RealView® Platform Baseboard Explore for Cortex™-A9

HBI-0182 (baseboard) and HBI0183 (daughterboard)

User Guide

ARM®

ARM DUI 0440B (ID060911)

RealView Platform Baseboard Explore for Cortex-A9
User Guide

Release Information

Change History

Date	Issue	Confidentiality	Change
24 March 2009	A	Non-Confidential	First release
23 May 2011	B	Non-Confidential	Second release

Proprietary Notice

Confidentiality Status

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

Unrestricted Access is an ARM internal classification.

Product Status

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

Web Address

`http://www.arm.com`

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 Platform Baseboard for A9 generates, uses, and can radiate radio frequency energy and may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment causes harmful interference to radio or television reception, which can be determined by turning the equipment off or on, you are encouraged to try to correct the interference by one or more of the following measures:

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

Note

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

Contents
RealView Platform Baseboard Explore for Cortex-A9 User Guide

List of Tables
RealView Platform Baseboard Explore for Cortex-A9 User Guide

List of Figures
RealView Platform Baseboard Explore for Cortex-A9 User Guide

Preface

This preface introduces the *RealView® Platform Baseboard Explore for Cortex™-A9 User Guide*. It contains the following sections:

About this book

This book describes how to set up and use the RealView Platform Baseboard Explore for Cortex-A9 (PBX-A9).

Intended audience

This document has been written for experienced hardware and software developers to aid the development of ARM-based products using the PBX-A9 as part of a development system.

Using this book

This book is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an introduction to the PBX-A9. 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 PBX-A9. This chapter describes connecting the debug equipment and initializing and configuring the baseboard.

Chapter 3 ***Hardware Description***

Read this chapter for a description of the hardware architecture of the PBX-A9. This chapter describes the peripherals, clocks, resets, and the debug hardware provided by the baseboard.

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

Read this chapter for a description of the PBX-A9 memory map and registers. There is also basic information on the peripherals and controllers present on the baseboard.

Appendix A ***Signal Descriptions***

Read this appendix for a description of the signals on the connectors.

Appendix B ***Specifications***

Read this appendix for electrical and timing specifications.

Appendix C ***RealView Logic Tile Expansion***

Read this appendix for details of the signals present on the Logic Tile expansion headers and the steps required to interface a Logic Tile to the baseboard.

Appendix D ***Boot Monitor and platform library***

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

Appendix E ***Boot Monitor Commands***

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

Appendix F ***Loading FPGA Images***

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

Product revision status

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

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

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

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

Typographical conventions

The following typographical conventions are used in this book:

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

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

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

mono`space`	Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
`monospace italic`	Denotes arguments to commands and functions where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.

Other conventions

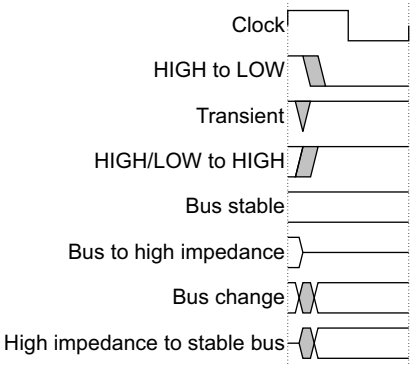
This document uses other conventions. They are described in the following sections:

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

Timing diagrams

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

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



Key to timing diagram conventions

Signals

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

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

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

Prefix AR Denotes AXI read address channel signals.

Prefix AW Denotes AXI write address channel signals.

Prefix B Denotes AXI write response channel signals.

Prefix C Denotes AXI low-power interface signals.

Prefix H AHB signals are prefixed by an upper case H.

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

Prefix R Denotes AXI read data channel signals.

Prefix W Denotes AXI write data channel signals.

Bytes, Halfwords, and Words

Byte Eight bits.

Halfword Two bytes (16 bits).

Word Four bytes (32 bits).

Quadword 16 contiguous bytes (128 bits).

Bits, bytes, k, and M

Suffix b Indicates bits.

Suffix B Indicates bytes.

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

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

Register fields

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

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

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

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

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

Numbering

The numbering convention is:

Hexadecimal numbers

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

Binary numbers

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

Ranges

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

Bit numbers

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

Further reading

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

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

ARM publications

This manual contains information that is specific to the PBX-A9. See the following documents for other relevant information:

The following publications provide reference information about the ARM architecture:

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

The following publications provide information about ARM PrimeCell® controllers used in the Cortex-A9 processor subsystem:

- *ARM PL341 Dynamic Memory Controller Technical Reference Manual* (ARM DDI 0331)
- *ARM PrimeCell Generic Interrupt Controller (PL390) Technical Reference Manual* (ARM DDI 0416)
- *ARM PL301 High-Performance Matrix Technical Summary* (ARM DDI 0422)
- *ARM Cortex-A9 MPCore Technical Reference Manual* (ARM DDI 0407)
- *ARM Cortex-A9 Technical Reference Manual* (ARM DDI 0388)
- *CoreLink Level 2 Cache Controller (L2C-310) Technical Reference Manual* (ARM DDI 0246).

The following publications provide information about ARM PrimeCell controllers and bus interconnect used in the PBX-A9 Northbridge ASIC:

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

- *ARM PrimeCell AXI Configurable Interconnect (PL300) Technical Reference Manual* (ARM DDI 0354).

The following publications provide information about ARM PrimeCell peripherals and controllers used in the Southbridge FPGA:

- *ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual* (ARM DDI 0173)
- *ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual* (ARM DDI 0172)
- *ARM PrimeCell System Controller (SP810) Technical Reference Manual* (ARM DDI 0254)
- *ARM PrimeCell PS2 Keyboard/Mouse Interface (PL050) Technical Reference Manual* (ARM DDI 0143)
- *ARM PrimeCell UART (PL011) Technical Reference Manual* (ARM DDI 0183)
- *ARM PrimeCell Synchronous Serial Port (PL022) Technical Reference Manual* (ARM DDI 0194)
- *ARM PrimeCell Smart Card Interface (PL131) Technical Reference Manual* (ARM DDI 0228)
- *ARM PrimeCell General Purpose Input/Output (PL061) Technical Reference Manual* (ARM DDI 0190)
- *ARM PrimeCell Real Time Clock (PL031) Technical Reference Manual* (ARM DDI 0224).

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

- *ARM RealView Logic Tile LT-XC5VLX330 User Guide* (ARM DUI 0365)
- *ARM RealView Logic Tile LT-XC4VLX100+ User Guide* (ARM DUI 0345)
- *ARM RealView® ICE User Guide* (ARM DUI 0155)
- *ARM RealView Debugger User Guide* (ARM DUI 0153)
- *ARM RealView Compilation Tools Compilers and Libraries Guide* (ARM DUI 0205)
- *ARM RealView Compilation Tools Developer Guide* (ARM DUI 0203)
- *ARM RealView Compilation Tools Linker and Utilities Guide* (ARM DUI 0206)
- *ARM CoreSight PTM-A9 Technical Reference Manual* (ARM DDI 0401).

Other Publications

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

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

Feedback

ARM® Limited welcomes feedback both on the PBX-A9 and on the documentation.

Feedback on this product

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

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

Feedback on this manual

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

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

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

Chapter 1
Introduction

This chapter introduces the *RealView® Platform Baseboard Explore for Cortex-A9 User Guide* (PBX-A9). It contains the following sections:

1.1 Precautions

This section contains safety information and advice on how to avoid damage to the PBX-A9 baseboard.

1.1.1 Ensuring safety

The PBX-A9 baseboard is powered from an ATX power supply unit within the ATX enclosure.

Warning

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

1.1.2 Preventing damage

The PBX-A9 baseboard is intended for use in a laboratory or engineering development environment. If removed from its enclosure, the board will become more sensitive to electrostatic discharges and will generate increased electromagnetic emissions.

Caution

To avoid damage, observe the following precautions:

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

Caution

Do not use near equipment that is:

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

1.2 About the PBX-A9 baseboard

The PBX-A9 baseboard is a highly integrated software and hardware development system based on the ARMv7-A architecture. It is supplied self-powered, in an ATX profile enclosure.

Note

The Cortex-A9 processor subsystem is located on a custom daughterboard (HBI0183). The daughterboard cannot be removed and used separately from the PBX-A9 baseboard (HBI0182).

Used standalone, the baseboard serves as a fast multiprocessor software development platform with either two or three Cortex-A9 processors (CPU0 − CPU2) and a memory system in a structured ASIC running at near ASIC speed.

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

The PBX-A9 is intended for development of both embedded and operating system based software for ARM processors. The ability to add Logic Tiles enables new hardware to be prototyped and validated, and drivers to be debugged.

The PBX-A9 includes:

- a Cortex-A9 structured ASIC
- a structured ASIC (Northbridge) with performance-critical peripherals
- an FPGA (Southbridge) with additional peripherals
- static and dynamic memory
- integrated peripherals
- a tile site to enable connection of RealView Logic Tiles
- connectors for external peripherals and JTAG configuration and debug.

Figure 1-1 on page 1-4 shows the top level system architecture of the baseboard.

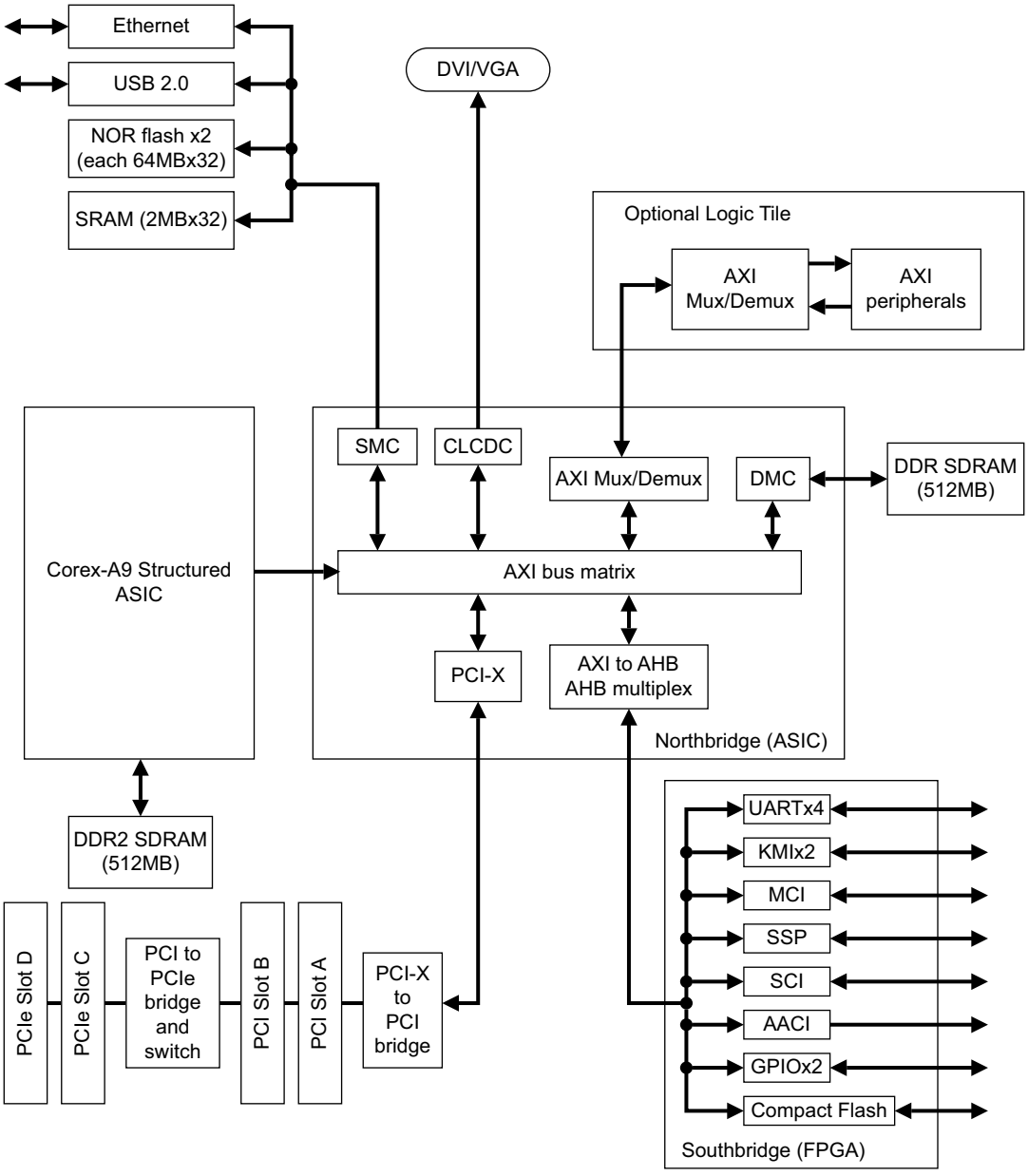


Figure 1-1 PBX-A9 system architecture

The following sections introduce the major system components:

- *Cortex-A9 structured ASIC*
- *Northbridge*
- *Southbridge* on page 1-6
- *PBX-A9 expansion* on page 1-6.

1.2.1 Cortex-A9 structured ASIC

The Cortex-A9 test chip is fabricated with structured ASIC technology and implements the following system components and interfaces:

- two Cortex-A9 processors
- separate 32KB data and instruction caches
- Dynamic Memory Controller (PL341) for DDR2 SDRAM
- Generic Interrupt Controller (Cortex-A9 specific)
- CoreSight debug and trace
- AXI Master interface (64-bit) to the Northbridge.

Note

The cortex-A9 structured ASIC is available with two Cortex-A9 CPUs. For more information, see *Cortex-A9 structured ASIC with dual core, PBXA9-BD-0241A* on page 3-13.

1.2.2 Northbridge

The Northbridge implements the following system components and interfaces:

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

See *Northbridge* on page 3-15 and the appropriate *Technical Reference Manual* (TRM) for details of the PrimeCell components that are integrated in the Northbridge structured ASIC.

1.2.3 Southbridge

The Southbridge implements the following peripheral components and interfaces:

- Advanced Audio CODEC Interface (PL041)
- Multimedia Card Interface (PL180)
- PS2 Keyboard/Mouse Interface (PL050) x 2
- UART (PL011) x 4
- Watchdog Module (SP805) x 2
- Dual-Timer Module (SP804) x 4
- Synchronous Serial Port (PL022)
- Smart Card Interface (PL131)
- General Purpose Input/Output (PL061) x 3
  - sixteen GPIO ports available for external use
  - eight GPIO ports reserved for internal use.
- Real Time Clock (PL031)
- Generic Interrupt Controller x 2 (customized)
- Multiplexed AHB-Lite interface to the Northbridge
- AHB interface to the baseboard CompactFlash memory.

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

Note

ARM does not support modifications made to the ARM Southbridge FPGA design. Custom peripheral development is only supported by ARM when using an attached Logic Tile such as the LT-XC5VLX330. Specific *Application Notes* are made available on the CD supplied with the PBX-A9.

1.2.4 PBX-A9 expansion

You can expand the PBX-A9 baseboard by adding:

- one or more stacked Logic Tiles containing your custom IP
- one or more PCI or PCI Express® cards
- CompactFlash card
- VGA monitor or color LCD display (DVI connection)
- MMC, SD, or SIM cards
- USB devices to the three USB ports (1 OTG and 2 standard host ports)
- serial devices to the synchronous serial port (SSP) and the four UARTs
- a keyboard and mouse

- audio devices to the onboard CODEC (AAC)
- a 10/100Mbps Ethernet network to the onboard Ethernet controller
- custom I/O to the 16-bit GPIO header.

Chapter 2
Getting Started

This chapter describes how to set up and prepare the PBX-A9 baseboard. It contains the following sections:

2.1 Setting up the baseboard

The following items are supplied:

- a custom ATX enclosure containing:
  - a micro ATX profile printed-circuit board (HBI-0182 and HBI-0183)

    Note

    The Cortex-A9 daughterboard is bolted to the baseboard.
  - front panel printed-circuit boards (HBI-0175, HBI-0176)
  - an ATX mains operated power supply (110V/230V AC).
- a CD containing sample programs, Boot Monitor code, FPGA and PLD images, and additional release documentation.

Note

For normal standalone operation, it is not necessary to change any of the default switch settings on the baseboard. See *Baseboard configuration switches* on page 2-8.

Before fitting any Logic Tiles, you must test that the standard PBX-A9 system boots normally.

To set up the PBX-A9 baseboard as a standalone development system:

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

   Note

   A DVI-I cable can be used to connect either a digital or analog display.

4. If you are using a debugger, connect to the JTAG connector (JTAG-ICE) on the back panel of the enclosure. See *Rear panel layout* on page 3-5 for the location of the JTAG-ICE connector, and *JTAG, USB config, and Trace support* on page 2-6 for information on debug support.

5. If you are using a *Trace Port Analyzer* (TPA), connect the *Trace Port Analyzer* (TPA) to the adaptor board and plug the adaptors into the daughterboard trace ports. See *Connecting the Trace Port Analyzer* on page 2-7.

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

7. If you are using the supplied Boot Monitor software to select and run an application, see Appendix D *Boot Monitor and platform library* for information on using this software.

2.2 Boot Monitor configuration

The Boot Monitor application is supplied preloaded into the NOR flash memory and selected to run at power on. If the flash becomes corrupt and the board no longer runs the Boot Monitor, follow the instructions in *Loading Boot Monitor into NOR flash* on page D-6 for details of loading the boot flash with the image from the supplied CD. How the Boot Monitor runs is determined by the setting of User Switches.

Note

It is not necessary to open the enclosure to configure the Boot Monitor, User Switches 1 to 3 on the front panel of the ATX enclosure (see *Front panel layout* on page 3-4) control this. The User Switches in switch bank S4 on the PBX-A9 baseboard (see *Baseboard layout* on page 3-2) duplicate the front panel User Switches and must be left in the default all switches OFF position for normal operation.

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

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

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

Table 2-1 Boot Monitor startup behavior

User Switch S4-1	Startup Behavior
OFF	A prompt is displayed enabling you to enter Boot Monitor commands.
ON	The Boot Monitor executes a boot script that has been loaded into NOR flash, a Multimedia (MMC) or Secure Digital (SD) card. If a boot script is not present, the Boot Monitor prompt is displayed.

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

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

Table 2-2 STDIO redirection

User Switch S4-2	User Switch S4-3	Output	Input	Description
OFF	OFF	UART0 or console	UART0 or console	STDIO autodetects whether to use semihosting I/O or a UART. If a debugger is connected and semihosting is enabled, STDIO is redirected to the debugger console window. Otherwise, STDIO goes to UART0.
OFF	ON	UART0	UART0	STDIO is redirected to UART0. This occurs even under semihosting.
ON	OFF	DVI	Keyboard	STDIO is redirected to the DVI display and keyboard. This occurs even under semihosting.
ON	ON	DVI	UART0	STDIO output is redirected to the DVI display and input is redirected to UART0. This occurs even under semihosting.

User Switches 2 and 3 (S4-2 and S4-3) do not affect file I/O operations performed under semihosting. Semihosting operation requires a compliant debugger, such as the ARM *RealView® Debugger* and a JTAG interface device. See *Redirecting character output to hardware devices* on page D-7 for more details on I/O.

2.3 JTAG, USB config, and Trace support

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

Note

You cannot program application flash memory from configuration mode.

2.3.1 JTAG debugger

You can use a JTAG debugger connected to the JTAG connector on the rear panel to connect to the Cortex-A9 structured ASIC and download programs to memory and debug them.

Note

You gain access to the Cortex-A9 structured ASIC *Debug Advanced Peripheral Bus* (Debug APB) through the Debug Access Port (DAP) integrated in the ASIC.

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

2.3.2 USB config port

The PBX-A9 contains custom logic that interfaces the USB config port to the onboard JTAG signals.You can use a PC connected to the custom USB config port connector on the front panel to:

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

Note

You cannot use the USB config port to debug devices on the PBX-A9.

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

Caution

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

2.3.3 Connecting the Trace Port Analyzer

The Cortex-A9 structured ASIC incorporates a *CoreSight ProgramFlow Trace Macrocell* (PTM). This performs real-time instruction flow tracing based on the *Program Flow Trace* (PFT) architecture. For more information, see the *CoreSight PTM-A9 Technical Reference Manual*.

Note

The Cortex-A9 structured ASIC includes a *Debug Access Port* (DAP). If supported by CoreSight enabled hardware and software the DAP enables system-wide debug and trace.

Ensure that power is disconnected and connect the Trace Probe from the TPA into the daughterboard trace ports that Figure 3-1 on page 3-2 shows.

Note

- RealView Trace requires connection to RealView ICE. The Ethernet and power supply cables connect to the RealView ICE run control unit.
- The PBX-A9 supports trace port widths of up to 32 bits.

Warning

Applying excessive sideways pressure to the Trace Probe interface board (either by handling or having the trace cable too tight) can damage the connectors and the interface board.

2.4 Baseboard configuration switches

Note

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

Figure 3-1 on page 3-2 shows the location of the configuration switch bank (S7) on the baseboard.

The switch settings determine at power up:

- boot memory configuration
- flash recovery mode.

2.4.1 Boot memory configuration

Use switch S7-1 to change the boot device. Table 2-3 shows the available options.

Table 2-3 Selecting the boot device

S7-1	Device
OFF	NOR flash (default)
ON	Reserved (do not use)

2.4.2 Flash recovery

Use switch S7-5 to select whether all or only processor CPU#[0] in the Cortex-A9 are released from reset. Table 2-4 shows the available options.

Table 2-4 Cortex-A9 reset behavior

S7-5	Reset Setting
OFF	CPU#[2:0] are all released from reset (default)
ON	CPU#[0] only is released from reset

Chapter 3
Hardware Description

This chapter describes the PBX-A9 on-board hardware. It contains the following sections:

3.1 Baseboard architecture

Figure 3-1 shows the layout of the PBX-A9 baseboard and attached daughterboard.

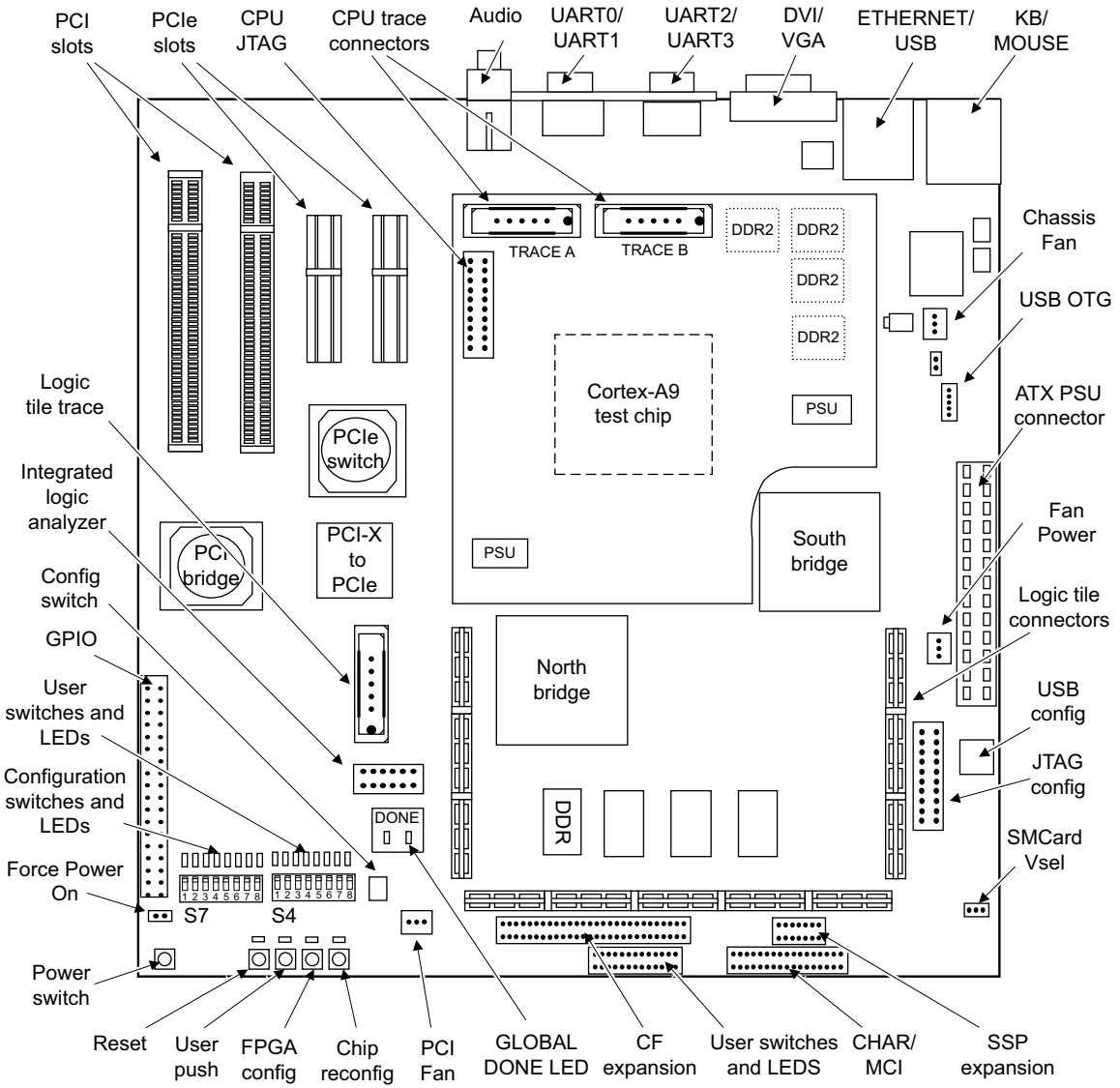


Figure 3-1 Baseboard layout

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

- a fixed daughterboard providing:
  - a Cortex-A9 structured ASIC with either two or three Cortex-A9 processors
  - 512MB of DDR2 SDRAM memory
  - CoreSight JTAG Debug and Trace interfaces
- a tile site to support ARM Logic Tiles
- a Northbridge implementing the major system controllers and interfaces
- a Southbridge implementing most of the system peripherals
- an 8MB configuration flash to hold the FPGA images
- 512MB of 32-bit wide (DDR) SDRAM (2 x 256MB banks)
- 2MB of 32-bit wide (Pseudo) SRAM
- 128MB of 32-bit wide NOR flash in two banks of 64MB
- PCI and PCI Express expansion buses (2 slots for each interface type)
- USB interface providing 1 OTG and 2 standard USB 2.0 host ports
- Ethernet interface providing 10Base-T and 100Base-TX support
- DVI-I interface providing color LCD display and analog VGA monitor support
- *Synchronous Serial Port* (SSP) interface
- 4 x RS232 interfaces with full handshake
- PS2 keyboard and mouse interfaces
- audio CODEC interface (AAC)
- CompactFlash, MMC, SD and Smart Card interfaces
- general purpose (User) switches and LEDs
- real-time clock (RTC)
- time-of-year clock (TOY) with backup battery
- programmable clock generators
- power supplies, voltage control, and current monitoring circuitry
- JTAG config and debug, and Integrated Logic Analyzer (ILA) support
- USB config port.

Figure 3-2 on page 3-4 shows the front panel of the ATX enclosure that provides:

- power on-off button and indicator
- USB Config port connector (USB CFG)
- USB Config switch and indicator
- Reset switch
- FPGA Config switch
- CompactFlash interface connector
- USB OTG interface connector
- SD and MMC memory card interface connector

- User switches 1 to 8
- User LEDs 1 to 8
- SIM Card interface.

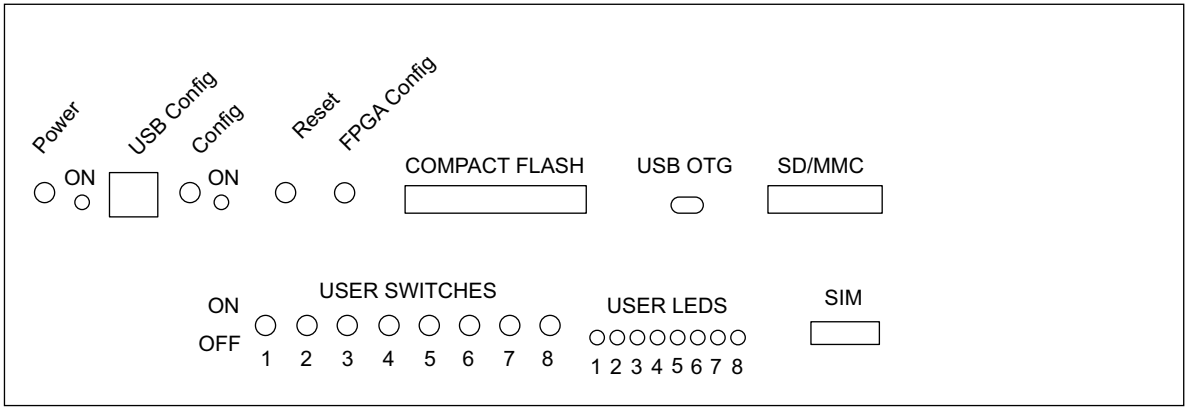


Figure 3-2 Front panel layout

Figure 3-3 on page 3-5 shows the ATX enclosure rear panel that provides:

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

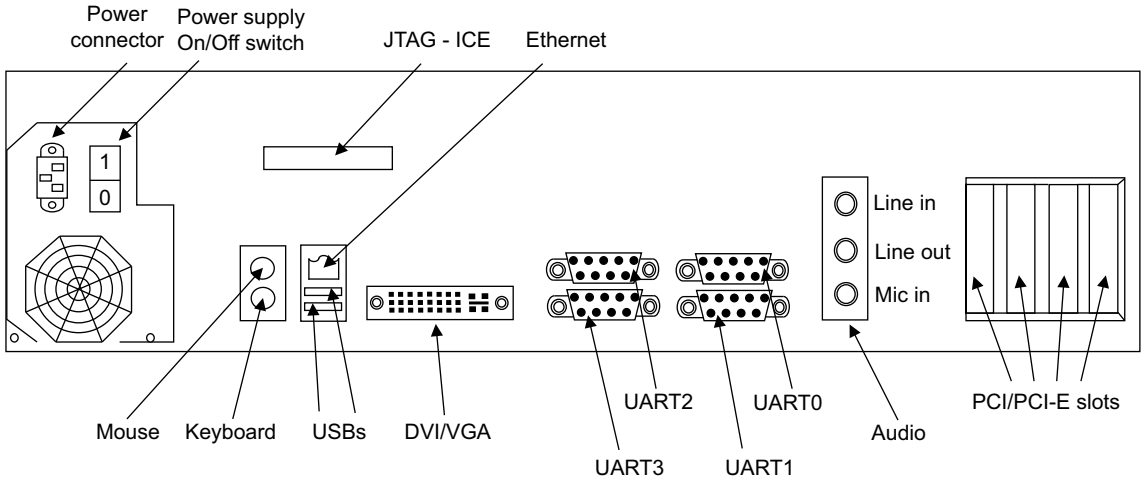


Figure 3-3 Rear panel layout

3.1.1 Bypassing the power switch

The power button on the front panel of the PBX-A9 baseboard is typically used to power on the system.

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

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

3.1.2 System architecture

Figure 3-4 on page 3-6 shows the top level system architecture of the PBX-A9 baseboard.

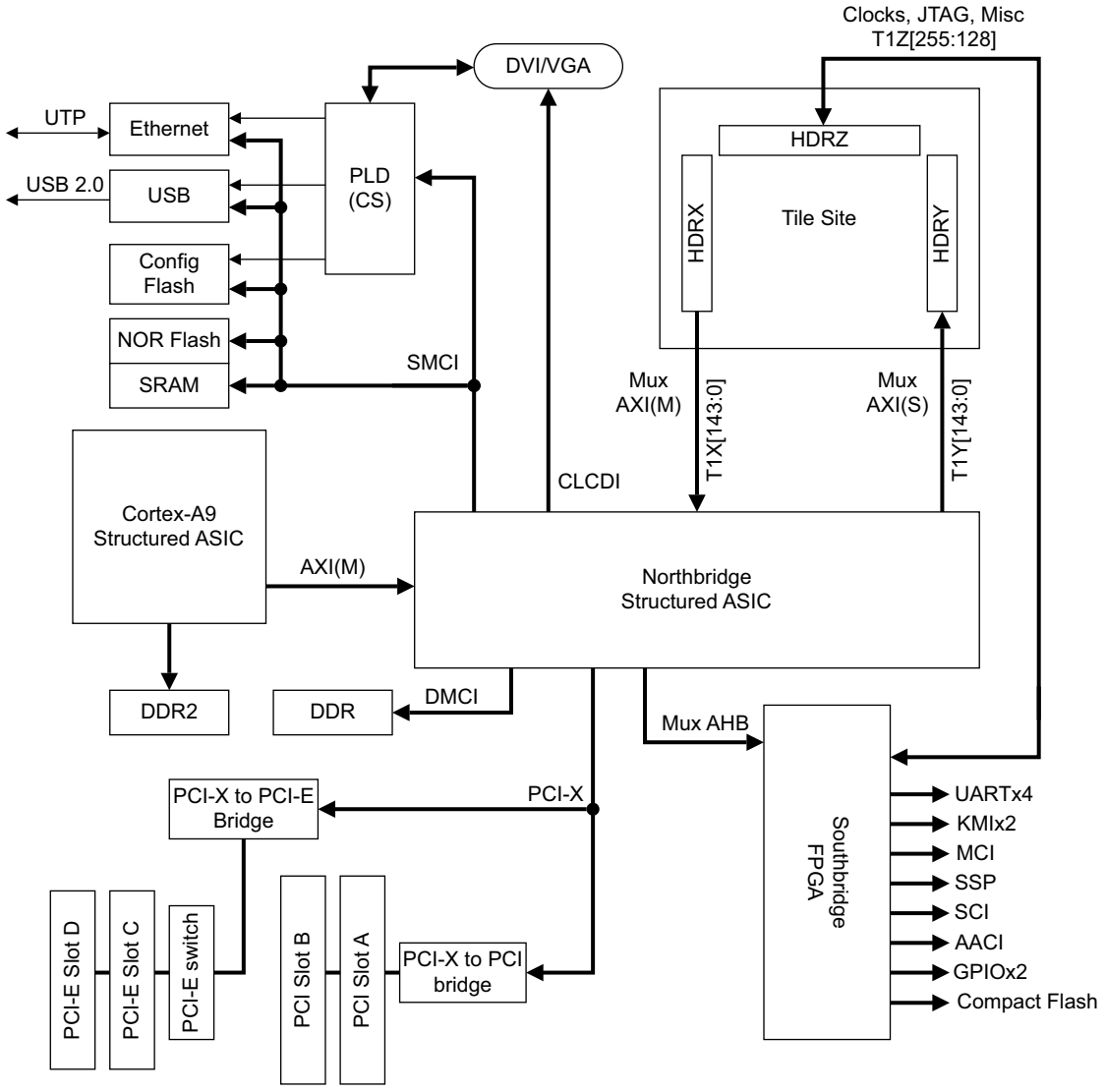


Figure 3-4 PBX-A9 top level block diagram

3.1.3 Cortex-A9 test chip

The Cortex-A9 test chip is a structured ASIC. It contains a Cortex-A9 MPCore with either two or three processors and a *Snoop Control Unit* (SCU). An *Accelerator Coherency Port* (ACP) suitable for coherent memory transfers is provided in the SCU. Each processor has 32KB Instruction and Data caches and an integrated Timer and

Watchdog unit. The test chip also contains a Cortex-A9 specific implementation of the ARM *Generic Interrupt Controller* (GIC) architecture, a DDR2 memory interface, and system wide CoreSight Debug and Trace.

3.1.4 Northbridge

The PBX-A9 Northbridge is a structured ASIC that contains the major system controllers and interfaces. It is connected to the Cortex-A9 test chip and enables the PBX-A9 to operate at near ASIC speeds as a standalone development system for Cortex-A9 based applications. See *Northbridge* on page 3-15 for details.

The Northbridge also interfaces to the tile site. This enables the PBX-A9 to operate as a standalone AMBA 3 AXI peripheral development system using an attached Logic Tile.

3.1.5 Southbridge

The PBX-A9 Southbridge is an FPGA (Xilinx XC5VLX110) and provides the majority of the system peripherals. It also implements the system control and configuration functions required by the PBX-A9. See *Southbridge peripherals* on page 3-20 for details.

Note

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

Caution

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

3.1.6 PCI bus connectors

The PBX-A9 baseboard has two PCI and two PCI Express bus connectors to enable PCI and PCI Express profile cards to be connected directly to the baseboard. Rear panel access to the card back plates is provided. See *Rear panel layout* on page 3-5 for details.

3.1.7 Displays

The color LCD signals from the CLCD controller in the PBX-A9 PBX-A9 Northbridge are processed by a DVI transmitter chip and fed to the baseboard DVI-I connector. The color LCD signals are also converted to analog VGA signals by a video DAC and fed to the baseboard DVI-I connector. A serial 2-wire interface implemented in the Southbridge provides the DDC2B interface. See *DVI Interface* on page 3-30 for details.

Note

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

3.1.8 Logic Tile expansion

Logic Tiles, such as the LT-XC5VLX330, can be connected to the PBX-A9 tile site to enable the development of additional AMBA 3 AXI peripherals, or custom logic, for use with ARM cores. See Appendix C *RealView Logic Tile Expansion* and *RealView Logic Tile header connectors* on page A-19 for details.

Caution

Because of pin limitations, the PBX-A9 implements multiplexed AMBA 3 AXI interfaces at the HDRX and HDRY tile site headers. The Logic Tile used must implement a similar multiplexing scheme to be compatible with the PBX-A9 external AMBA 3 AXI interfaces.

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

3.1.9 Clock generation

The PBX-A9 baseboard provides all necessary clock sources for the Cortex-A9 structured ASIC, the on-board peripherals, and Logic Tiles fitted to the tile site. See *Clock architecture* on page 3-33 for details.

3.1.10 Debug and test interfaces

The PBX-A9 baseboard provides separate debug and configuration JTAG interfaces.

The JTAG connector on the rear panel of the ATX enclosure enables JTAG hardware debugging equipment, such as RealView ICE, to be connected to the PBX-A9 for software debug of ARM-based applications.

The JTAG connector on the PBX-A9 baseboard enables configuration of the FPGA and PLDs on the PBX-A9 and attached Logic Tiles. Using a PC and the `progcards_usb` utility, you can also control the configuration JTAG signals using the dedicated USB config port provided on the front panel of the ATX enclosure.

Note

ARM recommended using the USB config port for PBX-A9 configuration.

Trace connectors are present on the processor daughterboard for multicore tracing with RVT. A trace connector is also present on the baseboard for accessing processor cores implemented on Logic Tiles.

See *JTAG, USB config, and Trace support* on page 2-6.

An *Integrated Logic Analyzer* (ILA) connector on the baseboard enables debugging of Logic Tile FPGA designs at the same time as the JTAG connector is being used to debug application code running on the Cortex-A9. An example of an ILA debugging device is the Xilinx ChipScope.

3.2 Tile interconnections

The tile site on the PBX-A9 baseboard enables the board to be used with Logic Tiles. The tiles are stackable and each tile has three connectors on the top and bottom (HDRX, HDRY, and HDRZ). See *RealView Logic Tile header connectors* on page A-19 for details.

Note

Logic Tiles can be fitted at the PBX-A9 tile site. Support is available through *Applications Notes* provided on the *Versatile Family CD* and on the ARM website at: `www.arm.com/documentation/Application_Notes/`

The bus usage on the tiles depend on the type of baseboard and the combination of tiles used. For PBX-A9, HDRX (Bus T1X) carries the multiplexed AXI master bus signals and HDRY carries the multiplexed AXI slave bus signals (Bus T1Y). HDRZ (Bus T1Z) carries the additional I/O signals required by the Logic Tile (clocks, resets, JTAG etc). See Figure 3-5 on page 3-11 for a simplified diagram of the main tile and system bus routing.

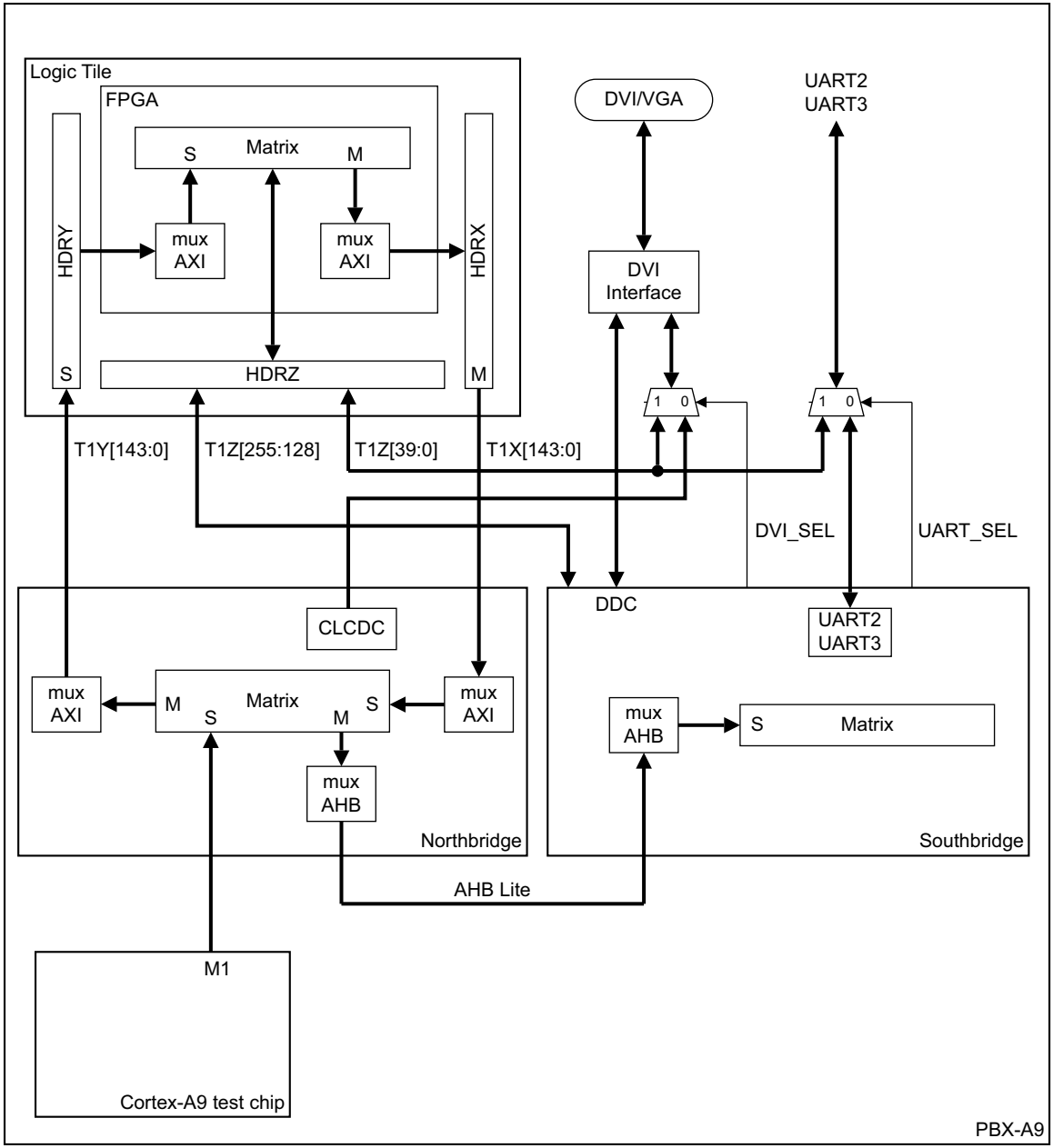


Figure 3-5 Overview of bus routing in baseboard and tile

Note

On the PBX-A9, CLCD, UART2, and UART3 interfaces can be sourced from the baseboard or logic implemented in the FPGA on an attached Logic Tile. The CLCD source and the UART 2 and UART3 source are controlled independently by signals **DVI_SEL** and **UART_SEL** respectively. See Table 4-15 on page 4-28 for details on selecting the source using the SYS_MISC register.

