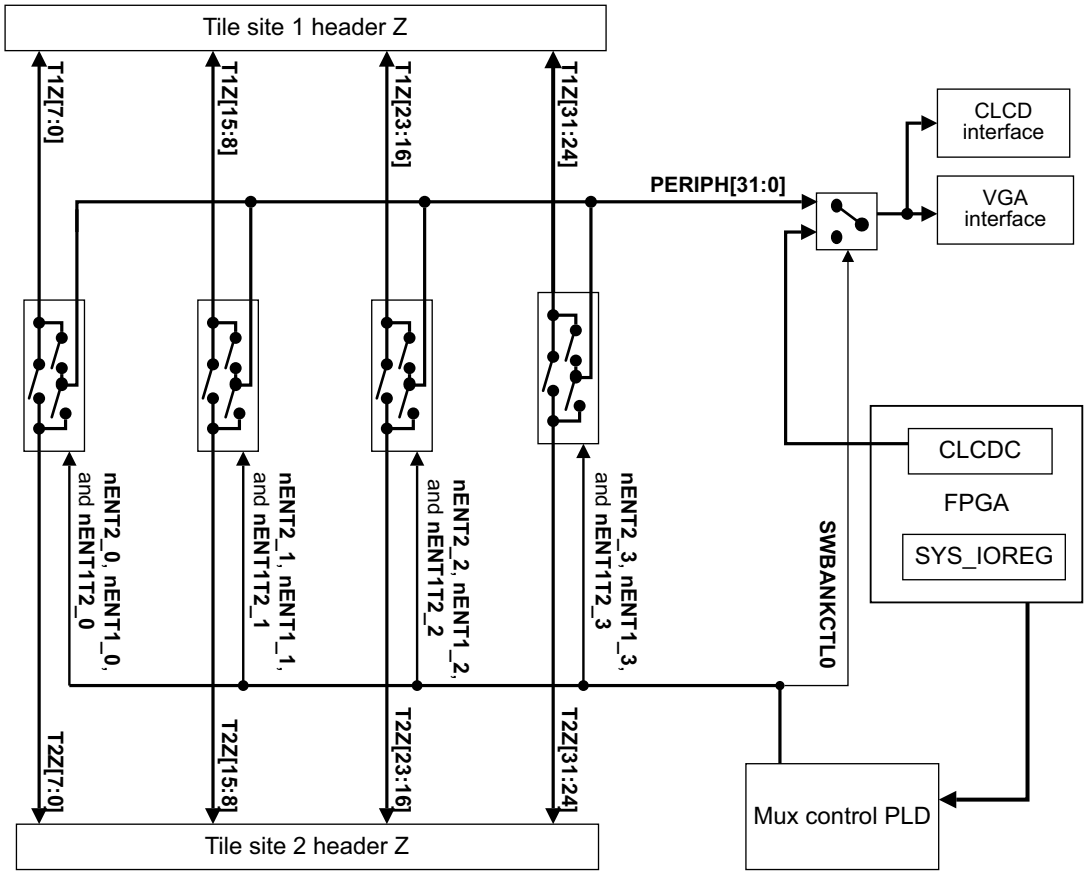

Figure 3-8 HDRZ switches for CLCDC

Note

Because the signals are spread over four banks, all four banks must be switched if a tile implements the CLCDC. The register values must be either `b01010101` or `b10101010`. In addition to setting the control signals for the tile routing, signal **SWBANKCTL0** must also be set in order to connect the **PERIPH[31:0]** signals to the CLCD interface logic and connectors.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Peripheral switches

Figure 3-9 on page 3-15 shows the peripherals that can be controlled from either of the tile sites. If neither tile is connected to peripherals, the corresponding header Z signals can be connected together to function as sideband signals. See Table 4-22 on page 4-28 for details of the mapping between the bits in the SYS_IOSEL register and the switch state.

Some control signals select the source for a single peripheral (for example **nENTx_4** for UART0). Other signals select several peripherals at once (for example **nENTx_5** for both UART1 and UART2).

Caution

Unlike the CLCDC signals, these peripheral signals are always connected to the FPGA. If a tile drives these signals, logic inside the standard FPGA image monitors these signals and disables the FPGA outputs if a tile drives the signal. If you create a custom image, ensure that the FPGA output signals are tri-stated.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

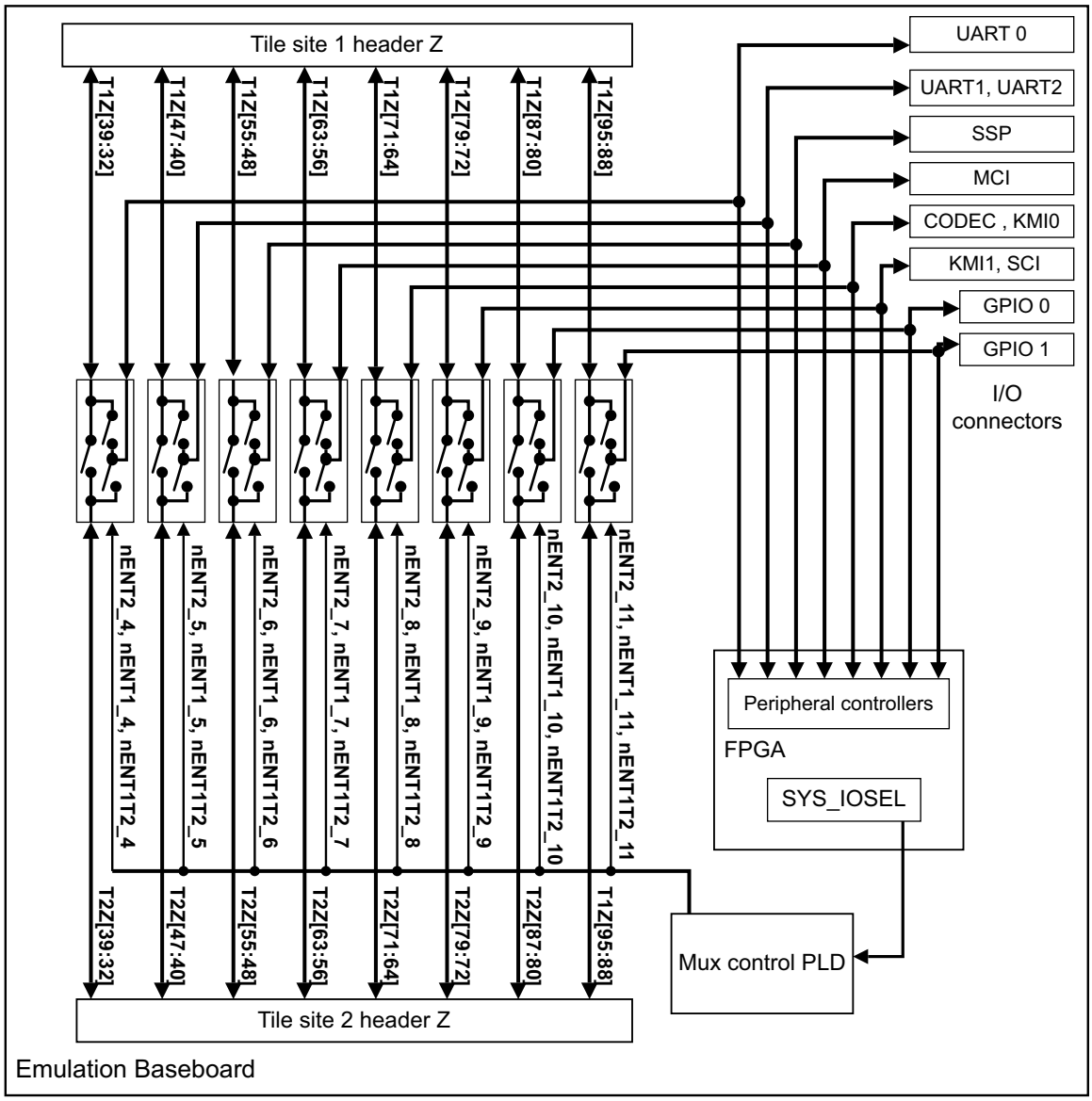

Figure 3-9 HDRZ switches for peripherals

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.2 FPGA

Figure 3-10 shows the architecture of the FPGA on the baseboard.

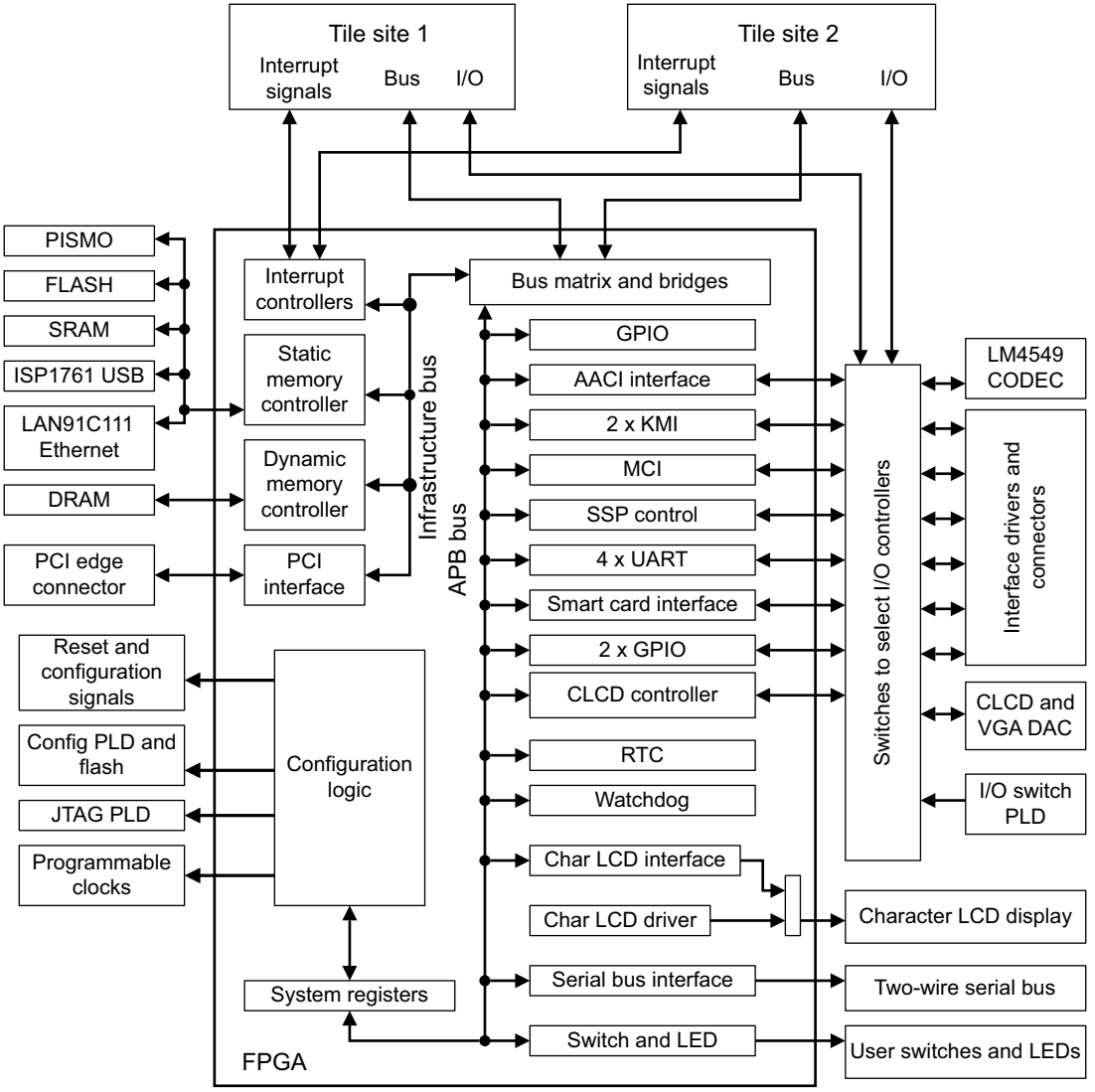

Figure 3-10 FPGA block diagram

For more detail on FPGA components, see:

- *FPGA configuration* on page 3-17

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

- *Reset logic* on page 3-20
- *Clock architecture* on page 3-37
- *Advanced Audio CODEC Interface, AACI* on page 3-47
- *Character LCD controller* on page 3-50
- *Ethernet interface* on page 3-57
- *Keyboard/Mouse Interface, KMI* on page 3-63
- *Multimedia Card Interface, MCI* on page 3-64
- *PCI interface* on page 3-67
- *Smart Card interface, SCI* on page 3-70
- *User switches and LEDs* on page 3-76
- *UART interface* on page 3-77
- *USB interface* on page 3-81.

Note

The FPGA buses on the baseboard are shared with the tile site 1 and tile site 2 headers. Use the appropriate FPGA image to ensure that the tiles bus signals match the FPGA signals.

The peripherals and controllers implemented in the baseboard FPGA are in the standard images distributed by ARM Ltd.

The logic shown in Figure 3-10 on page 3-16 is for the default image. You can replace some or all of the logic in the FPGA with your own designs, however, the details of how to do this are beyond the scope of this user guide. The CD includes FPGA HDL and signal definition files that can be used as a basis for custom designs.

The PrimeCell peripherals used in the baseboard FPGA are supplied as black-box implementations. The internal design of the peripheral is not configurable or viewable.

3.2.1 FPGA configuration

At power-up the FPGA loads its configuration data from one of two images stored in a flash memory device. Parallel data from the flash memory is streamed by the configuration PLD into the configuration ports of the FPGA. Figure 3-11 on page 3-18 and Figure 3-13 on page 3-25 show the FPGA configuration mechanism. The image loaded into the FPGAs is determined by the configuration switch as listed in *FPGA image select switch* on page 2-6.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

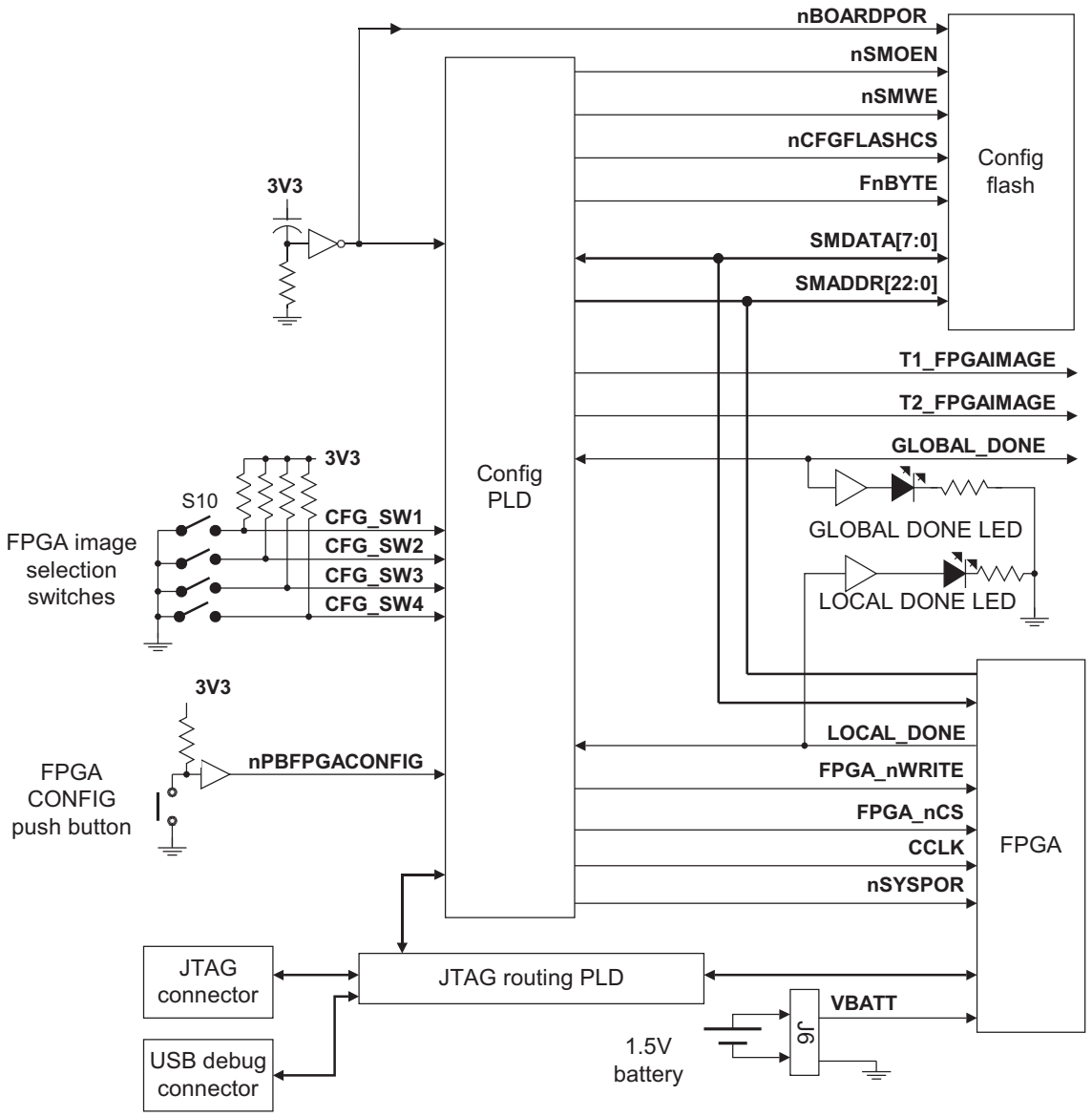

Figure 3-11 FPGA configuration

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

The configuration flash is a separate device and not part of the user NOR or DOC flash. The configuration flash can be accessed through the SMC static memory space, but this is not recommended because the contents might be corrupted.

The baseboard is shipped with a test image in the configuration flash. You must load a new image into the configuration flash before you can use the development board. For details on loading an image, see *Loading FPGA and PLD images* on page 2-20. The FPGA configuration to load depends on the type and combination of tiles attached.

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 baseboard. To replace the battery:

1. Power on the baseboard. 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 and ensure that the positive terminal is facing upwards in the holder.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.3 Reset logic

The reset logic initializes attached Logic Tiles and Core Tiles, the FPGA, and external controllers as a result of a reset. The baseboard reset sources are listed in Table 3-3.

Table 3-3 Reset sources

Source	Signal	Description
Power applied to board	**nBOARD_POR**	When the power is applied (or restored after a power failure), the **nBOARD_POR** signal is generated.
Reset push button	**nPBRESET**	Resets the system without reloading the FPGA image. The effect on the system is the same as the **nBOARD_POR** signal going active.
FPGA reload push button	**nPBFPGACONFIG**	Resets the system and reloads the FPGA images in the baseboard, Core Tiles, and Logic Tiles. **Note** Use the FPGA CONFIG push button to reload the FPGA image without repowering the entire system. The FPGA configuration registers are reloaded with their default values. (Pressing FPGA CONFIG also resets the core.)
PCI backplane	**P_nRST**	There is a reset switch on the PCI backplane. The signal can also be generated by a PCI card installed in the backplane.
Logic Tiles	**nSYSPOR** (or **D_nSRST**)	These signals can be generated by custom logic implemented in the Logic Tiles
JTAG	**nTRST**	The JTAG reset signal can be triggered by an attached JTAG debugging interface device.
Watchdog	-	If the Watchdog timer is enabled and times out, a reset is generated.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.3.1 Reset and reconfiguration logic

Figure 3-12 shows the reset and reconfigure logic. Not all JTAG signals are shown. The JTAG block includes interface logic and the JTAG routing PLD.

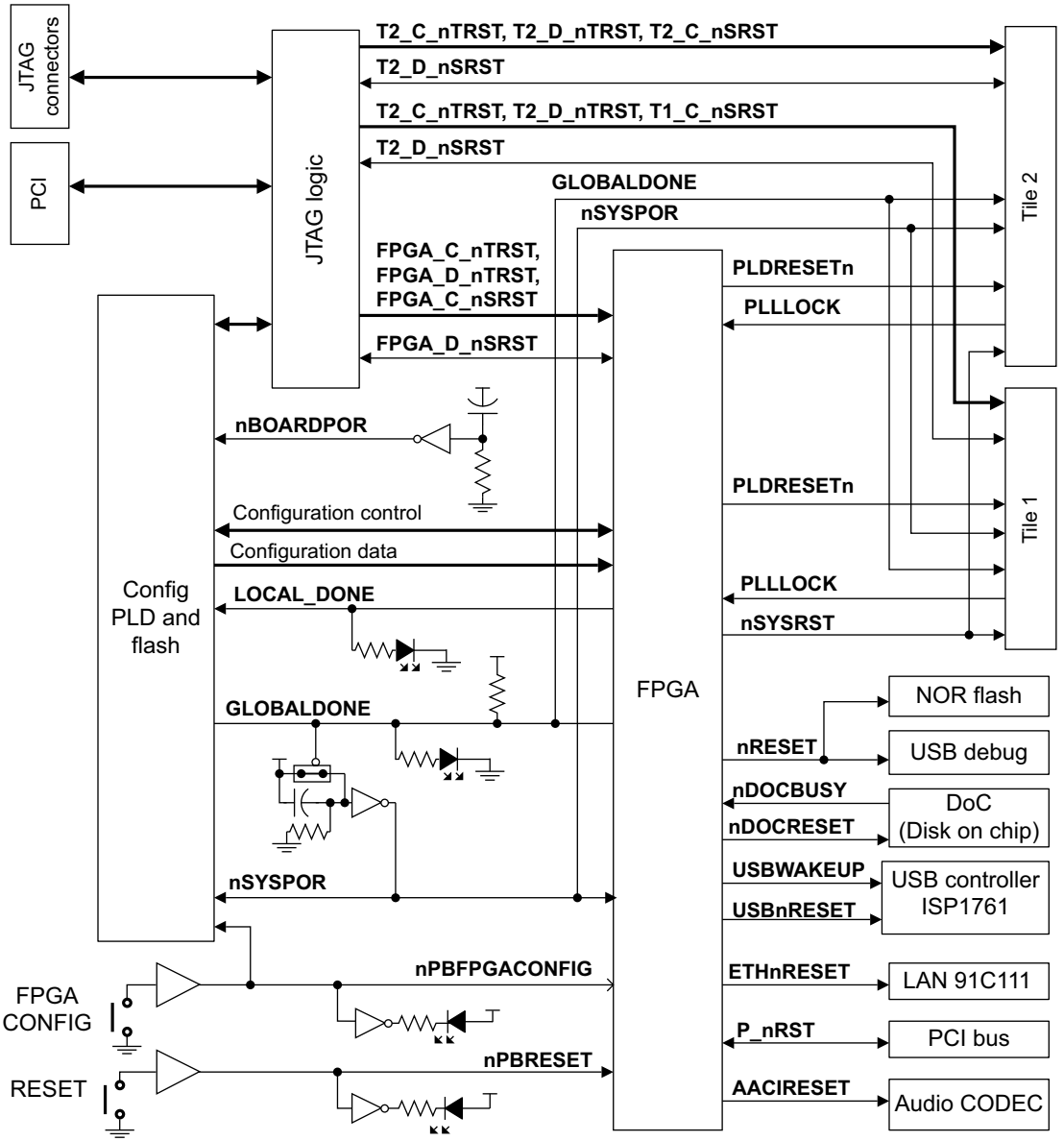

Figure 3-12 Baseboard reset logic

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.3.2 Reset signals

Table 3-4 describes signals related to reset and configuration. A reset controller implemented in the FPGA monitors the reset sources and generates the appropriate reset signals.

Note

In addition to the hardware reset signals, the system can also be reset from the Watchdog controller.

Table 3-4 Reset and configuration signals

Name	Function
AACIRESET	System reset to audio CODEC.
nBOARDPOR	This signal resets the configuration PLD and configuration flash. This signal is also used to generate the **nTRST** pulse at power on.
C_nSRST	JTAG open-collector reset signal to or from the tile. This signal is part of the configuration JTAG chain.
C_nTRST	JTAG open-collector **TRST** signal to the configuration JTAG chain in the tile. This signal is part of the configuration JTAG chain.
nDOCBUSY	Memory status signal from the Disk-on-Chip. The DoC requires approximately 27ms after reset is applied in order to initialize its internal controllers.
nDOCRESET	System reset to Disk-on-Chip memory.
D_nSRST	JTAG open-collector reset request signal to or from the tile. This signal is part of the debug JTAG chain. A device in the FPGA (the watchdog, for example) can generate a system level reset by pulling the **nSRST** signal LOW. This will cause the **nSYSRST** signal to pulse LOW and reset the system. The pulse does not generate a power on reset. Another example of a reset condition would be the DLLs in the FPGAs losing lock due to a frequency change. Pulling **nSRST** signal low allows the DLLs to relock. After lock is established, **nSRST** is released and **nSYSRST** goes HIGH and the system exits reset.
D_nTRST	JTAG open-collector **TRST** signal to the debug JTAG chain in the tile. This signal is part of the debug JTAG chain.
ETHRESET	System reset to Ethernet controller (active HIGH).
FPGA_nPROG	The **FPGA_nPROG** signal forces all FPGAs in the system to reconfigure. This signal enables the FPGAs to be reconfigured without powering-down the system.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 3-4 Reset and configuration signals (continued)

Name	Function
LOCAL_DONE	This signal goes HIGH when the baseboard FPGA has finished configuring. **GLOBAL_DONE** is LOW until this signal goes HIGH.
GLOBAL_DONE	This is an open-collector configuration signal that goes HIGH when all FPGAs have finished configuring. The system is held in reset until this signal goes HIGH.
nPBFPGACONFIG	This signal is generated from the FPGA CONFIG push button and causes a total reconfiguration of the system.
nPBRESET	Push-button reset signal to the FPGA. The signal is generated by pressing the reset button.
P_nRST	System reset from PCI backplane.
PLDRESETn (on Core Tiles)	Reset signals to the configuration PLD located on the Core Tile. (Baseboard FPGA signals **T1Z229** and **T2Z229**)
PLDD[1:0] (on Core Tiles)	Data signals to the configuration PLD located on the Core Tile. (Baseboard FPGA signals **T1Z[228:227]** and **T2Z[228:227]**)
PLDCLK (on Core Tiles)	Clock signals to the configuration PLD located on the Core Tile. (Baseboard FPGA signals **T1Z230** and **T2Z230**)
P_nTRST	JTAG TRST signal from PCI backplane. **Note** There is a separate JTAG connector and an independent scan chain on the PCI backplane. The JTAG chain on the baseboard does not normally extend to the PCI expansion backplane. There is a separate JTAG connector on the PCI backplane for configuring devices on the backplane and on installed PCI cards. There are also links that can be fitted to the baseboard that connects the two JTAG chains together, but these links are normally only fitted for manufacturing tests.
nRESET	Boot memory remapping signal internal to the FPGA.
nSRST	**nSRST** is an active LOW open-collector signal that can be driven by the JTAG equipment to generate a **nSYSRST** system reset request. Some JTAG equipment senses this line to determine when you have reset a board. This is also used in configuration mode to control the initialization of the FPGA. **Note** **nSRST** splits into **D_nSRST** and **C_nSRST** to provide separate debug and configuration signals on the tile connector HDRZ. A Logic Tile can generate a reset by pulling this open-collector signal LOW.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 3-4 Reset and configuration signals (continued)

Name	Function
nSYSPOR	Power-on reset signal that initializes the reset level state machine after **GLOBAL_DONE** goes HIGH. This signal is also fed to the tile headers.
nSYSRST	System reset to the Logic Tile headers.
nTRST	Open-collector TAP controller reset (the board drives this signal with **nBOARDPOR**). **Note** **nTRST** splits into **D_nTRST**, and **C_nTRST** to provide separate debug and configuration signals on HDRZ of the tile.
USBnRESET	System reset to USB controller.
USBWAKEUP	Signal to USB controller to re-initialize.

3.3.3 Power-on reset timing

Figure 3-13 on page 3-25 shows the power-on reset sequence. A reset controller implemented in the FPGA monitors **nSYSPOR** and generates the appropriate reset signals.

nBOARDPOR is generated at power-up and distributed to the memory expansion boards and to the FPGA configuration PLD. It also causes the assertion of the **nTRST** signal and guarantees the embedded ICE macrocell in the processor core is reset.

Note

The release time for **GLOBAL_DONE** depends on any Logic Tiles in the system. It is held LOW longer if the tiles take longer to configure.

The system can also be reset from the JTAG signal **nTRST** or from the system reset signal **nSRST** as shown in Figure 3-14 on page 3-25.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

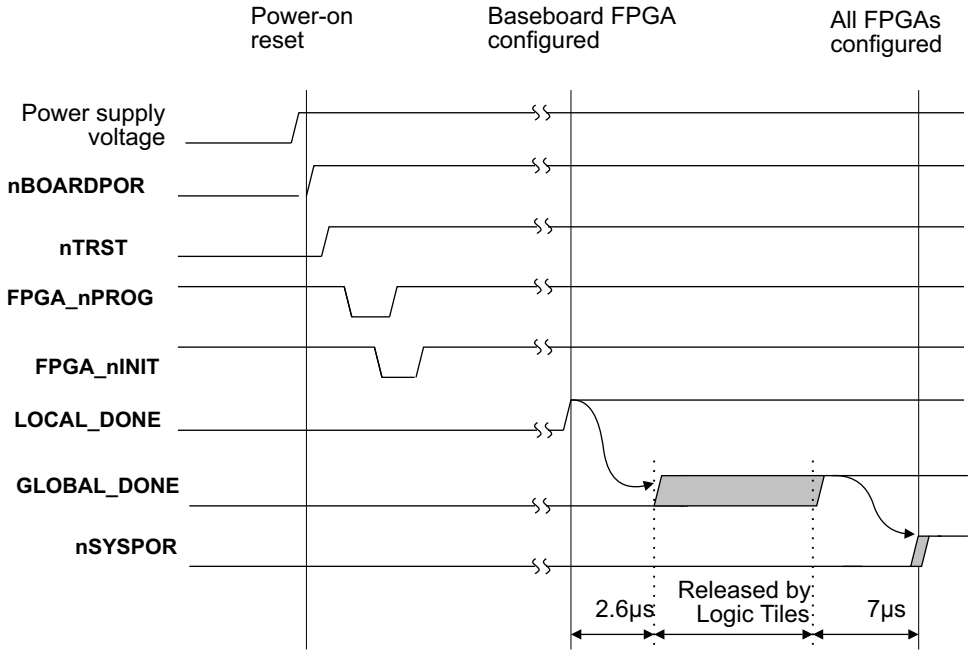

Figure 3-13 Power-on reset and configuration timing

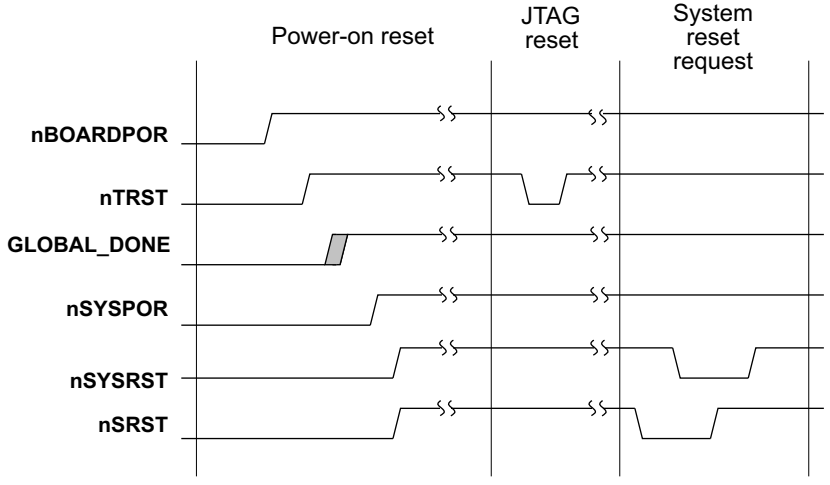

Figure 3-14 JTAG and system reset

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.3.4 Memory aliasing at reset

Under normal operation, the baseboard has dynamic memory located at 0x0. In order to run the boot code after a reset however, non-volatile memory must be remapped to the boot address.

Remapping is done by modifying the static memory controller chip select signals that connect to the memory devices. Figure 3-15 on page 3-27 shows a simplified equivalent diagram of the remapping logic circuitry.

At reset, the **REMAP** signal from the system controller is HIGH. The **DMCS0** signal that is normally generated by accesses to memory region 0x00000000–0x0FFFFFFF is disabled and routed to static memory:

- If switch S8[4] is OFF, one of the chip selects for static memory is enabled.
- If switch S8[4] is ON, the boot memory select is routed to the AXI or AHB master on tile site 2.

See *Remapping of boot memory* on page 4-8.

Note

If the size of the physical memory selected is less than the address range of 0x00000000–0x0FFFFFFF, the physical memory is aliased and repeated to fill the address space.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

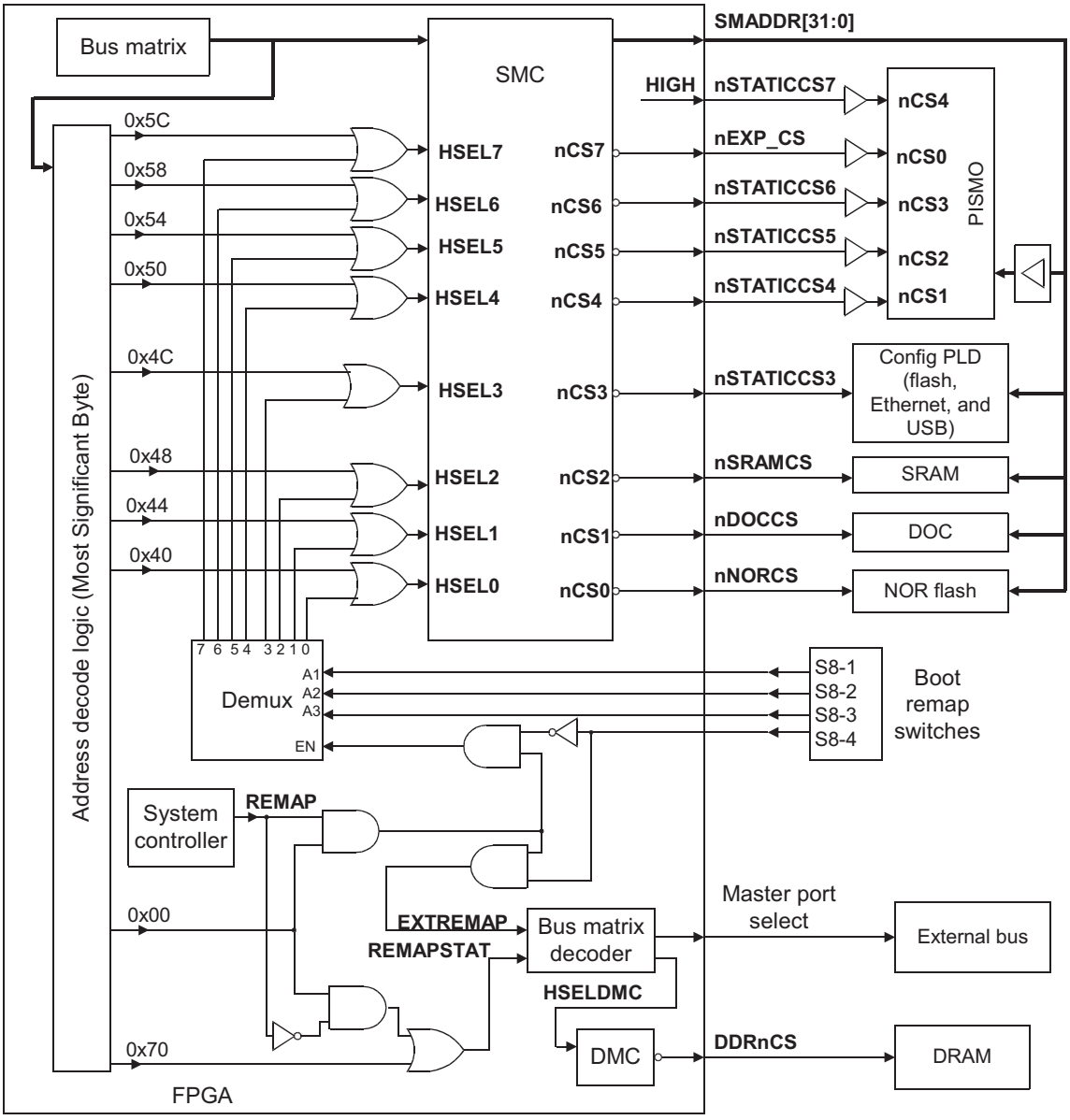

Figure 3-15 Boot memory remap logic

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.4 Power supply

The power supply circuit is shown in Figure 3-16 on page 3-30.

If the baseboard is powered from the 12V brick power supply:

- A nominal 12V level (**DC fused**) is supplied to the FETs (shown in Figure 3-17 on page 3-31). The FETs are switched on if **DC fused** has correct polarity supply and is within plus or minus 10%.

  If the input voltage is too high, the comparator switches off the FETs and disconnects **DC fused** from **VSMP** (the 12V power to the switched-mode regulators) and **12VDC**.

  If the input drops too low, shutdown signals **nSHDN1**, and **nSHDN2** become LOW and the regulators are switched off.

- The power supply can be toggled on and off by pressing the Power/Standby push button.

  The shutdown circuitry is shown in Figure 3-17 on page 3-31.

  If a link is placed across J48, the power is forced on and the on/off toggle switch has no effect.

- The regulators are supplied by the VSMP voltage. The track inputs on the voltage regulators ensure that the voltage is applied in the correct sequence (for example, the output from the 3V3 regulators cannot exceed the output from the 5V regulators during power on).

If the baseboard is powered from the PCI backplane or the screw terminals, the **VSMP** and **DC fused** voltages are not present. Therefore:

- the **3V3SB** standby voltage is not present
- The control signal for the 5V regulator to the AACI (**nSHDN2**) is held HIGH by the 12V supply from the PCI backplane
- The control signal to the other voltage regulators (**nSHDN1**) is held LOW
- the Power/Standby push button has no effect and you must use the external power source to turn the system on or off.

Caution

There is no overvoltage or polarity protection if the PCI backplane or screw terminals are used as the power source. Connecting an incorrect voltage to the screw terminals might damage the baseboard and any boards connected to it.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The 12V DCIN supply is fused (5A) before connection to the polarity and shutdown logic. The LCD expansion, USB, Keyboard, and Mouse connectors have an additional fuse (1A) on the 5V supply.

Warning

Refer to the *Bill of Materials* (BOM) file in the `schematics` directory for the fuse manufacturer and part numbers. You must use the same type and rating of fuse if you replace a blown fuse.

There are also two regulators for the USB debug circuit. These are powered from the 5V supplied by the host computer on the USB connector. See *JTAG and USB debug port support* on page 3-83.

Logic Tiles supply the correct interface voltage for memory or peripherals by driving the VDDIO power pins on the HDRX and HDRY Logic Tile connectors. If a tile is not present on a tile site:

- there is not a supply voltage provided by the tile to **VCCO1** on tile site 1 (or no **VCCO2** for tile site 2)
- the **T1_nTILEDET** signal is high if there is not a tile on tile site 1 (**T2_nTILEDET** is high if no tile on site 2)
- FETs Q1 and Q2 for tile site 1 (or Q3 and Q4 for site 2) connect the local 3.3V supply to the appropriate FPGA I/O supply pins.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

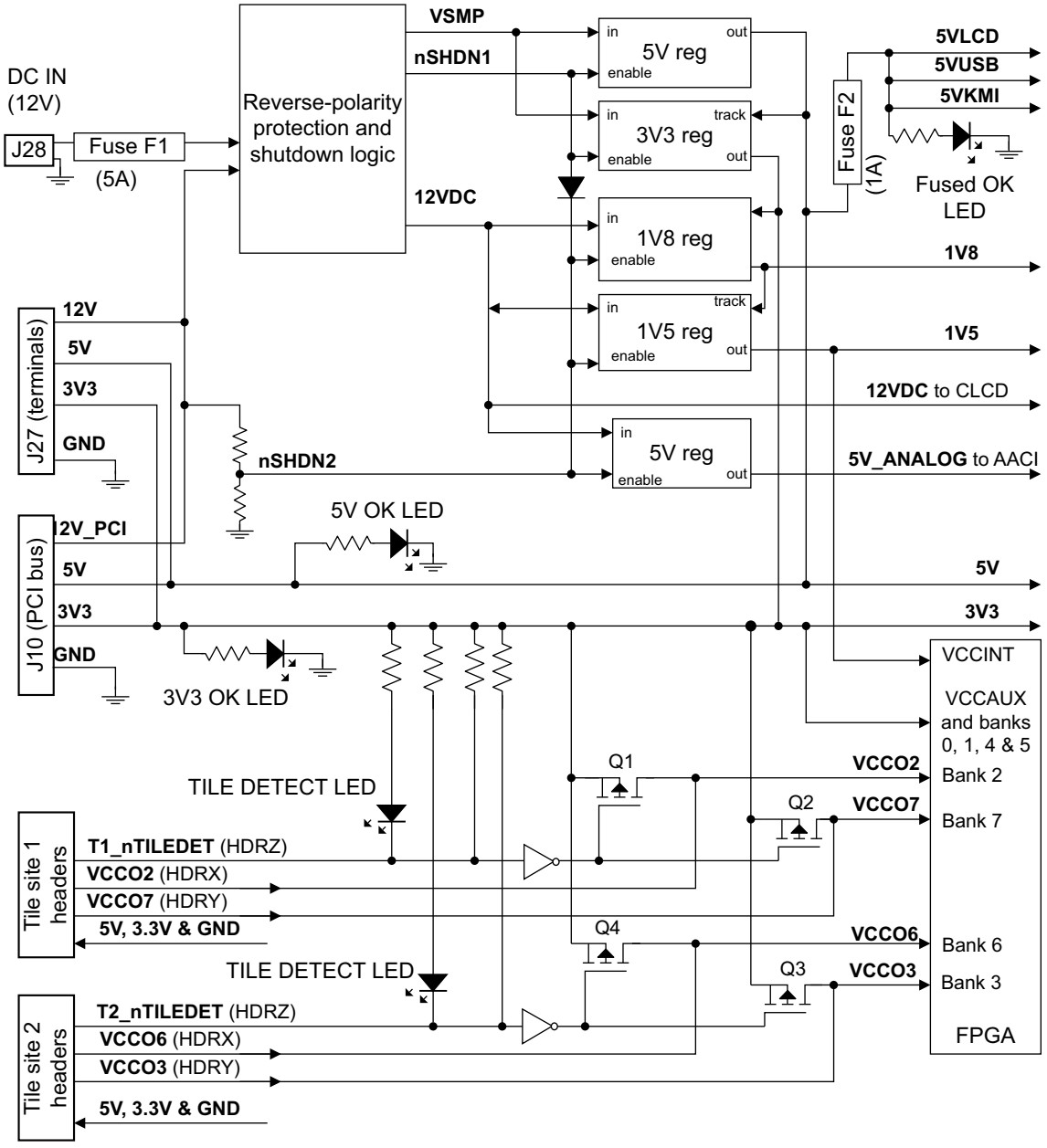

Figure 3-16 Power-supply regulators and protection circuitry

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

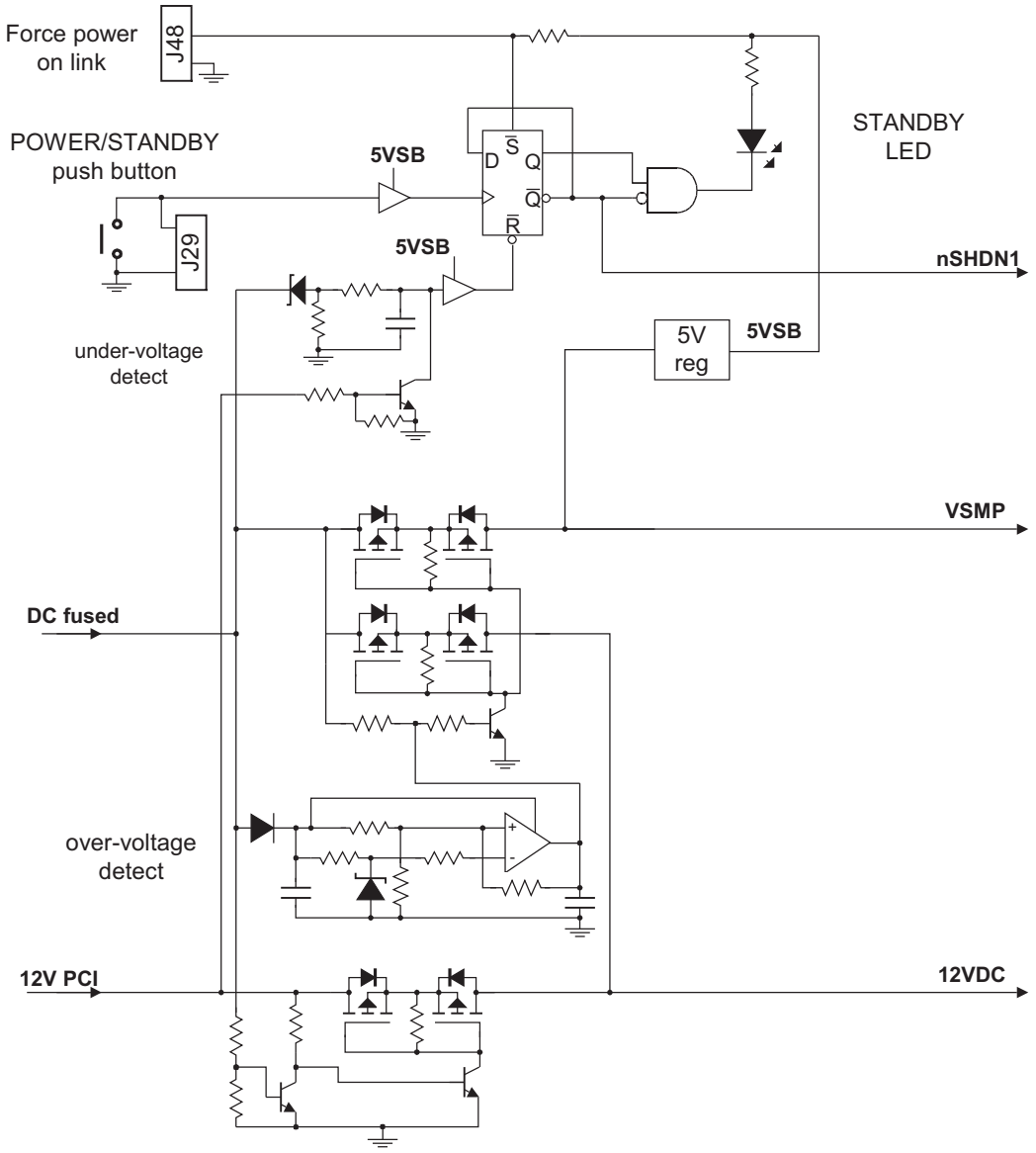

Figure 3-17 Reverse-polarity protection and shutdown circuit

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.5 Memory controllers

The baseboard has two memory controllers implemented in the FPGA:

SMC The *Static Memory Controller* (SMC) manages the SRAM, flash, PISMO memory expansion board, Ethernet and USB controllers. See also *Static Memory Controller, SMC* on page 4-86.

DMC The *Dynamic Memory Controller* (DMC) manages the DDR SDRAM mounted on the baseboard. See also *Dynamic Memory Controller, DMC* on page 4-51.

The static memory controller drives the devices listed in Table 3-5.

Table 3-5 Devices on the static memory bus

Device	Chip select	Address	Description
NOR flash	CS0	`0x40000000`	64MB of NOR flash
DOC flash	CS1	`0x44000000`	64MB of Disk-on-Chip flash
SRAM	CS2	`0x48000000`	2MB of SRAM
Configuration flash	CS3	`0x4C000000`	8MB of configuration flash. This flash holds the image for the FPGA. It must not be used by application code.
Ethernet	CS3	`0x4E000000`	The Ethernet controller is mapped into this 16MB space
USB	CS3	`0x4F000000`	The USB controller is mapped into this 16MB space

PISMO	CS[7:4]	`0x50000000`	The PISMO memory expansion connector has four chip selects that select four 64MB regions:
			`nEXPCS 0x5C000000 (PISMO CS0)`
			`CS4 0x50000000 (PISMO CS1)`
			`CS5 0x54000000 (PISMO CS2)`
			`CS6 0x58000000 (PISMO CS3)`
			(FPGA CS7, PISMO CS4, is permanently disabled in the FPGA) See Appendix E *PISMO Memory Expansion Boards* for more details of the PISMO memory expansion cards.

Note

The configuration PLD uses the address bus and the CS[3] signal to generate the device chip selects for the Ethernet controller, the USB controller, and the FPGA configuration flash memory. The FPGA reroutes the static chip selects at power-on-reset to place one of the static devices at address `0x0` as described in *Reset logic* on page 3-20.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Figure 3-18 shows the memory devices on the static memory bus.

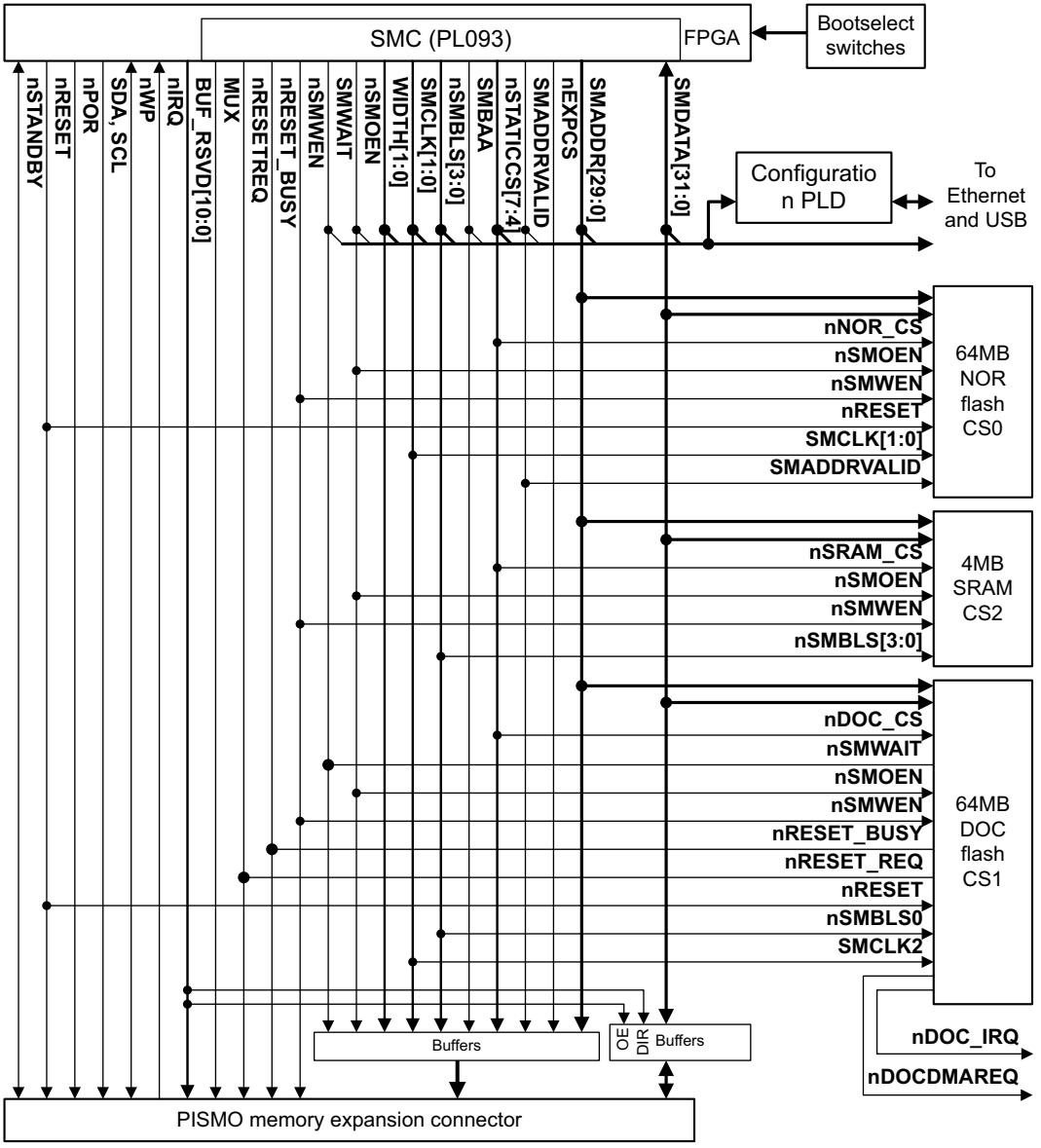

Figure 3-18 Static memory devices

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Figure 3-18 on page 3-33 shows the Ethernet controller IC on the static memory bus. For more details on the Ethernet controller, see *Ethernet interface* on page 3-57, *Ethernet* on page 4-54, and *Ethernet interface* on page A-6.

