Versatile Application Baseboard for ARM926EJ-S

HBI-0118

User Guide

ARM®

ARM DUI 0225D

Versatile Application Baseboard for ARM926EJ-S
User Guide

Release Information

Change history

Description	Issue	Confidentiality	Change
December 2004	A	Non-Confidential	First release
July 2006	B	Non-Confidential	Second release with fixes for errata
March 2009	C	Non-Confidential	Third release with fixes for errata
April 2011	D	Non-Confidential	Fourth release with fixes for errata

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

The Versatile/AB926EJ-S, Versatile/AB-IB1, and Versatile/IB-2 are test equipment and consequently are exempt from part 15 of the FCC Rules under section 15.103 (c).

The Versatile/AB-IB2 contains an Enfora Enabler II-G GSM Radio Module (FCC ID: MIVGSM0108). The GSM module complies with Part 15 of the FCC Rules for equipment intended for use in a commercial, industrial, or business environment. Use of the GSM module present on the Versatile/AB-IB2 is subject to the following conditions:

- This GSM module must not cause harmful interference.
- This GSM module must accept any interference received, including interference that may cause undesired operation.
- The Versatile/AB926EJ-S and the Versatile/AB-IB2 must be used as a mobile or fixed device as defined under FCC regulations. Do not operate the Versatile/AB926EJ-S and the Versatile/AB-IB2 as a portable device. Use of the device on a desktop or other applications where the antenna can easily be relocated are considered by the FCC to be mobile applications.
- Antenna gain for the GSM module output is limited to 3dBi. Modifications and/or additions to the Enfora Enabler II-G GSM transceiver, including use of antennas with higher than 3dBi gain, are prohibited. Mobile devices are limited to 2W EIRP under Part 24.

  The nominal RF output power of the GSM transceiver is 1.0W. The antenna supplied with the Versatile/AB-IB2 has a gain of 0dBi. The requirement for less than 2W EIRP will therefore be met if there are no modifications to the transceiver or antenna.

Enfora certifies that its Enfora Enabler II-G GSM Radio Module complies with the RF hazard requirements applicable to broadband PCS equipment operating under the authority of 47 CPR Part 24, Subpart E of the FCC Rules and Regulations. This certification is contingent upon installation, operation, and use of the Enfora Enabler II-G GSM Module in accordance with all instructions provided to the OEM and the end user. When installed and operated in a manner consistent with the instructions provided, the Enfora Enabler II-G meets the maximum permissible exposure (MPE) limits for general population/uncontrolled exposure as defined in Section 1.1310 of the FCC Rules and Regulations.

Warning

The AB-IB2 GSM module emits RF radiation. Keep at least 20cm (7.9in) separation distance from the antenna and the human body.

Follow all safety instructions in this manual, any warning notices on the product, and any applicable legal requirements and safety regulations (see also *GSM module* on page D-22 and the *Enfora Enabler IIG GSM/GPRS Radio Modem Integration Guide* GSM0108-01 at the Enfora website at `www.enfora.com`.)

The transmitter and antenna must not be collocated or operating in conjunction with any other antenna or transmitter. Failure to observe this warning could produce an RF exposure condition.

CE Declaration of Conformity

CE

The system should be powered down when not in use.

The board 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
Versatile Application Baseboard for ARM926EJ-S User Guide

List of Tables
Versatile Application Baseboard for ARM926EJ-S User Guide

List of Figures
Versatile Application Baseboard for ARM926EJ-S User Guide

Preface

This preface introduces the *Versatile Application Baseboard for ARM926EJ-S User Guide*. It contains the following sections:

About this document

This document describes how to set up and use the Versatile/AB926EJ-S.

Intended audience

This document has been written for experienced hardware and software developers to aid the development of ARM-based products using the Versatile/AB926EJ-S board as part of a development system.

Organization

This document is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an introduction to the Versatile/AB926EJ-S development board. 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 Versatile/AB926EJ-S development board.

Chapter 3 ***Hardware Description***

Read this chapter for a description of the hardware architecture of the Versatile/AB926EJ-S. This chapter describes the peripherals, clocks, resets, and debug hardware provided by the board.