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

3.2.1 AXI bus multiplexing

A bus multiplexing scheme is necessary to reduce the number of pins required on the HDRX and HDRY headers. The Logic Tile used must implement a similar multiplexing scheme to be compatible with the PBX-A9 external AMBA 3 AXI interfaces. See *Application Note AN151* for details and example code.

3.3 Cortex-A9 structured ASIC with dual core, PBXA9-BD-0241A

The Cortex-A9 structured ASIC with dual core is a test chip for the Cortex-A9 macrocell that enables you to build prototype Cortex-A9 systems designed for product and operating system development.

Figure 3-6 on page 3-14 shows the top-level functionality of the Cortex-A9 structured ASIC with dual core.

The Cortex-A9 structured ASIC functionality consists of:

- Two Cortex-A9 CPUs that implement the ARM architecture v7-AR. Each CPU might be suspended using an SWI instruction

  Note

  For details on the CPU cluster see *Cortex-A9 MPCore Technical Reference Manual* (ARM DDI 0407).

- *Generic Interrupt Controller* (GIC) that is Cortex-A9 specific, and is built into the CPU cluster
- *Snoop Control Unit* (SCU) to ensure coherency within the CPU cluster
- PL301 Bus Matrix
- PL341 DDR2 Memory Controller supporting 512MB of DDR2 external SDRAM
- Two level 1 (L1) memory subsystems providing 32KB instruction cache and 32KB data cache per CPU
- PL310 *Level Two Cache Controller* (L2CC)
- CoreSight DAP JTAG-based debug
- *Vector Floating Point* (VFP) coprocessor, supporting the ARM VFPv3 floating point coprocessor instruction set
- NEON™ *Media Processor Engine* (MPE) technology for multimedia and signal processing applications
- Embedded Trace with 8KB of trace buffer
- 64-bit external master AXI bus
- Register bank for configuring the Cortex-A9 structured ASIC.

Note

For L2CC configuration details see *CoreLink Level 2 Cache Controller (L2C-310) Technical Reference Manual.*

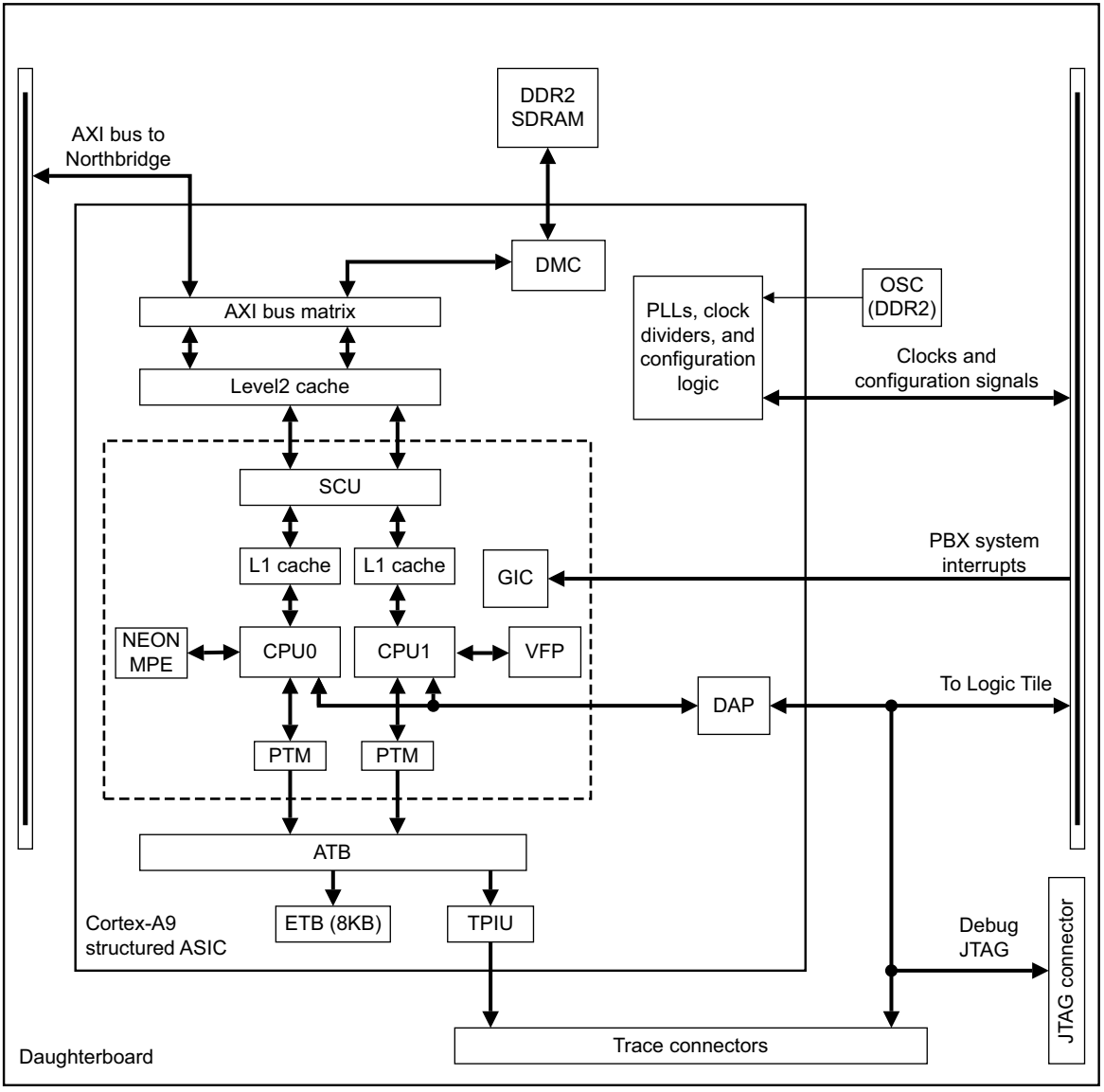


Figure 3-6 Top-level view of Cortex-A9 dual core structured ASIC and daughterboard

3.4 Northbridge

Figure 3-7 shows in simplified form the architecture of the PBX-A9 Northbridge. The Northbridge is implemented as a structured ASIC.

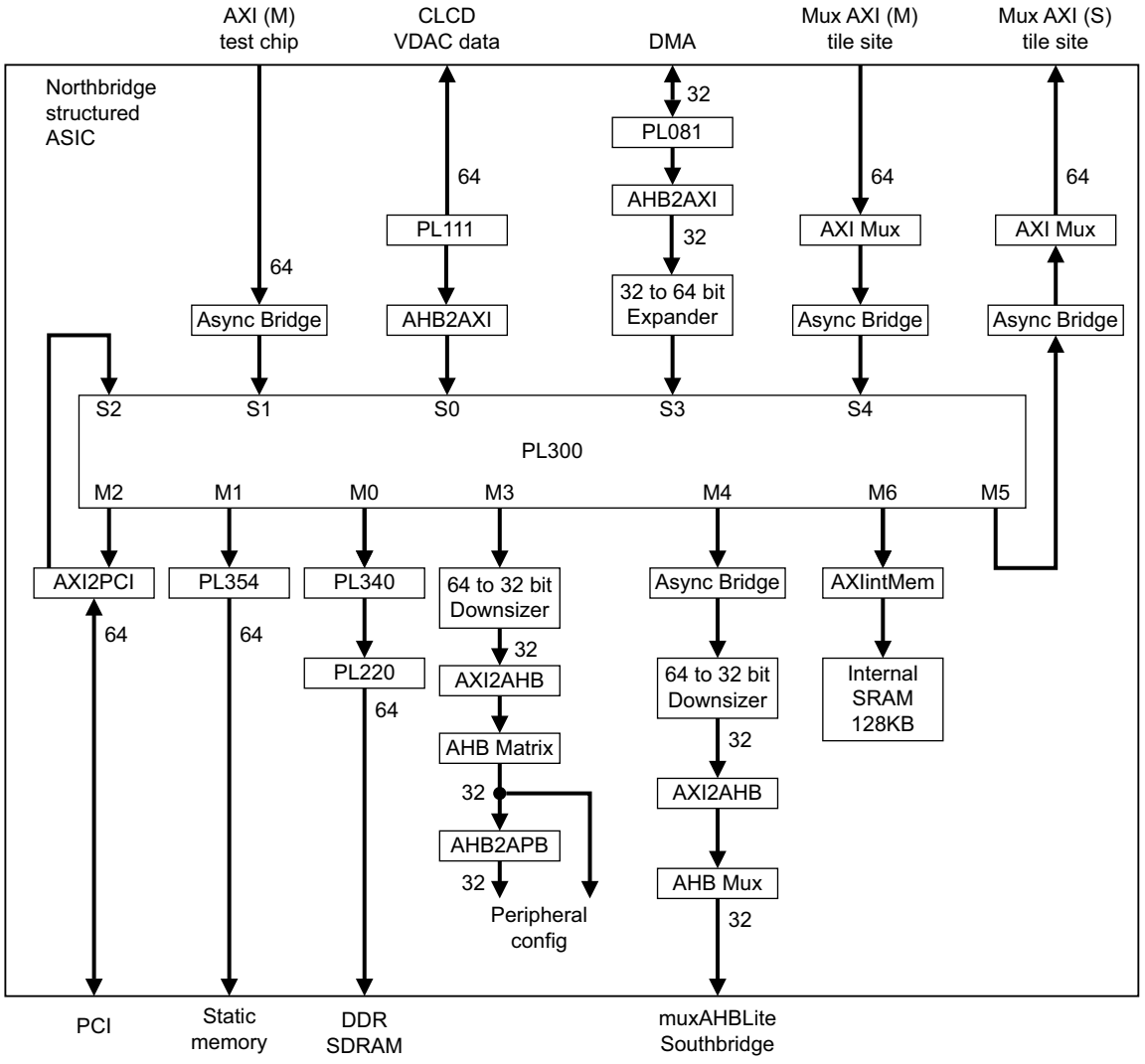


Figure 3-7 Northbridge block diagram

For details on the Northbridge see:

- *Cortex-A9 structured ASIC interface* on page 3-16

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

Note

For further information on the PrimeCells implemented in the Northbridge, see the *PrimeCell PLxxx Technical Reference Manual*.

3.4.1 Cortex-A9 structured ASIC interface

The Cortex-A9 structured ASIC interface to the Northbridge consists of an asynchronous non-multiplexed 64-bit master AXI bus to give the highest possible data transfer rate.

See *AMBA 3 AXI Protocol* (ARM IHI 0022) for details of the AXI interface for details of the Cortex-A9 structured ASIC interface.

3.4.2 CLCD controller

The PBX-A9 Northbridge implements the PrimeCell PL111 *Color LCD Controller*. This is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM.

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

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

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

See *Color LCD Controller, CLCDC* on page 4-47 for details of Cortex-A9 usage and the *ARM PrimeCell Color LCD Controller (PL111) Technical Reference Manual* (ARM DDI 0293) for full programming details.

3.4.3 Memory controllers

The PBX-A9 Northbridge implements memory controllers for:

- static memory
- dynamic memory
- direct memory access.

Static memory controller, SMC

The PBX-A9 Northbridge implements the PrimeCell PL354 *Static Memory Controller*, an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM.

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

In the Northbridge the PrimeCell PL354 supports:

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

See *Static Memory Controller, SMC* on page 4-93 for details of PBX-A9 usage and the *ARM PrimeCell Static Memory Controller (PL350 series) Technical Reference Manual* (ARM DDI 0380) for full programming details.

Dynamic memory controller, DMC

The PBX-A9 Northbridge implements the PrimeCell PL340 *Dynamic Memory Controller*, an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM.

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

In the Northbridge the PrimeCell PL354 supports:

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

Note

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

See *Dynamic Memory Controller, DMC* on page 4-53 for details of Cortex-A9 usage and the *ARM PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual* (ARM DDI 0331) for full programming details.

Single master direct memory access controller, SMDMAC

The PBX-A9 Northbridge implements the PrimeCell PL081 *Single Master DMA Controller*, an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral that is developed, tested, and licensed by ARM.

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

The Northbridge implementation the PrimeCell PL081 supports:

- two channels of direct memory access (with DMAC flow control only)
- two selectable DMA request line mappings, each to eight peripherals
- DMA access to the tile site

See *Single Master Direct Memory Access Controller, SMDMAC* on page 4-50 for details of Cortex-A9 usage and the *ARM PrimeCell Single Master DMA Controller (PL081) Technical Reference Manual* (ARM DDI 0218) for full programming details.

3.4.4 Multiplexed AHB-Lite interface

The PBX-A9 Northbridge connects to the CompactFlash interface and low speed peripherals in the Southbridge through a custom multiplexed AHB-Lite master port.

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

3.4.5 Multiplexed AXI interfaces

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

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

3.4.6 PCI interface

The PBX-A9 Northbridge implements an AXI to PCI bridge (AXI2PCI) that provides interface functions conforming to the PCI-X Protocol Addendum to the PCI Local Bus Specification Revision 2.0a.

See *AXI to PCI and PCI to PCIx bridges* on page 4-86 for details of PBX-A9 usage and user programming details.

3.5 Southbridge peripherals

Figure 3-8 shows the architecture of the PBX-A9 Southbridge. The Southbridge is implemented by ARM as a custom FPGA.

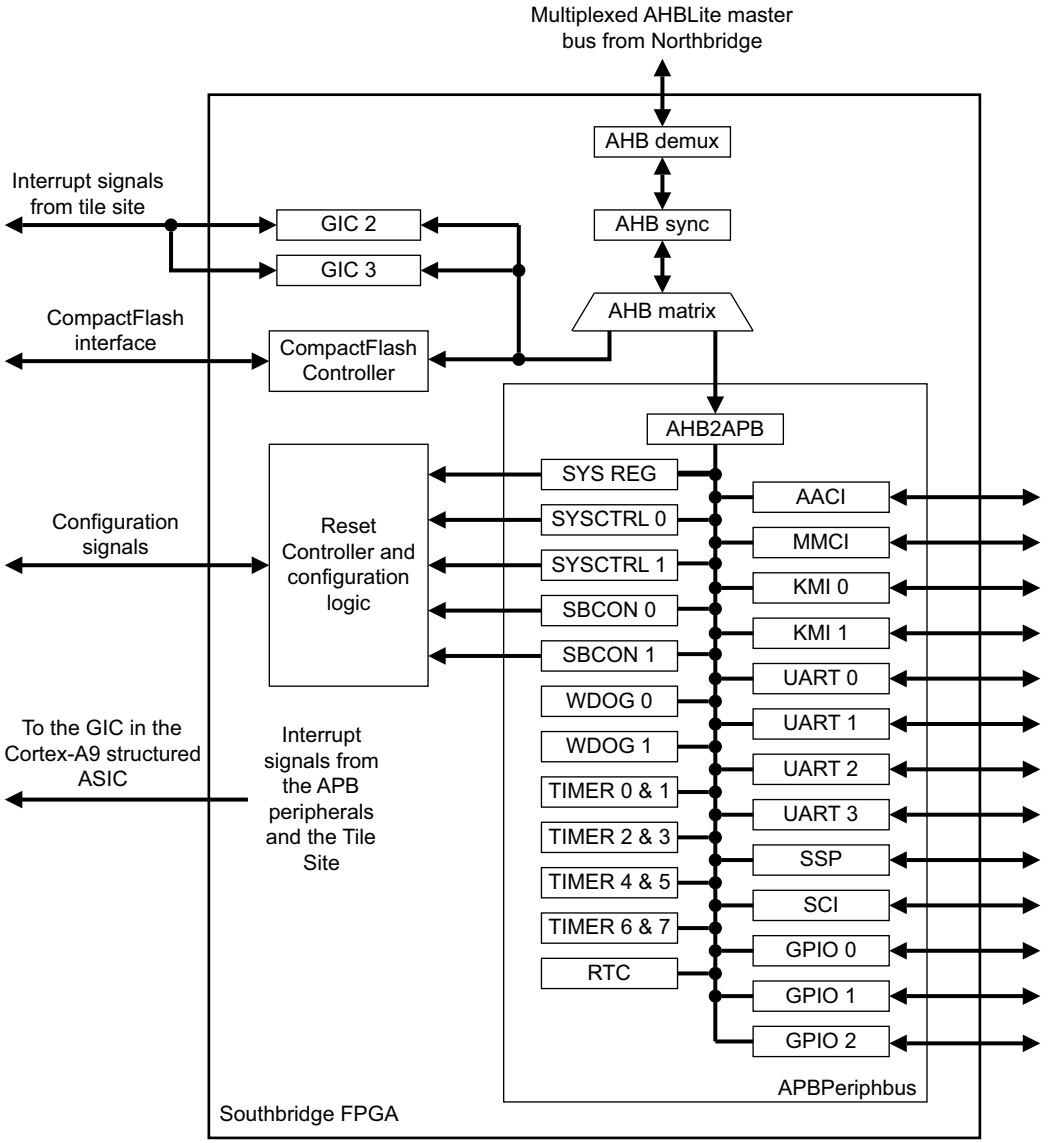


Figure 3-8 Southbridge block diagram

For details on the Southbridge see:

- *Reset controller*
- *CompactFlash*
- *APB peripherals*

3.5.1 Reset controller

Correct initialization of the PBX-A9 baseboard and an associated Logic Tile requires a timed reset sequence. The custom Reset controller in the Southbridge monitors the reset sources and sequences the baseboard and Logic Tile resets during power-up (power-on reset sequence) and during baseboard configuration (system reset sequence). See *Resets* on page 3-37 for details.

3.5.2 CompactFlash

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

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

See *CompactFlash interface* on page 4-100 for details of PBX-A9 usage. *CompactFlash interface* on page A-2 shows the physical interface provided by the PBX-A9 baseboard.

3.5.3 APB peripherals

The majority of the controllers and interfaces implemented in the PBX-A9 Southbridge are standard ARM *PrimeCell*® components:

- *Advanced Audio CODEC Interface, AACI* on page 3-22
- *Multimedia Card Interface, MCI* on page 3-22
- *Keyboard and Mouse Interface, KMI* on page 3-22
- *Watchdog Module* on page 3-23
- *Dual-Timer Module* on page 3-23
- *General Purpose Input/Output, GPIO* on page 3-23
- *Generic Interrupt Controllers* on page 3-24
- *Status and System Control Register Block* on page 3-24
- *System Controller* on page 3-24
- *UART* on page 3-25
- *Synchronous Serial Port, SSP* on page 3-25
- *Smart Card Interface, SCI* on page 3-26

- *Two-wire serial bus interface* on page 3-26
- *Real Time Clock, RTC* on page 3-27.

Advanced Audio CODEC Interface, AACI

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

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

See *Advanced Audio CODEC Interface, AACI* on page 4-45 for details of PBX-A9 usage and the *ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual* (ARM DDI 0173) for full programming details. *Audio CODEC interface* on page A-6 shows the physical interface provided by the PBX-A9 baseboard.

Multimedia Card Interface, MCI

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

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

See *MultiMedia Card Interface, MCI* on page 4-85 for details of PBX-A9 usage and the *ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual* (ARM DDI 0172) for full programming details. *MMC and SD card interface* on page A-7 shows the physical interface provided by the PBX-A9 baseboard.

Keyboard and Mouse Interface, KMI

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

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

See *Keyboard and Mouse Interface, KMI* on page 4-84 for details of PBX-A9 usage and the *ARM PrimeCell PS2 Keyboard/Mouse Interface (PL050) Technical Reference Manual* (ARM DDI 0143) for full programming details. *Keyboard and mouse interface* on page A-9 shows the physical interface provided by the PBX-A9 baseboard.

Watchdog Module

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

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

See *Watchdog* on page 4-99 for details of PBX-A9 usage and the *ARM Watchdog Module (SP805) Technical Reference Manual* (ARM DDI 0270) for full programming details.

Dual-Timer Module

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

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

See *Timers* on page 4-94 for details of PBX-A9 usage and the *ARM Dual-Timer Module (SP804) Technical Reference Manual* (ARM DDI 0271) for full programming details.

General Purpose Input/Output, GPIO

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

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

See *General Purpose Input/Output, GPIO* on page 4-57 for details of PBX-A9 usage and the *ARM PrimeCell General Purpose Input/Output (PL061) Technical Reference Manual* (ARM DDI 0190) for full programming details. *GPIO interface* on page A-10 shows the physical interface provided by the PBX-A9 baseboard.

Generic Interrupt Controllers

The *Generic Interrupt controllers* (GICs) are *Advanced Microcontroller Bus Architecture (AMBA)* slave blocks that connect to the *Advanced Peripheral Bus* (APB).

There are three GICs on the PBX-A9 baseboard:

- Two GICs in the Southbridge manage interrupt signals from peripherals and issue **nFIQ** and **nIRQ** to the Logic Tile. These are identical, customized versions of the ARM Generic Interrupt Controller architecture.
- A GIC in the Cortex-A9 structured ASIC manages interrupts from the Logic Tiles and from the peripherals in the Southbridge This GIC is a Cortex-A9 specific implementation of the ARM Generic Interrupt Controller. For more information, see the *ARM Generic Interrupt Controller Architecture Specification* (ARM IHI 0048).

The GICs can accept interrupts from multiple sources and generate **nFIQ** or **nIRQ** responses for the system.

See *Generic Interrupt Controller, GIC* on page 4-58 for PBX-A9 usage, and *Interrupts* on page 3-39 for the available routing options. For details of the GIC and the Programmer's Model see the *Cortex-A9 MPCore Technical Reference Manual*.

Status and System Control Register Block

The PBX-A9 baseboard status and system control registers enable the PBX-A9 to determine its environment and to control the on-board systems.

See *Status and system control registers* on page 4-12 for a description of each register.

System Controller

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

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

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

- system/peripheral clock control and status.

See *System Controller (SYSCTRL)* on page 4-42 for details of PBX-A9 usage and the *PrimeXsys System Controller (SP810) Technical Reference Manual* (ARM DDI 0254) for full programming details.

UART

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

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

Note

An IrDA SIR ENDEC interface is implemented but this is not supported by the PBX-A9.

See *UART* on page 4-95 for details of PBX-A9 usage and the *ARM PrimeCell UART (PL011) Technical Reference Manual* (ARM DDI 0183) for full programming details. *UART interface* on page A-11 shows the physical interface provided by the PBX-A9 baseboard.

Synchronous Serial Port, SSP

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

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

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

See *Synchronous Serial Port, SSP* on page 4-92 for details of PBX-A9 usage and the *ARM PrimeCell Synchronous Serial Port (PL022) Technical Reference Manual* (ARM DDI 0194) for full programming details. *Synchronous Serial Port interface* on page A-12 shows the physical interface provided by the PBX-A9 baseboard.

Smart Card Interface, SCI

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

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

See *Smart Card Interface, SCI* on page 4-91 for details of PBX-A9 usage and the *ARM PrimeCell Smart Card Interface (PL131) Technical Reference Manual* (ARM DDI 0228) for full programming details. *Smart Card interface* on page A-13 shows the physical interface provided by the PBX-A9 baseboard.

Two-wire serial bus interface

The FPGA implements a custom two-wire serial bus interface that is used to read and set the *Time-Of-Year* (TOY) clock on the baseboard. A second interface is used to identify and control display equipment connected to the DVI connector on the rear panel of the ATX enclosure.

Each device on the serial bus has its own slave address. Table 3-1 lists the unique write and read addresses for the TOY slave on the serial bus. The address for the DVI display slave is device dependant and must be obtained from the display manufacturer.

Table 3-1 Serial interface device addresses

Device	Write address	Read address	Description
TOY (DS1338 RTC)	`0xD0`	`0xD1`	Reads time data and writes control data to the RTC.
DVI (external display)	display dependant	display dependant	Reads the capabilities of the external display connected to the DVI connector on the rear panel of the ATX enclosure. Can control display settings of *E-DDC* displays.

See *Two-wire serial bus interface, SBCon* on page 4-89 for details of PBX-A9 usage and for more information on programming the interface.

Real Time Clock, RTC

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

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

See *Real Time Clock, RTC* on page 4-88 for details of PBX-A9 usage and the *ARM PrimeCell Real Time Clock (PL031) Technical Reference Manual* (ARM DDI 0224) for full programming details.

3.6 Ethernet interface

The Ethernet interface on the PBX-A9 baseboard is implemented using a SMCS LAN9118 10/100 Ethernet controller. The LAN9118 incorporates a *Media ACcess* (MAC) Layer, a *PHYsical* (PHY) layer, *Host Bus Interface* (HBI), receive and transmit FIFOs, power management controls, and a serial configuration EEPROM interface. The HBI models an asynchronous SRAM and interfaces directly to the Northbridge static memory bus.

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

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

See *Ethernet* on page 4-56 for details of PBX-A9 usage and the *SMC LAN9118 Data Sheet* for full programming details.

3.7 USB Interface

The PBX-A9 Northbridge provides a SMC bus interface to an external Philips ISP1761 USB 2.0 controller. Three USB interfaces are provided on the baseboard. The internal registers of the controller are memory-mapped starting at `0x4F000000`.

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

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

Note

The USB config interface has a dedicated USB controller and connects to the USB B-Type connector on the front panel of the enclosure. *Debug and Config port support* on page 3-47 for details.

3.8 DVI Interface

The PL111 CLCD controller in the PBX-A9 Northbridge is interfaced to the *Digital Visual Interface* (DVI) port on the PBX-A9 using on-board components to provide support for both analog and digital displays. See *CLCD controller* on page 3-16 for details.

The digital portion of the interface is provided by a Silicon Image SiI 1160 *Transition Minimized Differential Signaling* (T.M.D.S.) transmitter. The device is VESA compliant, operating in One Pixel/Clock Input/Output mode and supporting displays up to UXGA resolution.

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

The DDC2B portion of the interface is provided by a custom *Two-wire Serial Interface* (SBCon) implemented the Southbridge. See *Two-wire serial bus interface* on page 3-26 for details.

DVI display interface on page A-17 shows the physical interface provided by the PBX-A9 baseboard.

Note

- The on-board DVI interface can be controlled either by the PL111 *Color LCD Controller* implemented in the Northbridge, or by logic implemented in the FPGA on an attached Logic Tile. See *Tile interconnections* on page 3-10 and *RealView Logic Tile header connectors* on page A-19 for details.
- The PL111 CLCD controller output signal **CLLE** from the PBX-A9 Northbridge is not implemented.

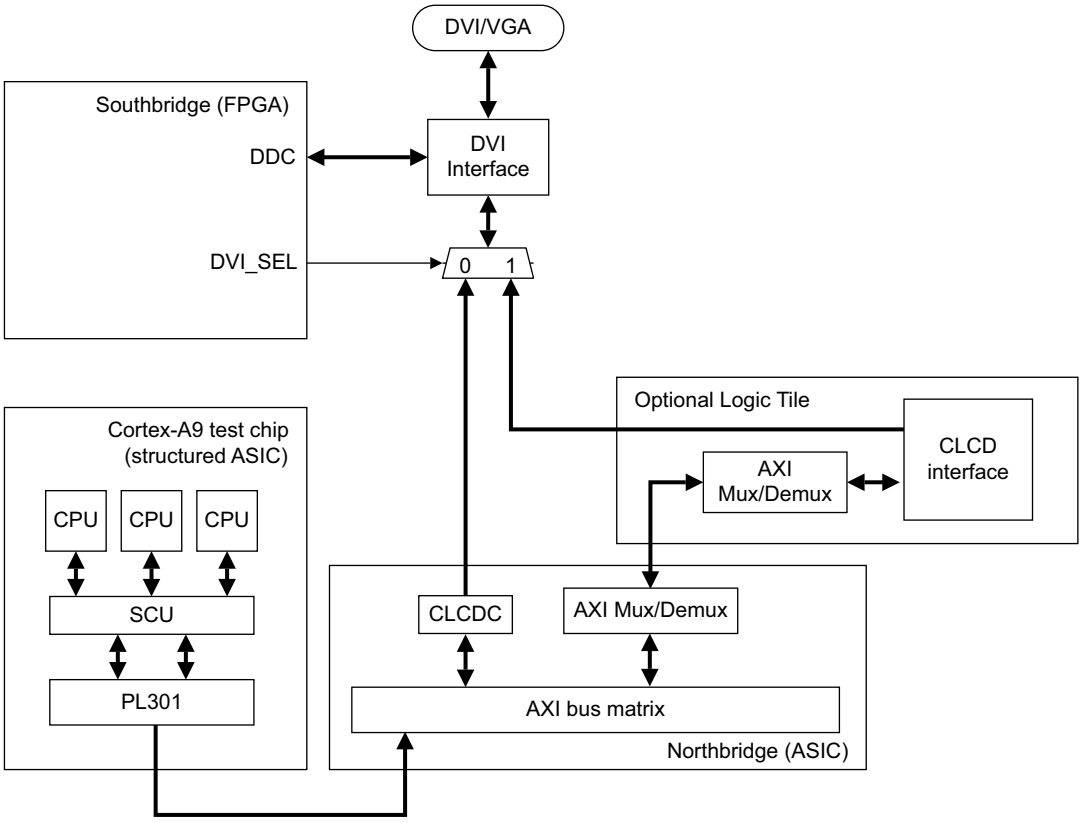


Figure 3-9 DVI output

Signals **CLCP** analog, and **CLCP** digital, shown as pins 237 and 249 in Table A-12 on page A-33 replace the pin names Z31 and Z25 to be specific to the baseboard. These are output signals from the LCD panel clock. See the Signal Descriptions chapter in the *PrimeCell Color LCD Controller (PL111) Technical Reference Manual*.

When **DVI_SEL** is LOW, both the video DAC IC, for the analog VGA output signals within the DVI connector, and the DVI IC, for the digital video output signals on the DVI connector, obtains the **CLCP** clock signal from the same source, an output signal on the Northbridge ASIC.

When **DVI_SEL** is HIGH, the video DAC obtains the **CLCP** signal from the HDRZ pin number 237, and the digital video IC obtains its clock signal from the HDRZ pin number 249.

3.9 PCI interface

The PBX-A9 PCI interface comprises of:

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

Figure 3-10 shows the PBX-A9 PCI and PCI Express implementation.

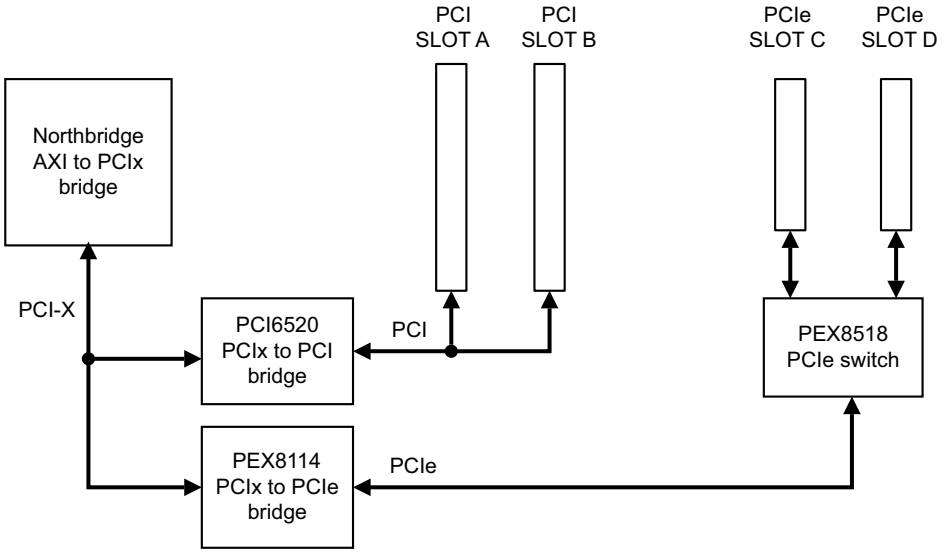


Figure 3-10 PCI-PCI Express interface

The AXI to PCIx bridge provides interface functions conforming to the PCI-X Specification.

The PCI-X to PCI asynchronous bridge (PCI6520) interfaces the PCI bridge to the PBX-A9 PCI slots.

The PCI-X to PCI Express asynchronous bridge (PEX8114) and PCI Express switch (PEX8518) interface the PCI bridge to the PBX-A9 PCI Express slots.

See *AXI to PCI and PCI to PCIx bridges* on page 4-86 for details of PBX-A9 usage and the *PCI-to-PCI Bridge Architecture Specification Revision 1.2* and the *PCI Express Base Specification Revision 1.1* for further details.

For details on the bridge and switch components used on the PBX-A9 see the PLX Technology Inc website: `www.plxtech.com`.

3.10 Clock architecture

This section describes the clock signals present on the PBX-A9 baseboard. Figure 3-11 shows the clock domains.

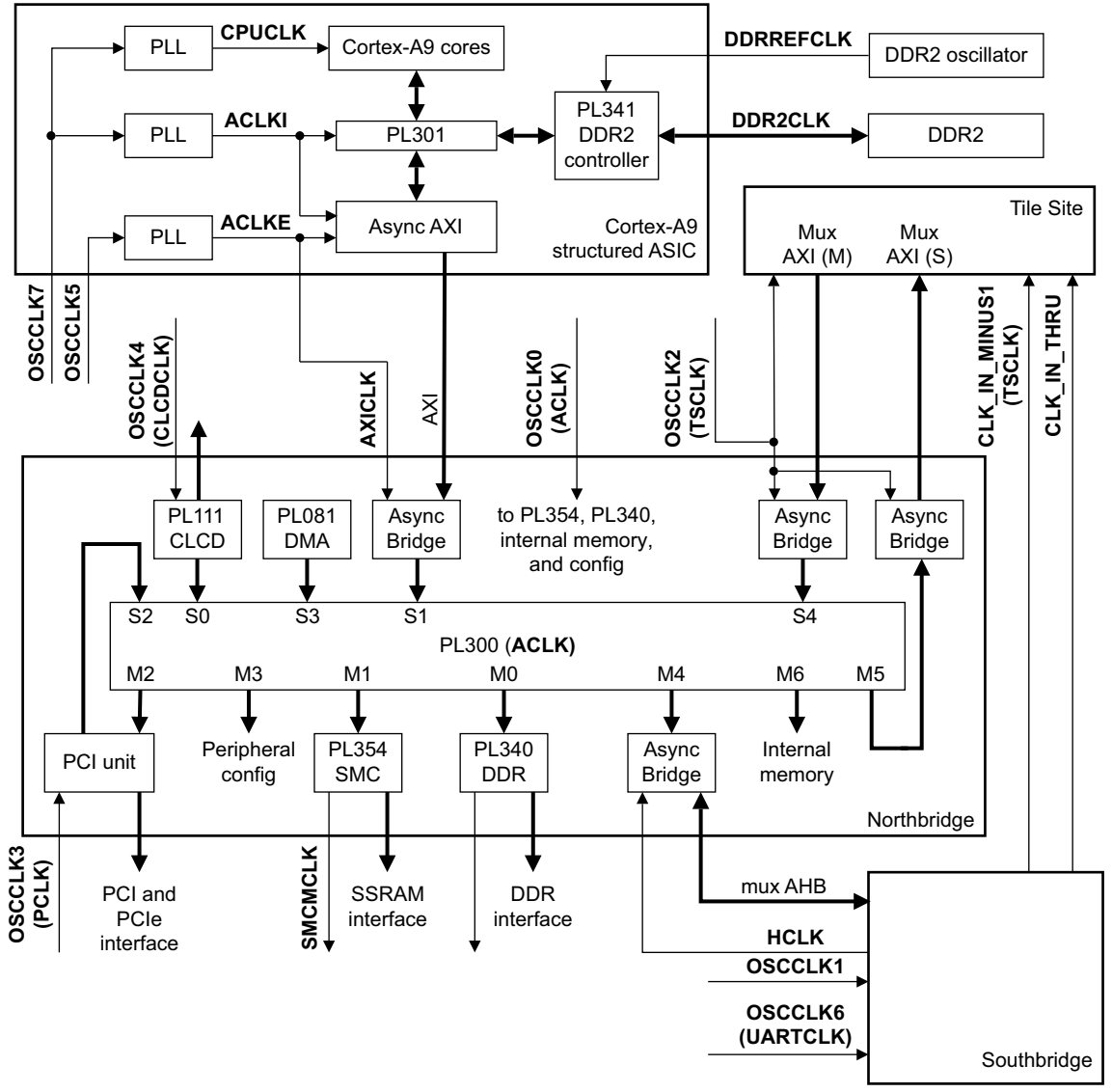


Figure 3-11 Northbridge clock domains

The Cortex-A9 structured ASIC includes PLLs and clock dividers.

Note

The PLLs and clock dividers are fixed. You cannot change the ratios. However, you can change the **OSCCLKx** clocks.

These components generate the clocks required by the Cortex-A9 processors and subsystems and the external AXI clock **AXICLK** required by the Northbridge AXI Async Bridge.

3.10.1 ICS307 programmable clock generators

Eight programmable (6–200 MHz) clocks on the PBX-A9 baseboard, **OSCCLK[7:0]**, are supplied by the programmable MicroClock ICS307 clock generators (OSC0–OSC7). The clock frequencies refer to reset values:

OSCCLK0 This **100MHz** clock generates the Northbridge **ACLK** that is the reference for:

- AXI infrastructure
- PL340 Dynamic Memory Controller
- Static Memory Controller (**SSMCCLK** = **OSCCLK0** / 2)
- PL081 DMA Controller
- internal memory
- peripheral configuration.

ACLK = **OSCCLK0**

This clock must not be changed from the default value set by ARM during manufacture.

OSCCLK1 This **33MHz** clock generates **HCLK** for the AHB interface between the Northbridge and Southbridge.

HCLK = **OSCCLK1**

This clock must not be changed from the default value set by ARM during manufacture.

OSCCLK2 This **25MHz** clock is the **TSCLK** reference for **CLK_IN_MINUS1** and **CLK_IN_THRU** for distribution to the tile site.

This clock can be changed to meet the clocking requirements of a Logic Tile fitted to the tile site.

OSCCLK3 This **50MHz** clock generates **PCLK** the reference clock for the PCI unit in the Northbridge, and the external PCI and PCI Express bridges.

This clock must not be changed from the default value set by ARM during manufacture.

OSCCLK4 This **25MHz** clock generates **CLCDCLK** the reference clock for the PL110 CLCD controller in the Northbridge, and the external video DAC and DVI transmitter (1024x768 resolution at **60Hz** frame rate support).

This clock can be changed to meet the clocking requirements of the LCD display.

OSCCLK5 This **14MHz** clock generates **ACLKE** for the Cortex-A9 structured ASIC and Northbridge AXI interface.

The default and maximum supported **ACLKE** is **70MHz** (**OSCCLK5** = **14MHz**).

ACLKE = **OSCCLK5** * 5

The allowable range of adjustment for **ACLKE** is **50MHz − 70MHz**. This limits the range of adjustment of **OSCCLK5** to **10MHz − 14MHz**.

OSCCLK6 This **24MHz** clock is divided by 24 to generate the **1MHz UARTCLK** the reference clock for the PL011 UART in the Southbridge.

This clock must not be changed from the default value set by ARM during manufacture.

OSCCLK7 This **14MHz** clock generates the **CPUCLK** and **ACLKI** references to the Cortex-A9 structured ASIC.

The default and maximum supported **CPUCLK** and **ACLKI** is **70MHz** (**OSCCLK7** = **14MHz**)

CPUCLK = **OSCCLK7** * 5

The allowable range of adjustment for **CPUCLK** is **50MHz − 100MHz**. This limits the range of adjustment of **OSCCLK7** to **10MHz − 20MHz**.

Note

- Before writing to a SYS_OSCx register, you must unlock it by writing the value `0x0000A05F` to the SYS_LOCK register. After writing the SYS_OSC register, you must relock it by writing any value other than `0x0000A05F` to the SYS_LOCK register. See also *Oscillator Registers, SYS_OSCx* on page 4-18.
- ARM recommends not to make changes to these registers at runtime, instead use the **SYS_OSCRESETx** registers followed by a software reset. See *Oscillator reset registers, SYS_OSCRESETx* on page 4-40.
- Because not all permutations of clock frequencies between **ACLKE** and **CPUCLK** have been tested, ARM cannot guarantee the correct behavior of the system for any particular combinations of clock settings that you might want to experiment with, except for the default ones.

Setting the programmable oscillator frequency

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

The serial interface logic is implemented in the Southbridge. The only user interface to the clock control logic is through the SYS_OSCx registers. See *Oscillator Registers, SYS_OSCx* on page 4-18 and *Oscillator reset registers, SYS_OSCRESETx* on page 4-40 for programming details.

You can calculate the oscillator output frequency from the formula:

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

where:

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

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

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

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

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

A crystal on the board provides a fixed frequency 24MHz reference clock for the programmable oscillators OSC0-OSC7 and to generate other fixed frequency clocks in the design

3.10.2 Southbridge clocks

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

By default, the Dual Timer Counter modules and the Watchdog modules are clocked by a 32.768Khz reference clock.

3.11 Resets

Figure 3-12 shows the reset sequence for the PBX-A9 baseboard. See *Reset Control Register, SYS_RESETCTL* on page 4-22 for details on setting the SW_RESET level.