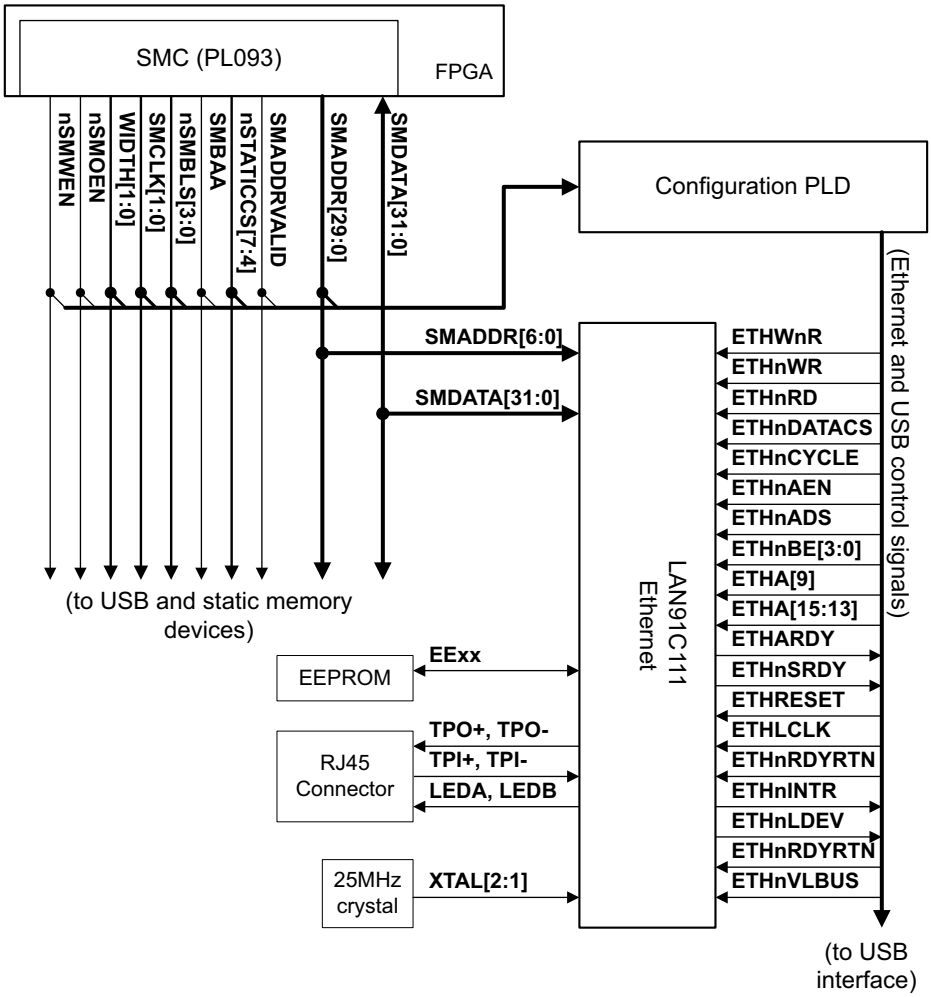

Figure 3-19 Ethernet device on static memory bus

Figure 3-18 on page 3-33 shows the USB controller IC on the static memory bus. For more details on the USB controller, see *USB interface* on page 3-81, *USB interface* on page 4-93, and *USB interface* on page A-53.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

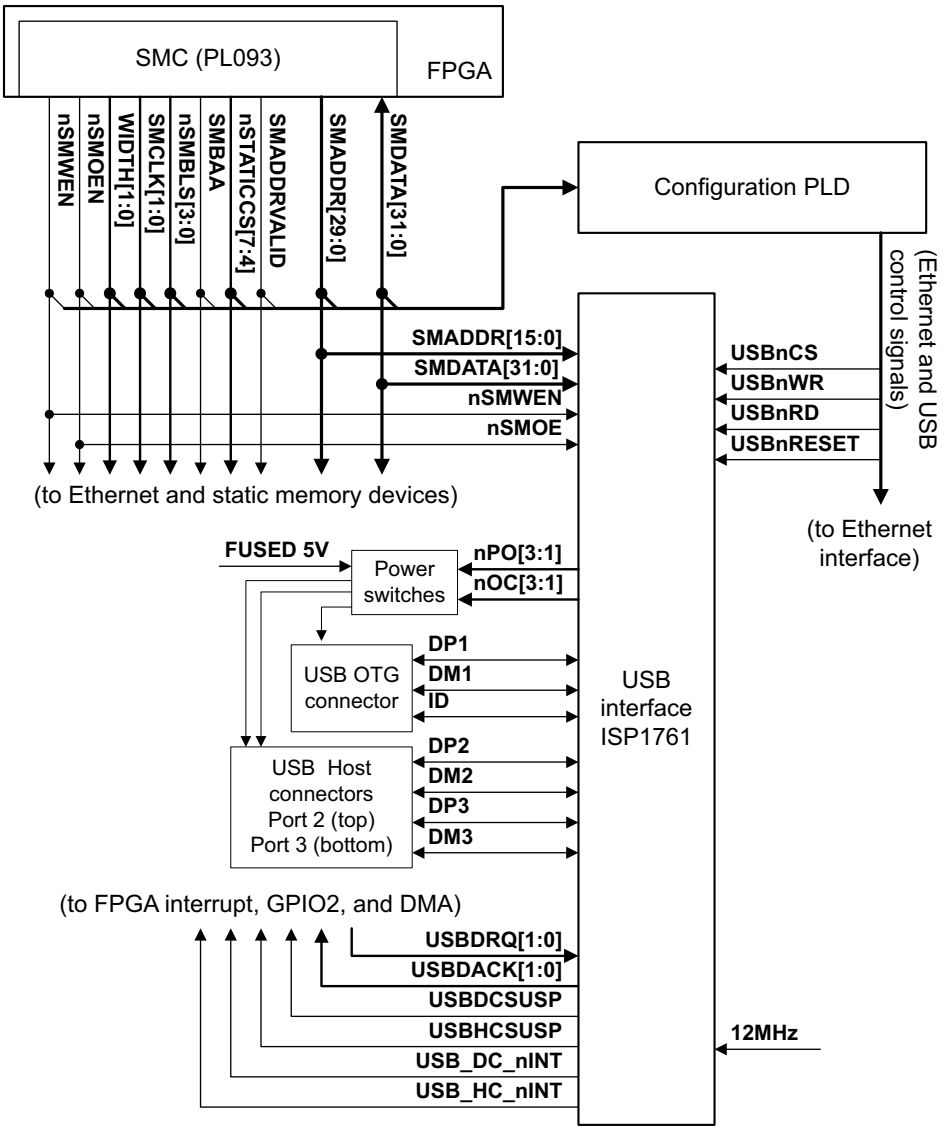

Figure 3-20 USB device on static memory bus

The DMC bus is connected to 256MB of DDR SDRAM as shown in Figure 3-21 on page 3-36. Series and termination resistors suppress reflections and noise on the control and data lines. The SDRAM is addressed from `0x70000000`–`0x7FFFFFFF`. If REMAPSTAT is HIGH, the memory is also aliased at `0x0`–`0x0FFFFFFF`.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

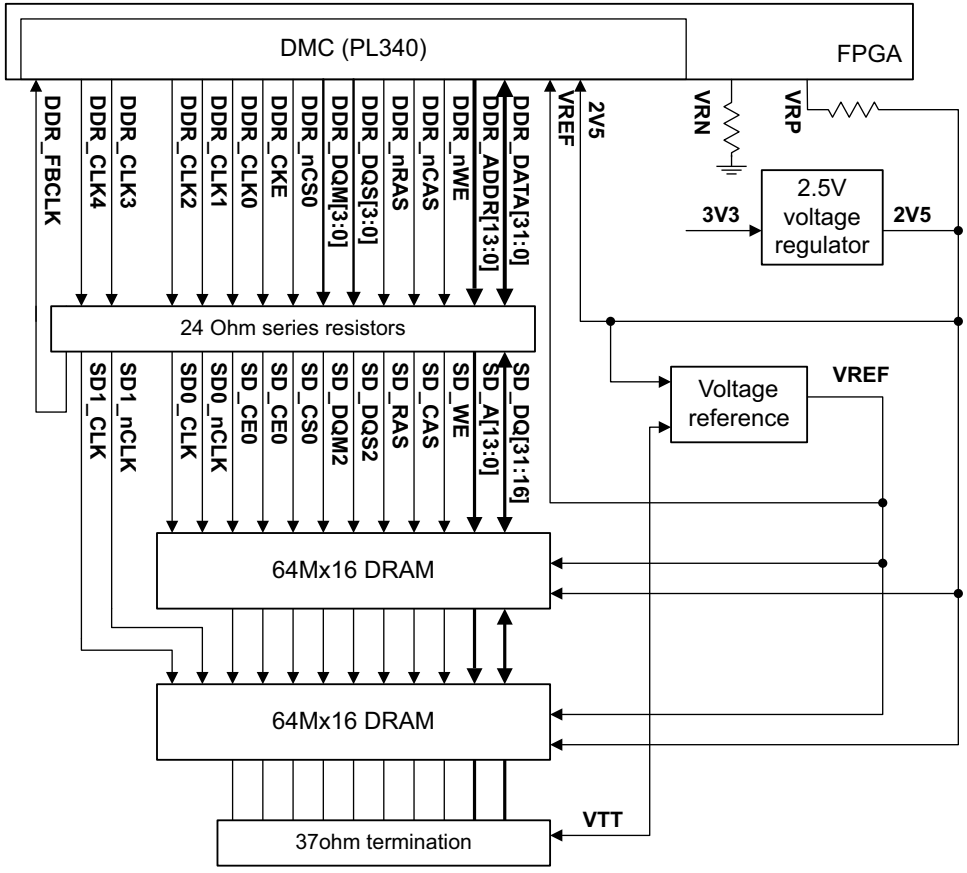

Figure 3-21 Dynamic memory bus

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.6 Clock architecture

The clock domains for the baseboard are:

FPGA reference clocks and tile clocks

The FPGA contains clock control logic that sets the frequency of the programmable clock generators. See *ICS307 programmable clock generators* on page 3-44.

The FPGA also controls the routing of clocks to and from the tile expansion connectors. See *Tile clocks* on page 3-39 for details of the clocks available.

Peripheral clocks

The peripherals on the baseboard use dedicated oscillators or one of the programmable oscillator outputs as a reference clock.

See *Peripheral clocks* on page 3-42 for details.

Debug The JTAG connector supplies the reference JTAG clock **TCK**. There is also an on-board USB debug port that is driven by the 24MHz reference and a dedicated 6MHz crystal oscillator. See *Test, configuration, and debug interfaces* on page 3-83.

Note

The clocking selection and control logic in the baseboard FPGA enables you to emulate many different clock systems and operating modes.

The clocks used depend on the type and configuration of tiles added to the baseboard. See the application notes for examples of different combinations of boards.

The clock domains for the baseboard are shown in Figure 3-22 on page 3-38.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

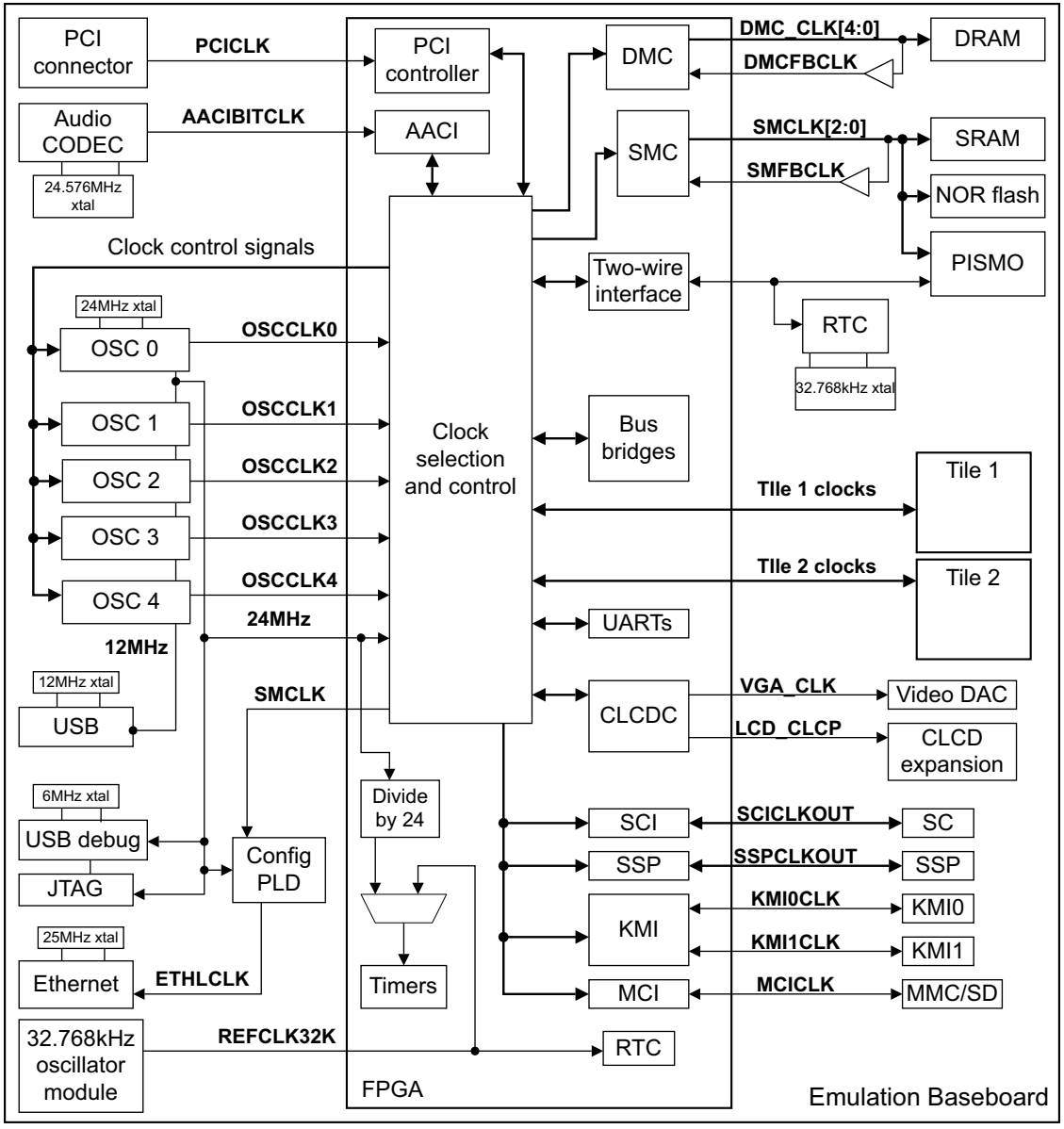

Figure 3-22 Clock architecture

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.6.1 Tile clocks

The baseboard can be expanded by adding Logic Tiles or Core Tiles. Table 3-6 lists the clock signals that are present on both the baseboard FPGA and the attached tile.

Note

Some of the clocks listed in Table 3-6 might not be present on the baseboard. The clocks used by the baseboard depend on the system configuration and the image loaded into the baseboard FPGA. Refer to the application note that covers your combination of baseboard and Core Tile for more details.

JTAG clocks are not listed in Table 3-6, see *JTAG signals* on page 3-87 for details.

See *Tile headers and signal interconnects* on page 3-3 and the manual for your Logic Tile for details on Logic Tile clocks. See the *Core Tile User Guide* for details of Core Tile clocks.

Table 3-6 Clock signals on tiles

Tile signal	Direction (relative to tile)	Description
GLOBALCLK	Input/output	A global clock shared with all tiles in the stack. Each tile can accept or drive the signal. This signal can be supplied by ICS307 programmable oscillator 0 (**OSCCKL0**) on the baseboard. If the baseboard signal **nGLOBALCLKEN** is HIGH, the baseboard FPGA drives the **GLOBALCLK** signal to the tile headers.
CLK_NEG_DN, **CLK_POS_DN**, **CLK_OUT_MINUS1**, **CLK_UP_THRU**	Output	Clock signals from the tile to the baseboard FPGA.
CLK_NEG_UP, **CLK_POS_UP**, **CLK_IN_MINUS1**	Input	Clock signals from the baseboard FPGA to the tile.
CLK_OUT_PLUS1, **CLK_OUT_PLUS2**, **CLK_IN_PLUS1**, **CLK_IN_PLUS2**	-	These Logic Tile clock signals are not used by the Core Tile or the baseboard. They might be used in some Logic Tile designs. Refer to the documentation supplied with the Logic Tile or the application note for your configuration for more details.
PLDCLK	Input	The baseboard clock **TnZ230** clocks configuration data into (and status data out of) the PLD on the Core Tile.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 3-6 Clock signals on tiles (continued)

Tile signal	Direction (relative to tile)	Description
REFCLK	Input	A reference clock to the test chip clock-dividers on the attached tile.
HCLKIN	Input	**Tn_X32** from the FPGA can be selected as the source of **HCLKIN** to the test chip on the Core Tile.
HCLKOUT	Output	**HCLK** from the test chip on the Core Tile.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

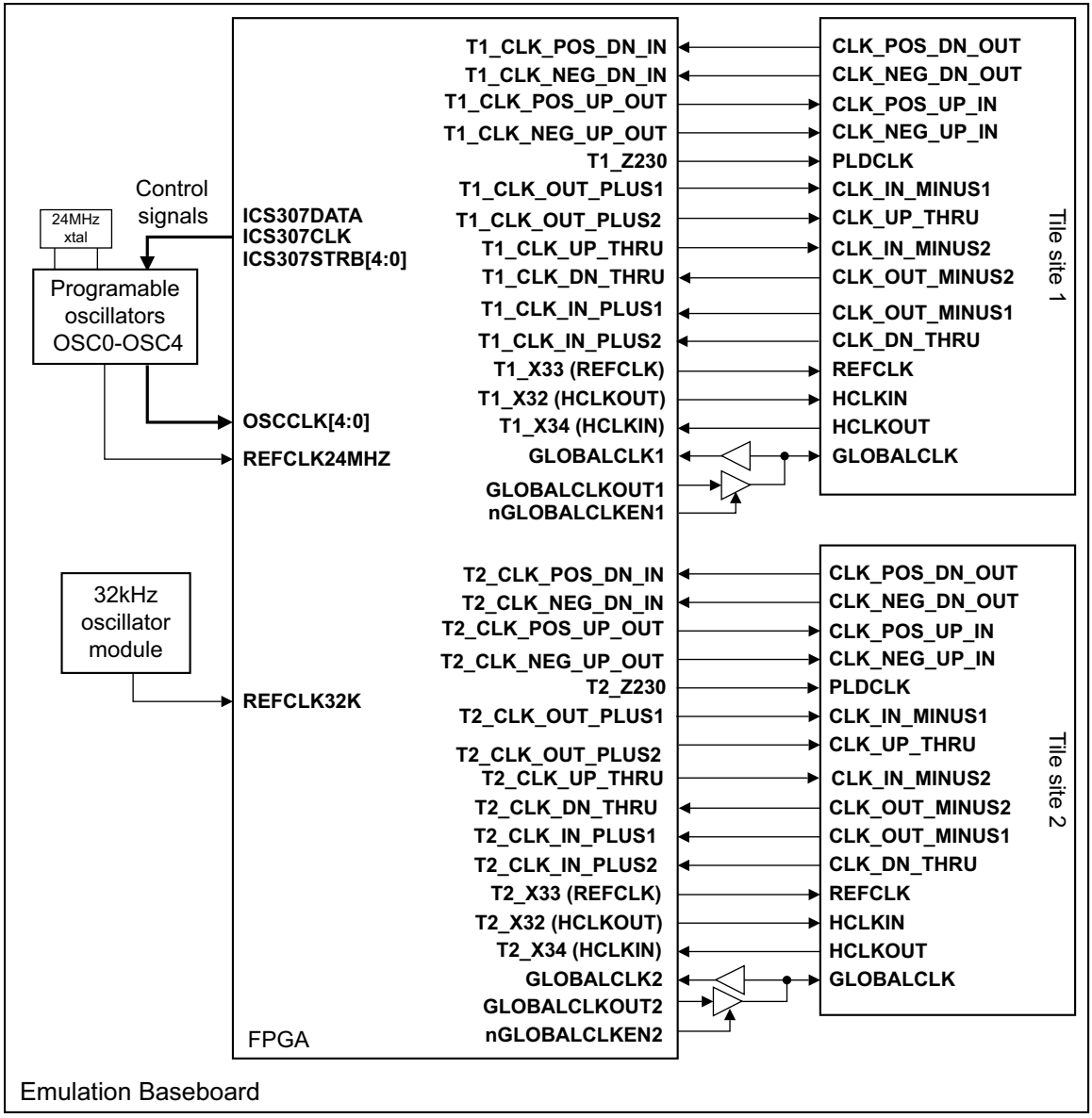

Figure 3-23 Tile clocks

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.6.2 Peripheral clocks

The following peripherals are present on the baseboard:

Audio CODEC

The Audio CODEC has a dedicated crystal oscillator. The reference clock from the CODEC is connected to the AACI in the FPGA.

DMC The DMC uses **OSCCLK1** for internal timing and to generate the external dynamic memory clocks.

Ethernet The Ethernet controller has a 25MHz dedicated crystal oscillator for timing the Ethernet bus.

CLCDC The CLCD controller uses OSC4 as the reference clock for video output.

MCI The MCI uses **REFCLK24MHZ** for internal and external timing.

KMI The two KMIs use **REFCLK24MHZ** for internal timing and the generation of the KMI external clocks.

PCI A **PCICLK** is derived from the 33MHz or 66MHz reference oscillator on the PCI backplane. The PCI clock is connected to the PCI controller in the FPGA to synchronize accesses with the PCI bus. See *Peripheral clocks* and *PCI interface* on page 3-67. A signal defined in the FPGA image selects between 33MHz and 66MHz by outputting the **P_66EN** signal to the backplane. The default is 66MHz.

RTC There is an on board real-time clock module clocked by a dedicated 32.768kHz crystal oscillator. The RTC module outputs full BCD time data to the FPGA via an I^2C serial interface.

SSP The SSP uses **REFCLK24MHZ** for internal and external timing.

SCI The SCI uses **REFCLK24MHZ** for internal and external timing.

SMC The SMC uses **OSCCLK0** for internal timing and to generate the external static memory clocks.

Timer The four timers are clocked with a 35MHz signal and the clock enable signal is selected by the System Controller. The clock enable signal can be configured as permanently high or pulsed. The System Controller has a 1MHz and a 32.768kHz clock. The timer can be configured to run at 35MHz, 1MHz, or 32.768kHz.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Two-wire serial bus

The custom serial bus uses the 24MHz reference for internal timing and to SCL clocks for communication with the expansion memory board and the external time-of-year clock.

UART The four UARTS use **REFCLK24MHZ** for internal timing and baud rate generation.

USB A 12MHz oscillator provides the reference frequency for the external USB controller.

Watchdog The watchdog timer is clocked with a 1MHz signal derived from **REFCLK24MHZ**.

Table 3-7 lists the memory and peripheral clocks on the baseboard. For more detail on the clocking system, see the files in the `Schematics` directory of the CD supplied with the baseboard.

Table 3-7 baseboard clocks and clock control signals

Clock signal	Frequency	Description	Source
AACIBITCLK	12.288MHz	This is the synchronization clock from the audio CODEC. The clock is an input to the AACI PrimeCell.	Crystal oscillator
CLCDCLKEXT	6–50MHz	The clock for PL110 CLCD Controller in the FPGA is typically derived from OSC4.	ICS307 OSC4
ETHLCLK	Static memory clock	**ETHLCLK** is used to synchronize data transfers between the external Ethernet controller and the FPGA. (The Ethernet controller uses a separate 25MHz crystal for clocking signals to and from the Ethernet connector.)	24MHz reference and 25MHz oscillator
REFCLK24MHZ	24MHz	**REFCLK24MHZ** is generated from OSC0 and the 24MHz crystal. The clock is used as a reference for the timers, USB, Ethernet, MCI, KMI, SSP, SCI, the timer modules, and the watchdog timer. Buffered versions of the reference are **REFCLK24MHZ2P**, **REFCLK24MHZ2F**, and **REFCLK24MHZ2J**.	ICS307 OSC0 reference out
REFCLK12MHZ	12MHz	**REFCLK12MHZ** is generated from OSC4 and the 24MHz reference output from OSC0.	ICS307 OSC4 reference out

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 3-7 baseboard clocks and clock control signals (continued)

Clock signal	Frequency	Description	Source
SCIREFCLKEXT	24MHz	The clock for PL131 SCI in the FPGA can be derived from this input. This is a buffered version of **REFCLK24MHZ**.	24MHz reference
SMCLK[2:0]and **SMFBCLK**	-	The static memory clocks from the SMC in the FPGA. **SMFBCLK** is a buffered version of **SMCLK2** that is fed back to the SMC to indicate the amount of delay present.	SMC
DMC_CLK[4:0] and **DMCFBCLK**	-	The dynamic memory clocks from the DMC in the FPGA. **DMCFBCLK** is a buffered version of **DMC_CLK4** that is fed back to the DMC to indicate the amount of delay present between the DMC and the memory device.	DMC

3.6.3 ICS307 programmable clock generators

Five programmable (6–200 MHz) clocks are supplied to the FPGA by the programmable ICS307 clock generators (OSC0–OSC4):

OSCCLK0 This is the reference clock. This is normally used as **GLOBALCLK**, the external AHB bridge clocks, and the reference for the PLL that generates **CPUCLK**.

OSC0 uses a 24MHz crystal as its reference. A fixed-frequency 24MHz signal, **REFCLK24MHZ**, is output from OSC0 and used as a reference signal for:

- the input for programmable oscillators OSC1–OSC4.
- the Ethernet controller clock (the Ethernet serial data clock is generated from a 25MHz crystal on the Ethernet controller)
- the USB controller clock
- the USB debug controller clock
- the input to divide-by-24 logic in the FPGA that produces the 1MHz reference clock for the timers.

OSCCLK1 An alternative reference clock.

OSCCLK2 An alternative reference clock.

OSCCLK3 An alternative reference clock.

OSCCLK4 This the reference for the CLCD controller.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The output frequencies of the ICS307s are controlled by divider values loaded into the serial data input pins on the oscillators. The divider values are defined by the SYS_OSCx registers implemented in the FPGA. The data stream and register format is shown in Figure 3-24. See *Oscillator registers, SYS_OSCx* on page 4-15 for details on the clock control registers.

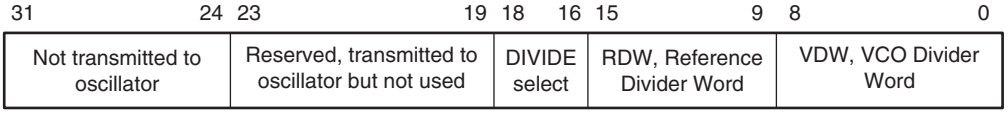

Figure 3-24 Serial data and SYS_OSCx register format

Note

Bit 23 is loaded into the shift register first and bit 0 is loaded last. Data is clocked into the **ICS307DATA** pins of the oscillators on the rising edge of **ICS307CLK**. One of the **ICS307STRB[4:0]** signals is pulsed HIGH to latch the serial data into the divider control register.

The serial interface logic is implemented in the standard FPGA design. The only interface to the clock control logic is through the SYS_OSCx registers.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

You can calculate the oscillator output frequency from the formula:

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

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]

DIVIDE Is the divide ratio (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 ICS web site at `www.icst.com`. For details of the clock control registers, see Status and system control registers on page 4-18.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.7 Advanced Audio CODEC Interface, AACI

The FPGA contains an ARM PrimeCell *Advanced Audio CODEC Interface* (AACI) that provides communication with a CODEC using the AC-link protocol. This section provides a brief overview of the AACI. For detailed information, see *PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual*.

The AACI in the baseboard FPGA can be disabled and the signals driven by an AACI implemented in the FPGA of an attached Logic Tile in tile site 1 or 2.

Note

For a description of the audio CODEC signals, refer to the LM4549 data sheet available from the National Semiconductor web site. See also *Advanced Audio CODEC Interface, AACI* on page 4-35.

The AACI on the baseboard connects to a National Semiconductor LM4549 audio CODEC. The audio CODEC is compatible with AC'97 Rev 2.1. Table 3-8 lists the specifications for the audio system.

Table 3-8 Audio system specification

Characteristic	Value
Raw digital audio data format	PCM
Number of audio channels	Out 2 (stereo) In 1 of 2 (mono)
Audio sample data width	12, 16 or 18-bit native. Other data sizes require software conversion of sample data.
Sample rates supported	4kHz to 48kHz, variable in 1Hz steps. Record and playback sample rates can be independently selected.
Audio power output	250mW RMS into 32Ω

Figure 3-25 on page 3-48 shows the architecture of audio interface.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

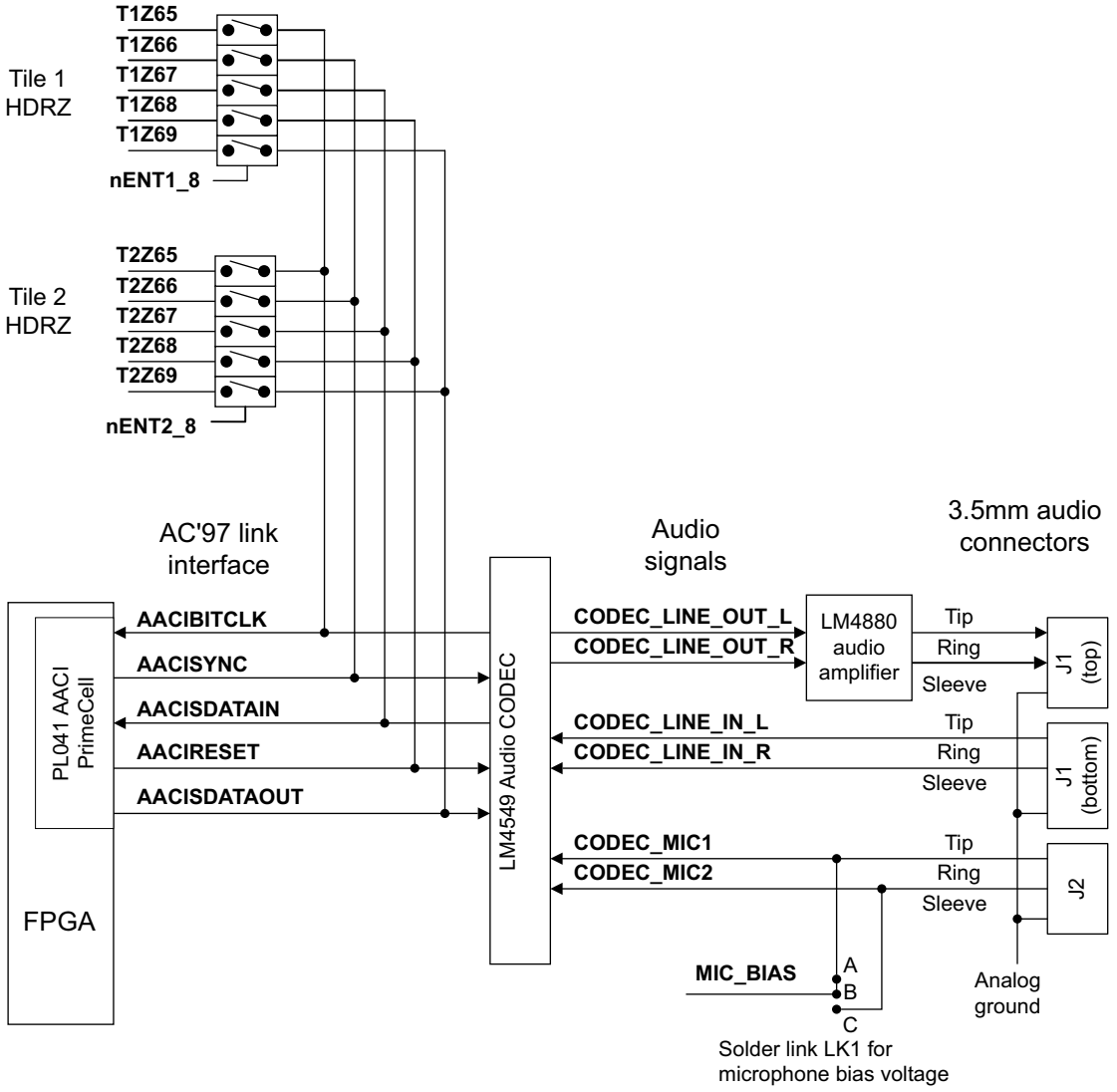

Figure 3-25 Audio interface

Two microphone inputs are present on J4. Only monophonic sound is supported, but microphone channel **CODEC_MIC1** or **CODEC_MIC2** can be selected in software. Solder link LK1 selects passive or active (electret) microphones:

Link AB Active microphone with power on **CODEC_MIC1** (tip). Passive microphone on **CODEC_MIC2** (not powered).

This is the default configuration.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Link BC Active microphone with power on **CODEC_MIC2** (ring). Passive microphone on **CODEC_MIC1** (not powered).

No link Passive microphone on **CODEC_MIC1** and **CODEC_MIC2**.

The signals associated with the audio CODEC interface are also assigned to connector J43, the AACI expansion socket pins, as shown in Table 3-9.

Note

The AACI expansion connector J45 is not fitted to the baseboard.

Table 3-9 AC'97 audio debug signals on J3

Pin number	Signal name	Description
1	**AACIBITCLK**	Clock from the CODEC to the AACI
2	**AACISYNC**	Frame synchronization signal from the AACI
3	**AACISDATAIN**	Serial data from the CODEC to the AACI
4	**AACI_RESET**	Reset signal from the AACI to the CODEC
5	**AACISDATAOUT**	Serial data from the AACI to the CODEC
6	**GND**	Signal ground

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.8 Character LCD controller

The FPGA contains a simple controller that provides an interface to a standard Varitronix MDL(S)-16263 16 x 2 character LCD alphanumeric display module.

The character display is accessed via a custom interface controller implemented in the FPGA. The standard ASCII character set is used to display data.

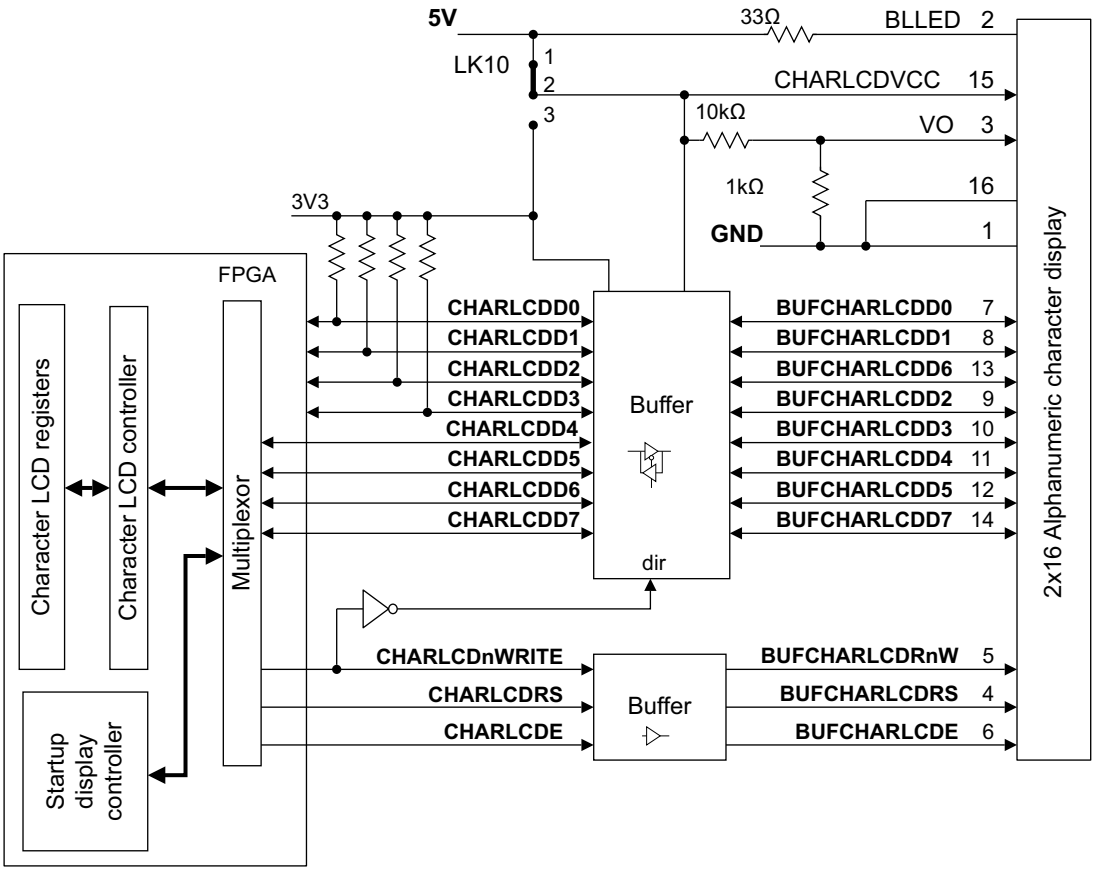

Figure 3-26 Character display

The character LCD display can function with either an eight-bit or four-bit data interface. Eight data signals are connected from the display to the FPGA, but only four signals are used by the FPGA to control the interface.

Link LK10 is fitted at manufacture to match the type of display device fitted.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.9 CLCDC interface

A PL111 PrimeCell CLCD controller is present in the FPGA.

A separate CLCD controller can be implemented in an attached Logic Tile. The peripheral switches (see *CLCDC routing switches* on page 3-12) control the routing from the CLCD controllers to the display interface logic and connectors.

The baseboard provides a display interface with outputs to:

- a VGA connector for connecting a VGA or SVGA monitor
- a CLCD adaptor board with CLCD, keypad, and touchscreen connectors. (See Appendix C *LCD Kits* for information on the touchscreen controller and the CLCD displays.)

A DAC re-arranges the CLCDC display signals for the selected resolution and color depth and converts the CLCD signals into VGA analog signals.

Use the *Synchronous Serial Port* (SSP) to access the touchscreen controller on the adapter board.

Figure 3-27 on page 3-52 shows the architecture of the display interface.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

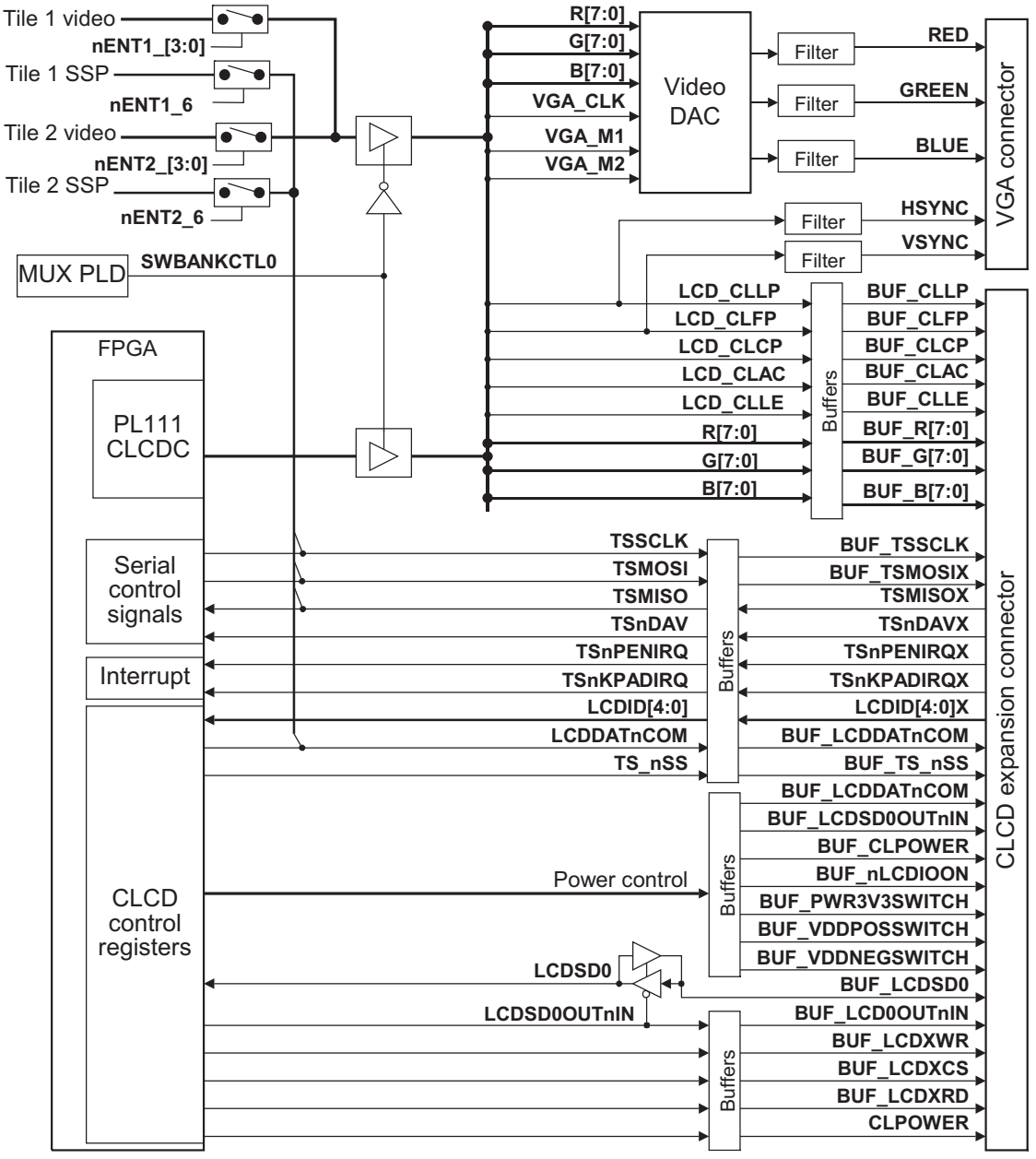

Figure 3-27 Display interface

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

See *Color LCD Controller, CLCDC* on page 4-40 for interface details.

Table 3-10 Display interface signals

Signal	Description
B[7:0]	Blue output signals to D/A converter and to or from the Logic Tile. A buffered version of these signals are output to the CLCD adaptor board as **BUF_B[7:0]**.
G[7:0]	Green output signals to D/A converter and to or from the Logic Tile. A buffered version of these signals are output to the CLCD adaptor board as **BUF_G[7:0]**.
R[7:0]	Red output signals to D/A converter and to or from the Logic Tile. A buffered version of these signals are output to the CLCD adaptor board as **BUF_R[7:0]**.
RED, **GREEN**, **BLUE**	Analog output from D/A converter for red, green, and blue signals to VGA connector.
VGA_HSYNC	The VGA horizontal synchronization signal. (This is a filtered version of **LCD_CLLP**.)
VGA_VSYNC	The VGA vertical synchronization signal (This is a filtered version of **LCD_CLFP**.)
VGA_M[1:0]	These signals select the VGA display resolution. The signals control remapping of the **R[7:0]**, **G[7:0]**, and **B[7:0]** data signals for the VGA display.
VGA_CLK	The VGA clock synchronizes the conversion of the **B[7:0]**, **G[7:0]**, and **R[7:0]** signals into the **RED**, **GREEN**, and **BLUE** analog signals.
nBLANK	The VGA video blanking signal.
TSSCLK	Clock to touchscreen controller.
TSMOSI	Data from touchscreen controller.
TSMISO	Data from touchscreen controller.
TSnDAV	Touchscreen controller data available signal.
TSnPENIRQ	Touchscreen controller pen down interrupt.
TSnKPADIRQ	Touchscreen controller key pressed interrupt.
TSnSS	Touchscreen controller chip select.
Power control	The **nLCDIOON**, **CLPOWER**, **PWR3V5VSWITCH**, **VDDNEGSWITCH**, and **VDDPOSSWITCH** signals can be used by the LCD adaptor board to select voltage for the display panel. Links on the board are set at manufacture to specify whether the panel voltages are fixed or programmable.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 3-10 Display interface signals (continued)

Signal	Description
LCDID[4:0]	These signals are determined by resistor links on the LCD adaptor board. They indicate the type of display that is attached to the adaptor board.
LCDDATnCOM	This signal indicates to the external controller on the CLCD expansion board whether the current value is data or a command.
LCDSD0	Serial data in or out for an external controller on the CLCD expansion board.
LCDSD0OUTnIN	This signal controls the direction of the serial data bus.
LCDXWR	Write signal to an external controller on the CLCD expansion board.
LCDXCS	Chip select signal to an external controller on the CLCD expansion board.
LCDXRD	Read signal to an external controller on the CLCD expansion board.
CLD[23:0]	LCD panel data. This is the digital RGB signals and synchronization signals.
CLCP	LCD panel clock.
CLLP	Line synchronization pulse (STN)/horizontal synchronization pulse (TFT).
CLFP	Frame pulse (STN)/vertical synchronization pulse (TFT). A buffered version of this signal is output to CLCD adaptor board as **BUF_CLFP**.
CLAC	STN AC bias drive or TFT data enable output. A buffered version of this signal is output to the CLCD adaptor board as **BUF_CLAC**.
CLLE	Line end signal. A buffered version of this signal is output to the CLCD adaptor board as **BUF_CLLE**. This signal can also be driven to the Logic Tile on **LT_CLLE**.
CLPOWER	LCD panel power enable. Depending on the link settings on the CLCD adaptor board, this signal can be used to turn off power to the display.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.10 DMA

A PrimeCell DMA controller might be implemented in the FPGA. (Some application notes describe builds that remove the DMA controller in order to increase performance. Refer to the application note that covers your configuration for more details.)

DMA requests from tile sites use DMA channels 0–2. Peripherals in the FPGA use DMA channels 6–15.

DMA control signals for channels 0–2 are passed to the DMA mapping multiplexors in the FPGA. Figure 3-28 on page 3-56 shows the DMA architecture.

See also *Direct Memory Access Controller* on page 4-48.

Note

The DMA control signals have pull-up or pull-down resistors as appropriate. It is not necessary therefore to drive unused signals.

The DMA control signals for external devices are listed in Table 3-11.

Note

Some FPGA peripherals do not use all of the DMA control signals. The USB controller, for example, uses only the **DMACSREQ** and **DMACCLR** signals.

Table 3-11 DMA signals for external devices

Signal	Description
DMACBREQ[5:0]	*Burst request* inputs to DMAC for channels 5 to 0.
DMACLBREQ[5:0]	*Last burst request* inputs to DMAC for channels 5 to 0.
DMACSREQ[5:0]	*Single request* inputs to DMAC for channels 5 to 0.
DMACLSREQ[5:0]	*Last single request* inputs to DMAC for channels 5 to 0.
DMACCLR[5:0]	*Clear* outputs from DMAC. These signals acknowledge the request from the corresponding **DMASREQ** or **DMABREQ** signals.
DMACTC[5:0]	*Terminal count* outputs from DMAC

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

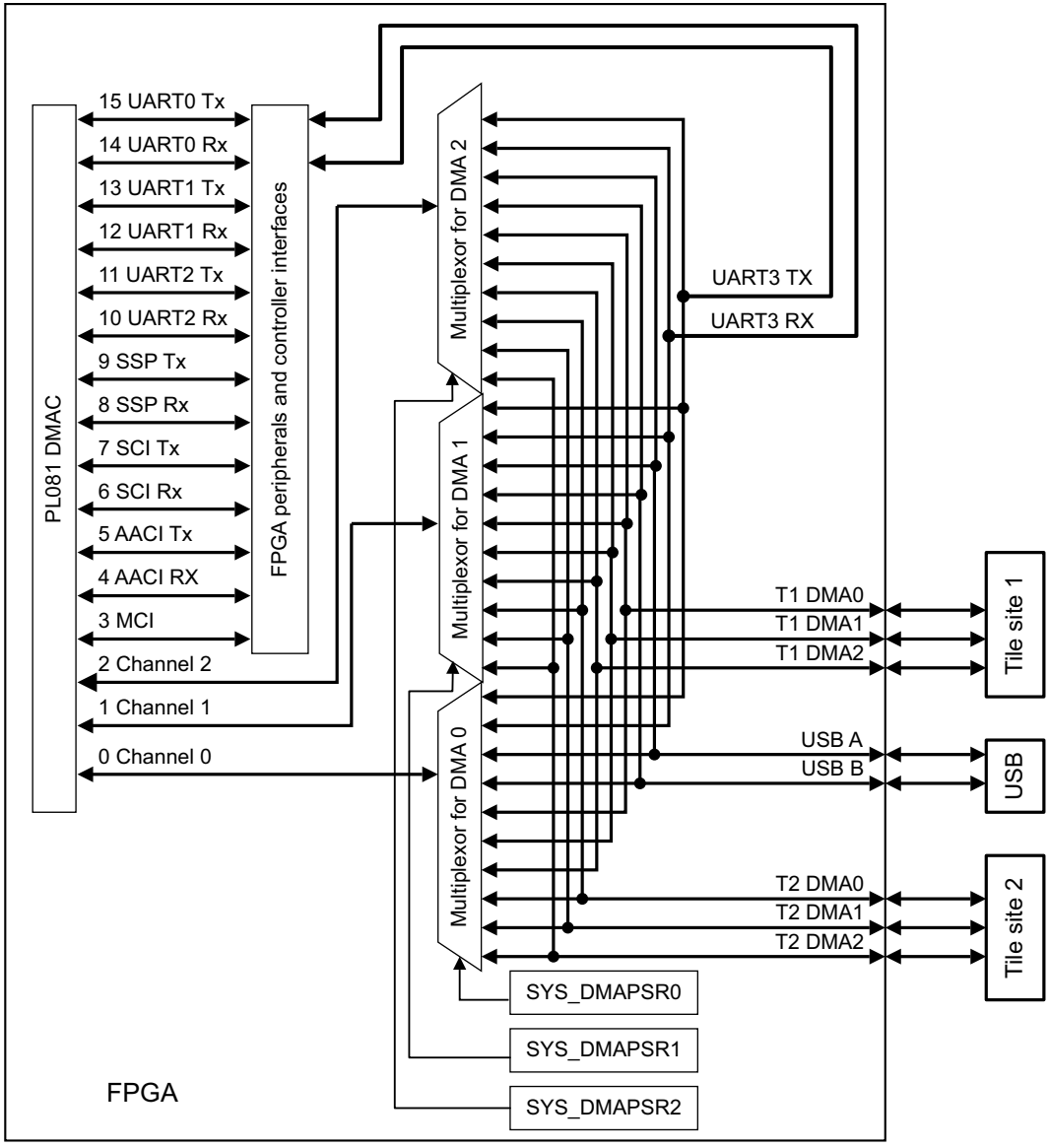

Figure 3-28 DMA channels

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.11 Ethernet interface

The Ethernet interface is implemented with a SMC LAN91C111 10/100 Ethernet single-chip MAC and PHY. This is provided with a interface to the static memory bus of the FPGA.

The internal registers of the LAN91C111 are mapped onto the static memory bus and occupy 16 words starting at location `0x4E000000`.

The isolating RJ45 connector incorporates two network status LEDs. The function of the LEDs can be set to indicate link, activity, transmit, receive, full duplex, or 10/100 selection. See the data sheet for the LAN91C111 for more details on programming the registers.

The architecture of the Ethernet interface is shown in Figure 3-29. See also Figure 3-19 on page 3-34.

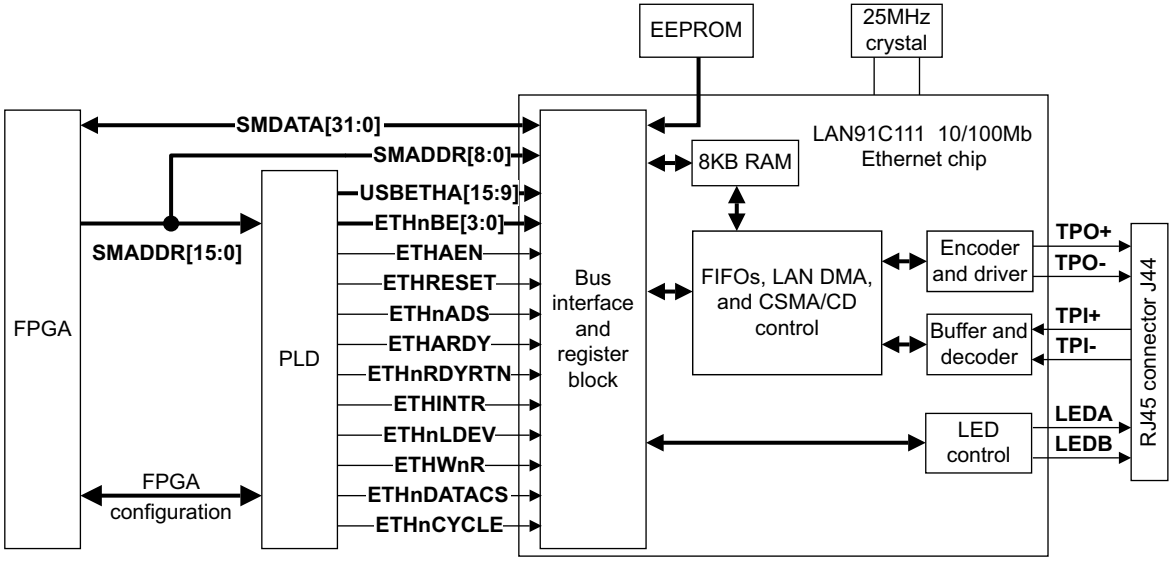

Figure 3-29 Ethernet interface architecture

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 3-12 Ethernet signals

Signal	Description
SMDATA[31:0]	Data lines to Ethernet controller from the SMC.
USBETHA[15:9]	Address lines to USB and Ethernet controllers from the configuration PLD.
SMADDR[8:0]	Address lines to Ethernet controller from the SMC.
ETHnBE[3:0]	Byte-enable signals to Ethernet controller
TPO+, **TPO-**	Signal from controller to Ethernet interface
TPI+, **TPI-**	Signal from interface to controller
LEDA, **LEDB**	Activity indicator LEDs. The function of the LEDs can be configured by writing to a LAN91C111 register.
ETHRESET	Reset signal to LAN91C111
ETHARDY	Asynchronous ready signal
ETHnRDYRTN	Signals to the controller to complete synchronous read cycles
ETHnADS	Latches address to controller
ETHLCLK	Clock to controller interface
ETHnRD	Read signal for asynchronous interface
ETHnWR	Write signal for asynchronous interface
ETHnDATACS	Enables accesses to the controller data path
ETHnCYCLE	Used to control EISA burst mode synchronous cycles if LOW
ETHAEN	Address valid signal to controller.
ETHnLDEV	Asserted LOW if the address enable signal, **ETHAEN**, is low and the address lines decode to the controller address programmed into the base address register
ETHWnR	Defines bus direction for synchronous accesses

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.11.1 About the SMSC LAN91C111

The SMCS LAN91C11 is a fast Ethernet controller that incorporates a *Media ACcess* (MAC) Layer, a *PHYsical* (PHY) layer, and an 8KB dynamically configurable transmit and receive FIFO.

The controller supports dual-speed 100Mbps or 10Mbps and auto configuration. When auto configuration is enabled, the chip is automatically configured for network speed and for full or half-duplex operation.

The controller uses a local VL-Bus host interface that is mapped onto the static memory bus by the configuration PLD. The FPGA generates the appropriate access control signals for the host side of the Ethernet controller.

The LAN91C111 is a little-endian device. The default configuration for the system bus is also little-endian. If you configure the system bus for big-endian operation you must perform half-word and byte swapping in software.

A serial EEPROM provides the following parameters to the LAN91C111 at reset:

- the individual MAC address, that is, the Ethernet MAC address
- *Media Independent Interface* (MII) interface configuration
- register base address.

When the baseboard is 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. A unique MAC address is programmed at manufacture, but the address can be reprogrammed if required. Reprogramming of the EEPROM is done through Bank 1 (general and control registers).

Caution

It is possible that application software corrupts the EEPROM contents by writing into the MAC address registers. The `fix_EEPROM` utility on the CD (and available as a download from the ARM website) can repair the corrupted EEPROM by writing a factory default back to the MAC address. The `fix_EEPROM` utility can also be used to program any Ethernet MAC address to the baseboard.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.12 GPIO interface

The *General Purpose Input Output* (GPIO) signals **GP0_[7:0]** and **GP1_[7:0]** from the FPGA are connected to the GPIO connector as shown in Figure 3-30.

The GPIO signals are also connected to the tile sites. This enables you to use the GPIO signals with custom logic you implement in the tile. The signals can also be used to generate interrupts. The GPIO is limited to medium-speed signals (less than 5MHz).

See also *General Purpose Input/Output, GPIO* on page 4-55 and the *ARM PrimeCell GPIO (PL061) Technical Reference Manual*. See *GPIO interface* on page A-7 for connector pinout information.

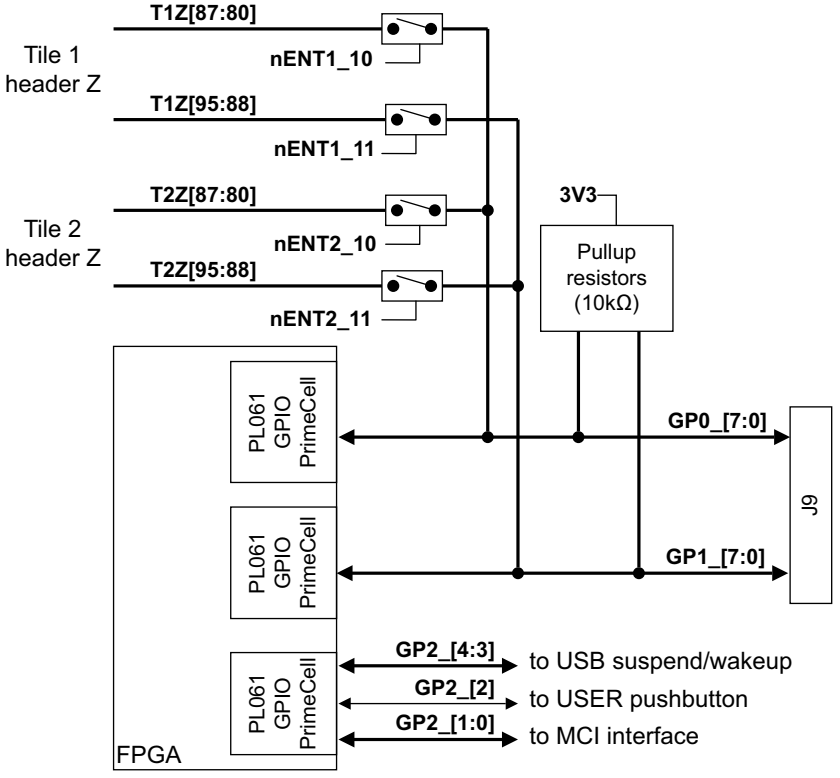

Figure 3-30 GPIO block diagram

GPIO2 is dedicated onboard I/O for the USB, USER push button and the MCI interrupts for card present and write protect. See *Multimedia Card Interface, MCI* on page 3-64 and *USB interface* on page 3-81.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.13 Interrupts

The baseboard FPGA contains four *Generic Interrupt Controllers* (GICs) as shown in Figure 3-31 on page 3-62. The GICs handle the nFIQ and nIRQ interrupts to tile site 1 and tile site 2:

GIC1 generates the **nIRQ** interrupt for tile site 1

GIC2 generates the **nFIQ** interrupt for tile site 1

GIC3 generates the **nIRQ** interrupt for tile site 2

GIC4 generates the **nFIQ** interrupt for tile site 2.

All four interrupt controllers are connected to the interrupt requests outputs from the FPGA devices, external peripherals, the PCI expansion bus, and the PISMO memory expansion bus

Tile site 1 interrupt requests (other than **COMMRX** and **COMMTX**) Tile are driven by the EB, the actual signals used are dependant on the application note.