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

Read this chapter for a description of the Versatile/AB926EJ-S memory map and registers. There is also basic information on the peripherals and controllers present in the board.

Appendix A ***Signal Descriptions***

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

Appendix B ***Specifications***

Refer to this appendix for electrical, timing, and mechanical specifications.

Appendix C ***Versatile/AB-IB1 Interface Board***

Refer to this appendix for details of the Versatile/AB-IB1 for the Versatile/AB926EJ-S.

Appendix D *Versatile/AB-IB2 Interface Board*

Refer to this appendix for details of the Versatile/AB-IB2 for the Versatile/AB926EJ-S.

Appendix E *LCD Kits*

Refer to this appendix for details of the CLCD expansion kits.

Appendix F *Static Memory Expansion Board*

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

Appendix G *Configuring a USB Debug Connection*

Refer to this appendix for details relating to configuring the USB debug port for use with the RealView Debugger.

Typographical conventions

The following typographical conventions are used in this book:

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

Further reading

This section lists related publications by ARM Limited and other companies that provide additional information.

ARM publications

The following publications provide information about the registers and interfaces on the ARM926PXP development chip:

- *ARM926EJ-S Development Chip Reference Guide* (ARM DDI 0287)
- *ARM926EJ-S Technical Reference Manual* (ARM DDI 0198)
- *ARM926EJ-S™ PrimeXsys Wireless Platform Virtual Component Technical Reference Manual* (ARM DDI 0232)
- *ARM926EJ-S™ PrimeXsys Platform Virtual Component User Guide* (ARM DUI 0213)
- *ARM MOVE Coprocessor Technical Reference Manual* (ARM DDI 0251)
- *ARM VFP9-S Coprocessor Technical Reference Manual* (ARM DDI 0238)
- *ARM MBX HR-S Graphics Core Technical Reference Manual* (ARM DDI 0241).

Note

The Technical Reference Manuals for the MOVE Coprocessor, the PrimeXsys Platform Virtual Components, and the MBX HR-S Graphics Core and the User Guide for the PrimeXsys Platform Virtual Components are only available to licensees of the products. Contact ARM Ltd. for more information.

The following publications provide reference information about the ARM architecture:

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

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

- *Versatile Platform Baseboard for ARM926EJ-S User Guide* (DUI0224)
- *Multi-ICE™ User Guide* (ARM DUI 0048)
- *ARM MultiTrace® User Guide* (ARM DUI 0150)
- *ARM Firmware Suite Reference Guide* (ARM DUI 0102)
- *RealView™ Debugger User Guid*e (ARM DUI 0153)
- *RealView ICE User Guide* (ARM DUI 0155)
- *RealView Compilers and Libraries Guid*e (ARM DUI 0205)
- *RealView Linker and Utilities Guid*e (ARM DUI 0206).

The following publications provide information about ARM PrimeCell® or processor devices:

- *ARM PrimeCell® Synchronous Serial Port Controller (PL022) Technical Reference Manual* (ARM DDI 0194)

- *ARM PrimeCell Real Time Clock Controller (PL031) Technical Reference Manual* (ARM DDI 0224)
- *ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual* (ARM DDI 0173).
- *ARM PrimeCell Keyboard Mouse Controller (PL050) Technical Reference Manual* (ARM DDI 0143)
- *ARM PrimeCell GPIO (PL061) Technical Reference Manual* (ARM DDI 0190)
- *ARM PrimeCell DMA (PL080) Technical Reference Manual* (ARM DDI 0196)
- *ARM PrimeCell Static Memory Controller (PL092) Technical Reference Manual* (ARM DDI 0203)
- *ARM PrimeCell Color LCD Controller (PL110) Technical Reference Manual* (ARM DDI 0161).
- *ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual* (ARM DDI 0172*)*
- *ARM PrimeCell Multiport Memory Controller (PL172) Technical Reference Manual* (ARM DDI 0215)
- *ARM PrimeCell Smart Card Interface (PL131) Technical Reference Manual* (ARM DDI 0205)
- *ARM PrimeXSys System Controller (SP810) Technical Reference Manual* (ARM DDI 0254)
- *ARM PrimeCell Vector Interrupt Controller (PL190) Technical Reference Manual* (ARM DDI 0181)
- *ARM PrimeCell Watchdog Controller (SP805) Technical Reference Manual* (ARM DDI 0270)
- *ARM PrimeCell UART (SP802) Technical Reference Manual* (ARM DDI 0239*)*
- *ETM9™ Technical Reference Manual* (ARM DDI 0157)
- *ARM926EJ-S™ Technical Reference Manual* (ARM DDI 0198)
- *ARM VFP9™ Coprocessor Technical Reference Manual* (ARM DDI 0238).