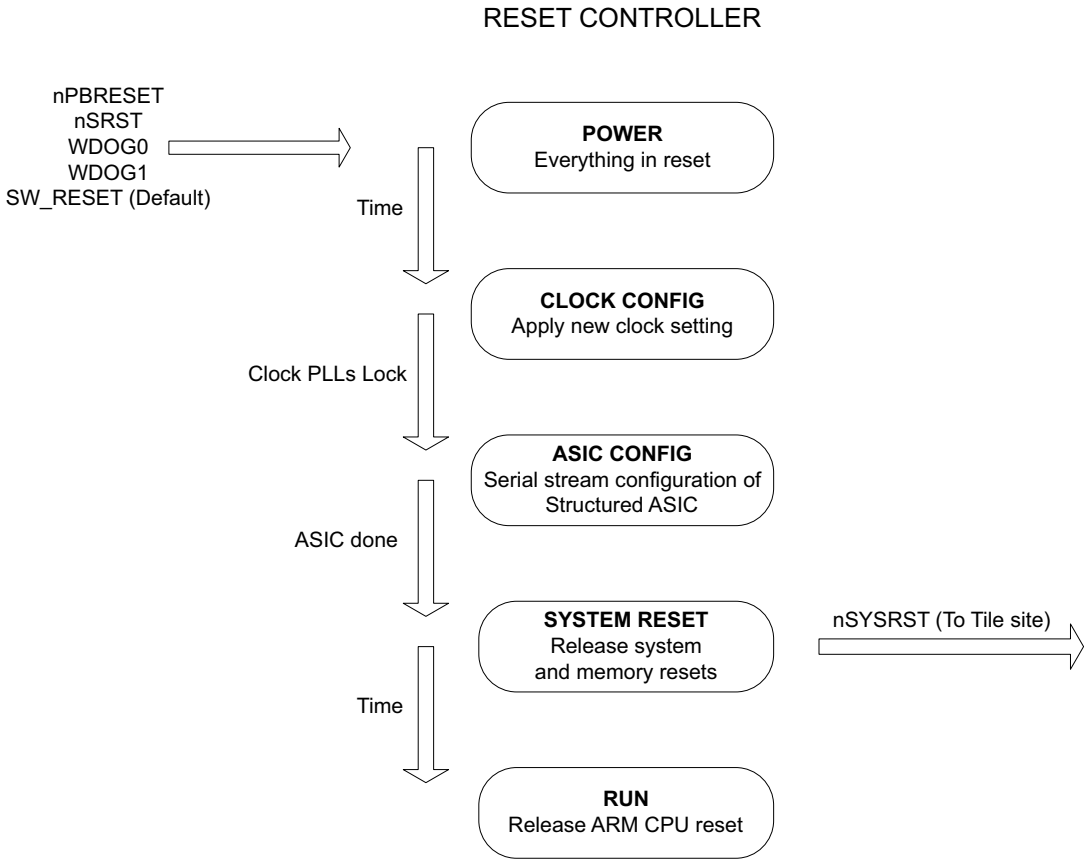


Figure 3-12 Reset sequence

The resets are described in Table 3-2.

Table 3-2 Reset signals

Name	Source	Direction	Description
nPBRESET	Push Button	To reset controller	Manual Reset
nSRST	Daughterboard	To reset controller	System level reset
WDOG0	Watchdog 0	To reset controller	Watchdog
WDOG1	Watchdog 1	To reset controller	Watchdog
SW_RESET	SYS_RESETCTL	To reset controller	Software Reset. See *Reset Control Register, SYS_RESETCTL* on page 4-22 for details.
nSYSRST	Reset Controller	To tile site	Main reset to the tile site

Note

If a peripheral implemented on a Logic Tile fitted to the tile site is required to generate a system level reset, it can pulse the **nSRST** signal low. This causes the **nSYSRST** signal to pulse low resetting the system. This is a system level reset and does not change the clock or Structured ASIC configuration settings.

3.12 Interrupts

Interrupts from the Cortex-A9 structured ASIC, the PBX-A9 baseboard, and the tile site, are routed directly to an interrupt controller in the Cortex-A9 CPU cluster. The first 32 interrupt numbers (0-31) are reserved for interrupts from the Cortex-A9 structured ASIC, the remaining 64 (32-95) are allocated to the PBX-A9 baseboard and tile site peripherals.

The PBX-A9 also implements three *Generic Interrupt Controllers* (GICs). Interrupts from the PBX-A9 baseboard and the tile site are routed to these GICs. Each GIC prioritizes and distributes the interrupts and generates either **nIRQ** or **nFIQ** requests for the tile site.

Figure 3-13 shows the interrupt routing.

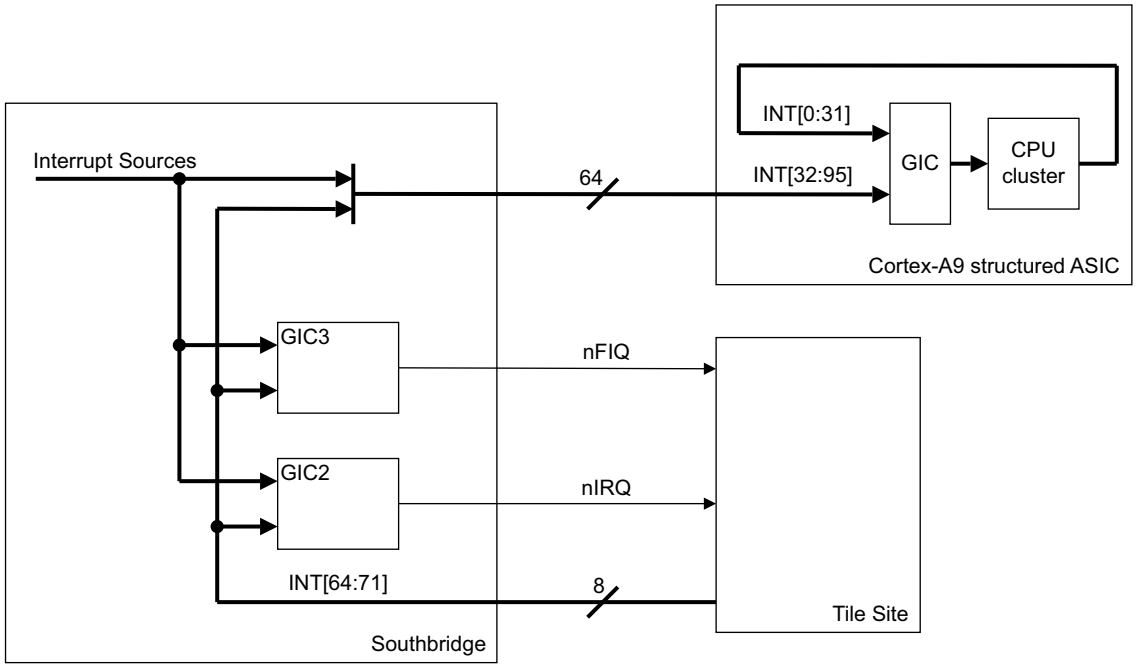


Figure 3-13 External and internal interrupt routing

3.12.1 Cortex-A9 structured ASIC interrupt controller

The interrupt controller in the Cortex-A9 CPU cluster distributes interrupts to the individual processors in the Cortex-A9 MPCore from a large number of sources and provides:

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

The interrupt controller control registers are accessed through the Cortex-A9*Snoop Control Unit* (SCU).

The base address is fixed by the PBX-A9 at `0x1F001000` and the address range is `0x1F001000` to `0x1F001FFF`.

See the *Cortex-A9 MPCore Technical Reference Manual* (DDI 0407) for register details.

3.12.2 PBX-A9 Southbridge interrupt controllers

The PBX-A9 interrupt controllers consist of two custom *Generic Interrupt Controllers* (GICs).The custom GIC is an AMBA compliant SoC peripheral that is developed and tested by ARM.

The GICs accept interrupts from peripherals in the Northbridge, Southbridge, on-board peripherals, and the tile site:

GIC2 generates the tile site **nIRQ**

GIC3 generates the tile site **nFIQ**

Note

GIC0 and GIC1 are not used in this implementation. GIC2 and GIC3 are for tile site use only.

The GIC allows you to:

- enable or disable an individual interrupt
- disable all interrupts below a given priority
- configure interrupt priority
- configure pre-emption.

The GIC holds three states for each interrupt source:

Inactive not asserted

Pending has been asserted, but is not yet complete
Active processing has started but is not yet complete.

Each GIC contains:

- a distributor
- a CPU interface.

Note

In the PBX-A9 implementation only one distributor and one CPU interface is implemented per GIC. In a multiprocessor system, such as the PB11MPCore, a distributor and multiple CPU interfaces are implemented, one CPU interface for each processor.

Distributor

The Distributor centralizes all interrupt sources and provides the highest priority interrupt to the corresponding CPU interface. Interrupts with a lower priority are forwarded to the appropriate CPU interface when they have attained the highest priority.

CPU interface

The CPU interface contains a progammable interrupt priority mask and a binary point mask. It only accepts pending interrupts that have a priority higher than the level set in the priority mask, the binary point mask, and a priority higher than those that the CPU is currently servicing.

Note

The binary point mask is a novel feature of the GIC that allows you to reduce the amount of pre-emption in the system and effectively acts as a *pre-emption mask*.

Calculating the next interrupt

The algorithm use by the GIC to calculate which interrupt to service next is described by the pseudo code:

```
if highest pending priority > priority mask
    if no interrupt currently being processed
        issue highest priority pending interrupt
    else if binary point mask calculation > running priority
        pre-empt with highest priority pending interrupt
    end if
end if
```

A diagram of the algorithm is given in Figure 3-14 on page 3-43.

Note

The finite state machine that Figure 3-14 on page 3-43shows is repeated for each interrupt source, and for each processor in the system.

For interrupt allocations see *Interrupt allocations* on page 3-44, for GIC programming details see *Generic Interrupt Controller, GIC* on page 4-58.

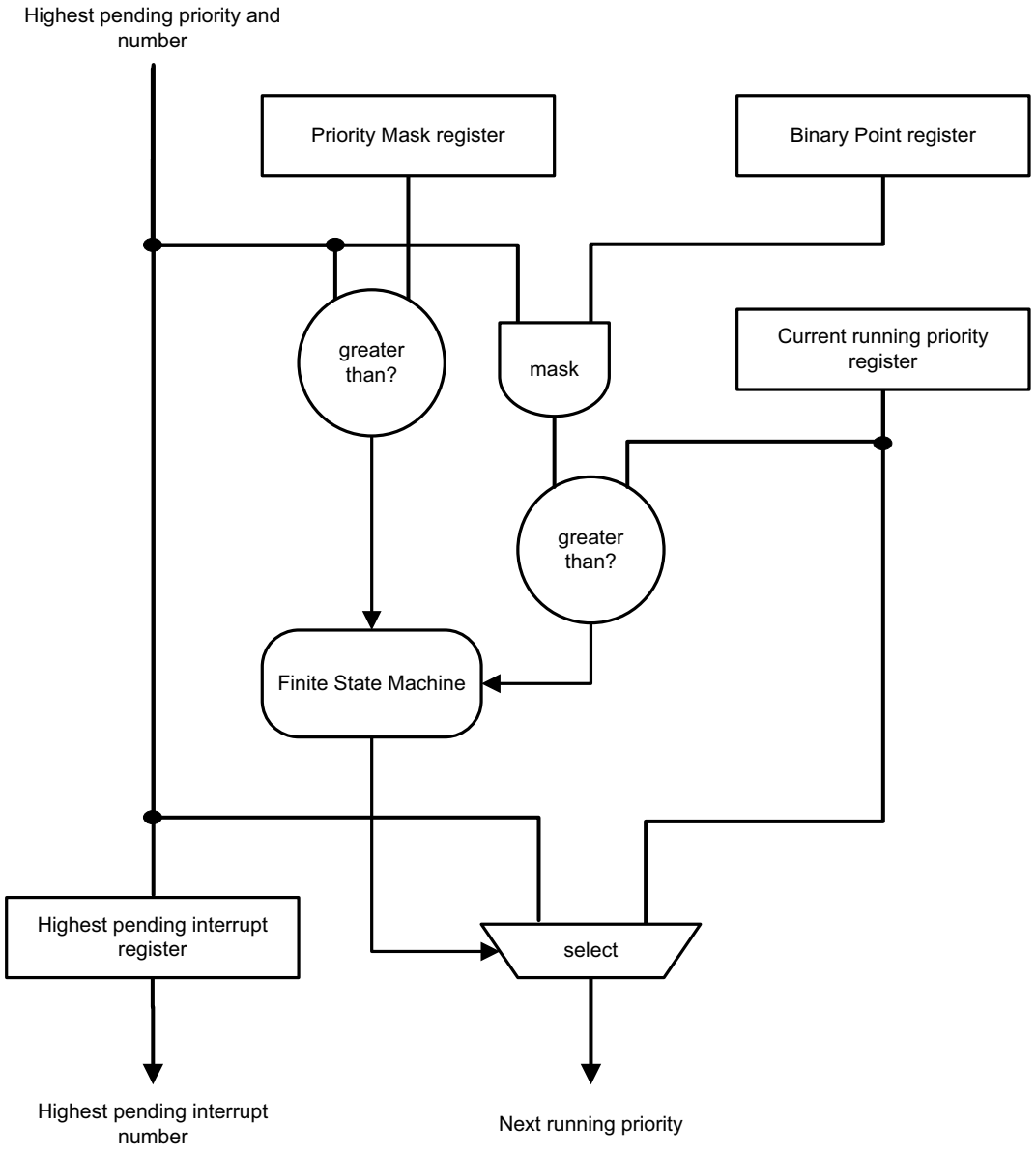


Figure 3-14 Interrupt priority calculation

3.12.3 Interrupt Controller routing

Table 3-3 lists the PBX-A9 interrupt signal allocation.

Table 3-3 Interrupt allocations

Peripheral	Source	GIC2 and GIC3 ID (tile site)	GIC ID (Cortex-A 9Core)	Description
Cortex-A9 MPCore daughterboard	Cortex-A9 structured ASIC	–	[31:0]	Generated by the Cortex-A9 CPU cluster
PBX-A9 baseboard	PBX-A9	[32]	–	Reserved for the PBX-A9
Watchdog0	System FPGA	[32]	[32]	Watchdog timer 0
S/W interrupt	System FPGA	[33]	[33]	Software Interrupt
Reserved	–	[34]	[34]	–
Reserved	–	[35]	[35]	–
Timer 0-1	Southbridge	[36]	[36]	Timer 0 and 1
Timer 2-3	Southbridge	[37]	[37]	Timer 2 and 3
GPIO 0	Southbridge	[38]	[38]	GPIO controller
GPIO 1	Southbridge	[39]	[39]	GPIO controller
GPIO 2	Southbridge	[40]	[40]	GPIO controller
Reserved	-	[41]	[41]	–
RTC	Southbridge	[42]	[42]	Real time clock
SSP	Southbridge	[43]	[43]	Synchronous serial port
UART0	Southbridge	[44]	[44]	UART 0
UART1	Southbridge	[45]	[45]	UART 1
UART2	Southbridge	[46]	[46]	UART 2
UART3	Southbridge	[47]	[47]	UART 3
SCI	Southbridge	[48]	[48]	Smart Card interface
MCIa	Southbridge	[49]	[49]	Multimedia Card interface interrupt a
MCIb	Southbridge	[50]	[50]	Multimedia Card interface interrupt b

Table 3-3 Interrupt allocations (continued)

Peripheral	Source	GIC2 and GIC3 ID (tile site)	GIC ID (Cortex-A 9Core)	Description
AACI	Southbridge	[51]	[51]	CODEC controller interrupt
KMI0	Southbridge	[52]	[52]	Keyboard/Mouse interface 0
KMI1	Southbridge	[53]	[53]	Keyboard/Mouse interface 1
Reserved	–	[54]	[54]	Character LCD (not supported)
CLCD	Northbridge	[55]	[55]	CLCD display
DMAC	Northbridge	[56]	[56]	DMA controller
PWRFAIL	Board	[57]	[57]	Power Fail
Reserved	–	[58]	[58]	PISMO (not supported)
CF_INT	Southbridge	[59]	[59]	CompactFlash interface (not supported)
Ethernet	Board	[60]	[60]	Ethernet controller
USB	Board	[61]	[61]	USB controller
Reserved	–	[62]	[62]	–
Reserved	–	[63]	[63]	–
T1_INT0	Tile site	[64]	[64]	Interrupts from tile site
T1_INT1	Tile site	[65]	[65]	
T1_INT2	Tile site	[66]	[66]	
T1_INT3	Tile site	[67]	[67]	
T1_INT4	Tile site	[68]	[68]	
T1_INT5	Tile site	[69]	[69]	
T1_INT6	Tile site	[70]	[70]	
T1_INT7	Tile site	[71]	[71]	
Watchdog1	System FPGA	[72]	[72]	Watchdog 1 alarm
Timer 4-5	System FPGA	[73]	[73]	Timer 4 and 5

Table 3-3 Interrupt allocations (continued)

Peripheral	Source	GIC2 and GIC3 ID (tile site)	GIC ID (Cortex-A 9Core)	Description
Timer 6-7	System FPGA	[74]	[74]	Timer 6and 7
Reserved	–	[75]	[75]	–
Reserved	–	[76]	[76]	–
Reserved	–	[77]	[77]	–
Reserved	–	[78]	[78]	–
Reserved	–	[79]	[79]	–
PCI_INTR	Southbridge	[80]	[80]	PCI controller INT
P_NMI	Southbridge	[81]	[81]	PCI controller NMI
P_nINT[0]	Board	[82]	[82]	PCI SLOT A **P_nINTD** or PCI SLOT B **P_nINTC**
P_nINT[1]	Board	[83]	[83]	PCI SLOT A **P_nINTA** or PCI SLOT B **P_nINTD**
P_nINT[2]	Board	[84]	[84]	PCI SLOT A **P_nINTB** or PCI SLOT B **P_nINTA**
P_nINT[3]	Board	[85]	[85]	PCI SLOT A **P_nINTD** or PCI SLOT B **P_nINTB**
P_nINT[4]	Board	[86]	]86]	PCIe bridge **P_nINTA**
P_nINT[5]	Board	[87]	[87]	PCIe bridge **P_nINTB**
P_nINT[6]	Board	[88]	[88]	PCIe bridge **P_nINTC**
P_nINT[7]	Board	[89]	[89]	PCIe bridge **P_nINTD**
Reserved	–	[90]	[90]	–
Reserved	–	[91]	[91]	–
Reserved	–	[92]	[92]	–
Reserved	–	[93]	[93]	–
Reserved	–	[94]	[94]	–
Reserved	–	[95]	[95]	–

3.13 Test, configuration, and debug interfaces

The following test and configuration interfaces are located on the PBX-A9 baseboard:

- configuration switches, see *Baseboard configuration switches* on page 2-8
- JTAG, see *Debug and Config port support*
- Logic analyzer, see *Integrated logic analyzer (ILA)* on page 3-50
- Boot Monitor, see *Using the baseboard Boot Monitor and platform library* on page D-4.

3.13.1 Debug and Config port support

The PBX-A9 baseboard supports debugging and configuration using embedded and external hardware:

Debug interface

The JTAG connector on the rear panel connects to the debug interface JTAG connector on the daughterboard. The RealView Debugger, for example, can be connected to this debug interface using an external RealView ICE interface box. See *JTAG connector* on page A-40 for pinout details.

Configuration interface

The configuration interface is controlled by either the JTAG connector on the baseboard or the USB config port.

Note

The JTAG connector on the baseboard only controls the config interface, you must use the JTAG connector either on the rear panel or on the daughterboard to control the debug interface.

An ARM custom PLD design provides access to the internal configuration JTAG signals from a dedicated USB config port.

A USB B-Type interface connector is provided on the front panel of the enclosure for USB config access by the host PC.

An application, `Progcards USB` controls the JTAG config signals from the USB port of the PC when the PC is connected to the USB config port on the front panel by a standard USB cable. See *USB debug connector* on page A-41 for pinout details. See Appendix F *Loading FPGA Images* for details on using the USB config port to load images.

Note

Connecting an ARM RealView ICE to the JTAG connector on the baseboard automatically disables the USB config port on the front panel.

JTAG debug (normal) mode

During normal operation and software development, the PBX-A9 baseboard operates in debug mode.

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

In debug mode:

- The CONFIG LED is off on the front panel (and on each tile in the stack).
- The JTAG signals are routed through the Cortex-A9 structured ASIC.
- The JTAG debug scan path is rerouted sequentially to:
  - Logic Tile debug scan chain (**D_x** signals)
  - Southbridge debug scan chain (**D_x** signals)
  - Cortex-A9 debug unit scan chains.
- A debugger (RealView Debugger for example), connected to the JTAG ICE connector on the rear panel, controls the scan chain.
- An Integrated Logic Analyzer (ILA) connected to the ChipScope connector (J9) on the baseboard can be used to debug the FPGAs on stacked tiles while the debugger is examining code on the Cortex-A9 structured ASIC. The signals are routed through the Logic Tile configuration scan chain (**C_x** signals).
- the FPGAs in the system load their images from configuration flash.

JTAG configuration mode

This mode is selected when the CONFIG switch on the front panel is ON. See Figure 3-2 on page 3-4 for the location of the CONFIG switch.

Caution

The JTAG connectors on the rear panel and on the daughterboard are not connected to the config scan chain. You must use the USB config on the front panel, or the JTAG connector on the baseboard, for configuration.

If a RealView ICE is connected to the baseboard JTAG connector, the USB config port is automatically disabled.

In configuration mode:

- the CONFIG LED is lit on the front panel (and on each tile in the stack)
- the JTAG configuration scan path is rerouted sequentially to:
  - Logic Tile configuration scan chain (**C_x** signals)
  - Southbridge configuration scan chain (**C_x** signals)
  - Configuration PLD scan chain
  - Chip Select PLD scan chain
  - Northbridge scan chain
  - Cortex-A9 structured ASIC boundary scan chain.
- the ARM configuration utility, `Progcards`, is used to control the scan chain
- the debug unit in the PBX-A9 is not visible and is replaced by the boundary scan chain in the Cortex-A9 structured ASIC that is used for board-level production testing
- the board can be configured or upgraded in the field using JTAG equipment or the onboard USB config port
- the non-volatile PLD devices can be reprogrammed directly by JTAG
- the volatile FPGA images can be loaded directly by JTAG.

Note

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

Caution

Third party JTAG debug equipment might not support this method of programming configuration flash. Where possible, use the ARM RealView Debugger.

After configuration you must:

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

3.13.2 Integrated logic analyzer (ILA)

See *Integrated Logic Analyzer (ILA)* on page A-41 for pinout details. For more details on the integrated logic analyzer, see the ChipScope details on the Xilinx website (www.xilinx.com).

3.13.3 Embedded trace support

The Cortex-A9 structured ASIC incorporates a *ProgramFlow Trace Macrocell* (PTM), an 8KB trace buffer, and a *Trace Port Interface Unit* (TPIU). This enables you to carry out real-time debugging by connecting external trace equipment to the Trace connectors on the PBX-A9 daughterboard. For more information, see the *CoreSight PTM-A9 Technical Reference Manual*.

A trace connector is also provided for the Logic Tile. Use the Logic Tile trace connector to either:

- monitor trace activity in a processor implemented in the Logic Tile FPGA
- monitor signal activity on the header connector.

Chapter 4
Programmer's Reference

This chapter describes the PBX-A9 memory map and the configuration registers for the baseboard peripherals and controllers. It contains the following sections:

- *Real Time Clock, RTC* on page 4-88
- *Two-wire serial bus interface, SBCon* on page 4-89
- *Smart Card Interface, SCI* on page 4-91
- *Synchronous Serial Port, SSP* on page 4-92
- *Static Memory Controller, SMC* on page 4-93
- *System Controller (SYSCTRL)* on page 4-42
- *Timers* on page 4-94
- *USB interface* on page 4-97
- *UART* on page 4-95
- *USB interface* on page 4-97
- *Watchdog* on page 4-99
- *CompactFlash interface* on page 4-100.

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 Southbridge are in the standard images distributed by ARM on the Versatile Family CD.
- ARM do not recommend or support replacing the standard ARM IP peripherals and controllers implemented in the Southbridge with your own custom designs.
- Custom IP development must be done using an attached Logic Tile for which ARM support is provided.

4.1 Memory map

The PBX-A9 system memory map is divided with sections assigned to the Northbridge, Southbridge, and the Logic Tile site as Table 4-1 shows.

Table 4-1 System memory map

Owner	Address range	Bus type	Memory region size
Northbridge	0x00000000-0x0FFFFFFF	DDR	256MB (DMC mirror)
Southbridge	0x10000000-0x1001FFFF	APB	128KB
Northbridge	0x10020000-0x100DFFFF	AHB	768KB
Northbridge	0x100E0000-0x100FFFFF	APB	128KB
Northbridge	0x10100000-0x17FFFFFF	Reserved	127MB
Southbridge	0x18000000-0x1EFFFFFF	AHB	112MB
Cortex-A9 structured ASIC	0x1F000000-0x1FFFFFFF	AXI (internal)	16MB
Cortex-A9 structured ASIC DMC (for local SDRAM)	0x20000000-0x3FFFFFFF	DDR2	512MB
Northbridge	0x40000000-0x5FFFFFFF	SMC	512MB
Northbridge	0x60000000-0x6FFFFFFF	PCI	256MB
Northbridge	0x70000000-0x8FFFFFFF	DDR	512MB
Northbridge	0x90000000-0xBFFFFFFF	PCI	768MB
Logic Tile site	0xC0000000-0xFFFFFFFF	External	1GB

Note

- The 64MB memory region 0x00000000–0x0FFFFFFF can be remapped to NOR flash (SMC CS0)
- The other memory regions have fixed decoding and are handled either internally or externally.

Table 4-2 on page 4-4 lists the locations for memory, peripherals, and controllers for the Northbridge and the Southbridge. *System memory map for standard peripherals* on page 4-8 also shows a pictorial overview of the memory map.

Table 4-2 Memory map for standard peripherals

Peripheral	Address range	Bus type	Region size
Dynamic memory mirror (0x70000000-0x7FFFFFFF) During boot remapping however, the bottom 64MB of this memory region (0x00000000-0x03FFFFFF) can be: • NOR flash:0x40000000-0x43FFFFFF	0x00000000-0x0FFFFFFF	DDR	256MB
System registers	0x10000000-0x10000FFF	APB	4KB
System controller 0	0x10001000-0x10001FFF	APB	4KB
3-Wire Serial Bus Control	0x10002000-0x10002FFF	APB	4KB
Reserved (TrustZone Controller)	0x10003000-0x10003FFF	APB	4KB
Advanced Audio CODEC	0x10004000-0x10004FFF	APB	4KB
MultiMedia Card Interface	0x10005000-0x10005FFF	APB	4KB
Keyboard/Mouse Interface 0	0x10006000-0x10006FFF	APB	4KB
Keyboard/Mouse Interface 1	0x10007000-0x10007FFF	APB	4KB
Character LCD (UART)	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
Watchdog 0 Interface	0x1000F000-0x1000FFFF	APB	4KB
Watchdog 1 Interface	0x10010000-0x10010FFF	APB	4KB
Timer modules 0 and 1 interface (Timer 1 starts at 0x10011020)	0x10011000-0x10011FFF	APB	4KB
Timer modules 2 and 3 interface (Timer 3 starts at 0x10020020)	0x10012000-0x10012FFF	APB	4KB
GPIO interface 0	0x10013000-0x10013FFF	APB	4KB

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

Peripheral	Address range	Bus type	Region size
GPIO interface 1	0x10014000-0x10014FFF	APB	4KB
GPIO interface 2 (miscellaneous onboard I/O)	0x10015000-0x10015FFF	APB	4KB
Serial Bus Control (DVI)	0x10016000-0x10016FFF	APB	4KB
Real Time Clock interface	0x10017000-0x10017FFF	APB	4KB
Timer Modules 4 and 5 interface	0x10018000-0x10018FFF	APB	4KB
Timer Modules 6 and 7 interface	0x10019000-0x10019FFF	APB	4KB
System Controller 1	0x1001A000-0x1001AFFF	APB	4KB
Reserved for future use (4K x 5)	0x1001B000-0x1001FFFF	APB	20KB
Color LCD Controller configuration	0x10020000-0x1002FFFF	AHB	64KB
DMA Controller configuration	0x10030000-0x1003FFFF	AHB	64KB
Reserved (64K x 2)	0x10040000-0x1005FFFF	AHB	128KB
Internal Northbridge SRAM	0x10060000-0x1007FFFF	AXI	128KB
Reserved (64K x 6)	0x10080000-0x100DFFFF	AHB	384KB
Dynamic Memory Controller configuration	0x100E0000-0x100E0FFF	APB	4KB
Static Memory Controller configuration	0x100E1000-0x100E1FFF	APB	4KB
Reserved	0x100E2000-0x100E2FFF	APB	4KB
APB Registers (PLL configuration)	0x100E3000-0x100E3FFF	APB	4KB
Reserved for future use	0x100E4000-0x100EFFFF	APB	48KB
Reserved for future use (DAP ROM table)	0x100F0000-0x100FFFFF	APB	64KB
Reserved	0x10100000-0x103FFFFF	–	3MB
Reserved for future use	0x10400000-0x16FFFFFF	AHB or AXI	108MB
Reserved for future use	0x17000000-0x17FFFFFF	AXI	16MB
CompactFlash interface	0x18000000-0x1B000FFF	AHB	4KB
Reserved for future use (CoreSight)	0x1C001000-0x1CFFFFFF	AHB	16MB

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

Peripheral	Address range	Bus type	Region size
Reserved for future use	0x1D000000-0x1DFFFFFF	AHB	16MB
Generic Interrupt Controller 0 (GIC0) (not used by PBX-A9)	0x1E000000-0x1E00FFFF	AHB	64KB
Generic Interrupt Controller 1(GIC1) (not used by PBX-A9)	0x1E010000-0x1E01FFFF	AHB	64KB
Generic Interrupt Controller 2 (GIC2) (**nIRQ** interrupt handling for tile site)	0x1E020000-0x1E02FFFF	AHB	64KB
Generic Interrupt Controller 3 (GIC3) (**nFIQ** interrupt handling for tile site)	0x1E030000-0x1E03FFFF	AHB	64KB
Reserved for future use	0x1E040000-0x1EFFFFFF	AHB	15.75MB
Cortex-A9 structured ASIC peripherals • See Figure 4-2 on page 4-9	0x1F000000-0x1FFFFFFF	AHB or AXI	16MB
Cortex-A9 structured ASIC DDR2	0x20000000-0x3FFFFFFF	AXI	512MB
Static memory (CS0) NOR flash	0x40000000–0x43FFFFFF	SMC	64MB
Static memory (CS1) NOR flash	0x44000000–0x47FFFFFF	SMC	64MB
Static memory (CS2) Cellular RAM	0x48000000–0x4BFFFFFF	SMC	64MB
Static memory (CS3) configuration flash	0x4C000000–0x4DFFFFFF	SMC[a]	32MB
Static memory (CS3) Ethernet	0x4E000000–0x4EFFFFFF	SMC	16MB
Static memory (CS3) USB	0x4F000000–0x4FFFFFFF	SMC	16MB
Static memory (CS4) Reserved (**nCS0**)	0x50000000–0x53FFFFFF	SMC	64MB
Static memory (CS5) Reserved (**nCS1**)	0x54000000–0x57FFFFFF	SMC	64MB
Static memory (CS6) Reserved (**nCS2**)	0x58000000–0x5BFFFFFF	SMC	64MB
Static memory (CS7) Reserved (**nCS3**)	0x5C000000–0x5FFFFFFF	SMC	64MB
PCI interface	0x60000000-0x6FFFFFFF	PCI	256MB
Dynamic memory (CS0)	0x70000000-0x7FFFFFFF	DDR	256MB

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

Peripheral	Address range	Bus type	Region size
Dynamic memory (CS1)	0x80000000-0x8FFFFFFF	DDR	256MB
PCI interface	0x90000000-0xBFFFFFFF	PCI	768MB
Logic Tile site expansion. (If a Logic Tile is not fitted, the baseboard aborts accesses to this memory region)	0xC0000000-0xFFFFFFFF	AHB or AXI	1GB

a. Devices on Static Memory Controller CS3 have additional address decoding performed by a PLD.

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

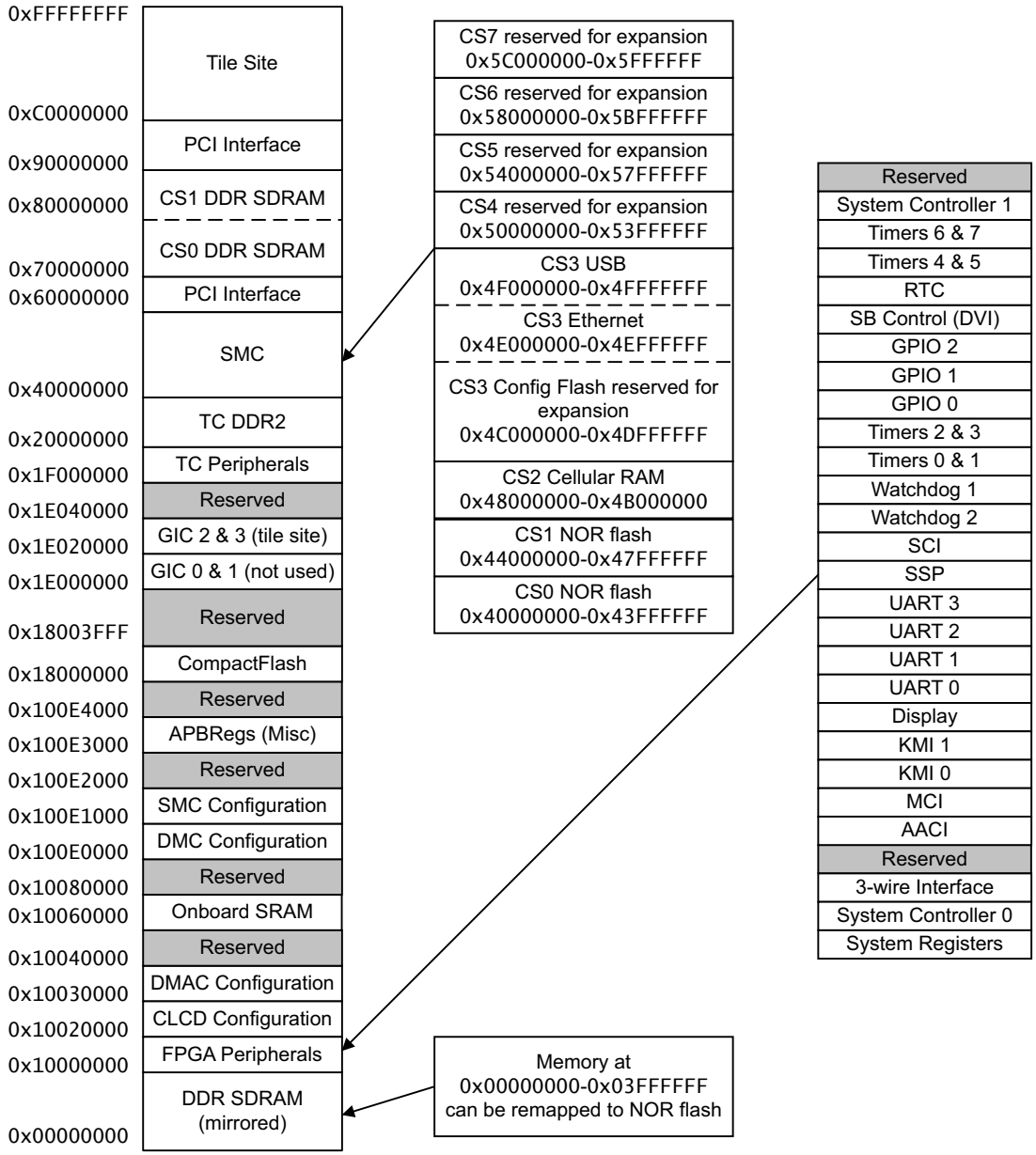


Figure 4-1 System memory map for standard peripherals

Figure 4-2 shows the memory map for Cortex-A9 structured ASIC peripherals.

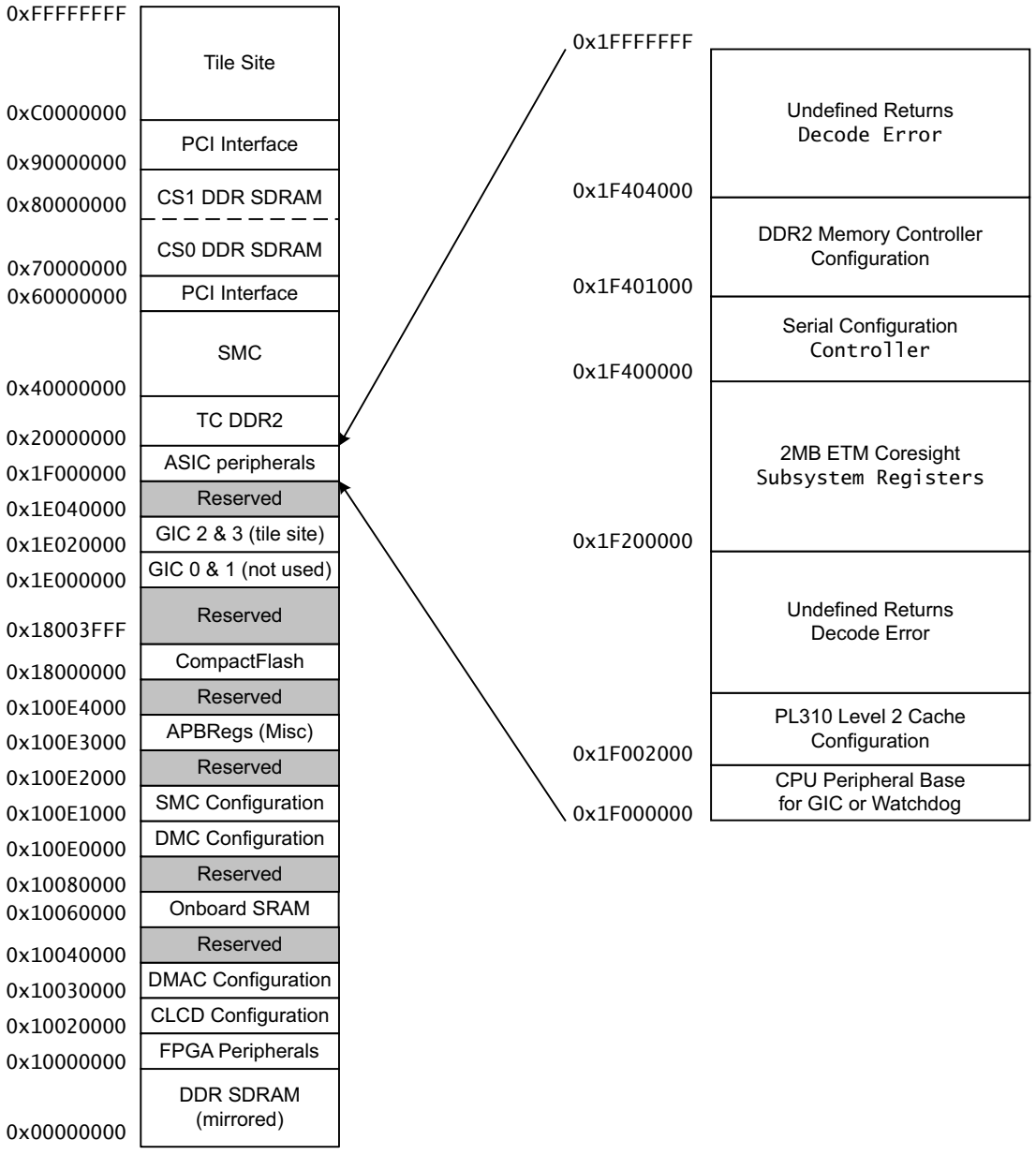


Figure 4-2 Memory map for Cortex-A9 structured ASIC peripherals

4.2 Configuration and initialization

This section describes how the PBX-A9 baseboard and external memory and peripherals are configured and initialized at power on. See *Status and system control registers* on page 4-12 for details on configuring the PBX-A9 when the system is out of reset.

4.2.1 Remapping of boot memory

On reset, the Cortex-A9 processor begins executing code at address `0x00000000`. During normal operation this address is mirrored volatile DRAM but at reset remapping enables mirrored non-volatile static memory to be accessed. The non-volatile memory to be remapped is determined by the configuration switch S7-1 that Table 4-3 lists.

Table 4-3 Boot memory

S7-1	Chip select	Memory Range	Comment
OFF	SMC CS0	`0x40000000`– `0x43FFFFFF`	Lower 64MB bank of NOR flash is remapped to `0x00000000-0x03FFFFFF`.
ON	SMC CS4	`0x50000000`– `0x53FFFFFF`	Reserved. Do not use.

4.2.2 Memory characteristics

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

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

Table 4-4 Memory chip selects and address range

Chip Select	Address range	Device
DMC CS0	`0x00000000-0x0FFFFFFF` (`0x70000000-0x7FFFFFFF`)	DRAM (DRAM mirror)
DMC CS1	`0x80000000-0x8FFFFFFF`	DRAM
SMC CS0	`0x40000000-0x43FFFFFF`	NOR flash
SMC CS1	`0x44000000-0x47FFFFFF`	NOR flash
SMC CS2	`0x48000000-0x4BFFFFFF`	Cellular RAM

Table 4-4 Memory chip selects and address range (continued)

Chip Select	Address range	Device
SMC CS3*Additional address decoding is handled by the CS PLD.*	0x4C000000-0x4DFFFFFF	Reserved
	0x4E000000-0x4EFFFFFF	Ethernet
	0x4F000000-0x4FFFFFFF	USB
SMC CS4	0x5000000-0x53FFFFFF	Reserved
SMC CS5	0x54000000-0x57FFFFFF	Reserved
SMC CS6	0x58000000-0x5BFFFFFF	Reserved
SMC CS7	0x5C000000-0x5FFFFFFF	Reserved

4.3 Status and system control registers

The PBX-A9 baseboard status and system control registers enable the Cortex-A9 to determine its environment and to control the on-board systems. Table 4-5 lists the register set.

Note

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

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

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

Register	Offset Value	Access[a]	Reset Value	Description
SYS_ID	`0x0000`	Read-only	`0x01820500`	System Identifier. See *ID Register, SYS_ID* on page 4-16.
SYS_USERSW	`0x0004`	Read-only	`0x0000000XX`	Bits [7:0] map to S4 (user switches). See *User Switch Register, SYS_USERSW* on page 4-16.
SYS_LED	`0x0008`	Read/Write	`0x000000000`	Bits [7:0] map to user LEDs. See *LED Register, SYS_LED* on page 4-17.
SYS_OSC[0:4]	`0x000C`–`0x001C`	Read/Write Lockable	0: `0x00012C5C` 1: `0x00002CC0` 2: `0x00002C75` 3: `0x00020211` 4: `0x00002C75`	Settings for the ICS307 programmable oscillators: OSC0 - OSC4. See *Oscillator Registers, SYS_OSCx* on page 4-18.
SYS_LOCK	`0x0020`	Read/Write	`0x00000000`	Write `0xA05F` to unlock lockable registers. See *Lock Register, SYS_LOCK* on page 4-19.
SYS_100HZ	`0x0024`	Read-only	`0x00000000`	100Hz counter. See *100Hz Counter, SYS_100HZ* on page 4-21.
Reserved	`0x0028`-`0x002C`	–	–	This region is reserved.
SYS_FLAGS	`0x0030`	Read	`0x00000000`	General-purpose flags (reset by any reset). See *Flag Registers, SYS_FLAGSx and SYS_NVFLAGSx* on page 4-21.
SYS_FLAGSSET	`0x0030`	Write	`0x00000000`	Set bits in general-purpose flags.