Tile site 2 interrupt requests (other than **COMMRX** and **COMMTX**) connect to all four GICs on the EB, the actual signals used are dependant on the application note.

Note

Refer to the appropriate *Application Note* for details of the supported interrupt interconnect when using a particular EB and Core Tile combination.

The interrupt masking must be set so that only one interrupt is generated for an interrupt request.

For details on the programming model for the interrupt controllers and the interrupt assignments, see *Interrupt controller registers* on page 4-56.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

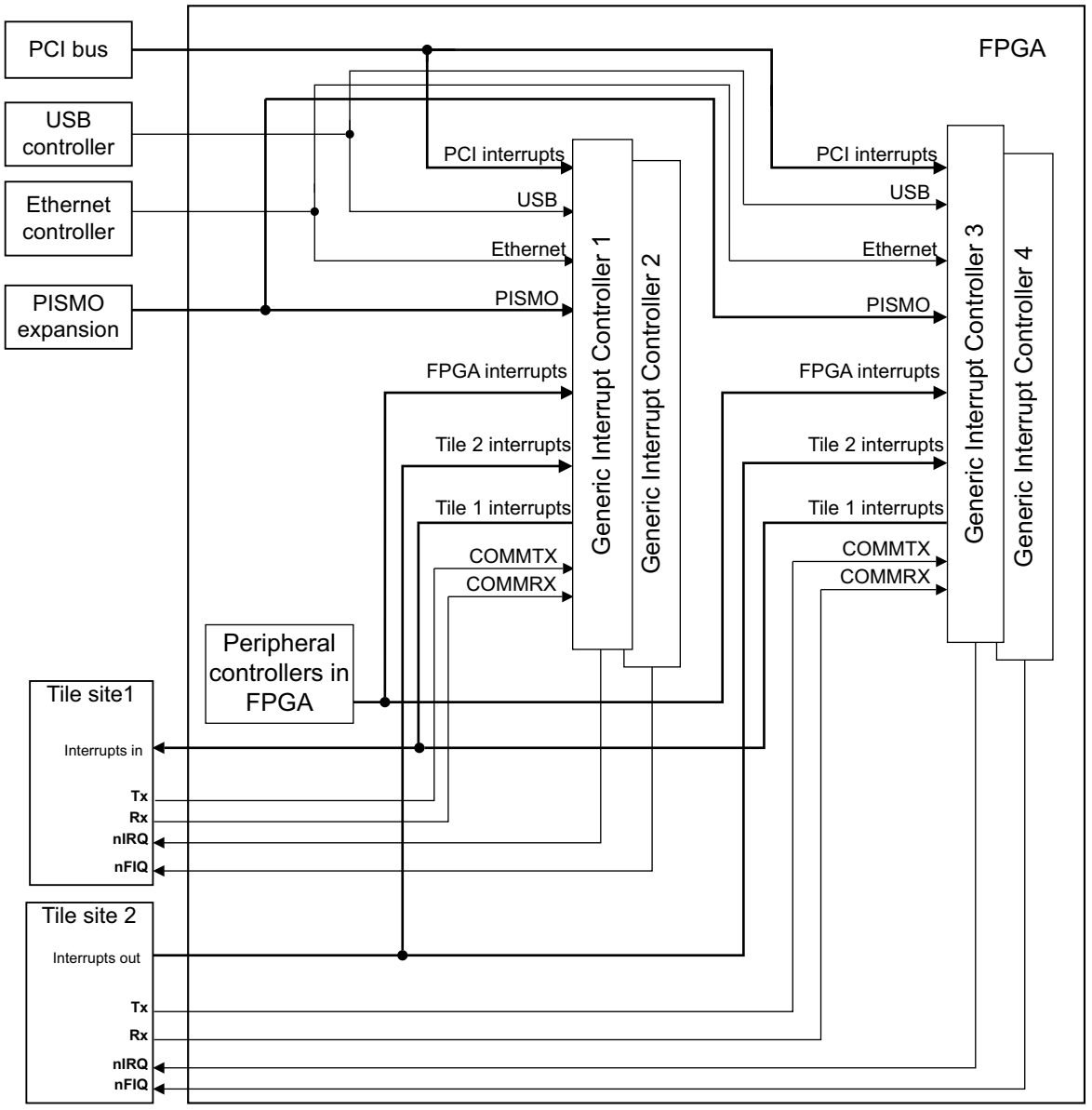

Figure 3-31 Interrupt controllers for tile site 1 and 2

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.14 Keyboard/Mouse Interface, KMI

The *Keyboard and Mouse Interfaces* (KMI) are implemented with two PrimeCells incorporated into the FPGA. This is shown in Figure 3-32.

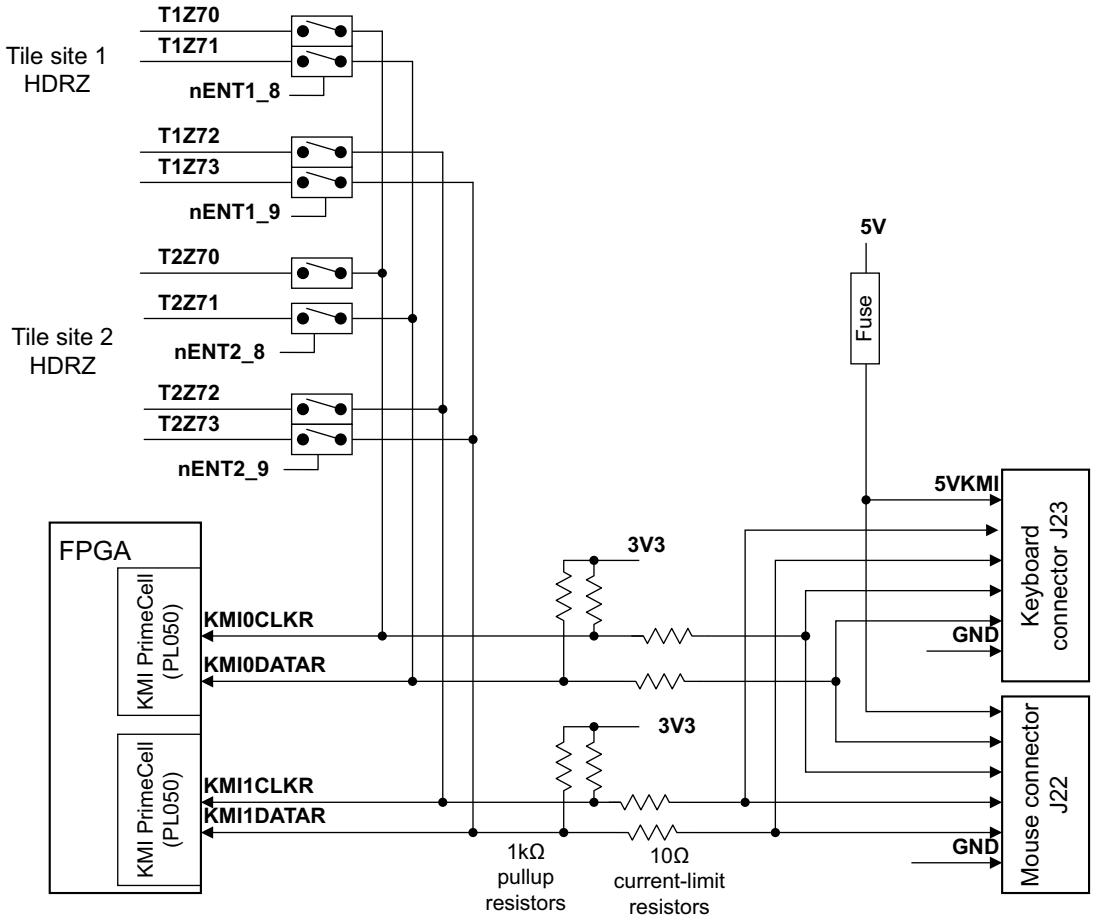

Figure 3-32 KMI block diagram

The keyboard and mouse signals go to both J22 and J23. Use a splitter cable to use only one of the connectors. The normal I/O pins for J22 are used for the mouse and the normally unused pins are used for the keyboard. J23 uses the normal I/O pins for the keyboard. See also *Keyboard and Mouse Interface, KMI* on page 4-66 and the *ARM PrimeCell PS2 Keyboard Mouse Controller (PL050) Technical Reference Manual.*

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.15 Multimedia Card Interface, MCI

An ARM PL180 PrimeCell MCI provides the interface to a *MultiMedia Card* (MMC) or *Secure Digital* (SD) card.

The interface can be driven as either an MMC or SD interface.

3.15.1 MMC or SD operation

The MMC socket provides nine pins that connect to the card when it is inserted into the socket. (The nine-way socket is compatible with Secure Digital cards. However MultiMedia Cards use only seven of the nine pins.)

The socket contains two switches that are operated by inserting or removing the card. These are used to provide signaling on the **nCARDIN** and **WPROT** signals.

The function of the interface signals depends on whether an MMC or SD card is fitted. Both card types default to MMC mode but the SD card has an additional operating mode called widebus mode. Table 3-13 lists the use of the signals for both modes of operation.

Table 3-13 MMC/SD interface signals

Signal	Widebus mode (SD card only)	MMC mode (default)
MCIDAT3	card detect/Data(3)	chip select (active LOW)
MCICMD	command/Response	command/Response
MCICLK	clock	clock
MCIDAT0	data (0)	data
MCIDAT1	data (1)	not used
MCIDAT2	data (2)	not used
nCARDIN	card presence detect (active LOW)	card presence detect (active LOW)
WPROT	card write-protection detect	card write-protection detect

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.15.2 Card insertion and removal

Insert the card into the socket with the contacts face down as viewed from the bottom of the board. Cards are normally labelled on the top surface and provide an arrow to indicate the correct way to insert them.

Remove the card by gently pressing it into the socket. It springs back and can be removed. This ensures that the card detection switches within the socket operate correctly.

3.15.3 Card interface description

Figure 3-33 on page 3-66 shows the memory card interface.

Table 3-14 MMC signals

Signal	Description
MCIPWR	Enables supply voltage to card
MCIxCMD	Command selection
CARDIN	Card detect signal. Read the current state from bit 1 of GPIO2.
MCIxDAT[3:0]	Card data bus
WPROT	Write protection indication. Read the current state from bit 0 of GPIO2.
MCICLK	Clock to card

See *MMC and SD flash card interface* on page A-9 for details of the MMC/SD card socket and pin numbering.

See also *MultiMedia Card Interface, MCI* on page 4-67 and the *ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual*.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

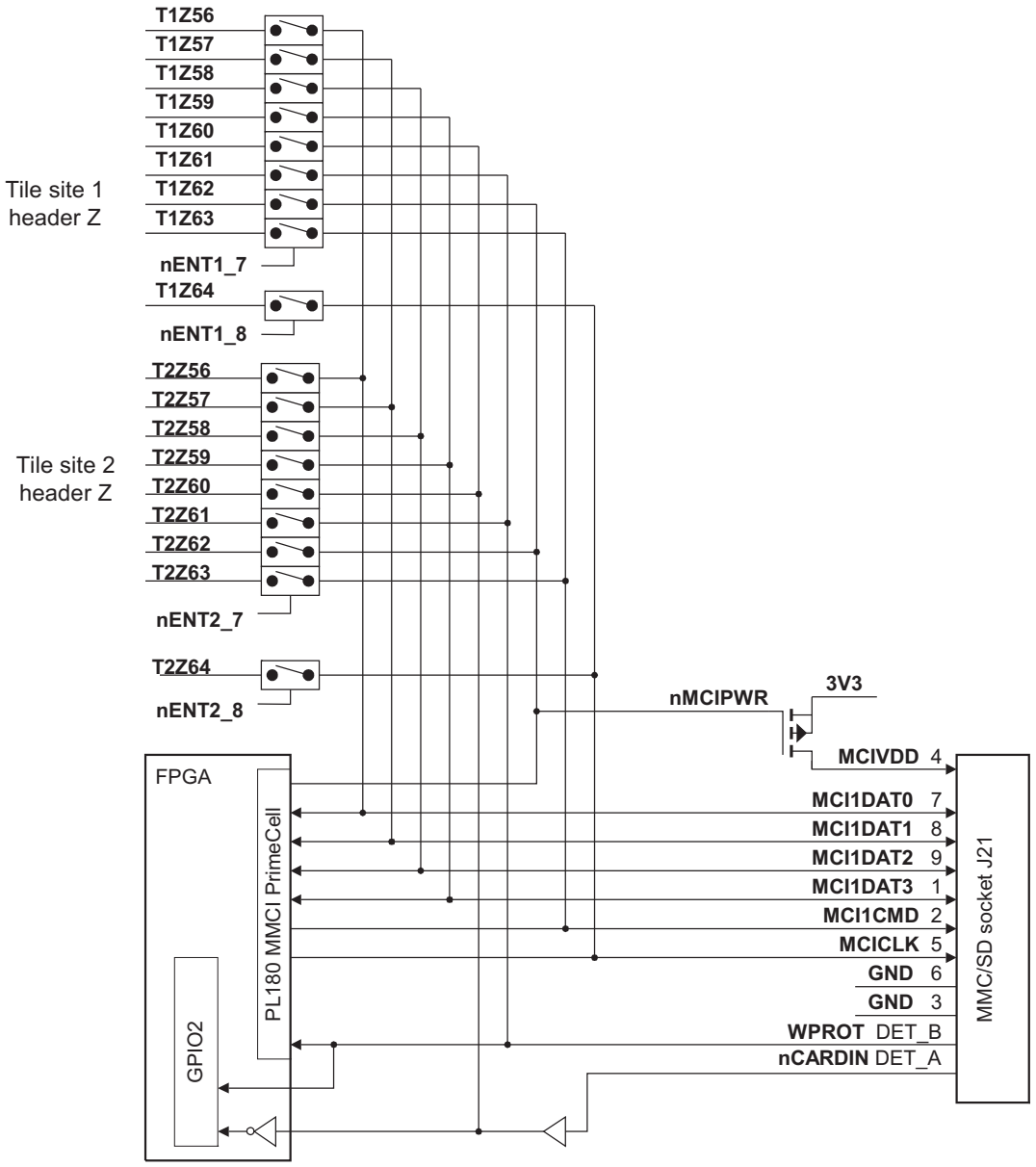

Figure 3-33 MCI interface

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.16 PCI interface

The Xilinx PCI subsystem enables you to use the baseboard in a PC or to use third-party PCI expansion cards with the baseboard and the PCI enclosure.

PCI bridges pass valid accesses between the baseboard and the PCI bus. The PCI interface supports 5V and 3.3V PCI buses. The baseboard functions as a PCI daughter-card, that is it does not generate clocks but uses external clocks from the PCI enclosure.

The FPGA bus matrix recognizes addresses `0x60000000` to `0x6FFFFFFF` as being intended for a target within the PCI address space of the memory map, and passes accesses within this region to the PCI bus. The PCI_IMAPx registers define the address translation values for the PCI I/O, PCI configuration, and PCI memory windows.

The PCI_SMAPx registers define the address translation values for PCI accesses to the FPGA internal bus.

The AHB to PCI bridge supports read and write accesses in both directions, as shown in Figure 3-34 on page 3-68.

Caution

The PCI controller is provided by Xilinx. The source HDL for this device is not provided on the CD. See the Xilinx web site for more information on the PCI controller.

The PCI controller will be deleted if you rebuild the FPGA image.

See Appendix D *PCI Backplane and Enclosure* and *PCI controller* on page 4-68.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

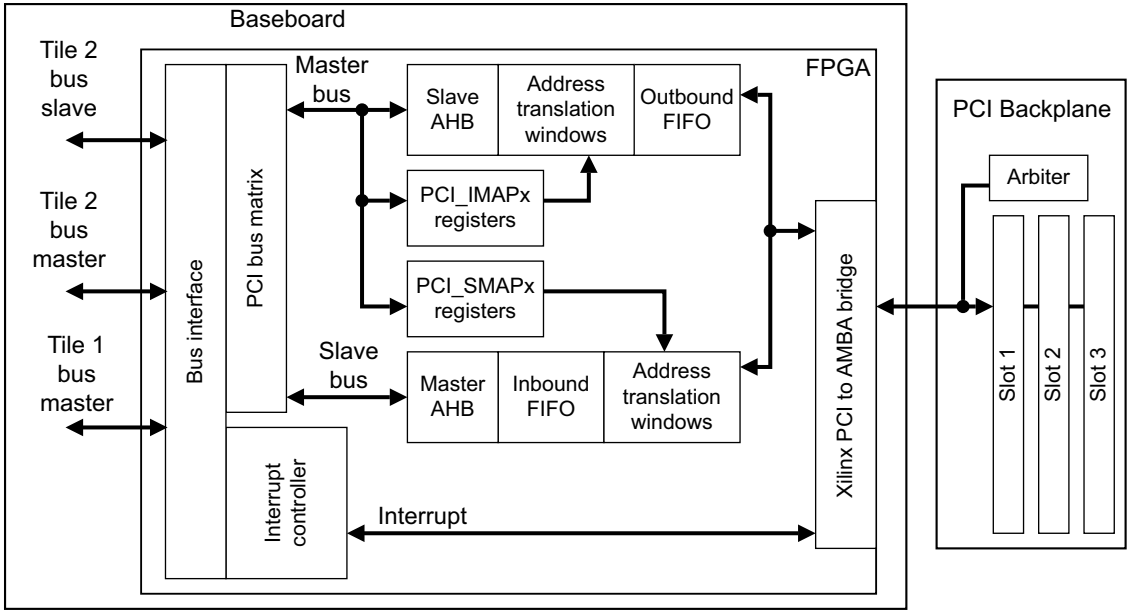

Figure 3-34 PCI bridge

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.17 Two-wire serial bus interface

The FPGA implements a two-wire serial bus interface that is used to identify the memory expansion modules and read and set the time-of-year clock.

Each device on the serial bus has its own slave address. The unique address for each slave on the serial bus is shown in Table 3-15.

Table 3-15 Serial bus addresses

Slave address (7-bit)	Slave device
b1010000	Reserved
b1010001	Static memory module
b1101000	Time-of-year clock

The block diagram of the interface is shown in Table 3-15. See *Two-wire serial bus interface* on page 4-81 for more information on the programming interface.

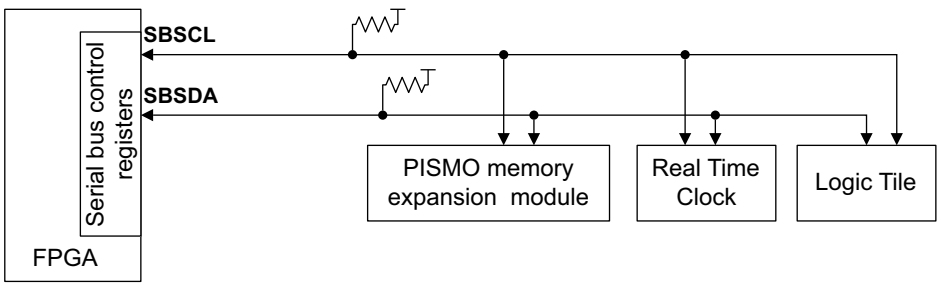

Figure 3-35 Serial bus block diagram

Table 3-16 Two-wire serial bus signals

Signal	Description
SBSCL	Open-collector clock. This clock is driven by the FPGA, but can be held LOW by an external device if it is not ready to receive or transmit data
SBSDA	Open-collector data signal.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.18 Smart Card interface, SCI

The baseboard FPGA contains a PrimeCell *Smart Card Interface* (SCI).

There are two types of Smart Card connector that can be fitted to the board:

- J31 is a small connector. This connector is not installed at manufacture.
- J33 is a large connector with a card detect switch. This connector is installed at manufacture.

Figure 3-36 on page 3-71 shows the tristate buffers that are used to provide the interface between the SCI and the smart card. The 16-way box header J28 enables you to monitor the signals or to connect an off-board smart card connector.

SCI output signals go to both the Logic Tile connectors and the Smart Card connector.

You can set the Smart Card interface voltage to operate at 5V, 3.3V or 1.8V by setting jumpers on J27.

- connect pins AB for 3.3V operation
- connect pins CB for 5V operation
- omit the link for 1.8V operation.

The default setting is linking pins AB. Both 3.3V and 5V cards will function with this setting.

Note

The Smart Card VCC is switched on and off by the **SCIVCCEN** signal from the PrimeCell.

See also *Smart Card Interface, SCI* on page 4-83 and the *SCI PrimeCell PL131 Technical Reference Manual*.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

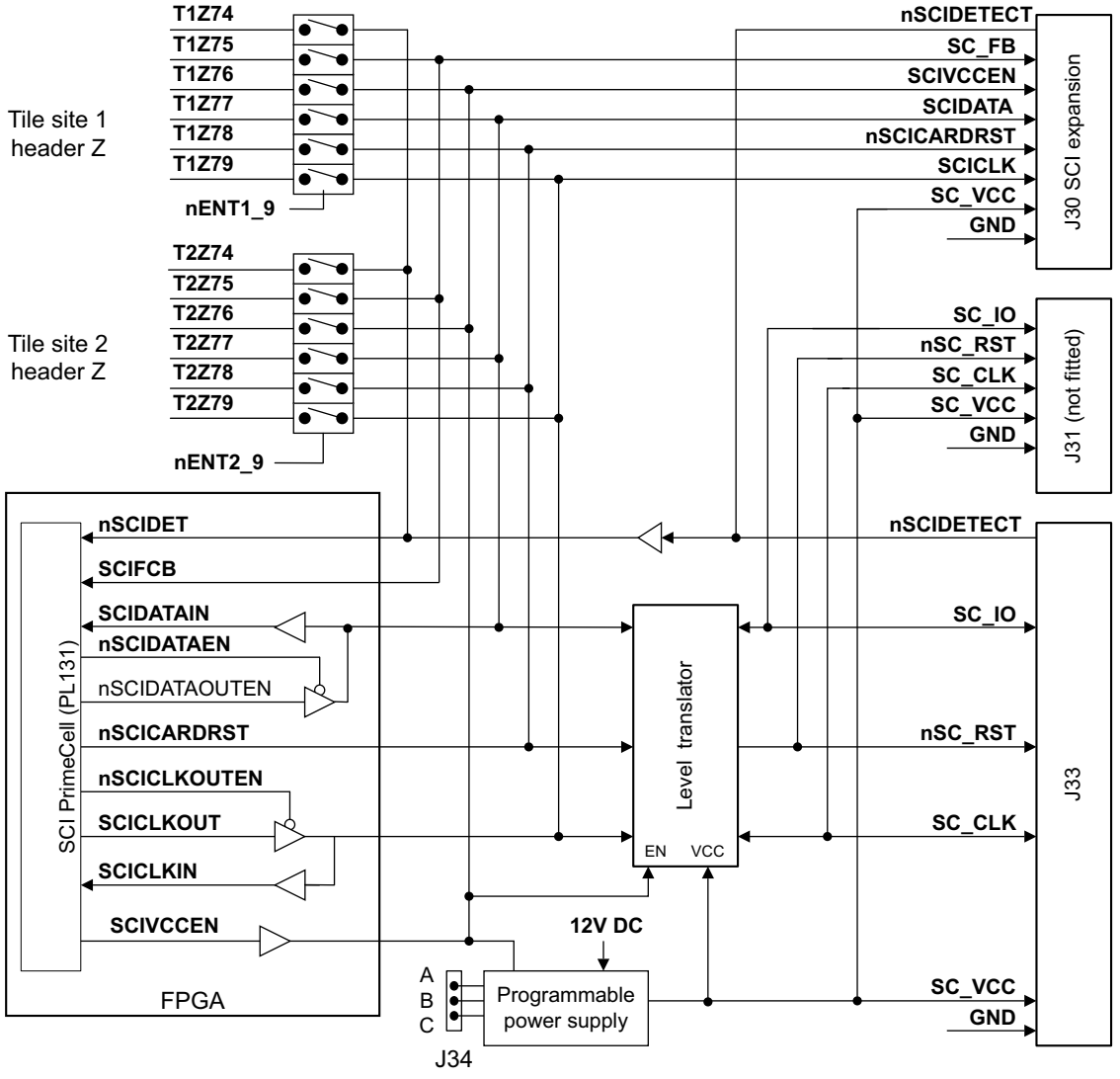

Figure 3-36 SCI block diagram

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 3-17 Smart Card interface signals

Signal	Description
SCICLKIN	PrimeCell SCI clock input.
nSCICLKEN	Tristate output buffer control for clock (active LOW).
SCICLKOUT	Clock output.
nSCIDATAEN	Tristate control for external off-chip buffer (active LOW).
SCIDATAIN	PrimeCell SCI serial data input.
nSCIDATAOUTEN	Data output enable (typically drives an open-drain configuration, active LOW).
nSCICARDRST	Reset to card (active LOW).
SCIFCB	Function code bit, used in conjunction with **nSCICARDRST**.
nSCIDET	Card detect signal from card interface device (active LOW).
SC_CLK	Smart Card clock (bidirectional)
SC_VCC	Programmable output voltage to connector and level translator
SCIDATA	Bidirectional data signal to level translator
SC_IO	Bidirectional data signal from level translator to connector

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.19 Synchronous Serial Port, SSP

The baseboard FPGA contains a PL022 PrimeCell SSP controller. Use the expansion connector J32 to connect to the SSP. The FPGA controls the SSP peripheral chip select, **SSPnCS**, as shown in Figure 3-37. The SSP signals are shared with the tiles sites and CLCD adaptor board.

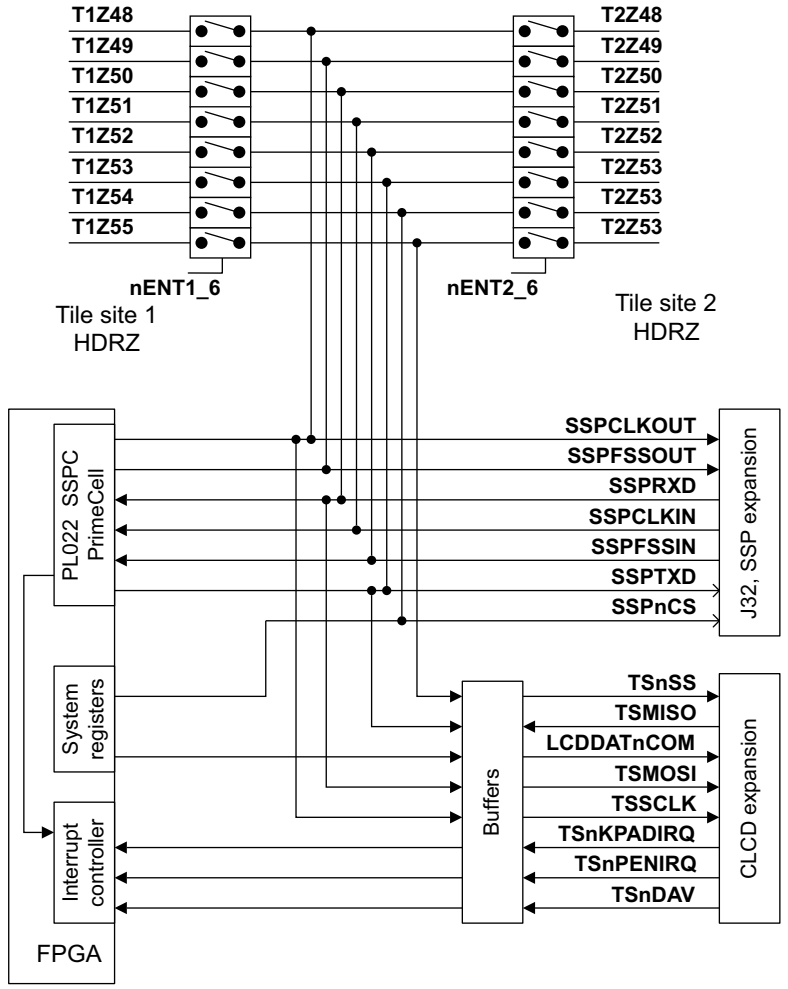

Figure 3-37 SSP block diagram

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 3-18 SSP signal descriptions

Name	Description
SSPnCS	Chip select to external device connected to SSP controller.
SSPFSSOUT	PrimeCell SSP frame or slave select output (master).
SSPCLKOUT	PrimeCell SSP clock output (master).
SSPRXD	PrimeCell SSP receive data input.
SSPTXD	PrimeCell SSP transmit data output.
nSSPCTLOE	Output enable signal (active LOW) for the **SSPCLKOUT** output from the PrimeCell SSP. This output is asserted (LOW) when the device is in master mode and de-asserted (HIGH) when the device is in slave mode.
SSPFSSIN	PrimeCell SSP frame input (slave).
SSPCLKIN	PrimeCell SSP clock input (slave).
nSSPOE	Output enable signal (active LOW) to indicate when **SSPTXD** is valid.

The SSP functions as 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.

Use the SSP controller to access:

- Touch screen, keypad, LCD bias, and analogue inputs on the optional LCD adaptor board. See Appendix C *LCD Kits*.
- Optional SSP devices, such as an EEPROM, that are connected to the expansion header J32.

  Note

  Although it is possible to connect both the CLCD adaptor board and an off board SSP device at the same time, care must be taken to ensure the correct SSP interface protocol is used when communicating with each device. The interface can be shared because the data from the touch screen controller data (**TSMISO**) is buffered with an open drain driver into **SSPRXD**.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

- Synthesized SSP peripherals in a Logic Tile FPGA. See *Connecting a Logic Tile* on page 2-11, *Tile headers and signal interconnects* on page 3-3 and the manual for your Logic Tile.

See also *Synchronous Serial Port, SSP* on page 4-84 and the *ARM PrimeCell Synchronous Serial Port Controller (PL022) Technical Reference Manual.*

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.20 User switches and LEDs

The FPGA provides a switch and LED register that enables you to read the general-purpose push button switch and the user switches (S6) and light the user LEDs (located next to switch S6). See Figure 1-1 on page 1-3 for the location of the switches and LEDs. Figure 3-38 shows the interface.

Note

Switch S6-1 and S6-2 are used to control the Boot Monitor. See *Boot Monitor configuration switch* on page 2-7.

Set bits [7:0] in the SYS_LED register at `0x10000008` to illuminate LEDs 7–0. The state of the user switches S6[8:1] is present on bits [7:0] of the SYS_SW register at `0x10000004`.

Note

The state of the general-purpose push button S3 can be read from GPIO 2. See *General Purpose Input/Output, GPIO* on page 4-55. The GPIO can be programmed to generate an interrupt when the switch is pressed.

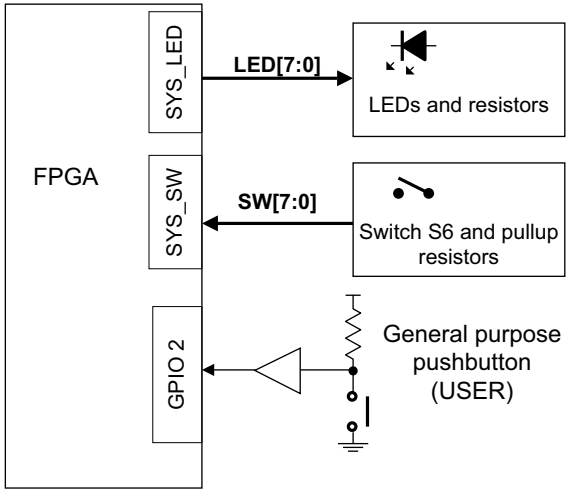

Figure 3-38 Switch and LED interface

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.21 UART interface

Four UARTs (SER0, SER1, SER2, and SER3) are implemented with PL011 PrimeCells incorporated into the baseboard FPGA.

The UARTs have the following features:

- functionally similar to standard 16C550 devices
- port function corresponds to the DTE configuration
- SER0 (UART0) has full CTS, RTS, DCD, DSR, DTR, and RI modem control signals
- SER1, SER2, and SER3 (UART1, UART2, and UART3) have simple modem control signals CTS and RTS
- programmable baud rates of up to 1.5Mbits per second (the line drivers however, are only guaranteed to 250kbps)
- 16-byte transmit FIFO
- 16-byte receive FIFO
- programmable interrupt.

Signals from UART0, UART1, and UART2 are also connected to the peripheral switches for the tile sites as shown in Figure 3-39 on page 3-78.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

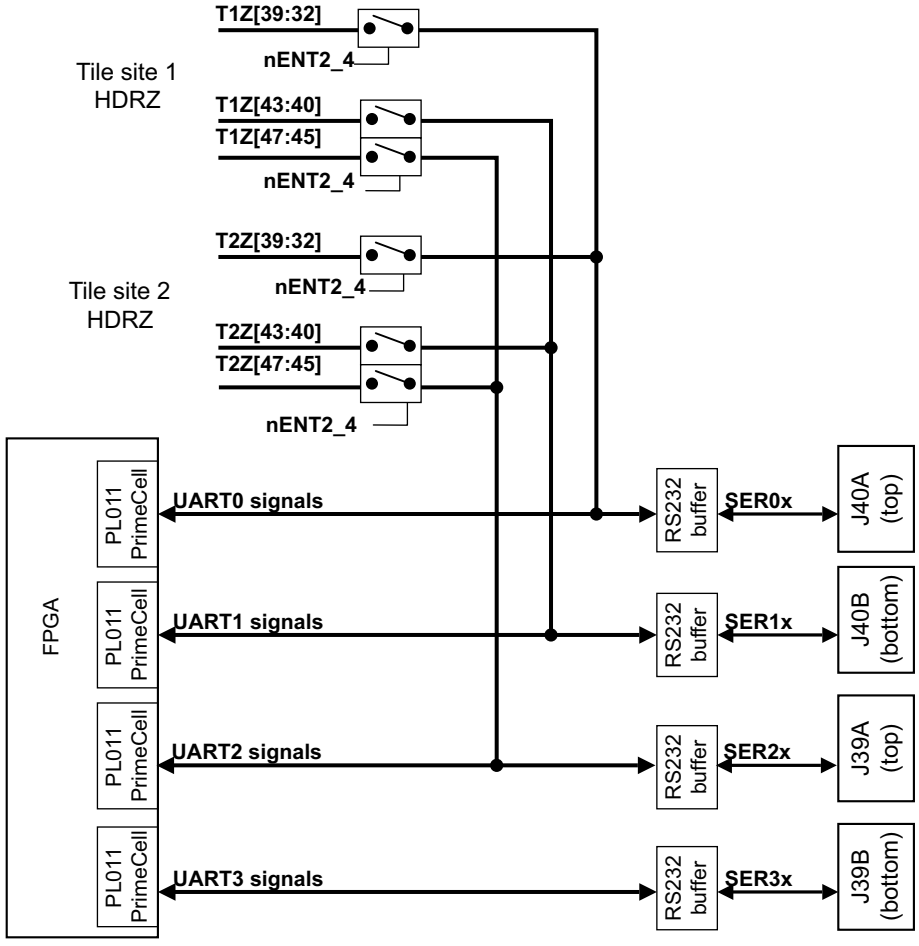

Figure 3-39 UARTs block diagram

The signals from the FPGA are converted from logic level to RS232 level by MAX3243E buffers as shown in Figure 3-40 on page 3-79 and Figure 3-41 on page 3-79.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

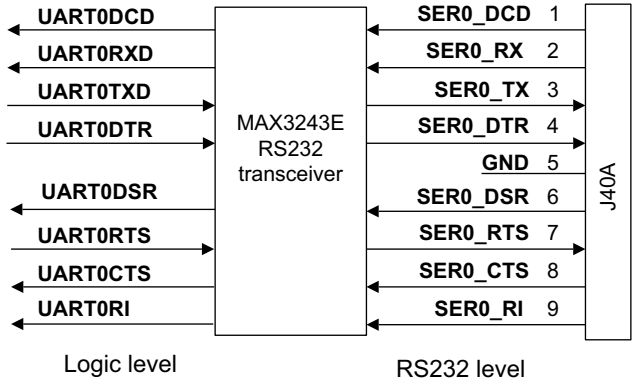

Figure 3-40 UART0 interface

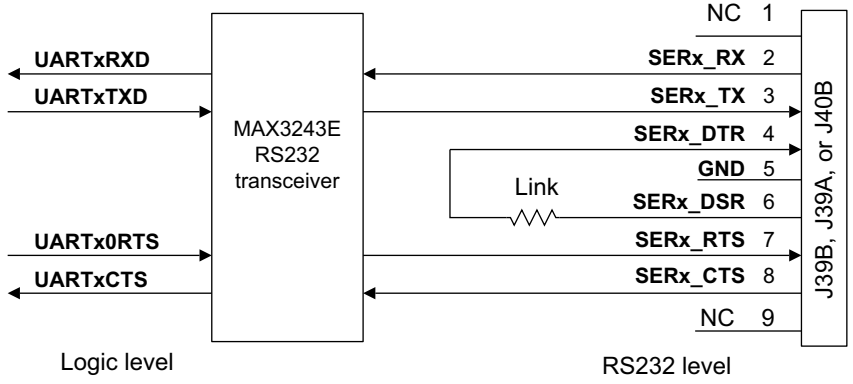

Figure 3-41 Simplified interface for UART[3:1]

See also *UART* on page 4-91 and the *ARM PrimeCell UART (PL011) Technical Reference Manual*.

Zero Ohm resistor links connect DTR to DSR for the following ports:

- R223 for UART1
- R224 for UART2
- R225 for UART3

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The signals associated with the UART interface are shown in Table 3-19.

Table 3-19 Serial interface signal assignment

Signal	Description
SERx_TXD	Transmit data
SERx_RTS	Ready to send
SERx_DTR[a]	Data terminal ready
SERx_CTS	Clear to send
SERx_DSR[a]	Data set ready
SERx_DCD[a]	Data carrier detect
SERx_RXD	Receive data
SERx_RI[a]	Ring indicator

a. For UART1, UART2, and UART3, the DTR and DSR signals are connected together and are not input to the FPGA. Also, the DCD and RI signals are not connected to the FPGA

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.22 USB interface

The FPGA provides the bus interface to an external Philips ISP1761 controller. Three USB interfaces are provided on the baseboard, see Figure 3-42. See also Figure 3-20 on page 3-35 for an overview of how the USB controller is attached to the static-memory bus.

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

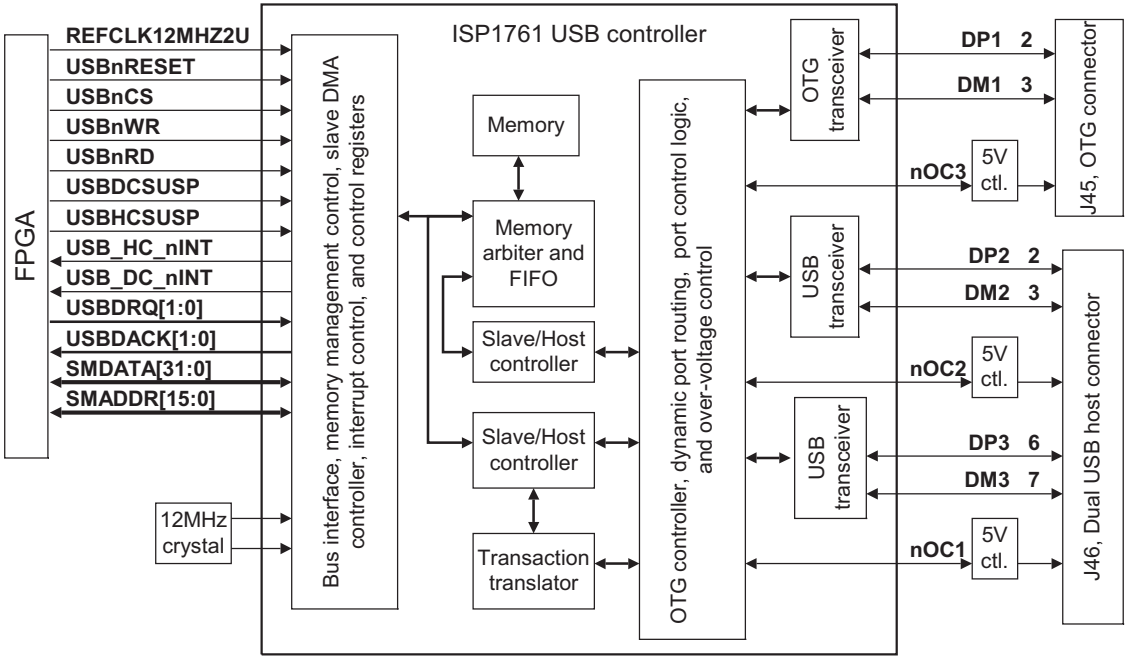

Figure 3-42 ISP1761 block diagram

USB port 1 provides an OTG device interface and connects to J45. USB ports 2 and 3 can function in either master or slave mode and connect to the dual type A connector J46 (USB2 is the top connector).

Note

The USB debug interface has a dedicated USB controller. See *JTAG and USB debug port support* on page 3-83.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The signals associated with the USB interfaces are shown in Table 3-20.

Table 3-20 USB interface signal assignment

Signal name	Direction	Description
DPx	Bidirectional	D+ data line
DMx	Bidirectional	D– data line
SMDATA[31:0]	Bidirectional	Data lines of USB controller
SMADDR[15:0]	From FPGA	Address lines of USB controller
USBnCS	From FPGA	Controller chip select
USBnRD	From FPGA	Read strobe to controller
USBnWR	From FPGA	Write strobe to controller
USBnINT	To FPGA	Controller interrupt out
USBnRESET	From FPGA	Controller reset
USBHCSUSP	From FPGA	GPIO2 in the FPGA drives this signal HIGH to wake up the host controller
USBDCSUSP	From FPGA	GPIO2 in the FPGA drives this signal HIGH to wake up the device controller
REFCLK12MHZ2U	From FPGA	24MHz reference clock to controller (this clock is not used and the 12MHz crystal output typically provides the reference clock)
nPO[3:1]	From FPGA	Switches 5V power to connectors
nOC[3:1]	From USB	Over current detect (disconnects power to connectors)
USBDRQ[1:0]	From FPGA	DMA request. **USBDRQ1** for channel 1, **USBDRQ0** for channel 0.
USBDACK[1:0]	To FPGA	DMA acknowledge. **USBDACK1** for channel 1, **USBEDACK0** for channel 0.

Note

For a full description of the USB controller, refer to the data sheet for the Philips ISP1761.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.23 Test, configuration, and debug interfaces

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

- JTAG, see *JTAG and USB debug port support*
- Logic analyzer, see *Integrated Logic Analyzer* on page 3-90
- Trace, see *Embedded trace support* on page 3-91
- Configuration switches and status indicators, see *Test, configuration, and debug interfaces* and *User switches and LEDs* on page 3-76.
- Boot Monitor, see *Using the baseboard Boot Monitor and platform library* on page 2-32.

Note

There are also test points and debug connectors for individual interface circuits. See *Test and debug connections* on page A-22 for the location of test connectors, links, switches, and indicators.

3.23.1 JTAG and USB debug port support

The baseboard supports debugging using embedded or external hardware. The debugging interface can be controlled by:

JTAG hardware

The RealView Debugger and the AXD debugger, for example, use an external interface box, such as RealView ICE or Multi-ICE, to connect to the JTAG connector. The `progcards_rvi.exe` (or `progcards_multiice.exe`) utility uses the JTAG interface to load FPGA and PLD images.

USB debug port

The USB debug port is embedded on the baseboard. The `progcards_usb.exe` utility controls the JTAG signals from the USB port of the PC. The PC and the baseboard are connected by a standard USB cable. The USB debug port is only used for loading FPGA and PLD images and is not available as a general debug port.

Note

ARM Multi-ICE and RealView ICE ground pin 20 of the JTAG connector as shown in Figure 3-43 on page 3-84. On the baseboard, pin 20 is connected to a pull-up resistor and the **nICEDETECT** signal. The USB debug 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

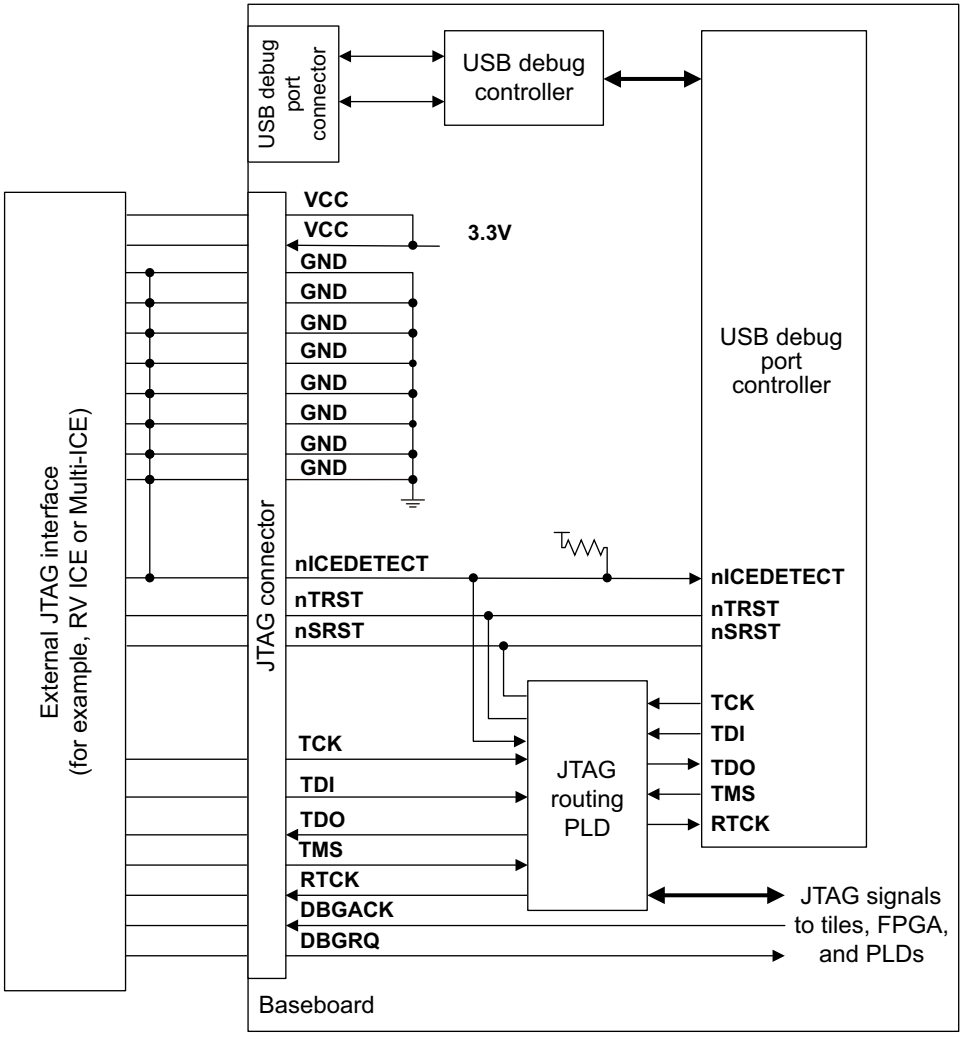

Figure 3-43 External connection to ground for nICEDETECT signal

The baseboard has two scan chains:

Debug The **D_x** signals are used for the processor in the Core Tiles and synthesized JTAG TAP controllers in the Logic Tile. This is the normal mode of operation (see *JTAG debug (normal) mode* on page 3-85).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Config The **C_x** signals are used to program the PLDs and to load the FPGA images into configuration memory. This chain is available in configuration mode (see *JTAG configuration mode* on page 3-86). See also *Integrated Logic Analyzer* on page 3-90.

JTAG debug (normal) mode

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

The debug mode is selected by default (when the CONFIG link is in the OFF position). In debug mode:

- The signal **nCFGEN** is HIGH.
- The CONFIG LED is OFF on the baseboard (and on each Logic Tile in the stack).
- A debugger, RealView Debugger for example, controls the scan chain.
- The PLDs and FPGAs are not visible on the scan chain unless they contain debuggable devices.

  Note

  The images supplied with the baseboard route the **D_TDO** and **D_TCK** signals through the FPGA. If you load a custom image into the FPGA, ensure that the image also routes the input signals to the output signals to prevent the debug chain from being broken.

- The JTAG signals are routed through the TAP controller in the Core Tile processor.
- If Logic Tiles are present and have debuggable devices, the **T1_D_x** and **T2_D_x** signals are present in the JTAG scan chain.

  If Logic Tiles are present, but do not have debuggable devices, the Logic Tile FPGA signal **D_TDI** must be routed to **D_TDO** and the **D_TCK** signal must be routed to **D_RTCK**
- The FPGAs in the system load their images from configuration flash.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

JTAG configuration mode

This mode is selected if the CONFIG switch is in the ON position. In configuration mode:

- The signal **nCFGEN** is low.
- The CONFIG LED is lit on the baseboard (and on each tile in the stack).
- The JTAG scan path is routed to include configurable devices.
- A configuration program (one of the `progcards` utilities for example) controls the scan chain.
- If one or more Logic Tiles are present, the **T1_C_x** and **T2_C_x** signals are part of the JTAG scan chain.
- All FPGAs and PLDs in the system (including any devices in a Logic Tile) are added into the scan chain.
- The TAP controller in the Core Tile processor is not visible and is replaced by a Boundary Scan TAP controller that is used for board-level production testing.

  Note

  Some versions of the CT7TDMI do not have a boundary scan tap controller and the test chip is bypassed when in configuration mode.

- This enables the board to be configured or upgraded in the field using JTAG equipment or the onboard USB debug port.
- The non-volatile PLD devices can be reprogrammed directly by JTAG.
- The FPGA image is loaded from the configuration flash, however, a new image for the FPGA or the configuration flash can be loaded from the scan chain.

  The FPGAs are volatile. In normal mode, they load their configuration from non-volatile flash memory. In configuration mode, they can be loaded from either JTAG or the configuration flash memory. (See *FPGA configuration* on page 3-17 for details of loading a PLD or FPGA image.)

  Note

  The configuration flash memory does not have a JTAG port, but it can be programmed using JTAG by loading a flash-loader design into the FPGAs and PLDs. The flash-loader can then transfer data from the JTAG programming utility to the configuration flash. This process is managed automatically by the `progcards` utility.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

After configuration you must:

1. move the config switch to the OFF position
2. power cycle the development system.

JTAG signals

Table 3-21 provides a description of the JTAG and related signals.

Note

In the description in Table 3-21, 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, Multi-ICE, or the embedded USB debug logic.

Table 3-21 JTAG related signals

Name	Description	Function
TDI	Test data in (from JTAG equipment)	**TDI** and **TDO** connect each component in the scan chain.
TDO	Test data out (to JTAG equipment)	**TDO** is the return path of the data input signal **TDI**. The JTAG components are connected in the return path so that the length of track driven by the last component in the chain is kept as short as possible.
TMS	Test mode select (from JTAG equipment)	**TMS** controls transitions in the TAP controller state machine.
TCK	Test clock (from JTAG equipment)	**TCK** synchronizes all JTAG transactions. **TCK** connects to all JTAG components in the scan chain. Series termination resistors are used to reduce reflections and maintain good signal integrity.
RTCK	Return **TCK** (to JTAG equipment)	Some devices sample **TCK** and delay the time at which a component actually captures data. Using a mechanism called *adaptive clocking*, the **RTCK** signal is returned by the core to the JTAG equipment, and the **TCK** is not advanced until the core has captured the data. In adaptive clocking mode, RealView ICE or Multi-ICE waits for an edge on **RTCK** before changing **TCK**. In a multiple device JTAG chain, the **RTCK** output from a component connects to the **TCK** input of the next device in the chain.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 3-21 JTAG related signals (continued)

Name	Description	Function
nCFGEN	Configuration enable	**nCFGEN** is an active LOW signal used to put the boards into configuration mode. In configuration mode all FPGAs and PLDs are connected to the scan chain so that they can be configured by the JTAG equipment. (The baseboard JTAG routing PLD is not accessible.)
nSRST	System reset (bidirectional)	**nSRST** is an active LOW open-collector signal that can be driven by the JTAG equipment to reset the target board. Some JTAG equipment senses this line to determine when a board has been reset by the user. This is also used in configuration mode to control the initialization pin (**nINIT**) on the FPGAs. Though not a JTAG signal, **nSRST** is described because it can be controlled by JTAG equipment.
nTRST	Test reset (from JTAG equipment)	This active LOW open-collector signal is used to reset the JTAG port and the associated debug circuitry on the tiles. It is asserted at power-up, and can be driven by the JTAG equipment. This signal is also used in configuration mode to control the programming pin, **nPROG**, on FPGAs.
DBGRQ	Debug request (from JTAG equipment)	**DBGRQ** is a request for the processor core to enter the debug state. It is provided for compatibility with third-party JTAG equipment. The processor enters debug state and issues a **DBGACK** to acknowledge the request.
GLOBAL_DONE	All FPGAs configured	**GLOBAL_DONE** is an open-collector signal that indicates when all FPGA configuration is complete. Although this signal is not a JTAG signal, it does affect **nSRST**. The **GLOBAL_DONE** signal is routed between all RealView boards.
nRTCKEN	Return **TCK** enable	**nRTCKEN** is an active LOW signal driven by any tile that requires **RTCK** to be routed back to the JTAG equipment. If **nRTCKEN** is HIGH, the baseboard drives **RTCK** LOW. If **nRTCKEN** is LOW, the baseboard drives the **TCK** signal back to the JTAG equipment.
nTILEDET	To JTAG routing PLD	**nTILEDET** is LOW if a tile is attached to the tile site. This signal controls routing the JTAG signals for the tile headers.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

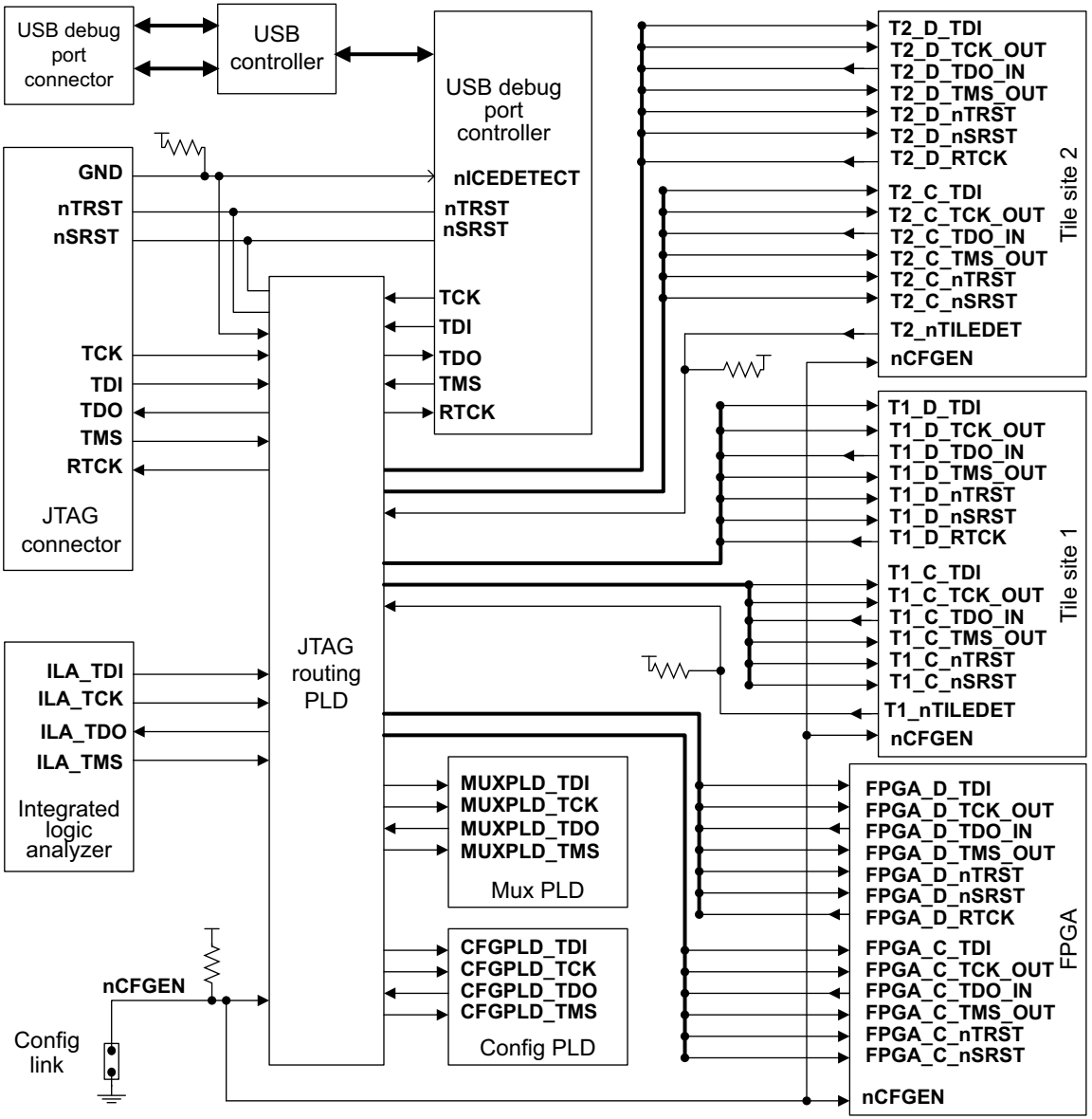

Figure 3-44 JTAG signal routing

The JTAG path chosen depends on whether the system is in configuration mode or debug mode. The CONFIG switch controls the **nCFGEN** signal that is routed through the baseboard and tile connectors. Figure 3-44 and Figure 3-45 on page 3-90 shows the JTAG signal routing.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

The JTAG routing PLD (not shown in the figures below) connects to both debug and configuration JTAG chains on the FPGA and tiles. The TAP chain selected is determined by the setting of the CONFIG switch.