Other publications

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

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

The following datasheets describe some of the integrated circuits or modules used on the Versatile/AB926EJ-S:

- *CODEC with Sample Rate Conversion and 3D Sound* (LM4549) National Semiconductor, Santa Clara, CA.

- *MultiMedia Card Product Manual* SanDisk, Sunnyvale, CA.
- *Serial Microwire Bus EEPROM* (M93C86) STMicroelectronics, Amsterdam, The Netherlands.
- *StrataFlash Memory* (28F128J3A) Intel Corporation, Santa Clara, CA.
- *USB Host/Function/OTG controller* (OTG243) Transdimension Inc., Irvine, CA.
- *Three-In-One Fast Ethernet Controller* (LAN91C111) SMSC, Hauppauge, NY.

Feedback

ARM Limited welcomes feedback both on the Versatile/AB926EJ-S and on the documentation.

Feedback on the Versatile/AB926EJ-S

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

- the product name
- an explanation of your comments.

Feedback on this document

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

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

General suggestions for additions and improvements are also welcome.

Chapter 1
Introduction

This chapter introduces the Versatile/AB926EJ-S. It contains the following sections:

1.1 About the Versatile/AB926EJ-S

The Versatile/AB926EJ-S provides a development system that you can use to develop software applications for the ARM7 and ARM9 processor families.

You can use the platform as a basic development system with a power supply and a connection to a JTAG interface unit. (The Versatile/AB926EJ-S has an onboard USB debug interface and it can be used without a JTAG unit if software that supports USB debug is installed on a connected PC.)

You can expand the basic development system by adding:

- a VGA monitor
- an MMC or SD card
- a USB device (using the OTG connector)
- a keyboard and mouse
- audio devices to the onboard CODEC
- an Ethernet network to the onboard Ethernet controller.
- an interface board
- a memory expansion board
- a battery pack.

The Versatile/AB-IB1 (Application Baseboard Interface Board 1) and Versatile/AB-IB1 (Application Baseboard Interface Board 2) boards for the Versatile/AB926EJ-S enable you to connect:

- a static memory module (Versatile/AB-IB1 and Versatile/AB-IB2)
- CLCD expansion connector (AB-IB1) or quarter-VGA CLCD display (AB-IB2)
- devices to the GPIO connector (AB-IB1)
- serial devices to one of the two serial port connectors (AB-IB1)
- camera module (AB-IB2)
- keypad (AB-IB2)
- serial devices to the synchronous serial port connector (AB-IB1)
- GSM module (AB-IB2, not fitted as standard)
- Bluetooth module (AB-IB2, not fitted as standard).

The basic system provides a good platform for developing code for the ARM7 and ARM9 processor families. The ARM926PXP development chip used in the Versatile/AB926EJ-S is much faster than a software simulator or a core implemented in an FPGA. Code developed for the ARM926PXP development chip will also run on ARM10 and ARM11 processors.

Figure 1-1 on page 1-3 and Figure 1-2 on page 1-4 shows the layout of the Versatile/AB926EJ-S.