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

Register	Offset Value	Access[a]	Reset Value	Description
SYS_FLAGSCLR	0x0034	Write-only	–	Clear bits in general-purpose flags.
SYS_NVFLAGS	0x0038	Read	0xXXXXXXXX	General-purpose nonvolatile flags (reset only on power up).
SYS_NVFLAGSSET	0x0038	Write	0x00000000	Set bits in general-purpose nonvolatile flags.
SYS_NVFLAGSCLR	0x003C	Write-only	–	Clear bits in general-purpose nonvolatile flags.
SYS_RESETCTL	0x0040	Read/Write Lockable	0x00000000	Controls the software reset level to be applied to system components. See *Reset Control Register, SYS_RESETCTL* on page 4-22.
Reserved	0x0044	–	–	This region is reserved.
SYS_MCI	0x0048	Read/Write	0x00000002	MCI status and control register. See *MCI Register, SYS_MCI* on page 4-23.
SYS_FLASH	0x004C	Read/Write	0x00000000	Controls write protection of flash devices. See *Flash Control Register, SYS_FLASH* on page 4-24.
SYS_CLCD	0x0050	Read/Write	0x00001F00	Controls LCD power and multiplexing. See *CLCD Control Register, SYS_CLCD* on page 4-25.
Reserved	0x0054	Read/Write	–	This region is reserved.
SYS_CFGSW	0x0058	Read-only	0x000000XX	Read register returns the current switch settings of switch S7. See *Configuration select switch, SYS_CFGSW* on page 4-26.
SYS_24MHZ	0x005C	Read-only	0x00000000	32-bit counter clocked at 24MHz. See *24MHz Counter, SYS_24MHZ* on page 4-27.
SYS_MISC	0x0060	Read-only	0x00000000	Miscellaneous control flags. See *Miscellaneous flags, SYS_MISC* on page 4-27.
SYS_DMAPSR0	0x0064	Read/Write	0x00000000	Selection control for remapping DMA from external peripherals to DMA. See *DMA peripheral map register, SYS_DMAPSR* on page 4-28.
SYS_PEX_STAT	0x0068	Read-only	0x0000FFFF	PCI status register. See *PCI Express status register, SYS_PEX_STAT* on page 4-30.
SYS_PCI_STAT	0x006C	Read/Write Lockable	0x00000337	PCI status register. See *PCI status register, SYS_PCI_STAT* on page 4-31.

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

Register	Offset Value	Access[a]	Reset Value	Description
Reserved	0x0070	–	–	This region is reserved.
SYS_CTRL1	0x0074	Read/Write Lockable	0x000000C0	This register resets the Cortex-A9 processors. See *Control register 1, SYS_CTRL1* on page 4-33 for details.
SYS_CTRL2	0x0078	Read/Write Lockable	–	This register reads the Config PLD read data register fields that provide status information from the Cortex-A9 structured ASIC. See *Control register 2, SYS_CTRL2* on page 4-33
SYS_SERIAL_DATA	0x007C	Read/Write Lockable	–	This register defines the Cortex-A9 structured ASIC configuration. See *Serial data register, SYS_SERIAL_DATA* on page 4-34.
SYS_SERIAL_ADDR	0x0080	Read/Write Lockable	0x00000000	This register defines the Cortex-A9 structured ASIC serial read address. See
SYS_PROCID0	0x0084	Read-only	0x0C000000	Read returns a description for the Cortex-A9 structured ASIC present on the platform baseboard (PB). See *Processor ID register 0, SYS_PROCID0* on page 4-37.
SYS_PROCID1	0x0088	Read-only	0xFF000000	Read returns the processor type implemented on the tile site. For Platform Baseboards, the value returned is always 0xFF000000 indicating that no Core Tile is fitted.
SYS_OSCRESET[0:4]	0x008C–0x009C	Read/Write	0x00000000	Value to load into the SYS_OSC[0:4] registers on a manual reset. At power-on reset, the SYS_OSCRESET[0:4] registers are loaded with the same default value as used for SYS_OSC[0:4].
Reserved	0x00A0–0x00BC	–	–	This region is reserved.

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

Register	Offset Value	Access[a]	Reset Value	Description
SYS_TEST_OSC[0]	0x00C0–0x00D0	Read-only	0x00000000	32-bit counter clocked from ICS307 oscillators. See *Oscillator test registers, SYS_TEST_OSCx* on page 4-41.
SYS_TEST_OSC[1]				
SYS_TEST_OSC[2]				
SYS_TEST_OSC[3]				
SYS_TEST_OSC[4]				
SYS_OSC[5]	0x00D4–0x00D8	Read/Write Lockable	0x00002C3E	Settings for the ICS307 programmable oscillators: OSC5 - OSC6. See *Oscillator Registers, SYS_OSCx* on page 4-18.
SYS_OSC[6]	0x00D4–0x00D8	Read/Write Lockable	0x00002C70	
SYS_OSCRESET[5]	0x00DC–0x00E0	Read/Write Lockable	0x00000000	Value to load into the SYS_OSC[5:6] registers on a manual reset. At power-on reset, the SYS_OSCRESET[5:6] registers are loaded with the same default value as used for SYS_OSC[5:6].
SYS_OSCRESET[6]				
SYS_TEST_OSC[5:6]	0x00E4–0x00E8	Read-only	–	32-bit counter clocked from ICS307 oscillators. See *Oscillator test registers, SYS_TEST_OSCx* on page 4-41.
SYS_OSC[7]	0x00EC	Read/Write Lockable	0x00002C5C	Settings for the ICS307 programmable oscillator OSC7. See *Oscillator Registers, SYS_OSCx* on page 4-18.
SYS_OSCRESET[7]	0x00F0	Read/Write Lockable	0x00002C5C	Value to load into the SYS_OSC[7] register on a manual reset. At power-on reset, the SYS_OSCRESET[7] register is loaded with the same default value as used for SYS_OSC[7].
SYS_TEST_OSC[7]	0x00F4	Read-only	0x00000000	32-bit counter clocked from ICS307 oscillator OSC7. See *Oscillator test registers, SYS_TEST_OSCx* on page 4-41.
SYS_DEBUG	0x00F8	Read/Write Lockable	–	Reserved for core debug function control and read back.
SYS_TESTMODE	0x00F8	Read/Write Lockable	–	Reserved for core test mode control.

a. If Access is lockable, the register can only be written if SYS_LOCK is unlocked (see *Lock Register, SYS_LOCK* on page 4-19).

4.3.1 ID Register, SYS_ID

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

Figure 4-3 shows the register bit assignment.

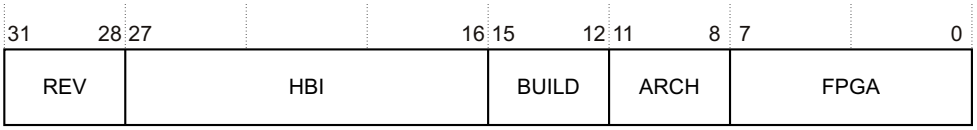


Figure 4-3 SYS_ID register

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

Table 4-6 SYS_ID register bit assignments

Bits	Access	Name	Reset	Description
[31:28]	Read-only	REV	`0x1`	Board revision: `0x0` = Rev A `0x1` = Rev B `0x2` = Rev C
[27:16]	Read-only	HBI	`0x182`	HBI board number in BCD
[15:12]	Read-only	BUILD	`0xF`	Build variant of board (from BOM)`0xF`: all builds
[11:8]	Read-only	ARCH	`0x5`	Bus architecture `0x4` = AHB `0x5` = AXI
[7:0]	Read-only	FPGA	`0x00`	FPGA build (BCD coded)

4.3.2 User Switch Register, SYS_USERSW

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

Figure 4-4 on page 4-17 shows the register bit assignment.

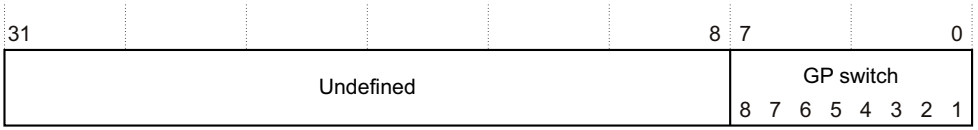


Figure 4-4 SYS_USERSW register

4.3.3 LED Register, SYS_LED

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

Figure 4-5 shows the register bit assignment.

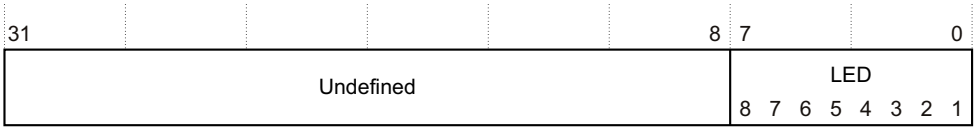


Figure 4-5 SYS_LED register

4.3.4 Oscillator Registers, SYS_OSCx

The oscillator registers are read and write registers that control the frequency of the clocks generated by the ICS307 programmable oscillators OSC0 through to OSC7.

Table 4-7 lists the register addresses.

Table 4-7 SYS_OSCx register

Register	Address
SYS_OSC0	0x1000000C
SYS_OSC1	0x10000010
SYS_OSC2	0x10000014
SYS_OSC3	0x10000018
SYS_OSC4	0x1000001C
SYS_OSC5	0x100000D4
SYS_OSC6	0x100000D8
SYS_OSC7	0x100000EC

Figure 4-6 shows the registers bit assignment.

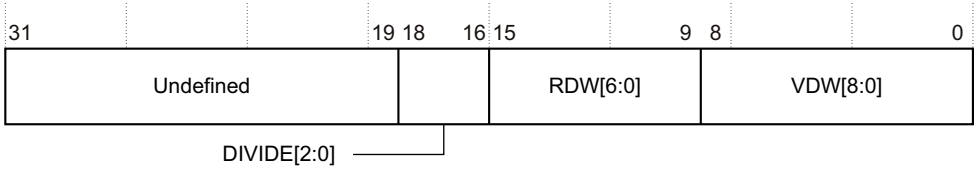


Figure 4-6 SYS_OSCx register

Table 4-8 shows the function of the register bits. For more detail on bit values, see *ICS307 programmable clock generators* on page 3-34 and *Clock frequency restrictions* on page B-3.

Table 4-8 SYS_OSCx register bit assignments

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

Note

- Before writing to a SYS_OSCx register, you must unlock it by writing the value 0x0000A05F to the SYS_LOCK register. After writing the SYS_OSC register, you must relock it by writing any value other than 0x0000A05F to the SYS_LOCK register.
- ARM recommends not to make changes to these registers at runtime, instead use the **SYS_OSCRESETx** registers followed by a software reset. See *Oscillator reset registers, SYS_OSCRESETx* on page 4-40.
- See also *ICS307 programmable clock generators* on page 3-34.

4.3.5 Lock Register, SYS_LOCK

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

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

Figure 4-7 on page 4-20 shows the register bit assignment.

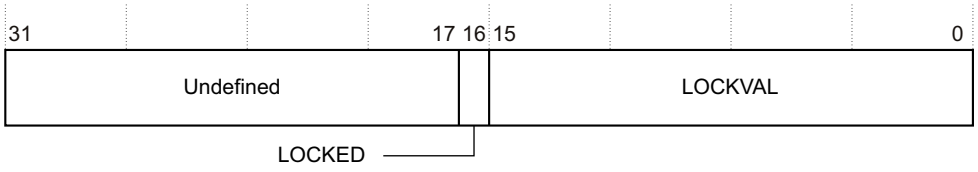


Figure 4-7 SYS_LOCK register

Table 4-9 shows the function of the register bits.

Table 4-9 SYS_LOCK register bit assignments

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

4.3.6 100Hz Counter, SYS_100HZ

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

Figure 4-8 shows the register bit assignment.

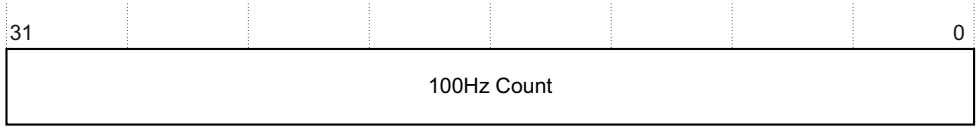


Figure 4-8 SYS_100HZ register

4.3.7 Flag Registers, SYS_FLAGSx and SYS_NVFLAGSx

The registers that Table 4-10 shows provide two 32-bit register locations containing general-purpose flags. You can assign any meaning to the flags.

Table 4-10 Flag registers

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

The board provides two distinct types of flag register:

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

Flag and Nonvolatile Flag Registers

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

Flag and Nonvolatile Flag Set Registers

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

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

Flag and Nonvolatile Flag Clear Registers

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

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

4.3.8 Reset Control Register, SYS_RESETCTL

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

Figure 4-9 shows the register bit assignment.

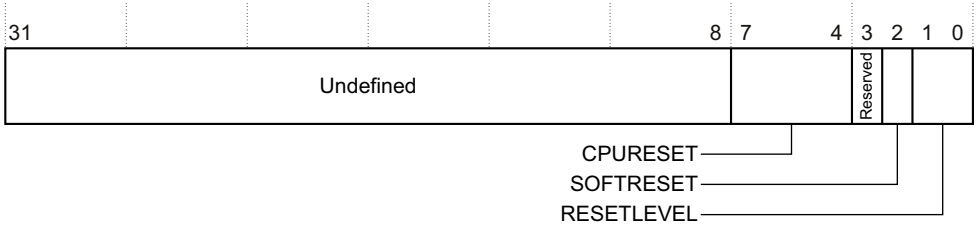


Figure 4-9 SYS_RESETCTL register

Table 4-11 shows the function of the register bits.

Table 4-11 SYS_RESETCTL register bit assignments

Bits	Access	Name	Reset	Description
[31:8]	Write ignored, read as zero	–	0x000000	Undefined
[7:4]	Write only	CPURESET	b1111	Determines which CPU core [3:0] is in reset when **nPBRESET** is pressed. Default b1111 puts all CPU Cores in reset when **nPBRESET** is pressed
[3]	–	–	b0	Reserved
[2]	Write only	SWRESET	b0	Forces a software reset at the depth set by RESETLEVEL.
[1:0]	Read/Write	RESETLEVEL	b00	Sets the depth of a software reset: • b00 = POWER • b01 = CLOCK CONFIG • b1x = SYSTEM RESET.

4.3.9 MCI Register, SYS_MCI

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

Figure 4-10 shows the register bit assignment.

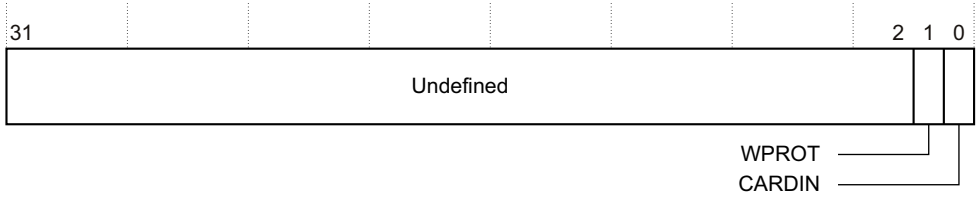


Figure 4-10 SYS_MCI register

Table 4-12 shows the function of the register bits.

Table 4-12 SYS_MCI register bit assignment

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

4.3.10 Flash Control Register, SYS_FLASH

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

Figure 4-10 on page 4-23 shows the register bit assignment.

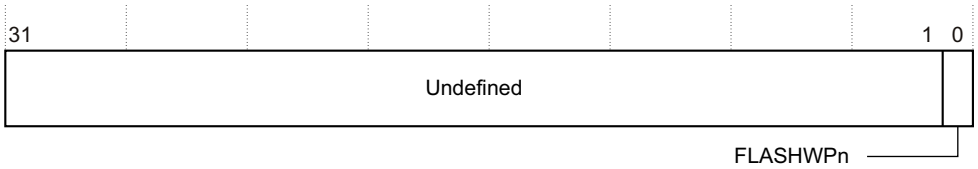


Figure 4-11 SYS_FLASH register

Table 4-13 lists the function of the register bits.

Table 4-13 SYS_FLASH register bit assignments

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

4.3.11 CLCD Control Register, SYS_CLCD

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

Note

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

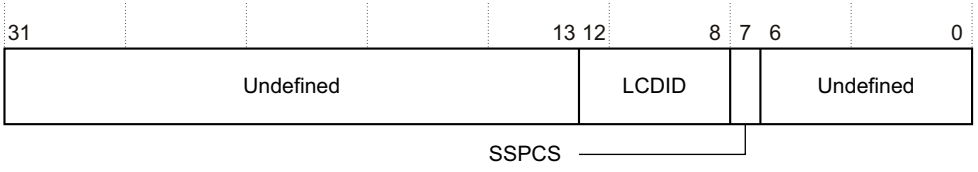


Figure 4-12 SYS_CLCD register

Table 4-14 lists the function of the register bits.

Table 4-14 SYS_CLCD register bit register assignments

Bits	Access	Name	Reset	Description
[31:16]	Write ignored, read as zero	–	0x0000	Undefined
[15:13]	Write ignored, read as zero	–	b000	Undefined
[12:8]	Read-only	LCDID	b11111	Returns the setting of the ID links on the CLCD adaptor board **Note** Retained for legacy compatibility only.
[7]	Read/Write	SSPCS	b0	Enables external serial communication using the baseboard *Synchronous Serial Port* SSP expansion connector J27: • b1 = **SSPnCS** at J27 is active • b0 = **SSPnCS** at J27 is not active See *Synchronous Serial Port, SSP* on page 3-25.
[6:0]	Write ignored, read as zero	–	b0000	Undefined

4.3.12 Configuration select switch, SYS_CFGSW

This SYS_CFGSW register at 0x10000058 reads the configuration select switch S7. A value of 1 indicates that the switch is on.

Note

This register does not enable the processor to change the PBX-A9 configuration set by the switches. See *Baseboard configuration switches* on page 2-8 for details of the available options.

Figure 4-13 shows the register bit assignment.

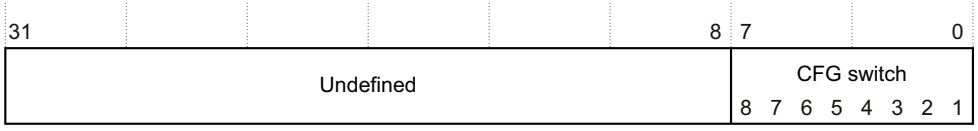


Figure 4-13 SYS_CFGSW register

4.3.13 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.

Figure 4-14 shows the register bit assignment.

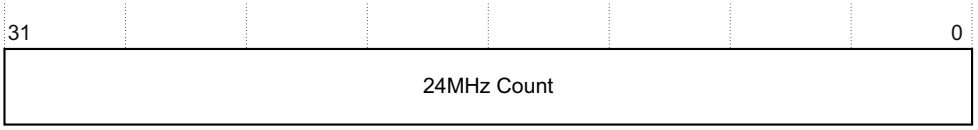


Figure 4-14 SYS_24MHZ register

4.3.14 Miscellaneous flags, SYS_MISC

The SYS_MISC register at `0x10000060` contains status flags for the Logic Tile and CF card detect functions. It also contains the configuration bits which select the sources of the rear panel UART and video signals. See Table 4-15 on page 4-28.

Figure 4-15 shows the register bit assignment.

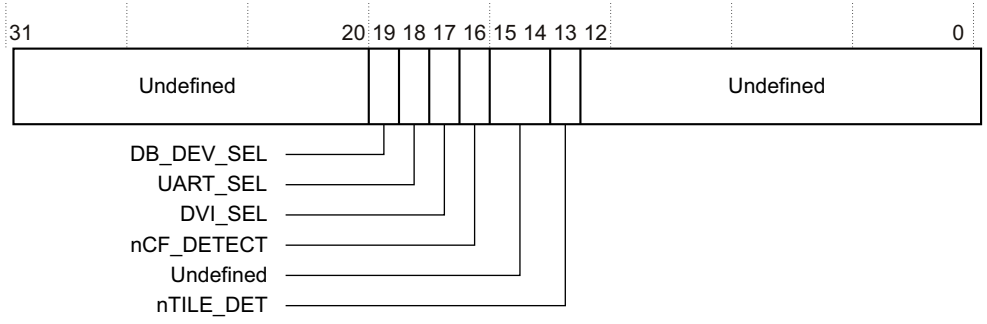


Figure 4-15 SYS_MISC register

Table 4-15 lists the function of the register bits.

Table 4-15 SYS_MISC register bit assignment

Bits	Access	Name	Reset	Description
[31:20]	Write ignored, read as zero	–	0x0000	Undefined
[19]	Read/Write	DB_DEV_SEL	b0	Selects CLCD source: • b0 = Northbridge • b1 = Cortex-A9 daughterboard.
[18]	Read/Write	UART_SEL	b0	Selects the source for the UART 2 and UART 3 connectors: • b0: baseboard interface • b1: tile site interface
[17]	Read/Write	DVI_SEL	b0	Selects the source for the DVI connector: • b0: baseboard interface • b1: tile site interface
[16]	Read-only	nCF_DETECT	b1	Flash present: • b0: CompactFlash card detected • b1: No CompactFlash card detected
[15:14]	Write ignored, read as zero	–	b000	Undefined
[13]	Read-only	nTILE_DET	b0	Tile Detect: • b0 = tile present • b1 = tile not present
[12:0]	Write ignored, read as zero	–	0	Undefined

4.3.15 DMA peripheral map register, SYS_DMAPSR

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

Figure 4-16 on page 4-29 shows the register bit assignment.

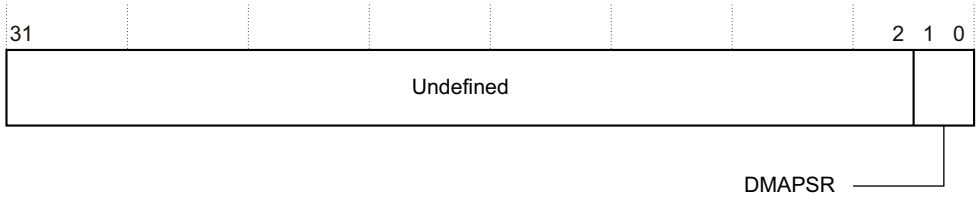


Figure 4-16 SYS_DMAPSR register

Table 4-16 lists the function of the register bits.

Table 4-16 SYS_DMAPSR register bit assignments

Bit	Access	Name	Reset	Description
[31:4]	Write ignored, read as zero	–	0x0000000	Undefined
[3:2]	Write ignored, read as zero	–	b00	Undefined
[1:0]	Read/Write	DMAPSR	b00	Selects the peripheral group to be mapped to DMA. See Table 4-17 for the bit encoding.

Table 4-17 lists the bit encoding. See *Single Master Direct Memory Access Controller, SMDMAC* on page 4-50 for more information on the DMA logic.

Table 4-17 SYS_DMAPSR register bit coding

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

Table 4-17 SYS_DMAPSR register bit coding (continued)

DMAPSR = b00	DMAPSR = b01	DMAPSR = b1X	DMA Request and Response
Peripheral	**Peripheral**	**Peripheral**	**DMACSREQ DMACBREQ DMACCLR**
T1DMAC[0]	UART2 RX	reserved	[2]
USB[1]	SSP TX	reserved	[1]
USB[0]	SSP RX	reserved	[0]

4.3.16 PCI Express status register, SYS_PEX_STAT

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

Figure 4-17 shows the register bit assignment.

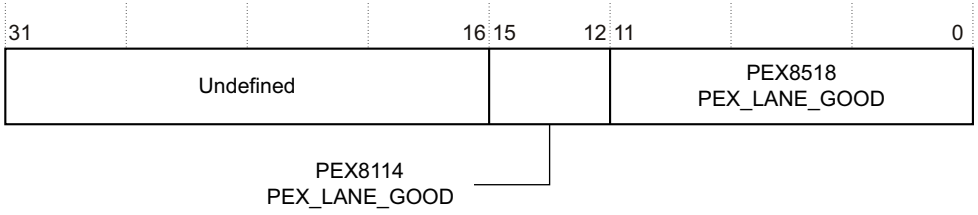


Figure 4-17 SYS_PEX_STAT register

Table 4-18 lists the function of the register bits.

Table 4-18 SYS_PEX_STAT register bit assignments

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

Note

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

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

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

4.3.17 PCI status register, SYS_PCI_STAT

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

Caution

ARM does not recommend changing the register default values because this might effect the reliability of PCI interface.

Figure 4-18 shows the register bit assignment.

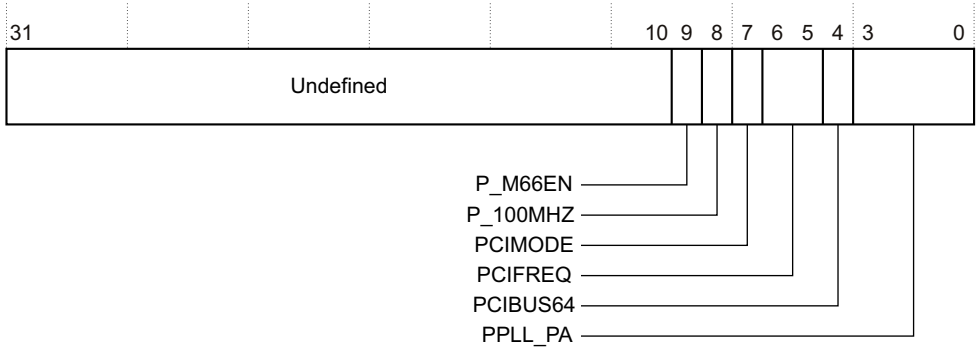


Figure 4-18 SYS_PCI_STAT register

Table 4-19 on page 4-32 lists the function of the register bits.

Table 4-19 SYS_PCI_STAT register bit assignments

Bits	Access	Name	Reset	Description
[31:12]	Write ignored, read as zero	–	0x00000	Undefined
[11:10]	Write ignored, read as zero	–	b00	Undefined
[9]	Read/Write	P_M66EN	b0	66MHz PCI-X enable
[8]	Read/Write	P_100MHZ	b1	Clock control
[7]	Read/Write	PCIMODE	b1	PCI mode: • b0 = PCI-X • b1 = PCI
[6:5]	Read/Write	PCIFREQ	b00	PCI frequency: • b00 = 33MHz • b01 = 66MHz • b10 = 100MHz • b11 = 133MHz
[4]	Read/Write	PCIBUS64	b0	64-bit PCI bus enable
[3:0]	Read/Write	PPLL_PA	0x7	PLL range: • 0x1 = PPLL_RANGE300_600MHz • 0x3 = PPLL_RANGE150_300MHz • 0x4 = PPLL_RANGE100_200MHz • 0x5 = PPLL_RANGE75_150MHz • 0x7 = PPLL_RANGE50_100MHz

4.3.18 Control register 1, SYS_CTRL1

The SYS_CTRL1 register at `0x10000074` resets the Cortex-A9 processors.

Figure 4-19 shows the register bit assignment.

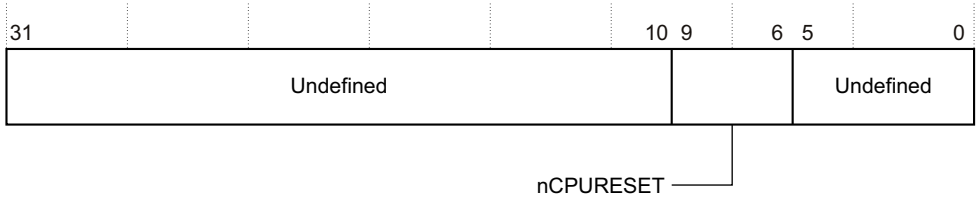


Figure 4-19 SYS_CTRL1 register

Table 4-20 lists the function of the register bits.

Table 4-20 SYS_CTRL1 register bit assignments

Bits	Access	Name	Reset	Description
[31:12]	Write ignored, read as zero	–	`0x00000`	Undefined
[11:10]	Write ignored, read as zero	–	`b00`	Undefined
[9:6]	Read/Write Lockable	nCPURESET	`b1111`	Individual Cortex-A9 processor resets. nCPURESET[x] resets CPU[x] See *Cortex-A9 MPCore Technical Reference Manual* (DDI 0407) for details.
[5:4]	Write ignored, read as zero	–	`b00`	Undefined
[3:0]	Write ignored, read as zero	–	`0x0`	Undefined

4.3.19 Control register 2, SYS_CTRL2

The SYS_CTRL2 register at `0x10000078` indicates whether a Cortex-A9 processor is in Standby mode.

Figure 4-20 on page 4-34 shows the register bit assignment.

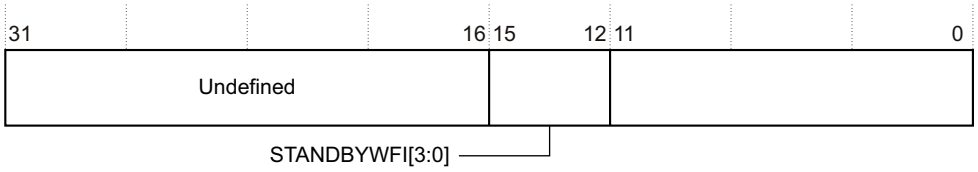


Figure 4-20 SYS_CTRL2 register

Table 4-21 lists the function of the register bits.

Table 4-21 SYS_CTRL2 register bit assignments

Bits	Access	Name	Reset	Description
[31:16]	Read as zero, write ignored.	–	0x0000	Undefined
[15:12]	Read/Write Lockable	STANDBYWFI	b0110	Individual WFI indicators. Indicates if a Cortex-A9 processor is in Standby mode. STANDBYWFI[x] indicates that CPU[x] is in Standby mode. See *Cortex-A9 MPCore Technical Reference Manual* (DDI 0407) for details.
[11:0]	Read as zero, write ignored.	–	0x000	Undefined

4.3.20 Serial data register, SYS_SERIAL_DATA

The SYS_SERIAL_DATA register at 0x1000007C sets the Cortex-A9 structured ASIC serial configuration stream values.

This register holds the data to be sent or read from the A9 daughterboard through a serial stream. For a write, after the register has been updated, the contents are sent to the A9 daughterboard to the address described in the register SYS_SERIAL_ADDR. When reading this register, a pulse is generated after a read is issued at the address described in SYS_SERIAL_ADDR, The serial stream is sent from the A9 daughterboard and updates the contents of this register. However, the contents of the register are only updated after the first read is issued. Therefore, to read the updated contents of the serial stream, this register needs to be read from twice.

By default, the updated contents of the serial register on the A9 daughterboard is retained if a software or push button reset be issued.

Figure 4-21 on page 4-35 shows the register bit assignment.

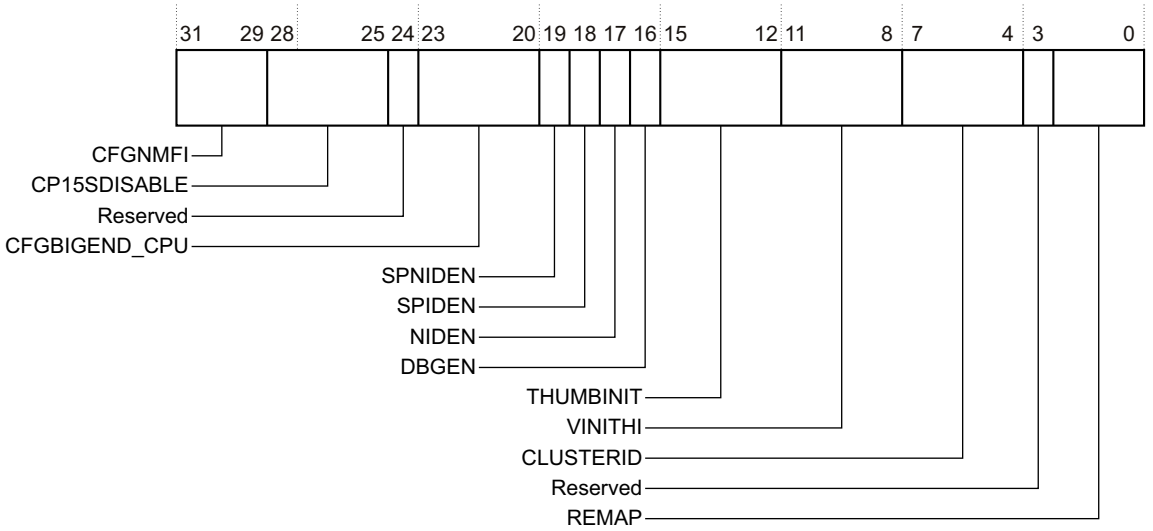


Figure 4-21 SYS_SERIAL_DATA register

Table 4-22 lists the function of the register bits.

Table 4-22 SYS_SERIAL_DATA register bit assignments

Bits	Access	Name	Reset	Description
[31:29]	Read/Write Lockable	CFGNMFI[2:0}	b000	Configures fast interrupts to be nonmaskable:0 = clear the NMFI bit in the CP15 c1 Control Register1 = set the NMFI bit in the CP15 c1 Control Register. CFGNMFI[x] configures fast interrupts to be nonmaskable for CPU[x].
[28:25]	Read/Write Lockable	CP15SDISABLE[3:0]	b0000	Disables write access to some system control processor registers. CP15SDISABLE[x] Disables write access to some system control processor registers in CPU[x]. See *Cortex-A9 Technical Reference Manual* (DDI 0388) for details of the affected registers.
[24]	Read/Write Lockable	–	b0	Reserved

Table 4-22 SYS_SERIAL_DATA register bit assignments (continued)

Bits	Access	Name	Reset	Description
[23:20]	Read/Write Lockable	CFGBIGEND_CPU[3:0]	b0000	Endianness configuration. Forces the EE bit in the CP15 c1 Control Register (SCTLR) to 1 at reset so that the Cortex-A9 processor boots with big-endian data handling.0 = EE bit is LOW1 = EE bit is HIGH CFGBIGEND_CPU[x] configures endianness for CPU[x].
[19]	Read/Write Lockable	SPNIDEN	b1	Secure privileged noninvasive debug enable for all CPUs:0 = not enabled1 = enabled.
[18]	Read/Write Lockable	SPIDEN	b1	Secure privileged invasive debug enable:0 = not enabled1 = enabled.
[17]	Read/Write Lockable	NIDEN	b1	Noninvasive debug enable fro all CPUs:0 = not enabled1 = enabled.
[16]	Read/Write Lockable	DBGEN	b1	Invasive debug enable fro all CPUs:0 = not enabled1 = enabled.
[15:12]	Read/Write Lockable	THUMBINIT[3:0]	b0000	Default exception handling state. When set to:0 = ARM1 = Thumb THUMBINIT[x] sets the default exception handling state for CPU[x].
[11:8]	Read/Write Lockable	VINITHI[3:0]	b0000	Controls the location of the exception vectors at reset:0 = start exception vectors at address 0x000000001 = start exception vectors at address 0xFFFF0000. VINITHI[x] controls the location of the exception vectors at reset for CPU[x].
[7:4]	Read/Write Lockable	CLUSTERID	b0000	Value read in Cluster ID register field, bits[11:8] of the MPIDR. See *Cortex-A9 Technical Reference Manual* (DDI 0388) for details of the MPIDR register.
[3]	Read/Write Lockable	–	b0	Reserved
[2:0]	Read/Write Lockable	REMAP	b000	Remaps the lower 256MB of DDR2 memory on the daughterboard. SeeTable 4-23 on page 4-37 for remap details.

Table 4-23 Daughterboard DDR2 memory remap

REMAP Value	Daughterboard DDR2 (Lower 256MB)	Baseboard DDR (lower 256MB)	Notes
b000	0x20000000-0x2FFFFFFF	0x00000000-0x0FFFFFFF	Lower 256MB of baseboard DDR memory at 0x70000000-0x7FFFFFFF is aliased at 0x0000000-0x0FFFFFFF
b001	0x00000000-0x0FFFFFFF	0x70000000-0x7FFFFFFF	Lower 256MB of daughterboard DDR2 memory at 0x20000000-0x2FFFFFFF is aliased at 0x0000000-0x0FFFFFFF.
b010	–	–	Reserved for Coherency Port validation.
b100	–	–	Reserved for Coherency Port validation.

4.3.21 Serial address register, SYS_SERIAL_ADDR

The SYS_SERIAL_ADDR register at 0x10000080 sets the Cortex-A9 structured ASIC serial configuration stream address value.

Figure 4-22shows the register bit assignment.

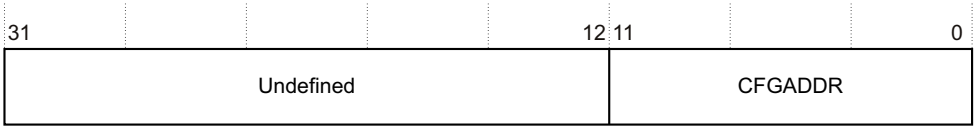


Figure 4-22 SYS_SERIAL_ADDR register

Table 4-24 lists the function of the register bits.

Table 4-24 SYS_SERIAL_ADDR register bit assignments

Bits	Access	Name	Reset	Description
[31:12]	Read as zero, write ignored.	–	0x00000	Undefined
[11:0]	Read/Write Lockable	CFGADDR	0x000	Serial configuration stream address value.

4.3.22 Processor ID register 0, SYS_PROCID0

This register at 0x10000084 indicates the processor type fitted to the baseboard and to the tile site, if a Core Tile is supported at the tile site.

Figure 4-23 on page 4-38 shows the register bit assignment.

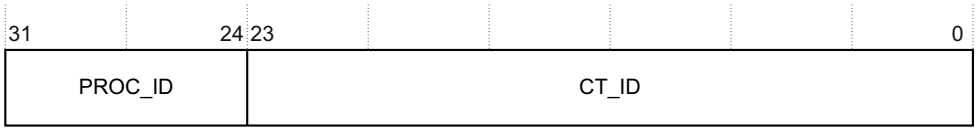


Figure 4-23 SYS_PROCID0 register

Table 4-25 lists the function of the register bits.

Table 4-25 SYS_PROCID0 register bit assignments

Bits	Access	Name	Reset	Description
[31:24]	Read-only	PROC_ID	0x06	Returns the processor type fitted to the baseboard: • 0x00 = ARM7TDMI • 0x02 = ARM9xx • 0x04 = ARM1136 • 0x06 = ARM11MPCore • 0x08 = ARM1156 • 0x0A = ARM1176 • 0x0C = Cortex-A9 • 0x0E = Cortex-A8 • 0xFF = Core Tile not supported.
[23:0]	Read-only	CT_ID	0x000000	ID register read from a Core Tile CPLD at the tile site if supported, else 0x000000. **Note** Only the Emulation Baseboard supports a direct Core Tile connection at the tile site.

4.3.23 Processor ID register 1, SYS_PROCID1

This register at 0x10000088 provides control and status information for a Mali tile fitted to the tile site.

Figure 4-24 on page 4-39 shows the register bit assignment.

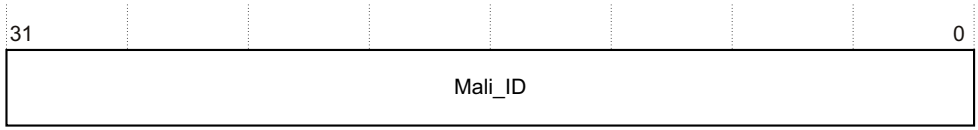


Figure 4-24 SYS_PROCID1 register

Table 4-26 lists the function of the register bits.

Table 4-26 SYS_PROCID1 register bit assignments

Bits	Access	Name	Reset	Description
[31:0]	Read/Write Lockable	Mali_ID	0xFF000000	Control and status information for a Mali tile fitted to the tile site.

4.3.24 Oscillator reset registers, SYS_OSCRESETx

The oscillator reset registers are read and write registers that control the frequency of the clocks generated by clock generators OSC0 through to OSC7 when a soft reset is generated.

Table 4-27 lists the register addresses.

Table 4-27 SYS_TEST_OSCRESETx register

Register	Address
SYS_OSCRESET0	0x1000008C
SYS_OSCRESET1	0x10000090
SYS_OSCRESET2	0x10000094
SYS_OSCRESET3	0x10000098
SYS_OSCRESET4	0x1000009C
SYS_OSCRESET5	0x100000DC
SYS_OSCRESET6	0x100000E0
SYS_OSCRESET7	0x100000F0

Figure 4-25 shows the register bit assignment.

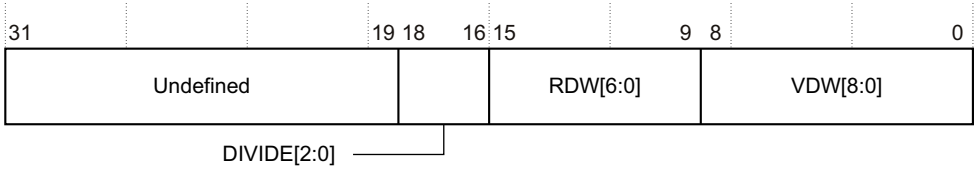


Figure 4-25 SYS_OSCRESETx register

Note

Before writing to a SYS_OSCRESETx register you must unlock it by writing the value 0x0000A05F to the SYS_LOCK register, see *Lock Register, SYS_LOCK* on page 4-19. After writing the SYS_OSCRESETx register, you must relock it by writing any value other than 0x0000A05F to the SYS_LOCK register.

For more detail on bit values, see *ICS307 programmable clock generators* on page 3-34 and *Oscillator Registers, SYS_OSCx* on page 4-18.

Note

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

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

4.3.25 Oscillator test registers, SYS_TEST_OSCx

The oscillator test registers provide 32-bit count values. The count increments at the frequency of the corresponding ICS307 programmable oscillator (OSC0 through to OSC6). The registers are set to zero by a reset.

Table 4-28 lists the register addresses.