The JTAG routing PLD is pre-programmed at manufacture and is not part of the JTAG scan chain in either mode.

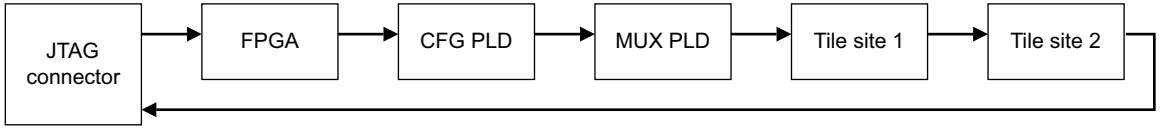

Figure 3-45 JTAG signal routing in config mode

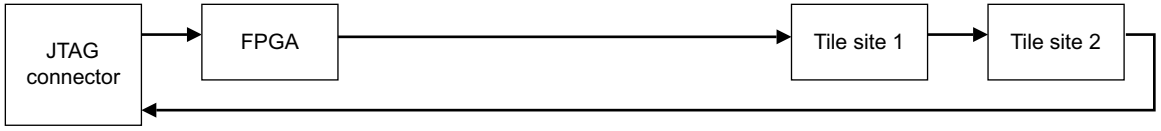

Figure 3-46 JTAG signal routing in debug mode

3.23.2 Integrated Logic Analyzer

The *Integrated Logic Analyzer* (ILA) connector J19 enables you to connect an ILA compatible analyzer to access the baseboard and Logic Tile FPGA JTAG signals at the same time as a JTAG debugger is being used to examine code on the CPU.

If the board is in debug mode, the configuration scan chain is not normally accessible. The integrated logic analyzer connector, however, provides access to the baseboard configuration scan chain and enables debugging of the baseboard and Logic Tile FPGA designs and the software simultaneously. For an example of an ILA system, see the ChipScope details on the Xilinx web site (www.xilinx.com).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

3.23.3 Embedded trace support

Many Core Tile test chips incorporate an *Embedded Trace Macrocell* (ETM). This enables you to carry out real-time debugging by connecting external trace equipment to the Trace connector on the Core Tile.

The baseboard also has two trace connectors. These are connected to the Logic Tile signals that are used to implement trace in an *Soft Macrocell Model* (SMM). These connectors do not connect to the trace signals present on a Core Tile.

To trace program flow, the ETM broadcasts branch addresses, data accesses, and status information through the trace port. Later in the debug process, the complete instruction flow can be reconstructed by the ARM *Trace Debug Tools* (TDT) or RealView Debugger.

Note

Connection of the trace port analyzer is described in *Connecting the Trace Port Analyzer to the baseboard* on page 2-16.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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:

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

- *Smart Card Interface, SCI* on page 4-83
- *Synchronous Serial Port, SSP* on page 4-84
- *Static Memory Controller, SMC* on page 4-86
- *System Controller* on page 4-88
- *Timers* on page 4-90
- *USB interface* on page 4-93
- *UART* on page 4-91
- *Watchdog* on page 4-95.

For detailed information on the programming interface for ARM PrimeCell peripherals and controllers, see the appropriate technical reference manual. For the DMA channels, interrupt signals, and release versions of ARM IP, see the section of this chapter that describes the peripheral.

Note

The peripherals and controllers implemented in the baseboard FPGA are in the standard images distributed by ARM Ltd.

You can replace some or all of the logic in the FPGA with your own designs, however, the details of how to do this are beyond the scope of this user guide. The CD includes FPGA HDL and signal definition files that can be used as a basis for custom designs.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.1 Memory map

The memory map is divided with sections assigned to the system, Core Tiles and Logic Tiles as shown in Table 4-1.

Table 4-1 System memory map

Owner	Address range	Bus type	Memory region size
System FPGA	0x00000000–0x0FFFFFFF	External or Internal	256MB
System FPGA	0x10000000–0x100FFFFF	Internal only	1MB
Reserved	0x10100000–0x103FFFFF	External only	3MB
System FPGA	0x10400000–0x17FFFFFF	Internal only	124MB
Logic Tile site 1	0x18000000–0x1FFFFFFF	External only	128MB
Reserved	0x20000000–0x3FFFFFFF	External only	512MB
System FPGA	0x40000000–0x7FFFFFFF	Internal only	1GB
Logic Tile site 2	0x80000000–0xFFFFFFFF	External only	2GB

Note

The memory region 0x00000000–0x0FFFFFFF can be mapped to:

- internal decode logic for access by the dynamic or static memory controller
- the external AXI or AHB bus for decode by an external device on the tile sites.

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

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The locations for memory, peripherals, and controllers for the standard FPGA image are listed in Table 4-2 and *System memory map for standard peripherals* on page 4-7.

Table 4-2 Memory map for standard peripherals

Peripheral	Address range	Bus type	Region size
Dynamic memory. During boot remapping however, the bottom 64MB of this memory region can be: • NOR flash • Disk-on-Chip • static expansion memory • memory on the external AXI or AHB bus (slave fitted on tile site 2).	0x00000000-0x0FFFFFFF	AHB or AXI	256MB
System registers	0x10000000-0x10000FFF	APB	4KB
System controller	0x10001000-0x10001FFF	APB	4KB
Two-Wire Serial Bus Interface	0x10002000-0x10002FFF	APB	4KB
Reserved	0x10003000-0x10003FFF	APB	4KB
Advanced Audio CODEC Interface	0x10004000-0x10004FFF	APB	4KB
MultiMedia Card Interface (MCI)	0x10005000-0x10005FFF	APB	4KB
Keyboard/Mouse Interface 0	0x10006000-0x10006FFF	APB	4KB
Keyboard/Mouse Interface 1	0x10007000-0x10007FFF	APB	4KB
Character LCD Interface	0x10008000-0x10008FFF	APB	4KB
UART 0 Interface	0x10009000-0x10009FFF	APB	4KB
UART 1 Interface	0x1000A000-0x1000AFFF	APB	4KB
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
Reserved	0x1000F000-0x1000FFFF	APB	4KB
Watchdog Interface	0x10010000-0x10010FFF	APB	4KB

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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

Peripheral	Address range	Bus type	Region size
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 `0x10020020`)	`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
Reserved	`0x10016000-0x10016FFF`	APB	4KB
Real Time Clock Interface	`0x10017000-0x10017FFF`	APB	4KB
Dynamic Memory Controller configuration	`0x10018000-0x10018FFF`	APB	4KB
PCI controller configuration registers	`0x10019000-0x10019FFF`	AHB	4KB
Reserved	`0x1001A000-0x1001FFFF`	APB	24KB
Color LCD Controller	`0x10020000-0x1002FFFF`	AHB	64KB
DMA Controller configuration registers	`0x10030000-0x1003FFFF`	AHB	64KB
Generic Interrupt Controller 1 (nIRQ for tile 1)	`0x10040000-0x1004FFFF`	AHB	64KB
Generic Interrupt Controller 2 (nFIQ for tile 1)	`0x10050000-0x1005FFFF`	AHB	64KB
Generic Interrupt Controller 3 (nIRQ for tile 2)	`0x10060000-0x1006FFFF`	AHB	64KB
Generic Interrupt Controller 4 (nFIQ for tile 2)	`0x10070000-0x1007FFFF`	AHB	64KB
Static Memory Controller configuration registers	`0x10080000-0x1008FFFF`	AHB	64KB
Reserved	`0x100A0000-0x100EFFFF`	AHB	448MB
Debug Access Port (DAP) ROM table Some debuggers read information on the target processor and the debug chain from the DAP table. This region might be unused on some baseboard variants.	`0x10090000-0x100FFFFF`	AHB	64KB
Reserved	`0x10100000-0x103FFFFF`	-	3MB
Reserved	`0x10400000-0x17FFFFFF`	AHB or AXI	124MB

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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

Peripheral	Address range	Bus type	Region size
Tile site 1 expansion If a tile is not fitted, the baseboard aborts accesses to this memory region	0x18000000-0x1FFFFFFF	AHB or AXI	128MB
Reserved	0x20000000-0x3FFFFFFF	-	512MB
SMC Chip Selects: • CS0 NOR flash (**nNORCS**) 0x40000000–0x43FFFFFF • CS1 DoC flash (**nDOCCS**) 0x44000000–0x47FFFFFF • CS2 SRAM (**nSRAMCS**) 0x48000000–0x4BFFFFFF • CS3 (**nSTATICCS3** and configuration PLD decoding) Config flash 0x4C000000–0x4DFFFFFF Ethernet 0x4E000000–0x4EFFFFFF USB 0x4F000000–0x4FFFFFFF • CS4 (**nEXPCS**) PISMO (**nCS0**) 0x50000000–0x53FFFFFF • CS5 (**nSTATICCS4**) PISMO (**nCS1**) 0x54000000–0x57FFFFFF • CS6 (**nSTATICCS5**) PISMO (**nCS2**) 0x58000000–0x5BFFFFFF • CS7 (**nSTATICCS6**) PISMO (**nCS3**) 0x5C000000–0x5FFFFFFF • CS8 (**nSTATICCS7**) PISMO (**nCS4**) reserved (tied HIGH)	0x40000000-0x5FFFFFFF	AHB or AXI	512MB
PCI interface bus windows • PCI SelfCfg window: 0x60000000–0x60FFFFFF • PCI Cfg window: 0x61000000–0x61FFFFFF • PCI I/O window: 0x62000000–0x62FFFFFF • PCI memory window 0: 0x63000000–0x63FFFFFF • PCI memory window 1: 0x64000000–0x67FFFFFF • PCI memory window 2: 0x68000000–0x6FFFFFFF	0x60000000-0x6FFFFFFF	AHB or AXI	752MB
Dynamic memory (mirror)	0x70000000-0x7FFFFFFF	AHB or AXI	256MB
Tile site 2 expansion If a tile is not fitted, the baseboard aborts accesses to this memory region	0x80000000-0xFFFFFFFF	AHB or AXI	2GB

Figure 4-1 on page 4-7 shows an overview of the memory map.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

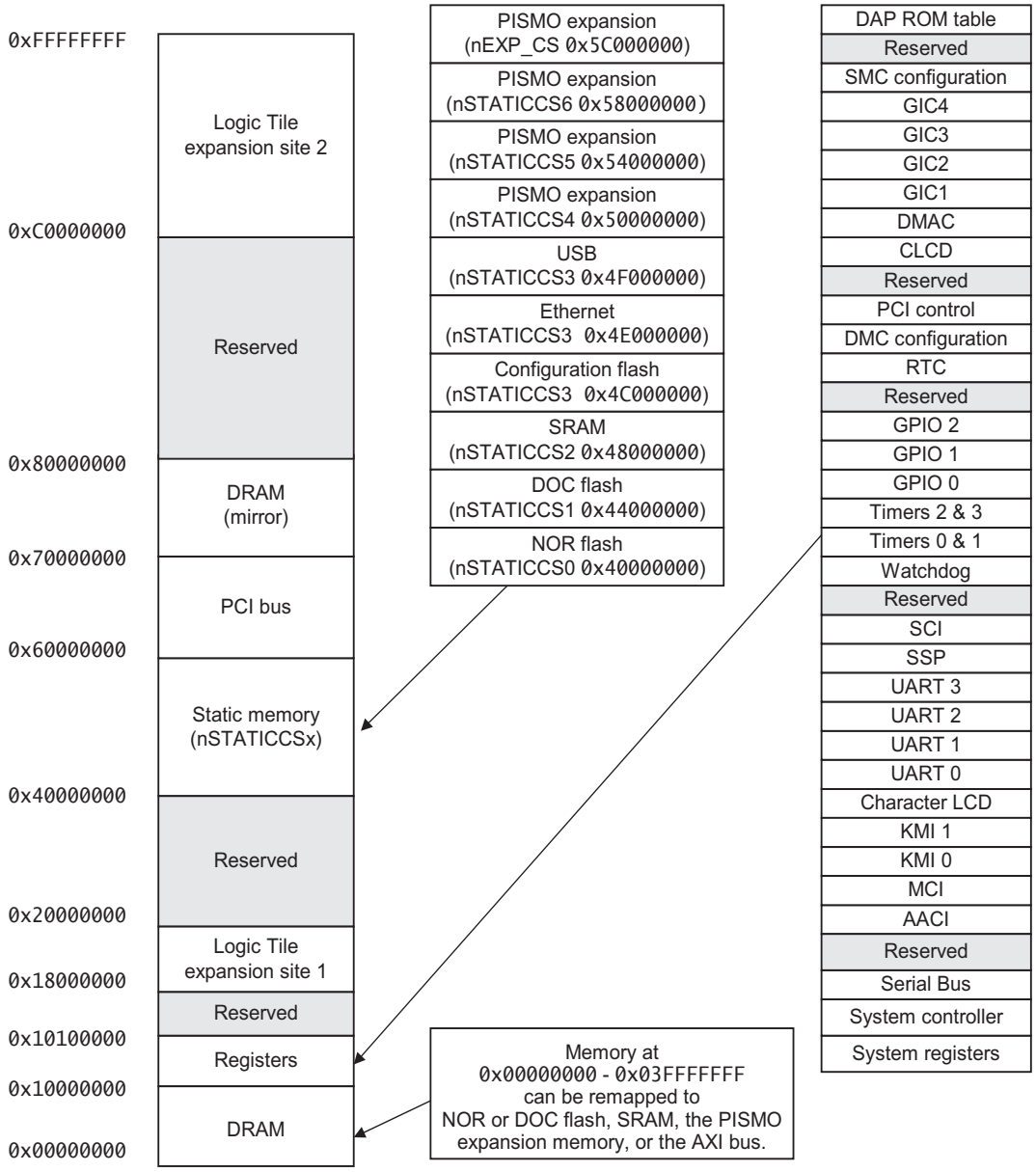

Figure 4-1 System memory map for standard peripherals

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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 4-10 for details on configuration operations that can be performed after the system has started.

4.2.1 Remapping of boot memory

On reset, the processor in the Core Tile begins executing code at address `0x0`. This address is normally volatile DRAM. Remapping enables non-volatile static memory to be decoded for accesses to low memory. Remapping of non-volatile memory to the boot region at `0x00000000`–`0x03FFFFFF` is done by the boot select switches (if the REMAP signal inside the FPGA is HIGH) as listed in Table 4-3.

Table 4-3 Boot memory

Switch S8[4:1]	Chip select	Memory Range	Comment
0000	**nNOR_CS** (**nSTATICCS0**)	`0x40000000`–`0x4FEFFFFF`	NOR flash remapped to `0x0`.
0001	**nNOR_CS** (**nSTATICCS1**)	`0x44000000`–`0x47FFFFFF`	NOR flash remapped to `0x0`.
0010	**nSRAM_CS** (**nSTATICCS2**)	`0x48000000`–`0x4BFFFFFF`	SRAM remapped to `0x0`. The SRAM contents are volatile and are not valid after a power cycle. The contents are valid, however, after the reset switch is pressed.
0011	Reserved (**nSTATICCS3**)	`0x4C000000`–`0x4FFFFFFF`	Reserved
0100	**nSTATICCS4**	`0x50000000`–`0x53FFFFFF`	PISMO CS1 remapped to `0x0`.
0101	**nSTATICCS5**	`0x54000000`–`0x57FFFFFF`	PISMO CS2 remapped to `0x0`.
0110	**nSTATICCS6**	`0x58000000`–`0x5BFFFFFF`	PISMO CS3 remapped to `0x0`.
0111	**nSTATICCS7**	`0x5C000000`–`0x5FFFFFFF`	PISMO CS0 remapped to `0x0` (this signal is output from the FPGA as **nEXP_CS**).
1xxx	External master	`0x00000000`–`0x0FFFFFFF`	All accesses are placed on the AXI or AHB master bus. (These go to a slave fitted on tile site 2.)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Removing boot remapping and enabling DRAM at 0x0

The processor in the Core Tile begins executing at `0x0` after a reset. But because memory mapping is active at reset, the remapping logic causes boot instructions to be fetched from non-volatile static memory. The System Controller SYSCTRL register controls memory remapping. See *System Controller* on page 4-88.

Note

Refer to the code examples supplied on the CD for an example of boot source code.

4.2.2 Memory characteristics

Table 4-4 lists the controller memory banks, chip selects, and memory range. Addresses not listed are decoded by the FPGA or passed onto the external buses.

The static memory controller signal **nSTATICCS3** and the **SMADDR** signals are decoded by the configuration PLD to generate **nCFGFLASH_CS**, **ETHnDATACS**, and **USB_nCS**.

Table 4-4 Memory chip selects and address range

Bank	Signal	Address range	Device
DMC bank 4	**nDDRnCS0**	`0x00000000-0x0FFFFFFF` (`0x70000000-0x7FFFFFFF`)	DRAM (DRAM mirror)
SMC bank 0	**nNOR_CS**	`0x40000000-0x43FFFFFF`	NOR flash
SMC bank 1	**nNOR_CS**	`0x44000000-0x47FFFFFF`	NOR flash
SMC bank 2	**nSRAM_CS**	`0x48000000-0x4BFFFFFF`	SRAM
SMC bank 3 (additional decoding done by configuration PLD)	**nCFGFLASH_CS**	`0x4C000000-0x4DFFFFFF`	Configuration flash
	ETHnDATACS	`0x4E000000-0x4EFFFFFF`	Ethernet
	USB_nCS	`0x4F000000-0x4FFFFFFF`	USB
SMC bank 7	**nEXP_CS**	`0x5C000000-0x5FFFFFFF`	PISMO expansion memory (**nCS0**)
SMC bank 4	**nSTATICCS4**	`0x50000000-0x53FFFFFF`	PISMO expansion memory (**nCS1**)
SMC bank 5	**nSTATICCS5**	`0x54000000-0x57FFFFFF`	PISMO expansion memory (**nCS2**)
SMC bank 6	**nSTATICCS6**	`0x58000000-0x5BFFFFFF`	PISMO expansion memory (**nCS3**)
-	**nSTATICCS7**	-	PISMO expansion memory (this signal is tied HIGH inside the FPGA).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.3 Status and system control registers

The baseboard status and system control registers enable the processor to determine its environment and to control some on-board operations. The registers, listed in Table 4-5, are located from `0x10000000`.

The SP810 System Controller manages memory remapping at reset and some clock control functions. See *System Controller* on page 4-88. See also *Reset logic* on page 3-20 for a description of the reset logic.

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.

Table 4-5 Register map for system control registers

Name	Address	Access[a]	Description
SYS_ID	`0x10000000`	Read-only	System Identity. See *ID Register, SYS_ID* on page 4-13.
SYS_SW	`0x10000004`	Read-only	Bits [7:0] map to S6 (user switches). See *Switch Register, SYS_SW* on page 4-14.
SYS_LED	`0x10000008`	Read/Write	Bits [7:0] map to user LEDs (located next to S6). See *LED Register, SYS_LED* on page 4-15.
SYS_OSC0	`0x1000000C`	Read/Write Lockable	Settings for the ICS307 programmable oscillator chip OSC0. See *Oscillator registers, SYS_OSCx* on page 4-15.
SYS_OSC1	`0x10000010`	Read/Write Lockable	Settings for the ICS307 programmable oscillator chip OSC1.
SYS_OSC2	`0x10000014`	Read/Write Lockable	Settings for the ICS307 programmable oscillator chip OSC2.
SYS_OSC3	`0x10000018`	Read/Write Lockable	Settings for the ICS307 programmable oscillator chip OSC3.
SYS_OSC4	`0x1000001C`	Read/Write Lockable	Settings for the ICS307 programmable oscillator chip OSC4.
SYS_LOCK	`0x10000020`	Read/Write	Write `0xA05F` to unlock. See *Lock Register, SYS_LOCK* on page 4-16.
SYS_100HZ	`0x10000024`	Read-only	100Hz counter. See *100Hz Counter, SYS_100HZ* on page 4-17.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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

Name	Address	Access[a]	Description
SYS_CONFIGDATA[2:1]	0x10000028-0x1000002C	-	This region is reserved for registers that are used to configure the clocks and clock dividers on the attached Core Tiles. The location and function of the registers depends on the tile fitted and the FPGA image. See *SYS_CONFIGDATAx* on page 4-17. **Note** See the application note for the specific combination of baseboard and Core Tile for details of these registers.
SYS_FLAGS	0x10000030	Read-only	General-purpose flags (reset by any reset). See *Flag registers, SYS_FLAGx and SYS_NVFLAGx* on page 4-19.
SYS_FLAGSSET	0x10000030	Write-only	Set bits in general-purpose flags.
SYS_FLAGSCLR	0x10000034	Write-only	Clear bits in general-purpose flags.
SYS_NVFLAGS	0x10000038	Read-only	General-purpose nonvolatile flags (reset only on power up).
SYS_NVFLAGSSET	0x10000038	Write-only	Set bits in general-purpose nonvolatile flags.
SYS_NVFLAGSCLR	0x1000003C	Write-only	Clear bits in general-purpose nonvolatile flags.
SYS_PCICTL	0x10000044	Read/Write	Read returns a HIGH in bit [0] if a PCI board is present in the expansion backplane. See *PCI Control Register, SYS_PCICTL* on page 4-20.
SYS_MCI	0x10000048	Read-only	This was the register for the "card present" and "write enabled" status for the MCI card on the PB926EJ-S development board and is retained for compatibility. Use GPIO 2 to read the status of these signals on the baseboard.
SYS_FLASH	0x1000004C	Read/Write	Controls write protection of flash devices. See *Flash Control Register, SYS_FLASH* on page 4-21.
SYS_CLCD	0x10000050	Read/Write	Controls LCD power and multiplexing. See *CLCD Control Register, SYS_CLCD* on page 4-22.
SYS_CLCDSER	0x10000054	Read/Write	Control interface to activate the 2.2 inch display on the LCD adaptor. See *2.2 inch LCD Control Register SYS_CLCDSER* on page 4-23.
SYS_BOOTCS	0x10000058	Read-only	Read register returns the current switch settings of switch S8. See *Boot select switch, SYS_BOOTCS* on page 4-24.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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

Name	Address	Access[a]	Description
SYS_24MHZ	0x1000005C	Read-only	32-bit counter clocked at 24MHz. See *24MHz Counter, SYS_24MHZ* on page 4-24.
SYS_MISC	0x10000060	Read-only	Miscellaneous control flags. See *Miscellaneous flags, SYS_MISC* on page 4-24.
SYS_DMAPSR0	0x10000064	Read/Write	Selection control for remapping DMA from external peripherals to DMA channel 0. See *DMA peripheral map registers, SYS_DMAPSRx* on page 4-25.
SYS_DMAPSR1	0x10000068	Read/Write	Selection control for remapping DMA from external peripherals to DMA channel 1.
SYS_DMAPSR2	0x1000006C	Read/Write	Selection control for remapping DMA from external peripherals to DMA channel 2.
SYS_IOSEL	0x10000070	Read/Write	Selects internal or tile peripheral signals for routing to the peripheral I/O pins. See *Peripheral I/O select, SYS_IOSEL* on page 4-26.
SYS_PLDCTL[2:1]	0x10000074–0x10000078	Read/Write	These registers are to configure the attached Core Tiles (See *SYS_PLDCTL[2:1]* on page 4-30.). See the application note for the specific combination of baseboard and Core Tile for details of these registers.
Reserved	0x1000007C	-	Reserved.
SYS_BUSID	0x10000080	Read-only	Responds with the AXI/AHB bus ID. This enables multiprocessor platforms to determine the primary boot processor. See *Bus ID register, SYS_BUSID* on page 4-31.
SYS_PROCID[1:0]	0x10000084–0x10000088	Read-only	Read returns a description for the Core Tile present in tile site See *Processor ID registers, SYS_PROCID[1:0]* on page 4-31.
SYS_OSCRESET[4:0]	0x1000008C–0x1000009C	Read/Write	Value to load into the SYS_OSC[4:0] registers on a manual reset. At power-on reset, the SYS_OSCRESET[4:0] registers are loaded with the same default value as used for SYS_OSC[4:0].

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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

Name	Address	Access[a]	Description
Core Tile configuration (SYS_VOLTAGE[7:0])	0x100000A0–0x100000BC	Read/Write	This region can contain registers that are used for power management and voltage monitoring of the attached Core Tiles. The location and function of the registers depends on the tile fitted and the FPGA image. See *SYS_VOLTAGE[7:0]* on page 4-33. See the application note for the specific combination of baseboard and Core Tile for details of these registers.
SYS_TEST_OSC[4:0]	0x100000C0–0x100000D0	Read-only	32-bit counter clocked from ICS307 oscillators. See *Oscillator test registers, SYS_TEST_OSCx* on page 4-34.
Reserved	0x100000D4–0x10000FFC	-	Reserved.

a. If Access is lockable, the register can only be written if SYS_LOCK is unlocked (see *Lock Register, SYS_LOCK* on page 4-16).

4.3.1 ID Register, SYS_ID

The SYS_ID register at 0x10000000 is a read-only register that identifies the board type, and revision. Figure 4-2 shows the bit assignment of the register.

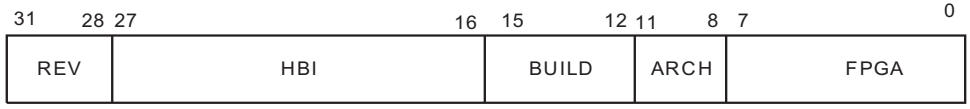

Figure 4-2 ID Register, SYS_ID

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-6 describes the baseboard ID Register assignment. The register value depends on the image loaded into the FPGA.

Table 4-6 ID Register, SYS_ID bit assignment

Bits	Access	Description
[31:28]	Read-only	Board revision: `0x0` = Rev A `0x1` = Rev B `0x2` = Rev C
[27:16]	Read-only	HBI board number (`0x140`)
[15:12]	Read-only	Build variant of board (from BOM)
[11:8]	Read-only	Bus architecture `0x4` = AHB `0x5` = AXI
[7:0]	Read-only	FPGA build

4.3.2 Switch Register, SYS_SW

Use the read-only SYS_SW register at `0x10000004` to read the general purpose (user) switch S6 and the value of the configuration (boot select) switch S8. A value of 1 indicates that the switch is on.

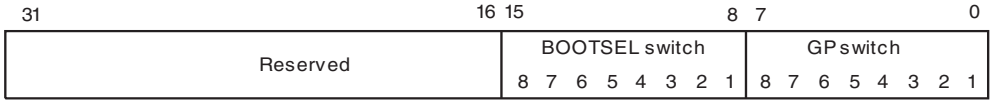

Figure 4-3 SYS_SW

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.3.3 LED Register, SYS_LED

Use the SYS_LED register at `0x10000008` to set the user LEDs. (The LEDs are located next to user switch S6.) Set the corresponding bit to 1 to illuminate the LED.

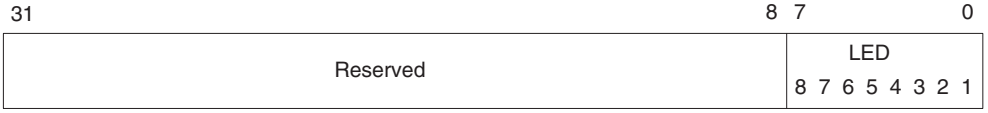

Figure 4-4 SYS_LED

4.3.4 Oscillator registers, SYS_OSCx

The oscillator registers, SYS_OSC0 to SYS_OSC4, at `0x1000000C`–`0x1000001C` are read/write registers that control the frequency of the clocks generated by the ICS307 programmable oscillators. A serial interface transfers the values in the registers to the programmable oscillators when a reset occurs.

Figure 4-5 shows the bit assignment of the registers.

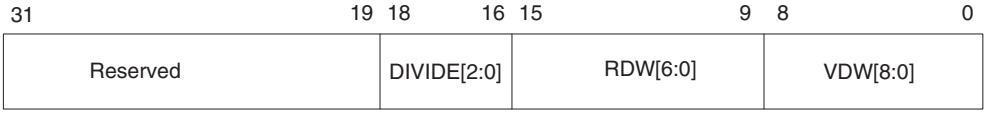

Figure 4-5 Oscillator Register, SYS_OSCx

Table 4-7 lists the details of the SYS_OSCx registers. For more detail on bit values, see *ICS307 programmable clock generators* on page 3-44 and *Clock frequency restrictions* on page B-4.

Table 4-7 Oscillator Register, SYS_OSCx bit assignment

Bits	Access	Description
[31:19]	Reserved, Use read-modify-write to preserve value.	
[18:16]	Read/Write	DIVIDE[2:0], output divider select
[15:9]	Read/Write	RDW[6:0], reference divider word
[8:0]	Read/Write	VDW[8:0], VCO divider word

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

Before writing to a SYS_OSC register, unlock it by writing the value `0x0000A05F` to the SYS_LOCK register. After writing the SYS_OSC register, 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 the oscillator registers, SYS_OSCx.

The control 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 bit assignment of the register.

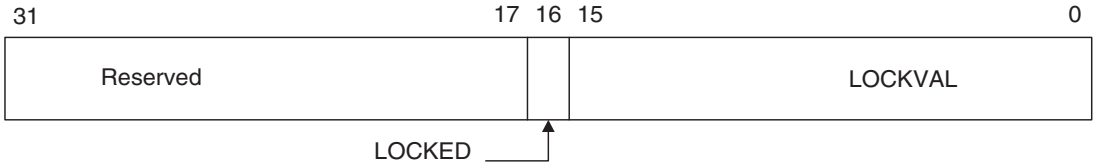

Figure 4-6 Lock Register, SYS_LOCK

Table 4-8 describes the baseboard Lock Register bit assignment.

Table 4-8 Lock Register, SYS_LOCK bit assignment

Bits	Access	Description
[31:17]	Reserved. Use read-modify-write to preserve value.	
[16]	Read-only	LOCKED, this bit indicates if the control registers are locked or unlocked: 0 = unlocked 1 = locked.
[15:0]	Read/Write	LOCKVAL, write the value `0xA05F` to unlock the control registers. Write any other value to this register to lock the registers.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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 32.768kHz crystal oscillator. The register is set to zero by a reset.

4.3.7 SYS_CONFIGDATAx

These registers (at `0x10000028` and `0x1000002C`) are used to control configuration of the system clocks.

Note

The function of these registers might be different for the combination of boards that you are using. Read the application note for your system for any changes to this register area.

Table 4-9 and Table 4-10 on page 4-18 describe the bit assignments for the configuration registers for a Core Tile CT1136JF-S.

Table 4-9 Init register 1, SYS_CONFIGDATA1 bit assignment

Bits	Access	Description
[31:30]	Reserved. Use read-modify-write to preserve value.	
[29:24]	Read-only	USERIN[5:0]
[23:17]	Reserved. Use read-modify-write to preserve value.	
[16]	Read/Write	Internal RAM enable: 0 = disabled 1 = enabled.
[15:8]	Read/Write	PLLFBDIV[7:0]
[7]	Reserved. Use read-modify-write to preserve value.	
[6:4]	Read/Write	HCLK divider (the default value depends on the setting of switch SW-8)
[3]	Reserved. Use read-modify-write to preserve value.	

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-9 Init register 1, SYS_CONFIGDATA1 bit assignment (continued)

Bits	Access	Description
[2]	Read/Write	VINITHI: 0 = vectors at `0x0` 1 = vectors at `0xFFFF0000`.
[1]	Read-only	PLLBYPASS
[2]	Read/Write	PLLBYPASS (takes effect only after reset): 0 = off 1 = on.

Table 4-10 Init register 2, SYS_CONFIGDATA2 bit assignment

Bits	Access	Description
[31]	Reserved. Use read-modify-write to preserve value.	
[30]	Read-only	ENIRW_SYNC (default is 0)
[29]	Read-only	ENPD_SYNC (default is 0)
[28]	Read-only	HCLK_SYNC (default is 0)
[27]	Read-only	IRW_SYNC (default is 0)
[26]	Read-only	PD_SYNC (default is 0)
[25:20]	Read/Write	HCLKE (default is 3, depends on setting of switch SW-8)
[19:14]	Read/Write	HCLK1 (default is 1)
[13:8]	Read/Write	CLK (default is 0)
[7:0]	Reserved. Use read-modify-write to preserve value.	

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.3.8 Flag registers, SYS_FLAGx and SYS_NVFLAGx

The registers shown in Table 4-11 provide two 32-bit register locations containing general-purpose flags. You can assign any meaning to the flags.

Table 4-11 Flag registers

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

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

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.3.9 PCI Control Register, SYS_PCICTL

The SYS_PCICTL register at `0x10000044` are used with the bridge to the PCI bus as listed in TheTable 4-12.

Table 4-12 PCI control

Bits	Access	Description
[31:4]	-	Reserved. Use read-modify-write to preserve value.
[3:2]	Read/Write	(PCICONTROLOUT) Signals to PCI block.
[1]	Read-only	(PCICONTROLIN) Signals from PCI block.
[0]	Read/Write	(PCIDETECT) Setting bit 0 HIGH enables PCI bus accesses. Read returns a HIGH in bit 0 if a PCI board is present in the expansion backplane.

Additional registers that control the initialization and mapping of PCI are described in *PCI controller* on page 4-68.

See also Appendix D *PCI Backplane and Enclosure* and *PCI interface* on page 3-67.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.3.10 MCI Register, SYS_MCI

The SYS_MCI register at `0x10000048` provides status information on the Multimedia card sockets. The function of the register bits are shown in Table 4-13.

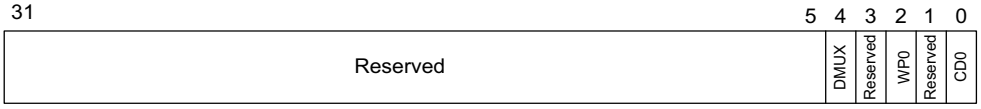

Figure 4-7 SYS_MCI

Table 4-13 MCI control

Bits	Access	Description
[31:5]	-	Reserved. Use read-modify-write to preserve value.
[4]	-	Reserved (data multiplex)
[3]	Read-only	Write protect 1
[2]	Read-only	Write protect 0
[1]	Read-only	Card detect 1
[0]	Read-only	Card detect 0

4.3.11 Flash Control Register, SYS_FLASH

Bit 0 of the SYS_FLASH register at `0x1000004C` controls write protection of static memory devices. The function of the register bits are listed in Table 4-14.

Table 4-14 Flash control

Bits	Access	Description
[31:1]	-	Reserved. Use read-modify-write to preserve value.
[0]	Read/Write	if LOW, disables writing to any static memory device that has a write-protect signal (power-on reset state is LOW)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.3.12 CLCD Control Register, SYS_CLCD

The SYS_CLCD register at `0x10000050` controls LCD power and multiplexing and controls the interface to the touchscreen as listed in Table 4-15. See also *LCD power control* on page C-7.

Note

These signals are always driven to the adapter board from the FPGA, even if the CLCD signals come from HDRZ I/O.

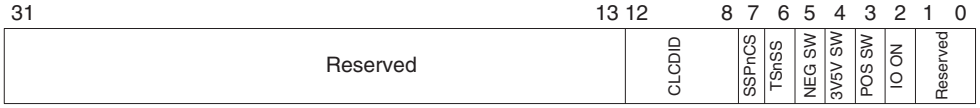

Figure 4-8 SYS_CLCD

Table 4-15 SYS_CLCD register

Bits	Access	Description
[31:14]	-	Reserved. Use read-modify-write to preserve value.
[13]	-	Reserved (touchscreen data available on PB926)

[12:8]	Read-only	CLCDID[4:0], returns the setting of the ID links on the CLCD adaptor board
		`Value Display`
		`0 320x240`
		`1 640x480`
		`2 220x176`
		`3-31 Reserved`

[7]	Read/Write	SSP expansion chip select. If HIGH, the chip select (**SSPnCS**) on the SSP expansion connector is active. SSPCS is inverted to **SSPnCS** at the FPGA pin. See *Synchronous Serial Port, SSP* on page 3-73.
[6]	Read/Write	Touchscreen enable (**TSnSS**) to controller on CLCD adaptor board
[5]	Read/Write	VDDNEGSWITCH[a], enable NEG voltage on the CLCD adaptor board
[4]	Read/Write	PWR3V5VSWITCH[a], enable FIXED voltage on the CLCD adaptor board
[3]	Read/Write	VDDPOSSWITCH[a], enable POS voltage on the CLCD adaptor board
[2]	Read/Write	LCDIOON[a], enable the RGB signal buffers on CLCD adaptor board
[1:0]	-	Reserved (LCD mode)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

a. The voltage control selection in the SYS_CLCD register might be overridden by links on the CLCD adaptor board.

4.3.13 2.2 inch LCD Control Register SYS_CLCDSER

The SYS_CLCDSER register at `0x10000054` controls the interface to the serial power-on logic in the 2.2inch display on the LCD adaptor board. See Table 4-16 and *LCD power control* on page C-7. Use this register to configure the 2.2inch display at power-on.

Figure 4-9 SYS_CLCDSER

Table 4-16 SYS_CLCDSER register

Bits	Access	Description
[31:7]	-	Reserved. Use read-modify-write to preserve value.
[6]	Read-only	Serial data in (**LCDSD0IN**)
[5]	Read/Write	Serial data out (**LCDSD0OUT**)
[4]	Read/Write	Serial data direction control (**LCDSD0OUTnIN**) 1 Data from FPGA 0 Data to FPGA
[3]	Read/Write	Device selection control (**LCDXCS**)
[2]	Read/Write	Write data control (**LCDXWR**)
[1]	Read/Write	Read data control (**LCDDXRD**)
[0]	Read/Write	Serial interface data or command select (**LCDDATnCOM**) 1 Data 0 Command

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.3.14 Boot select switch, SYS_BOOTCS

This read-only register at `0x10000058` returns the value of the boot select switch S8.

Table 4-17 SYS_BOOTCS register

Bits	Access	Description
[31:8]	-	Reserved.
[7:0]	Read-only	Switch settings.

4.3.15 24MHz Counter, SYS_24MHZ

The SYS_24MHZ register at `0x1000005C` 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.

4.3.16 Miscellaneous flags, SYS_MISC

The SYS_MISC register at `0x10000060` returns the values of miscellaneous flags related to communication. See Table 4-18.

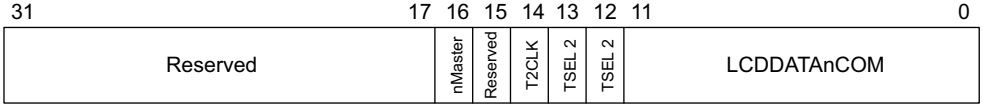

Figure 4-10 SYS_MISC register

Table 4-18 SYS_MISC register

Bits	Access	Description
[31:17]	-	Reserved. Use read-modify-write to preserve value.
[16]	Read-only	nMaster
[15]	-	Reserved. Use read-modify-write to preserve value.
[14]	Read/Write	T2_CLKSEL. Controls clock selection for CT11MPCore tiles on tile site 2. 1 Selects OSCLK2 as clock source 0 Selects HCLK as clock source (default).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-18 SYS_MISC register (continued)

Bits	Access	Description
[13]	Read-only	Tile detect 2
[12]	Read-only	Tile detect 1
[11:0]	Read/Write	Serial interface data or command select (**LCDDATnCOM**) 1 Data 0 Command

4.3.17 DMA peripheral map registers, SYS_DMAPSRx

The DMA map registers, SYS_DMAPSR0 to SYS_DMAPSR3 at `0x10000068` to `0x1000006C`, permit the mapping of DMA channels 0, 1, and 2 to three of the external peripherals.

Table 4-19 DMA map registers

Name	Address	Access	Description
SYS_DMAPSR0	`0x10000064`	Read/Write	controls mapping to DMA channel 0
SYS_DMAPSR1	`0x10000068`	Read/Write	controls mapping to DMA channel 1
SYS_DMAPSR2	`0x1000006C`	Read/Write	controls mapping to DMA channel 2

The registers are set to zero by a reset. The DMA mapping is disabled by default. Table 4-20 on page 4-26 lists the bit assignments. See *Direct Memory Access Controller* on page 4-48 for more information on the DMA logic.

Figure 4-11 DMA mapping register

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-20 SYS_DMAPx, DMA mapping register format

Bit	Access	Description
[31:8]	-	Reserved. Use read-modify-write to preserve value.
[7]	Read/Write	Set to 1 to enable mapping of external peripheral DMA signals to the DMA controller channel.
[6:5]	-	Reserved. Use read-modify-write to preserve value.

[4:0]	Read/Write	FPGA peripheral mapped to this channel
		`b00000 = USB A`
		`b00001 = USB B`
		`b00010 = UART3 TX`
		`b00011 = UART3 RX`
		`b00100 = Tile site 1 DMA0`
		`b00101 = Tile site 1 DMA1`
		`b00110 = Tile site 1 DMA2`
		`b00111 = Tile site 2 DMA0`
		`b01000 = Tile site 2 DMA1`
		`b01001 = Tile site 2 DMA2`
		`b01010-b11111 Reserved`

4.3.18 Peripheral I/O select, SYS_IOSEL

Table 4-21 on page 4-27 lists the I/O devices that can be routed to either the FPGA, tile site1, or tile site 2. The SYS_IOSEL register (at `0x10000070`) controls the state of the switches. A serial interface in the FPGA transmits the control signals to the routing PLD. The routing PLD outputs the individual control signals, **nENT1_1** for example, to the electronic switches.

See Table 4-22 on page 4-28 for more detail on the mapping between switch state and the values of the control bits.

Note

The baseboard FPGA monitors these signals and if either tile site is selected to drive these I/O signals it will tri-state its drivers for those I/O signals.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

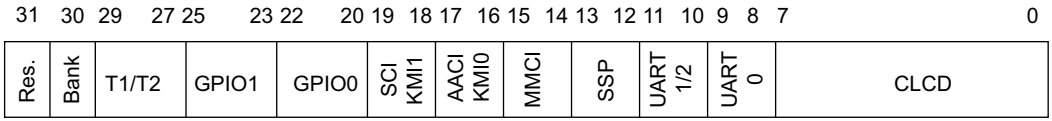

Figure 4-12 SYS_IOSEL

Table 4-21 Peripheral select signals

I/O device	Bits	Access	Reset	Note
Reserved	[31]	-	-	Reserved for future use
Bank select	[30]	Read/Write	b0	Selects between FPGA and tile as source of CLCD signals to CLCD logic and VGA connector.
T1 T2 sideband	[29:26]	Read/Write	b000	Controls sideband signals between tile site 1 and tile site 2. See *Bus signals on headers* on page 3-7.
GPIO1	[25:23]	Read/Write	b000	The GPIOs are controlled separately to enable more complex interconnection.
GPIO0	[22:20]	Read/Write	b000	
SCI and KMI1	[19:18]	Read/Write	b000	
AACI and KMI0	[17:16]	Read/Write	b000	**Note** **MCICLK** is included in this bank.
MMCI	[15:14]	Read/Write	b000	
SSP	[13:12]	Read/Write	b000	
UART1/2	[11:10]	Read/Write	b000	
UART0	[9:8]	Read/Write	b000	UART0 has more control signals than UART1/2.
CLCD	[7:0]	Read/Write	b000	Routing for CLCD signal groups.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-22 Peripheral routing signals

Control signals	SYS_IOSEL bits	Interconnected signals		Peripheral
nENT2_0	[1:0] = b01	**T2Z[7:0]**	**PERIPH[7:0]**	CLCD
nENT1_0	[1:0] = b10	**T1Z[7:0]**	**PERIPH[7:0]**	
nENT1T2_0	[1:0] = b11	**T1Z[7:0]**	**T2Z[7:0]**	
nENT2_1	[3:2] = b01	**T2Z[15:8]**	**PERIPH[15:8]**	
nENT1_1	[3:2] = b10	**T1Z[15:8]**	**PERIPH[15:8]**	
nENT1T2_1	[3:2] = b11	**T1Z[15:8]**	**T2Z[15:8]**	
nENT2_2	[5:4] = b01	**T2Z[23:16]**	**PERIPH[23:16]**	
nENT1_2	[5:4] = b10	**T1Z[23:16]**	**PERIPH[23:16]**	
nENT1T2_2	[5:4] = b11	**T1Z[23:16]**	**T2Z[23:16]**	
nENT2_3	[7:6] = b01	**T2Z[31:24]**	**PERIPH[31:24]**	
nENT1_3	[7:6] = b10	**T1Z[31:24]**	**PERIPH[31:24]**	
nENT1T2_3	[7:6] = b11	**T1Z[31:24]**	**T2Z[31:24]**	
nENT1_4	[9:8] = b10	**T1Z[39:32]**	**PERIPH[39:32]**	UART0
nENT2_4	[9:8] = b01	**T2Z[39:32]**	**PERIPH[39:32]**	
nENT1T2_4	[9:8] = b11	**T1Z[39:32]**	**T2Z[39:32]**	
nENT1_5	[11:10] = b10	**T1Z[47:40]**	**PERIPH[47:40]**	UART1 on **PERIPH[43:40]** UART2 on **PERIPH[47:44]**
nENT2_5	[11:10] = b01	**T2Z[47:40]**	**PERIPH[47:40]**	
nENT1T2_5	[11:10] = b11	**T1Z[47:40]**	**T2Z[47:40]**	
nENT1_6	[13:12] = b10	**T1Z[55:48]**	**PERIPH[55:48]**	SSP
nENT2_6	[13:12] = b01	**T2Z[55:48]**	**PERIPH[55:48]**	
nENT1T2_6	[13:12] = b11	**T1Z[55:48]**	**T2Z[55:48]**	
nENT1_7	[15:14] = b10	**T1Z[63:56]**	**PERIPH[63:56]**	MCI
nENT2_7	[15:14] = b01	**T2Z[63:56]**	**PERIPH[63:56]**	**Note** **MCICLK** is in bank 8
nENT1T2_7	[15:14] = b11	**T1Z[63:56]**	**T2Z[63:56]**	

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-22 Peripheral routing signals (continued)

Control signals	SYS_IOSEL bits	Interconnected signals		Peripheral
nENT1_8	[17:16] = b10	**T1Z[71:64]**	**PERIPH[71:64]**	**MCICLK** on **PERIPH[64]** AACI on **PERIPH[69:65]** KMI0 on **PERIPH[71:70]**
nENT2_8	[17:16] = b01	**T2Z[71:64]**	**PERIPH[71:64]**	
nENT1T2_8	[17:16] = b11	**T1Z[71:64]**	**T2Z[71:64]**	
nENT1_9	[19:18] = b10	**T1Z[79:72]**	**PERIPH[79:72]**	KMI1 on **PERIPH[73:72]** SCI on **PERIPH[79:74]**
nENT2_9	[19:18] = b01	**T2Z[79:72]**	**PERIPH[79:72]**	
nENT1T2_9	[19:18] = b11	**T1Z[79:72]**	**T2Z[79:72]**	
nENT1_10	[22:20] = b1xx	**T1Z[87:80]**	**PERIPH[87:80]**	GPIO0
nENT2_10	[22:20] = bx1x	**T2Z[87:80]**	**PERIPH[87:80]**	
nENT1T2_10	[22:20] = bxx1	**T1Z[87:80]**	**T2Z[87:80]**	
nENT1_11	[25:23] = b1xx	**T1Z[95:88]**	**PERIPH[95:88]**	GPIO1
nENT2_11	[25:23] = bx1x	**T2Z[95:88]**	**PERIPH[95:88]**	
nENT1T2_11	[25:23] = bxx1	**T1Z[95:88]**	**T2Z[95:88]**	
nENT1T2_A	[26] = b0	**T1Z[103:96]**	**T2Z[103:96]**	Sideband signals between tile site 1 and tile site 2. (There is no connection between these signals and any I/O on the board.) 0 T1Z is not connected to T2Z 1 T1Z is connected to T2Z
nENT1T2_B	[27] = b0	**T1Z[111:104]**	**T2Z[111:104]**	
nENT1T2_C	[28] = b0	**T1Z[119:112]**	**T2Z[119:112]**	
nENT1T2_D	[29] = b0	**T1Z[127:120]**	**T2Z[127:120]**	
nSWBANKCTL	[30] = b0	FPGA CLCD signals	**R[7:0]**, **G[7:0]**, **B[7:0]**, and CLCD timing and power control signals	Selects between FPGA and tile as source of CLCD signals to CLCD logic and VGA connector.
SWBANKCTL	[30] = b1	**PERIPH[31:0]** (from tile switches)		

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.3.19 SYS_PLDCTL[2:1]

These register (at 0x10000074 and 0x10000078) are used to control configuration of the PLD located on the Core Tiles.

Note

SYS_PLDCTL1 at 0x10000074 is used for tile site 1 and SYS_PLD2 at 0x10000078 is used for tile site 2.

The function of these register might be different for the combination of boards that you are using. Read the application note for your system for any changes to this register area.

Table 4-9 on page 4-17 lists the bit assignments for the PLD configuration registers.

Table 4-23 Core PLD control register, SYS_PLDCTL bit assignment

Bits	Access	Description
[31:11]	Reserved. Use read-modify-write to preserve value.	
[10]	Read-only	PGOOD signal
[9:4]	Read/Write	CLKSEL Bit 4: 0 = drive from below 1 = drive from above. Bit 6:5: b00 = CLK_NEG_UP/DN_IN b01 = X[32] b10 = GND b11 = GLOBALCLK (in). Bit 7: 0 = drive GLOBALCLK out with HCLK 1 = disable. Bit 8: 0 = CLK_NEG_UP_IN drives CLK_NEG_UP_OUT 1 = HCLK drives CLK_NEG_UP_OUT. Bit 9: 0 = CLK_NEG_DN_IN drives CLK_NEG_DN_OUT 1 = HCLK drives CLK_NEG_DN_OUT.
[3:0]	Read/Write	ZCT

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.3.20 Bus ID register, SYS_BUSID

This register at `0x10000080` enables multiprocessor systems to identify the processor bus.

AXI bus For AXI buses, the AXI port used to create the read will append its own ID bits for forwarded accesses from the matrix. This value is returned on a read.

AHB bus For AHB buses the returned value is passed from the HSEL input into the bus matrix.

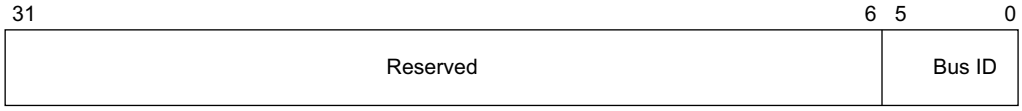

Figure 4-13 SYS_BUSID register

Table 4-24 Bus ID

Bits	Access	Description
[31:6]	-	Reserved. Use read-modify-write to preserve value.
[5:0]	Read-only	Responds with ID of the bus that the access was made with.

4.3.21 Processor ID registers, SYS_PROCID[1:0]

These registers (at `0x10000084` and `0x10000088`) returns the information about Core Tiles mounted on the tile sites. SYS_PROCID0 is for tile site 1 and SYS_PROCID1 is for tile site 2.

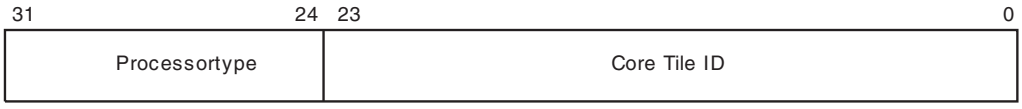

Figure 4-14 Processor ID registers

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-25 Processor ID

Bits	Access	Description

[31:24]	Read-only	Returns the processor type:
		`x00 ARM7TDMI`
		`x02 ARM9xx`
		`x04 ARM11xx`
		`x05-xFE reserved`
		`xFF Core Tile not fitted in tile site`

[23:0]	Read-only	Returns with the value in the Core Tile ID register

4.3.22 Oscillator reset registers, SYS_OSCRESETx

The oscillator reset registers, SYS_OSCRESET0 to SYS_OSCRESET4, at `0x1000008C`–`0x1000009C` are read/write registers that control the frequency of the clocks generated by clock generators OSC0, OSC1, OSC2, OSC3, and OSC4 when a manual reset is generated.

Figure 4-15 shows the bit assignment of the registers.

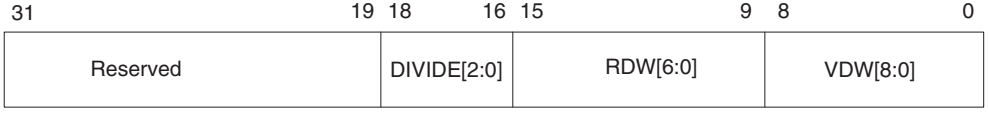

Figure 4-15 Oscillator Register, SYS_OSCRESETx

Note

Before writing to a SYS_OSCRESETx register, unlock it by writing the value `0x0000A05F` to the SYS_LOCK register (see *Lock Register, SYS_LOCK* on page 4-16). After writing the SYS_OSCRESETx register, relock it by writing any value other than `0x0000A05F` to the SYS_LOCK register.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

For more detail on bit values, see *ICS307 programmable clock generators* on page 3-44 and *Oscillator registers, SYS_OSCx* on page 4-15.

Note

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

The values of the SYS_OSCRESETx values can be changed after powering on. 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.

4.3.23 SYS_VOLTAGE[7:0]

These registers are used to read voltage and current measurements for the attached Core Tiles.

Note

The function of these registers might be different for the combination of boards that you are using. Read the application note for your system for any changes to this register area.

Table 4-26 lists the bit mapping for reading the different voltage measurements:

Table 4-26 SYS_VOLTAGE register

Register	Address	Bits [31:20] (read only)	Bits [19:8] (read-only)	Bits [7:0] (read/write)
CT_VOLTAGE_CTL0	0x100000A0	VDDCORE_DIFF1	VDDCORE1	VDDCOREA
CT_VOLTAGE_CTL1	0x100000A4	VDDCORE_DIFF2	VDDCORE2	VDDCOREB
CT_VOLTAGE_CTL2	0x100000A8	VDDCORE_DIFF3	VDDCORE3	VDDCOREC
CT_VOLTAGE_CTL3	0x100000AC	VDDCORE_DIFF4	VDDCORE4	Reserved
CT_VOLTAGE_CTL4	0x100000B0	VDDCORE_DIFF5	VDDCORE5	Reserved
CT_VOLTAGE_CTL5	0x100000B4	VDDCORE_DIFF6	VDDCORE6	Reserved
CT_VOLTAGE_CTL6	0x100000B8	VDDPLL2	VDDPLL1	Reserved
CT_VOLTAGE_CTL7	0x100000BC	VDDTP	VDDIO	Reserved

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.3.24 Oscillator test registers, SYS_TEST_OSCx

The oscillator test registers, SYS_TEST_OSC0 to SYS_TEST_OSC4, provide 32-bit count values. The count increments at frequency of the corresponding ICS307 programmable oscillator. The registers are set to zero by a reset.

Table 4-27 Oscillator test registers

Name	Address	Access	Description
SYS_TEST_OSC0	0x100000C0	Read-only	Counter clocked from clock 0
SYS_TEST_OSC1	0x100000C4	Read-only	Counter clocked from clock 1
SYS_TEST_OSC2	0x100000C8	Read-only	Counter clocked from clock 2
SYS_TEST_OSC3	0x100000CC	Read-only	Counter clocked from clock 3
SYS_TEST_OSC4	0x100000D0	Read-only	Counter clocked from clock 4

4.3.25 SYS_GPIO

This register (at 0x10000084) connects to interrupt lines [18:0]. Writing to this location can be used to test interrupt handlers.