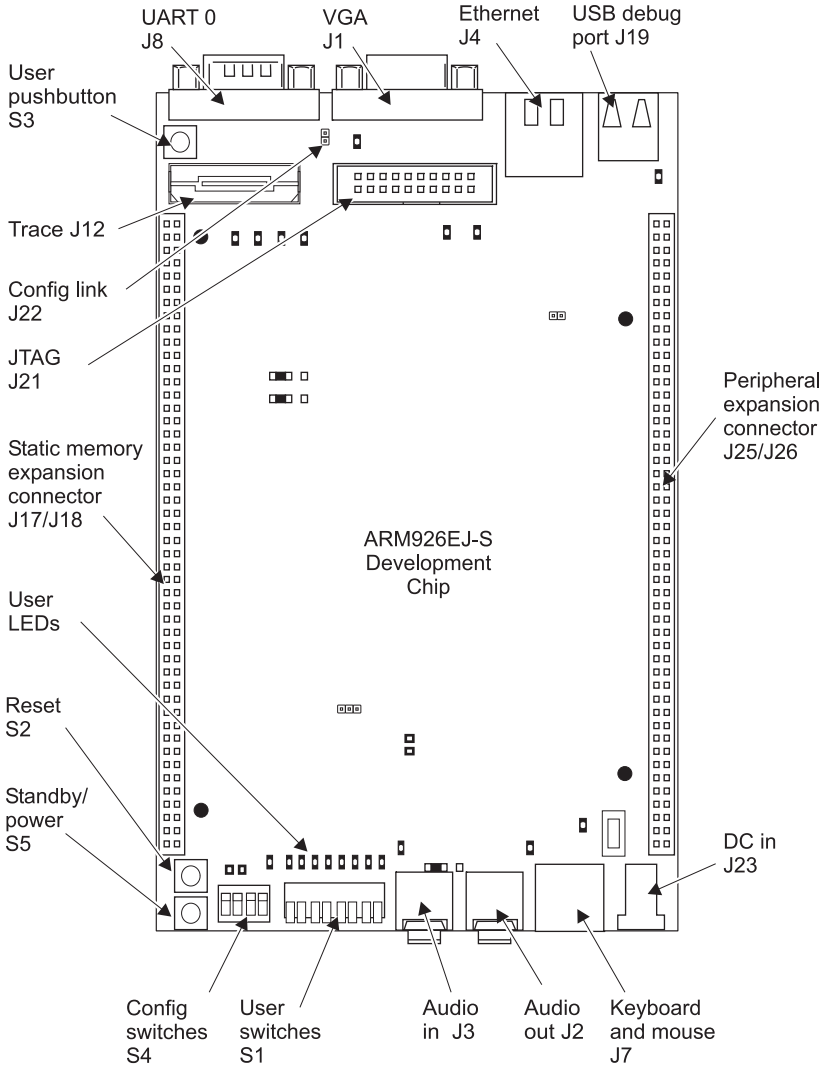

Figure 1-1 Versatile/AB926EJ-S layout (top)