Table 4-28 SYS_TEST_OSCx register

Register	Address	Access	Description
SYS_TEST_OSC0	0x100000C0	Read-only	Counter clocked from OSC0
SYS_TEST_OSC1	0x100000C4	Read-only	Counter clocked from OSC1
SYS_TEST_OSC2	0x100000C8	Read-only	Counter clocked from OSC2
SYS_TEST_OSC3	0x100000CC	Read-only	Counter clocked from OSC3
SYS_TEST_OSC4	0x100000D0	Read-only	Counter clocked from OSC4
SYS_TEST_OSC5	0x100000E4	Read-only	Counter clocked from OSC5
SYS_TEST_OSC6	0x100000E8	Read-only	Counter clocked from OSC6
SYS_TEST_OSC7	0x100000F4	Read-only	Counter clocked from OSC6

4.4 System Controller (SYSCTRL)

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

Table 4-29 SYSCTRL implementation

Property	Value
Location	Southbridge
Memory base address	• SYSCTRL 0: `0x10001000` • SYSCTRL 1: `0x1001A000`
Interrupt	–
DMA	–
Release version	ARM SYSCTRL SP810 r0p0
Reference documentation	*PrimeXsys System Controller (SP810) Technical Reference Manual* DDI 0254B See also *System Controller* on page 3-24.

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

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

4.4.1 PrimeCell modifications

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

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

SYS_CTRL0 register on page 4-43 lists the function of the bits in the SYS_CTRL0 register at address `0x10001000`.

Note

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

Table 4-30 SYS_CTRL0 register

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

Table 4-31 on page 4-44 lists the function of the bits in the SYS_CTRL1 register at address `0x1001A000`.

Table 4-31 SYS_CTRL1 register

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

4.5 Advanced Audio CODEC Interface, AACI

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

Table 4-32 AACI implementation

Property	Value
Location	Southbridge (the CODEC is an external component, LM4549)
Memory base address	`0x10004000`
Interrupt	51
DMA	• ACCI TX: channel 4 • ACCI RX: channel 5 **Note** You must set **DMAPSR** = `b00` in the SYS_DMAPSR register to select this peripheral for DMA access.
Release version	ARM AACI PL041 r1p0 (modified to one channel and 256 FIFO depth in compact mode and 512 FIFO depth in non-compact mode)
Platform Library support	No support provided.
Reference documentation	*ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual* DDI 0173 and *National Semiconductor LM4549 Data Sheet*. See also *Modified AACI PeriphID3 register* on page 4-46.

4.5.1 PrimeCell Modifications

The AACI PrimeCell in the Southbridge has a different FIFO depth than the standard PL041. Therefore, the AACIPeriphID3 register contains the values that Table 4-33 on page 4-46 lists.

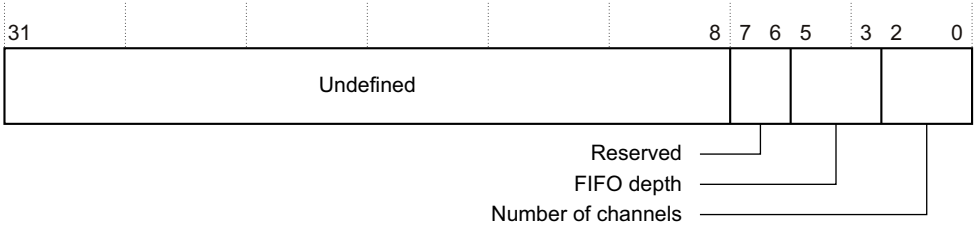


Figure 4-26 AACI ID register

Table 4-33 Modified AACI PeriphID3 register

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

4.6 Color LCD Controller, CLCDC

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

Table 4-34 CLCDC implementation

Property	Value
Location	Northbridge
Memory base address	`0x10020000` **Note** There is also a LCD system control register at `0x10000050`. See *CLCD Control Register, SYS_CLCD* on page 4-25.
Interrupt	55
DMA	–
Release version	ARM CLCDC PL111 (version r0p0)
Reference documentation	*ARM PrimeCell Color LCD Controller (PL111) Technical Reference Manual* DDI 0293.

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

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

Note

Different display resolutions require different data and synchronization timing. **OSCCLK4** (25MHz default) is assigned as **CLCDCLK** for the LCD controller. Default display resolution is 1024x768 at 60Hz frame rate. The default color depth is 16 bit. See *PrimeCell Color LCD Controller (PL111) Technical Reference Manual* (DDI 0293) for details of the LCD timing registers.

4.6.1 Display resolutions and display memory organization

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
XGA (1024x768)	63MHz, 0x35CF6	0x972F67FC	0x17030EFF	0x07FF3800

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

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

Note

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

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

The PBX-A9 has been tested at 1024 x 768 x 24-bit with a static color chart. However, practical resolution and color depth depend on available bus bandwidth.

4.7 Single Master Direct Memory Access Controller, SMDMAC

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

Table 4-36 SMDMAC implementation

Property	Value
Location	Northbridge
Memory base address	`0x10030000` for SMDMAC configuration `0x10000064` for SMDMAC mapping register SYS_DMAPSR
Interrupt	56
DMA	–
Release version	ARM DMAC PL081 r1p2
Reference documentation	*ARM PrimeCell Single Master DMA Controller (PL081) Technical Reference Manual*. See also *Dynamic memory controller, DMC* on page 3-17.

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

4.7.1 DMAC flow control

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

4.7.2 DMA channel allocation

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

4.8 DAP memory map

The Cortex-A9 structured ASIC contains a CoreSight debug interface. Table 4-37 lists the CoreSight Debug Memory Map.

Table 4-37 CoreSight APB memory map

Address	Description
0x1F400000	Undefined
0x1F31F000	Reserved
0x1F31E000	PTM2[a]
0x1F31D000	PTM1
0x1F31C000	PTM0
0x1F31B000	CTI3
0x1F31A000	CTI2
0x1F319000	CTI1
0x1F318000	CTI0
0x1F117000	Reserved
0x1F116000	Reserved
0x1F315000	PMU2[a]
0x1F314000	CoreDbg2[a]
0x1F313000	PMU1
0x1F312000	CoreDbg1
0x1F311000	PMU0
0x1F310000	CoreDbg0
0x1F300000	A9 Debug ROM table
0x12005000	ITM
0x1F204000	Funnel
0x1F203000	TPIU

Table 4-37 CoreSight APB memory map (continued)

Address	Description
0x1F20 2000	CTI
0x1F20 1000	ETB
0x1F20 0000	DAP ROM table

a. Only applies to the triple core ASIC. Otherwise reserved.

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 4-38 DMC implementation

Property	Value
Location	Northbridge
Memory base address	`0x100E0000`
Interrupt	–
DMA	–
Release version	ARM DMC PL340 r0p0
Reference documentation	*ARM PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual* See also *Remapping of boot memory* on page 4-10.

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

Note

The CD-ROM contains a Boot Monitor and platform library that initializes the DDR registers after power up (the `platform.a` library contains memory setup routines). See *Using the baseboard Boot Monitor and platform library* on page D-4 for details on using these utilities.

4.9.1 Register values

For detailed register descriptions, see the *ARM PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual*. Table 4-39 lists typical values.

Table 4-39 Typical DMC values for PL340 configuration registers

Register	Address	Value
Refresh period	`0x100E0010`	`0x200`
Cas Latency (2)	`0x100E0014`	`0x4`
t_dqss	`0x100E0018`	`0x1`

Table 4-39 Typical DMC values for PL340 configuration registers (continued)

Register	Address	Value
t_mrd	0x100E0020	0x2
t_ras	0x100E0024	0x3
t_rc	0x100E0028	0x4
t_rcd	0x100E002C	0x7
t_rfc	0x100E0030	0x1F2
t_rp	0x100E0034	0x15
t_rrd	0x100E0038	0x2
t_wr	0x100E003C	0x3
t_wtr	0x100E0040	0x2
t_xp	0x100E0044	0x1
t_xsr	0x100E0048	0x0A
t_esr	0x100E004c	0x14
config	0x100E0050	0x006D002A
chip_cfg0	0x100E0054	0x0FF

Caution

Care must be taken when adjusting these DMC values because incompatible values can cause erratic operation.

4.10 DDR2 Dynamic Memory Controller, DMC

The PL341 PrimeCell *DDR2 SDRAM Dynamic Memory Controller* (DMC) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-40 DDR2 DMC implementation

Property	Value
Location	Cortex-A9 structured ASIC
Memory base address	`0x1F401000`
Interrupt	–
DMA	–
Release version	ARM DMC PL341
Reference documentation	*ARM PrimeCell Dynamic Memory Controller (PL340) Technical Reference Manual* See also *Remapping of boot memory* on page 4-10.

The DMC controls the 512MB of DDR2 dynamic memory that is present on the daughterboard.

Note

Sample programs that configure and use dynamic memory can be found on the CD that accompanies the baseboard.

A Boot Monitor and platform library are provided in the `firmware` directory on the Versatile CD. The Boot Monitor initializes the DDR2 registers after power up (the `platform.a` library contains memory setup routines), see *Using the baseboard Boot Monitor and platform library* on page D-4 for details on using these utilities.

4.11 Ethernet

The PBX-A9 ethernet interface is implemented in an external SMCS LAN9118 10/100 Ethernet single-chip MAC and PHY. The internal registers of the LAN9118 are memory-mapped onto the static memory bus and occupy locations starting at base address `0x4E000000`.

Table 4-41 Ethernet implementation

Property	Value
Location	Board (LAN9118 chip)
Memory base address	`0x4E000000` (mapped onto the SMC bus)
Interrupt	60
DMA	None, use memory to memory DMA to access the FIFO buffers in the LAN9118 Host Bus Interface.
Release version	Custom interface to external controller.
Reference documentation	*LAN9118 Data Sheet* (see also *Ethernet interface* on page 3-28).

See the LAN9118 data sheet 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 and the unique MAC address is displayed on a sticker on the baseboard. A utility is provided on the Versatile CD to reprogram the MAC address if required.

4.12 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 4-42 GPIO implementation

Property	Value
Location	Southbridge
Memory base address	• GPIO 0: `0x10013000` • GPIO 1: `0x10014000` • GPIO 2: `0x10015000`
Interrupt	• GPIO 0: 38 • GPIO 1: 39 • GPIO 2: 40
DMA	NA
Release version	ARM GPIO PL061 r1p0
Reference documentation	*ARM PrimeCell General Purpose Input/Output (PL061) Technical Reference Manual* (see also *General Purpose Input/Output, GPIO* on page 3-23)

4.12.1 Onboard I/O control

GPIO2 is dedicated to the USB, push button, and MCI status signals as Table 4-43 lists.

Table 4-43 GPIO2 and MCI status signals

GPIO2 bit	Description
[7:5]	Reserved
[4]	USB Host Controller suspend/wakeup (pin 119 of ISP1761 USB Controller)
[3]	USB Device Controller suspend/wakeup (pin 120 of ISP1761 USB Controller)
[2]	General-Purpose push button
[1]	MCI Write-protect status
[0]	MCI Card-present status

4.13 Generic Interrupt Controller, GIC

The PBX-A9 baseboard *Generic Interrupt Controllers* (GIC) are AMBA compliant SoC peripherals that are developed and tested by ARM Limited.

Table 4-44 Generic Interrupt Controller implementation

Property	Value
Location	Southbridge (GIC2 and GIC3), Cortex-A9 structured ASIC (GIC)
Memory base address	`0x1E020000` GIC2 `0x1E030000` GIC3 `0x1F000000` GIC
Interrupt	GIC2 and GIC3 output **nIRQ** and **nFIQ** signals respectively to the tile site in response to an interrupt from a peripheral. GIC outputs nIRQ and nFIQ signals to the Cortex-A9 processors. **Note** The ICPICR register enables you to select the signals that drive the interrupt outputs of the Cortex-A9 processor interfaces in the GIC. See *Cortex-A9 MPCore Technical Reference Manual* (DDI 0407) for details.
DMA	–
Release version	ARM custom logic
Reference documentation	*Cortex-A9 MPCore Technical Reference Manual* (DDI 0407)

4.13.1 Generic interrupt controller registers

This section describes the GIC2 and GIC3 programming registers. See the *Cortex-A9 MPCore Technical Reference Manual* (DDI 0407) for details on programming the GIC in the Cortex-A9 CPU cluster.

Memory map

To access a GIC register, use the base address for the specific GIC that Table 4-45 lists together with the offset value for the specific CPU interface register that Table 4-46 lists or the specific Distributor register that Table 4-55 on page 4-66 lists.

Table 4-45 Interrupt control register addresses

Registers	Base Address
PBX-A9 GIC2 CPU interface registers	0x1E020000
PBX-A9 GIC2 distributor registers	0x1E021000
PBX-A9 GIC3 CPU interface registers	0x1E030000
PBX-A9 GIC3 distributor registers	0x1E031000

CPU Interface registers

Table 4-46 lists the CPU interface registers address offset values.

Table 4-46 CPU interface registers address offset values

Register	Offset Address
CPU control	0x000
Priority mask	0x004
Binary point	0x008
Interrupt acknowledge	0x00C
End of interrupt	0x010
Running interrupt	0x014
Highest pending interrupt	0x018

CPU control register

Figure 4-27 on page 4-60 shows the CPU control register.

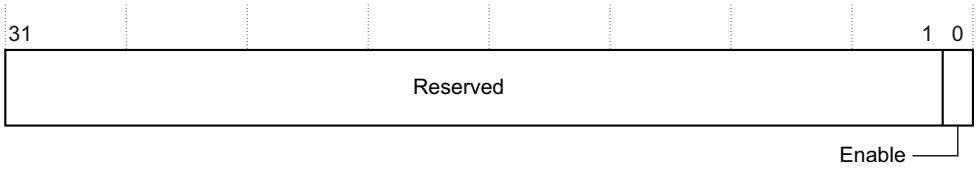


Figure 4-27 CPU control register

Table 4-47 lists the function of the register bits.

Table 4-47 CPU control register

Bits	Access	Name	Reset	Description
[31:4]	Write ignored, read as zero	–	0x0000000	Reserved
[3:2]	Write ignored, read as zero	–	b00	
[0]	Read/Write	Enable	b0	b0 = disable the CPU interface for this GICb1 = enable the CPU interface for this GIC

Priority mask register

Figure 4-28 shows the Priority mask register.

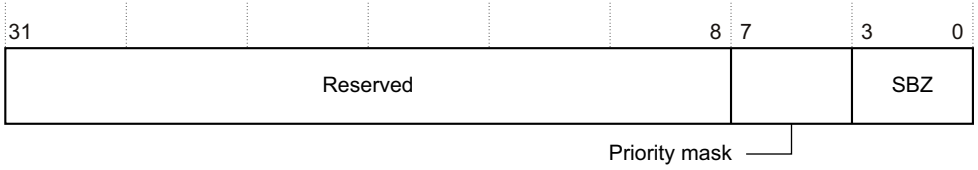


Figure 4-28 Priority mask register

Table 4-48 lists the function of the register bits.

Table 4-48 Priority mask

Bits	Access	Name	Reset	Description
[31:8]	Write ignored, read as zero	–	0x000000	Reserved
[7:4]	Read/Write	Priority mask	0x0	The Priority mask is used to prevent interrupts from being sent to the processor. The CPU Interface asserts an interrupt request if the priority of the highest pending interrupt sent by the Distributor is greater than the priority set in the Priority mask field. For example: 0x0 means all interrupts are masked. A Priority mask value of 0xF means interrupts with priority 0xF are masked but interrupts with higher priority values 0x0 to 0xE are not masked.
[3:0]	Write ignored, read as zero	–	0x0	Reserved

Binary point register

Figure 4-29 shows the Binary point register.

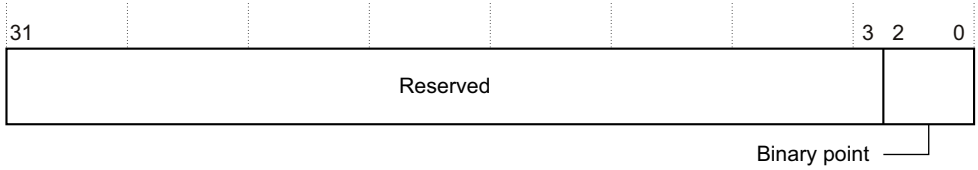


Figure 4-29 Binary point register

Table 4-49 lists the function of the register bits.

Table 4-49 Binary point

Bits	Access	Name	Reset	Description
[31:4]	Write ignored, read as zero	–	0x0000000	Reserved
[3]	Write ignored, read as zero	–	b0	Reserved
[2:0]	Read/Write	Binary point	b011	Sets the position of a 'binary point' that controls which bits of an interrupt's priority are compared for pre-emption purposes. This allows software to adjust the level of interrupt pre-emption in the system.

Table 4-50 lists the Binary point register bit assignments.

Table 4-50 Binary Point bit values assignment

Bit value	Meaning
b011	All priority bits are compared for pre-emption
b100	Only bits [7:5] of priority are compared for pre-emption
b101	Only bits [7:6] of priority are compared for pre-emption
b110	Only bit [7] of priority is compared for pre-emption
b111	No pre-emption is performed

Note

Writing a value that Table 4-50 does not list has the same effect as writing b011.

Figure 4-30 on page 4-63 shows an example of Binary point register operation.

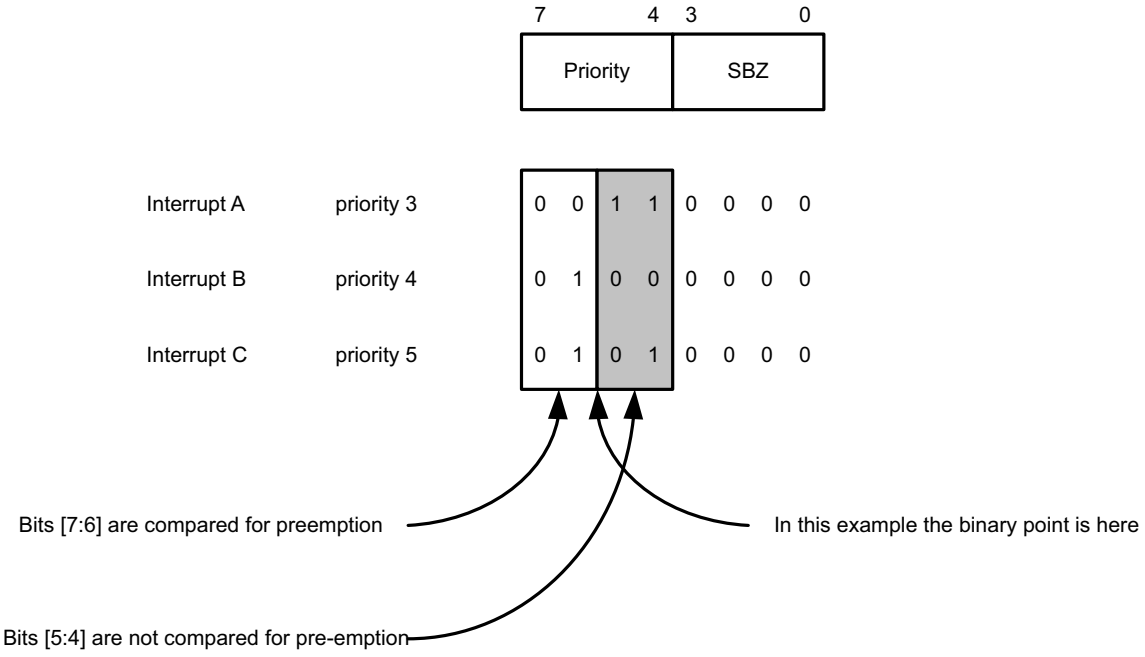


Figure 4-30 Binary point example

In this example there are three interrupts A, B, C that use 4 priority bits, and the Binary point register is set to `0x00000005` so the *binary point* is set between bit-6 and bit-5.

Zero (`b0000`) is the highest priority, so A is a higher priority interrupt than B and C. For pre-emption, any bits to the right of the *binary point* are ignored, so A can interrupt B or C, but B cannot interrupt C.

If interrupt A is active and interrupts B and C are pending, when A has completed B is taken because it has a higher priority than C.

Interrupt acknowledge

Figure 4-31 on page 4-64 shows the Interrupt acknowledge register.

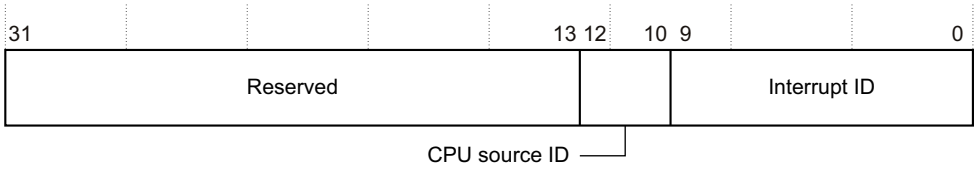


Figure 4-31 Interrupt acknowledge register

Table 4-51 lists the function of the register bits.

Table 4-51 Interrupt acknowledge

Bits	Access	Name	Reset	Description
[31:16]	Write ignored, read as zero	–	0x0000	Reserved
[15:13]	Write ignored, read as zero	–	b00	Reserved
[12:10]	Read-only	CPU source ID	b000	Reserved for multi-processor use
[9:0]	Read-only	Interrupt ID	b1111111111	The processor acquires the interrupt number by reading this register from the interrupting GIC. Pre-empted interrupts are recorded as active.

Note

In the event that interrupt priorities are changed before the processor reads the interrupt number, and the interrupt has become a lower priority, the interrupt number returned is 1023 to indicate a spurious interrupt.

End of interrupt

Figure 4-32 shows the End of interrupt register.

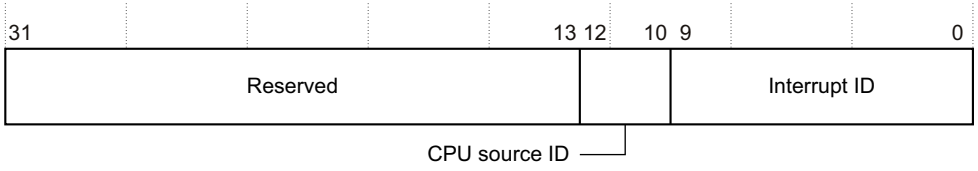


Figure 4-32 End of interrupt register

Table 4-52 lists the function of the register bits.

Table 4-52 End of interrupt

Bits	Access	Name	Reset	Description
[31:13]	WriteIgnored	–	–	Reserved
[12:10]	Write-only	CPU source ID	–	Reserved for multi-processor use
[9:0]	Write-only	Interrupt ID	–	When the interrupt has been completed by the processor, it writes the interrupt number to this register in the interrupting GIC.

Running interrupt

Figure 4-33 shows the Running interrupt register.

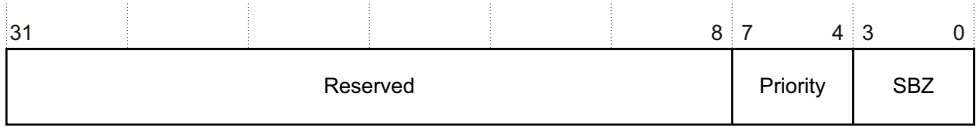


Figure 4-33 Running interrupt register

Table 4-53 lists the function of the register bits.

Table 4-53 Running interrupt

Bits	Access	Name	Reset	Description
[31:8]	Read-only	–	0x000000	Reserved
[7:4]	Read-only	Priority	0xF	Contains the priority level of the currently running interrupt.
[3:0]	Read-only	–	0x0	Reserved

Highest pending interrupt

Figure 4-34 shows the Highest pending interrupt register.

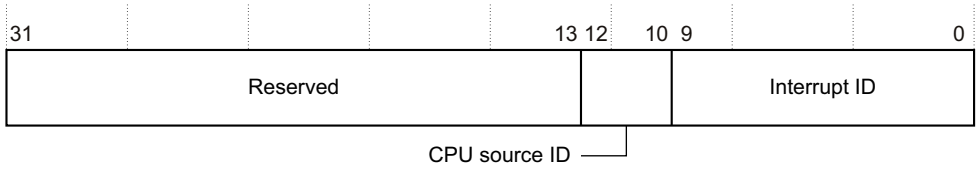


Figure 4-34 Highest pending interrupt register

Table 4-54 lists the function of the register bits.

Table 4-54 Highest pending interrupt

Bits	Access	Name	Reset	Description
[31:16]	Write ignored, read as zero	–	0x0000	Reserved
[15:13]	Write ignored, read as zero	–	b000	Reserved
[12:10]	Read-only	CPU source ID	b000	Reserved for multi-processor use
[9:0]	Read-only	Interrupt ID	b1111111111	The processor acquires the interrupt number of the highest pending interrupt being presented to the CPU Interface by the Distributor. If no interrupt is Pending then the Interrupt ID returned is1023, indicating a spurious interrupt.

Distribution registers

Table 4-55 lists the Distribution registers address offset values. See *Interrupt control register addresses* on page 4-59 for each GICs Distributor base address.

Table 4-55 Distribution registers address offset values

Distribution Register	Offset Address
Distributor control	0x000
Controller type	0x004
Reserved	0x008 – 0x0FC
Set-enable0 [a]	0x100
Set-enable1	0x104
Set-enable2	0x108
Reserved	0x10C – 0x17C
Clear-enable0 [a]	0x180
Clear-enable1	0x184
Clear-enable2	0x188
Reserved	0x18C – 0x1FC

Table 4-55 Distribution registers address offset values (continued)

Distribution Register	Offset Address
Set-pending0 [a]	0x200
Set-pending1	0x204
Set-pending2	0x208
Reserved	0x20C – 0x27C
Clear-pending0 [a]	0x280
Clear-pending1	0x284
Clear-pending2	0x288
Reserved	0x28C – 0x2FC
Active0 [a]	0x300
Active1	0x304
Active2	0x308
Reserved	0x30C – 0x3FC
Priority	0x400 – 0x45C
Reserved	0x460 – 0x7FC
CPU targets	0x800 – 0x85C
Reserved	0x860 –0xBFC
Configuration	0xC00 – 0xC14
Reserved	0xC18 – EFC
Software interrupt	0xF00
Reserved	0xF04 – 0xFFC

a. Private on PBX-A9.

Distributor control register

Figure 4-35 on page 4-68 shows the Distributor control register.

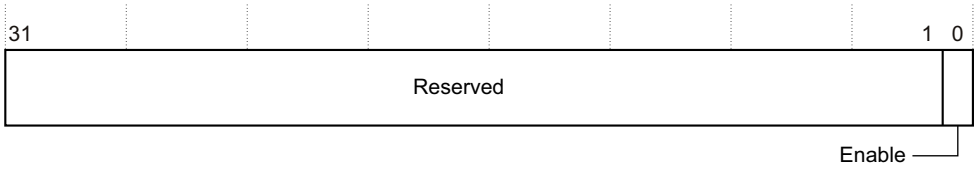


Figure 4-35 Distributor control register

Table 4-56 lists the function of the register bits.

Table 4-56 Distributor control

Bits	Access	Name	Reset	Description
[31:4]	Write ignored, read as zero	–	0x0000000	Reserved
[3:2]	Write ignored, read as zero	–	b00	Reserved
[0]	Read/Write	Enable	b0	b0 = interrupts are disable for this GICb1 = interrupts are enabled for this GIC

Controller type register

Figure 4-36 shows the Controller type register.

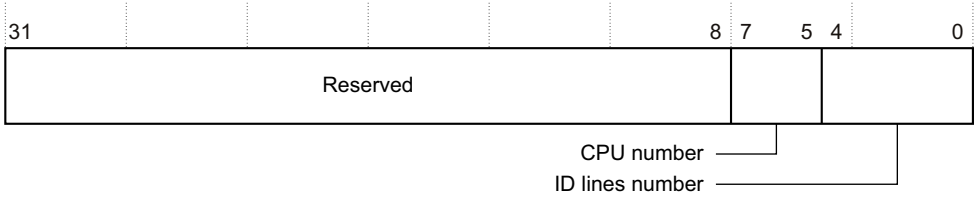


Figure 4-36 Controller type register

Table 4-57 lists the function of the register bits.

Table 4-57 Controller type

Bits	Access	Name	Reset	Description
[31:8]	Write ignored, read as zero	–	0x000000	Reserved
[7:5]	Read-only	CPU number	b000	Fixed value indicating a single CPU is serviced by this GIC.
[4:0]	Read-only	ID lines number	b00010	Fixed value indicating 64 external interrupt input lines are available for this GIC.

Set-enable0 register

The Set-enable0 register at address offset 0x100 is reserved for private use in the PBX-A9.

Set-enable1 register

Figure 4-37 shows the Set-enable1 register.

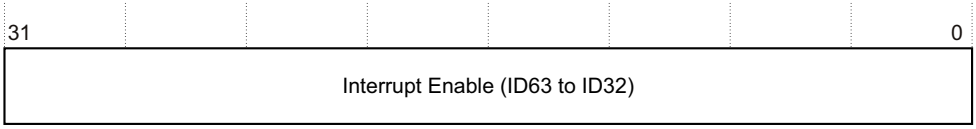


Figure 4-37 Set-enable1 register

Table 4-58 lists the function of the register bits.

Table 4-58 Set-enable1

Bits	Access	Name	Reset	Description
[31:0]	Read/Write	Interrupt Enable	0x00000000	Read this register to determine which interrupts are enabled. Bits 0 to 31 correspond to PBX-A9 interrupt input lines 32 to 63 respectively. A bit set to 1 indicates an enabled interrupt. Write a 1 to a bit to enable the corresponding interrupt. Use Read-Modify-Write to maintain reserved interrupt states. See *Interrupt allocations* on page 3-44 for details of the PBX-A9 external interrupt sources.

Note

There are a number of PBX-A9 reserved interrupt input lines that must not be enabled using the Set-enable1 register because the result is unpredictable. Table 4-59 lists the PBX-A9 reserved interrupt input lines for the corresponding Set-enable1 register bit.

Table 4-59 Reserved interrupts

Set-enable1Bit	Reserved Interrupt
[2]	34
[3]	35
[9]	41
[22]	54
[30]	62
[31]	63

Set-enable2 register

Figure 4-38 shows the Set-enable2 register.

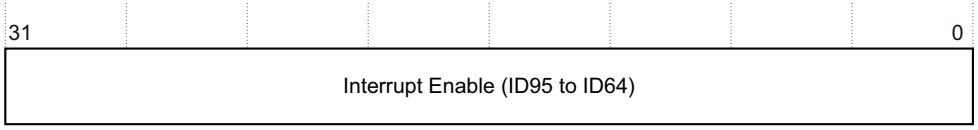


Figure 4-38 Set-enable2

Table 4-60 lists the function of the register bits.

Table 4-60 Set-enable2

Bits	Access	Name	Reset	Description
[31:0]	Read/Write	Interrupt Enable	0x00000000	Read this register to determine which interrupts are enabled. Bits 0 to 31 correspond to PBX-A9 interrupt input lines 64 to 95 respectively. A bit set to 1 indicates an enabled interrupt. Write a 1 to a bit to enable the corresponding interrupt. Use Read-Modify-Write to maintain reserved interrupt states. See *Interrupt allocations* on page 3-44 for details of the PBX-A9 external interrupt sources.

Note

There are a number of PBX-A9 reserved interrupt input lines that must not be enabled using the Set-enable2 register because the result is unpredictable. Table 4-61 lists the PBX-A9 reserved interrupt input lines for the corresponding Set-enable2 register bit.

Table 4-61 Reserved interrupts

Set-enable2 Bit	Reserved Interrupt
[11]	75
[12]	76
[13]	77
[14]	78
[15]	79
[26]	90
[27]	91
[28]	92
[29]	93
[30]	94
[31]	95

Clear-enable0 register

The Clear-enable0 register at address offset `0x180` is reserved for private use in the PBX-A9.

Clear-enable1 register

Figure 4-39 shows the Clear-enable1 register.

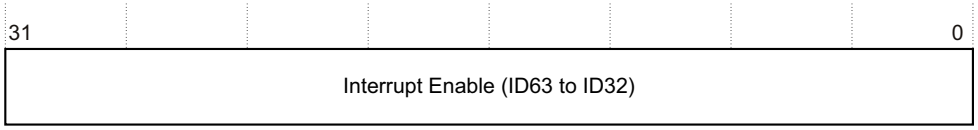


Figure 4-39 Clear-enable1 register

Table 4-62 lists the function of the register bits.

Table 4-62 Clear-enable1

Bits	Access	Name	Reset	Description
[31:0]	Read/Write	Interrupt Enable	`0x00000000`	Read this register to determine which interrupts are cleared. Bits 0 to 31 correspond to PBX-A9 interrupt input lines 32 to 63 respectively. A bit set to 0 indicates a cleared interrupt. Write a 1 to a bit to clear the corresponding interrupt. Use Read-Modify-Write to maintain reserved interrupt states. See *Interrupt allocations* on page 3-44 for details of the PBX-A9 external interrupt sources.

Clear-enable2 register

Figure 4-40 shows the Clear-enable2 register.

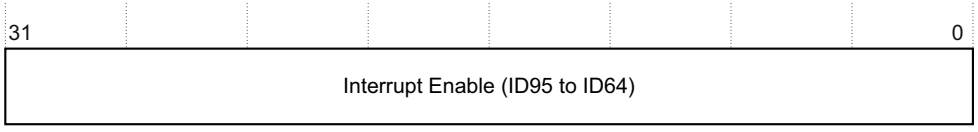


Figure 4-40 Clear-enable2 register

Table 4-63 lists the function of the register bits.

Table 4-63 Clear-enable2

Bits	Access	Name	Reset	Description
[31:0]	Read/Write	Interrupt Enable	0x00000000	Read this register to determine which interrupts are cleared. Bits 0 to 31 correspond to PBX-A9 interrupt input lines 64 to 95 respectively. A bit set to 0 indicates a cleared interrupt. Write a 1 to a bit to clear the corresponding interrupt. Use Read-Modify-Write to maintain reserved interrupt states. See *Interrupt allocations* on page 3-44 for details of the PBX-A9 external interrupt sources.

Set-pending0 register

The Set-pending0 register at address offset 0x200 is reserved for private use in the PBX-A9.

Set-pending1 register

Figure 4-41 shows the Set-pending1 register.

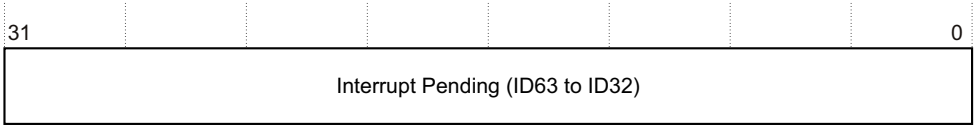


Figure 4-41 Set-pending1 register

Table 4-64 lists the function of the register bits.

Table 4-64 Set-pending1

Bits	Access	Name	Reset	Description
[31:0]	Read/Write	Interrupt Pending	0x00000000	Read this register to determine which interrupts are pending. Bits 0 to 31 correspond to PBX-A9 interrupt input lines 32 to 63 respectively. A bit set to 1 indicates a pending interrupt. Write a 1 to a bit to set the corresponding interrupt into *Pending* state. Use Read-Modify-Write to maintain reserved interrupt states. See *Interrupt allocations* on page 3-44 for details of the PBX-A9 external interrupt sources.

Set-pending2 register

Figure 4-42 on page 4-74 shows the Set-pending2 register.

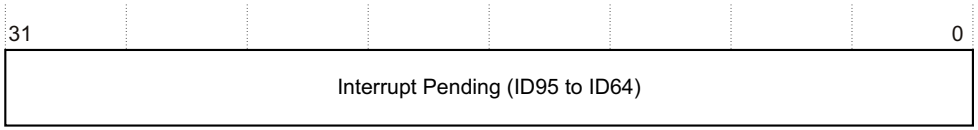


Figure 4-42 Set-pending2 register

Table 4-65 lists the function of the register bits.

Table 4-65 Set-pending2

Bits	Access	Name	Reset	Description
[31:0]	Read/Write	Interrupt Pending	`0x00000000`	Read this register to determine which interrupts are pending. Bits 0 to 31 correspond to PBX-A9 interrupt input lines 64 to 95 respectively. A bit set to 1 indicates a pending interrupt. Write a 1 to a bit to set the corresponding interrupt into *Pending* state. Use Read-Modify-Write to maintain reserved interrupt states. See *Interrupt allocations* on page 3-44 for details of the PBX-A9 external interrupt sources.

Clear-pending0 register

The Clear-pending0 register at address offset `0x280` is reserved for private use in the PBX-A9.

Clear-pending1 register

Figure 4-43 shows the Clear-pending1 register.

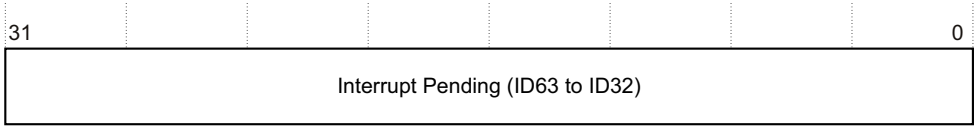


Figure 4-43 Clear-pending1 register

Table 4-66 lists the function of the register bits.

Table 4-66 Clear-pending1

Bits	Access	Name	Reset	Description
[31:0]	Read/Write	Interrupt Pending	0x00000000	Read this register to determine which interrupts are pending. Bits 0 to 31 correspond to PBX-A9 interrupt input lines 32 to 63 respectively. A bit set to 1 indicates a pending interrupt. Write a 1 to a bit to set the corresponding interrupt into the *Inactive* state. Use Read-Modify-Write to maintain reserved interrupt states. See *Interrupt allocations* on page 3-44 for details of the PBX-A9 external interrupt sources.

Clear-pending2 register

Figure 4-44 shows the Clear-pending2 register.

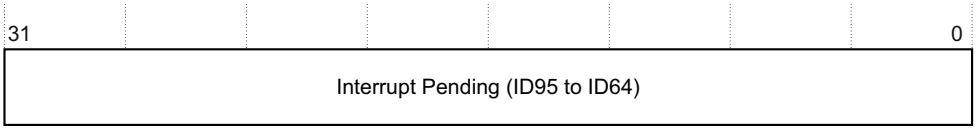


Figure 4-44 Clear-pending2 register

Table 4-67 lists the function of the register bits.

Table 4-67 Clear-pending2

Bits	Access	Name	Reset	Description
[31:0]	Read/Write	Interrupt Pending	0x00000000	Read this register to determine which interrupts are pending. Bits 0 to 31 correspond to PBX-A9 interrupt input lines 64 to 95 respectively. A bit set to 1 indicates a pending interrupt. Write a 1 to a bit to set the corresponding interrupt into the *Inactive* state. Use Read-Modify-Write to maintain reserved interrupt states. See *Interrupt allocations* on page 3-44 for details of the PBX-A9 external interrupt sources.

Active0 register

The Active0 register at address offset 0x300 is reserved for private use in the PBX-A9.

Active1 register

Figure 4-45 on page 4-76 shows the Active1 register.

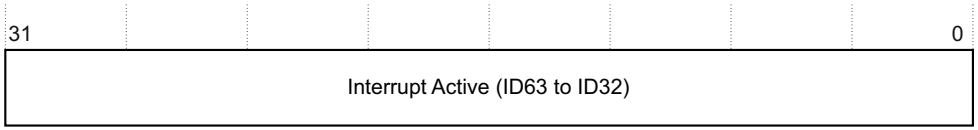


Figure 4-45 Active1 register

Table 4-68 lists the function of the register bits.

Table 4-68 Active1

Bits	Access	Name	Reset	Description
[31:0]	Read-only	Interrupt Active	0x00000000	Read this register to determine which interrupts are active. Bits 0 to 31 correspond to PBX-A9 interrupt input lines 32 to 63 respectively. A bit set to 1 indicates an interrupt is in the *Active* state. See *Interrupt allocations* on page 3-44 for details of the PBX-A9 external interrupt sources.

Active2 register

Figure 4-46 shows the Active2 register.

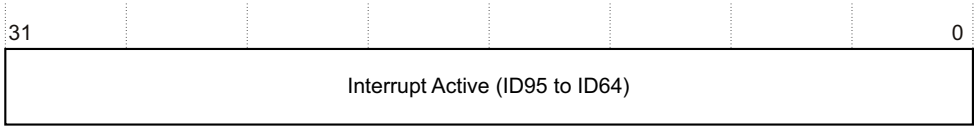


Figure 4-46 Active2 register

Table 4-69 lists the function of the register bits.

Table 4-69 Active2

Bits	Access	Name	Reset	Description
[31:0]	Read-only	Interrupt Active	0x00000000	Read this register to determine which interrupts are active. Bits 0 to 31 correspond to PBX-A9 interrupt input lines 64 to 95 respectively. A bit set to 1 indicates an interrupt is in the *Active* state. See *Interrupt allocations* on page 3-44 for details of the PBX-A9 external interrupt sources.

Priority registers

There are 24 Priority registers. Each register holds the priority level for 4 interrupt IDs. Priority registers 0 to 7 at address offsets 0x400 to 0x41C are reserved for Interrupt IDs 0 to 31 that are for private use in the PBX-A9. Priority registers 8 to 23 at address offsets 0x420 to 0x45C hold priority levels for Interrupt IDs 32 to 95 respectively. Table 4-70 lists Priority registers 8 to 23 address offsets and Interrupt IDs.

Table 4-70 Priority register address offsets and Interrupt IDs

Priority Register	Address Offset	Interrupt ID
Priority8	0x420	ID32 − ID35
Priority9	0x424	ID36 − ID39
Priority10	0x428	ID40 − ID43
Priority11	0x42C	ID44 − ID47
Priority12	0x430	ID48− ID51
Priority13	0x434	ID52 − ID55
Priority14	0x438	ID56− ID59
Priority15	0x43C	ID60 − ID63
Priority16	0x440	ID64 − ID67
Priority17	0x444	ID68 − ID71
Priority18	0x448	ID72 − ID75
Priority19	0x44C	ID76 − ID79
Priority20	0x450	ID80 − ID83
Priority21	0x454	ID84 − ID87
Priority22	0x458	ID88 − ID91
Priority23	0x45C	ID92 − ID95

Figure 4-47 on page 4-78 shows the Priority register fields.

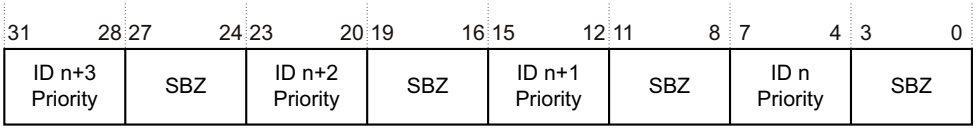


Figure 4-47 Priority register

Note

Priority is a 4-bit number where zero is the highest priority. Future implementations of the GIC might use 8-bits for priority so the priority is stored in the most significant bits of an 8-bit field.