Table 4-28 Interrupt test

Bits	Access	Description
[31:19]	-	Reserved. Use read-modify-write to preserve value.
[18:0]	Read/Write	Setting a bit triggers the corresponding interrupt line.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.4 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	FPGA (the CODEC is an external LM4549).
Memory base address	`0x10004000`
Interrupt	19
DMA	4 AACI Tx 5 AACI Rx.
Release version	ARM AACI PL041 r1p0 (256 FIFO depth in compact mode).
Reference documentation	*ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual* and *National Semiconductor LM4549 Data Sheet*. See also changed PrimeCell ID listed in Table 4-30 on page 4-36 and *Advanced Audio CODEC Interface, AACI* on page 3-47.

4.4.1 PrimeCell Modifications

The AACI PrimeCell in the baseboard has a different FIFO depth than the standard PL041. Therefore, the AACIPeriphID3 register contains the values listed in Table 4-30 on page 4-36.

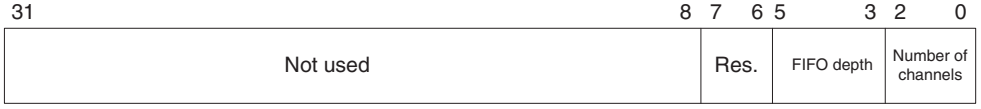

Figure 4-16 AACI ID register

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-30 Modified AACI PeriphID3 register

Bit	Access	Description
[31:8]	-	Not used
[7:6]	-	Reserved. Use read-modify-write to preserve value.
[5:3]	Read-only	FIFO depth in compact mode: b000 8 b001 16 b010 32 b011 64 b100 128 b101 256 (default) b110 512 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

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.5 Character LCD display

This is a custom peripheral that provides an interface to a standard HD44780 16 x 2 character LCD module.

Table 4-31 Character LCD display implementation

Property	Value
Location	Board (custom register interface implemented in FPGA)
Memory base address	`0x10008000-0x10008FFF` (control registers in FPGA)
Interrupt	22
DMA	NA
Release version	custom logic
Reference documentation	(see also *Character LCD controller* on page 3-50)

Note

The HD44780 display interface is very slow.

Requests to read or write data or a command to the character LCD are captured and executed later. It is not until 500ns later that the access is completed. Poll access complete flag (bit 0 of CHAR_RAW) or wait for a Char LCD interrupt on SIC7 to check that the last access has completed.

After accepting a command, the character LCD typically requires 37µs to finish processing. Some commands, `Return Home` for example, take substantially longer (20ms). Poll the busy signal to determine when the display is ready for a new data or command write.

An interrupt signal is generated by the character LCD controller a short time after the raw data is valid. However this interrupt signal is reserved for future use and you must use a polling routine instead of an interrupt service routine.

At power on, the character LCD displays status information on connected tiles (see *Tile status messages displayed on the LCD* on page 2-10).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The control and data registers for the character LCD interface are listed in Table 4-32.

Table 4-32 Character LCD control and data registers

Address	Name	Type	Description
0x10008000	CHAR_COM	Write command, read busy status	A write to this address will cause a write to the HD44780 command register some cycles later.A read from this address will cause a read from the HD44780 busy register some cycles later. **Note** The data read from this address is not valid LCD register data. Use the CHAR_RAW and CHAR_RD registers to return LCD register data.
0x10008004	CHAR_DAT	Write data RAM, read data RAM	A write to this address will cause a write to the HD44780 data register some cycles later.The first write transfers data bits [7:4] to bits [7:4] of the HD44780 data register. The second write transfers data bits [7:4] to bits [3:0] of the HD44780 data register. A read to this address will cause a read to the HD44780 data register some cycles later.The data read from this address is not valid LCD register data. Use the CHAR_RAW and CHAR_RD registers to return LCD register data.
0x10008008	CHAR_RD	Read captured data from an earlier read command	Bits [7:4] contain data from the last request read, valid only when bit 8 is set in CHAR_RAW. Bits [31:8] and [3:0] should be ignored. Two read operations are required to return the 8 bits of data in the HD44780 data register. The HD44780 returns data in bits [7:4] only. The first read returns bits [7:4] of the data register and the second read returns bits [3:0] of the data register.
0x1000800C	CHAR_RAW	Write to reset access complete flag, read to determine if data in CHAR_RD is valid	Bit 8 (CHAR_DONE) is set by the display to indicate access is complete (write 0 to clear). The bit is set if read data is valid. Bits [31:9] and [7:0] should be ignored.
0x10008010	CHAR_MASK	Write interrupt mask	Set bit 0 to 1 to enable generating an interrupt when access completes (CHAR_DONE HIGH).
0x10008014	CHAR_STAT	Read status	Bit 0 is the state of CHAR_DONE ANDed with the CHAR_MASKINT.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

An overview of the commands available is listed in Table 4-33.

Table 4-33 Character LCD display commands

Command	Bit pattern	Description
Clear display	b00000001	Clears entire display and sets display RAM address counter to zero.
Return home	b0000001x	Sets display RAM address counter to zero and returns the cursor to the first character position. Display RAM contents are not erased.
Entry mode set	b000001*DS*	Sets cursor move direction to increment (*D* HIGH) or decrement (*D* LOW). Specifies display shift (*S* HIGH). This setting affects future display RAM read or write operation.
Display on/off control	b00001*DCB*	Sets entire display on /off (*D* HIGH for on) Sets cursor on/off (*C* HIGH for on) Sets cursor position character blinking on/off (*B* HIGH for on).
Cursor or display shift	b0001*CD*xx	Moves cursor (*C* LOW) or shifts display (*C* HIGH) right (*D* HIGH) or left (*D* LOW) without changing display RAM contents.
Function set	b001*LNF*xx	Sets interface data length to 8 (*L* HIGH, the default) or 4 (*L* LOW). Sets number of display lines to two (*N* HIGH, the default) or Sets one (*N* LOW). Sets character font to 5x10 (*F* HIGH, the default) or 5x8 (*F* LOW).
Set CGRAM address	b01*AAAAAA*	Sets character generator RAM address to b*AAAAAA*. Character generator RAM data is sent and received after this setting.
Set DDRAM address	b1*AAAAAAA*	Sets display RAM address to b*AAAAAAA*. Display RAM data is sent and received after this setting.

For more details on the character display, see the example code for accessing the character LCD is provided on the CD as part of the Boot Monitor and Selftest applications. This code is copied to your hard disk during installation, see:

- *install_dir*\software\firmware\Platform\source\lcd_dbg.c
- *install_dir*\software\projects\selftest\apcharlcd\apcharldc.c.

Note

The interface bus to the character LCD is eight-bits wide, but only four bits are used. An eight-bit write access to the controller requires is done as two four-bit accesses. This is done transparently to the software.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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 4-34 CLCDC implementation

Property	Value
Location	FPGA
Memory base address	`0x10020000` **Note** There are also LCD power control registers at `0x10000050` and `0x10000054`. See *CLCD Control Register, SYS_CLCD* on page 4-22 and *2.2 inch LCD Control Register SYS_CLCDSER* on page 4-23.
Interrupt	23 for CLCD display (from adaptor board) 30 for touch screen pen interrupt (from adaptor board) 31 for touch screen pen interrupt (from adaptor board)
DMA	NA
Release version	ARM CLCDC PL111 (version r0p0)
Reference documentation	*ARM PrimeCell Color LCD Controller (PL111) Technical Reference Manual* See also *Display resolutions and display memory organization* on page 4-41, and *CLCDC interface* on page 3-51)

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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.6.1 Display resolutions and display memory organization

Different display resolutions require different data and synchronization timing. Use registers CLCD_TIM0, CLCD_TIM1, CLCD_TIM2, and SYS_OSCCLK4 to define the display timings. Table 4-35 lists the register and clock values for different display resolutions.

Table 4-35 Values for different display resolutions

Display resolution	CLCDCLK frequency and SYS_OSCCLK4 register value	CLCD_TIM0 register at 0x10020000	CLCD_TIM1 register 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
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.

Note

Rx, Gx, and Bx in Table 4-36 on page 4-42 lists the memory bits used to set the red, green, and blue brightness for direct (non-palettized) 24 and 16-bit color modes.

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. For details on using the palette RAM, see the *CLCD Technical Reference Manual*.

Memory encoding for one to eight bits per pixel are listed in Table 4-38 on page 4-45. The bit correspondence in the table is based on little-endian byte and little-endian pixel encoding.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-36 Assignment of display memory to R[7:0], G[7:0], and B[7:0]

Memory bit	8/8/8	1/5/5/5	5/6/5	4/4/4
31	unused	pixel1 I (intensity)	pixel1 R4 (msb)	unused
30	unused	pixel1 B4 (msb)	pixel1 R3	unused
29	unused	pixel1 B3	pixel1 R2	unused
28	unused	pixel1 B2	pixel1 R1	unused
27	unused	pixel1 B1	pixel1 R0 (lsb)	pixel1 B3
26	unused	pixel1 B0 (lsb)	pixel1 G5 (msb)	pixel1 B2
25	unused	pixel1 G4 (msb)	pixel1 G4	pixel1 B1
24	unused	pixel1 G3	pixel1 G3	pixel1 B0 (lsb)
23	B7 (msb)	pixel1 G2	pixel1 G2	pixel1 G3
22	B6	pixel1 G1	pixel1 G1	pixel1 G2
21	B5	pixel1 G0 (lsb)	pixel1 G0 (lsb)	pixel1 G1
20	B4	pixel1 R4 (msb)	pixel1 B4 (msb)	pixel1 G0 (lsb)
19	B3	pixel1 R3	pixel1 B3	pixel1 R3
18	B2	pixel1 R2	pixel1 B2	pixel1 R2
17	B1	pixel1 R1	pixel1 B1	pixel1 R1
16	B0 (lsb)	pixel1 R0 (lsb)	pixel1 B0	pixel1 R0 (lsb)
15	G7 (msb)	pixel0 I (intensity)	pixel0 B4 (msb)	unused
14	G6	pixel0 B4	pixel0 B2	unused
13	G5	pixel0 B3	pixel0 B2	unused
12	G4	pixel0 B2	pixel0 B1	unused
11	G3	pixel0 B1	pixel0 B0 (lsb)	pixel0 B3
10	G2	pixel0 B0 (lsb)	pixel0 G5 (msb)	pixel0 B2
9	G1	pixel0 G4 (msb)	pixel0 G4	pixel0 B1

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-36 Assignment of display memory to R[7:0], G[7:0], and B[7:0] (continued)

Memory bit	8/8/8	1/5/5/5	5/6/5	4/4/4
8	G0 (lsb)	pixel0 G3	pixel0 G3	pixel0 B0 (lsb)
7	R7 (msb)	pixel0 G2	pixel0 G2	pixel0 G3
6	R6	pixel0 G1	pixel0 G1	pixel0 G2
5	R5	pixel0 G0 (lsb)	pixel0 G0 (lsb)	pixel0 G1
4	R4	pixel0 R4 (msb)	pixel0 R4 (msb)	pixel0 G0 (lsb)
3	R3	pixel0 R3	pixel0 R3	pixel0 R3
2	R2	pixel0 R2	pixel0 R2	pixel0 R2
1	R1	pixel0 R1	pixel0 R2	pixel0 R1
0	R0 (lsb)	pixel0 I (intensity)	pixel0 R0 (lsb)	pixel0 R0 (lsb)

Table 4-37 Assignment of display memory for 8, 4, 2, and 1 bits per pixel

Memory bit	8 bits per pixel	4 bits per pixel	2 bits per pixel	1 bit per pixel
31	pixel 3 X7	pixel 7 X3	pixel 15 X1	pixel 31
30	pixel 3 X6	pixel 7 X2	pixel 15 X0	pixel 30
29	pixel 3 X5	pixel 7 X1	pixel 14 X1	pixel 29
28	pixel 3 X4	pixel 7 X0	pixel 14 X0	pixel 28
27	pixel 3 X3	pixel 6 X3	pixel 13 X1	pixel 27
26	pixel 3 X2	pixel 6 X2	pixel 13 X0	pixel 26
25	pixel 3 X1	pixel 6 X1	pixel 12 X1	pixel 25
24	pixel 3 X0	pixel 6 X0	pixel 12 X0	pixel 24
23	pixel 2 X7	pixel 5 X3	pixel 11 X1	pixel 23
22	pixel 2 X6	pixel 5 X2	pixel 11 X0	pixel 22
21	pixel 2 X5	pixel 5 X1	pixel 10 X1	pixel 21

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-37 Assignment of display memory for 8, 4, 2, and 1 bits per pixel (continued)

Memory bit	8 bits per pixel	4 bits per pixel	2 bits per pixel	1 bit per pixel
20	pixel 2 X4	pixel 5 X0	pixel 10 X0	pixel 20
19	pixel 2 X3	pixel 4 X3	pixel 9 X1	pixel 19
18	pixel 2 X2	pixel 4 X2	pixel 9 X0	pixel 18
17	pixel 2 X1	pixel 4 X1	pixel 8 X1	pixel 17
16	pixel 2 X0	pixel 4 X0	pixel 8 X0	pixel 16
15	pixel 1 X7	pixel 3 X3	pixel 7 X1	pixel 15
14	pixel 1 X6	pixel 3 X2	pixel 7 X0	pixel 14
13	pixel 1 X5	pixel 3 X1	pixel 6 X1	pixel 13
12	pixel 1 X4	pixel 3 X0	pixel 6 X0	pixel 12
11	pixel 1 X3	pixel 2 X3	pixel 5 X1	pixel 11
10	pixel 1 X2	pixel 2 X2	pixel 5 X0	pixel 10
9	pixel 1 X1	pixel 2 X1	pixel 4 X1	pixel 9
8	pixel 1 X0	pixel 2 X0	pixel 4 X0	pixel 8
7	pixel 0 X7	pixel 1 X3	pixel 3 X1	pixel 7
6	pixel 0 X6	pixel 1 X2	pixel 3 X0	pixel 6
5	pixel 0 X5	pixel 1 X1	pixel 2 X1	pixel 5
4	pixel 0 X4	pixel 1 X0	pixel 2 X0	pixel 4
3	pixel 0 X3	pixel 0 X3	pixel 1 X1	pixel 3
2	pixel 0 X2	pixel 0 X2	pixel 1 X0	pixel 2
1	pixel 0 X1	pixel 0 X1	pixel 0 X1	pixel 1
0	pixel 0 X0	pixel 0 X0	pixel 0 X0	pixel 0

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-38 Assignment of display memory to pixels

Memory bit	8 bits per pixel	4 bits per pixel	2 bits per pixel	1 bit per pixel
31	pixel 3 X7	pixel 7 X3	pixel 15 X1	pixel 31
30	pixel 3 X6	pixel 7 X2	pixel 15 X0	pixel 30
29	pixel 3 X5	pixel 7 X1	pixel 14 X1	pixel 29
28	pixel 3 X4	pixel 7 X0	pixel 14 X0	pixel 28
27	pixel 3 X3	pixel 6 X3	pixel 13 X1	pixel 27
26	pixel 3 X2	pixel 6 X2	pixel 13 X0	pixel 26
25	pixel 3 X1	pixel 6 X1	pixel 12 X1	pixel 25
24	pixel 3 X0	pixel 6 X0	pixel 12 X0	pixel 24
23	pixel 2 X7	pixel 5 X3	pixel 11 X1	pixel 23
22	pixel 2 X6	pixel 5 X2	pixel 11 X0	pixel 22
21	pixel 2 X5	pixel 5 X1	pixel 10 X1	pixel 21
20	pixel 2 X4	pixel 5 X0	pixel 10 X0	pixel 20
19	pixel 2 X3	pixel 4 X3	pixel 9 X1	pixel 19
18	pixel 2 X2	pixel 4 X2	pixel 9 X0	pixel 18
17	pixel 2 X1	pixel 4 X1	pixel 8 X1	pixel 17
16	pixel 2 X0	pixel 4 X0	pixel 8 X0	pixel 16
15	pixel 1 X7	pixel 3 X3	pixel 7 X1	pixel 15
14	pixel 1 X6	pixel 3 X2	pixel 7 X0	pixel 14
13	pixel 1 X5	pixel 3 X1	pixel 6 X1	pixel 13
12	pixel 1 X4	pixel 3 X0	pixel 6 X0	pixel 12
11	pixel 1 X3	pixel 2 X3	pixel 5 X1	pixel 11
10	pixel 1 X2	pixel 2 X2	pixel 5 X0	pixel 10
9	pixel 1 X1	pixel 2 X1	pixel 4 X1	pixel 9
8	pixel 1 X0	pixel 2 X0	pixel 4 X0	pixel 8

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-38 Assignment of display memory to pixels (continued)

Memory bit	8 bits per pixel	4 bits per pixel	2 bits per pixel	1 bit per pixel
7	pixel 0 X7	pixel 1 X3	pixel 3 X1	pixel 7
6	pixel 0 X6	pixel 1 X2	pixel 3 X0	pixel 6
5	pixel 0 X5	pixel 1 X1	pixel 2 X1	pixel 5
4	pixel 0 X4	pixel 1 X0	pixel 2 X0	pixel 4
3	pixel 0 X3	pixel 0 X3	pixel 1 X1	pixel 3
2	pixel 0 X2	pixel 0 X2	pixel 1 X0	pixel 2
1	pixel 0 X1	pixel 0 X1	pixel 0 X1	pixel 1
0	pixel 0 X0	pixel 0 X0	pixel 0 X0	pixel 0

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.7 Debug Access Port ROM table

The *Debug Access Port* (DAP) provides an internal ROM table connected to the master Debug APB port of the APB Mux. The DAP provides multiple master driving ports, all accessible and controlled through a single external interface port to provide system wide debug.

The ROM table stores the locations of the components on the Debug APB. The ROM table is a read-only device, writes are ignored.

Note

The DAP is not present in all FPGA builds. Refer to the application note for your configuration to determine if the DAP (and therefore the DAP ROM table) are used in your system.

The DAP table might not be implemented in a physical ROM. In some applications it might be implemented in a reserved area of RAM.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.8 Direct Memory Access Controller

The PL081 PrimeCell *Direct Memory Access Controller* (DMAC) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-39 DMAC implementation

Property	Value
Location	FPGA **Note** Some builds do not include the DMA controller. Refer to the Application Note for your product configuration for more details.
Memory base address	`0x10030000` for DMAC `0x10000064` for DMA mapping register SYS_DMAPSR0 `0x10000068` for DMA mapping register SYS_DMAPSR1 `0x1000006C` for DMA mapping register SYS_DMAPSR2
Interrupt	24
DMA	NA
Release version	ARM DMAC PL081 r1p2
Reference documentation	*ARM PrimeCell DMA (PL081) Technical Reference Manual* (see also *DMA* on page 3-55)

Sixteen peripheral DMA interfaces are provided by the PrimeCell DMAC, of which twelve are used by the FPGA peripherals (UART0–3, SCI, and SSP) and three are made available for devices in the Logic Tiles.

Note

The DMA controller cannot typically access the Tightly Coupled Memory in the processor on the Core Tile.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The DMA controller is designed to work in two modes:

DMAC flow control

The DMAC is programmed with the amount of data is to be transferred and requires only request signals from the peripheral to show that its buffer is ready for access.

In DMAC flow controller mode each channel requires three signals **DMASREQ**, **DMABREQ** and **DMACLR**.

Peripheral flow control

The DMAC does not know how much data is to be transferred and relies on the peripheral to tell it when the last burst or transfer is to be done.

If the peripheral is the flow controller then five signals are required **DMASREQ**, **DMABREQ**, **DMACLR**, **DMALSREQ** and **DMALBREQ**.

Table 4-40 lists the DMA channel allocation.

Table 4-40 DMA channel allocation

Peripheral	Channel	Signals	Source	Note
UART0 Tx	15	5	System FPGA	DMAC flow control only
UART0 Rx	14	5	System FPGA	DMAC flow control only
UART1 Tx	13	3	System FPGA	DMAC flow control only
UART1 Rx	12	3	System FPGA	DMAC flow control only
UART2 Tx	11	3	System FPGA	DMAC flow control only
UART2 Rx	10	3	System FPGA	DMAC flow control only
SSP0 Tx	9	3	System FPGA	DMAC flow control only
SSP0 Rx	8	3	System FPGA	DMAC flow control only
SCI Tx	7	3	System FPGA	DMAC flow control only
SCI Rx	6	3	System FPGA	DMAC flow control only
AACITx	5	3	System FPGA	DMAC flow control only
AACIRx	4	3	System FPGA	DMAC flow control only
MMCI	3	3	System FPGA	DMAC flow control only

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-40 DMA channel allocation (continued)

Peripheral	Channel	Signals	Source Note
User Defined 2	2	5	Tile Sites ,selectable between different sources **Note** The three DMA channels 0, 1, and 2 are connected to the FPGA, but there might be more than three tile peripherals that can use DMA. Three DMA mapping registers control the device that has access to the channels. Table 4-19 on page 4-25 lists the register format and possible values.
User Defined 1	1	5	
User Defined 0	0	5	

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.9 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 4-41 lists the DMC implementation.

Table 4-41 DMC implementation

Property	Value
Location	FPGA
Memory base address	`0x10018000`
Interrupt	NA
DMA	NA
Release version	See the relevant application note of your implementation and read the identification registers in your TRM.
Reference documentation	*ARM PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual* See also *Memory aliasing at reset* on page 3-26.

The DMC controls the dynamic memory on the baseboard.

For information on default values for the memory controllers, see *Memory characteristics* on page 4-9. Sample programs that configure and use dynamic memory can be found on the CD that accompanies the baseboard.

4.9.1 Register values

Figure 4-17 on page 4-52 shows a register overview. Table 4-42 on page 4-52 lists the DMC registers. The **Typical value** column describes values loaded by a version of Boot Monitor. For detailed register descriptions, see the *PL340 Technical Reference Manual*.

Caution

Refer to the application note for your baseboard and Core Tile combination for values for your system. The application note package contains a Boot Monitor that initializes the DDR registers after power up.

The `platform.a` library contains memory setup routines. See *Building an application with the platform library* on page 2-46.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

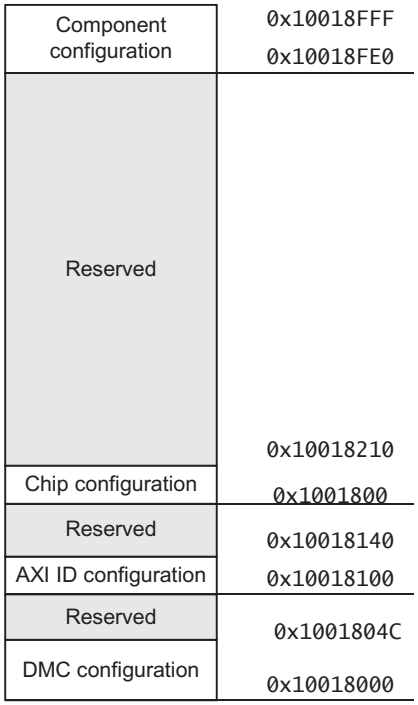

Figure 4-17 DMC register overview

Table 4-42 DMC register summary

Name	Address	Type	Typical value	Function
memc_status	0x10018000	Read-only	-	Return DMC status
memc_cmd	0x10018004	Write-only	-	Set DMC configuration
direct_cmd	0x10018008	Write-only	-	Pass commands to external memory
memory_cfg	0x1001800C	Read/Write	0x00010020	Set memory configuration
refresh_prd	0x10018010	Read/Write	0x00000200	Set refresh period
cas_latency	0x10018014	Read/Write	0x00000006	Set CAS latency
t_dqss	0x10018018	Read/Write	0x00000001	Set duration for write to DQS

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-42 DMC register summary (continued)

Name	Address	Type	Typical value	Function
t_mrd	0x1001801C	Read/Write	0x00000002	Set the mode register in the defined number of memory cycles
t_ras	0x10018020	Read/Write	0x00000003	Set RAS precharge delay
t_rc	0x10018024	Read/Write	0x00000004	Set the active bank delay
t_rcd	0x10018028	Read/Write	0x00000007	Set RAS to CAS minimum delay
t_rfc	0x1001802C	Read/Write	0x000001F2	Set autorefresh duration
t_rp	0x10018030	Read/Write	0x00000015	Set precharge to RAS delay
t_rrd	0x10018034	Read/Write	0x00000002	Set the active bank delay
t_wr	0x10018038	Read/Write	0x00000003	Set write to precharge delay
t_wtr	0x1001803C	Read/Write	0x00000002	Set write to read delay
t_xp	0x10018040	Read/Write	0x00000001	Set duration for exit power down command
t_xsr	0x10018044	Read/Write	0x0000000A	Set duration for exit self refresh
t_esr	0x10018048	Read/Write	0x00000014	Set duration for self refresh command
config	0x10018100	Read/Write	0x00000000	Set quality of service
chip_cfg0	0x10018200	Read/Write	0x0000FF00	Set external memory configuration
reserved	0x10018158-0x10018FDC	-	-	-
periph_id_n	0x10018FE0-0x10018FEC	Read-only	-	Contains peripheral ID
comp_id_n	0x10018FF0-0x10018FFC	Read-only	-	Contains peripheral version

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.10 Ethernet

The Ethernet interface is implemented in an external SMC LAN91C111 10/100 Ethernet single-chip MAC and PHY. The internal registers of the LAN91C111 are memory-mapped onto the static memory bus and occupy 16 word locations at `0x4E000000`.

Table 4-43 Ethernet implementation

Property	Value
Location	Board (LAN91C111 chip)
Memory base address	`0x4E000000` (mapped onto the SMC bus)
Interrupt	28
DMA	None, use memory to memory DMA to access the buffer memory. The master interface located in the LAN91C11 is not supported.
Release version	The FPGA contains a custom interface to the LAN91C111 chip
Reference documentation	*LAN91C111 Data Sheet* (see also *Ethernet interface* on page 3-57).

To access the PHY MII registers, you must implement a synchronous serial connection in software to control the management register in Bank 3. By default, the PHY is set to isolate in the control register. This disables the external interface. Refer to the LAN91C111 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. Reprogramming of the EEPROM is done through Bank 1 (general and control registers). See *About the SMSC LAN91C111* on page 3-59 for more information on programming the EEPROM.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.11 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 4-44 GPIO implementation

Property	Value
Location	FPGA
Memory base address	`0x10013000` for GPIO 0 `0x10014000` for GPIO 1 `0x10015000` for GPIO 2
Interrupt	6 for GPIO 0 7 for GPIO 1 8 for GPIO 2 (onboard I/O)
DMA	NA
Release version	ARM GPIO PL061 r1p0
Reference documentation	*ARM PrimeCell GPIO (PL061) Technical Reference Manual* (see also *GPIO interface* on page 3-60)

4.11.1 Onboard I/O control

GPIO2 is dedicated to the USB, push button, and MCI status signals as listed in Table 4-45.

Table 4-45 GPIO2 and MCI status signals

GPIO2 bit	Description
[7:5]	Reserved
[4]	USB HC suspend and wakeup control
[3]	USB suspend and wakeup control
[2]	General-Purpose push button
[1]	MCI Write-protect status
[0]	MCI Card-present status

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.12 Interrupt controllers

The *Generic Interrupt Controller* (GIC) is an AMBA compliant SoC peripheral that is provided by ARM Limited.

ARM processors have two interrupt signals:

- **FIQ** for fast, low latency interrupt handling
- **IRQ** for more general interrupts.

Because there are two tile sites that can have a processor fitted, there are four Generic Interrupt Controllers. The GICs accept interrupts from peripherals in the FPGA, external peripherals, or tiles and generates the FIQ and IRQ signals to the tile sites.

Table 4-46 Generic Interrupt Controller implementation

Property	Value
Location	FPGA
Memory base address	`0x10040000` GIC1 for tile site 1 IRQ `0x10050000` GIC2 for tile site 1 FIQ `0x10060000` GIC3 for tile site 2 IRQ `0x10070000` GIC4 for tile site 2 FIQ
Interrupt	FIQ and IRQ signals are output to tile sites
DMA	NA
Release version	Custom peripheral
Reference documentation	*Interrupt controller registers*, and *Interrupts* on page 3-61.

4.12.1 Interrupt controller registers

There are two logical banks of register used in the interrupt controllers:

CPU interface

The CPU interface is essentially a slave to each CPU which supports the Generic Interrupt Controller Architecture, and is responsible for holding the interrupt priority mask which is software programmable and for handling preempted interrupts. A pending interrupt is only accepted if its priority is higher than the priority mask and also higher than the priority of the highest priority active interrupt active on that CPU (that is, including those that have been pre-empted).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Distribution

The Interrupt Distributor centralizes all interrupt sources for handling by the Distributor before dispatching the highest priority ones to the interrupt interface appropriate CPU for Priority Masking and Pre-emption handling.

When an interrupt is signalled to the CPU, the CPU responds and the initial interrupt handler reads the number of the interrupt, and as a side-effect of this read, the priority of this interrupt is recorded within the CPU Interface, and that interrupt is now denoted as being active for that CPU within the distributor. Due to the programmability of the priority mask, it is possible for the interrupt to be of a lower priority than the priority mask at the time that the interrupt number is read – in this case, the interrupt number returned is 1023, indicating a spurious interrupt.

When an interrupt is completed by a CPU, the CPU software writes to the CPU Interface in the controller to indicate completion of the interrupt, passing the interrupt number as a final argument. This causes the interrupt to be marked as Inactive for that CPU within the distributor.

Note

The GIC design supports multiple CPUs for each controller. For the standard image shipped in the baseboard however, only one CPU is managed by each controller.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

CPU interface registers

The CPU interface registers for the are listed in Table 4-47.

Table 4-47 GIC interface registers

Address offset	Name	Access	Description
0x0000	CPUControl	Read/Write	Control register. Set bit 0 to 1 to enable interrupts to this CPU.
0x0004	Priority	Read/Write	Priority mask register. The CPU interface asserts an interrupt request to CPU if, and only if, the priority of the highest pending interrupt sent by the interrupt distributor is strictly higher than the mask set in Priority mask register.
0x0008	Point	Read/Write	Binary point register. Bits [2:0] determine which bits of the priority register are used to determine pre-emption: 0 Bits[7:1] of the priority are used to determine pre-emption 1 Bits[7:2] of the priority are used to determine pre-emption 2 Bits[7:3] of the priority are used to determine pre-emption 3 Bits[7:4] of the priority are used to determine pre-emption 4 Bits[7:5] of the priority are used to determine pre-emption 5 Bits[7:6] of the priority are used to determine pre-emption 6 Bit[7] of the priority is used to determine pre-emption 7 No pre-emption is performed, but all bits of the priority are used for prioritization
0x000C	Acknowledge	Read-only	Interrupt acknowledge register. Bits [9:0] contain the interrupt identifier.
0x0010	EndInterrupt	Write-only	End of interrupt register. This write-only register is used when the software has finished handling an interrupt. This will reset the interrupt to Inactive for this processor.
0x0014	Running	Read-only	Running priority register. This read-only register contains the priority level of the currently running interrupt on this CPU. When no interrupt is running (that is, acknowledged but reading acknowledge register but not ended by writing to EOI register), priority value read is 0xFF.
0x0018	Highest Pending	Read-only	The Highest Pending Interrupt Register contains the Interrupt ID of the Highest Pending Interrupt that has been selected by the Distributor for this CPU. If no interrupt is pending, the Interrupt ID returned is 1023, Spurious Interrupt.
0x001C–0x00CF	Reserved	-	-
0x00D0–0x00FF	CPU_IF_ID	Read-only	Identification registers

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Distribution registers

The distribution registers for the interrupt controllers are listed in Table 4-48.

Table 4-48 GIC distribution registers

Address offset	Name	Access	Description
0x1000	Control	Read/Write	Control register. Set bit 0 to 1 to enable interrupts.
0x1004	Controller type	Read-only	Identifies number of CPUs for this controller and the number of interrupt lines the controller services. This register must contain the value 0x00000002 (1 CPU and 96 interrupt lines).
0x1008–0x10FC	Reserved	-	-
0x1100	Enable set	Read/Write	Enable set register (interrupts 0–31) Bit 0 corresponds to interrupt signal 0.
0x1104	Enable set	Read/Write	Enable set register (interrupts 32–63)
0x1108	Enable set	Read/Write	Enable set register (interrupts 64–95)
0x110C–0x117C	Reserved	-	-
0x1180	Clear	Read/Write	Interrupt enable clear register (0–31)
0x1184	Clear	Read/Write	Interrupt enable clear register (32–63)
0x1188	Clear	Read/Write	Interrupt enable clear register (95–64)
0x118C–0x11FC	Reserved	-	-
0x1200	Pending set	Read/Write	Pending set register (0–31)
0x1204	Pending set	Read/Write	Pending set register (32–63)
0x1208	Pending set	Read/Write	Pending set register (95–64)
0x120C–0x127C	Reserved	-	-
0x1280	Pending clear	Read/Write	Pending clear register (0–31)
0x1284	Pending clear	Read/Write	Pending clear register (32–63)
0x1288	Pending clear	Read/Write	Pending clear register (95–64)
0x128C–0x12FC	Reserved	-	-

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-48 GIC distribution registers (continued)

Address offset	Name	Access	Description
0x1300	Active	Read-only	Active bit register (0–31). The active register is used to enable the software to know which interrupts are currently active (bit read as 1) on one or more CPUs. They are read-only registers, and writes to these registers are ignored.
0x1304	Active	Read-only	Active bit register(32–63)
0x1308	Active	Read-only	Active bit register (95–64)
0x130C–0x13FC	Reserved	-	-
0x1400–0x143C	Priority	Read/Write	Priority registers (interrupts 0–63) Eight bits are used to set the priority for each interrupt source. There are therefore four interrupt lines configured for each 32-bit word.
0x1440–0x17FC	Reserved	-	-
0x1800–0x185C	CPU Targets	Read/Write	CPU Targets (0-31) You must only access Bit[0]. At reset this value is set to 0, you must write a 1 to this bit to enable interrupts to pass to the Core Tile. This bit is automatically set to 1 if you are running boot monitor.
0x1860–0x1BFC	Reserved	-	-
0x1C00	Configuration	Read/Write	Configuration registers (0–15). Interrupt configuration registers are used to define what event denotes the assertion of the interrupt. Each interrupt uses two bits: b00 Active-high, N-N software model b01 Active-high, 1-N software model b10 Rising-edge, N-N software model b11 Rising-edge, 1-N software model
0x1C04	Configuration	Read/Write	Configuration registers (16–31)
0x1C08	Configuration	Read/Write	Configuration registers (32–47)
0x1C0C	Configuration	Read/Write	Configuration registers (48–63)
0x1C10	Configuration	Read/Write	Configuration registers (64–79)
0x1C14	Configuration	Read/Write	Configuration registers (80–95)
0x1C18–0x1EFC	Reserved	-	-

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-48 GIC distribution registers (continued)

Address offset	Name	Access	Description
0x1F00	Software	Write-only	Software interrupt register. The Software interrupt register is a write-only register that is used to trigger an interrupt to a CPU. Bits [9:0] contain the interrupt ID.
0x1FD0	GICPeriphID	Read-only	Peripheral identification registers
0x1FFC	GICPCellID	Read-only	Identification registers

4.12.2 Interrupt signals

The device interrupts are listed in Table 4-49.

Note

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

Table 4-49 Interrupt signals to controllers

Bit	Interrupt source	Description
[95:84]	Reserved	-
[83]	PCI3	Interrupts from PCI expansion bus
[82]	PCI2	
[81]	PCI1	
[80]	PCI0	

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-49 Interrupt signals to controllers (continued)

Bit	Interrupt source	Description
[79]	T2_INT7	Interrupts from tile site 2
[78]	T2_INT6	
[77]	T2_INT5	
[76]	T2_INT3	
[75]	T2_INT3	
[74]	T2_INT2	
[73]	T2_INT1	
[72]	T2_INT0	
[71]	T1_INT7	Interrupts from tile site 1
[70]	T1_INT6	
[69]	T1_INT5	
[68]	T1_INT3	
[67]	T1_INT3	
[66]	T1_INT2	
[65]	T1_INT1	
[64]	T1_INT0	
[63]	TSnKPADIRQ	Touch screen pen interrupt
[62]	TSnPENIRQ	Touch screen keypad interrupt
[61]	USB	Interrupt from USB controller IC
[60]	Ethernet	Interrupt from Ethernet controller IC
[59]	DOC or PISMO	Disk-on-Chip interrupt or PISMO interrupt from memory expansion board
[58]	–	Reserved
[57]	PWRFAIL	Power failure signal from FPGA
[56]	DMAC	DMA controller

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-49 Interrupt signals to controllers (continued)

Bit	Interrupt source	Description
[55]	CLCD	CLCD display (from adapter board)
[54]	LCD	Character LCD display
[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
[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 (various board I/O signals)
[39]	GPIO1	GPIO controller
[38]	GPIO0	GPIO controller
[37]	Timer 2 or 3	Timers
[36]	Timer 0 or 1	Timers
[35]	COMMTX	Debug communications transmit interrupt. tile site 1 to GIC1 and 2 tile site 2 to GIC 3 and 4 This interrupt indicates that the communications channel is available for the processor to pass messages to the debugger.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-49 Interrupt signals to controllers (continued)

Bit	Interrupt source	Description
[34]	COMMRX	Debug communications receive interrupt. tile site 1 to GIC 1 and 2 tile site 2 to GIC 3 and 4 This interrupt indicates to the processor that messages are available for the processor to read.
[33]	Software interrupt	Software interrupt. Enabling and disabling the software interrupt is done with the Enable Set and Enable Clear Registers. Triggering the interrupt however, is done from the Soft Interrupt Set register.
[32]	Watchdog	Watchdog timer
[31:16]	GIC	Reserved for GIC software interrupts
[15:0]	GIC	Software interrupts

4.12.3 Handling interrupts

This section describes interrupt handling and clearing in general.

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

Except for the processor communication interrupts, 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 1.
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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

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

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.13 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 4-50 KMI implementation

Property	Value
Location	FPGA
Memory base address	`0x10006000` KMI 0 (keyboard) `0x10007000` KMI 1 (mouse)
Interrupt	20 KMI 0 21 KMI 1
DMA	NA
Release version	ARM KMI PL050 r1p0
Reference documentation	*ARM PrimeCell Keyboard Mouse Controller (PL050) Technical Reference Manual* (see also *Keyboard/Mouse Interface, KMI* on page 3-63)

The keyboard and the mouse signals go to both J22 and J23. Use a splitter cable to use only one of the connectors to connect both devices.

The normal I/O pins for J22 are used for the mouse and the normally unused pins are used for the keyboard. A mouse can be plugged directly into J22.

J23 uses the normal I/O pins for the keyboard and the normally unused pins for the mouse. A keyboard can be plugged directly into J23.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.14 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 4-51 MCI implementation

Property	Value
Location	FPGA
Memory base address	`0x10005000`
Interrupt	17 for MCI A 18 for MCI B
DMA	3
Release version	ARM MCI PL180 r1p0
Reference documentation	*ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual* (see also *Multimedia Card Interface, MCI* on page 3-64)

The interrupts for the MCI card are managed by the GPIO 2 PrimeCell. See *General Purpose Input/Output, GPIO* on page 4-55.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.15 PCI controller

The PCI controller is implemented in the FPGA and controls the interface to the PCI bus.

Table 4-52 PCI controller implementation

Property	Value
Location	FPGA
Memory base address	`0x10019000` for the map and control registers `0x60000000` for PCI configuration, I/O, and memory The memory base for PCI master usage is configurable
Interrupt	48 PCI0 49 PCI1 50 PCI2 51 PCI3
DMA	None. Memory to memory transfers can be set up in the DMAC.
Release version	Custom logic (Xilinx LogiCore DO-DI-PCI-AL)
Reference documentation	*PCI v2.2 Specification* (see the PCI SIG web site at `www.pcisig.com`) and the PCI section on the Xilinx web site. See also Table 4-54 on page 4-69, *PCI interface* on page 3-67, and Appendix D *PCI Backplane and Enclosure*)

The PCI slave bridge connected to the FPGA recognizes addresses `0x60000000` to `0x6FFFFFFF` as being intended for a target within the PCI address space of the memory map, and passes accesses within this region to the PCI bus.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The windows that provide access to the PCI expansion bus are listed in Table 4-53.

Table 4-53 PCI bus memory map

Usage	Size	Address
PCI self config	16MB	0x60000000-0x60FFFFFF
PCI config	16MB	0x61000000-0x61FFFFFF
PCI I/O region	16MB	0x62000000-0x62FFFFFF **Note** I/O accesses from the baseboard are output on the PCI bus with addresses 0x00000000-0x00FFFFFF
PCI memory region 0	16MB	0x63000000-0x63FFFFFF
PCI memory region 1	64MB	0x64000000-0x67FFFFFF
PCI memory region 2	128MB	0x68000000-0x6FFFFFFF

4.15.1 Control registers

The PCI_IMAP*x*, PCI_SMAP*x*, PCI_SELFID, and PCI_FLAGS registers control the operation of the PCI bus and provide status information. The PCI_IMAP*x* and PCI_SMAP*x* registers define the address translation values for the PCI I/O, PCI configuration, and PCI memory windows. See Table 4-54.

Table 4-54 PCI controller registers

Address	Name	Access	Description
0x10019000	PCI_IMAP0	Read/Write	Translate board address to PCI address for accesses 0x63000000-0x63FFFFFF.
0x10019004	PCI_IMAP1	Read/Write	Translate board address to PCI address for accesses 0x64000000-0x67FFFFFF.
0x10019008	PCI_IMAP2	Read/Write	Translate board address to PCI address for accesses 0x68000000-0x6FFFFFFF.
0x1001900C	PCI_SELFID	Read/Write	Slot location of the baseboard.
0x10019010	PCI_FLAGS	Read/Write	Master and target abort flags.
0x10019014	PCI_SMAP0	Read/Write	Translate PCI base address region 0 to board address.
0x10019018	PCI_SMAP1	Read/Write	Translate PCI base address region 1 to board address.
0x1001901C	PCI_SMAP2	Read/Write	Translate PCI base address region 2 to board address.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

PCI_IMAPx registers

The PCI_IMAPx registers translate memory address bits for the PCI regions. (The number of register bits used in the transformation depend on the memory region size, see Table 4-55 on page 4-71). In the example shown in Figure 4-18, the PCI_IMAP2 register contains `0x80000000` and this is used for the six high bits of the PCI address bus. Bits [31:26] are used because a 64MB region is being addressed.

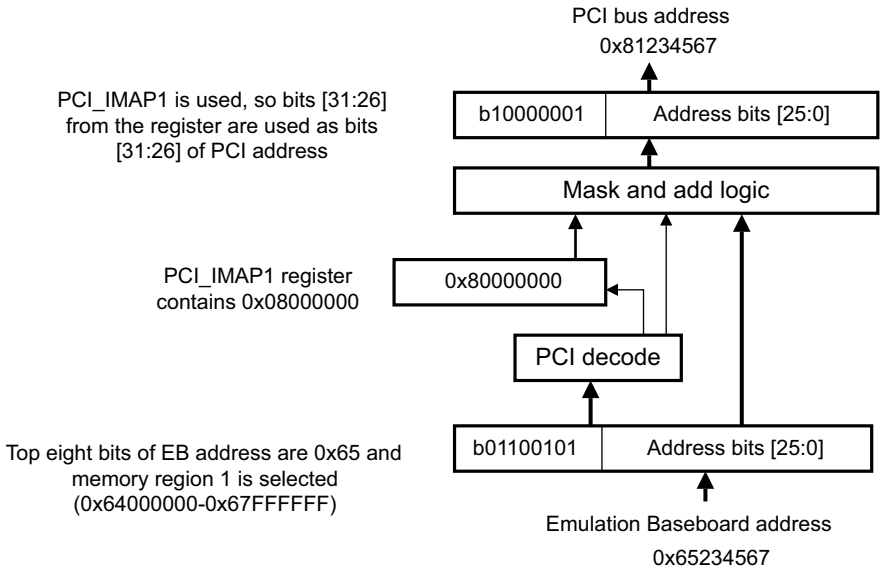

Figure 4-18 Baseboard to PCI mapping

The map register formats are shown in Figure 4-19 and Table 4-58 on page 4-74.

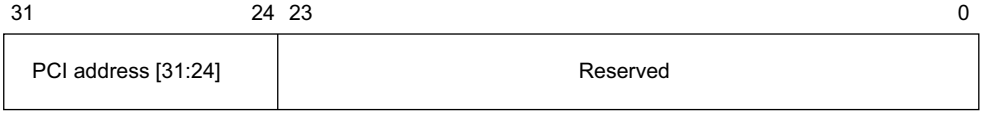

Figure 4-19 PCI_IMAPx register

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-55 PCI_IMAPx register format

Bits	Description
[31:24]	Contains the value to use for the upper bits of the PCI address for accesses to this region. **Note** Bits that are not used for remapping are ignored and the corresponding bits from the baseboard address are used to complete the PCI address: • PCI_IMAP2 uses bits [31:27] to map the 128MB baseboard region to PCI addresses • PCI_IMAP1 uses bits [31:26] to map the 64MB baseboard region to PCI addresses • PCI_IMAP0 uses bits [31:24] to map the 16MB baseboard region to PCI addresses.
[23:0]	Reserved.

PCI_SELFID register

Writing the slot location of the baseboard into this register enables normal configuration accesses to return information on the baseboard. That is, normal configuration accesses to this slot position are converted automatically into self-configuration accesses.

The register format is shown in Figure 4-20 and Table 4-56.

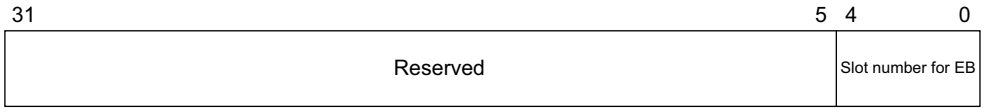

Figure 4-20 PCI_SELFID register

Table 4-56 PCI_SELFID register format

Bits	Description
[31:5]	Reserved. Use read-modify-write to preserve value.
[4:0]	Contains the slot location of the baseboard on the PCI backplane.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

PCI_FLAGS register

This read-only register returns status information about abort conditions on the PCI bus. The register format is shown in Figure 4-21 and Table 4-57.

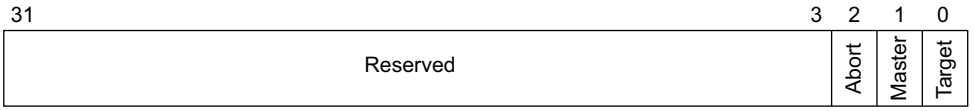

Figure 4-21 PCI_FLAGS register

Table 4-57 PCI_FLAGS register format

Bits	Description
[31:3]	Reserved.
[2]	Set this bit to enable aborts. Clear the bit to mask aborts.
[1]	Initiator abort flag. The bit value is the same as bit 39 of the Command Status Register in the Xilinx PCI controller. This bit will be HIGH if an error occurred while the baseboard was operating as a master.
[0]	Target abort flag. The bit value is the same as bit 38 of the Command Status Register in the Xilinx PCI controller. This bit position is reserved for future use.

PCI_SMAPx registers

The map registers relate to the PCI *Base Address Registers* (BAR) specified in the *PCI Local Bus Specification*.

Two remap registers provide the base addresses (bits [31:28] of the FPGA internal bus) for remapping PCI accesses to two 256MB blocks of memory.
A third remap register provides the base address (bits [31:24] of the FPGA internal bus) for remapping PCI accesses to a 16MB block of I/O.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

An example of memory remapping as shown in Figure 4-22:

1. The PCI controller has been configured to react to addresses in the range `0x40000000` to `0x4FFFFFFF` as belonging to target region 1.
2. A master on the PCI bus drives `0x41234567` onto the bus.
3. In this example, PCI_SMAP1 contains `0x80000000`. The most significant four bits of the incoming PCI address are replaced with `b1000`.
4. The modified address is output onto the EB bus. In this case, the EB address is `0x81234567`.
5. The EB processes the transaction.

Note

The SMAP registers decode both external I/O and memory requests. That is, any external access to the specified address range is converted into an AHB or AXI transfer for the mapped baseboard address.

The memory controller might process the I/0 and memory accesses differently however. For example, a memory read might use prefetched data, but an I/O read would generate a single memory read at the time the access was received.

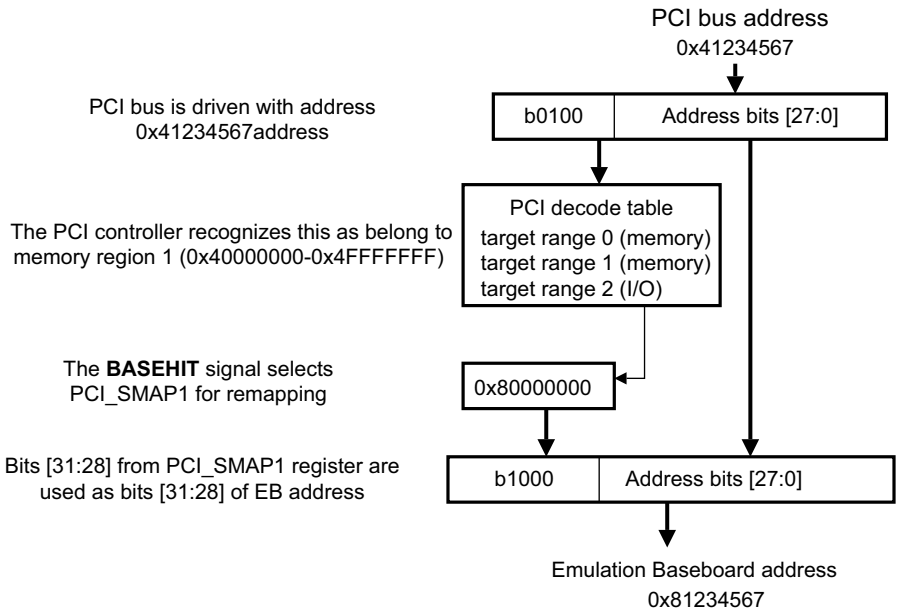

Figure 4-22 PCI to baseboard mapping

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The memory base address remap register format is shown in Figure 4-23 and Table 4-58.

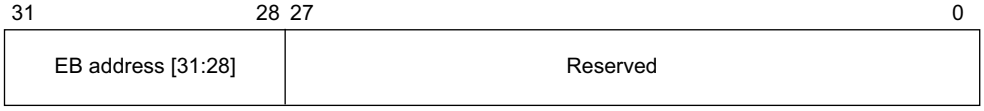

Figure 4-23 memory remap register

Table 4-58 memory remap register format

Bits	Description
[31:28]	Contains the value to use for address bits [31:28] when the PCI accesses the baseboard slave port associated with a PCI memory region.
[27:0]	Reserved.

The I/O base address remap register format is shown in Figure 4-24 and Table 4-59.

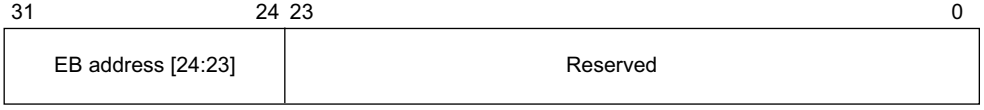

Figure 4-24 I/O remap register

Table 4-59 I/O remap register format

Bits	Description
[31:24]	Contains the value to use for address bits [31:24] when the PCI accesses the baseboard slave port associated with the PCI I/O region.
[23:0]	Reserved.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.15.2 PCI configuration

This section describes how to configure the PCI controllers on the baseboard and any PCI cards attached to the PCI backplane.

Locating the self-config header table

The slot positions for PCI cards are numbered from 11 to 31. The numbering is based on the address bit that is connected to the **IDSEL** line. The base address for the PCI configuration header is determined as follows:

```
0x60000000 + ((slot position)<<11)
```

For example, if the baseboard is put into slot C where PCI address bit 29 is connected to the **IDSEL** signal, then the base address for the baseboard header table is at memory location:

```
0x60000000 + (29<<11) = 0x6000E800
```

The self-configuration addresses for the slot A, B, and C in the PCI backplane are listed in Table 4-60.

Table 4-60 PCI backplane configuration header addresses (self-config)

Slot	Address connected to IDSEL	Configuration header memory
C	29	0x6000E800–0x6000E83F
B	30	0x6000F000–0x6000F03F
A	31	0x6000F800–0x6000F83F

The base address for normal configuration is `0x61000000`. The normal configuration addresses for the slot A, B, and C in the PCI backplane are listed in Table 4-60.

Table 4-61 PCI backplane configuration header addresses (normal configuration)

Slot	Address connected to IDSEL	Configuration header memory
C	29	0x6100E800–0x6100E83F
B	30	0x6100F000–0x6100F03F
A	31	0x6100F800–0x6100F83F

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The contents of the PCI configuration header is listed in Table 4-62. The default values refer to the baseboard.

Table 4-62 PCI configuration space header

Address offset	Configuration word function	Default value
+0x00	Device ID Vendor ID	0x030010EE
+0x04	Status Command	0x02200000
+0x08	Class Code Rev ID	0x0B400000
+0x0C	BIST (Reserved in baseboard) Header Type Lat. timer Line Size (Reserved in baseboard)	0x00000000
+0x10	Base Address Register 0 (PCI memory space)	0x00000001
+0x14	Base Address Register 1 (PCI memory space)	0x00000008
+0x18	Base Address Register 2 (PCI I/O space)	0x00000008
+0x1C	Base Address Register 3 (reserved in baseboard)	-
+0x20	Base Address Register 4 (reserved in baseboard)	-
+0x24	Base Address Register 5 (reserved in baseboard)	-
+0x28	Cardbus CIS Pointer (reserved in baseboard)	-
+0x2C	Subsystem ID Subsystem Vendor ID	0x00000000
+0x30	Expansion ROM Base Address (Reserved in baseboard)	-
+0x34	Unused (Reserved in baseboard) CapPtr	0x00000000
+0x38	(Reserved in baseboard)	-
+0x3C	Max_Lat Min_Gnt Interrupt Pin Interrupt line	0x000001FF

The PCI backplane uses the top 3 bits of PCI address to determine whether that slot should respond to configuration cycles. When the baseboard generates PCI configuration cycles by accessing the 0x60000000 or 0x61000000 region, the only one of the PCI cards responds.

See the PCI v2.2 specification for more detail on the configuration space header.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Configuring the PCI interface

To configure a PCI card in the expansion bus, first find the memory location that maps the baseboard into the system:

1. Scan addresses `0x60000000 + (n<<11)` to locate the PCI slot holding the baseboard. The slot range for *n* is 11 to 31. If you are using the horizontal slot on the PCI expansion backplane, *n* is 29.

2. Write the value of *n* that indicates the slot position into the PCI_SELFID register.

3. Set bit 2 of the Command/Status Register (at offset `+0x04`) to enable the baseboard to be initiator on the system. This enables initiator transfers.

4. Because the PCI_SELFID register now holds the slot number for the baseboard, scanning the normal configuration space at `0x61000000` reveals all PCI cards in the backplane.

   Perform normal configuration cycles on other slot positions to see what else is on the bus. Instead of the self config area at `0x61000000`, use memory locations in Config area `0x61000000 + (n<<11)`, where *n* is 11 to 31. That is, scan:

   `0x61005800`, `0x61006000`, `0x61006800`, and so on to `0x6100F800`.

5. The accesses return `0xFFFFFFFF` if the slot is empty, or the device and vendor id for card present. (For the baseboard, the device/vendor id is `0x030010EE`.)

   If a card is present, read the base address registers to determine how much and what type of memory is required by each of target boards found in the system.

6. Write to the base address registers in the header table to setup the PCI memory map and tell each target the PCI memory addresses they should respond to (see Table 4-62 on page 4-76).

7. Set the PCI control registers at `0x10001000` appropriately so an access to one of the three memory regions causes a PCI access to the correct PCI memory location.