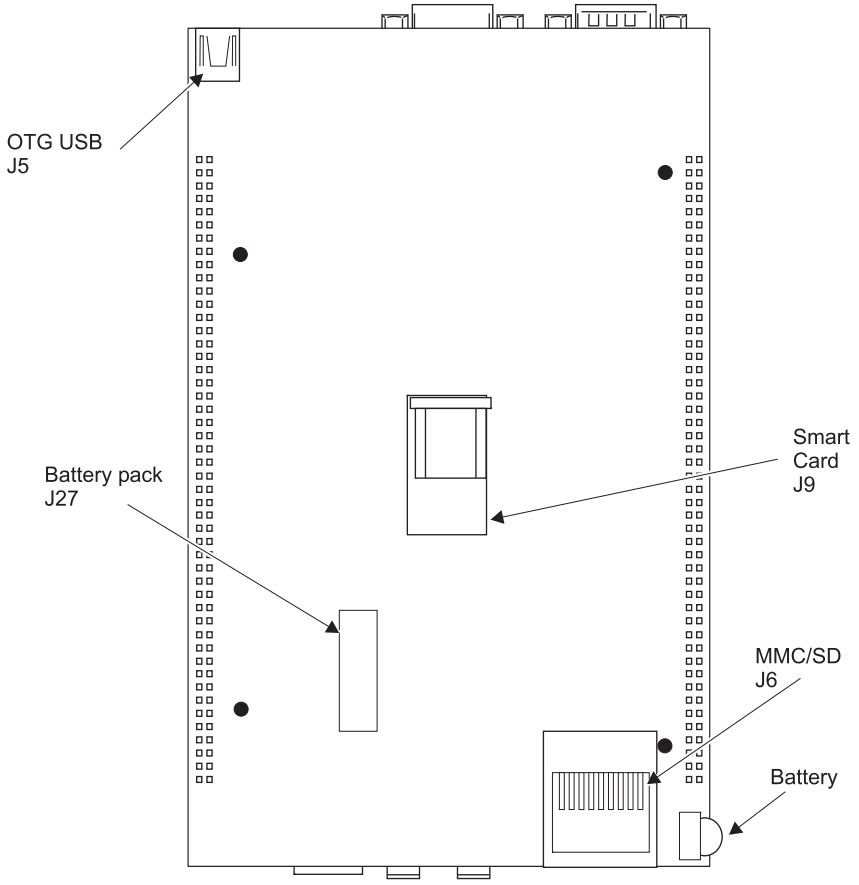

Figure 1-2 Versatile/AB926EJ-S layout (bottom)

1.2 Versatile/AB926EJ-S architecture

The major components on the platform are:

- ARM926PXP development chip equipped with:
  - ARM926EJ-S processor that supports 32-bit ARM and 16-bit Thumb instructions sets and includes features for direct execution of Java byte codes. Executing Java byte codes requires the *Java Technology Enabling Kit* (JTEK)
  - *Tightly-Coupled Memory* (TCM) for code (32KB) and data (32KB)
  - cache memory for code (32KB) and data (32KB)
  - *Memory Management Unit* (MMU)
  - Multi-layer bus matrix that enables simultaneous transfers from several bus masters.
  - MOVE™ Graphics coprocessor
  - MBX graphics accelerator
  - Multi-Port Memory Controller (MPMC) for direct connection to dynamic memory
  - Synchronous Static Memory Controller (SSMC) for direct connection to static memory
  - VFP9 Vector Floating Point coprocessor
  - AHB bridge that provides access to peripherals outside the development chip (such as those in the FPGA)
  - system controller
  - DMA controller
  - *Vectored Interrupt Controller* (VIC)
  - Color LCD controller
  - Three UARTs
  - *Synchronous Serial Port* (SSP)
  - *Smart Card Interface* (SCI)
  - Two eight-bit GPIOs (another GPIO is dedicated to board status signals)
  - *Real Time Clock* (RTC)
  - Two programmable timers
  - Watchdog timer
  - *Embedded Trace Macrocell* (ETM9)
  - Embedded-ICE logic for JTAG debugging
  - *Phase-Locked Loop* (PLL)
  - configuration block.
- *Field Programmable Gate-Array* (FPGA) that implements:
  - MMC/SD card MCI controller

  - — two KMI controllers for keyboard and mouse
  - — interface to onboard Ethernet controllers
  - — interface to onboard audio CODEC
  - — interface to onboard real-time clock
  - — interface to onboard audio CODEC
  - — interface to EEPROM on memory expansion boards
  - — registers for status, ID, onboard switches, LEDs, and clock control
  - — a secondary interrupt controller and external DMA control logic.
- 128MB of 32-bit wide SDRAM
- 2MB of 32-bit wide static RAM
- 64MB of 32-bit wide NOR flash
- 64MB of 16-bit wide NAND Disk-on-Chip flash
- programmable clock generators
- expansion connectors for UARTs
- expansion connectors on AB-IB1 for CLCD, UARTs, GPIO, and static memory
- debug and test connectors for JTAG and Trace port
- DIP switches and LEDs
- power conversion circuitry
- time of year clock with backup battery.

Figure 1-3 on page 1-7 shows the architecture of the Versatile/AB926EJ-S. Most of the peripherals are part of the ARM926PXP development chip. The internal peripherals and controllers are connected to a bus matrix that manage the bus routing. Additional peripherals are implemented in the FPGA. The FPGA connects to the AHB M2 bus on the ARM926PXP development chip.

Note

The Versatile/AB926EJ-S is a compact version of the Versatile/PB926EJ-S. Some of the functionality of the Versatile/PB926EJ-S is not present in the Versatile/AB926EJ-S and missing signals or registers are listed where relevant to simplify moving code between the two products. See the *Versatile Platform Baseboard for ARM926EJ-S User Guide* for more information on the Versatile/AB926EJ-S. See *Differences between the Versatile/PB926EJ-S and Versatile/AB926EJ-S* on page 4-9 for a summary of the differences in between the development boards.