Each 32-bit register holds the priority for 4 interrupts. For example, interrupt ID 32 − 35 are stored in Priority8 register at address offset `0x420`:

- Interrupt ID 32 is stored in bits [7:4], bits 3:0 are unused and set to zero
- Interrupt ID 33 is stored in bits [15:12], bits [11:8] are unused and set to zero
- Interrupt ID 34 is stored in bits [23:20], bits [19:16] are unused and set to zero
- Interrupt ID 35 is stored in bits [31:28], bits [27:24] are unused and set to zero.

CPU targets registers

There are 24 CPU targets registers. Each register holds the target CPU data for 4 interrupt IDs. CPU targets registers 0 to 7 at address offsets `0x800` to `0x81C` are reserved for Interrupt IDs 0 to 31 that are for private use in the PBX-A9. CPU targets registers 8 to 23 at address offsets `0x820` to `0x85C` hold CPU targets data for Interrupt IDs 32 to 95 respectively. Table 4-71 lists CPU targets registers 8 to 23, address offsets, and Interrupt IDs.

Table 4-71 CPU targets register address offsets and Interrupt IDs

Priority Register	Address Offset	Interrupt ID
CPUtargets8	`0x820`	ID32 − ID35
CPUtargets9	`0x824`	ID36 − ID39
CPUtargets10	`0x828`	ID40 − ID43
CPUtargets11	`0x82C`	ID44 − ID47
CPUtargets12	`0x830`	ID48− ID51
CPUtargets13	`0x834`	ID52 − ID55
CPUtargets14	`0x838`	ID56− ID59

Table 4-71 CPU targets register address offsets and Interrupt IDs (continued)

Priority Register	Address Offset	Interrupt ID
CPUtargets15	0x83C	ID60 − ID63
CPUtargets16	0x840	ID64 − ID67
CPUtargets17	0x844	ID68 − ID71
CPUtargets18	0x848	ID72 − ID75
CPUtargets19	0x84C	ID76 − ID79
CPUtargets20	0x850	ID80 − ID83
CPUtargets21	0x854	ID84 − ID87
CPUtargets22	0x858	ID88 − ID91
CPUtargets23	0x85C	ID92 − ID95

Figure 4-48 shows the CPU targets register fields.

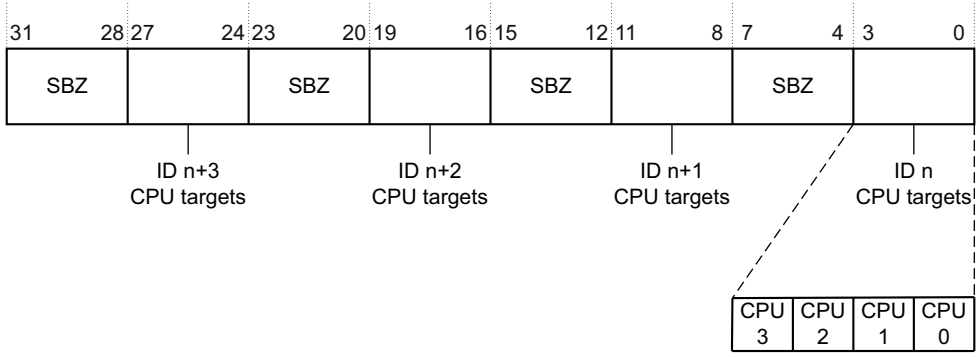


Figure 4-48 CPU targets register

Each register can store a bit-map for 4 Interrupt IDs x 4 CPUs. The GICs in the PBX-A9 have been implemented for one target CPU (CPU 0) at the tile site so the CPU target registers are initialized to 0x01010101.

Configuration register

There are 6 Configuration registers. Each register holds the configuration data for 16 interrupt IDs. Configuration registers 0 and 1 at address offsets 0xC00 and 0xC04 are reserved for Interrupt IDs 0 to 31 that are for private use in the PBX-A9. Configuration

registers 2 to 5 at address offsets 0xC08 to 0xC14 hold configuration data for Interrupt IDs 32 to 95 respectively. Table 4-72 lists Configuration registers 2 to 5, address offsets, and Interrupt IDs.

Table 4-72 Configuration register address offsets

Configuration register	Address Offset	Interrupt IDs
Configuration2	0xC08	ID32 − ID47
Configuration3	0xC0C	ID48 − ID63
Configuration4	0xC10	ID64 − ID79
Configuration5	0xC14	ID80 − ID95

Figure 4-49 shows the Configuration register fields.

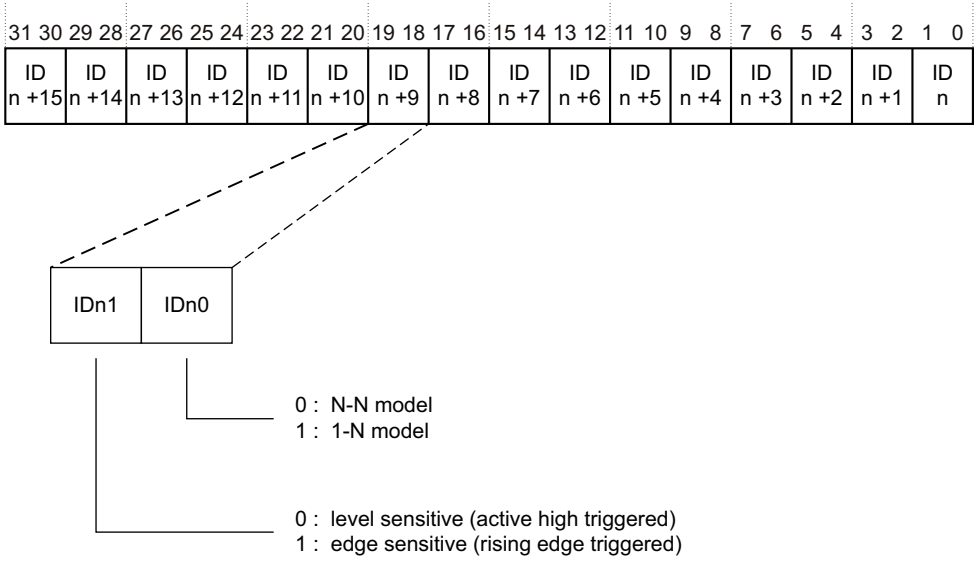


Figure 4-49 Configuration register

The Interrupt Configuration registers have 2 bits IDn[1:0] for each interrupt ID n. These two bits set each interrupt to be level or edge sensitive, and determine which *software model* is used:

1-N model Only one CPU takes the interrupt. An interrupt that is taken up by any CPU clears the pending status on all CPUs.

Note

This is the only model available for the PBX-A9 GIC implementation because only a single CPU is supported at the tile site.

N-N model This model has been deprecated and must not be used.

For example, for Interrupt ID 32 set bits [1:0] of Configuration2 register at address offset `0xC08` to `b01` for level-sensitive or `b11` for edge sensitive. The default is all interrupts are level-sensitive, set by the Boot Monitor writing `0x55555555` to the Configuration registers.

Software interrupt register

This is a write-only register. Write to this register to trigger an interrupt ID32 − ID95.

Figure 4-50 shows the Software interrupt register fields.

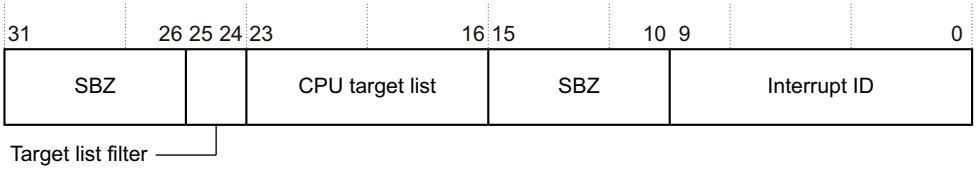


Figure 4-50 Software interrupt register

Table 4-73 on page 4-82 lists the function of the register bits.

Caution

If you attempt to trigger an interrupt with an ID larger than the number of supported interrupts, or that references a CPU that is not present, there can be unpredictable results in the Distributor.

Table 4-73 Software interrupt

Bits	Access	Name	Reset	Description
[31:26]	Write-only	–	–	Reserved
[25:24]	Write-only	Target list filter	–	The filter options are: 00: Interrupt sent to CPUs listed in CPU target list. 01: CPU target list is ignored, interrupt is sent to all but the requesting CPU. 10: CPU target list is ignored, interrupt is sent to the requesting CPU only. 11: Reserved. Valid entries for the PBX-A9 are:`b00` if CPU target list = `b00000001b10` otherwise.
[23:16]	Write-only	CPU target list	–	There can be up to 8 CPU targets. PBX-A9 has 1 CPU target only at the tile site. Valid entry is: `b00000001` (CPU0)
[15:10]	Write-only	–	–	Reserved
[9:0]	Write-only	Interrupt ID	–	ID of interrupt to be triggered. Valid range for the PBX-A9 is ID32 to ID95:`b0000100000` to `b0001011111`

For example, write `0x02000021` to the PBX-A9 Software interrupt register to trigger Interrupt ID 33. Bits [9:0] contain the interrupt ID, bits [25:24] are set to `b10` to ignore the list of CPU targets in bits [23:16]. The remaining bits must be set to zero. You must see the *Set-pending1* register at offset `0x204` set to `0x00000002`.

4.13.2 Handling interrupts

This section describes interrupt handling and clearing in general.

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

All interrupts are routed to all GICs.

The sequence to determine and clear an interrupt is:

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

Warning

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

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

Note

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

4.14 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 4-74 KMI implementation

Property	Value
Location	Southbridge
Memory base address	`0x10006000` KMI 0 (keyboard) `0x10007000` KMI 1 (mouse)
Interrupt	KMI0 52 KMI1 53
DMA	–
Release version	ARM KMI PL050 r1p0
Reference documentation	*ARM PrimeCell PS2 Keyboard/Mouse Interface (PL050) Technical Reference Manual*

4.15 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 4-75 MCI implementation

Property	Value
Location	Southbridge
Memory base address	`0x10005000`
Interrupt	49 for MCIa 50 for MCIb
DMA	3 (DMAPSR0 set to b00)
Release version	ARM MCI PL180 r1p0
Reference documentation	*ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual*

The interrupts for the MCI card are managed by the GPIO 2 PrimeCell. See *General Purpose Input/Output, GPIO* on page 4-57.

4.16 AXI to PCI and PCI to PCIx bridges

The AXI to PCI and PCI to PCIx bridges are implemented in the PBX-A9 Northbridge.

Table 4-76 AXI to PCI bridge implementation

Property	Value
Location	Northbridge
Memory base address	`0x90000000` `0x60000000` (reserved for PCI expansion)
Interrupt	82 P_nINT[0] - Slot A **P_nINTD** or Slot B **P_nINTC** 83 P_nINT[1] - Slot A **P_nINTA** or Slot B **P_nINTD** 84 P_nINT[2] - Slot A **P_nINTB** or Slot B **P_nINTA** 85 P_nINT[3] - Slot A **P_nINTD** or Slot B **P_nINTB** 86 P_nINT[4] - PEX **P_nINTA** 87 P_nINT[5] - PEX **P_nINTB** 88 P_nINT[6] - PEX **P_nINTC** 89 P_nINT[7] - PEX **P_nINTD**
DMA	None. Memory to memory transfers can be set up in the DMAC.
Release version	-
Reference documentation	-

Table 4-77 lists the windows that provide access to the PCI expansion bus.

Table 4-77 PCI bus memory map

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

The AXI to PCI bridge enables you to use the PBX-A9 with third-party PCI or PCI-Express expansion cards. PBX-A9 functions as a PCI host, that is, it generates clocks to the PCI or PCI-Express card.

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

Note

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

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

4.17 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 4-78 RTC implementation

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

Note

There is also a separate *Time-of-Year* (TOY) implemented in an external RTC chip (DS1338) on the baseboard. The external RTC can be accessed by the serial bus interface (see *Two-wire serial bus interface, SBCon* on page 4-89). For details on the programming interface to the Time-of-Year RTC, see the data sheet for the Maxim DS1338 integrated circuit (`www.maxim-ic.com`).

4.18 Two-wire serial bus interface, SBCon

Two custom two-wire serial bus interfaces (SBCon 0 and SBCon 1) are implemented in the PBX-A9 Southbridge.

SBCon 0 provides access to the Maxim DS1338 RTC on the baseboard.

SBCon 1 provides access to the Digital Data Channel (DDC) of the external display connected to the DVI connector on the rear panel.

Table 4-79 Serial bus implementation

Property	Value
Location	Southbridge
Memory base address	SBCon 0: `0x10002000` SBCon 1: `0x10016000`
Interrupt	–
DMA	–
Release version	Custom logic
Reference documentation	• *Two-wire serial bus interface* on page 3-26 • data sheet for the Maxim DS1338 RTC (`www.maxim-ic.com`) • *VESA DDC Specification Version 3.0*

Table 4-80 lists the registered device addresses.

Table 4-80 Serial interface device addresses

Device	Write address	Read address	Description
TOY (DS1338 RTC)	`0xD0`	`0xD1`	Reads time data and writes control data to the RTC.
DVI (external display)	display dependant	display dependant	Reads external display capabilities at the DVI connector. Can control display settings of *E-DDC* displays.

Table 4-81 and Table 4-82 list the registers that control the serial bus interfaces.

Table 4-81 SBCon 0 serial bus register

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

Table 4-82 SBCon 1 serial bus register

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

Note

Software must manipulate the **SCL** and **SDA** bits directly to access the data in the devices. **SDA** is an open-collector signal that is used for sending and receiving data. Set the output (sending) value HIGH before reading the current value.

4.19 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 4-83 SCI implementation

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

See the self-test software that is supplied on the CD accompanying the PBX-A9 for an example of detecting a SIM card response to a reset.

4.20 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 4-84 SSP implementation

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

4.21 Static Memory Controller, SMC

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

Table 4-85 SMC implementation

Property	Value
Location	Northbridge
Memory base address	`0x100E1000`
Interrupt	–
DMA	–
Release version	ARM SMC PL354 r0p0
Reference documentation	*ARM PrimeCell Static Memory Controller (PL350 series) Technical Reference Manual*, *Configuration and initialization* on page 4-10

4.22 Timers

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

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

Table 4-86 Timer implementation

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

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

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 4-87 UART implementation

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

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

- UART0 has full handshaking signals (RTS, CTS, DSR, DTR, DCD and RI)
- Handshaking signals for UART1-3 consist of only RTS and CTS
- 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.

4.24 USB interface

The PBX-A9 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 4-88 USB implementation

Property	Value
Location	Board (an ISP1761 chip)
Memory base address	`0x4F000000` (mapped onto the SMC bus)
Interrupt	61
DMA	There are two DMA channels available for the USB controller. These are selectable as 0 or 1. See *Single Master Direct Memory Access Controller, SMDMAC* on page 4-50. **Note** You must set **DMAPSR** = `b00` in the SYS_DMAPSR register to select this peripheral for DMA access.
Release version	Custom interface to external controller
Reference documentation	*ISP1761 Hi-Speed Universal Serial Bus On-The-Go controller Product data sheet* (see also *USB Interface* on page 3-29 and test program supplied on the CD)

The ISP1761 has the following features:

- 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 supports 480Mb/s, 12Mb/s, and 1.5Mb/s
- programmable interrupts and DMA
- FIFO and 63KB on-chip RAM for USB.

Table 4-89 shows the ISP1761 register base addresses.

Table 4-89 USB controller base address

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

Note

The suspend/wakeup signals for the device and host controllers are connected to GPIO2 (see *General Purpose Input/Output, GPIO* on page 4-57).

4.25 Watchdog

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

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

Table 4-90 Watchdog implementation

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

Note

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

4.26 CompactFlash interface

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

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

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

Table 4-91 CompactFlash implementation

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

4.26.1 CompactFlash Control Register, CF_CTRL

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

Figure 4-51 on page 4-101 shows the register bit allocations.

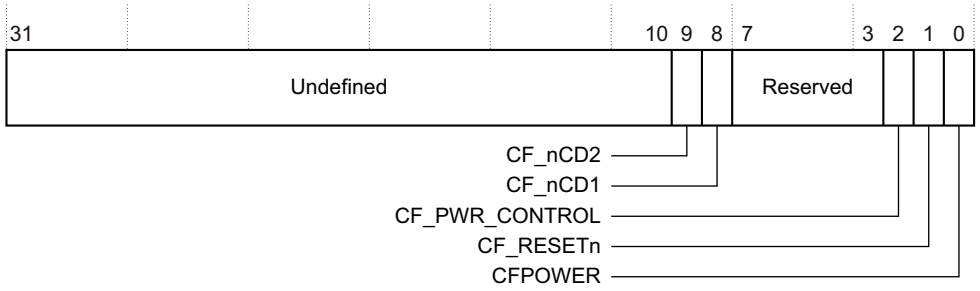


Figure 4-51 CF_CTRL Register

Table 4-92 shows the function of the register bits.

Table 4-92 CF_CTRL register bit assignments

Bits	Access	Name	Reset	Description
[31:12]	Write ignored, read as zero	–	0x00000	Undefined
[11;10]	Write ignored, read as zero	–	b00	Undefined
[9]	Read only	CF_nCD2	b1	Card Detection:b00: card insertedbx1: card not insertedb1x: card not inserted
[8]	Read only	CF_nCD1	b1	
[7:3]	Write ignored, read as zero	–	b00000	Reserved
[2]	Read/Write	CF_PWR_CONTROL	b0	Power Control:b0: determined by CFPOWER (bit 0)b1: determined by chip detect (CF card)
[1]	Read/Write	CF_RESETn	b0	Card Reset (active low)
[0]	Read/Write	CFPOWER	b0	Card Power:b0: no power applied to cardb1: 3V3 applied to card

Appendix A
Signal Descriptions

This appendix provides a summary of signals present on the PBX-A9 connectors. It contains the following sections:

A.1 CompactFlash interface

The PBX-A9 baseboard provides a CompactFlash connector on the front panel that enables you to connect a CompactFlash Storage Card to the CompactFlash interface. Figure A-1 shows the connector pin numbering.

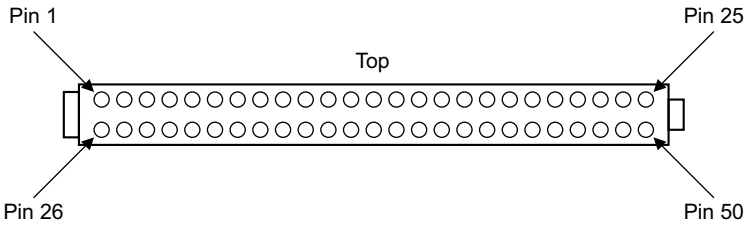


Figure A-1 CompactFlash connector pin numbering

The CompactFlash interface supports the three basic CompactFlash Storage Card modes:

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

Some pin functions change dependant on the mode, and some also change dependant on the interface protocol used. For example, when PC Card I/O Mode is used, pin 34 has the following three protocol dependant functions:

~IORD If Ultra DMA Protocol is not active: This is an I/O Read strobe generated by the host.

~HDMARDY When Ultra DMA Protocol DMA Read is active: This signal indicates that the host is ready to receive Ultra DMA data-in bursts.

HSTROBE When Ultra DMA Protocol DMA Write is active: This signal is the data out strobe generated by the host.

Table A-1 on page A-3 lists the CompactFlash connector pinout for each supported mode. See the *CF+ & CF Specification Rev 4.1* for signal descriptions.

Table A-1 CompactFlash connector pinout

PC Card Memory Mode			PC Card I/O Mode			True IDE Mode		
Pin	**Signal**	**Type**	**Pin**	**Signal**	**Type**	**Pin**	**Signal**	**Type**
1	**GND**		1	**GND**		1	**GND**	
2	**D03**	I/O	2	**D03**	I/O	2	**D03**	I/O
3	**D04**	I/O	3	**D04**	I/O	3	**D04**	I/O
4	**D05**	I/O	4	**D05**	I/O	4	**D05**	I/O
5	**D06**	I/O	5	**D06**	I/O	5	**D06**	I/O
6	**D07**	I/O	6	**D07**	I/O	6	**D07**	I/O
7	**~CE1**	I	7	**~CE1**	I	7	**~CE0**	I
8	**A10**	I	8	**A10**	I	8	**A10**	I
9	**~OE**	I	9	**~OE**	I	9	**~ATA_SEL**	I
10	**A09**	I	10	**A09**	I	10	**A09**	I
11	**A08**	I	11	**A08**	I	11	**A08**	I
12	**A07**	I	12	**A07**	I	12	**A07**	I
13	**VCC**		13	**VCC**		13	**VCC**	
14	**A06**	I	14	**A06**	I	14	**A06**	I
15	**A05**	I	15	**A05**	I	15	**A05**	I
16	**A04**	I	16	**A04**	I	16	**A04**	I
17	**A03**	I	17	**A03**	I	17	**A03**	I
18	**A02**	I	18	**A02**	I	18	**A02**	I
19	**A01**	I	19	**A01**	I	19	**A01**	I
20	**A00**	I	20	**A00**	I	20	**A00**	I
21	**D00**	I/O	21	**D00**	I/O	21	**D00**	I/O
22	**D01**	I/O	22	**D01**	I/O	22	**D01**	I/O
23	**D02**	I/O	23	**D02**	I/O	23	**D02**	I/O
24	**WP**	O	24	**~IOIS16**	O	24	**~IOCS16**	O

Table A-1 CompactFlash connector pinout (continued)

PC Card Memory Mode			PC Card I/O Mode			True IDE Mode		
Pin	**Signal**	**Type**	**Pin**	**Signal**	**Type**	**Pin**	**Signal**	**Type**
25	**~CD2**	O	25	**~CD2**	O	25	**~CD2**	O
26	**~CD1**	O	26	**~CD1**	O	26	**~CD1**	O
27	**D11**	I/O	27	**D11**	I/O	27	**D11**	I/O
28	**D12**	I/O	28	**D12**	I/O	28	**D12**	I/O
29	**D13**	I/O	29	**D13**	I/O	29	**D13**	I/O
30	**D14**	I/O	30	**D14**	I/O	30	**D14**	I/O
31	**D15**	I/O	31	**D15**	I/O	31	**D15**	I/O
32	**~CE2**	I	32	**~CE2**	I	32	**~CS1**	
33	**~VS1**	O	33	**~VS1**	O	33	**~VS1**	O
34	**~IORDHSTROBE~ HDMARDY**	I	34	**~IORDHSTROBE~ HDMARDY**	I	34	**~IORDHSTROBE~HDMARDY**	I
35	**~IOWRSTOP**	I	35	**~IOWRSTOP**	I	35	**~IOWRSTOP**	I
36	**~WE**	I	36	**~WE**	I	36	**~WE**	I
37	**READY**	O	37	**~IREQ**	O	37	**INTRQ**	O
38	**VCC**		38	**VCC**		38	**VCC**	
39	**~CSEL**	I	39	**~CSEL**	I	39	**~CSEL**	I
40	**~VS2**	O	40	**~VS2**	O	40	**~VS2**	O
41	**RESET**	I	41	**RESET**	I	41	**RESET**	I
42	**~WAIT~DDMARDY DSTROBE**	O	42	**~WAIT~DDMARDY DSTROBE**	O	42	**IORDY~DDMARDYDSTROBE**	O
43	**~INPACK~DMARQ**	O	43	**~INPACK~DMARQ**	O	43	**DMARQ**	O
44	**~REG~DMACK**	I	44	**~REGDMACK**	I	44	**~DMACK**	I
45	**BVD2**	O	45	**~SPKR**	O	45	**~DSAP**	I/O
46	**BVD1**	O	46	**~STSCHG**	O	46	**~PDIAG**	I/O

Table A-1 CompactFlash connector pinout (continued)

PC Card Memory Mode			PC Card I/O Mode			True IDE Mode		
Pin	**Signal**	**Type**	**Pin**	**Signal**	**Type**	**Pin**	**Signal**	**Type**
47	**D08**	I/O	47	**D08**	I/O	47	**D08**	I/O
48	**D09**	I/O	48	**D09**	I/O	48	**D09**	I/O
49	**D10**	I/O	49	**D10**	I/O	49	**D10**	I/O
50	**GND**		50	**GND**		50	**GND**	

A.2 Audio CODEC interface

The PBX-A9 baseboard provides three stacked 3.5mm jack connectors on the rear panel that enable you to connect to the analog microphone and auxiliary line level input and output on the CODEC. If no jack plug is inserted, the tip and sleeve of both the Mic In and Line In jack sockets are connected to analog ground to help prevent noise pickup.Figure A-2 shows the pinouts of the sockets. The *Rear panel layout* on page 3-5 shows the connector location.

Note

A link (LK3) on the baseboard enables a 5V bias voltage to be applied to the microphone.

The available link options are:

- Fit A-B For BIAS at the tip (standard active mic)
- Fit B-C For BIAS at the middle sleeve
- Omit For no BIAS (passive microphone).

When no plug is inserted, both the Microphone and Line In jack sockets tip and sleeve are connected to analog ground to avoid noise pickup

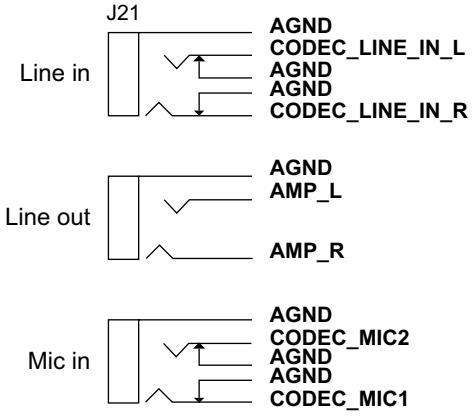


Figure A-2 Audio connectors

A.3 MMC and SD card interface

The PBX-A9 baseboard MMC/SD card socket that Figure A-3 shows is positioned on the front panel of the enclosure, see *Front panel layout* on page 3-4.

Caution

The MMC or SD card must be inserted into the front panel socket with the contacts facing down.

Figure A-3 shows the pin numbering and signal assignment. In addition, the socket contains switches that are operated by card insertion that provide signaling on the **CARDINx** and **MCI_WPROT** signals.

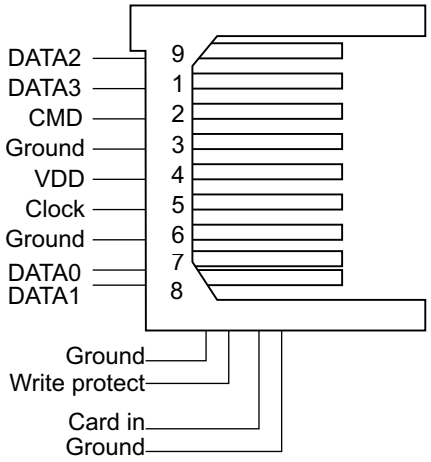


Figure A-3 MMC/SD card socket pin numbering

The MMC card uses seven pins, and the SD card uses all nine pins. Figure A-3 shows the additional pins with pin 9 next to pin 1 and pins 7 and 8 spaced more closely together than the other pins. Figure A-4 on page A-8 shows an MMC card, with the contacts face up.

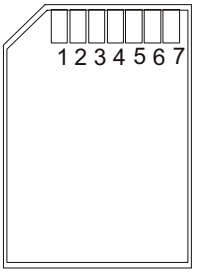


Figure A-4 MMC card

Table A-2 lists the signal assignments.

Table A-2 Multimedia Card interface signals

Pin	Signal	Function SD widebus mode	Function MCI
1	**MCIxDATA3**	Data	Chip select
2	**MCIxCMD**	Command/response	Data in
3	**GND**	Ground	Ground
4	**MCIVDDx**	Supply voltage	Supply voltage
5	**MCICLKx**	Clock	Clock
6	**GND**	Ground	Ground
7	**MCIxDATA0**	Data 0	Data out
8	**MCIxDATA1**	Data 1	NC
9	**MCIxDATA2**	Data 2	NC
10 (DET A)	**CARDINx**	Card insertion detect	Card insertion detect
11 (DET B)	**WPROTx**	Write protect status	Write protect status

A.4 Keyboard and mouse interface

Figure A-5 shows the pinout of the PBX-A9 KMI connectors J30A and J30B.

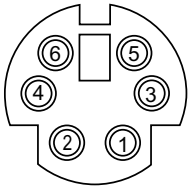


Figure A-5 KMI connector

Table A-3 shows signals on the KMI connectors.

Table A-3 Mouse and keyboard port signal descriptions

Pin	Keyboard (KMI0, J30B)		Mouse (KMI1, J30A)	
	Signal	Function	Signal	Function
1	**KDATA_FILT**	Keyboard data (filtered)	**MDATA_FILT**	Mouse Data (filtered)
2	NC	Not connected	NC	Not connected
3	**GNDKBD_FILT**	Ground (filtered)	**GNDMSE_FILT**	Ground (filtered)
4	**5VF2**	5V (filtered)	**5VF1**	5V (filtered)
5	**KCLK_FILT**	Keyboard clock (filtered)	**MCLK_FILT**	Mouse clock (filtered)
6	NC	Not connected	NC	Not connected

A.5 GPIO interface

Three eight-bit *General Purpose Input/Output* (GPIO) controllers are implemented in the PBX-A9 Southbridge. GPIO ports 0 and 1 are available for use as general purpose external I/O on header J33 on the baseboard. GPIO port 3 is used internally by the PBX-A9. See *Baseboard layout* on page 3-2 for the location of the GPIO header. Figure A-6 shows the signals.

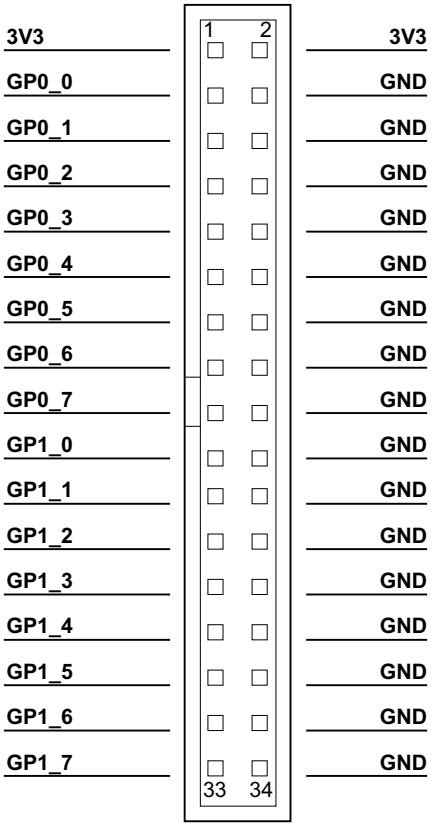


Figure A-6 GPIO connector

Note

Each data pin has an on-board 10KΩ pullup resistor to 3.3V.

A.6 UART interface

The PBX-A9 baseboard provides four serial transceivers on the rear panel of the enclosure.

Figure A-7 shows the pin numbering for the 9-pin D-type male connector used on the PBX-A9 and Table A-4 shows the signal assignment for the connectors.

The pinout that Figure A-7 shows is configured as a *Data Communications Equipment* (DCE) device.

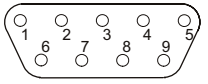


Figure A-7 Serial connector

Table A-4 Serial plug signal assignment

Pin	UART0 J24A (top)	UART1 J24B (bottom)	UART2 J25A (top)	UART3 J25B (bottom)
1	**SER0_DCD**	NC	NC	NC
2	**SER0_RX**	**SER1_RX**	**SER2_RX**	**SER3_RX**
3	**SER0_TX**	**SER1_TX**	**SER2_TX**	**SER3_TX**
4	**SER0_DTR**	**SER1_DTR**[a]	**SER2_DTR**[a]	**SER3_DTR**[a]
5	**SER0_GND**	**SER1_GND**	**SER2_GND**	**SER3_GND**
6	**SER0_DSR**	**SER1_DSR**[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.

A.7 Synchronous Serial Port interface

Figure A-8 shows the signals on the expansion PBX-A9 baseboard SSP interface connector J28. See *Baseboard layout* on page 3-2 for the location of the connector.

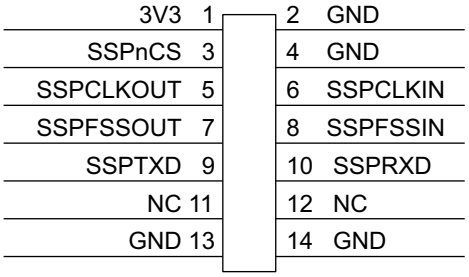


Figure A-8 SSP expansion interface

Table A-5 shows the signals associated with the SSP.

Table A-5 SSP signal assignment

Signal name	Description
SSPCLKOUT	Clock output from controller
SSPCLKIN	Clock input to controller
SSPFSSOUT	Frame sync output
SSPFSSIN	Frame sync input
SSPTXD	Data output
SSPRXD	Data input
SSPnCS	Chip select

A.8 Smart Card interface

The PBX-A9 baseboard provides a SIM socket on the front panel of the enclosure. See *Front panel layout* on page 3-4 for the location of the SIM socket.

Caution

The SIM card must be inserted into the SIM socket with the contacts facing down and the chamfered edge facing out.

Table A-6 shows the signals associated with the SCI.

Table A-6 Smartcard connector signal assignment

Pin	Signal	Description
1	**SC_VCC**	Card power (1.8V, 3.3V, or 5V)
2	**SC_RST**	Reset to card
3	**SC_CLK**	Clock to or from card
4	GND	Ground
5	SC_VCC	Programming voltage
6	**SC_IO**	Serial data to or from the card
7	–	Reserved for future use
8	–	Reserved for future use

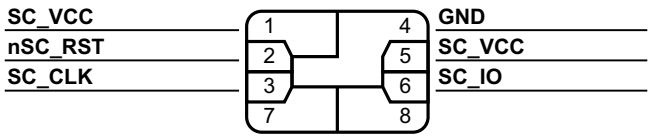


Figure A-9 Smartcard contacts assignment

Figure A-9 shows the signal assignment of a Smartcard. Pins 7 and 8 are not connected and are omitted on some cards.

Figure A-10 on page A-14 shows the pinout of the SCI Expansion connector. The connector (J2) is located on the front panel PCB (HBI 0176) below the CompactFlash socket (J8) and CompactFlash header (J7). This can be used to connect to an external smart card device.

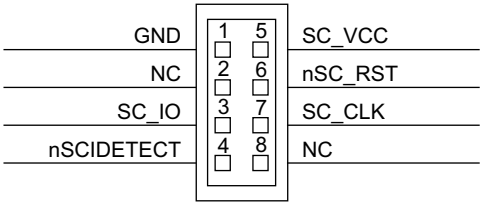


Figure A-10 SCI expansion

Table A-7 lists the signals on the SCI expansion connector.

Table A-7 Signals on SCI expansion connector

Pin	Signal	Description
1	**GND**	Ground
2	**NC**	Not connected
3	**SCI_IO**	SIM data
4	**nSCIDETECT**	Card detect for SIM
5	**SC_VCC**	SIM power
6	**nSC_RST**	Active LOW reset to SIM
7	**SC_CLK**	SIM clock
8	NC	Not connected

A.9 Ethernet interface

Figure A-11 shows the PBX-A9 baseboard RJ45 Ethernet connector. It is part of the combined RJ45 and Dual USB Type A connector positioned on the rear panel. See Figure A-12 on page A-16 for details of the combined Dual USB Type A connector.

LEDA (green) and LEDB (yellow) are connected to the LAN9118 controller. The function of the LEDs is determined by registers in the controller. Typical usage is to monitor transmit activity and packet detection.

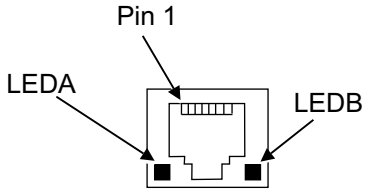


Figure A-11 Ethernet connector

Table A-8 shows the signals on the Ethernet cable.

Table A-8 Ethernet signals

Pin	Signal
1	**Transmit +**
2	**Transmit -**
3	**Receive +**
4	NC
5	NC
6	**Receive -**
7	NC
8	NC

A.10 USB interface

An OTG interface (USB1 in Figure A-12) is provided and connects through the USB Type A/B connector positioned on the front panel of the enclosure.

USB host interfaces (USB2 and USB3 in Figure A-12) are also provided and connect through the combined RJ45 and Dual USB Type A connector J14, positioned on the rear panel of the enclosure. See Figure A-11 on page A-15 for details of the combined RJ45 connector.

Note

For a full description of the USB signals, see to the datasheet for the NXP ISP1761 On-The-Go controller.

Figure A-12 shows the USB connectors and signals.

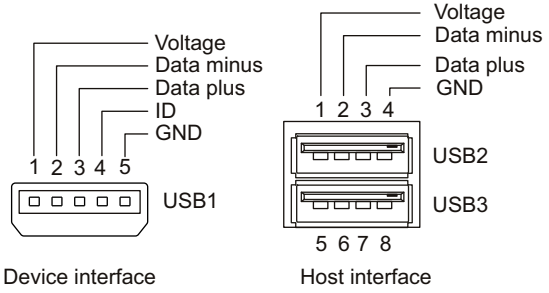


Figure A-12 USB interfaces

A.11 DVI display interface

The PBX-A9 baseboard DVI combined connector (J16) that Figure A-13 shows is positioned on the rear panel of the enclosure, see *Rear panel layout* on page 3-5. Table A-9 lists the connector signals. The digital CLCD data from the Northbridge is passed to a T.M.D.S. transmitter to provide the DVI digital data and to a triple video DAC to provide the analogue RGB signals. The DDC2B interface is provided by a custom *Two-wire Interface* (SBCon) implemented in the Southbridge.

Note

The mechanical interconnect includes 29 signal contacts, that are divided into two sections. The first section is organized as three rows of eight contacts. The second section contains five signals that are designed specifically for analog video signals. Horizontal Sync, Vertical Sync, R, G, and B are all required for analog implementations.

A fused (1A anti-surge) +5V supply (pin 14) is provided by the PBX-A9.

Figure A-13 shows the combined DVI connector.

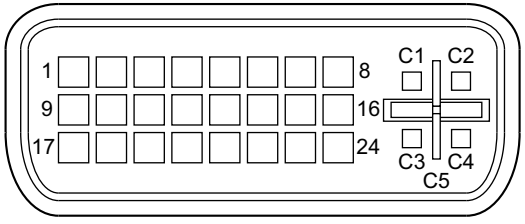


Figure A-13 DVI connector

Table A-9 lists the DVI connector signals.

Table A-9 DVI connector signals

Pin	Signal	Pin	Signal	Pin	Signal
1	T.M.D.S. Data2-	9	T.M.D.S. Data1-	17	T.M.D.S. Data0-
2	T.M.D.S. Data2+	10	T.M.D.S. Data1+	18	T.M.D.S. Data0+
3	T.M.D.S. Data2/4 Shield	11	T.M.D.S. Data1/3 Shield	19	T.M.D.S. Data0/5 Shield
4	T.M.D.S. Data4-	12	T.M.D.S. Data3-	20	T.M.D.S. Data5-
5	T.M.D.S. Data4+	13	T.M.D.S. Data3+	21	T.M.D.S. Data5+

Table A-9 DVI connector signals (continued)

Pin	Signal	Pin	Signal	Pin	Signal
6	DDC Clock	14	+5V Power	22	T.M.D.S. Clock Shield
7	DDC Data	15	Ground (for +5V)	23	T.M.D.S. Clock+
8	Analog Vertical sync	16	Hot Plug Detect	24	T.M.D.S. Clock-
C1	Analog Red	C2	Analog Green	C3	Analog Blue
C4	Analog Horizontal Sync	C5	Analog Ground (analog R, G, and B return)		

Note

The PBX-A9 implements a single link, T.M.D.S. Data 3, 4, and 5 connections are not used.

A.12 RealView Logic Tile header connectors

These headers enable the connection of a RealView Logic Tile to the PBX-A9 baseboard. Figure A-14 shows the pin numbers and power-blade usage of the HDRX, HDRY, and HDRZ headers.

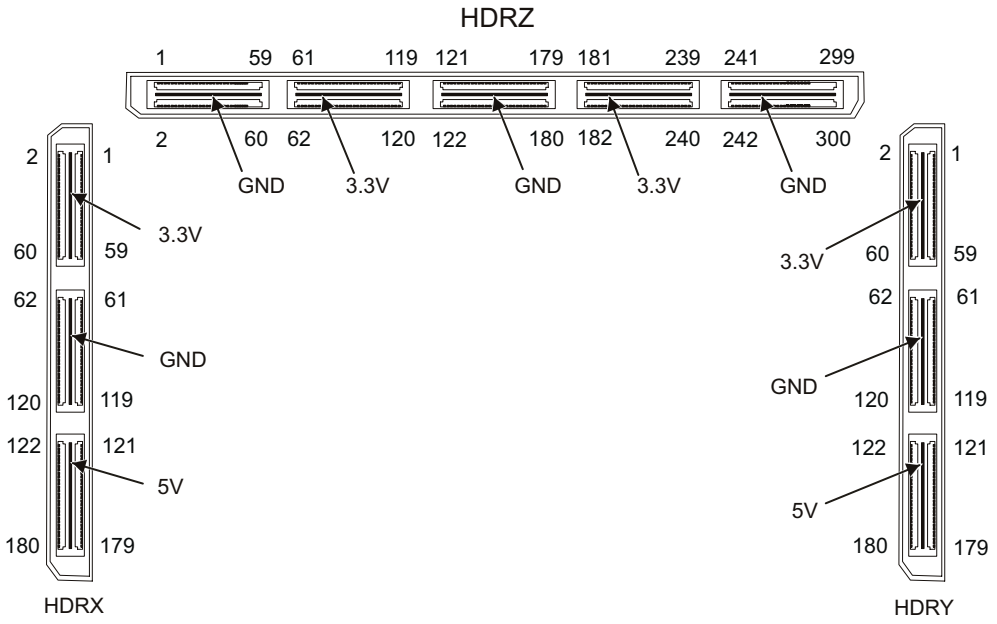


Figure A-14 HDRX, HDRY HDRZ pin numbering

Warning

The I/O voltage on a RealView Logic Tile (**VCCO1** and **VCC02**) can be changed by removing resistors on the tile and supplying the I/O voltage from either the tile above or the tile below in a tile stack. However, all signals from the PBX-A9 to a RealView Logic Tile use 3.3V I/O levels and all signals from a RealView Logic Tile to the PBX-A9 must use 3.3V I/O levels.

The 5V supply on the headers is to power voltage converters that might be present on the Logic Tile.

Tables *HDRX signals* on page A-20, *HDRY signals* on page A-26, and *HDRZ signals* on page A-33 list the signals on each header pin.

Note

The designation used for a multiplexed signal is X / Y, where signal X is present when **CLKOUTDIV** is HIGH and signal Y is present when **CLKOUTDIV** is LOW. See *Application Note AN151* for details of the AXI multiplexing scheme.

A.12.1 HDRX signals

Table A-10 describes the signals on the HDRX header pins. For a description of the signals, see *AMBA 3 AXI Protocol* (ARM IHI 0022).