Note

An example of PCI scanning and configuration is provided as an example on the CD.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Limitations of the PCI interface

The following limitations apply to the PCI interface present on the baseboard:

- The interface is 32-bit only (no 64-bit regions).
- 0-bit, 24-bit and unaligned 16-bit transfers are not supported.
- The PCI interface is usable with both 3.3V and 5V systems.
- The initiator creates only single reads and writes. This is quite inefficient and results in low performance. It does, however, simplify the logic in the FPGA and allows 66MHz performance.
- The target issues a retry response for reads until the data is ready.
- The target issues a retry response for reads or writes when the fifo is full (target has a 512 deep FIFO, initiator fifo is 16 deep)
- If another master accesses the baseboard and it responds with 'retry' or 'disconnect without data', then this access must repeated before any other master accesses to the baseboard.
- The baseboard breaks up burst transfers. It typically completes the first cycle and then responds with 'disconnect without data'. The initiator must then retry with the address that responded with the disconnect.
- Only three out of five configuration base registers are usable.
- Cardbus CIS Pointer and Expansion ROM configuration registers are not implemented.
- There is no support for BIST.
- The target will only respond to some of sixteen PCI bus commands, and initiator only creates six of the cycle types (see Table 4-63 on page 4-79).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-63 PCI bus commands supported

Command code	Name	Supported on target	Supported on initiator
b0000	Interrupt Acknowledge	Ignored	Not available
b0001	Special Cycle	Ignored	Not available
b0010	I/O read	Yes	Yes
b0011	I/O write	Yes	Yes
b0100	Reserved	Ignored	Not available
b0101	Reserved	Ignored	Not available
b0110	Memory Read	Yes	Yes
b0111	Memory Write	Yes	Yes
b1000	Reserved	Ignored	Not available
b1001	Reserved	Ignored	Not available
b1010	Configuration Read	Yes	Yes
b1011	Configuration Write	Yes	Yes
b1100	Memory Read Multiple	Yes	Not available
b1101	Dual Address Cycle	Ignored	Not available
b1110	Memory Read Line	Yes	Not available
b1111	Memory Write Invalidate	Yes	Not available

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.16 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 4-64 RTC implementation

Property	Value
Location	FPGA
Memory base address	`0x10017000`
Interrupt	10
DMA	NA
Release version	ARM RTC PL031 r1p0
Reference documentation	*ARM PrimeCell Real Time Clock Controller (PL031) Technical Reference Manual*

Note

There is also a separate Time-of-Year RTC implemented in an external DS1338 chip on the baseboard. The external RTC can be accessed by the serial bus interface (see *Two-wire serial bus interface* on page 4-81). For details on the programming interface to the Time-of-Year RTC, see the data sheet for the Maxim DS1338 integrated circuit.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.17 Two-wire serial bus interface

A two-wire serial bus interface is implemented in the FPGA. The registers shown in Table 4-66 control the serial bus and provides access to control signals on the two memory expansion boards and to the time-of-year clock.

Table 4-65 Serial bus implementation

Property	Value
Location	FPGA
Memory base address	`0x10002000`
Interrupt	NA
DMA	NA
Release version	Custom logic
Reference documentation	*Two-wire serial bus interface* on page 3-69, Appendix E *PISMO Memory Expansion Boards*, and the data sheet for the Dallas Maxim DS1338 Real Time Clock.

Table 4-66 Serial bus register

Address	Name	Access	Description
`0x10002000`	SB_CONTROL	Read	Read serial control bits: Bit [0] is **SCL** Bit [1] is **SDA**
`0x10002004`	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**

Note

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

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Software must manipulate the **SCL** and **SDA** bits directly to access the data in the three devices. The pre-defined eight-bit device addresses are listed in Table 4-67. See the `\firmware\examples` directory on the CD for example code for reading the EEPROM that is on the memory expansion board.

Table 4-67 Serial bus device addresses

Device	Write address	Read address
Dynamic expansion E2PROM	`0xA0`	`0xA1`
Static expansion E2PROM	`0xA2`	`0xA3`
Time-of-year clock	`0xD0`	`0xD1`

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.18 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 4-68 SCI implementation

Property	Value
Location	FPGA
Memory base address	`0x1000E0000`
Interrupt	16
DMA	7 SCI transmit 6 SCI receive.
Release version	ARM SCI PL131 r1p0
Reference documentation	*SCI PrimeCell PL131 Technical Reference Manual* (see also *Smart Card interface, SCI* on page 3-70)

The following key parameters are programmable:

- Smart Card clock frequency and communication baud rate
- protocol convention
- card activation and deactivation time
- check for maximum time for first character of *Answer To Reset* (ATR) reception
- check for maximum duration of ATR character stream
- check for maximum time for receipt of first character of data stream
- check for maximum time allowed between characters
- character and block guard time
- transmit and receive character retry and FIFO level
- clock start and stop time and inactive level.

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

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.19 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 4-69 SSP implementation

Property	Value
Location	FPGA
Memory base address	`0x1000D000`
Interrupt	11
DMA	9 for transmit 8 for receive
Release version	ARM SSP PL022 r1p2
Reference documentation	*ARM PrimeCell Synchronous Serial Port Controller (PL022) Technical Reference Manual* (see also *Synchronous Serial Port, SSP* on page 3-73)

The SSP functions as 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.

In both master and slave configurations, the PrimeCell SSP performs:

- parallel-to-serial conversion on data written to a transmit FIFO
- serial-to-parallel conversion and FIFO buffering of received data.

Interrupts are generated to:

- request servicing of the transmit and receive FIFO
- inform the system that a receive FIFO over-run has occurred
- inform the system that data is present in the receive FIFO.

The SSP controller can be shared with the following resources:

- If the LCD adaptor board is fitted with a touch screen, the controller interfaces to the SSP port to provide touch screen, keypad, LCD bias and analogue inputs. See the LCD adaptor board TSCI appendix for further details.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

Use the SYS_CLCD register to control the SSP chip selects. See *CLCD Control Register, SYS_CLCD* on page 4-22.

- An off board SSP device, such as an EEPROM, can be connected to expansion header. If you connect both the LCD adaptor board and the off board SSP device at the same time, ensure the correct SSP interface protocol is used when communicating with each device.
- Synthesized SSP peripherals in a Logic Tile FPGA can be connected using the tile expansion connectors.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.20 Static Memory Controller, SMC

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

Table 4-70 SSMC implementation

Property	Value
Location	FPGA
Memory base address	`0x10080000`
Interrupt	NA
DMA	NA
Release version	ARM SSMC r0p3
Reference documentation	*ARM PrimeCell Static Memory Controller (PL093) Technical Reference Manual*, *Configuration and initialization* on page 4-8

The following key parameters are programmable for each SSMC memory bank:

- external memory width, 8, 16, or 32-bit
- burst mode operation
- write protection
- external wait control enable
- external wait polarity
- write WAIT states for static RAM devices
- read WAIT states for static RAM and ROM devices
- initial burst read WAIT state for burst devices
- subsequent burst read WAIT state for burst devices
- read byte lane enable control
- bus turn-around (idle) cycles
- output enable and write enable output delays.

For information on default values for the memory controllers, see *Memory characteristics* on page 4-9.

Note

To enable write access to the NOR flash (static chip select 0), set bit 0 of SYS_FLASH to HIGH. The default at power-on reset is LOW.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.20.1 Register values

Table 4-71 lists the register values for the SSMC for typical operation the PISMO memory expansion with static memory devices and a 25MHz system clock.

Note

The platform.a library contains memory setup routines. See *Building an application with the platform library* on page 2-46.

The SSMCCR register at `0x200` is always loaded with `0x1` to select clock ratio of 1:1 with the clock always running.

Table 4-71 Register values for PISMO CS0 (SMC CS4)

Address	Name of SSMC register	Value	Description
`0x10080080`	SMBIDCYR	`0x0F`	Idle Cycle Control Register for bank
`0x10080084`	SMBWSTRDR	`0x1F`	Read Wait State Control Reg bank
`0x10080088`	SMBWSTWRR	`0x1F`	Write Wait State Control Reg Bank
`0x1008008C`	SMBWSTOENR	`0x01`	Output Enable Assertion Delay
`0x10080090`	SMBWSTWENR	`0x00`	Write Enable Assertion Delay
`0x10080094`	SMBCR	`0x00303021`	Control Register for memory bank. The default bus with is set by CFGWIDTH.
`0x10080098`	SMBSR	-	Status Register for bank (read-only)
`0x1008009C`	SMBWSTBRDR	`0x0F`	Burst Read Wait state Control Reg

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.21 System Controller

The *ARM PrimeXsys System Controller* is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-72 System controller implementation

Property	Value
Location	FPGA
Memory base address	`0x10001000`
Interrupt	NA
DMA	NA
Release version	ARM SYSCTRL SP810 r0p0-00ltd0
Reference documentation	*ARM PrimeCell System Controller (SP810) Technical Reference Manual*. **Note** There are also system control registers in the FPGA. See *Status and system control registers* on page 4-10. Bit 8 of the System controller register at `0x101E000` controls remapping of static memory devices to address `0x0`. See *Remapping of boot memory* on page 4-8.

4.21.1 PrimeCell modifications

The PrimeXsys System Controller used in the baseboard FPGA 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_CTRL register at address `0x10001000` is listed in Table 4-73 on page 4-89.

Note

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

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table 4-73 SYS_CTRL register

Bits	Function
[31:24]	Reserved. Use read-modify-write to preserve value.
[23]	Watchdog 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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.22 Timers

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

Table 4-74 Timer implementation

Property	Value
Location	FPGA
Memory base address	`0x10011000` for Timer 0 `0x10011020` for Timer 1 `0x10012000` for Timer 2 `0x10012020` for Timer 3.
Interrupt	4 for Timers 0 and 1 5 for Timers 2 and 3.
DMA	NA
Release version	ARM Dual-Timer SP804 r1p0-02ltd0
Reference documentation	*ARM PrimeCell Timer Module (SP804) Technical Reference Manual*

The features of the Dual-Timer module are:

- Two 32/16-bit down counters with free-running, periodic and one-shot modes.
- Common clock with separate clock-enables for each timer gives flexible control of the timer intervals.
- Interrupt output generation on timer count reaching zero.
- Identification registers that uniquely identify the Dual-Timer module. These can be used by software to automatically configure itself.

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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.23 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 4-75 UART implementation

Property	Value
Location	FPGA
Memory base address	`0x10009000` for UART0 `0x1000A000` for UART1 `0x1000B000` for UART2 `0x1000C000` for UART3
Interrupt	12 for UART0 13 for UART1 14 for UART2 15 for UART3
DMA	15 UART0 Tx 14 UART0 Rx 13 UART1 Tx 12 UART1 Rx 11 UART2 Tx 10 UART2 Rx **Note** You can use UART3 with DMA by selecting any two of the SYS_DMAPSRx Registers. You must configure separate channels for both Tx and Rx operation. See *DMA peripheral map registers, SYS_DMAPSRx* on page 4-25.
Release version	ARM UART PL011 r1p3
Reference documentation	*ARM PrimeCell UART (PL011) Technical Reference Manual* (see also *UART interface* on page 3-77).

The following key parameters are programmable:

- communication baud rate, integer, and fractional parts
- number of data and stop bits
- parity mode

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

- FIFO enable and FIFO trigger levels
- UART or IrDA protocol
- hardware flow control.

4.23.1 PrimeCell Modifications

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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.24 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 4-76 USB implementation

Property	Value
Location	Board (an ISP1761 chip)
Memory base address	`0x4F000000`, the registers are memory-mapped onto the SMC bus
Interrupt	29
DMA	There are two DMA channels available for the USB controller. These are selectable as 0,1, or 2. See *Direct Memory Access Controller* on page 4-48.
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 3-81 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 a full speed of 12MB/s
- programmable interrupts and DMA
- FIFO and 63KB on-chip RAM for USB.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The ISP1761 register base addresses are shown in Table 4-77.

Table 4-77 USB controller base address

Address	Description
0x4F000000	Host controller EHCI registers
0x4F000200	Peripheral controller registers
0x4F000300	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 4-55).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

4.25 Watchdog

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

Note

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

Table 4-78 Watchdog implementation

Property	Value
Location	FPGA
Memory base address	`0x10010000`
Interrupt	0
DMA	NA
Release version	ARM WDOG SP805 r1p0-02ltd0
Reference documentation	*ARM PrimeCell Watchdog Controller (SP805) Technical Reference Manual*

The following Watchdog module parameters are programmable:

- interrupt generation enable/disable
- interrupt masking
- reset signal generation enable/disable
- interrupt interval.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Appendix A
Signal Descriptions

This appendix provides a summary of signals present on the baseboard connectors. For more information on connectors, see the parts list spreadsheet in the CD `schematics` directory. This appendix contains the following sections:

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.1 Audio CODEC interface

The baseboard provides three jack connectors that enable you to connect to the microphone and auxiliary inputs, and line level output on the CODEC. Figure A-1 shows the pinouts of the sockets. Table A-1 lists the signals.

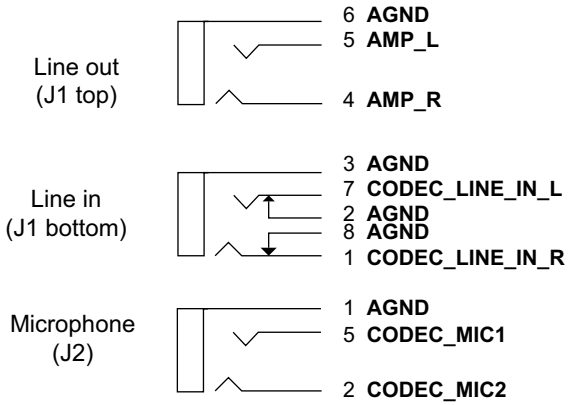

Figure A-1 Audio connectors

Table A-1 Audio connectors

Connector	Pin number	Signal
J1 bottom, line in	Sleeve	Analog ground
	Tip	Line in (left channel)
	Ring	Line in (right channel)
J1 top, line out	Sleeve	Analog ground
	Tip	Line output (left channel)
	Ring	Line output (right channel)
J2, microphone[a]	Sleeve	Analog ground
	Tip	Microphone input 1
	Ring	Microphone input 2

a. A link on the board enables bias voltage to be applied to the microphone (see *Advanced Audio CODEC Interface, AACI* on page 3-47).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.2 CLCD display interface

The CLCD interface adaptor board connector (J18) is shown in Figure A-2 on page A-5. The connectorsignals are listed in Table A-2. See Appendix C *LCD Kits* for details on the CLCD adaptor board. See *CLCDC interface* on page 3-51 for details on CLCD signals.

Table A-2 CLCD Interface board connector J4

Pin	Signal	Pin	Signal
1	**B0**	35	**B1**
2	**B2**	36	**B3**
3	**B4**	37	**B5**
4	**B6**	38	**B7**
5	**G0**	39	**G1**
6	**G2**	40	**G3**
7	**G4**	41	**G5**
8	**G6**	42	**G7**
9	**R0**	43	**R1**
10	**R2**	44	**R3**
11	**R4**	45	**R5**
12	**R6**	46	**R7**
13	**CLLE**	47	**GND**
14	**CLAC**	48	**GND**
15	**CLCP**	49	**GND**
16	**CLLP**	50	**GND**
17	**CLFP**	51	**GND**
18	**TSnKPADIRQ**	52	**GND**
19	**TSnPENIRQ**	53	**GND**
20	**TSnDAV**	54	**LCDID0**
21	**TSSCLK**	55	**LCDID1**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-2 CLCD Interface board connector J4 (continued)

Pin	Signal	Pin	Signal
22	**TSnSS**	56	**LCDID2**
23	**TSMISO**	57	**LCDID3**
24	**TSMOSI**	58	**LCDID4**
25	**LCDXWR**	59	**GND**
26	**LCDSD0**	60	**GND**
27	**LCDXRD**	61	**GND**
28	**LCDXCS**	62	**3V3**
29	**LCDDATnCOM**	63	**3V3**
30	**LCDSD0OUTnIN**	64	**5V**
31	**CLPOWER**	65	**5V**
32	**nLCDIOON**	66	**VLCD**
33	**PWR3V5VSWITCH**	67	**VLCD**
34	**VDDPOSSWITCH**	68	**VDDNEGSWITCH**

Note

The **R[7:0]**, **G[7:0]**, and **B[7:0]** signals are digital CLCD signals. The digital signals must be converted by the PLC and DAC to produce the **R**, **G**, and **B** analog signals used on the VGA connector.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

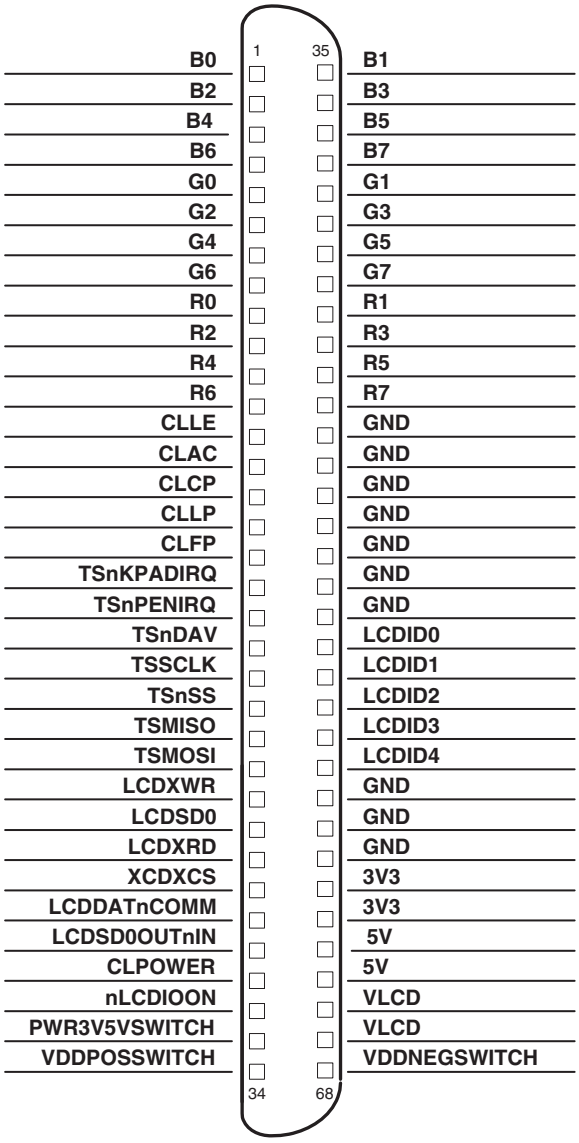

Figure A-2 CLCD Interface connector J4

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.3 Ethernet interface

The RJ45 Ethernet connector J44 is shown in Figure A-3.

LEDA (green) and LEDB (yellow) are connected to the LAN91C111 controller. The function of the LEDs is determined by registers in the controller. Typical usage would be to monitor transmit activity and packet detection.

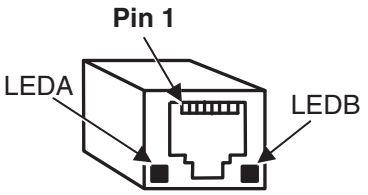

Figure A-3 Ethernet connector J44

The signals on the Ethernet cable are listed in Table A-3.

Table A-3 Ethernet signals

Pin (RJ45)	Signal
1	Transmit +
2	Transmit -
3	Receive +
4	NC
5	Transmit common
6	Receive -
7	Receive common
8	NC

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.4 GPIO interface

Two eight-bit *General Purpose Input/Output* (GPIO) controllers are incorporated into the FPGA. The signals on the GPIO connector are shown in Figure A-4.

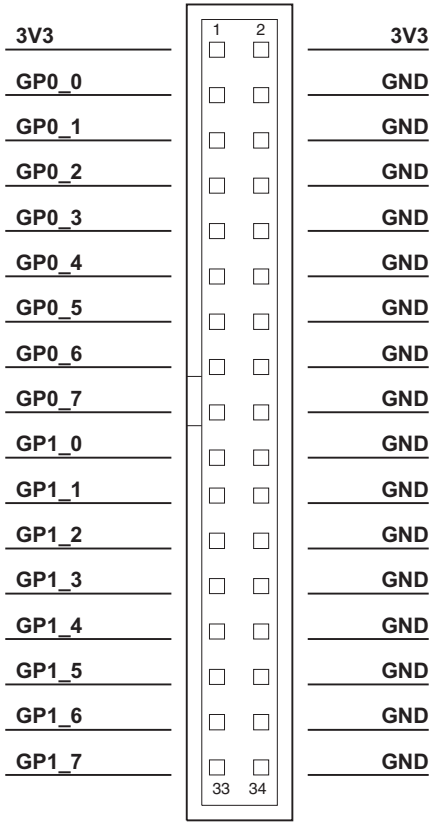

Figure A-4 GPIO connector J9

Note

Each data pin has an on-board 10kΩ pullup resistor to 3.3V.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.5 Keyboard and mouse interface

The pinout of the KMI connectors J23 and J24 is shown in Figure A-5.

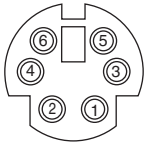

Figure A-5 KMI connector J22 and J23

Table A-4 lists the signals on the KMI connectors.

Table A-4 Mouse and keyboard port signal descriptions

Pin	Keyboard (KMI0, J23)		Mouse (KMI1, J22)	
	Signal	Function	Signal	Function
1	**KDATA**	Keyboard data	**MDATA**	Mouse Data
2	**MDATA**	Mouse Data	**KDATA**	Keyboard data
3	**GND**	Ground	**GND**	Ground
4	**5V**	5V	**5V**	5V
5	**KCLK**	Keyboard clock	**MCLK**	Mouse clock
6	**MCLK**	Mouse clock	**KCLK**	Keyboard clock

Note

The keyboard and the mouse signals go to both J22 and J23. Use a splitter cable to use only one of the connectors to connect both devices.

The normal data and clock pins (1 and 5) for J22 are used for the mouse and the normally unused pins (2 and 6) are used for the keyboard. A mouse can be plugged directly into J22.

J23 uses the normal I/O pins for the keyboard and the normally unused pins for the mouse. A keyboard can be plugged directly into J23.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.6 MMC and SD flash card interface

The MMC/SD card sockets provides nine pins that connect to the MultiMedia Card or Secure Digital card when it is inserted into the socket. Figure A-6 shows the pin numbering and signal assignment. In addition, the socket contains switches that are operated by card insertion and provide signaling on the **nCARDIN** and **WPROT** signals.

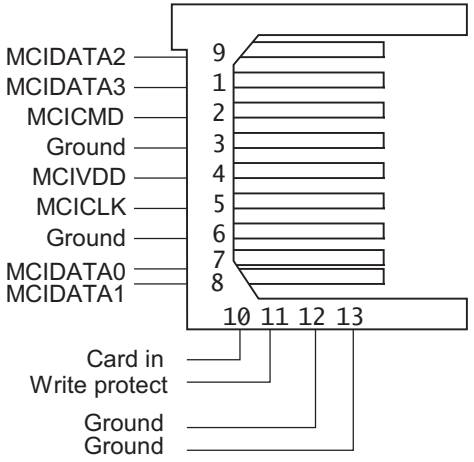

Figure A-6 MMC/SD card socket pin numbering, J21

The MMC card uses seven pins, and the SD card uses all nine pins. The additional pins are located as shown in Figure A-6 with pin 9 next to pin 1 and pins 7 and 8 spaced more closely together than the other pins. Figure A-7 shows an MMC card, with the contacts face up.

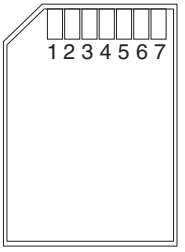

Figure A-7 MultiMedia Card (MMC)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-5 lists the signal assignments.

Table A-5 MultiMedia Card interface signals

Pin	Signal	Function in SD widebus mode	Function in MCI mode
1	**MCIDATA3**	Data	Chip select
2	**MCICMD**	Command/response	Data in
3	**GND**	Ground	Ground
4	**MCIVDD**	Supply voltage	Supply voltage
5	**MCICLK**	Clock	Clock
6	**GND**	Ground	Ground
7	**MCIDATA0**	Data 0	Data out
8	**MCIDATA1**	Data 1	NC
9	**MCIDATA2**	Data 2	NC
10 (DET A)	**CARDIN**	Card insertion detect	Card insertion detect
11 (DET B)	**WPROT**	Write protect status	Write protect status

Insert and remove the card as follows:

Insertion Insert the card into the socket with the contacts face down. Cards are normally labeled on the top surface with an arrow to indicate correct insertion.

Removal Remove the card by gently pressing it into the socket. It springs back and can be removed. Removing the card in this way ensures that the card detection switches within the socket operate correctly.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.7 PCI connector

The signals on the PCI connector are listed in Table A-6.

Table A-6 PCI connectors

Signal	Pin (B)	Pin (A)	Signal
-12V	1	1	**P5_nTRST**
P5_TCK	2	2	**12V**
GND	3	3	**P5_TMS**
P5_TDO	4	4	**P5_TDI**
5V	5	5	**5V**
5V	6	6	**P5_INTA**
P5_nINTB	7	7	**P5_INTC**
P5_nINTD	8	8	**5V**
P5_nPRSNT1	9	9	Reserved
NC	10	10	**P_VIO**
P5_nPRSNT2	11	11	Reserved
NC	12	12	Reserved
NC	13	13	Reserved
Reserved	14	14	Reserved
GND	15	15	**P5_nRST**
P5_CLK	16	16	**P_VIO**
GND	17	17	**P5_nGNT**
P5_nREQ	18	18	**GND**
PVIO	19	19	NC
P5_AD31	20	20	**P5_AD30**
P5_AD29	21	21	**3V3**
GND	22	22	**P5_AD28**
P5_AD27	23	23	**P5_AD26**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-6 PCI connectors (continued)

Signal	Pin (B)	Pin (A)	Signal
P5_AD25	24	24	**GND**
3V3	25	25	**P5_AD24**
PB5_nCBE3	26	26	**P5_IDSEL**
P5_AD23	27	27	**3V3**
GND	28	28	**P5_AD22**
P5_AD21	29	29	**P5_AD20**
P5_AD19	30	30	**GND**
3V3	31	31	**P5_AD18**
P5_AD17	32	32	**P5_AD16**
P5_nCBE2	33	33	**3V3**
GND	34	34	**P5_nFRAME**
P5_nIRDY	35	35	**GND**
GND	36	36	**P5_nTRDY**
P5_nDEVSEL	37	37	**GND**
XCAP	38	38	**P5_nSTOP**
NC (**nLOCK**)[a]	39	39	**3V3**
P5_nPERR	40	40	NC (**SDONE**)[a]
3V3	41	41	NC (**nSBO**)[a]
P5_nSERR	42	42	**GND**
3V3	43	43	**P5_PAR**
P5_nCBE1	44	44	**P5_AD15**
P5_AD14	45	45	**3V3**
GND	46	46	**P5_AD13**
P5_AD12	47	47	**P5_AD11**
P5_AD10	48	48	**GND**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-6 PCI connectors (continued)

Signal	Pin (B)	Pin (A)	Signal
p5_M66EN	49	49	**P5_AD9**
NC	50	50	NC
NC	51	51	NC
P5_AD8	52	52	**P5_nCBE0**
P5_AD7	53	53	**3V3**
3V3	54	54	**P5_AD6**
P5_AD5	55	55	**P5_AD4**
P5_AD3	56	56	**GND**
GND	57	57	**P5_AD2**
P5_AD1	58	58	**P5_AD0**
PVIO	59	59	**PVIO**
NC (**nACK64**)[a]	60	60	NC (**nREQ64**)[a]
5V	61	61	**5V**
5V	62	62	**5V**

a. This pin is no connect on the baseboard. The PCI signal normally present on this pin is shown in parentheses.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.8 PISMO connector

This section describes the pinout for the PISMO expansion memory board.

A.8.1 Expansion connector

The memory expansion board uses a 120-way QSH Samtec connector as shown in Figure A-8. The baseboard uses the corresponding QTH connector. The pinout is listed in Table A-8 on page A-15.

Note

The numbering of pins is for the view facing the connector. Some PISMO boards also have a QTH connector on the top of the memory board to enable stacking.

Table A-7 Samtec part numbers

Header	Part number	Mating height
baseboard (and PISMO board top)	QTH-060-01-F-D-A	5mm
PISMO board (bottom)	QSH-060-01-F-D-A	5mm

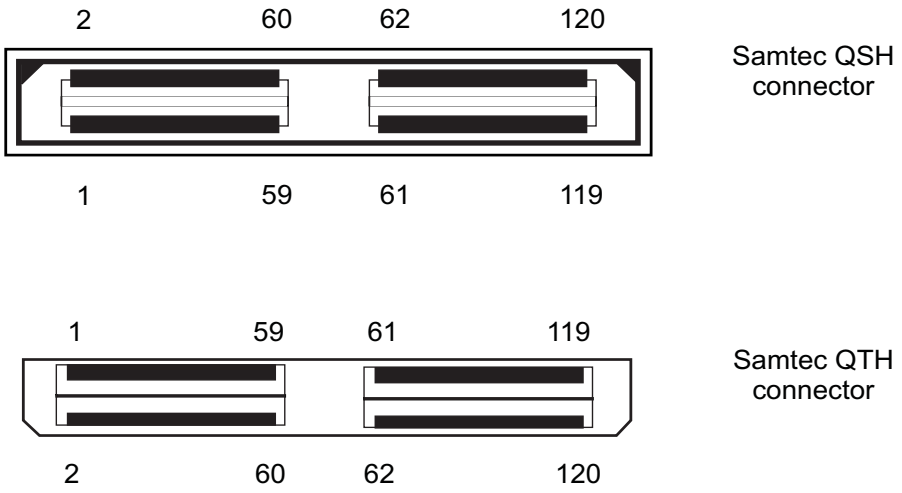

Figure A-8 Samtec connector

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-8 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	**nDRQ**
27	**DATA[13]**	28	Reserved, do not drive
29	**DATA[14]**	30	Reserved, do not drive
31	**DATA[15]**	32	Reserved, do not drive
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, do not drive
43	**DATA[21]**	44	Reserved, do not drive
45	**DATA[22]**	46	Reserved, do not drive
47	**DATA[23]**	48	Reserved, do not drive
49	**DATA[24]**	50	Reserved, do not drive

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-8 Memory connector signals (continued)

Pin No.	Signal	Pin No.	Signal
51	**DATA[25]**	52	Reserved, do not drive
53	**DATA[26]**	54	**DEMUX**
55	**DATA[27]**	56	**nSTANDBY**
57	**DATA[28]**	58	Reserved, do not drive
59	**DATA[29]**	60	Reserved, do not drive
61	**DATA[30]**	62	**SCL**, E2PROM serial interface clock (3.3V signal level)
63	**DATA[31]**	64	**SDA**, E2PROM serial interface data (3.3V signal level)
65	**ADDR[0]**	66	**nRESET**
67	**ADDR[1]**	68	**nPOR**, asserted on hardware power cycle
69	**ADDR[2]**	70	**nWP**, flash write protect. Drive HIGH to write to flash.
71	**ADDR[3]**	72	**nRESET_REQ**, Reset signal. Differs from **nRESET** in that it is not delayed by **nWAIT**.
73	**ADDR[4]**	74	**nBUSY**, Wait mode input from external memory controller. Pull HIGH if not used.
75	**ADDR[5]**	76	**nBWAIT**, Synchronous burst wait input. This is used by the external device to delay a synchronous burst transfer if LOW. Pull to HIGH if not used.
77	**ADDR[6]**	78	Reserved, do not drive
79	**ADDR[7]**	80	**nCS[4]**
81	**ADDR[8]**	82	**nCS[3]**
83	**ADDR[9]**	84	**nCS[2]**
85	**ADDR[10]**	86	**nCS[1]**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-8 Memory connector signals (continued)

Pin No.	Signal	Pin No.	Signal
87	**ADDR[11]**	88	**ADDR[29]**
89	**ADDR[12]**	90	**ADDR[28]**
91	**ADDR[13]**	92	**ADDR[27]**
93	**ADDR[14]**	94	**ADDR[26]**
95	**ADDR[15]**	96	**nCS[0]**
97	**ADDR[16]**	98	**nRESET_BUSY**, 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	**nWE**
103	**ADDR[19]**	104	**nOE**
105	**ADDR[20]**	106	**nBE[3]**, Byte Lane Select for bits [31:24]
107	**ADDR[21]**	108	**nBE[2]**, Byte Lane Select for bits [23:16]
109	**ADDR[22]**	110	**nBE[1]**, Byte Lane Select for bits [15:8]
111	**ADDR[23]**	111	**nBE[0]**, Byte Lane Select for bits [7:0]
113	**ADDR[24]**	114	**WIDTH[0]**, Indicates bus width for fitted part. Do not route through stackable boards.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-8 Memory connector signals (continued)

Pin No.	Signal	Pin No.	Signal
115	**ADDR[25]**	116	**WIDTH[1]**, Indicates bus width for fitted part. Do not route through stackable boards.
117	**nLBA**, Indicates that the address output is stable during synchronous burst transfers	118	**CLK[1]**, a buffered version of **SMCLK[1]**.
119	**BAA**, Burst Address Advance. Used to advance the address count in the memory device	120	**CLK[0]**, a buffered version of **SMCLK[0]**.

a. VDDIO is the data voltage to host. Do not route through on stackable boards

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.9 Smart Card interface

J33 is a Smart Card (SIM) socket. (J31 is for an alternate layout socket and is not fitted).

J33 includes a switch for card detection.

The signals on the SIM socket are also connected to the SCI expansion socket. The signals associated with the SCI are listed in Table A-9.

Table A-9 Smart Card connector signal assignment

Pin	Signal	Description
C1	**SC_VCC**	Card power (1.8V, 3.3V, or 5V)
C2	**SC_RST**	Reset to card
C3	**SC_CLK**	Clock to or from card
C4	-	Not present on J33, NC on J31.
C5	**GND**	Ground
C6	**SC_VCC**	Card power (1.8V, 3.3V, or 5V)
C7	**SC_IO**	Serial data to or from the card
C8	-	Not present on J33, NC on J31.
SW1	**nSCIDETECTx**	Card detect signal from switch in socket (not present on J31)

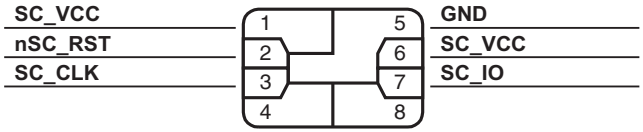

Figure A-9 Smart Card contacts assignment

Figure A-9 shows the signal assignment of a Smart Card. Pins 4 and 8 are not connected and are omitted on some cards.

The SIM card is inserted into one of the SIM card sockets with the contacts face down.

Figure A-10 on page A-20 shows the pinout of the connector J30. This can be used to connect to an off-PCB smart card device.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

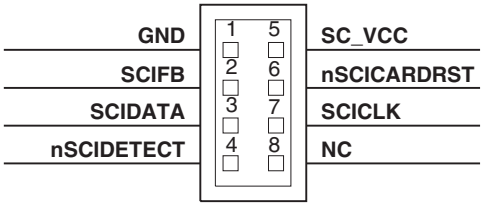

Figure A-10 J30 SCI expansion connector J30

Table A-10 lists the signals on the SCI expansion connector.

Table A-10 Signals on expansion connector

Pin	Signal	Description
1	**GND**	Ground
2	**SCIFCB**	SIM function code bit
3	**SCIDATA**	SIM data
4	**nSCIDETECT**	Card detect for SIM 0
5	**SC_VCC**	SIM power
6	**nSCICARDRST**	Active LOW reset to SIM 0
7	**SCICLK**	SIM clock
8	NC	Not connected

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.10 Synchronous Serial Port interface

Figure A-11 shows the signals on the expansion SSP interface connector J32.

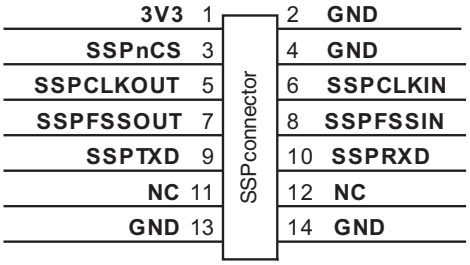

Figure A-11 SSP expansion interface, J32

The signals associated with the SSP are listed in Table A-11.

Table A-11 SSP signal assignment

Pin	Signal name	Description
1	**3V3**	3.3V supply
2	**GND**	Ground
3	**SSPnCS**	Chip select
4	**GND**	Ground
5	**SSPCLKOUT**	Clock output from controller
6	**SSPCLKIN**	Clock input to controller
7	**SSPFSSOUT**	Frame sync output
8	**SSPFSSIN**	Frame sync input
9	**SSPTXD**	Data output
10	**SSPRXD**	Data input
11	NC	Not connected
12	NC	Not connected
13	**GND**	Ground
14	**GND**	Ground

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.11 Test and debug connections

This section contains the following subsections:

- *Overview of test points*
- *JTAG* on page A-30
- *USB debug port* on page A-31
- *Trace connector pinout* on page A-31
- *Integrated Logic Analyzer* on page A-33.

The baseboard provides test points, ground points, and connectors to aid diagnostics as shown in Figure A-12 on page A-24, Figure A-13 on page A-26, and Figure A-14 on page A-29.

A.11.1 Overview of test points

The test points and test connectors are shown in Figure A-12 on page A-24. The functions of the test points on the baseboard are summarized in Table A-12.

Test points not listed are not present or are reserved for manufacturing use only.

Table A-12 Test point functions

Test point	Signal	Function
TP1	**VBATT**	Backup battery voltage
TP2	**OSCCLK0**	Output from programmable oscillator 0
TP3	**GLOBALCLKOUTTP1**	Global clock output
TP4	**REFCLK24MHZ**	24MHz reference from OSC0
TP5	**OSCCLK1**	Output from programmable oscillator 1
TP6	**OSCCLK2**	Output from programmable oscillator 2
TP7	**OSCCLK3**	Output from programmable oscillator 3
TP8	**OSCCLK4**	Output from programmable oscillator 4
TP9	**REFCLK32K**	32.768kHz reference clock
TP10	**INTCLK**	Test clock from USB debug logic
TP11	**EnRST**	Reset output signal related to USB debug test

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-12 Test point functions (continued)

Test point	Signal	Function
J7	**DXP** and **DXN**	Temperature monitor. Connect a meter to this voltage to monitor the FPGA operating temperature. (Refer to the Xilinx data sheet for more information.)
TP14	-	Not present
TP15	**RAW_CLK**	Clock signal from USB debug interface
TP16	**12VDC**	12V DC supply
TP17	**5V_ANALOG**	5V DC supply to analog circuitry
TP18	**1V8**	1.8V supply for PLDs, flash memory, and PISMO memory boards
TP19	**1V5**	1.5V supply for FPGA and DDR memory
TP20	**VDDCORE25_SW1**	2.5V supply to DDR
TP22	**GLOBALCLKINTP1**	Global clock input
TP23	**GLOBALCLKINTP2**	Global clock input
TP24	**GLOBALCLKOUTTP2**	Global clock output
TP30	**VDDCORE18_SW**	1.8V power supply to USB debug PLD

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

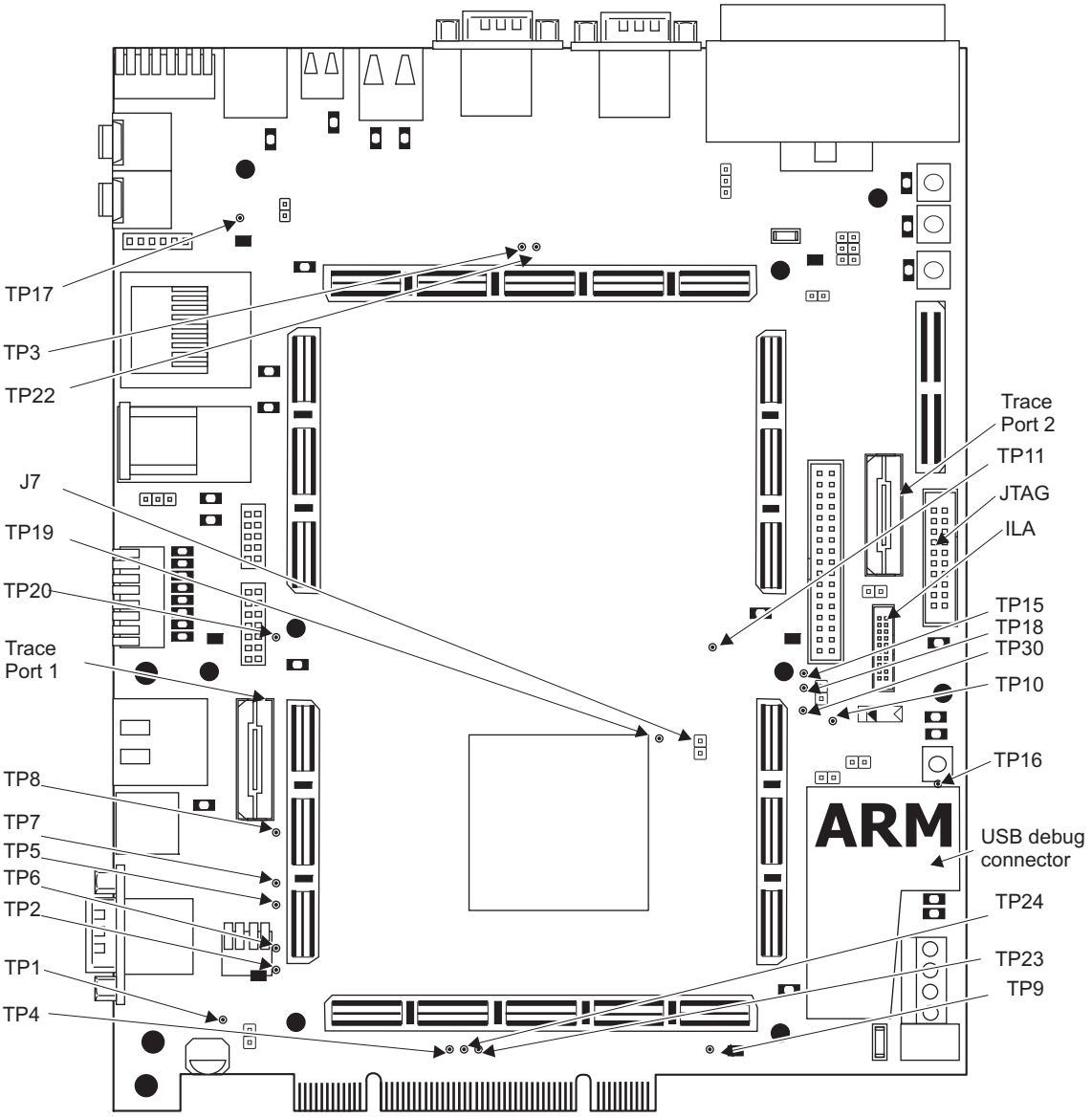

Figure A-12 Test points and test connectors

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Overview of links

The links are shown in Figure A-13 on page A-26. The functions of the links are summarized in Table A-13.

Table A-13 Links

Link	Function
J34	Smart Card voltage select. Fit link across AB for 3.3V operation or across BC for 5V operation. Omit link for 1.8V Smart Cards.
J47	Used for manufacturing test only
J48	Force on. Connect link to force the power on and prevent the standby push button from switching the power off.
LK1	Microphone voltage. Fit link across AB to connect microphone bias voltage to microphone 1 or fit BC to connect voltage to microphone 2. Omit link if microphone does not require bias voltage.
LK2	Character LCD voltage (fitted at manufacture). Link fitted on AB for 5V devices or BC for 3.3V devices. This link is fitted at manufacture and must not be changed.
LK4-8	Manufacturing test. Remove link to disconnect switching regulator outputs. Remove LK4 to disconnect the 5V regulator for the analog circuitry Remove LK5 to disconnect the 5V regulator for the digital logic Remove LK6 to disconnect the 3.3V regulator Remove LK7 to disconnect the 1.8V regulator Remove LK8 to disconnect the 1.5V regulator
LK9	NOR flash VCDQ voltage select. Link fitted to BC if the flash memory devices fitted to the board are L30 parts that operate at 1.8V. Link fitted to AB for all other parts. This link is fitted at manufacture and must not be changed.
R410	Manufacturing test link. Selects 12MHz external clock for USB interface This link is not fitted for production boards.
R223-R225	Links DSR to DTR for serial ports 1 to 3. This link is fitted as standard on production boards.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

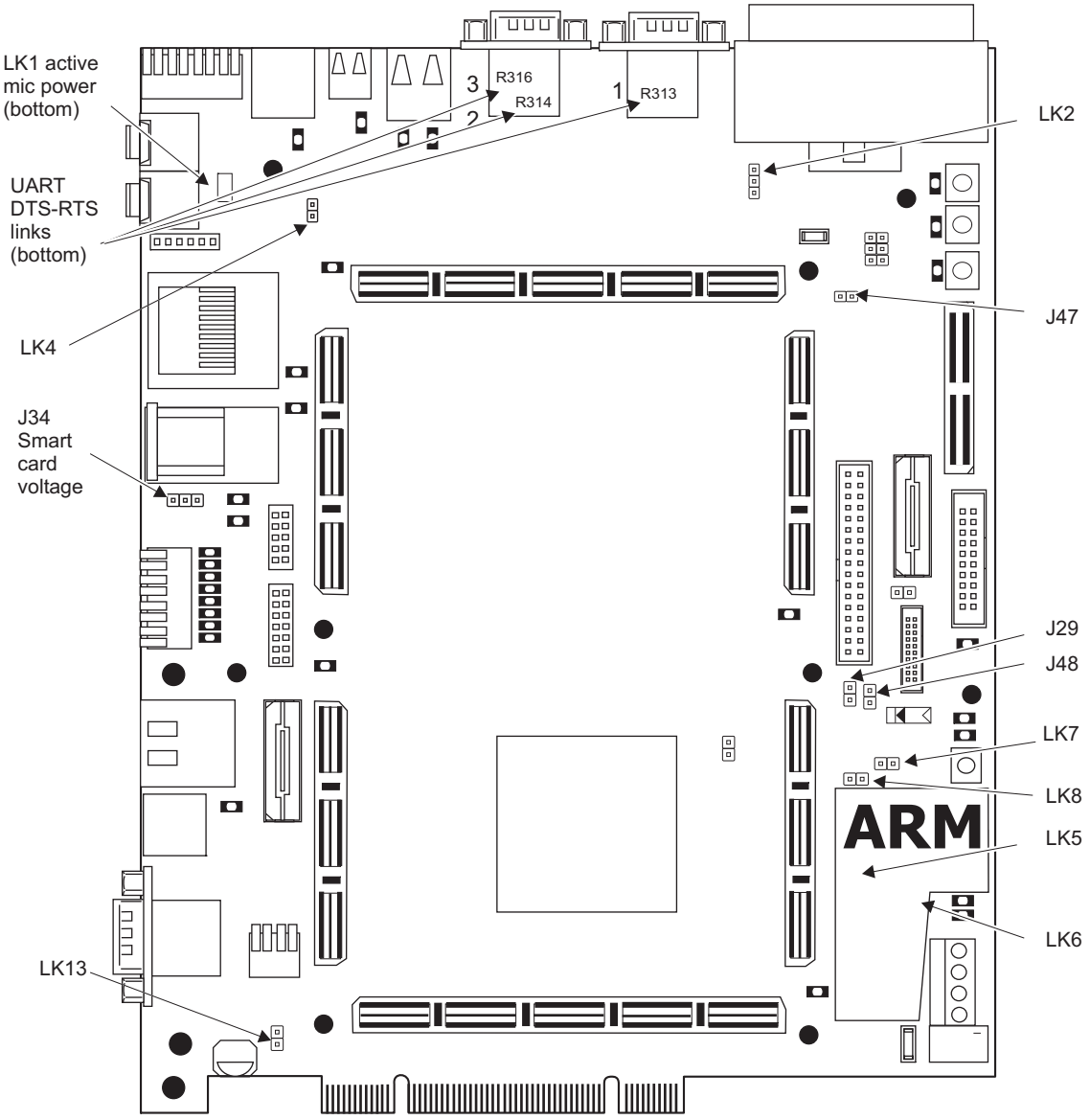

Figure A-13 Links

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Overview of switches and indicators

The switches and indicators are shown in Figure A-14 on page A-29. The functions of the switches and indicators are summarized in Table A-15 on page A-28 and Table A-14.

Table A-14 Indicators

LED	Function
D1	Fused 5V OK
D2, D5	Tile site 1 detect
D4	CONFIG LED
D6, D7	Tile site 2 detect
D8	Reset switch
D9	Indicator for user push button switch
D10	Indicator for FPGA config push button switch
D12	USB debug busy
D13	USB debug on
D[21:14]	User LEDs (**LED[7:0]**)
D25	On/Standby
D29	3.3V OK
D30	USB1 power
D31	USB2 power
D35	Global done (**GLOBAL_DONE**), all FPGAs in the system have been configured
D36	Local done (**LOCAL_DONE**), the baseboard FPGA has been configured
D37	USB 3 power

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-15 Switches

Switch	Function
S1	Config switch. Slide switch to ON position to put the system into JTAG configuration mode.
S2	Reset push button
S3	General-purpose push button
S4	Reconfigure FPGA push button
S5	Not fitted
S6	User switches **USERSW[7:0]**
S7	Power/standby push button
S8	Boot switches **BOOTSEL[7:0]**
S9	Not fitted
S10	FPGA image select switches **CFG_SEL[4:1]**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

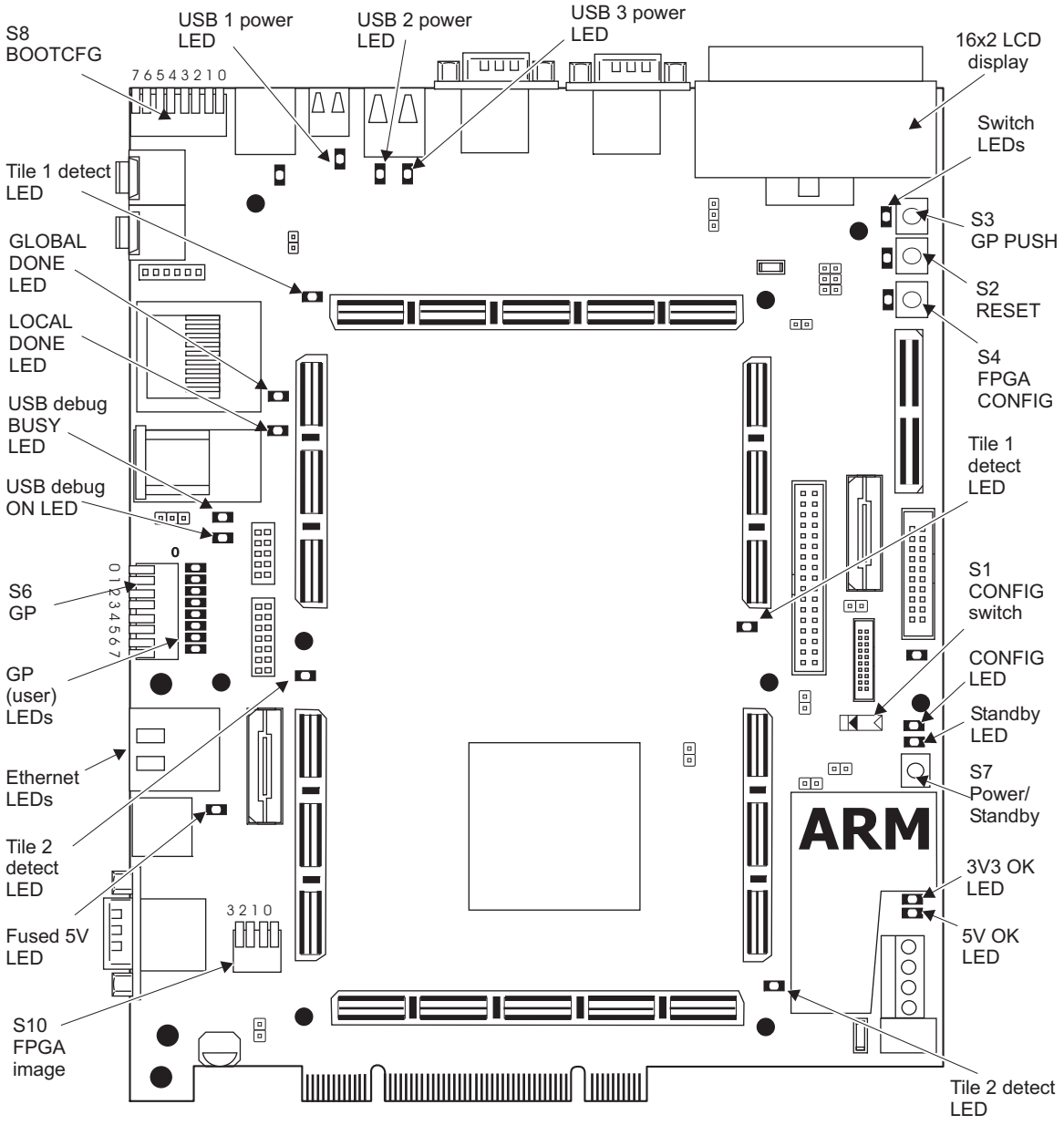

Figure A-14 Switches and indicators

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.11.2 JTAG

Figure A-15 shows the pinout of the JTAG connector J18 and Table A-16 list the JTAG and signals (see also Table 3-21 on page 3-87). All JTAG active HIGH input signals have pull-up resistors (DGBRQ is active LOW and has a pull-down resistor).

Note

The term JTAG equipment refers to any hardware (typically this is RealView ICE or Multi-ICE) that can drive the JTAG signals to devices in the scan chain. All pins not listed in Table A-16 are connected to ground.

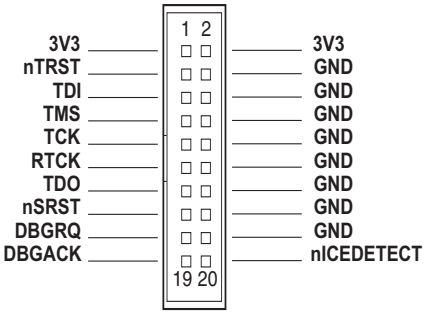

Figure A-15 JTAG connector J18

Table A-16 JTAG signals

Signal	Pin	Description
3V3	1, 2	3.3V supply (and reference voltage) to JTAG interface unit
nTRST	3	Open-collector test reset (from JTAG equipment)
TDI	5	Test data in
TMS	7	Test mode select
TCK	9	Test clock
RTCK	11	Return **TCK**
TDO	13	Test data out
nSRST	15	Open collector system reset (bidirectional)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-16 JTAG signals (continued)

Signal	Pin	Description
DBGRQ	17	Debug request to processor
DBGACK	19	Debug acknowledge from processor
nICEDETECT	20	Detects debug device connected to baseboard. The external devices connects pin 20 to ground.

A.11.3 USB debug port

Figure A-16 shows the signals on the USB debug connector J16. **USBDP** and **USBDM** are the positive and negative USB data signals.

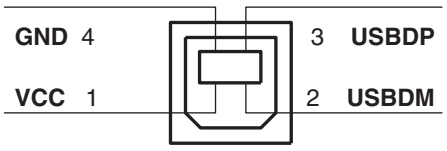

Figure A-16 USB debug connector J16

A.11.4 Trace connector pinout

Table A-17 on page A-32 and Table A-18 on page A-33 list the pinout of the trace connector J49 (tile site 1) and J50 (tile site 2). The Mictor connector (part number AMP 2-767004-2) is shown in Figure A-17 on page A-32.

Note

Connectors J49 and J50 can be used as trace connectors for Logic Tiles that contain an implementation of a processor. If the Logic Tile does not contain a processor design, the FPGA might output debug signals on the connectors.

For Core Tiles, use the trace connectors on the tile.