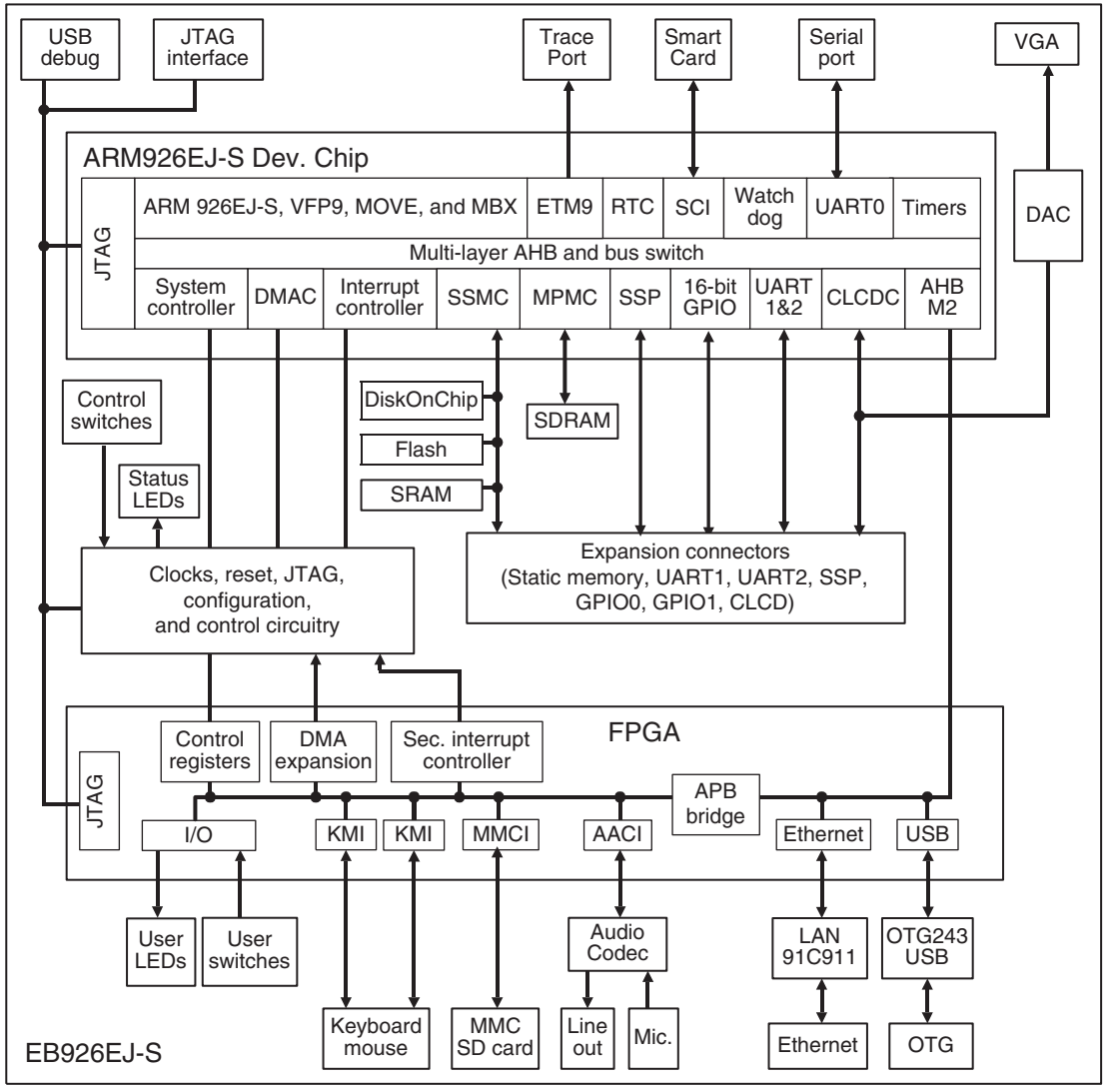

Figure 1-3 Versatile/AB926EJ-S block diagram

1.2.1 ARM926PXP development chip

For details on the ARM926PXP development chip, see *ARM926PXP development chip* on page 3-3 and the *ARM926PXP development chip Reference Manual*.