Table A-10 HDRX signals

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
ARADDR12 /ARADDR28	**X89**	2	1	**X90**	**ARADDR13 /ARADDR29**
ARADDR11 /ARADDR27	**X88**	4	3	**X91**	**ARADDR14 /ARADDR30**
ARADDR10 /ARADDR26	**X87**	6	5	**X92**	**ARADDR15 /ARADDR31**
ARADDR9 /ARADDR25	**X86**	8	7	**X93**	**ARID0 /ARID2**
ARADDR8 /ARADDR24	**X85**	10	9	**X94**	**ARID1 /ARID3**
ARADDR7 /ARADDR23	**X84**	12	11	**X95**	**ARLEN0 /ARLEN2**
ARADDR6 /ARADDR22	**X83**	14	13	**X96**	**ARLEN1 /ARLEN3**
ARADDR5 /ARADDR21	**X82**	16	15	**X97**	**ARSIZE0 /ARSIZE1**
ARADDR4 /ARADDR20	**X81**	18	17	**X98**	**ARID4 /ARPROT2**
ARADDR3 /ARADDR19	**X80**	20	19	**X99**	**ARPROT0 /ARPROT1**
ARADDR2 /ARADDR18	**X79**	22	21	**X100**	**ARBURST0 /ARBURST1**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
ARADDR1 /ARADDR17	**X78**	24	23	**X101**	**ARLOCK0 /ARLOCK1**
ARADDR0 /ARADDR16	**X77**	26	25	**X102**	**ARCACHE0 /ARCACHE2**
BREADY	**X76**	28	27	**X103**	**ARCACHE1 /ARCACHE3**
BVALID	**X75**	30	29	**X104**	**ARVALID** / b0[a]
BRESP0 /BRESP1	**X74**	32	31	**X105**	**ARREADY**
BID4 / (not connected)	**X73**	34	33	**X106**	**RDATA0 /RDATA32**
BID1 / **BID3**	**X72**	36	35	**X107**	**RDATA1 /RDATA33**
BID0 / **BID2**	**X71**	38	37	**X108**	**RDATA2 /RDATA34**
AWREADY	**X70**	40	39	**X109**	**RDATA3 /RDATA35**
AWVALID / b0[b]	**X69**	42	41	**X110**	**RDATA4 /RDATA36**
AWCACHE1 /AWCACHE3	**X68**	44	43	**X111**	**RDATA5 /RDATA37**
AWCACHE0 /AWCACHE2	**X67**	46	45	**X112**	**RDATA6 /RDATA38**
AWLOCK0 /AWLOCK1	**X66**	48	47	**X113**	**RDATA7 /RDATA39**
AWBURST0 /AWBURST1	**X65**	50	49	**X114**	**RDATA8 /RDATA40**
AWPROT0 /AWPROT1	**X64**	52	51	**X115**	**RDATA9 /RDATA41**
ARM_nRESET	**X63**	54	53	**X116**	**RDATA10 /RDATA42**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
AWID4 /AWPROT2	**X62**	56	55	**X117**	**RDATA11 /RDATA43**
AWSIZE0 /AWSIZE1	**X61**	58	57	**X118**	**RDATA12 /RDATA44**
AWLEN1 /AWLEN3	**X60**	60	59	**X119**	**RDATA13 /RDATA45**
AWLEN0 /AWLEN2	**X59**	62	61	**X120**	**RDATA14 /RDATA46**
AWID1 /AWID3	**X58**	64	63	**X121**	**RDATA15 /RDATA47**
AWID0 /AWID2	**X57**	66	65	**X122**	**RDATA16 /RDATA48**
AWADDR15 /AWADDR31	**X56**	68	67	**X123**	**RDATA17 /RDATA49**
AWADDR14 /AWADDR30	**X55**	70	69	**X124**	**RDATA18 /RDATA50**
AWADDR13 /AWADDR29	**X54**	72	71	**X125**	**RDATA19 /RDATA51**
AWADDR12 /AWADDR28	**X53**	74	73	**X126**	**RDATA20 /RDATA52**
AWADDR11 /AWADDR27	**X52**	76	75	**X127**	**RDATA21 /RDATA53**
AWADDR10 /AWADDR26	**X51**	78	77	**X128**	**RDATA22 /RDATA54**
AWADDR9 /AWADDR25	**X50**	80	79	**X129**	**RDATA23 /RDATA55**
AWADDR8 /AWADDR24	**X49**	82	81	**X130**	**RDATA24 /RDATA56**
AWADDR7 /AWADDR23	**X48**	84	83	**X131**	**RDATA25 /RDATA57**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
AWADDR6 /AWADDR22	**X47**	86	85	**X132**	**RDATA26 /RDATA58**
AWADDR5 /AWADDR21	**X46**	88	87	**X133**	**RDATA27 /RDATA59**
AWADDR4 /AWADDR20	**X45**	90	89	**X134**	**RDATA28 /RDATA60**
AWADDR3 /AWADDR19	**X44**	92	91	**X135**	**RDATA29 /RDATA61**
AWADDR2 /AWADDR18	**X43**	94	93	**X136**	**RDATA30 /RDATA62**
AWADDR1 /AWADDR17	**X42**	96	95	**X137**	**RDATA31 /RDATA63**
AWADDR0 /AWADDR16	**X41**	98	97	**X138**	**RID0 /RID2**
WREADY	**X40**	100	99	**X139**	**RID1 /RID3**
WVALID / b0[c]	**X39**	102	101	**X140**	**RRESP0 /RRESP1**
WLAST / WID4	**X38**	104	103	**X141**	**RLAST /RID4**
WSTRB3 /WSTRB7	**X37**	106	105	**X142**	**RVALID** / (not connected)
WSTRB2 /WSTRB6	**X36**	108	107	**X143**	**RREADY**
WSTRB1 /WSTRB5	**X35**	110	109	**X144**	**X144**
WSTRB0 /WSTRB4	**X34**	112	111	**X145**	**X145**
WID1 /WID3	**X33**	114	113	**X146**	**X146**
WID0 /WID2	**X32**	116	115	**X147**	**X147**
WDATA31 /WDATA63	**X31**	118	117	**X148**	**X148**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
WDATA30 /WDATA62	**X30**	120	119	**X149**	**X149**
WDATA29 /WDATA61	**X29**	122	121	**X150**	**X150**
WDATA28 /WDATA60	**X28**	124	123	**X151**	**X151**
WDATA27 /WDATA59	**X27**	126	125	**X152**	**X152**
WDATA26 /WDATA58	**X26**	128	127	**X153**	**X153**
WDATA25 /WDATA57	**X25**	130	129	**X154**	**X154**
WDATA24 /WDATA56	**X24**	132	131	**X155**	**X155**
WDATA23 /WDATA55	**X23**	134	133	**X156**	**X156**
WDATA22 /WDATA54	**X22**	136	135	**X157**	**X157**
WDATA21 /WDATA53	**X21**	138	137	**X158**	**X158**
WDATA20 /WDATA52	**X20**	140	139	**X159**	**X159**
WDATA19 /WDATA51	**X19**	142	141	**X160**	**X160**
WDATA18 /WDATA50	**X18**	144	143	**X161**	**X161**
WDATA17 /WDATA49	**X17**	146	145	**X162**	**X162**
WDATA16 /WDATA48	**X16**	148	147	**X163**	**X163**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
WDATA15 /WDATA47	**X15**	150	149	**X164**	**X164**
WDATA14 /WDATA46	**X14**	152	151	**X165**	**X165**
WDATA13 /WDATA45	**X13**	154	153	**X166**	**X166**
WDATA12 /WDATA44	**X12**	156	155	**X167**	**X167**
WDATA11 /WDATA43	**X11**	158	157	**X168**	**X168**
WDATA10 /WDATA42	**X10**	160	159	**X169**	**X169**
WDATA9 /WDATA41	**X9**	162	161	**X170**	**X170**
WDATA8 /WDATA40	**X8**	164	163	**X171**	**X171**
WDATA7 /WDATA39	**X7**	166	165	**X172**	**X172**
WDATA6 /WDATA38	**X6**	168	167	**X173**	**X173**
WDATA5 /WDATA37	**X5**	170	169	**X174**	**X174**
WDATA4 /WDATA36	**X4**	172	171	**X175**	**X175**
WDATA3 /WDATA35	**X3**	174	173	**X176**	**X176**

Table A-10 HDRX signals (continued)

Baseboard signals	X Bus	Even pins	Odd pins	X Bus	Baseboard signals
WDATA2 /WDATA34	**X2**	176	175	**X177**	**X177**
WDATA1 /WDATA33	**X1**	178	177	**X178**	**X178**
WDATA0 /WDATA32	**X0**	180	179	**X179**	**X179**

a. These are AXI **ARVALID/ARID5** signals. b0 indicates that for the **ARID5** phase of the controlling clock signal, **ARID5** is always set to logic level zero
b. These are AXI **AWVALID/AWID5** signals. b0 indicates that for the **AWID5** phase of the controlling clock signal, **AWID5** is always set to logic level zero.
c. These are AXI **WVALID/WID5** signals. b0 indicates that for the **WID5** phase of the controlling clock signal, **WID5** is always set to logic level zero.

A.12.2 HDRY signals

Table A-11 describes the signals on the HDRY header pins. For a description of the signals, see *AMBA 3 AXI Protocol* (ARM IHI 0022).

Table A-11 HDRY signals

Baseboard signals	Y Bus	Even pins	Odd pins	Y Bus	Baseboard signals
ARADDR12 /ARADDR28	**Y89**	2	1	**Y90**	**ARADDR13 /ARADDR29**
ARADDR11 /ARADDR27	**Y88**	4	3	**Y91**	**ARADDR14 /ARADDR30**
ARADDR10 /ARADDR26	**Y87**	6	5	**Y92**	**ARADDR15 /ARADDR31**
ARADDR9 /ARADDR25	**Y86**	8	7	**Y93**	**ARID0 /ARID2**
ARADDR8 /ARADDR24	**Y85**	10	9	**Y94**	**ARID1 /ARID3**
ARADDR7 /ARADDR23	**Y84**	12	11	**Y95**	**ARLEN0 /ARLEN2**

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Y Bus	Baseboard signals
ARADDR6 /ARADDR22	**Y83**	14	13	**Y96**	**ARLEN1 /ARLEN3**
ARADDR5 /ARADDR21	**Y82**	16	15	**Y97**	**ARSIZE0 /ARSIZE1**
ARADDR4 /ARADDR20	**Y81**	18	17	**Y98**	**ARID4 /ARPROT2**
ARADDR3 /ARADDR19	**Y80**	20	19	**Y99**	**ARPROT0 /ARPROT1**
ARADDR2 /ARADDR18	**Y79**	22	21	**Y100**	**ARBURST0 /ARBURST1**
ARADDR1 /ARADDR17	**Y78**	24	23	**Y101**	**ARLOCK0 /ARLOCK1**
ARADDR0 /ARADDR16	**Y77**	26	25	**Y102**	**ARCACHE0 /ARCACHE2**
BREADY	**Y76**	28	27	**Y103**	**ARCACHE1 /ARCACHE3**
BVALID	**Y75**	30	29	**Y104**	**ARVALID** / b0[a]
BRESP0 /BRESP1	**Y74**	32	31	**Y105**	**ARREADY**
BID4 / (not connected)	**Y73**	34	33	**Y106**	**RDATA0 /RDATA32**
BID1 / **BID3**	**Y72**	36	35	**Y107**	**RDATA1 /RDATA33**
BID0 / **BID2**	**Y71**	38	37	**Y108**	**RDATA2 /RDATA34**
AWREADY	**Y70**	40	39	**Y109**	**RDATA3 /RDATA35**
AWVALID / b0[b]	**Y69**	42	41	**Y110**	**RDATA4 /RDATA36**
AWCACHE1 /AWCACHE3	**Y68**	44	43	**Y111**	**RDATA5 /RDATA37**

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Y Bus	Baseboard signals
AWCACHE0 /AWCACHE2	**Y67**	46	45	**Y112**	**RDATA6 /RDATA38**
AWLOCK0 /AWLOCK1	**Y66**	48	47	**Y113**	**RDATA7 /RDATA39**
AWBURST0 /AWBURST1	**Y65**	50	49	**Y114**	**RDATA8 /RDATA40**
AWPROT0 /AWPROT1	**Y64**	52	51	**Y115**	**RDATA9 /RDATA41**
ARM_nRESET	**Y63**	54	53	**Y116**	**RDATA10 /RDATA42**
AWID4 /AWPROT2	**Y62**	56	55	**Y117**	**RDATA11 /RDATA43**
AWSIZE0 /AWSIZE1	**Y61**	58	57	**Y118**	**RDATA12 /RDATA44**
AWLEN1 /AWLEN3	**Y60**	60	59	**Y119**	**RDATA13 /RDATA45**
AWLEN0 /AWLEN2	**Y59**	62	61	**Y120**	**RDATA14 /RDATA46**
AWID1 /AWID3	**Y58**	64	63	**Y121**	**RDATA15 /RDATA47**
AWID0 /AWID2	**Y57**	66	65	**Y122**	**RDATA16 /RDATA48**
AWADDR15 /AWADDR31	**Y56**	68	67	**Y123**	**RDATA17 /RDATA49**
AWADDR14 /AWADDR30	**Y55**	70	69	**Y124**	**RDATA18 /RDATA50**
AWADDR13 /AWADDR29	**Y54**	72	71	**Y125**	**RDATA19 /RDATA51**
AWADDR12 /AWADDR28	**Y53**	74	73	**Y126**	**RDATA20 /RDATA52**

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Y Bus	Baseboard signals
AWADDR11 /AWADDR27	**Y52**	76	75	**Y127**	**RDATA21 /RDATA53**
AWADDR10 /AWADDR26	**Y51**	78	77	**Y128**	**RDATA22 /RDATA54**
AWADDR9 /AWADDR25	**Y50**	80	79	**Y129**	**RDATA23 /RDATA55**
AWADDR8 /AWADDR24	**Y49**	82	81	**Y130**	**RDATA24 /RDATA56**
AWADDR7 /AWADDR23	**Y48**	84	83	**Y131**	**RDATA25 /RDATA57**
AWADDR6 /AWADDR22	**Y47**	86	85	**Y132**	**RDATA26 /RDATA58**
AWADDR5 /AWADDR21	**Y46**	88	87	**Y133**	**RDATA27 /RDATA59**
AWADDR4 /AWADDR20	**Y45**	90	89	**Y134**	**RDATA28 /RDATA60**
AWADDR3 /AWADDR19	**Y44**	92	91	**Y135**	**RDATA29 /RDATA61**
AWADDR2 /AWADDR18	**Y43**	94	93	**Y136**	**RDATA30 /RDATA62**
AWADDR1 /AWADDR17	**Y42**	96	95	**Y137**	**RDATA31 /RDATA63**
AWADDR0 /AWADDR16	**Y41**	98	97	**Y138**	**RID0 /RID2**
WREADY	**Y40**	100	99	**Y139**	**RID1 /RID3**
WVALID / b0[c]	**Y39**	102	101	**Y140**	**RRESP0 /RRESP1**
WLAST / **WID4**	**Y38**	104	103	**Y141**	**RLAST /RID4**
WSTRB3 /WSTRB7	**Y37**	106	105	**Y142**	**RVALID** / (not connected)

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Y Bus	Baseboard signals
WSTRB2 /WSTRB6	**Y36**	108	107	**Y143**	**RREADY**
WSTRB1 /WSTRB5	**Y35**	110	109	**Y144**	**Y144**
WSTRB0 /WSTRB4	**Y34**	112	111	**Y145**	**Y145**
WID1 /WID3	**Y33**	114	113	**Y146**	**Y146**
WID0 /WID2	**Y32**	116	115	**Y147**	**Y147**
WDATA31 /WDATA63	**Y31**	118	117	**Y148**	**Y148**
WDATA30 /WDATA62	**Y30**	120	119	**Y149**	**Y149**
WDATA29 /WDATA61	**Y29**	122	121	**Y150**	**Y150**
WDATA28 /WDATA60	**Y28**	124	123	**Y151**	**Y151**
WDATA27 /WDATA59	**Y27**	126	125	**Y152**	**Y152**
WDATA26 /WDATA58	**Y26**	128	127	**Y153**	**Y153**
WDATA25 /WDATA57	**Y25**	130	129	**Y154**	**Y154**
WDATA24 /WDATA56	**Y24**	132	131	**Y155**	**Y155**
WDATA23 /WDATA55	**Y23**	134	133	**Y156**	**Y156**
WDATA22 /WDATA54	**Y22**	136	135	**Y157**	**Y157**
WDATA21 /WDATA53	**Y21**	138	137	**Y158**	**Y158**

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Y Bus	Baseboard signals
WDATA20 /WDATA52	**Y20**	140	139	**Y159**	**Y159**
WDATA19 /WDATA51	**Y19**	142	141	**Y160**	**Y160**
WDATA18 /WDATA50	**Y18**	144	143	**Y161**	**Y161**
WDATA17 /WDATA49	**Y17**	146	145	**Y162**	**Y162**
WDATA16 /WDATA48	**Y16**	148	147	**Y163**	**Y163**
WDATA15 /WDATA47	**Y15**	150	149	**Y164**	**Y164**
WDATA14 /WDATA46	**Y14**	152	151	**Y165**	**Y165**
WDATA13 /WDATA45	**Y13**	154	153	**Y166**	**Y166**
WDATA12 /WDATA44	**Y12**	156	155	**Y167**	**Y167**
WDATA11 /WDATA43	**Y11**	158	157	**Y168**	**Y168**
WDATA10 /WDATA42	**Y10**	160	159	**Y169**	**Y169**
WDATA9 /WDATA41	**Y9**	162	161	**Y170**	**Y170**
WDATA8 /WDATA40	**Y8**	164	163	**Y171**	**Y171**
WDATA7 /WDATA39	**Y7**	166	165	**Y172**	**Y172**
WDATA6 /WDATA38	**Y6**	168	167	**Y173**	**Y173**

Table A-11 HDRY signals (continued)

Baseboard signals	Y Bus	Even pins	Odd pins	Y Bus	Baseboard signals
WDATA5 /WDATA37	**Y5**	170	169	**Y174**	**Y174**
WDATA4 /WDATA36	**Y4**	172	171	**Y175**	**Y175**
WDATA3 /WDATA35	**Y3**	174	173	**Y176**	**Y176**
WDATA2 /WDATA34	**Y2**	176	175	**Y177**	**Y177**
WDATA1 /WDATA33	**Y1**	178	177	**Y178**	**Y178**
WDATA0 /WDATA32	**Y0**	180	179	**Y179**	**Y179**

a. These are AXI **ARVALID/ARID5** signals. b0 indicates that for the **ARID5** phase of the controlling clock signal, **ARID5** it is always set to logic level zero.
b. These are AXI **AWVALID/AWID5** signals. b0 indicates that for the **AWID5** phase of the controlling clock signal, **AWID5** it is always set to logic level zero.
c. These are AXI **WVALID/WID5** signals. b0 indicates that for the **WID5** phase of the controlling clock signal, **WID5** it is always set to logic level zero.

A.12.3 HDRZ signals

Table A-12 on page A-33 lists the signals on the HDRZ header pins.

Note

The following pins are reserved or have special functionality on the PBX-A9:

Z[127:40] Not connected. Reserved for use by Logic Tiles
Z[218:200] Tile interrupt and DMA signals
Z[39:32] Alternate source for UARTs
Z[31:0] Alternate source for DVI output.

See the user guide for the fitted Logic Tile for the Logic Tile specific HDRZ upper (U) and HDRZ lower (L) signal listing.

Table A-12 HDRZ signals

Baseboard signals	Even pins	Odd pins	Baseboard signals
Z255	2	1	**Z128**
Z254	4	3	**Z129**
Z253	6	5	**Z130**
Z252	8	7	**Z131**
Z251	10	9	**Z132**
Z250	12	11	**Z133**
Z249	14	13	**Z134**
Z248	16	15	**Z135**
Z247	18	17	**Z136**
Z246	20	19	**Z137**
Z245	22	21	**Z138**
Z244	24	23	**Z139**
Z243	26	25	**Z140**
Z242	28	27	**Z141**
Z241	30	29	**Z142**
Z240	32	31	**Z143**
Z249	34	33	**Z144**
Z248	36	35	**Z145**
Z237	38	37	**Z146**
Z236	40	39	**Z147**
Z235	42	41	**Z148**
Z234	44	43	**Z149**
MBTYPE	46	45	**Z150**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
MBTYPE	48	47	**Z151**
Z231	50	49	**Z152**
Z230	52	51	**Z153**
Z229	54	53	**Z154**
Z228	56	55	**Z155**
Z227	58	57	**Z156**
Z226	60	59	**Z157**
Z225	62	61	**Z158**
Z224	64	63	**Z159**
Z223	66	65	**Z160**
Z222	68	67	**Z161**
Z221	70	69	**Z162**
Z220	72	71	**Z163**
Z219	74	73	**Z164**
DMACCLR[1]	76	75	**Z165**
Z217	78	77	**Z166**
Z216	80	79	**Z167**
DMACBREQ[1]	82	81	**Z168**
DMACSREQ[1]	84	83	**Z169**
Z213	86	85	**Z170**
DMACCLR[0]	88	87	**Z171**
Z211	90	89	**Z172**
Z210	92	91	**Z173**
DMACBREQ[0]	94	93	**Z174**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
DMACSREQ[0]	96	95	**Z175**
INT[7]	98	97	**Z176**
INT[6]	100	99	**Z177**
INT[5]	102	101	**Z178**
INT[4]	104	103	**Z179**
INT[3]	106	105	**Z180**
INT[2]	108	107	**Z181**
INT[1]	110	109	**Z182**
INT[0]	112	111	**Z183**
Z199	114	113	**Z184**
Z198	116	115	**Z185**
Z197	118	117	**Z186**
Z196	120	119	**Z187**
Z195	122	121	**Z188**
Z194	124	123	**Z189**
Z193	126	125	**Z190**
Z192	128	127	**Z191**
CLK_POS_DN_IN	130	129	**D_nSRST**
CLK_NEG_DN_IN	132	131	**D_nTRST**
CLK_POS_UP_OUT	134	133	**D_TDO_IN**
CLK_NEG_UP	136	135	**D_TDI**
GND (**CLK_UP_THRU**)	138	137	**D_TCK_OUT**
CLK_OUT_PLUS1	140	139	**D_TMS_OUT**
CLK_OUT_PLUS2	142	141	**D_RTCK**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
CLK_IN_PLUS2	144	143	**C_nSRST**
CLK_IN_PLUS1	146	145	**C_nTRST**
Z182 (**CLK_DN_THRU**)	148	147	**C_TDO_IN**
CLK_GLOBAL	150	149	**C_TDI**
FPGA_IMAGE	152	151	**C_TCK_OUT**
nSYSPOR	154	153	**C_TMS_OUT**
nSYSRST	156	155	**nTILE_DET**
nRTCKEN	158	157	**nCFGEN**
SPARE12 (reserved)	160	159	**GLOBAL_DONE**
SPARE10 (reserved)	162	161	**SPARE11** (reserved)
SPARE8 (reserved)	164	163	**SPARE9** (reserved)
SPARE6 (reserved)	166	165	**SPARE7** (reserved)
SPARE4 (reserved)	168	167	**SPARE5** (reserved)
SPARE2 (reserved)	170	169	**SPARE3** (reserved)
SPARE0 (reserved)	172	171	**SPARE1** (reserved)
Z64	174	173	**Z63**
Z65	176	175	**Z62**
Z66	178	177	**Z61**
Z67	180	179	**Z60**
Z68	182	181	**Z59**
Z79	184	183	**Z58**
Z70	186	185	**Z57**
Z71	188	187	**Z56**
Z72	190	189	**Z55**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
Z73	192	191	**Z54**
Z74	194	193	**Z53**
Z75	196	195	**Z52**
Z76	198	197	**Z51**
Z77	200	199	**Z50**
Z78	202	201	**Z49**
Z79	204	203	**Z48**
Z80	206	205	**Z47**
Z81	208	207	**Z46**
Z82	210	209	**Z45**
Z83	212	211	**Z44**
Z84	214	213	**Z43**
Z85	216	215	**Z42**
Z86	218	217	**Z41**
Z87	220	219	**Z40**
Z88	222	221	**UART3RXD**
Z89	224	223	**nUART3CTS**
Z90	226	225	**UART3TXD**
Z91	228	227	**nUART3RTS**
Z92	230	229	**UART2RXD**
Z93	232	231	**nUART2CTS**
Z94	234	233	**UART2TXD**
Z95	236	235	**nUART2RTS**
Z96	238	237	**CLCP** analog

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
Z97	240	239	**Z30**
Z98	242	241	**CLPOWER**
Z99	244	243	**CLLP**
Z100	246	245	**Z27**
Z101	248	247	**CLFP**
Z102	250	249	**CLCP** digital
Z103	252	251	**CLAC**
Z104	254	253	**CLD23**
Z105	256	255	**CLD22**
Z106	258	257	**CLD21**
Z107	260	259	**CLD20**
Z108	262	261	**CLD19**
Z109	264	263	**CLD18**
Z110	266	265	**CLD17**
Z111	268	267	**CLD16**
Z112	270	269	**CLD15**
Z113	272	271	**CLD14**
Z114	274	273	**CLD13**
Z115	276	275	**CLD12**
Z116	278	277	**CLD11**
Z117	280	279	**CLD10**
Z118	282	281	**CLD9**
Z119	284	283	**CLD8**
Z120	286	285	**CLD7**

Table A-12 HDRZ signals (continued)

Baseboard signals	Even pins	Odd pins	Baseboard signals
Z121	288	287	**CLD6**
Z122	290	289	**CLD5**
Z123	292	291	**CLD4**
Z124	294	293	**CLD3**
Z125	296	295	**CLD2**
Z126	298	297	**CLD1**
Z127	300	299	**CLD0**

A.13 Test and debug connections

The PBX-A9 baseboard provides connectors to aid diagnostics.

This section contains the following subsections:

- *JTAG*
- *USB config port* on page A-41
- *Integrated Logic Analyzer (ILA)* on page A-41
- *Trace connector* on page A-42.

A.13.1 JTAG

Figure A-15 shows the pinout of the JTAG connectors:

- the JTAG ICE connector located on the rear panel of the ATX enclosure
- the duplicated JTAG ICE connector (J8) on the daughterboard.
- the JTAG config connector (J10) on the baseboard.

All JTAG active HIGH input signals have pull-up resistors.

See *Rear panel layout* on page 3-5 to locate the connector and see *Test, configuration, and debug interfaces* on page 3-47 for a description of how the JTAG config and debug interfaces are implemented on the PBX-A9.

Note

The term JTAG equipment refers to any hardware that can drive the JTAG signals to devices in the scan chain. Typically, RealView ICE is used to debug and configure the PBX-A9. JTAG hardware from other suppliers cannot be used to configure the design.

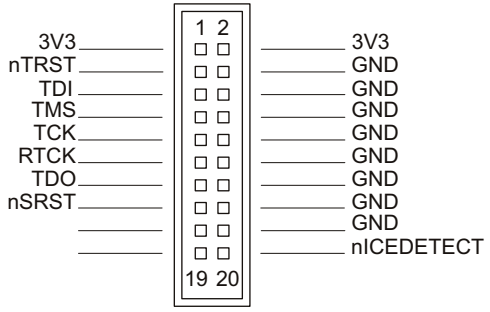


Figure A-15 JTAG connector

Note

The JTAG connector on the baseboard is not connected to the debug scan chain. You must use the connector rear panel (on the daughterboard) for debugging. The baseboard JTAG connector has the same pin arrangement, but it is only used for configuration.

A.13.2 USB config port

Figure A-16 shows the signals on the USB config connector (J12).**USBDP** and **USBDM** are the positive and negative USB data signals.

See *Front panel layout* on page 3-4 to locate the connector and see *Test, configuration, and debug interfaces* on page 3-47 for a description of how the USB debug interface is implemented on the PBX-A9. The USB config port is disabled if the JTAG configuration connector on the baseboard is used.

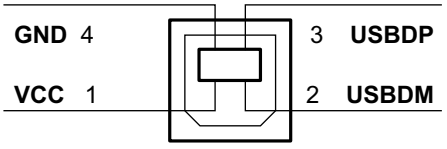


Figure A-16 USB debug connector

A.13.3 Integrated Logic Analyzer (ILA)

Figure A-17 shows the signals on the ILA connector (J4). You can use an ILA to debug designs in a Logic Tile fitted to the tile site.

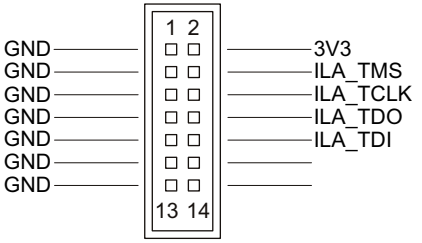


Figure A-17 Integrated Logic Analyzer (ILA) connector

Note

Pins 4, 6, and 10 on the ILA connector have pullup resistors to 3V3.

See *Baseboard layout* on page 3-2 to locate the connector, and for more information, see the documentation supplied with your analyzer. The ChipScope product is described on the Xilinx web site at `www.xilinx.com`.

A.13.4 Trace connector

Trace connectors are provided on the PBX-A9 baseboard and daughterboard. The baseboard trace connector provides access to the *Trace Port Interface Unit* (TPIU) implemented in a Logic Tile fitted to the tile site. The trace connectors on the daughterboard provide access to the TPIU implemented in the Cortex-A9 structured ASIC.

Figure A-18 shows the Mictor connector (part number AMP 2-767004-2).

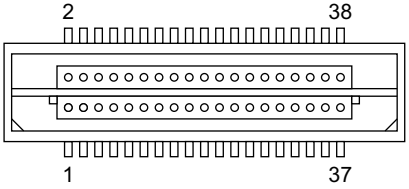


Figure A-18 Trace Connector

Table A-13 lists the pinout for the Logic Tile and Cortex-A9 structured ASIC Trace Port A (TRACEA) connector (J5 and J6).

Table A-13 Trace Port A (TRACEA) connectors

Trace Port signal	Pin	Pin	Trace Port signal
Not connected	1	2	Not connected
Not connected	3	4	Not connected
GND	5	6	**TRACECLKA**
EDBGRQ	7	8	**DBGACK**
nSRST	9	10	**GND** (pull-down)
TDO	11	12	**VTREF** (3.3V)
RTCK	13	14	**VSUPPLY** (3.3V)
TCK	15	16	**TRACEDATA7**
TMS	17	18	**TRACEDATA6**

Table A-13 Trace Port A (TRACEA) connectors (continued)

Trace Port signal	Pin	Pin	Trace Port signal
TDI	19	20	**TRACEDATA5**
nTRST	21	22	**TRACEDATA4**
TRACEDATA15	23	24	**TRACEDATA3**
TRACEDATA14	25	26	**TRACEDATA2**
TRACEDATA13	27	28	**TRACEDATA1**
TRACEDATA12	29	30	**GND**
TRACEDATA11	31	32	**GND**
TRACEDATA10	33	34	**Logic 1** (3.3V)
TRACEDATA9	35	36	**TRACECTL**
TRACEDATA8	37	38	**TRACEDATA0**

Caution

Do not use the **Vsupply** pin to supply power to any connected device, because it might cause damage.

Table A-14 lists the pinout for the Cortex-A9 structured ASIC Trace Port B (TRACEB) connector (J7).

Table A-14 Trace Port B (TRACEB) connector

Trace Port signal	Pin	Pin	Trace Port signal
Not connected	1	2	Not connected
Not connected	3	4	Not connected
GND	5	6	**TRACECLK**
Not connected	7	8	Not connected
Not connected	9	10	Not connected
Not connected	11	12	**VTREF** (3.3V)
Not connected	13	14	Not connected

Table A-14 Trace Port B (TRACEB) connector (continued)

Trace Port signal	Pin	Pin	Trace Port signal
Not connected	15	16	**TRACEDATA23**
Not connected	17	18	**TRACEDATA22**
Not connected	19	20	**TRACEDATA21**
Not connected	21	22	**TRACEDATA20**
TRACEDATA31	23	24	**TRACEDATA19**
TRACEDATA30	25	26	**TRACEDATA18**
TRACEDATA29	27	28	**TRACEDATA17**
TRACEDATA28	29	30	**GND**
TRACEDATA27	31	32	**GND**
TRACEDATA26	33	34	**VDDIO** (pullup)
TRACEDATA25	35	36	**GND**
TRACEDATA24	37	38	**TRACEDATA16**

Appendix B
Specifications

This appendix contains the specification for the PBX-A9 baseboard. It contains the following sections:

B.1 Electrical Specification

This section provides details of the voltage and current characteristics for the PBX-A9 baseboard.

B.1.1 Bus interface characteristics

Table B-1 lists the PBX-A9 baseboard electrical characteristics for normal operation.

Table B-1 Baseboard electrical characteristics

Symbol	Description	Min.	Max.	Unit
3V3	3V3 from power connector J39 or tile site interface (VIO)	3.1	3.5	V
5V	5V from power connector J39	4.75	5.25	V
12V	12V from power connector J39	11.4	12.6	V
-12V	-12V from power connector J39	-11.4	-12.6	V
V_{IH}	High-level input voltage at tile site interface	2.0	3.6	V
V_{IL}	Low-level input voltage at tile site interface	0	0.8	V
V_{OH}	High-level output voltage at tile site interface	2.4	−	V
V_{OL}	Low-level output voltage at tile site interface	−	0.4	V
C_{IN}	Capacitance on any pin	−	20	pF

B.2 Timing specifications

This section provides details of the PBX-A9 baseboard maximum clock frequencies and the tile site AXI bus timings when used stand-alone or with a Logic Tile fitted.

B.2.1 Clock frequency restrictions

The maximum tile site clock (**TSCLK**) frequency that can be used for reliable operation depends on the type of Logic Tile fitted and the complexity of the logic design implemented. The default frequency setting is 25MHz.

Caution

The ICS307 programmable oscillator OSC2 that provides the **TSCLK** reference for **CLK_IN_MINUS1** and **CLK_IN_THRU** for distribution to the tile site can be programmed to deliver very high frequency clock signals (200MHZ). The settings for VCO divider, output divider, and output select values are interrelated and must be set correctly. Some combinations of settings do not result in stable operation. For more information on the ICS clock generator and a frequency calculator, see the IDT web site: `www.idt.com`.

B.2.2 AXI bus timings

Table B-2 lists the tile site multiplexed AXI bus timings.

Table B-2 AC Specifications

Parameter	Symbol	Min	Max	Units	Notes
Clock Cycle	**tTScyc**	30	–	ns	Cmax=15pF
Output valid time after clock edge	**tTSov**	–	6.771	ns	
Output hold time after clock edge	**tTSoh**	0.881	–	ns	
Input setup time to clock edge	**tTSis**	–	4.223	ns	
Input hold time after clock edge	**tTSih**	0.975	–	ns	

Figure B-1 on page B-4 shows the tile site multiplexed AXI timing diagram.

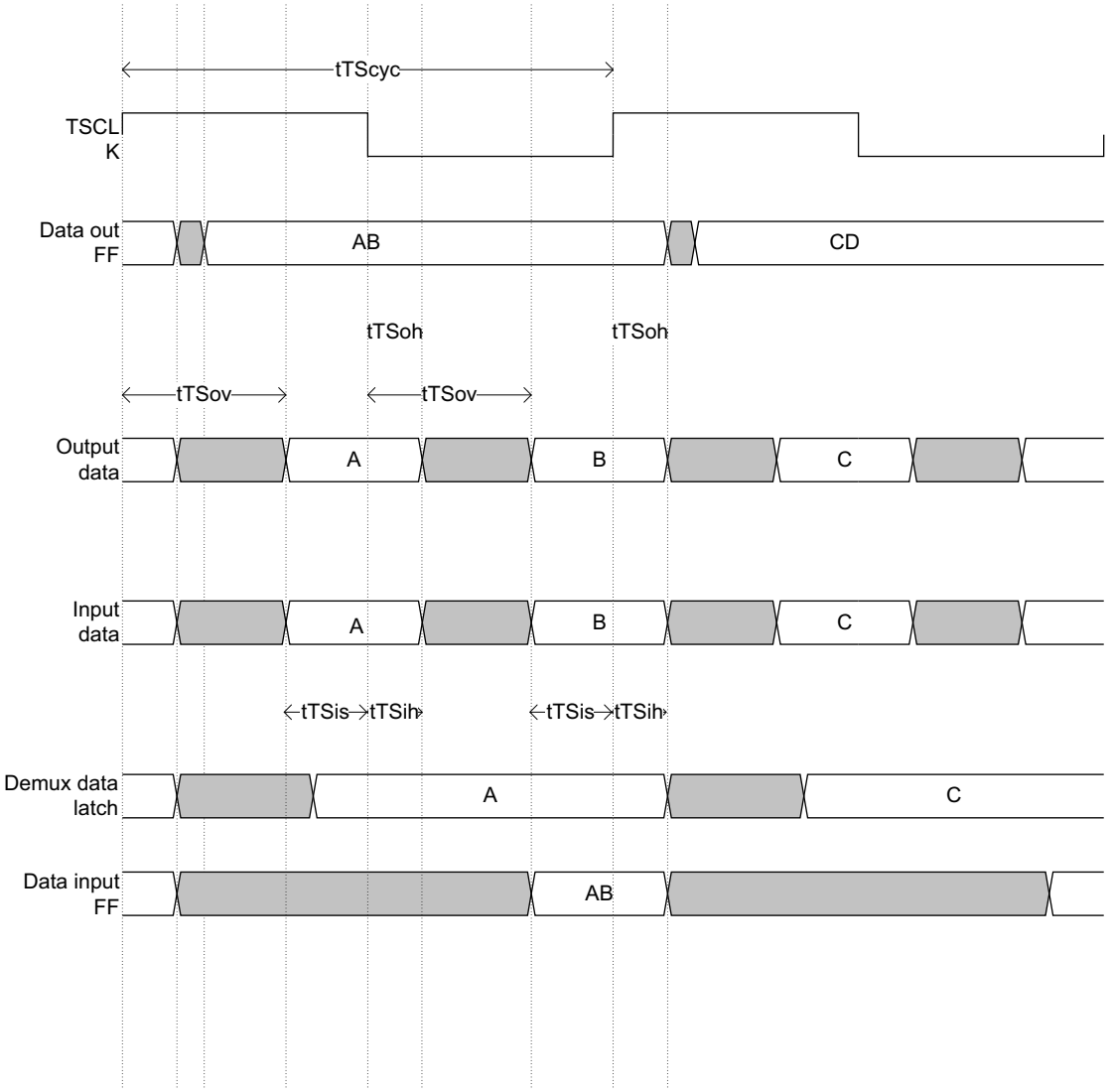


Figure B-1 Tile site multiplexed AXI timing

Appendix C
RealView Logic Tile Expansion

This appendix describes the signals present on the PBX-A9 Logic Tile expansion headers and the steps required to interface a Logic Tile to the baseboard. It contains the following sections:

C.1 About the RealView Logic Tile

RealView Logic Tiles, such as the LT-XC5VLX330, enable developing AMBA 3 AXI, and AMBA APB peripherals, or custom logic, for use with ARM processors.

Figure C-1 shows the signal grouping on the tile site provided for RealView Logic Tile expansion.

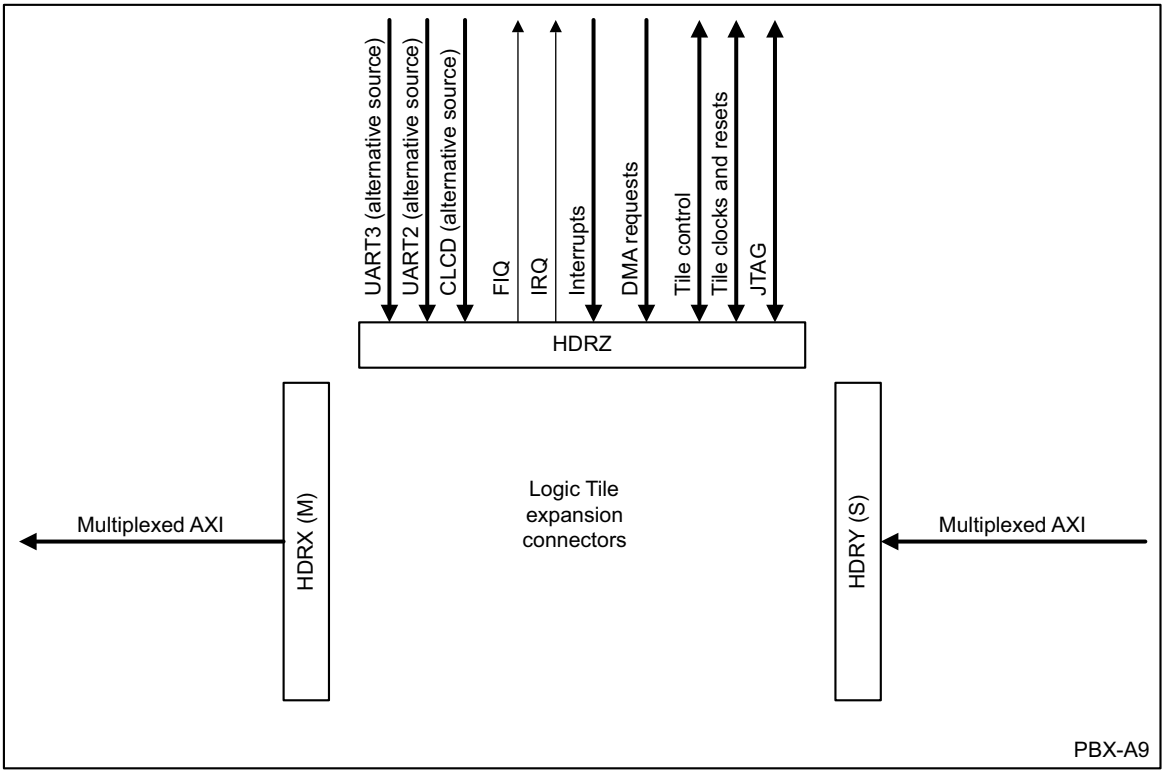


Figure C-1 Signal groups on the PBX-A9 tile site

Note

If you connect a RealView Logic Tile, the design in the tile FPGA must implement logic to handle the Multiplexed AXI bus signals, see Application Note AN151: *Example AXI design for a Logic Tile on top of AXI Versatile baseboards* for an example of suitable code.

C.2 Header connectors

This section gives a brief overview of the RealView Logic Tile header connectors and the associated signal groups. For detailed information on a specific tile site interface, see the documentation for the RealView Logic Tile you are using.