Agilent (formerly HP) and Tektronix label these connectors differently, but the assignments of signals to physical pins is appropriate for both systems and pin 1 is always in the same place. The figure is labelled according to the Agilent pin assignment.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

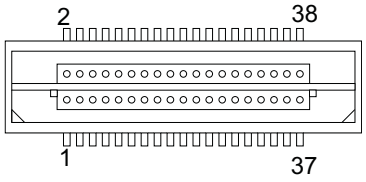

Figure A-17 AMP Mictor connector

Table A-17 Trace connector J49

Channel	Pin	Pin	Channel
Not connected	1	2	Not connected
GND	3	4	Not connected
T1Y178	5	6	**T1Y179**
T1Y177	7	8	**T1Y168**
T1Y152	9	10	**T1Y167**
T1Y151	11	12	**T1Y166**
T1Y150	13	14	**T1Y165**
T1Y149	15	16	**T1Y164**
T1Y148	17	18	**T1Y163**
T1Y147	19	20	**T1Y162**
T1Y146	21	22	**T1Y161**
T1Y176	23	24	**T1Y160**
T1Y175	25	26	**T1Y159**
T1Y174	27	28	**T1Y158**
T1Y173	29	30	**T1Y157**
T1Y172	31	32	**T1Y156**
T1Y171	33	34	**T1Y155**
T1Y170	35	36	**T1Y154**
T1Y169	37	38	**T1Y153**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-18 Trace connector J50

Channel	Pin	Pin	Channel
Not connected	1	2	Not connected
GND	3	4	Not connected
T2Y178	5	6	**T2Y179**
T2Y177	7	8	**T2Y168**
T2Y152	9	10	**T2Y167**
T2Y151	11	12	**T2Y166**
T2Y150	13	14	**T2Y165**
T2Y149	15	16	**T2Y164**
T2Y148	17	18	**T2Y163**
T2Y147	19	20	**T2Y162**
T2Y146	21	22	**T2Y161**
T2Y176	23	24	**T2Y160**
T2Y175	25	26	**T2Y159**
T2Y174	27	28	**T2Y158**
T2Y173	29	30	**T2Y157**
T2Y172	31	32	**T2Y156**
T2Y171	33	34	**T2Y155**
T2Y170	35	36	**T2Y154**
T2Y169	37	38	**T2Y153**

A.11.5 Integrated Logic Analyzer

Figure A-18 on page A-34 shows the signals on the *Integrated Logic Analyzer* (ILA) connector J19. Use an ILA to debug the baseboard and Logic Tile FPGA designs and software at the same time. For more information, see the documentation supplied with your analyzer. (The ChipScope ILA product, for example, is described on the Xilinx web site.)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

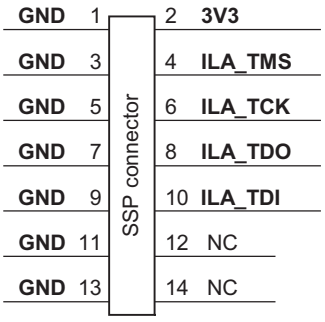

Figure A-18 Integrated logic analyzer connector J19

Note

All signals on the ILA connector have 10kΩ pullup resistors to 3.3V.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.12 Tile header connectors

Figure A-19 on page A-36 shows the pin numbers and power-blade usage of the HDRX, HDRY, and HDRZ headers on the baseboard.

The pin numbering and supply voltages are the same for both tile sites.

Caution

The I/O voltage on some pins of Logic Tiles can be programmed by changing resistors on the tile. Signals between Logic Tiles can be altered safely if both the sending and receiving tile use the same voltage. (See *Power supply* on page 3-28.)

HDRX signals on page A-37, *HDRY signals* on page A-41, and *HDRZ signals* on page A-45 list the signals on each header pin. See the *ARM LT-XC2V4000+ RealView Logic Tile User Guide* for details of the header signals present on the Logic Tile. See the *Core Tile User Guide* for details of the header signals present on the Core Tile.

Table A-19 lists the Samtec part numbers.

The baseboard uses QTH-xxx-05-F-D-A-K connectors on the top of the board.

Core Tiles and Logic Tiles use the corresponding QSH-xxx-01-F-D-A-D connectors on the bottom of the tiles.

Note

Samtec describes the QTH connector (used on the baseboard) as a header and the QSH connector (used on the lower side of the tiles) as a socket.

Table A-19 Samtec part numbers

Header	Part number	Mating height
baseboard HDRX	QTH-090-05-F-D-A-K	19mm
baseboard HDRY	QTH-090-05-F-D-A-K	19mm
baseboard HDRZ	QTH-150-05-F-D-A-K	19mm
Core Tile HDRX (bottom)	QSH-090-01-F-D-A-K	5mm
Core Tile HDRY (bottom)	QSH-090-01-F-D-A-K	5mm
Core Tile HDRZ (bottom)	QSH-150-01-F-D-A-K	5mm

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

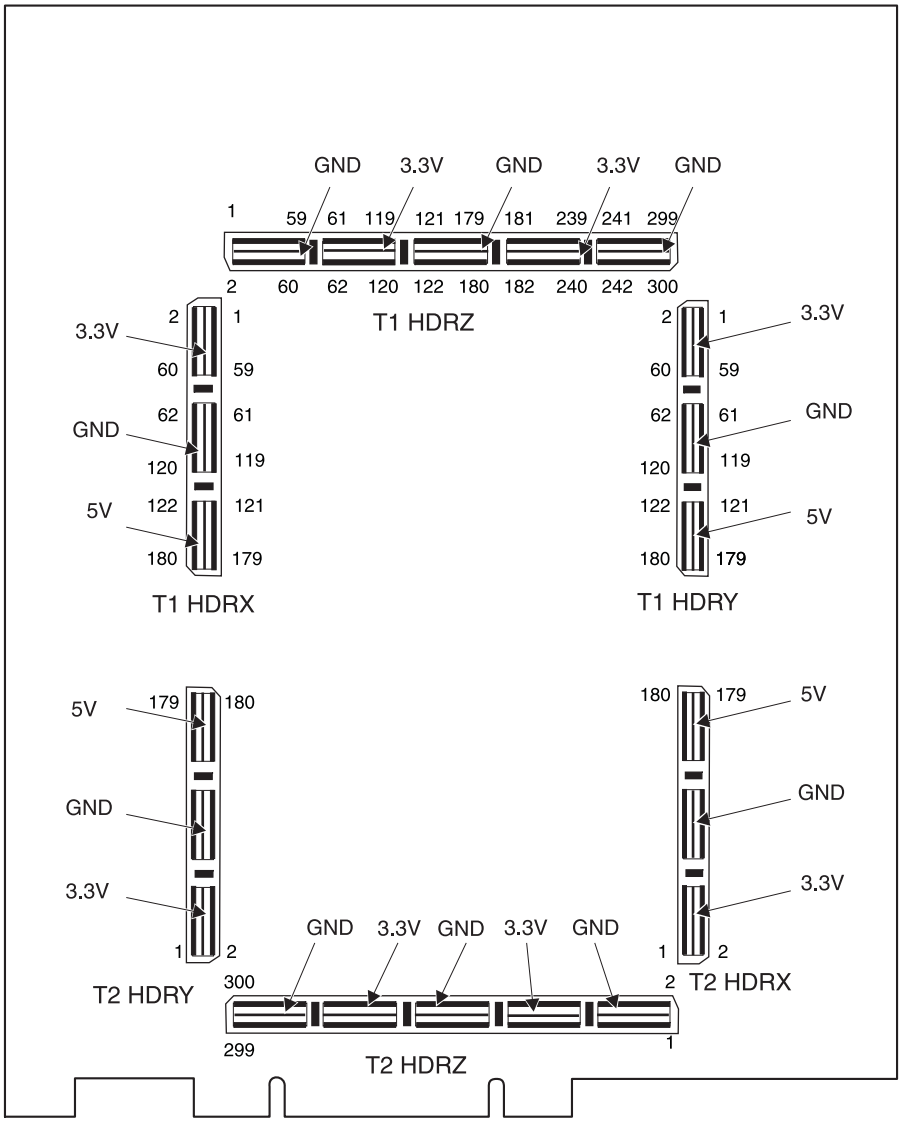

Figure A-19 HDRX, HDRY, and HDRZ pin numbering

A.12.1 HDRX signals

Table A-20 on page A-37 lists the signals on the HDRX (J37 and J41) pins.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

There are two HDRX connectors (tile site 1 and tile site 2), replace the **TnX** in the table by **T1X** or **T2X** to get the signal name for tile site 1 or 2.

Table A-20 HDRX signals

baseboard signal	Pin	Pin	baseboard signal
TnX89	2	1	**TnX90**
TnX88	4	3	**TnX91**
TnX87	6	5	**TnX92**
TnX86	8	7	**TnX93**
TnX85	10	9	**TnX94**
TnX84	12	11	**TnX95**
TnX83	14	13	**TnX96**
TnX82	16	15	**TnX97**
TnX81	18	17	**TnX98**
TnX80	20	19	**TnX99**
TnX79	22	21	**TnX100**
TnX78	24	23	**TnX101**
TnX77	26	25	**TnX102**
TnX76	28	27	**TnX103**
TnX75	30	29	**TnX104**
TnX74	32	31	**TnX105**
TnX73	34	33	**TnX106**
TnX72	36	35	**TnX107**
TnX71	38	37	**TnX108**
TnX70	40	39	**TnX109**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-20 HDRX signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TnX69	42	41	**TnX110**
TnX68	44	43	**TnX111**
TnX67	46	45	**TnX112**
TnX66	48	47	**TnX113**
TnX65	50	49	**TnX114**
TnX64	52	51	**TnX115**
TnX63	54	53	**TnX116**
TnX62	56	55	**TnX117**
TnX61	58	57	**TnX118**
TnX60	60	59	**TnX119**
TnX59	62	61	**TnX120**
TnX58	64	63	**TnX121**
TnX57	66	65	**TnX122**
TnX56	68	67	**TnX123**
TnX55	70	69	**TnX124**
TnX54	72	71	**TnX125**
TnX53	74	73	**TnX126**
TnX52	76	75	**TnX127**
TnX51	78	77	**TnX128**
TnX50	80	79	**TnX129**
TnX49	82	81	**TnX130**
TnX48	84	83	**TnX131**
TnX47	86	85	**TnX132**
TnX46	88	87	**TnX133**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-20 HDRX signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TnX45	90	89	**TnX134**
TnX44	92	91	**TnX135**
TnX43	94	93	**TnX136**
TnX42	96	95	**TnX137**
TnX41	98	97	**TnX138**
TnX40	100	99	**TnX139**
TnX39	102	101	**TnX140**
TnX38	104	103	**TnX141**
TnX37	106	105	**TnX142**
TnX36	108	107	**TnX143**
TnX35	110	109	NC
TnX34	112	111	NC
TnX33	114	113	NC
TnX32	116	115	NC
TnX31	118	117	NC
TnX30	120	119	NC
TnX29	122	121	NC
TnX28	124	123	NC
TnX27	126	125	NC
TnX26	128	127	NC
TnX25	130	129	NC
TnX24	132	131	NC
TnX23	134	133	NC
TnX22	136	135	NC

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-20 HDRX signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TnX21	138	137	NC
TnX20	140	139	NC
TnX19	142	141	NC
TnX18	144	143	NC
TnX17	146	145	NC
TnX16	148	147	NC
TnX15	150	149	NC
TnX14	152	151	NC
TnX13	154	153	NC
TnX12	156	155	NC
TnX11	158	157	NC
TnX10	160	159	NC
TnX9	162	161	NC
TnX8	164	163	NC
TnX7	166	165	NC
TnX6	168	167	NC
TnX5	170	169	NC
TnX4	172	171	NC
TnX3	174	173	NC
TnX2	176	175	NC
TnX1	178	177	NC
TnX0	180	179	NC

A.12.2 HDRY signals

Table A-21 on page A-41 lists the signals on the HDRY (J38 and J43) pins.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

There are two HDRY connectors (tile site 1 and tile site two), replace the **TnY** in the table by **T1Y** or **T2Y** to get the signal name for tile site 1 or 2.

Table A-21 HDRY signals

baseboard signal	Pin	Pin	baseboard signal
TnY90	2	1	**TnY89**
TnY91	4	3	**TnY88**
TnY92	6	5	**TnY87**
TnY93	8	7	**TnY86**
TnY94	10	9	**TnY85**
TnY95	12	11	**TnY84**
TnY96	14	13	**TnY83**
TnY97	16	15	**TnY82**
TnY98	18	17	**TnY81**
TnY99	20	19	**TnY80**
TnY100	22	21	**TnY79**
TnY101	24	23	**TnY78**
TnY102	26	25	**TnY77**
TnY103	28	27	**TnY76**
TnY104	30	29	**TnY75**
TnY105	32	31	**TnY74**
TnY106	34	33	**TnY73**
TnY107	36	35	**TnY72**
TnY108	38	37	**TnY71**
TnY109	40	39	**TnY70**
TnY110	42	41	**TnY69**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-21 HDRY signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TnY111	44	43	**TnY68**
TnY112	46	45	**TnY67**
TnY113	48	47	**TnY66**
TnY114	50	49	**TnY65**
TnY115	52	51	**TnY64**
TnY116	54	53	**TnY63**
TnY117	56	55	**TnY62**
TnY118	58	57	**TnY61**
TnY119	60	59	**TnY60**
TnY120	62	61	**TnY59**
TnY121	64	63	**TnY58**
TnY122	66	65	**TnY57**
TnY123	68	67	**TnY56**
TnY124	70	69	**TnY55**
TnY125	72	71	**TnY54**
TnY126	74	73	**TnY53**
TnY127	76	75	**TnY52**
TnY128	78	77	**TnY51**
TnY129	80	79	**TnY50**
TnY130	82	81	**TnY49**
TnY131	84	83	**TnY48**
TnY132	86	85	**TnY47**
TnY133	88	87	**TnY46**
TnY134	90	89	**TnY45**
TnY135	92	91	**TnY44**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-21 HDRY signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TnY136	94	93	**TnY43**
TnY137	96	95	**TnY42**
TnY138	98	97	**TnY41**
TnY139	100	99	**TnY40**
TnY140	102	101	**TnY39**
TnY141	104	103	**TnY38**
TnY142	106	105	**TnY37**
TnY143	108	107	**TnY36**
NC	110	109	**TnY35**
NC	112	111	**TnY34**
TnY146	114	113	**TnY33**
TnY147	116	115	**TnY32**
TnY148	118	117	**TnY31**
TnY149	120	119	**TnY30**
TnY150	122	121	**TnY29**
TnY151	124	123	**TnY28**
TnY152	126	125	**TnY27**
TnY153	128	127	**TnY26**
TnY154	130	129	**TnY25**
TnY155	132	131	**TnY24**
TnY156	134	133	**TnY23**
TnY157	136	135	**TnY22**
TnY158	138	137	**TnY21**
TnY159	140	139	**TnY20**
TnY160	142	141	**TnY19**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-21 HDRY signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TnY161	144	143	**TnY18**
TnY162	146	145	**TnY17**
TnY163	148	147	**TnY16**
TnY164	150	149	**TnY15**
TnY165	152	151	**TnY14**
TnY166	154	153	**TnY13**
TnY167	156	155	**TnY12**
TnY168	158	157	**TnY11**
TnY169	160	159	**TnY10**
TnY170	162	161	**TnY9**
TnY171	164	163	**TnY8**
TnY172	166	165	**TnY7**
TnY173	168	167	**TnY6**
TnY174	170	169	**TnY5**
TnY175	172	171	**TnY4**
TnY176	174	173	**TnY3**
TnY177	176	175	**TnY2**
TnY178	178	177	**TnY1**
TnY179	180	179	**TnY0**

A.12.3 HDRZ

Table A-22 on page A-45 lists the signals on the HDRZ (J36 and J42) pins.

Note

There are two HDRZ connectors (tile site 1 and tile site two), replace the **TnZ** in the table by **T1Z** or **T2Z** to get the signal name for tile site 1 or 2.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

HDRZ signals **TZ[233:232]** and **TZ[199:128]** are common on both tiles.

Table A-22 HDRZ signals

baseboard signal	Pin	Pin	baseboard signal
NC	2	1	**TZ128**
NC	4	3	**TZ129**
NC	6	5	**TZ130**
NC	8	7	**TZ131**
NC	10	9	**TZ132**
NC	12	11	**TZ133**
NC	14	13	**TZ134**
NC	16	15	**TZ135**
NC	18	17	**TZ136**
NC	20	19	**TZ137**
NC	22	21	**TZ138**
NC	24	23	**TZ139**
NC	26	25	**TZ140**
NC	28	27	**TZ141**
NC	30	29	**TZ142**
NC	32	31	**TZ143**
NC	34	33	**TZ144**
NC	36	35	**TZ145**
NC	38	37	**TZ146**
NC	40	39	**TZ147**
NC	42	41	**TZ148**
NC	44	43	**TZ149**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-22 HDRZ signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TZ233	46	45	**TZ150**
TZ232	48	47	**TZ151**
TnZ231	50	49	**TZ152**
TnZ230	52	51	**TZ153**
TnZ229	54	53	**TZ154**
TnZ228	56	55	**TZ155**
TnZ227	58	57	**TZ156**
TnZ226	60	59	**TZ157**
TnZ225	62	61	**TZ158**
TnZ224	64	63	**TZ159**
TnZ223	66	65	**TZ160**
TnZ222	68	67	**TZ161**
TnZ221	70	69	**TZ162**
TnZ220	72	71	**TZ163**
TnZ219	74	73	**TZ164**
TnZ218	76	75	**TZ165**
TnZ217	78	77	**TZ166**
TnZ216	80	79	**TZ167**
TnZ215	82	81	**TZ168**
TnZ214	84	83	**TZ169**
TnZ213	86	85	**TZ170**
TnZ212	88	87	**TZ171**
TnZ211	90	89	**TZ172**
TnZ210	92	91	**TZ173**
TnZ209	94	93	**TZ174**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-22 HDRZ signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TnZ208	96	95	**TZ175**
TnZ207	98	97	**TZ176**
TnZ206	100	99	**TZ177**
TnZ205	102	101	**TZ178**
TnZ204	104	103	**TZ179**
TnZ203	106	105	**TZ180**
TnZ202	108	107	**TZ181**
TnZ201	110	109	**TZ184**
TnZ200	112	111	**TZ182**
TZ199	114	113	**TZ183**
TZ198	116	115	**TZ185**
TZ197	118	117	**TZ186**
TZ196	120	119	**TZ187**
TZ195	122	121	**TZ188**
TZ194	124	123	**TZ189**
TZ193	126	125	**TZ190**
TZ192	128	127	**TZ191**
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_OUT	136	135	**D_TDI**
CLK_UP_THRU	138	137	**D_TCK_OUT**
CLK_OUT_PLUS1	140	139	**D_TMS_OUT**
CLK_OUT_PLUS2	142	141	**D_RTCK**
CLK_IN_PLUS2	144	143	**C_nSRST**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-22 HDRZ signals (continued)

baseboard signal	Pin	Pin	baseboard signal
CLK_IN_PLUS1	146	145	**C_nTRST**
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)
TnZ64	174	173	**TnZ63**
TnZ65	176	175	**TnZ62**
TnZ66	178	177	**TnZ61**
TnZ67	180	179	**TnZ60**
TnZ68	182	181	**TnZ59**
TnZ69	184	183	**TnZ58**
TnZ70	186	185	**TnZ57**
TnZ71	188	187	**TnZ56**
TnZ72	190	189	**TnZ55**
TnZ73	192	191	**TnZ54**
TnZ74	194	193	**TnZ53**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-22 HDRZ signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TnZ75	196	195	**TnZ52**
TnZ76	198	197	**TnZ51**
TnZ77	200	199	**TnZ50**
TnZ78	202	201	**TnZ49**
TnZ79	204	203	**TnZ48**
TnZ80	206	205	**TnZ47**
TnZ81	208	207	**TnZ46**
TnZ82	210	209	**TnZ45**
TnZ83	212	211	**TnZ44**
TnZ84	214	213	**TnZ43**
TnZ85	216	215	**TnZ42**
TnZ86	218	217	**TnZ41**
TnZ87	220	219	**TnZ40**
TnZ88	222	221	**TnZ39**
TnZ89	224	223	**TnZ38**
TnZ90	226	225	**TnZ37**
TnZ91	228	227	**TnZ36**
TnZ92	230	229	**TnZ35**
TnZ93	232	231	**TnZ34**
TnZ94	234	233	**TnZ33**
TnZ95	236	235	**TnZ32**
TnZ96	238	237	**TnZ31**
TnZ97	240	239	**TnZ30**
TnZ98	242	241	**TnZ29**
TnZ99	244	243	**TnZ28**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-22 HDRZ signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TnZ100	246	245	**TnZ27**
TnZ101	248	247	**TnZ26**
TnZ102	250	249	**TnZ25**
TnZ103	252	251	**TnZ24**
TnZ104	254	253	**TnZ23**
TnZ105	256	255	**TnZ22**
TnZ106	258	257	**TnZ21**
TnZ107	260	259	**TnZ20**
TnZ108	262	261	**TnZ19**
TnZ109	264	263	**TnZ18**
TnZ110	266	265	**TnZ17**
TnZ111	268	267	**TnZ16**
TnZ112	270	269	**TnZ15**
TnZ113	272	271	**TnZ14**
TnZ114	274	273	**TnZ13**
TnZ115	276	275	**TnZ12**
TnZ116	278	277	**TnZ11**
TnZ117	280	279	**TnZ10**
TnZ118	282	281	**TnZ9**
TnZ119	284	283	**TnZ8**
TnZ120	286	285	**TnZ7**
TnZ121	288	287	**TnZ6**
TnZ122	290	289	**TnZ5**
TnZ123	292	291	**TnZ4**
TnZ124	294	293	**TnZ3**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table A-22 HDRZ signals (continued)

baseboard signal	Pin	Pin	baseboard signal
TnZ125	296	295	**TnZ2**
TnZ126	298	297	**TnZ1**
TnZ127	300	299	**TnZ0**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.13 UART interface

The baseboard provides four serial transceivers.

Figure A-20 shows the pin numbering for the 9-pin D-type male connector used on the baseboard and Table A-23 lists the signal assignment for the connectors.

The pinout shown in Figure A-20 is configured as a *Data Communications Equipment* (DCE) device.

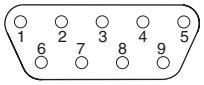

Figure A-20 Serial connector

Table A-23 Serial plug signal assignment

Pin	UART0 J40A (top)	UART1 J40B (bottom)	UART2 J39A (top)	UART3 J39B (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_DTRa**
5	**SER0_GND**	**SER1_GND**	**SER2_GND**	**SER3_GND**
6	**SER0_DSR**	**SER1_DSR**[a]	**SER2_DSR**[a]	**SER3_DSR**[a]
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.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.14 USB interface

USB2 and USB3 provide USB host interfaces and connect through the type A connector J7. USB1 provides an OTG interface and connects through the OTG connector J6.

Note

For a full description of the USB signals refer to the data sheet for the Philips ISP1761.

Figure A-21 shows the USB connectors.

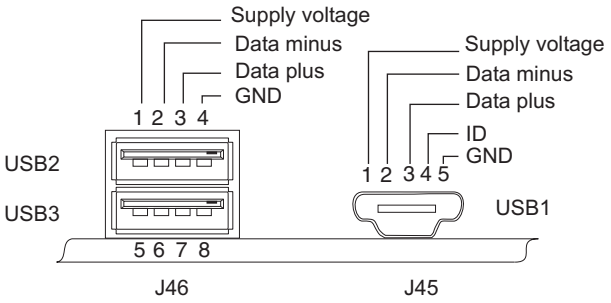

Figure A-21 USB interfaces

Note

There is also a dedicated USB debug port (see *USB debug port* on page A-31).

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

A.15 VGA display interface

The VGA connector (J5) is shown in Figure A-22. The connector signals are listed in Table A-24. A Digital to Analog Converter (DAC) converts the digital CLCD data and synchronization signals into the analogue VGA signals.

Table A-24 VGA connector signals

Pin	Description
1	**RED**
2	**GREEN**
3	**BLUE**
4	NC
5	**GND**
6	**GND**
7	**GND**
8	**GND**
9	NC
10	**GND**
11	NC
12	NC
13	**HSYNC**
14	**VSYNC**
15	NC

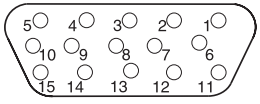

Figure A-22 VGA connector J5

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Appendix B
Specifications

This appendix contains the specification for the baseboard. It contains the following sections:

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

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.

Note

The DC IN voltage passes through a FET that disconnects the supply if the voltage is out of the normal operating range.

Table B-1 baseboard electrical characteristics

Symbol	Description	Min.	Max.	Unit
DC IN	DC input voltage	11.4	12.6	V
12V	Supply voltage from terminal or PCI	11.4	12.6	V
3V3	Supply voltage from attached tile (interface signals)	3.1	3.5	V
5V	Supply voltage	4.75	5.25	V
V_{IH}	High-level input voltage	2.0	3.6	V
V_{IL}	Low-level input voltage	0	0.8	V
V_{OH}	High-level output voltage	2.4	-	V
V_{OL}	Low-level output voltage	-	0.4	V
C_{IN}	Capacitance on any input pin	-	20	pF

Note

The HRDX and HRDY headers signals have the interface voltage set by external voltage reference determined by the tile connected (see *Power supply* on page 3-28). The signals are, by default, LVCMOS levels (1.2V–3.3V).

B.1.2 Current requirements

This section lists the typical current requirements of the baseboard.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table B-2 lists the typical current requirements at room temperature and nominal voltage powered from the DC IN connector. These measurements include the current drawn by the JTAG interface, approximately 160mA at 3.3V. The actual operating current might be different for different versions of Core Tile or for Logic Tile FPGA images.

The maximum load on the external 12V supply is 43W.

Table B-2 Current requirements (from 12V DC IN)

System	Total power	DC IN (average)
baseboard and one Core Tile	22W	1.8A
baseboard, one Core Tile, and two LT-XC2V6000 tiles	31W	2.6A
baseboard, one Core Tile, and two LT-XC2V8000 tiles	32W	2.7A
baseboard, one Core Tile, and four LT-XC2V8000 tiles	43W	3.6A
baseboard, one Core Tile, and 3.8” CLCD	32W	2.7A
baseboard, one Core Tile, and 8.4” CLCD	38W	3.2A

Powered from J27 or PCI bus

Table B-3 lists the current requirements on the individual baseboard supplies.

If the board is powered from terminal connector J27 or the PCI connector, the maximum loading on any supply is limited to 6A.

Table B-3 Typical current loading on baseboard supplies

System	3.3V typical	5V typical	12V typical
baseboard	1.0A	1.0A	<0.1A
Core Tile	0.5A	2.0A	-
LT-XC2V8000	0.5A	0.8A	-
8.4” CLCD	0.1A	0.1A	0.5A

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

B.2 Clock frequency restrictions

The maximum clock frequencies that can be used for reliable operation depend on the Core Tiles or Logic Tiles fitted.

Caution

The ICS307 programmable oscillators OSC0, OSC1, OSC2, OSC3, and OSC4 can be programmed to deliver very high clock signals (200MHZ). The only Core Tile clock input that can function at or near this frequency is **PLLCLKEXT** (that is, an external clock input to the Core Tile with the PLL bypassed).

Also, 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 ICS web site.

See the application note for your system configuration for timing details.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

B.3 Mechanical details

Figure B-1 shows the mechanical outline of the baseboard and the location of the tile headers. Figure B-2 on page B-6 shows the location of the mounting holes. See *Tile header connectors* on page A-35 for details on header pin numbering.

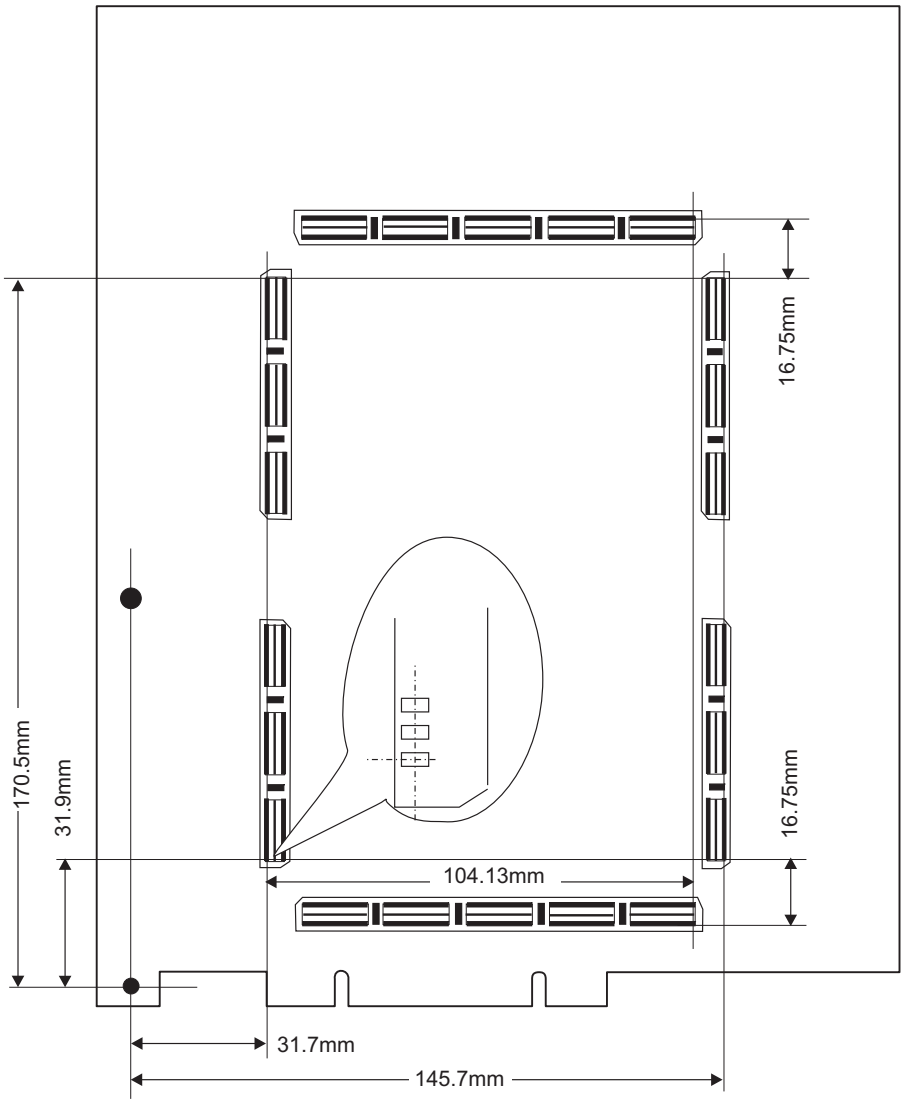

Figure B-1 Baseboard mechanical details

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

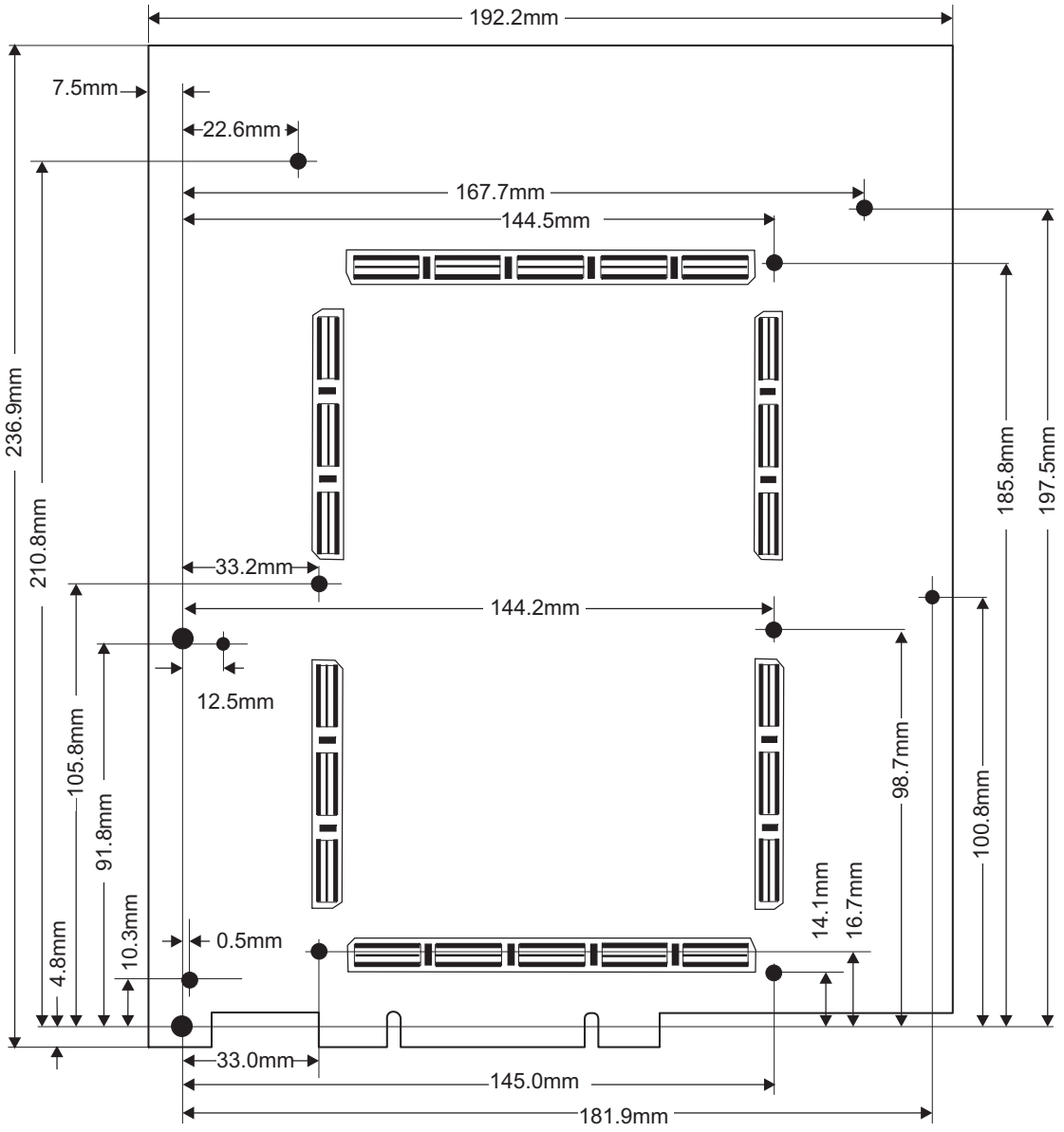

Figure B-2 Baseboard mounting holes

For more information on hole locations, see the Gerber drill file supplied on the CD.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Appendix C
LCD Kits

This appendix describes the external CLCD adaptor board and displays that are available as LCD kits (for example, the LCD38-BD-0193A kit). It contains the following sections:

Note

The CLCD adaptor board is typically supplied with the baseboard as standard. ARM Ltd. might change the type or size of displays distributed as LCD kits. For details of the current LCD kits that are available, contact your distributor or see the ARM web site.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

C.1 About the CLCD display and adaptor board

The CLCD interface board provides multiple sockets for different types of CLCD displays and touchscreens. It connects to the baseboard by a single cable.

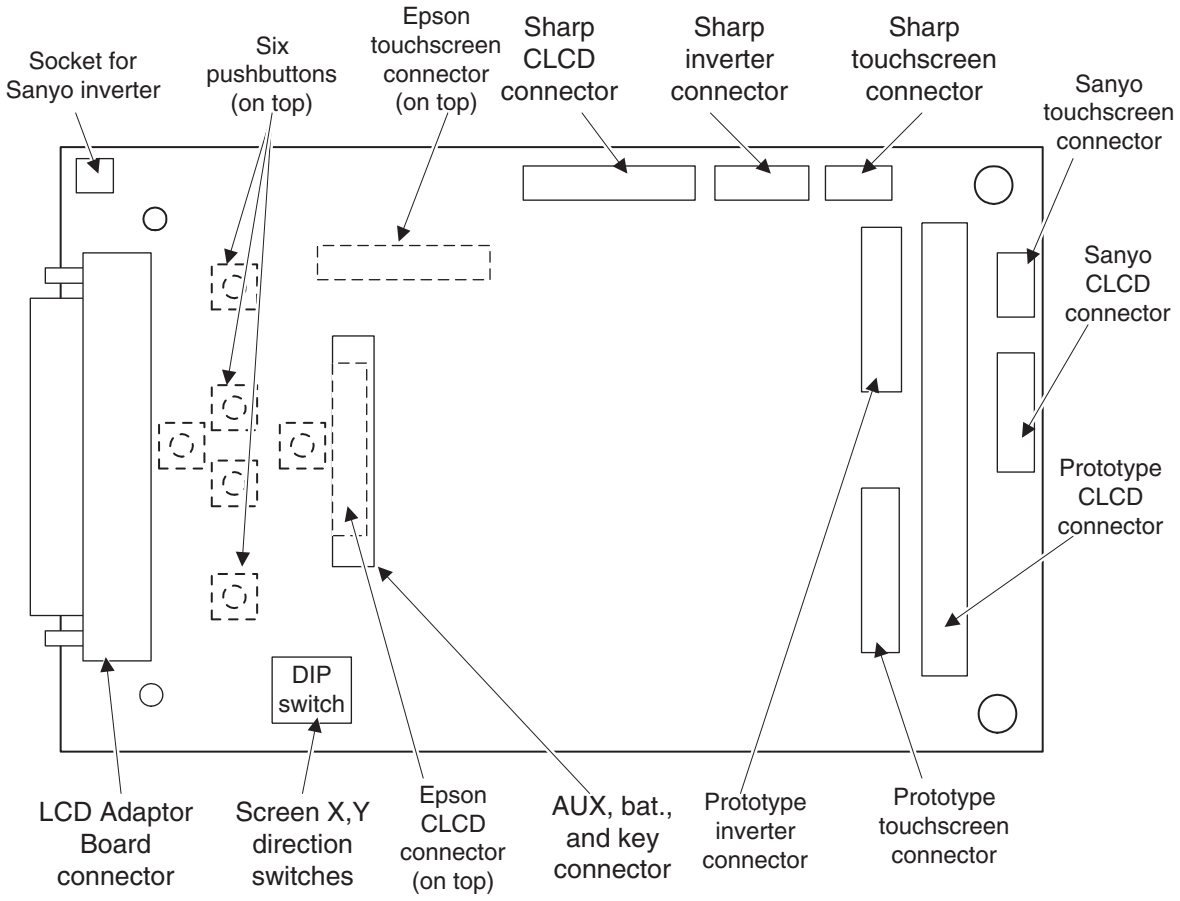

Figure C-1 CLCD adaptor board connectors (bottom view)

The design of the interface board enables you to use to choose the CLCD display that is appropriate for your application. The CLCD displays and touchscreens currently available for the baseboard are:

LCD22-BD-0192A

Epson 2.2 inch 176x220 pixel color TFT with LED backlight.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

LCD38-BD-0193A

Sanyo 3.8 inch QVGA color TFT with touchscreen and fluorescent backlight

LCD84-BD-0194A

Sharp 8.4 inch VGA color TFT with touchscreen and fluorescent backlight

Six push button switches are mounted on the interface board below the 2.2 or 3.8 inch display. The state of the switches can be read from the touchscreen controller interface.

The touchscreen interface on the CLCD interface board is described in *Touchscreen controller interface* on page C-11. The selftest program supplied on the CD reads the position of a pen on the touchscreen and displays it on the CLCD or VGA display connected to the board.

The 2.2 and 3.8 inch CLCD displays and the adaptor board are mounted in a small enclosure as shown in Figure C-2.

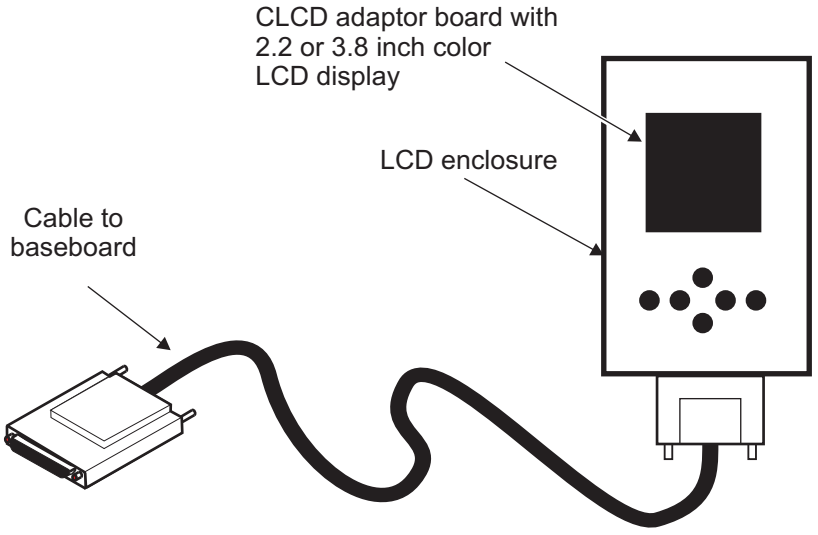

Figure C-2 Small CLCD enclosure

The 8.4 inch display is mounted into a large enclosure that has two connectors: one for a keypad and one for the baseboard. (See Figure C-3 on page C-4.)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

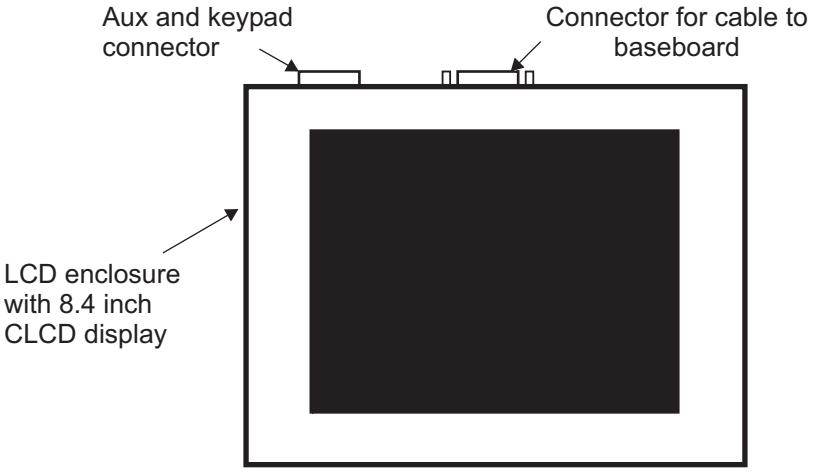

Figure C-3 Large CLCD enclosure

The 2.2 and 3.8 inch CLCD displays are mounted on the top side of the adaptor board as shown in Figure C-4 on page C-5.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

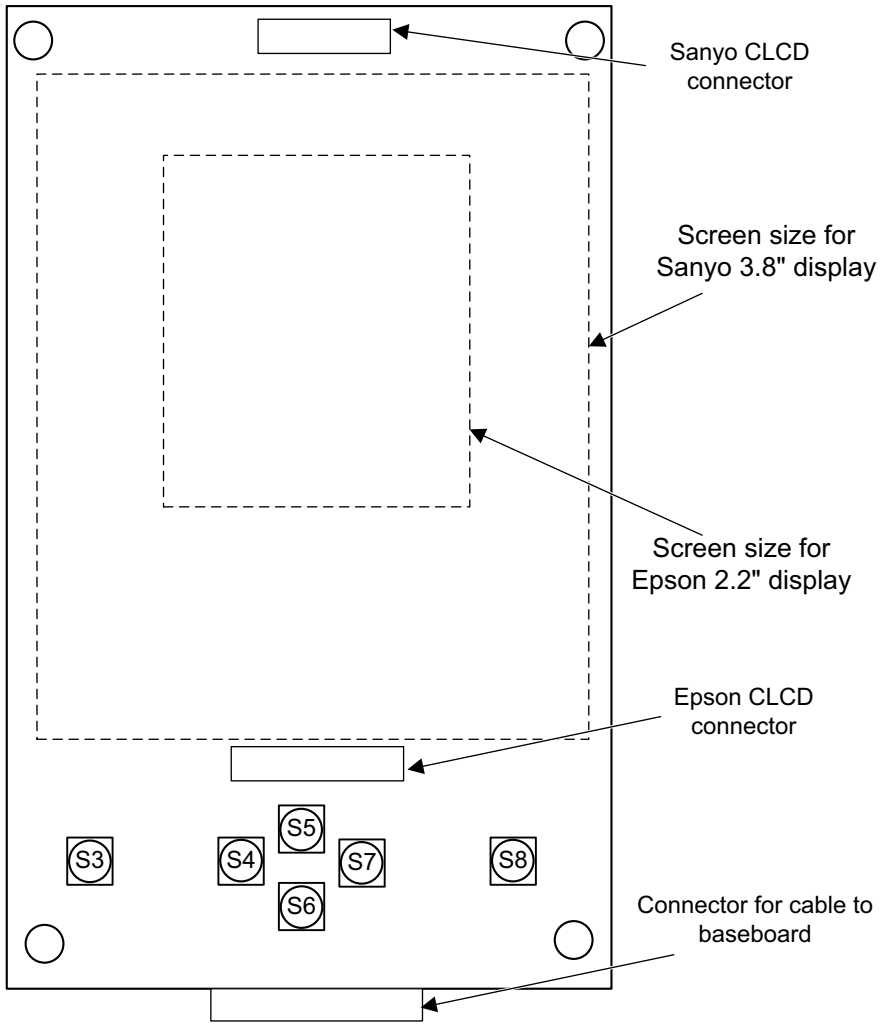

Figure C-4 Displays mounted directly onto top of adaptor board.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

C.2 Installing the CLCD display

To install the CLCD display:

1. Connect one end of the CLCD expansion cable to the CLCD adaptor board.
2. Connect the other end of the cable to the baseboard CLCD expansion connector on the enclosure.
3. If required, program the CLCD control registers SYS_CLCD and SYS_CLCDSER to sequence the power to the LCD display and specify the bit format. See the *CLCDC interface* on page 3-51.

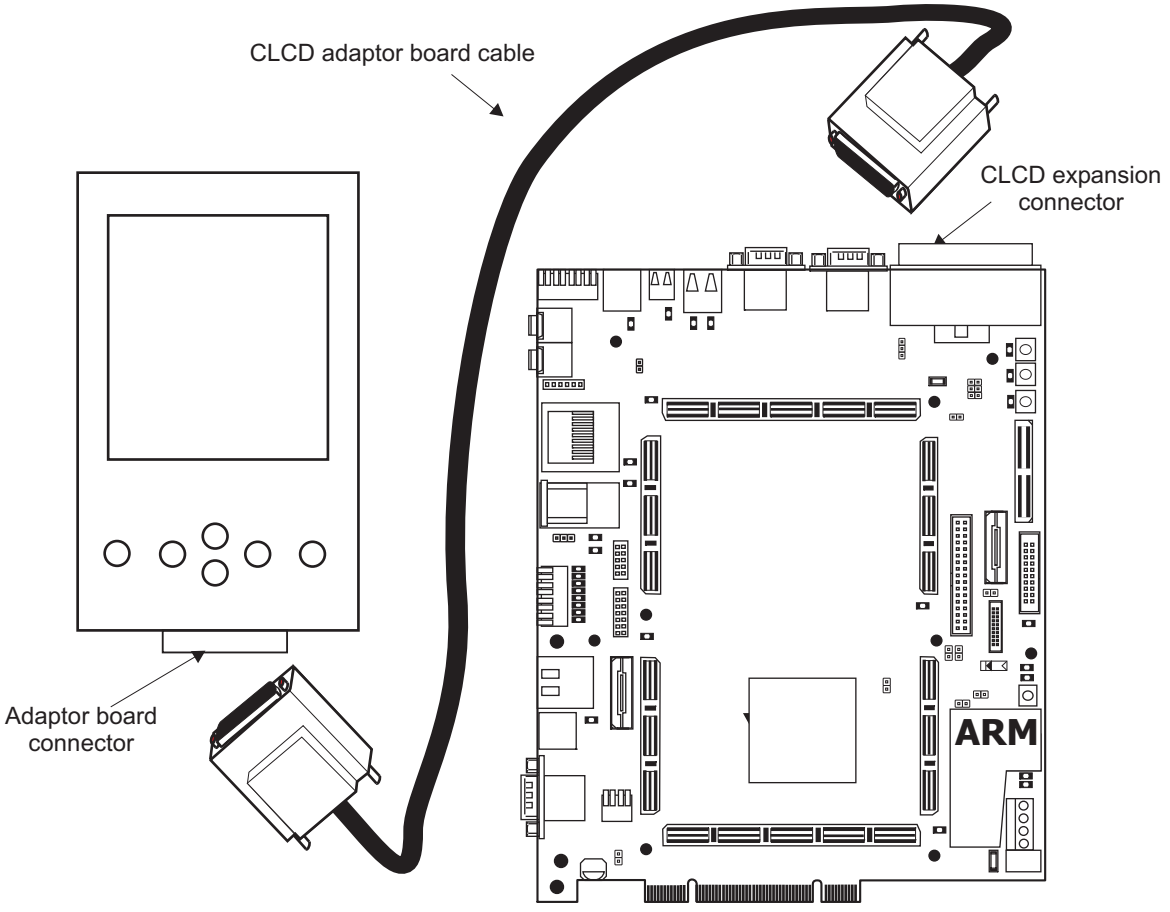

Figure C-5 CLCD adaptor board connection

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

C.2.1 Configuration

The CLCD adaptor board contains factory-installed links that identify the type of display. The display matching the identification links settings are listed in Table C-1. The value of the bits **CLCDID[4:0]** in the SYS_CLCD register can be read from software to determine the display in use with the board.

Table C-1 Displays available with adaptor board

LCD_ID[4:0]	Manufacturer	Backlight inverter	Touchscreen	Display
b00000	Sanyo TM38QV67A02A	TDK CXA-0341	Part of display	3.8 inch QVGA Color TFT
b00001	Sharp LQ084V1DG21	TDK CXA-L0612VJL	DynaPro/3M 95643	8.4 inch VGA color TFT
b00010	Epson L2F50113T00	LED backlight	None	2.2 inch 176x220 Color TFT
b01111	No display fitted	-	-	-

C.2.2 LCD power control

The LCD adaptor board accommodates a wide range of LCDs. Displays can require from 1 to 4 power supplies that can either be turned on/off simultaneously or need to be switched on/off in a certain order. System control register SYS_CLCD and the CLCD PrimeCell system register control power switching. The voltage supplies on the board are:

Vin This is permanently on and is not switched. This provides power to the board (nominal 12V) for the backlight converter.

1V8 This supply is permanently on. It is generated from 5V.

SWITCHED_FIXED

The supply is generated from the 1.8V, 3.3V or 5V supply. It can be enabled by **PWR3V5VSWITCH** in SYS_CLCD or permanently enabled by link 13.

SWITCHED_CLPWR

This supply is generated from 5V. It can be enabled by the **CLPOWER** signal in the CLCD PrimeCell control register or permanently enabled by link 15.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

SWITCHED_VDD_NEG

This –5V to –28V supply is generated from 5V. It can be enabled by **VDDNEGSWITCH** in SYS_CLCD or permanently enabled by link 14.

SWITCHED_VDD_POS

This 11V to 28V supply is generated from 5V. It can be controlled by the touchscreen D/A converter or manually with a pot. It can be enabled by **VDDPOSSWITCH** in SYS_CLCD or permanently enabled by link 11. This supply is used to generate the STN bias voltage.

LCD_IO_VDD and Buffer I/O voltage

This is the voltage to the interface logic on the adaptor board and the display. Link 16 selects the adaptor board interface level as **CLPWR** or **FIXED**. Link 3 selects the display interface level as **SWITCHED_FIXED** or **SWITCHED_CLPWR**.

Caution

Link 3 and link 16 must be set to use the same power source.

INV_IO This is the voltage to the interface logic on the prototype board. Link 2 selects the level as 5V or 3.3V.

Note

The I/O signals to the CLCD adaptor board pass through tri-state buffers. The buffers must be powered from the same IO voltage as that required by the CLCD. This enables the translation of the IO signals from the 3V3 signal levels present on the baseboard. The buffers are enabled by **LCDIOON** in SYS_CLCD.

Figure C-6 on page C-10 shows the block diagram of the adaptor board power-control circuitry.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table C-2 lists the power configuration for the three displays. For additional information on configuring the CLCD displays, see the selftest code provided on the CD.

Table C-2 Power configuration

Voltage control	Epson 2.2"	Sanyo 3.8"	Sharp 8.4"
Buffer IO	SWITCHED_FIXED	CLPOWER	CLPOWER
SWITCHED_VDD_POS	Software control	15V	Software control
SWITCHED_VDD_NEG	–10V	–10V	–10V
CLPOWER	2.85V	3.3V	3.3V
FIXED_SWITCH	1.8V	5V	5V
INV_IO	5V	3.3V	5V
Buffer enabled (software control)	Always on	Always on	Set from CLPOWER register

Caution

The links for power control are set during manufacture. Do not modify the links unless you are producing a new custom display board.

Use connector J4 to supply power to an inverter for a backlight. The backlight pins **VIN** are provide a nominal 12V supply. The backlight inverter must consume less than 5W. The I/O voltage level **INV_IO** is also present on J4. **INV_IO** can be link selected to be 5V or 3.3V.

In addition to voltage and ground pins, the connector also supplies the brightness adjustment voltage (0 to **INV_IO** voltage). The brightness is adjusted by a variable resistor, VR4, located near J4.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

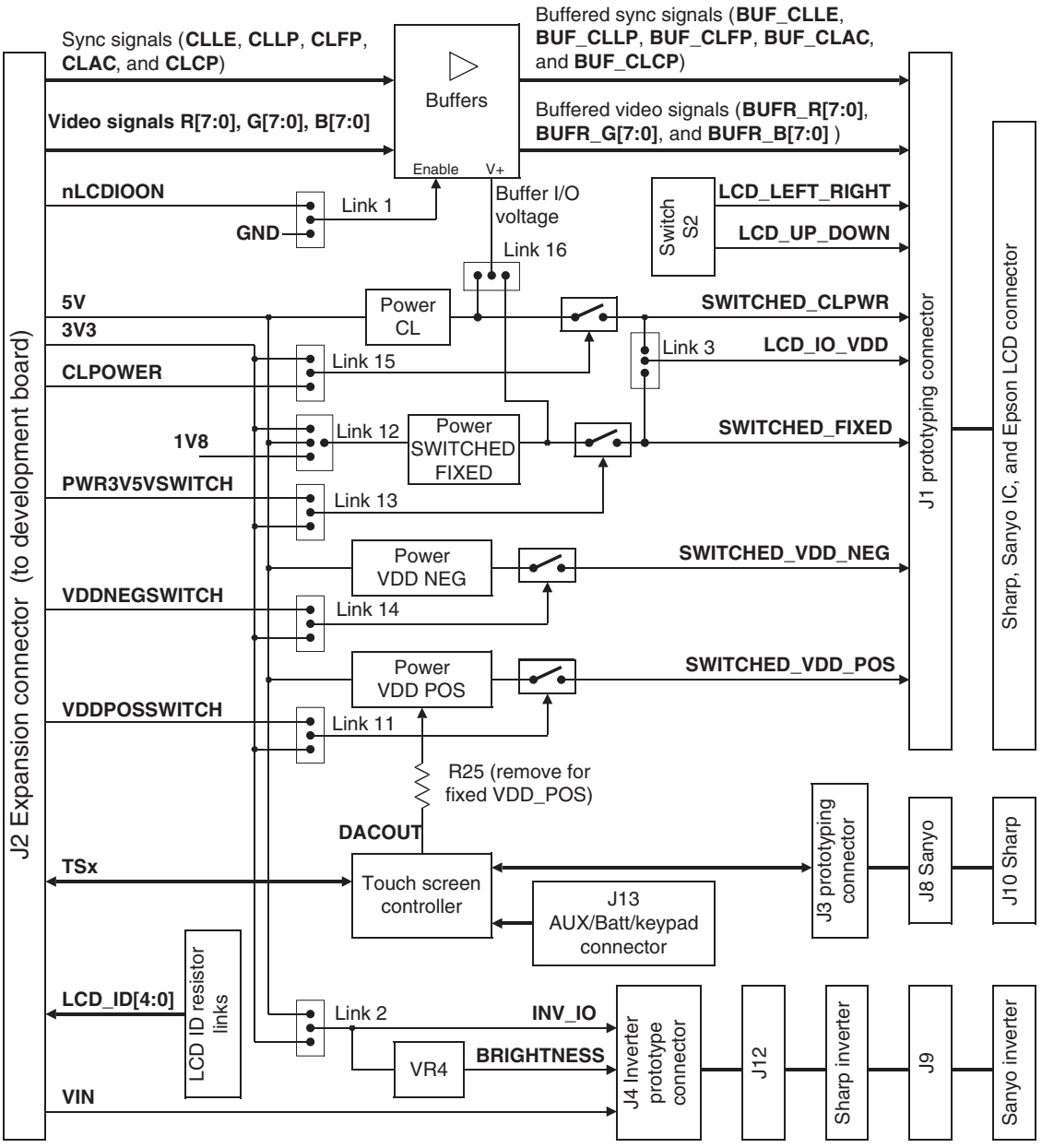

Figure C-6 CLCD buffer and power supply control links

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

C.3 Touchscreen controller interface

The touchscreen interface is designed to connect to a four-wire resistive touchscreen. It is driven by the TouchScreen controller TSC2200 and described in:

- *Touchscreen interface architecture*
- *Touchscreen controller programmer's interface* on page C-13.