There are three headers on the top and bottom of the tile. The HDRX and HDRY headers are 180-way and the HDRZ connectors are 300-way.

Warning

There is a limit to the number of RealView Logic Tiles that can be stacked on a RealView baseboard. When stacking tiles ensure that the power source can maintain the required voltage at the top tile when supplying maximum current to the system.

Caution

The FPGA signals on a RealView Logic Tile are fully programmable. Ensure that there are no clashes between the signals on the tiles or with the signals from the PBX-A9.

The FPGA can be damaged if several pins configured as outputs are connected together and attempt to output different logic levels.

Figure C-2 on page C-4 shows the pin numbers and power-blade usage of the HDRX, HDRY, and HDRZ headers on the upper side of the tile. See *RealView Logic Tile header connectors* on page A-19 for details of the signals on the PBX-A9 header connectors.

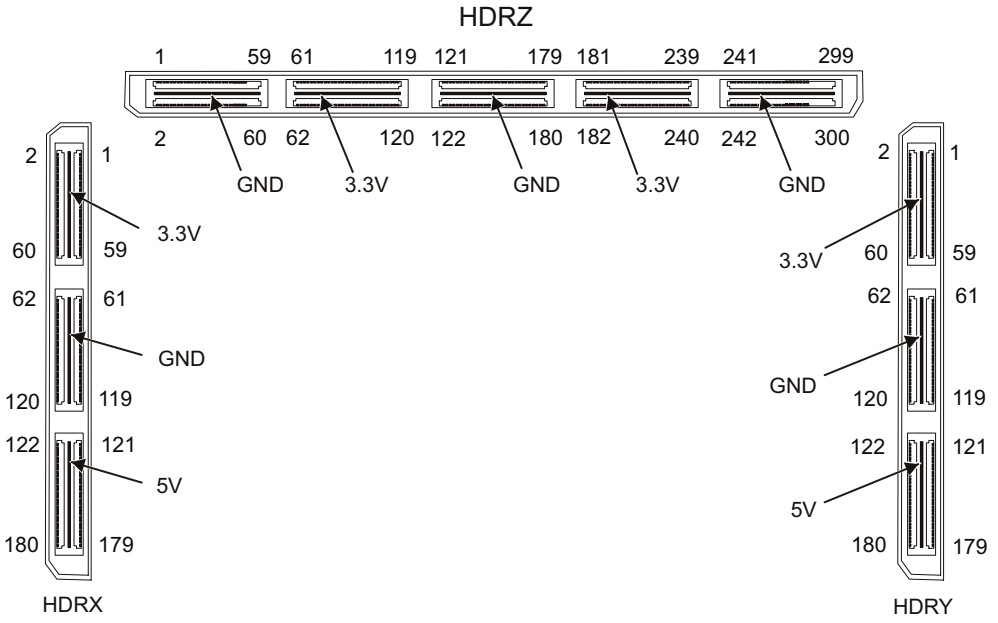


Figure C-2 HDRX, HDRY, and HDRZ (upper) pin numbering

C.2.1 Variable I/O levels

All HDRX, HDRY, and HDRZ connector signals on the PBX-A9 are fixed at a 3.3V I/O signalling level.

Caution

The RealView Logic Tile mounted on the PBX-A9 must use the default 3.3V signal levels.

C.2.2 RealView Logic Tile clock

The PBX-A9 baseboard provides an individually buffered tile site clock **TSCLK** at the HDRZ connector on **CLK_OUT_PLUS1** (pin 140) and **CLK_OUT_PLUS2** (pin 142). The default frequency for **TSCLK** is 25MHz and is set by **OSCCLK2** on the PBX-A9. **TSCLK** is also supplied to the Northbridge async bridges to synchronize AXI signals to and from the tile site.

Note

The remaining RealView Logic Tile clock sources are not used. Ensure that your RealView Logic Tile configuration is compatible with the available clock sources. See *Northbridge clock domains* on page 3-33 for more information on PBX-A9 clock routing.

C.2.3 JTAG

There is no JTAG connector on the RealView Logic Tile, you must use the JTAG debug or USB config connector on the ATX enclosure to debug and configure the Logic Tile.

If multiple RealView Logic Tiles are stacked on the baseboard, the JTAG signals are automatically routed upwards to the top tile and then back down to the baseboard.

Use the JTAG interface to program the configuration flash in the RealView Logic Tile or to directly load the RealView Logic Tile FPGA image. For more information on the JTAG signals, see *Test, configuration, and debug interfaces* on page 3-47.

C.2.4 AXI buses used by the Northbridge and RealView Logic Tiles

Multiplexed AXI buses AXI M and AXI S are connected between the Northbridge and the RealView Logic Tile stack. The user-implemented system in the Logic Tile must co-operate with the system implemented within the Northbridge when using these buses:

AXI M The AXI M bus can only be connected to Multiplexed AXI slaves in the Logic Tile stack.

AXI S The Multiplexed AXI S bus can only be connected to Multiplexed AXI masters in the Logic Tile stack.

AXI slaves on the AXI M bus

The PBX-A9 Northbridge does not contain any slaves attached to the AXI M bus. The PBX-A9 memory map assigns the top 1GB of address space (`0xC0000000`–`0xFFFFFFFF`) to this bus, so a RealView Logic Tile can contain user-supplied slaves that occupy any of this space. The RealView Logic Tile FPGA must give a response to all transfers that are generated on the AXI M bus, even those to addresses in the range `0x00000000`–`0xBFFFFFFF`. The Northbridge never generates these addresses on the AXI M bus. A separate tile master might, however, generate accesses to this region.

In a system without a fully-decoded address map, there can be addresses at which there are no slaves to respond to a transaction. In such a system, the tile site must provide a suitable error response to flag the access as illegal and also to prevent the system from locking up by trying to access a nonexistent slave.

If the tile site cannot successfully decode a slave access, it must route the access to a default slave that returns the **DECERR** response.

The AXI protocol requires that all data transfers for a transaction are completed, even if an error condition occurs. Therefore any component giving a **DECERR** response must meet this requirement.

See *Timing specifications* on page B-3 for details of the PBX-A9 tile site multiplexed AXI bus timings.

C.2.5 Reset

A user design in a RealView Logic Tile can reset the baseboard by driving the **nSRST** signal LOW. This has the same effect as pressing the Soft Reset push-button and forces the reset controller to the level specified by the SYS_RESETCTRL register RESETCTRL field. See *Reset Control Register, SYS_RESETCTL* on page 4-22 for details.

nSRST is synchronized by the reset controller and can be driven from any clock source. It must, however, be driven active for a minimum of 84ns (two cycles of 24MHz) to ensure that it is sampled by the reset controller. To avoid a deadlock condition, the user design must stop driving the **nSRST** signal after **nSYSRST** is asserted.

nSRST is active low and open-drain. It is shared with the JTAG interface and must not be driven to HIGH state. A resistor on the baseboard pulls the signal HIGH.

The RealView Logic Tile also uses the **nPORESET** signal to generate a local **D_nTRST** pulse.

The **GLOBAL_DONE** signal is held LOW until the FPGA on the RealView Logic Tile has finished configuration. The system is held in reset until this signal goes HIGH.

Appendix D
Boot Monitor and platform library

This appendix describes using the Boot Monitor and platform library provided with the PBX-A9 baseboard. It contains the following sections:

D.1 About the Boot Monitor

This is the standard ARM application that runs when the system is booted. It is built with the ARM platform library.

Note

Any application that is built using the ARM platform library (or handles its own initialization) can replace the Boot Monitor.

The Boot Monitor supports the following functions:

- general file operations
- MMC and SD card utilities
- board configuration
- programming images into flash memory
- loading and running another application
- a semihosting server that handles standard ARM semihosting SVC calls.

D.2 About the platform library

The ARM platform library handles the system initialization and re-targets the standard C library. To achieve this, it provides a basic I/O subsystem that supports simple device drivers.

Included with the platform library there is a simple terminal driver, UART, PS/2 keyboard and LCD drivers and support for semihosting I/O.

D.3 Using the baseboard Boot Monitor and platform library

The baseboard Boot Monitor is a collection of tools and utilities designed as an aid to developing applications on the baseboard.

When the Boot Monitor starts on reset, the following actions are performed:

- the memory controllers are initialized
- a stack is set up in memory
- Boot Monitor code is copied into DRAM
- Boot memory remapping is reset
- C library I/O routines are remapped and redirected depending on the settings of User Switches 2 and 3.
- if Boot Monitor configuration switch User Switch 1 is ON, the current boot script, if any, is run.

Caution

The firmware must match the system configuration or the results might be unpredictable. See 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.

D.3.1 Boot Monitor configuration switches

The Boot Monitor application is typically loaded into the NOR flash memory and selected to run at power on. Follow the instructions in *Loading Boot Monitor into NOR flash* on page D-6 for details of loading the boot flash with the image from the Versatile CD.

Caution

The User Switches on the front panel of the ATX enclosure do not operate correctly if switch bank S4 on the PBX-A9 baseboard is not in the default state of all switches OFF.

The setting of User Switch 1 determines how the Boot Monitor starts after a reset:

User Switch 1 OFF A prompt is displayed enabling you to enter Boot Monitor commands.

User Switch 1 ON The Boot Monitor executes a boot script that has been loaded into a Multimedia (MMC) or Secure Digital (SD) card. If a boot script is not present, the Boot Monitor prompt is displayed.

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

Output and input of text from STDIO for both applications and Boot Monitor I/O depends on the setting of User Switches 2 and 3 that Table D-1 lists.

Table D-1 STDIO redirection

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

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

Note

User Switches 4 to 8 are not used by the Boot Monitor and are always available for user applications.

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

D.3.2 Running the Boot Monitor

To run Boot Monitor and have it display a prompt to a terminal connected to UART0, set User Switches 1, 2, and 3 to OFF and reset the system. Standard input and output functions use UART0 by default. The default setting for UART0 is 38400 baud, 8 data

bits, no parity, 1 stop bit. There is no hardware or software flow control. Use these values to configure a terminal application on your PC to communicate with the Boot Monitor.

Note

If the Boot Monitor has been accidently deleted from flash memory, see *Rebuilding the Boot Monitor or platform library* on page D-8 for information on loading the monitor.

Boot Monitor commands

See Appendix E *Boot Monitor Commands* for details of the available commands in Version 4 of the Boot Monitor.

Commands are accepted in uppercase or lowercase. The Boot Monitor accepts abbreviations of commands if the meaning is not ambiguous. For example, for `QUIT`, you can enter `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).

Enter `HELP` at the Boot Monitor prompt to display a full list of the available commands.

D.3.3 Loading Boot Monitor into NOR flash

If the flash becomes corrupt and the board no longer runs the Boot Monitor, the Boot Monitor must be reprogrammed into flash.

Because the debugger does not initialize DRAM, the Boot Monitor image cannot be loaded and run directly. To reload the Boot Monitor into NOR flash:

1. Power off the board
2. Set all User Switches to OFF
3. Connect a cable from the JTAG device (RealView ICE for example) to the JTAG connector on the rear panel.
4. Power on the board.
5. Connect to the target:
   - For RVD, select **Tools → Include Commands From File**
     Select **PBX-A9_DDR_Init_rvd_DLL.inc**

6. DRAM is now initialized and the memory is remapped. To load Boot Monitor into flash:
   a. From the debugger, load and execute the file `Boot_Monitor.axf`
   b. at the Boot Monitor prompt enter:
      ```
      >FLASH
      Flash> WRITE IMAGE path\Boot_Monitor_platform.axf
      ```
      where path is the *path* to the version of the Boot Monitor that is being loaded and *platform* is the name of the target system.
7. Loading the image into flash takes a few minutes to complete. Wait until the prompt is displayed again before proceeding.
8. Turn the board off and then on.

Boot Monitor starts automatically.

D.3.4 Redirecting character output to hardware devices

The redirection of character I/O is carried out within the Boot Monitor platform library routines in `retarget.c` and `boot.s`. During startup, the platform library executes a *Software Interrupt* instruction (SWI). If the image is being executed without a debugger (or the debugger is not capturing semihosting calls) the value returned by this SWI is –1, otherwise the value returned is positive. The platform library uses the return value to determine the hardware device used for outputting from the C library I/O functions Redirection is through a SWI to the debugger console or directly to a hardware device.

Supported devices for character output are:

- `:UART-0` (default destination if debugger is not capturing semihosting calls)
- `:UART-1`
- `:UART-2`
- `:UART-3`.

The STDIO calls are redirected within `retarget.c`. Redirection depends on the setting of User Switches 2 and 3, see *Boot Monitor configuration switches* on page D-4 for details.

D.3.5 Using a boot script to run an image automatically

Use a boot script to run an image automatically after power-on:

1. Create a boot script from the Boot Monitor by entering:
   - if your image is in MMC or SD Card:
     M:\> CREATE myscript.txt
     ; put any startup code here RUN *path\file_name*

- if your image is in CompactFlash:

  ```
  K:\>CREATE myscript.txt
  ; put any startup code here RUN path\file_name
  ```

- if your image is in NOR flash:

  Create an image from the Flash submenu by entering:

  ```
  >Flash
  Flash>CREATE BOOTSCRIPT
  ; put any startup code here FLASH RUN file_name.
  ```

2. Press **Ctrl-Z** to indicate the end of the boot script and return to the Boot Monitor prompt.

   Note

   RVD does not support **Ctrl-Z** when creating a boot script using the debugger. You must instead enter `+++` on a new line to indicate the end of a boot script.

3. Verify the file was entered correctly by entering:

   ```
   M:\>TYPE myscript.txt
   ```

   The contents of the file is displayed to the currently selected output device.

   Note

   In the particular case where the bootscript is in NOR Flash, to display the contents of the bootscript, enter DISPLAY BOOTSCRIPT from the Flash submenu. Step 4 is not required.

4. Specify the boot script to use at reset from the Boot Monitor by typing:

   ```
   M:\>SET BOOTSCRIPT myscript.txt
   ```

5. Set User Switch 1 ON to instruct the Boot Monitor to run the boot script at power on.

6. Reset the platform. The Boot Monitor runs and executes the boot script `myscript.txt`. In this case, it relocates the image *`file_name`* and executes it.

D.3.6 Rebuilding the Boot Monitor or platform library

For Windows and Linux, ARM RVDS should be used to rebuild the Boot Monitor or platform library.

Note

Use a CodeWarrior project file as the RVDS `make` utility is not supported.

After rebuilding the Boot Monitor, load it into NOR flash, see *Loading Boot Monitor into NOR flash* on page D-6.

After rebuilding the platform library, you can link `platform.a` from the target build subdirectories with your application, see *Building an application with the platform library*.

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, enter `make clean all`.

D.3.7 Building an application with the platform library

The platform library on the CD provides all required initialization code to bring the baseboard up from reset. The library is used by the Boot Monitor, but it can be used by an application independently of the other code in the Boot Monitor.

Note

It is not necessary to build your application with the platform library. You can instead let the boot monitor run at power on and remap the memory. After the remap has finished, you can load a typical application built with RVDS that is linked at address `0x8000` and uses semihosting.

The platform library supports:

- remapping of boot memory
- DRAM initialization
- UARTs
- Time-of-Year clock
- output to the character LCD display
- C library system calls.

To build an image that uses the I/O and memory control features present in the platform library:

1. Write the application as normal. There must be a `main()` routine in the application.

   Linking an application with the platform requires that the application is built using RVCT V2.1 or greater. If you are using RVCT V2.0 or ADS it is not possible to link the application with the platform library, however an application can still utilize the hardware on the board through semihosting.

2. Link the application against the Boot Monitor platform library file `platform.a`. The file `platform.a` is in one of the target build subdirectories (*`install_dir`*`\software\firmware\Platform\Builds\`*`target_build`*).

   Choose the `Builds` subdirectory that matches your application. For example, `Release_ARM_Little_Endian` for ARM code. If the subdirectory does not exist, see *Rebuilding the Boot Monitor or platform library* on page D-8 for details on rebuilding `platform.a`.

   Note

   If you are not using the `platform.a` library, you must provide your own initialization and I/O routines.

   You can also build the platform library functionality directly into your application without building the platform code as a separate library. This might be useful, for example, if you are using an IDE to develop your application.

   See the `filelist.txt` file in the `software` directory for more details on software included on the CD. The `selftest` directory, for example, contains source files that can be used as a starting point for your own application.

3. If required, select the link time selection for the platform library options.

   Platform selection uses special symbols in your application:

   From C `#pragma import(_platform_option_`*`xxxxx`*`)`

   From assembler `IMPORT _platform_option_`*`xxxxx`*

Table D-2 lists the platform options.

Table D-2 Platform library options

Option	Description
_platform_option_no_cache	Stops the cache from being enabled by default
_platform_option_no_lcd_kbd	Disables LCD & Keyboard support and stops the driver code from being loaded
_platform_option_no_mmu	Stops the MMU from being initialized by default.

4. Scatter loading is supported for applications using the platform library, however the scatter file must follow Example D-1. The execution regions INIT and SDRAM must be present, the execution regions ITCM and DTCM are optional, if they are present, the relevant *Tightly Coupled Memory* (TCM) is enabled and the base address is set to the address supplied in the scatter file. The `sys_vectors.o` object must be located at address zero. Additional execution regions, such as one for the SSRAM, can be added. An example scatter loading application can be found in the `examples` directory.

Example D-1 Scatter loading

```
LR_ROOT 0x8000
{
     INIT +0 FIXED
    {
       sys_boot.o (!!!_platform_area_boot, +FIRST)
       *(+RO)
     }
     SDRAM +0
    {
      *(+RW,+ZI)
    }
    ITCM 0x0
   {
        sys_vectors.o(_platform_area_vectors, +FIRST)
        'application code'.o(+RO)
    }
    DTCM 0x08000000
    {
       'application code'.o(+RW,+ZI)    }
}
```

5. To run the image from RAM, load the image with a debugger and execute as normal. The image uses the procedure described in *Redirecting character output to hardware devices* on page D-7 to redirect standard I/O either to the debugger or to be handled by the application itself.

   See *Loading and running an application from NOR flash* 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.

D.3.8 Building an application that uses semihosting

The boot monitor handles semihosting SWIs the same as a debugger handles SWIs. This enables an image that is not linked against with the platform library to be loaded into flash memory and have the boot monitor manage the I/O.

All I/O using `stdio()`, for example `printf()`, uses the same devices as the boot monitor console (either UART-0 or the Keyboard/LCD depending on the Boot Monitor configuration switch settings). File functions access the flash and character I/O devices using the mechanisms described in *Redirecting character output to hardware devices* on page D-7.

There are no specific tools requirements. Images built with RealView tools typically run if they are built to use semihosting. This means that an image built using the tool kit defaults can be loaded onto the baseboard and run.

D.3.9 Loading and running an application from NOR flash

To run an image from NOR flash:

1. Build the application as described in *Building an application with the platform library* on page D-9 and specify a link address suitable for flash. There are the following options for selecting the address:

   Load region in flash

   The image is linked such that its load region, though not necessarily its execution region, is in flash. The load region specified when the image was linked is used as the location in flash and the FLASH_ADDRESS option is ignored. If the blocks in flash are not free, the command fails. Use the `FLASH RUN` command to run the image.

Load region not in flash and image location not specified

The image is programmed into the first available contiguous set of blocks in flash that is large enough to hold the image. Use the `FLASH LOAD` and then the `FLASH RUN` commands to load and run the image.

Load region not in flash, but image stored at a specified flash address

Use the FLASH_ADDRESS option to specify the location of the image in flash. If the option is not used, the image is programmed into the first available contiguous set of blocks in flash that is large enough to hold the image. Use the `FLASH LOAD` or `FLASH RUN` commands to load and run the image.

Note

Images with multiple load regions are not supported.

If the image is loaded into flash, but the `FLASH RUN` command relocates code to DRAM for execution, the execution address must not be in the top 4MBytes of DRAM because 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, enter `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`* is derived from the file name.

4. The image is now in flash and can be run by the Boot Monitor. At the prompt, enter:

   ```
   flash> RUN name
   ```

D.3.10 Running an image from MMC or SD card or CompactFlash

To run an image from the MMC or SD card or CompactFlash:

1. Build and link the application as described in *Loading and running an application from NOR flash* on page D-12.

   Note

   Images with multiple load regions are not supported.

   The image must have an execution region in RAM or DRAM.

   The execution address must not be in the top 4MBytes of DRAM because this area is used by the Boot Monitor.

2. The image can be programmed into MMC or SD card using the Boot Monitor.

   Connect a debugger and use semihosting to load the file:

   For MMC or SD card:

   ```
   M:\>COPY C:\software\elf_file_name file_name
   ```

   For CompactFlash:

   ```
   K:\>COPY C:\software\elf_file_name file_name
   ```

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

   For MMC or SD card:

   ```
   M:\>RUN file_name
   ```

   For CompactFlash:

   ```
   K:\>RUN file_name
   ```

D.3.11 Using the Network Flash Utility

The *Network Flash Utility* (NFU) uses the TFTP protocol to access files over the Ethernet network. You can use this utility to program files into flash.

To connect to a server and program a file to flash:

1. Start the NFU utility from the debugger console:
   a. Set the Boot Monitor configuration switches to force the console to use either UART-0 or the LCD and keyboard. (See *Boot Monitor configuration switches* on page D-4 for details.)

      Note

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

   b. Start the NFU utility.

      Note

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

2. Use the DHCP protocol to get an IP address by entering:

   ```
   manage dhcpc start
   ```

3. Use the map command to map a drive letter to the TFTP server. For example to access a file on a TFTP server with the IP address 192.168.0.1, use:

   ```
   manage map n: 192.168.0.1
   ```

4. After the drive letter has been mapped, use the normal Boot Monitor command on the remote file by specifying the drive letter. For example, to write a file to NOR flash, enter:

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

NFU commands

The NFU supports a subset of the standard Boot Monitor commands and adds a new `MANAGE` sub-menu.

Table D-3 lists the NFU commands.

Table D-3 NFU commands

Command	Action
`CD directory path`	Change directory
`CONVERT BINARY binary_file LOAD_ADDRESS address [ENTRY_POINT address]`	Provides information to the system that is required by the RUN command to execute a binary file. A new file with name *binary_file* is produced, but with an `.exe` file extension.
`COPY file1 file2`	Copy *file1* to *file2*. For example, to copy the `leds` code from the PC to the flash enter: `COPY C:\software\projects\examples\rvds2.0\leds.axf leds.axf` **Note** Remote file access requires semihosting. Use a debugger connection to provide semihosting.
`CREATE filename`	Create a new file in the flash by inputting text. Press Ctrl-Z to end the file.
`DELETE filename`	Delete *file* from flash.
`DIRECTORY [directory]`	List the files in a directory. Files that are 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 E-4 on page E-6 for NOR flash commands.
`HELP`	Lists the NFU commands.
`M:`	Change drive (allocated to MMC or SD card).
`K:`	Change drive (allocated to CompactFlash card).
`MANAGE`	Enter the network management sub-menu. See Table D-4 on page D-17 for MANAGE commands.
`MKDIR directory path`	Create a new directory.
`QUIT`	Alias for `EXIT`. Exit the Boot Monitor.
`RENAME old_name new_name`	Rename flash file named *old_name* to *new_name*.

Table D-3 NFU commands (continued)

Command	Action
RMDIR *directory path*	Remove a directory.
SDCARD	Enter the SD card subsystem.
TYPE *filename*	Display the flash file *filename*.

The MANAGE sub-menu that Table D-4 lists contains the network management commands.

Entering MANAGE on the command line means that all future commands (until EXIT is entered) are commands from the MANAGE sub-menu.

A single command can be executed by entering MANAGE followed by the command. For example, MANAGE DHCP START gets a IP address from the DHCP server. The next command entered must be from an NFU command.

Table D-4 NFU MANAGE commands

Command	Action
ARP [-a]	Display *Address Resolution Protocol* host table.
ARP -s *hostname*	Add static entry to ARP table.
ARP -d *hostname*	Delete static entry from ARP table.
DHCPC START *ifname*	Use *Dynamic Host Configuration Protocol* (DHCP) to start a connection with the network interface *ifname*.
DHCPC RELEASE *ifname*	Use DHCP to release the connection with the network interface ifname.
DHCPC SIZEOF	Returns information on the size of the DHCP packet.
DHCP INFORM *ifname ip_address*	Uses the DHCP protocol send information to the server located at *ip_address* with the network interface *ifname*.
EXIT	Exit the MANAGE sub-menu. The commands that Table D-3 on page D-16 lists 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*.

Table D-4 NFU MANAGE commands (continued)

Command	Action
IFCONFIG [*ifname* [*option*]]	Configures the IP interface *ifname*. The value for *option* can be: `netmask maskvalue` set the netmask `dstaddr address` set the destination IP address `mtu n` set the maximum transfer unit `up` activate the interface `down` shutdown the interface
MAP *drive address*	Maps the IP address specified in *address* to *drive*.
NETSTAT [-*option*]	Displays active network connections. The value for *option* can be: `a` display all connections `m` display all multicast connections `i` display interface information `im` display interface information for multicast `r` display routing table `s` display statistics `b` display buffer usage
PING *ip_address*	Send ICMP ECHO_REQUEST packets to the network host. The data in the packet is returned by the host. Reception of the return packet indicates that the TCP/IP connection is functioning.
QUIT	Alias for EXIT. Exit the MANAGE sub-menu.
ROUTE ADD *type target* [NETMASK *mask*] *gateway*	Adds a static route to the network address specified by *target*. The gateway address is specified by *gateway*. *type* can be either -net or -host. If NETMASK is used, *mask* is the netmask for the target network address.
ROUTE DEL *target*	Deletes the static route to the network address specified by *target*.
SHOW DNS	Displays *Domain Name System* (DNS) configuration details received from DHCP.

Using a script file with NFU

When NFU starts, it attempts to run the NETSTART.BAT file in the flash. If the script does not exist, a prompt is displayed on the console. For example, to map a drive and write a file to flash, create the following script file:

```
manage dhcp start
manage map n: 192.168.0.1
flash write image n:/hello.axf
```

After the file is executed, you can enter additional NFU commands. To run the file, reset the board and use the `RUN` command from Boot Monitor.

Appendix E
Boot Monitor Commands

This appendix describes Version 4 of the PBX-A9 Boot Monitor command set. It contains the following section:

E.1 About Boot Monitor commands

The Boot Monitor is the standard ARM application that runs when the system is booted.

The Boot Monitor accepts user commands from the debugger console window or an attached terminal. A command interpreter carries out the necessary actions to complete the user commands.

Note

Commands are accepted in uppercase or lowercase. The Boot Monitor accepts abbreviations of commands if the meaning is not ambiguous. For example, for `QUIT`, you can enter `QUIT`, `QUI`, `QU`, `Q`, `quit`, `qui`, `qu`, or `q`.

Optional parameters for commands are indicated by `[param]`. For example `DIRECTORY [directory]` indicates that the `DIRECTORY` command can take an optional parameter to specify which directory to list the contents of. If no parameter is supplied, the default is used, in this case, the current directory.

Enter `HELP` at the Boot Monitor prompt to display a full list of the available commands.

E.2 Boot Monitor command set

Table E-1 Standard Boot Monitor command set

Command	Action
`@ script_file`	Runs a script file.
`ALIAS alias commands`	Create an alias command *alias* for the string of commands contained in *commands*.
`CLEAR BOOTSCRIPT`	Clear the current boot script. The Boot Monitor prompts for input on reset even if the S6-1 is set to ON to indicate that a boot script must be run.
`CONFIGURE`	Enter Configure subsystem. Commands that Table E-2 on page E-4 lists 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 to execute a binary file. A new file with name *binary_file* is produced, but with an `.exe` file extension.
`COPY file1 file2`	Copy *file1* to *file2*. For example, to copy the `leds` code from the PC to the MMC or SD card enter: `COPY C:\software\projects\examples\rvds2.0\leds.axf leds.axf` **Note** Remote file access requires semihosting. Use a debugger connection to provide semihosting.
`CREATE filename`	Create a new file by inputting text. Press Ctrl-Z to end the file.
`DEBUG`	Enter the debug subsystem. Commands that Table E-3 on page E-5 lists can now be executed.
`DELETE filename`	Delete *file* from MMC or SD card or CompactFlash card
`DIRECTORY [directory]`	List the files in a MMC or SD card or CompactFlash card directory. Files only accessible from semihosting cannot be listed.
`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 E-4 on page E-6 for flash commands.

Table E-1 Standard Boot Monitor command set (continued)

Command	Action
HELP [*command*]	List the Boot Monitor commands. Entering HELP followed by a command displays help for that command.
LOAD *name*	Load the image *name* into memory and run it.
QUIT	Alias for EXIT. Exit the Boot Monitor.
RENAME *old_name new_name*	Rename file named *old_name* to *new_name*.
RUN *image_name*	Load the 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 S4-1 is ON, *script_file* is run at system reset.
TYPE *filename*	Display the file *filename*.

Table E-2 lists the commands for the Configure sub-menu.

Table E-2 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. Entering HELP followed by a command displays help for that command.
QUIT	Alias for EXIT. Exit the Configure commands and return to standard Boot Monitor commands.

Table E-2 Boot Monitor Configure commands (continued)

Command	Action
SET CLOCK n FREQUENCY *frequency*	Set the *frequency* in MHz of the requested CLOCK n. **Note** The Boot Monitor does not set any of the clocks on startup. The clock values are determined by the default values in the FPGA image.
SET DATE *dd/mm/yy*	Set date. The date can also be entered as *dd-mm-yy*.
SET TIME *hh:mm:ss*	Set time. The time can also be entered as *hh-mm-ss*.

Table E-3 lists the commands for the Debug sub-menu.

Table E-3 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.

Table E-4 lists the commands for the NOR Flash subsystem.

Table E-4 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* is erased if they are not on block boundaries. If the optional *end* parameter is not specified, only the single block of flash that contains *start* is erased. **Caution** This command can erase the Boot Monitor image. See *Loading Boot Monitor into NOR flash* on page D-6.
`EXIT`	Exit the NOR flash commands and return to executing standard Boot Monitor commands.
`HELP [command]`	List the flash commands. Entering `HELP` followed by a command displays help for that command.
`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 is not 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.

Table E-4 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.axf is identified as boot_monitor. Use `NAME new_name` to specify a name instead of using the default name.
	Use `FLASH_ADDRESS address` to specify where in flash the image is to be located. If the image is linked to run from flash, the link address is used and *address* is ignored.
	Note
	Remote file access requires semihosting. Use a debugger connection to provide semihosting.

Appendix F
Loading FPGA Images

This section describes the format of the PBX-A9 board files and how to use the `progcards` utilities to load images from the supplied CD into the baseboard and Logic Tile FPGAs and PLDs. It contains the following sections:

F.1 General procedure

The general procedure to load an image is:

1. Follow the instructions in the *Versatile CD Installation Guide* to install the software and data files on your hard disk.
2. Set up the baseboard, the daughter board, and any optional Logic Tiles you might be using, as described in *Setting up the baseboard* on page 2-2.
3. Connect RealView ICE to the JTAG connector or use a USB cable to connect a PC to the USB config port. See *JTAG, USB config, and Trace support* on page 2-6.
4. Place the baseboard in configuration mode (Config switch ON) and power on the development system.
5. Locate the board file (`.brd`) that matches your configuration. See *Board files* on page F-3.

   Caution

   The baseboard is shipped with multiplexed AXI connectivity in place for the tile site but you must load an image into the Logic Tile FPGA before you can use the baseboard with a Logic Tile. See the application note for your Logic Tile for specific instructions.

   The Versatile CD also includes Boot Monitor code for NOR flash. You can load the Boot Monitor code separately without reloading FPGA images, see *Loading Boot Monitor into NOR flash* on page D-6.
6. Run the `progcards` utility and load the image files into the FPGAs. See *Upgrading your hardware* on page F-7). The board file selects the image files to load. Board files have a `.brd` extension.

F.2 Board files

The CD includes both the `progcards` programming utilities and the images to load into the FPGAs and PLDs.

F.2.1 Naming conventions for board files

The file name of a board file identifies the PCB, Logic Tile, all programmable devices, and revision. Using the board file eliminates the need to load several individual image files. All file names are in lower case with an underscore character separating the fields:

appnote_boardname_number_core_endian_build_devicelist.brd

where:

appnote The *Application Note* related to this board file.

boardname_number The name and number identify a specific development board. For example, `eb_140c` is for the baseboard that uses PCB board HBI-140C.

Note

The full board number is `HBI-XXXX`R, this is abbreviated to the significant digits and revision, for example, `HBI-0140C` becomes `140C`.

core The name of the core that is used with this configuration. For example, `mpcore_li` identifies the PBX-A9 baseboard 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.

F.2.2 Naming conventions for image files

The image files contain the image for a single FPGA, PLD, or programmable memory device. The file name of an FPGA or PLD image file identifies the PCB, device and revision. All file names are in lower case with an underscore character separating the fields:

*boardname_number_devicename_device*_build*n*.*extension*

where:

boardname_number The name and number identify a specific development board. For example, eb_140c is for the baseboard that uses PCB board HBI-140C.

devicename_device

The name of the programmable devices that match this file. For example, cfg_xc2c128 identifies the Xilinx XC2C128 configuration PLD.

build*n* The build number. The number *n* increments from 0.

extension The type of device used by this file. For example .svf is used for PLD programming and .bit is used for FPGA programming.

Caution

The baseboard is supplied with the PLD images already programmed. The information in this section is provided, however, in case of accidental erasure of the PLDs.You are advised not to reprogram the PLDs with any images other than those provided by ARM Limited.

Using the board file to control image file loading minimizes the risk of incorrectly programming the board. The board files contains a list of correct image files and eliminates the requirement to select individual image files for the programmable devices on the baseboard or attached Logic Tile.

F.3 The progcards utilities

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

progcards reads a description of the board JTAG scan chain and a list of operations from a board description (*.brd) file. The file describes which bitstream and configuration files (*.bit, *.svf) must be downloaded to devices on the board.

The upgrade package for a board contains the new files and all previously released versions. This enables you to return to the original configuration.

Example F-1 shows a board file.

Example F-1 Board file

```
[General]
Name = PBX (HBI-0182B) FPGA build 2, CSPLD build 0, ispClock build 1 for PBX-A9
Priority = 1
Board = 182b
[ScanChain]
TAPs = 4
TAP0 = XC5VLX110
TAP1 = XC2C32A
TAP2 = ispClock5304
TAP3 = XC2C32A
[Program]
SequenceLength = 5
Step1TCKSpd    = 5
Step1Method    = SVF
Step1TCKSpd    = 5
Step1TAP       = 1
Step1File      = pbx_hbi0182\pbx_182b_xc2c32a_cspld_build0.svf
Step2Method    = PLD
Step2TAP       = 2
Step2File      = pbx_hbi0182\pbx_183b_isp5304_ispclock_build1.svf
Step3Method    = Virtex5
Step3TAP       =
Step3File      = via\pbx_182b_xc5vlx110_via_build0.bit
Step4TAP       = 0
Step4Method    = IntelFlash
Step4Address   = 0000000
Step4ByteFix   = 1
Step4File      = pbx_hbi0182\pbx_182b_xc5vlx110_pbxa9sb_revb_build2.bit
Step5TAP       = 0
Step5Method    = IntelFlashVerify
```

```
Step5Address    = 0000000
Step5ByteFix    = 1
Step5File       = pbx_hbi0182\pbx_182b_xc5vlx110_pbxa9sb_revb_build2.bit
```

F.4 Upgrading your hardware

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

progcards_rvi.exe

`progcards_rvi.exe` uses RealView ICE and the JTAG interface. It runs on a PC host in a DOS window and communicates with the RealView ICE interface box.

See *Procedure for progcards_rvi.exe* for detailed instructions.

progcards_usb.exe

`progcards_usb.exe` uses the built-in USB Config port (USB to JTAG interface logic) on the baseboard. A standard USB cable connects the baseboard to the PC running `progcards_usb.exe`.

See *Procedure for progcards_usb.exe* on page F-9 for detailed instructions.

Note

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

F.4.1 Procedure for progcards_rvi.exe

1. Ensure that the RealView ICE firmware is version 3.2.1 (or later) and has the additional patch required for running `progcards_rvi`. The patch can be downloaded from the ARM web site. See the `readme` file supplied with `progcards_rvi` for information on updating the firmware.
2. Connect the 20-way JTAG cable from the RealView ICE JTAG interface to the JTAG connector (J10) on the baseboard in the ATX enclosure. See Figure 3-1 on page 3-2.
3. Move the Config switch on the front panel of the ATX enclosure to the ON position. The orange Config indicator lights.
4. Turn on the power, the orange Config indicator on the front panel of the ATX enclosure lights.
5. Start a command window by selecting **Run** from the **Start** menu and entering `command` in the text box.

6. Change directory to the directory that contains board description (`*.brd`) files for the design to be programmed.
7. Start the programming utility by entering `progcards_rvi` at the command prompt.
8. A menu is displayed asking which interface box you want to connect to. Select the interface that is connected to the baseboard.

   Note

   See the documentation supplied with `progcards_rvi` for more information on connecting directly to a specified interface that is connected to either the network or the local USB connection on the PC.

9. RVI attempts to autoconfigure. If auto-configuration fails, see the documentation supplied with `procards_rvi`.
10. `progcards_rvi` searches for board description files that match the JTAG scan chain. All board descriptions matching the first part of the chain are presented as a menu and you can select the file to use.

    `progcards_rvi` runs through the steps required to completely reprogram the boards.
11. To bypass programming a Logic Tile or the baseboard select the `Skip` option from the menu. `progcards_rvi` then looks for board description files that match the next segment of scan chain and so on.

    Typically one menu is displayed for the Logic Tile and one menu is displayed for the baseboard. If only one board description matches your hardware, it is automatically selected and no menu is displayed.

    Caution

    Ensure that the image file you are loading matches your system configuration. If the incorrect files are loaded, the baseboard and tiles might not function or might be unreliable.

12. After downloading the image completes, turn the power off and move the Config switch on the front panel of the ATX enclosure to the OFF position.
13. Set the configuration switches to match the boot option you are using. See *Baseboard configuration switches* on page 2-8.
14. Power on and use the Boot Monitor to load your application. See Appendix D *Boot Monitor and platform library*.

    If you are not using the Boot Monitor, use a JTAG debugger to load and run an application. See the documentation supplied with your debugger for details.

F.4.2 Procedure for progcards_usb.exe

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

   Note

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

2. Connect the USB cable from the host PC to the USB Config port on the front panel of the ATX enclosure. See Figure 3-2 on page 3-4.
3. Move the Config switch on the front panel of the ATX enclosure to the ON position.
4. Turn on the power. The orange Config LED lights.
5. Start a command window by selecting **Run** from the **start** menu and entering `command` in the text box.
6. Change directory to the directory that contains board description (`*.brd`) files for the design to be programmed.
7. Start the programming utility by entering `progcards_usb` at the command prompt.

   Note

   If `progcards_usb.exe` is not in the current working directory, set your `PATH` environment variable to point to the directory that contains `progcards_usb.exe`.

8. `Progcards_usb` searches for board description files that match the scan chain. All board descriptions matching the first part of the chain are presented as a menu and you can select the file to use.

   `progcards_usb` runs through the steps required to completely reprogram the boards.

9. To bypass programming a Logic Tile or the baseboard select the `Skip` option from the menu. `progcards_usb` then looks for board description files that match the next segment of scan chain and so on.

   Typically one menu is displayed for the Logic Tile and one menu is displayed for the baseboard. If only one board description matches your hardware, it is automatically selected and no menu is displayed.

10. After downloading the image completes, turn the power off and move the Config switch to the OFF position.

11. Set the configuration switches to match the boot option you are using. See *Baseboard configuration switches* on page 2-8.
12. Power on the board and use the Boot Monitor to load your application. See Appendix D *Boot Monitor and platform library*.

    If you are not using the Boot Monitor, use a JTAG debugger to load and run an application. See the documentation supplied with your debugger for details.

F.4.3 Troubleshooting

If the upgrade process fails at any time, or the `progcards` utility reports an error, then check the following:

1. The Config switch is ON and the orange Config LED is on.
2. If you are using `progcards_rvi.exe`, see the documentation supplied with `progcards_rvi`.
3. If you are using `progcards_usb.exe` ensure that:
   a. the USB cable is plugged into the USB Config port on the front panel of the ATX enclosure and not one of the USB user ports on the rear panel.
   b. the USB Config drivers are installed on your host machine.
4. Ensure that any optional Logic Tiles are correctly stacked on the baseboard. If necessary, push down firmly on the tile connectors to ensure a proper connection.
5. Look for any `readme.txt` files that might be present in the directory that contains the board description (`*.brd`) files. Follow the instructions provided for new or updated software or engineering changes.

F.5 Loading PLD images

Caution

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

Program the configuration PLD as follows:

1. Connect an interface cable to either the JTAG or USB config port.
2. Set the Config switch on the front panel to ON (up position).
3. Power-up the board. The Config LED lights.
4. Run the appropriate `Progcards` utility from:
   `install_directory``\Versatile\PBX-A9_HBI0183\`*`Release`*`\`*`build`*`\boardfiles\`

Glossary

This glossary lists abbreviations used in the PBX-A9 User Guide.

ASIC	Application Specific Integrated Circuit.
ADC	Analog to Digital Converter. A device that converts an analog signal into digital data.
AHB	Advanced High-performance Bus. An ARM open standard bus protocol.
AMBA 3	Advanced Microcontroller Bus Architecture version 3.
AXI	AMBA 3 Advanced eXtensible Interface. An ARM open standard bus protocol.
BIST	Built In Self Test
DAC	Digital to Analog Converter. A device that converts digital data into analog level signals.
EB	RealView Emulation Board. A hardware platform used for system prototyping and debugging of ARM microprocessors.
FPGA	Field Programmable Gate Array.
ICE	In Circuit Emulator. A interface device for configuring and debugging processor cores.
I/O	Input/Output.

JTAG Joint Test Action Group. The committee that defined the IEEE test access port and boundary-scan standard.

LED Light Emitting Diode.

Multi-ICE Equipment for running and controlling a JTAG interface for system debug. This is legacy JTAG equipment. Does not support the Cortex-A9 processor.

PCI Peripheral Component Interconnect. A circuit board level bus interconnect.

PISMO Memory specification for plug in memory modules.

PLD Programmable Logic Device.

PLL Phase-Locked Loop. A type of programmable oscillator that tracks the input frequency and can generate arbitrary multiples or divisions of the input frequency.

RAM Random Access Memory.

RVI RealView ICE. Equipment for running and controlling a JTAG interface for system debug. This is current JTAG equipment. Supports the Cortex-A9 processor.

TCM Tightly Coupled Memory, a fast memory block that connects directly to a dedicated I/O port on the processor.

USB Universal Serial Bus. Hardware interface for connecting peripheral devices.