The `Selftest` program supplied on the CD demonstrates how to communicate with the touchscreen controller. The program uses the interface code to plot the touchscreen X and Y coordinates on the LCD or VGA screen.

Connectors J8 and J10 are used for the standard touchscreens provided with the CLCD assembly. Prototyping connector J3 enables the use of other resistive touchscreens, see *Touchscreen prototyping connector* on page C-17.

C.3.1 Touchscreen interface architecture

Figure C-7 on page C-12 shows the touchscreen interface. Table C-3 lists the touchscreen control signals. The signals to the touchscreen are routed to connector J13.

Table C-3 Touchscreen host interface signal assignment

Signal name	Description
TSMOSI	Serial data input to controller
TS_nSS	Chip select
TSSCLK	Clock input
TSMISO	Data output
TSnDAV	Data available
TSnPENIRQ	Pen down interrupt
TSnKPADIRQ	Keypad interrupt
VBAT[2:1] and **AUX[2:1]**	External voltage to analog to digital converter in touchscreen controller. These are reserved for expansion for external devices connected to the AD and keypad connector J13.
R[4:1] and **C[4:1]**	Row and column scan signals for a keyboard. The expansion board switches S3 to S8 currently use eight positions on the scan matrix, but additional switches can be fitted using the AD and keypad connector J13.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

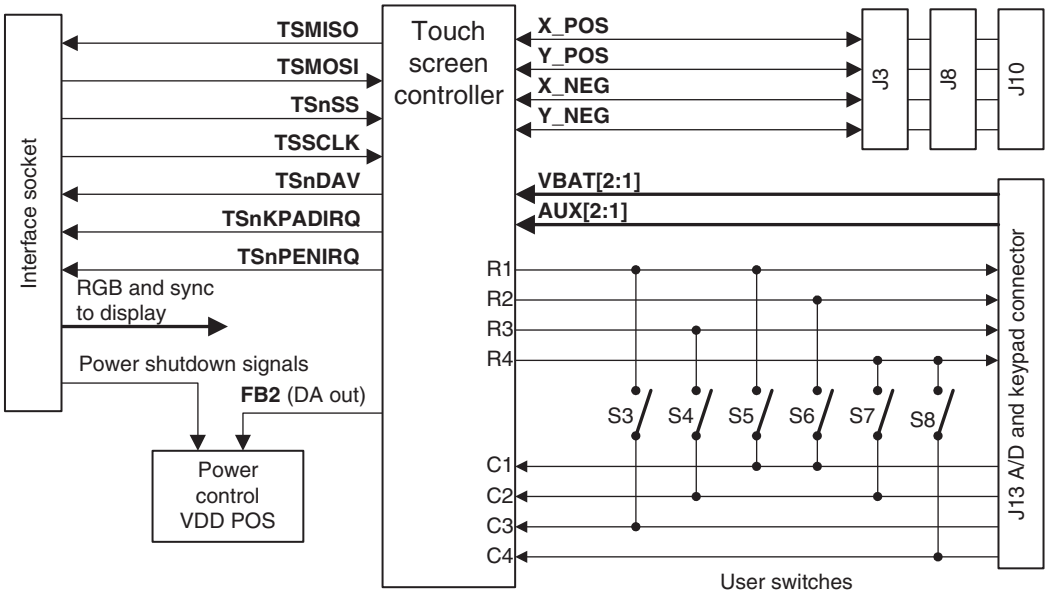

Figure C-7 Touchscreen and keypad interface

The connection between the resistive elements of the touchscreen and J3, J8, or J10 is shown in Figure C-8. When the pen is down, the two resistive elements touch and form a four-resistor network. Measuring the voltages at the two dividers indicates the X and Y positions.

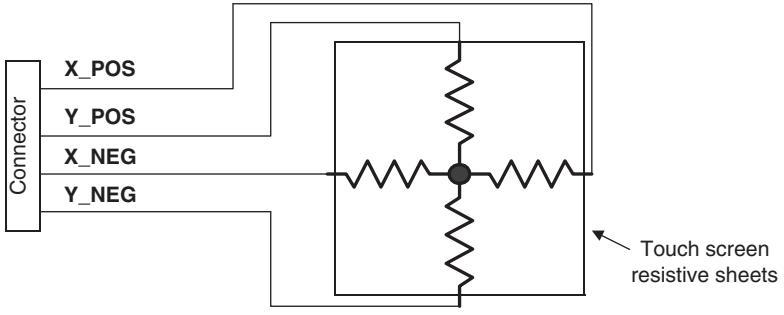

Figure C-8 Touchscreen resistive elements

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

C.3.2 Touchscreen controller programmer's interface

The LCD *Touch Screen Controller Interface* (TSCI) is based on a TSC2200 PDA analogue interface circuit. Use the baseboard SSP interface to configure and read the touch screen. For information on the touch screen registers, see the TSC2200 data sheet.

The TSC2200 also incorporates a sixteen key keypad interface and two 12bit analogue inputs that are available through the LCD expansion header J13. With the 3.8 inch Sanyo and 2.2 inch Epson build options, six keypad push buttons are mounted on the LCD board.

SSP and TSCI Configuration

The SSP interface is controlled by the SSP PrimeCell and the SSP TSCI chip select is enabled through SYS_CLCD **TSnSS** signal. Configuration of the SSP to TSCI interface requires the data format, phase, size, and clock to be set correctly. Example configuration code is given in the selftest (TSCI) software on the CD, a code fragment from this is shown in Example C-1.

Example C-1 SSP to TSCI interface setup

```
// Set serial clock rate (/3), Phase (SPH), Format (MOT), data size (16bit)
*SSPCR0 = SSPCR0_SCR_DFLT | SSPCR0_SPH | SSPCR0_FRF_MOT | SSPCR0_DSS_16;
// Clock prescale register (/8), with SCR gives 0.78MHz SCLK: 24MHz / 8*(1+3)
*SSPCPSR = SSPCPSR_DFLT;
// Enable serial port operation
*SSPCR1 = SSPCR1_SSE;
```

The TSC2200 TSCI controller registers must be configured through the SSP interface to enable correct touch screen operation.

After the TSCI is configured, conversion of touch screen X/Y values is fully automated by the TSCI controller and the application code simply reads the converted values. Use either the pen down flag in the touch screen controller interface or SIC interrupt 8 to detect the current pen state. The pseudo code in Example C-2 on page C-14 shows the sequence for configuring and reading the TSCI interface.

Read and write functions are used in the selftest code to transfer data to and from the TSCI registers TSCI_RTSC and TSCI_WTSC. The selftest example configures the TSCI for 12bit operation and 16 data averages with minimum precharge and sense times. This gives high accuracy and fast reading of the current pen position.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Example C-2 Configuring and reading the TSCI interface

```
Configure the SSP interface
Configure the TSCI registers
Enable the touch screen pendown interrupt (on SIC)
(other general setup code here)
On touch screen pendown interrupts
    (touch screen interrupt handler)
    Enable the touch screen event timer (TIMER 1-4) for 2mS intervals
    (other pendown handling code here)
On touch screen timer events
   (touch screen reading code here)
   If (pendown flag (PSM) is cleared)
        Disable the touch screen event timer
        Clear and re-enable the touch screen interrupt
   Else
        Read the pen X/Y values
        Draw the pen position on the screen
```

Note

The selftest example provided on the CD uses a simple polled system to determine pen down and timer events.

The pseudo code in Example C-2 is recommended for OS ports as they typically require interrupt-driven device drivers.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

C.4 Connectors

This section describes the connectors present on the CLCD adaptor board. For details of the connectors present on the baseboard, see Appendix A *Signal Descriptions*.

C.4.1 Interface connector

The signals on the CLCD interface connector J2 are shown in Table C-4.

Table C-4 CLCD interface connector J2

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	**B0**	2	**B2**	35	**B1**	36	**B3**
3	**B4**	4	**B6**	37	**B5**	38	**B7**
5	**G0**	6	**G2**	39	**G1**	40	**G3**
7	**G4**	8	**G6**	41	**G5**	42	**G7**
9	**R0**	10	**R2**	43	**R1**	44	**R3**
11	**R4**	12	**R6**	45	**R5**	46	**R7**
13	**CLLE**	14	**CLAC**	47	**GND**	48	**GND**
15	**CLCP**	16	**CLLP**	49	**GND**	50	**GND**
17	**CLFP**	18	**TSnKPADIRQ**	51	**GND**	52	**GND**
19	**TSnPENIRQ**	20	**TSnDAV**	53	**GND**	54	**LCDID0**
21	**TSSCLK**	22	**TSnSS**	55	**LCDID1**	56	**LCDID2**
23	**TSMISO**	24	**TSMOSI**	57	**LCDID3**	58	**LCDID4**
25	**LCDXWR**	26	**LCDSD0**	59	**GND**	60	**GND**
27	**LCDXRD**	28	**LCDXCS**	61	**GND**	62	**3V3**
29	**LCDDATAnCOMM**	30	**LCDSD0DIR**	63	**3V3**	64	**5V**
31	**CLPOWER**	32	**nLCDIOON**	65	**5V**	66	**VIN**
33	**PWRFIXEDSWITCH**	34	**VDDPOSSWITCH**	67	**VIN**	68	**VDDNEGSWITCH**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

C.4.2 LCD prototyping connector

The signals on the LCD prototyping connector J1 are shown in Table C-5.

Table C-5 LCD prototyping connector J1

Signal	Pin	Pin	Signal
BUF_CLLP	1	2	**BUF_G2**
GND	3	4	**BUF_G3**
CLCP0	5	6	**GND**
GND	7	8	**BUF_G4**
BUF_CLFP	9	10	**BUF_G5**
GND	11	12	**GND**
BUF_CLAC	13	14	**BUF_G6**
GND	15	16	**BUF_G7**
BUF_CLLE	17	18	**GND**
GND	19	20	**BUF_R0**
BUF_B0	21	22	**BUF_R1**
BUF_B1	23	24	**GND**
GND	25	26	**BUF_R2**
BUF_B2	27	28	**BUF_R3**
BUF_B3	29	30	**GND**
GND	31	32	**BUF_R4**
BUF_B4	33	34	**BUF_R5**
BUF_B5	35	36	**GND**
GND	37	38	**BUF_R6**
BUF_B6	39	40	**BUF_R7**
BUF_B7	41	42	**SWITCHED_FIXED**
GND	43	44	**LCD LEFT_RIGHT**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table C-5 LCD prototyping connector J1 (continued)

Signal	Pin	Pin	Signal
BUF_G0	45	46	**LCD UP_DOWN**
BUF_G1	47	48	**SWITCHED_VDD_POS**
SWITCHED_CLPWR	49	50	**SWITCHED_VDD_NEG**

C.4.3 Touchscreen prototyping connector

The signals on the touchscreen prototyping connector J3 are shown in Table C-6.

Table C-6 Touchscreen prototyping connector J3

Signal	Pin	Pin	Signal
GND	1	2	**GND**
X_POS	3	4	**GND**
Y_NEG	5	6	**GND**
X_NEG	7	8	**GND**
Y_POS	9	10	**GND**

C.4.4 Inverter prototyping connector

The signals on the inverter prototyping connector J4 are shown in Table C-7.

Table C-7 Inverter prototyping connector J4

Signal	Pin	Pin	Signal
VIN	1	2	**VIN**
VIN	3	4	**VIN**
GND	5	6	**GND**
BRIGHTNESS	7	8	**GND**
GND	9	10	**INV_IO**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

C.4.5 A/D and keypad connector

The signals on the connector J13 are shown in Table C-7 on page C-17.

This connector enables the connection of an external keypad (**R[4:1]** are the keypad row scan output signals and **C[4:1]** are the column detect input signals). There are also connections to the analog to digital converter inputs on the CLCD adaptor board (**AUX[2:1]** and **VBAT[2:1]**).

Table C-8 A/D and keypad J13

Signal	Pin	Pin	Signal
3V3	1	20	**3V3**
AUX1	2	19	**GND**
AUX2	3	18	**GND**
VBAT1	4	17	**GND**
VBAT2	5	16	**GND**
R1	6	15	**C1**
R2	7	14	**C2**
R3	8	13	**C3**
R4	9	12	**C4**
GND	10	11	**GND**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

C.5 Mechanical layout

Figure C-9 shows the board layout.

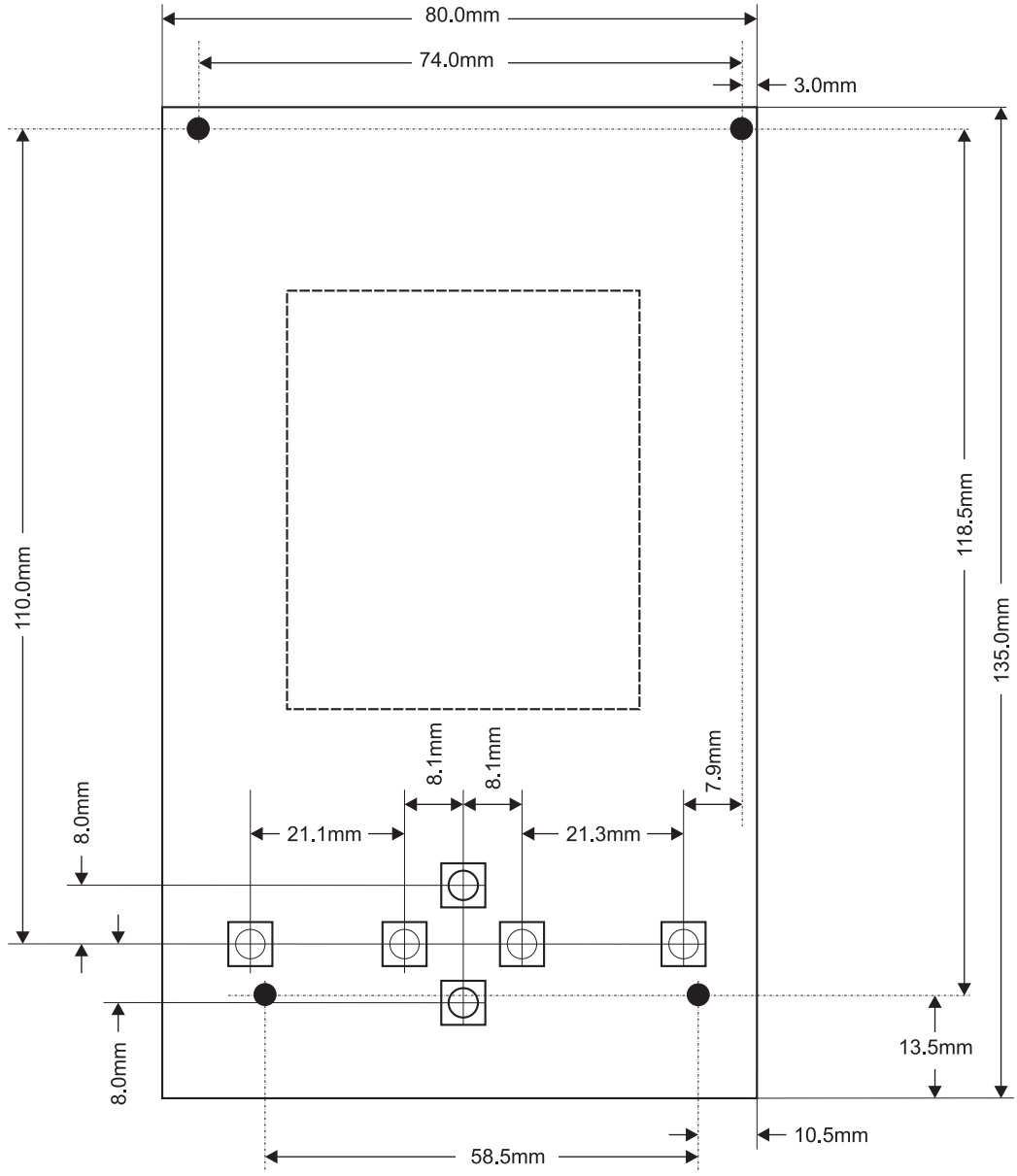

Figure C-9 CLCD adaptor board mechanical layout

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Appendix D
PCI Backplane and Enclosure

This appendix describes the PCI backplane and enclosure. It contains the following sections:

- *Connecting the baseboard to the PCI enclosure* on page D-2
- *Backplane hardware* on page D-8
- *Connectors* on page D-13.

For details on configuring the PCI controller and PCI expansion cards, see *PCI controller* on page 4-68.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

D.1 Connecting the baseboard to the PCI enclosure

This section describes how to configure the PCI backplane and connect the baseboard to the PCI enclosure.

To use the baseboard with the PCI backplane and enclosure:

1. Configure the baseboard by installing Core Tiles as described in *Setting up the baseboard* on page 2-2.

   Caution

   Do not connect power to the baseboard yet.

2. Connect a JTAG debugger (such as Multi-ICE) to the board, or use the USB debug port. See *Connecting JTAG debugging equipment* on page 2-13.

   Note

   The JTAG connection on the baseboard does not connect to the PCI backplane. Use the baseboard JTAG for debugging applications.

   There is also a JTAG socket on the PCI backplane. Only use this connector if you are reprogramming the PAL on the PCI backplane.

3. Set the configuration switches on the PCI backplane. See *Setting the backplane configuration switches* on page D-5.

4. If you are using an external display:

   - For VGA displays, connect the cable from the display to the VGA connector on the baseboard.

     Note

     If you are using a VGA card in the PCI bus, connect the VGA display to the VGA connector on the PCI card. You must provide the interface code for the PCI display card.

   - For CLCD displays, connect the CLCD expansion board cable to the baseboard and if necessary, connect the display interface cable from the expansion board to the CLCD display. See Appendix C *LCD Kits*.

5. Slide the baseboard into the PCI connector on the side of the enclosure. Figure D-1 on page D-4 illustrates an baseboard mounted in the PCI backplane in the supplied enclosure.

6. Apply power to the PCI enclosure.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Caution

Do not connect power to the screw terminals or to the power socket on the baseboard.

7. Execute the initialization code to setup the PCI address-mapping registers (see *PCI controller* on page 4-68)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

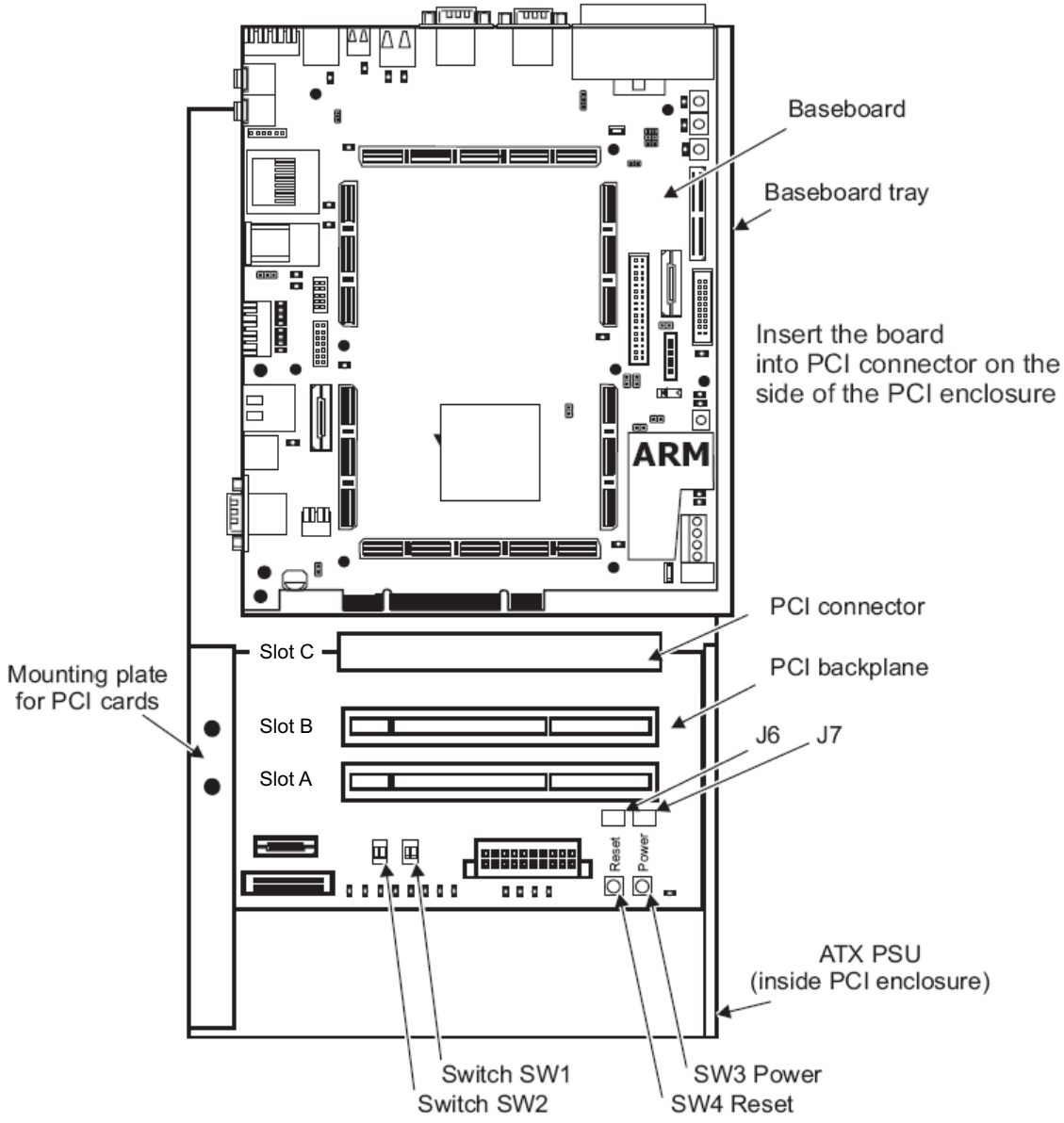

Figure D-1 Installing the platform board into the PCI enclosure

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Note

The PCI card INTA# is the interrupt output for a single-function device. INTB#, INTC# and INTD# are only used for multi-function devices.

Figure D-2 shows the interrupt routing. Signals are swapped around to enable simultaneous interrupts.

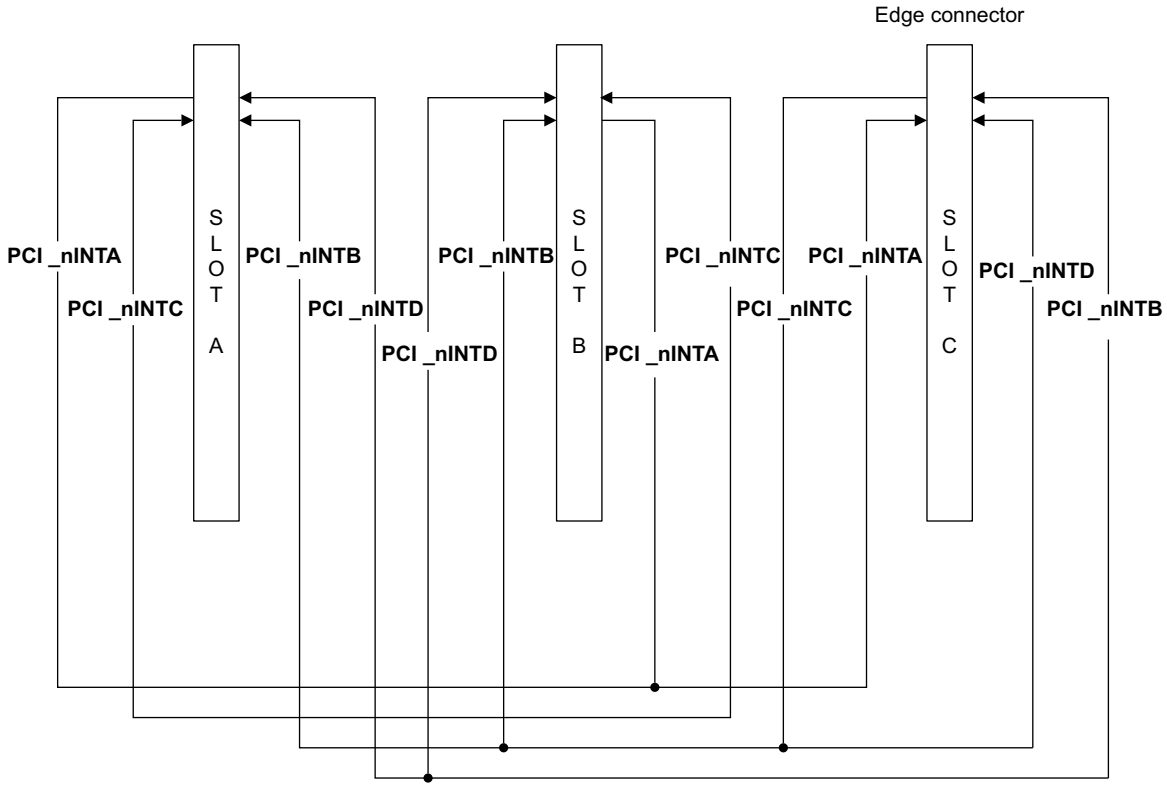

Figure D-2 PCI backplane interrupt routing

D.1.1 Setting the backplane configuration switches

There are four control switches on the PCI backplane board as shown in Figure D-4 on page D-10. The switches are arranged into two switch blocks, SW1 and SW2.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

SW1-1 and SW1-2 control clock rate:

SW1-1 SW1-1 positions are labeled **Manual** and **Auto** and select between manual or automatic clock selection:

- In **Manual** position, the clock is determined by the settings of the manual clock select switch SW1-2.
- In **Auto** position, the clock rate is determined by the capabilities of the PCI cards installed. The default rate is 33MHz. If however all PCI cards are capable of functioning at 66MHz, the clock rate will automatically be increased to 66MHz.

SW1-2 SW1-2 positions are labeled **Man1** and **Man2** and determine the clock rate when SW1-1 is in the **Manual** position. **Man1** selects low frequency (10MHz) and **Man2** selects high frequency (50MHz).

Switches SW2-1 and SW2-2 control the PCI backplane JTAG scan chain:

SW2-1 Switch positions for SW2-1 are labeled **omitPLD** and **incPLD** and omits or includes the PLD on the PCI backplane from the PCI scan chain.

SW2-2 Switch positions for SW2-2 are labeled **omitPCI** and **incPCI** and omit or include the PCI sockets in the PCI scan chain. If a PCI card is not present in a socket, the socket is bypassed by an automatic switch.

If a PCI card does not support JTAG, place a jumper across the **TDI** and **TDO** signals for that card or place insulating tape over the **nPRSNT** pins on the socket.

The JTAG scan chain on the baseboard does not extend to the PCI backplane.

There are also two connectors on the board that can be connected to external switches:

J6 This connector is paralleled with SW3 and permits control of the power from a front-panel switch.

J7 This connector is paralleled with SW4 and enables a front-panel switch to reset the PCI arbiter on the backplane and reset all of the PCI cards.

Note

Front panel switches are not provided as part of the PCI enclosure.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

D.1.2 Connecting two baseboard boards

Figure D-3 shows two baseboard boards and a VGA controller connected to the PCI backplane. The PCI controller in the top baseboard is operating as a PCI bus slave and the PCI controller in the bottom baseboard is operating as a PCI bus master. The VGA card is also operating as a slave.

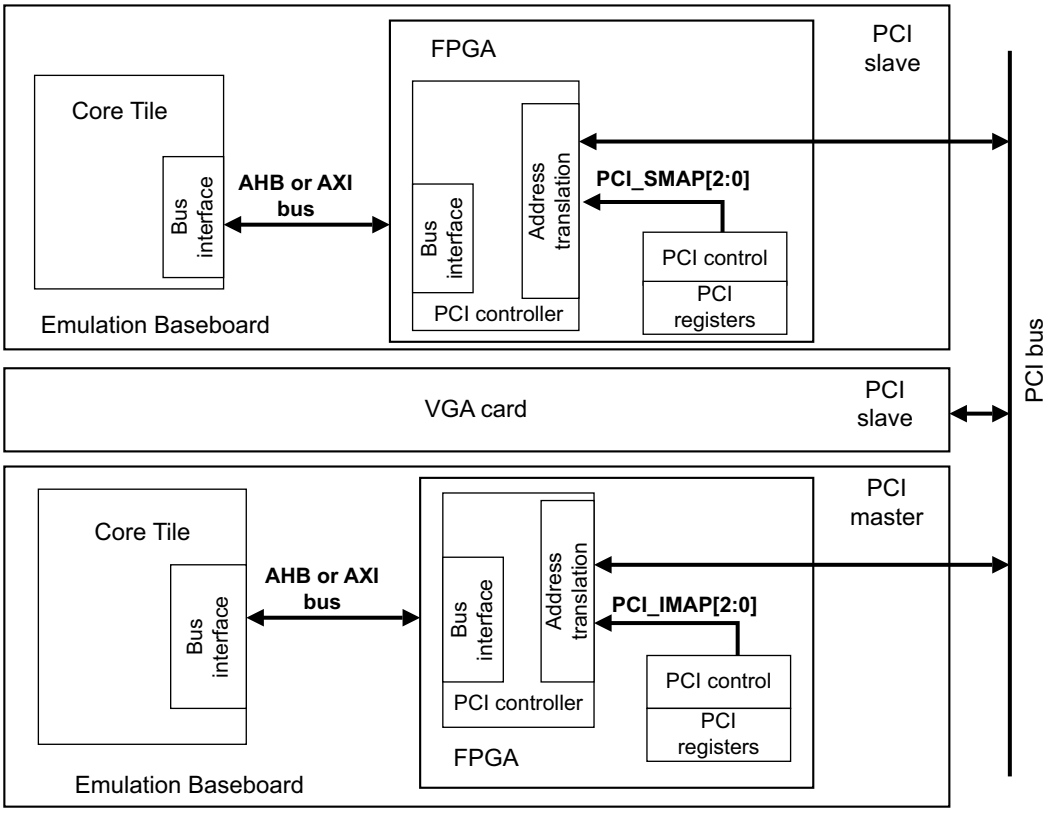

Figure D-3 Multiple boards on PCI bus

Note

You can only use 32-bit PCI expansion cards in the PCI enclosure.

You can, however, plug the baseboard into a PC that has a PCI motherboard with either 32 or 64-bit sockets.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

D.2 Backplane hardware

The mechanical layout for the PCI backplane is shown in Figure D-4 on page D-10.

The PCI backplane mechanical layout complies to PCI Specification v2.3 for a two slot short card system board.

The switches, indicators, and test points for the PCI backplane are listed in Table D-1, Table D-2 on page D-11, and Table D-3 on page D-11.

Table D-1 LED indicators

LED	Signal	Description
1	**MAN1nMAN2**	This LED illuminates to indicate **Man2** clock selection
2	**MANnAUTO**	This LED illuminates to indicate **Auto** clock selection
5	**CLK33ACTIVE**	If **Auto** clock selection is selected, this LED illuminates to indicate 33MHz bus speed. If manual clock selection of 10MHz is selected, this LED is illuminated.
6	**CLK66ACTIVE**	If **Auto** clock selection is selected, this LED illuminates to indicate 66MHz bus speed. If manual clock selection (of either 10MHz or 50MHz) is selected, this LED is illuminated.
7	**EXT**	If **Auto** clock selection is selected, this LED is reserved for use by manufacturing tests. If manual clock selection of 50MHz is selected, this LED is illuminated.
9	**PSON**	Power supply on/off indicator. The LED is illuminated when the unit is off (standby).
10	**3V3**	Power supply voltage present
11	**5V**	Power supply voltage present
12	**12V**	Power supply voltage present
13	**–12V**	Power supply voltage present

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table D-1 LED indicators (continued)

LED	Signal	Description
14	**PCI_nPRSNT1A** and **PCI_nPRSNT2A**	This LED illuminates to indicate PCI card present and enabled in slot A.
15	**PCI_nPRSNT1B** and **PCI_nPRSNT2B**	This LED illuminates to indicate PCI card present and enabled in slot B.
16	**PCI_nPRSNT1C** and **PCI_nPRSNT2C**	This LED illuminates to indicate PCI card present and enabled in slot C. (This is the slot for the baseboard.)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

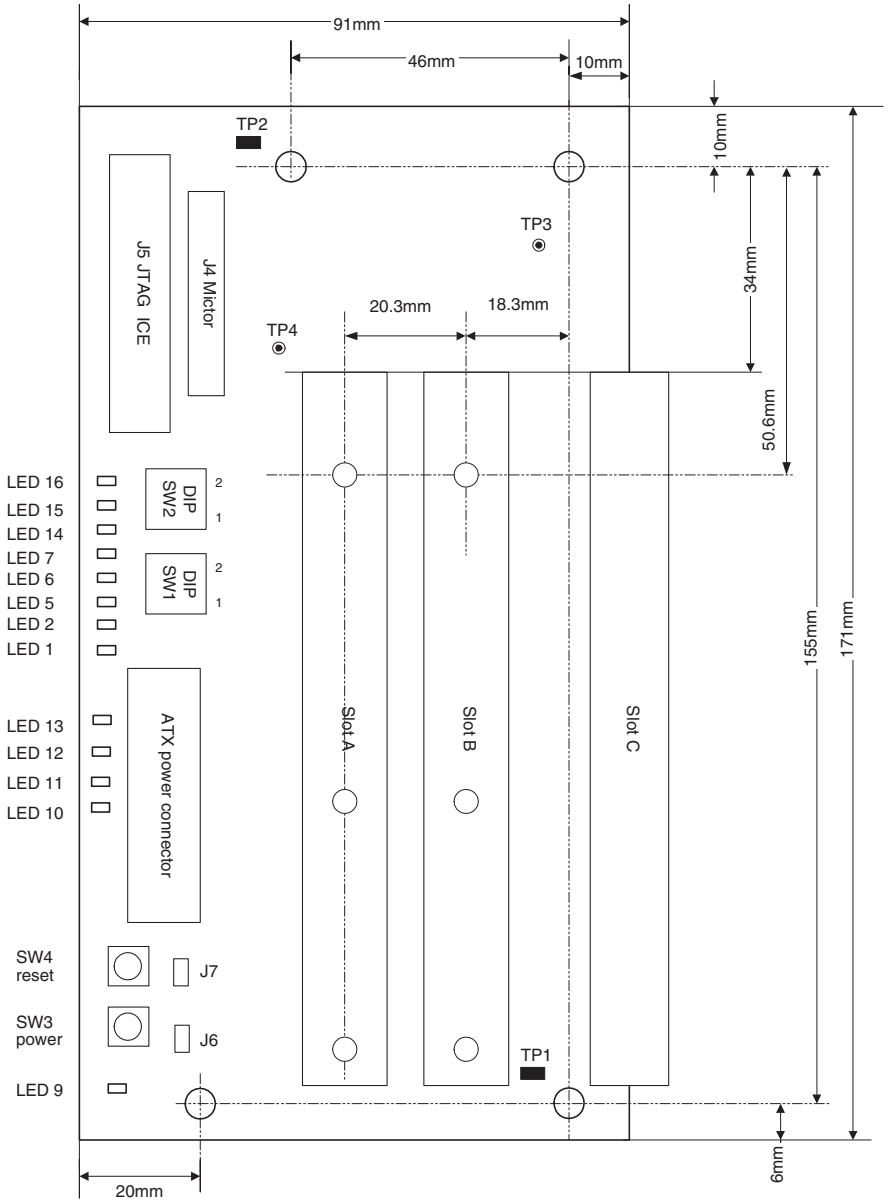

Figure D-4 PCI backplane

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table D-2 Configuration switches

Switch	Signal	Description
SW1-1	**MAN1nMAN2**	Determines the clock rate when SW1[1] is in the **Manual** position. See *Setting the backplane configuration switches* on page D-5.
SW1-2	**MANnAUTO**	Selects between manual (ON) or automatic (OFF) clock selection
SW2-1	**nINCPLD**	Omits (ON) or includes (OFF) the PLD in the JTAG scan chain.
SW2-2	**TESTnEN**	Omits (ON) or includes (OFF) the PCI sockets in the JTAG scan chain.

Table D-3 Power and reset switches

Switch	Signal	Description
SW3	**PSON**	Power on/off push button. Pressing the switch toggles the power between on and standby. SW3 signals are also connected to J6 and this enables the use of an external front-panel switch.
SW4	**SYSTEM_nRESET**	System reset push button. Pressing the switch generates a reset to the PCI arbiter on the backplane and all of the PCI cards. SW4 signals are also connected to J7 and this enables the use of an external front-panel switch.

Table D-4 Test points

Test point	Signal	Description
TP1	**GND**	Ground
TP2	**GND**	Ground
TP3	**TCLK**	JTAG clock (This signal is called **P_TCK** on the baseboard)
TP4	**PCI CLK**	PCI clock (This signal is called **P_CLK** on the baseboard)

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

D.2.1 JTAG signals

The JTAG signal flow is shown in Figure D-5.

Note

The JTAG chain on the PCI expansion board is independent of the JTAG chain on the baseboard.

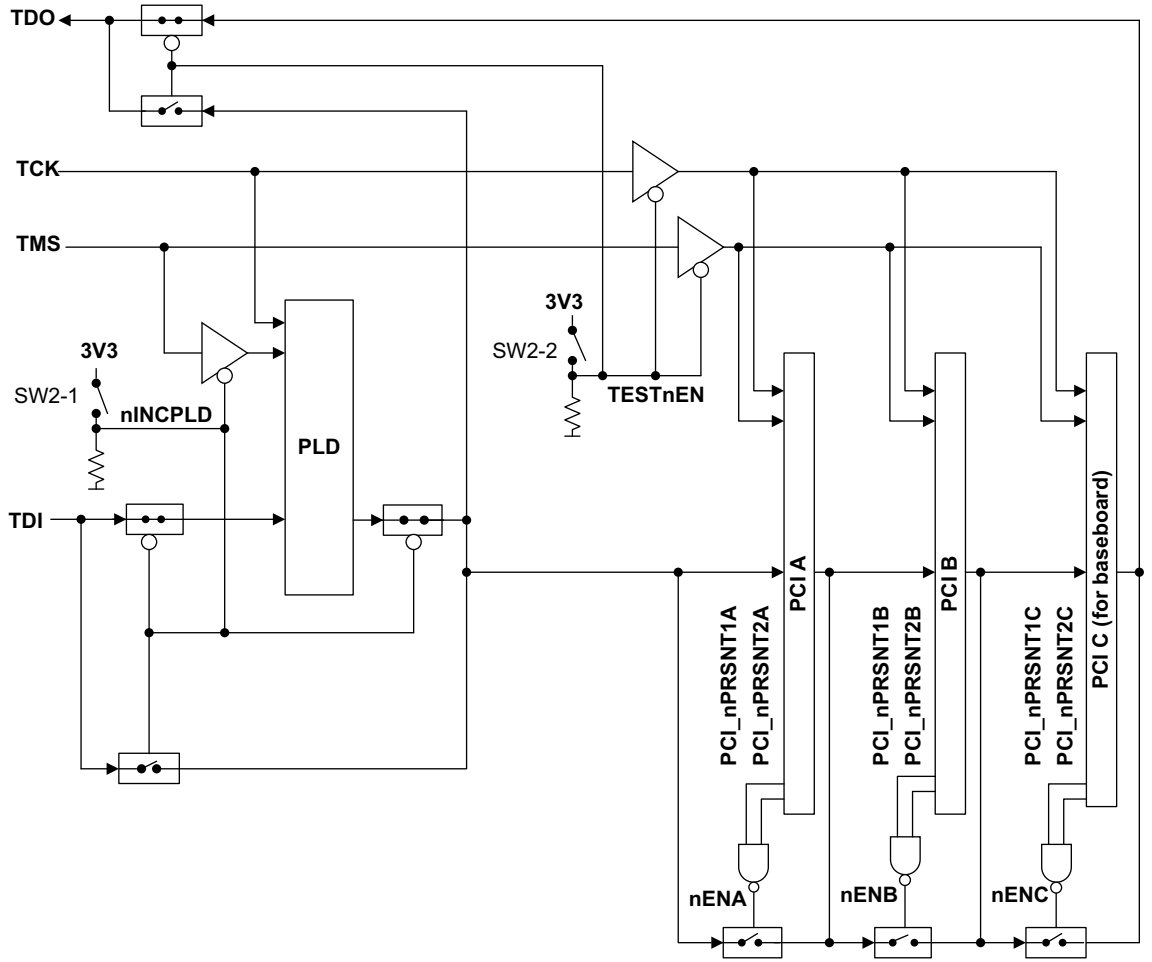

Figure D-5 JTAG signal flow on the PCI backplane

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

D.3 Connectors

This section describes the connectors present on the PCI backplane.

D.3.1 Power connector

The power connector is a standard ATX style connector as used in PCs. The pinout for the connector is listed in Table D-5.

Table D-5 ATX power connector

Signal	Pin	Pin	Signal
3V3	1	11	**3V3**
3V3	2	12	**–12V**
GND	3	13	**GND**
5V	4	14	**nPSON**
GND	5	15	**GND**
5V	6	16	**GND**
GND	7	17	**GND**
PWOK	8	18	NC
ATX5VSB	9	19	**5V**
12V	10	20	**5V**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

D.3.2 Logic analyzer connector

Figure D-6 and Table D-6 show the pinout of the Mictor connector J4. You can use this connector to monitor PCI signals on the backplane.

Note

Agilent (formerly HP) and Tektronix label these connectors differently, but the assignments of signals to physical pins is appropriate for both systems and pin 1 is always in the same place.

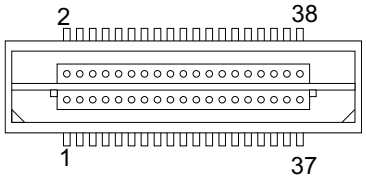

Figure D-6 AMP Mictor connector J4

Table D-6 Mictor connector pinout J4

Channel	Pin	Pin	Channel
NC	1	2	NC
GND	3	4	NC
PCI_CLKE	5	6	NC
PCI_nIRDY	7	8	NC
PCI_nTDRY	9	10	NC
PCI_nINTD	11	12	NC
PCI_nINTC	13	14	**SYSTEM_nRESET**
PCI_nINTB	15	16	**PCI_PAR64**
PCI_nINTA	17	18	**PCI_nSERR**
PCI_nGNTC	19	20	**PCI_nPERR**
PCI_nGNTB	21	22	**PCI_nACK64**
PCI_nGNTA	23	24	**PCI_nREQ64**
PCI_nREQC	25	26	**PCI_M66EN**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table D-6 Mictor connector pinout J4 (continued)

Channel	Pin	Pin	Channel
PCI_nREQB	27	28	**PCI_nRST**
PCI_nREQA	29	30	**PCI_nSTOP**
SPARE4	31	32	**PCI_nDEVSEL**
SPARE3	33	34	**PCI_nFRAME**
SPARE2	35	36	**PCI_nLOCK**
SPARE1	37	38	**PCI_PAR**

D.3.3 JTAG connector

The signals on the JTAG connector J5 are shown in Figure D-7.

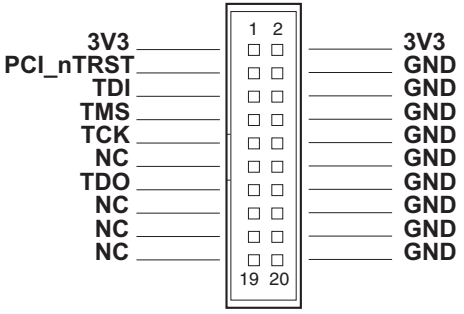

Figure D-7 PCI expansion board JTAG connector J5

D.3.4 PCI connector

The pinout for the PCI connectors are listed in Table D-7.

Table D-7 PCI connectors

Signal	Pin	Pin	Signal
-12V	1	1	**P5_nTRST**
P5_TCK	2	2	**12V**
GND	3	3	**P5_TMS**
P5_TDO	4	4	**P5_TDI**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table D-7 PCI connectors (continued)

Signal	Pin	Pin	Signal
5V	5	5	**5V**
5V	6	6	**P5_INTD**
P5_nINTA	7	7	**P5_INTB**
P5_nINTC	8	8	**5V**
P5_nPRSNT1	9	9	-
-	10	10	**P_VIO**
P5_nPRSNT2	11	11	-
-	12	12	-
-	13	13	-
-	14	14	-
GND	15	15	**P5_nRST**
P5_CLK	16	16	**P_VIO**
GND	17	17	**P5_nGNT**
P5_nREQ	18	18	**GND**
PVIO	19	19	-
P5_AD31	20	20	**P5_AD30**
P5_AD29	21	21	**3V3**
GND	22	22	**P5_AD28**
P5_AD27	23	23	**P5_AD26**
P5_AD25	24	24	**GND**
3V3	25	25	**P5_AD24**
PB5_nCBE3	26	26	**P5_IDSEL**
P5_AD23	27	27	**3V3**
GND	28	28	**P5_AD22**
P5_AD21	29	29	**P5_AD20**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table D-7 PCI connectors (continued)

Signal	Pin	Pin	Signal
P5_AD19	30	30	**GND**
3V3	31	31	**P5_AD18**
P5_AD17	32	32	**P5_AD16**
P5_nCBE2	33	33	**3V3**
GND	34	34	**P5_nFRAME**
P5_nIRDY	35	35	**GND**
GND	36	36	**P5_nTRDY**
P5_nDEVSEL	37	37	**GND**
XCAP	38	38	**P5_nSTOP**
-	39	39	**3V3**
P5_nPERR	40	40	-
3V3	41	41	-
P5_nSERR	42	42	**GND**
3V3	43	43	**P5_PAR**
P5_nCBE1	44	44	**P5_AD15**
P5_AD14	45	45	**3V3**
GND	46	46	**P5_AD13**
P5_AD12	47	47	**P5_AD11**
P5_AD10	48	48	**GND**
M66EN	49	49	**P5_AD9**
GND	50	50	-
GND	51	51	**GND**
P5_AD8	52	52	**P5_nCBE0**
P5_AD7	53	53	**3V3**
3V3	54	54	**P5_AD6**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Table D-7 PCI connectors (continued)

Signal	Pin	Pin	Signal
P5_AD5	55	55	**P5_AD4**
P5_AD3	56	56	**GND**
GND	57	57	**P5_AD2**
P5_AD1	58	58	**P5_AD0**
PVIO	59	59	**PVIO**
-	60	60	-
5V	61	61	**5V**
5V	62	62	**5V**

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Appendix E
PISMO Memory Expansion Boards

This appendix describes PISMO expansion memory modules for the baseboard. It contains the following sections:

- *About memory expansion* on page E-2
- *Mechanical layout* on page E-5.

Note

The baseboard supports memory modules that follow the PISMO version 1.0 specification. For more details on the PISMO specification, see the PISMO web site at www.pismoworld.org.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

E.1 About memory expansion

You can fit a static memory expansion board to the baseboard. There are four chip select signals available for use by the static expansion board. Each of these can select 64MB of SRAM.

The block diagrams for a typical memory board is shown in Figure E-1.

Note

Different expansion boards might have different features. For example, the links selecting which chip select to use might be omitted. See the documentation provided with your memory board for details on signals and link options.

PISMO signal **nCS4** (**nSTATICCS7**) is tied permanently HIGH on the baseboard.

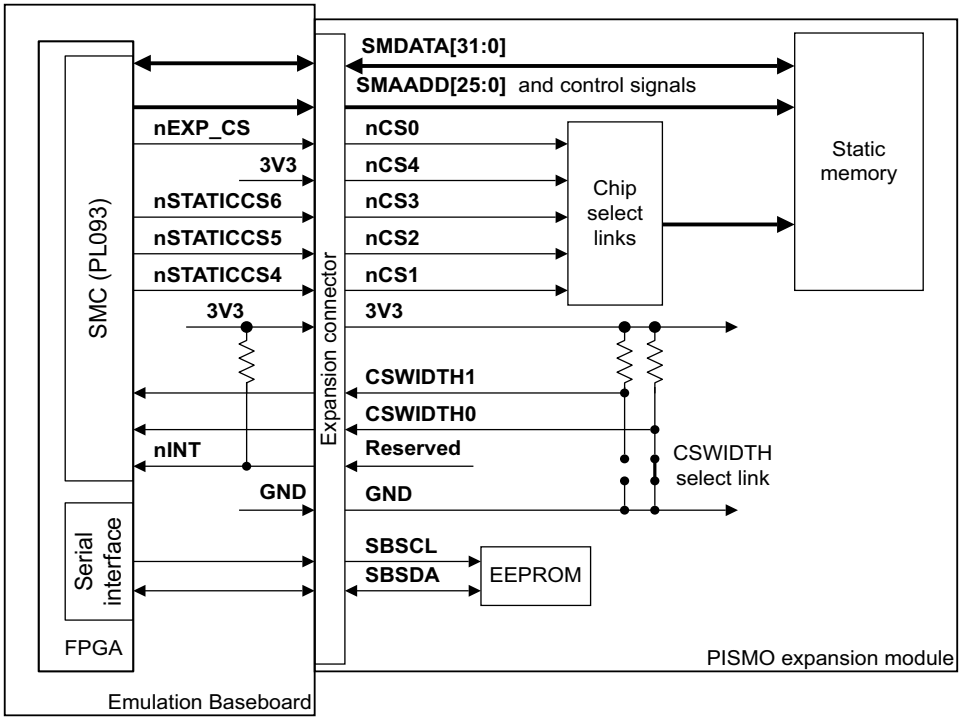

Figure E-1 Static memory board block diagram

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

E.1.1 Operation without expansion memory

You can operate the baseboard without a memory expansion board because it has 2MB of SRAM, 128MB DDR SRAM, 64MB NOR flash, and 64MB NAND flash permanently fitted.

You can use the expansion boards, however, to prototype or develop memory devices that are not available on the baseboard.

E.1.2 Memory board configuration

The memory width for the memory board is coded into signals present on the connector. The serial EEPROM on the memory board can be read from the baseboard 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.

Memory width selection on the static memory board

The memory width on the memory board is encoded into the **CSWIDTH[1:0]** signals as shown in Table E-1.

If the configuration switches are set to boot from PISMO memory and more than one PISMO expansion board is present, the CSWIDTH signals on the bottom-most memory board determine the memory width for the boot memory.

Table E-1 Memory width encoding

CSWIDTH[1:0]	Width
00	8 bit
01	16 bit
10	32 bit (default)
11	No memory present

EEPROMs

There are two serial devices on the baseboard serial bus:

- Memory expansion board EEPROM at `0xA2` for write, `0xA3` for read
- Real Time Clock (Time of Year) at `0xD0` for write, `0xD1` for read

See *Two-wire serial bus interface* on page 4-81 for details on the serial bus interface.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

The expansion EEPROM provides information about the type of memory expansion board installed:

- PISMO vendor and product name
- power supply and I/O voltages supported
- details of the local memory banks (for example, bank architecture, access mode, access time).

See the PISMO specification for more details about the EEPROM contents.

E.1.3 Fitting a memory board

See *Connecting expansion memory* on page 2-12 for instructions on installing a memory expansion board.

Static memory expansion boards can also be fitted to Core Tiles, but this not supported by the default FPGA images.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

E.2 Mechanical layout

Figure E-2 shows the memory expansion board (viewed from above).

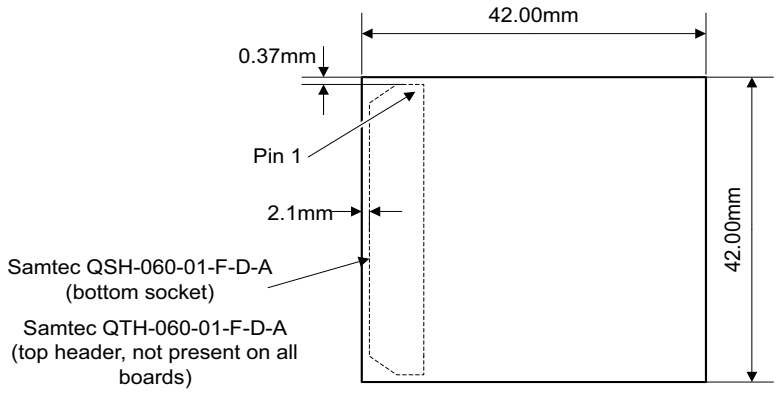

Figure E-2 Static memory board layout

Note

For details on the connector pinout, see *PISMO connector* on page A-14.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Glossary

This glossary lists abbreviations used in the User Guide.

ADC *Analog to Digital Converter*. A device that converts an analog signal into digital data.

AACI *Advanced Audio CODEC Interface*.

AHB *Advanced High-performance Bus*. The ARM open standard for on-chip buses.

AMBA *Advanced Microcontroller Bus Architecture*.

APB *Advanced Peripheral Bus*. The ARM open standard for peripheral buses. This design is optimized for low power and minimal interface complexity.

AXD *ARM eXtended Debugger*.

AXI *Advanced eXtensible Interface*. The ARM open standard for high-performance and high-frequency buses.

BOM *Bill Of Materials*. A list of all the parts used on the printed circuit board and any specific build instructions for board variants.

CLCD *Color Liquid-Crystal Display*.

CODEC *COder DECoder* for converting between analog and digital audio signals.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

DAC *Digital to Analog Converter*. A device that converts digital data into analog level signals.

DCC *Debug Communications Channel*.

DDR *Double Data Rate*, DRAM with high-speed data access.

DOC *Disk-On-Chip*. A non-volatile flash memory device with an interface that simplifies file accesses. Also called NAND flash referring to the logic gates used internally. The memory can only be accessed sequentially in blocks.

DMC *Dynamic Memory Controller*.

DMA *Direct Memory Access*.

DRAM *Dynamic Random Access Memory*.

DSR *Data Set Ready*, a UART flow-control signal.

DTR *Data Terminal Ready*, a UART flow-control signal.

ETB *Embedded Trace Buffer*.

ETM *Embedded Trace Macrocell*.

FET *Field Effect Transistor*.

FPGA *Field Programmable Gate Array*.

GIC *Generic Interrupt Controller*.

GPIO *General Purpose Input/Output*.

GTC *Generic Test Chip*. A packaging and signal assignment specification for test chips.

ICE *In Circuit Emulator*. A interface device for configuring and debugging processor cores.

Integrator/CP Integrator *Compact Platform*.

Integrator/IM-LT1 *Interface Module* used to connect tile-format boards to an Integrator system.

Integrator/IM-LT3 *Interface Module* used to connect tile-format boards to an Integrator system. This module has an FPGA that provides the interface between the Integrator signals and the tile signals.

I/O *Input/Output*.

IP *Intellectual Property*.

JTAG *Joint Test Action Group*. The committee which defined the IEEE test access port and boundary-scan standard.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

KMI	*Keyboard/Mouse Interface*.
LCD	*Liquid Crystal Display*.
LED	*Light Emitting Diode*.
MAC	*Media Access Control*. A layer in the Ethernet specification.
MCI	*MultiMedia Card Interface*.
MMC	*MultiMedia Card*.
Multi-ICE	Multi-ICE is a system for debugging embedded processor cores using a JTAG interface. See ICE.
NAND flash	Non-volatile memory. *NAND* refers to the type of logic gate used internally. See *DOC*.
NOR flash	Non-volatile memory. *NOR* refers to the type of logic gate used internally. Any memory address can be accessed randomly.
OTG	*On-The-Go*, a USB specification.
PCB	*Printed Circuit Board*.
PCI	*Peripheral Component Interconnect*. A computer bus for attaching peripherals.
PHY	*PHYsical* layer. The layer in the Ethernet specification that describes the physical interface.
PISMO	*Platform Independent Storage Module*. Memory specification for plug in memory modules.
PLD	*Programmable Logic Device*.
PLL	*Phase-Locked Loop*, a type of programmable oscillator.
POR	*Power On Reset*.
RAM	*Random Access Memory*.
RVI	*RealView ICE*. A system for debugging embedded processor cores using a JTAG interface. See also ICE.
RTC	*Real-Time Clock*.
RVD	*RealView Debugger*.
SRAM	*Static Random Access Memory*.
SCI	*Smart Card Interface*.
SD	*Secure Digital* memory card specification.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

SMC *Static Memory Controller*.

SMM *Soft Macrocell Model* of a CPU core.

SSP *Synchronous Serial Port*.

TCM *Tightly Coupled Memory*. Memory present inside the test chip that typically runs at or near the processor speed.

TPA *Trace Port Analyzer*.

UART *Universal Asynchronous Receiver/Transmitter*.

USB *Universal Serial Bus*.

VGA *Video Graphics Array*.

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Index

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

R

S

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential

T

U

W

Copyright © 2005-2011 ARM Limited. All rights reserved.
Non-Confidential