1.2.2 Versatile/AB926EJ-S FPGA

The FPGA provides system control and configuration functions for the Versatile/AB926EJ-S that enable it to operate as a standalone development system or with an interface board. See *FPGA* on page 3-14.

The FPGA also implements additional peripherals, for example the audio CODEC, USB, and Ethernet interfaces.

1.2.3 Displays

The CLCD signals from the ARM926PXP development chip are converted on the Versatile/AB926EJ-S to a VGA signal. The resolution of the VGA signal is configurable.

The CLCD signals are also output to the expansion connector.

1.2.4 Memory

The volatile memory system includes 2MB of SRAM and 128MB of SDRAM memory. The ARM926PXP development chip Tightly Coupled Memory (TCM) has 32KB code and 32KB data that operates with one wait state. The ARM926PXP development chip also has 32KB of code cache and 32KB of data cache.

The nonvolatile memory system consists of 64MB of 32-bit flash and 64MB of 16-bit NAND Disk-on-Chip flash. The flash is managed by the static memory controller in the ARM926PXP development chip.

Note

The expansion connectors J17 and J18 (static memory expansion) can be used to attach an interface board such as the Versatile/AB-IB1 or Versatile/AB-IB2. The interface boards accept static memory expansion modules as described in Appendix F *Static Memory Expansion Board*.

1.2.5 Clock generators

The Versatile/AB926EJ-S contains the following clock sources:

- crystal oscillators (these are the reference frequencies for the Real Time Clock, USB, AACI, Ethernet, and programmable oscillators)
- two programmable ICS307 clock sources that are used as the reference for the ARM926PXP development chip CPU system clock and for the CLCD controller clock.

See *Clock architecture* on page 3-29.

1.2.6 Debug and test interfaces

The JTAG connector enables JTAG hardware debugging equipment, such as Multi-ICE or RVI, to be connected to the Versatile/AB926EJ-S. The JTAG signals can also be controlled by the on-board USB debug port controller. See *JTAG and USB debug port support* on page 3-64.

A Mictor connector on the Versatile/AB926EJ-S enables monitoring of the ARM926PXP development chip *Embedded Trace Macrocell* (ETM9) signals by a *Trace Port Analyzer* (TPA). The trace port supports 4, 8, and 16-bit trace packet sizes. See *Trace connector pinout* on page A-22 for connection information.

1.3 Precautions

This section contains safety information and advice on how to avoid damage to the Versatile/AB926EJ-S.

1.3.1 Ensuring safety

The Versatile/AB926EJ-S is powered from the supplied power supply connected to J23.

Warning

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

Warning

The GSM module on the Versatile/AB-IB2 (see Appendix D *Versatile/AB-IB2 Interface Board*) emits radio frequency radiation. See the Enfora website at `www.enfora.com` for information on safe operation.

Keep the Versatile/AB-IB2 GSM module at least 20cm (7.9in) separation distance from the antenna and the human body.

The transmitter and antenna must not be collocated or operating in conjunction with any other antenna or transmitter. Failure to observe this warning could produce an RF exposure condition.

1.3.2 Preventing damage

The Versatile/AB926EJ-S is intended for use in a laboratory or engineering development environment. If operated without an enclosure, the board is sensitive to electrostatic discharges and generates electromagnetic emissions.

Caution

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

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

Caution

Do not use the board near equipment that is:

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

Chapter 2
Getting Started

This chapter describes how to set up and prepare the Versatile/AB926EJ-S for use. It contains the following sections:

2.1 Setting up the Versatile/AB926EJ-S

The following items are supplied with the Versatile/AB926EJ-S:

- the Versatile/AB926EJ-S printed-circuit board mounted in an enclosure
- a 12V power supply
- a CD containing sample programs, Boot Monitor code, FPGA and PLD images, and additional documentation
- this user guide.

To set up the Versatile/AB926EJ-S as a standalone development system:

1. Set the configuration switches to select the boot memory location, JTAG access mode, and FPGA image. See *Selecting the boot memory type* on page 2-3.
2. If you are using an external display, connect the cable from the display to the VGA connector on the Versatile/AB926EJ-S.
3. If you are using a *Trace Port Analyzer* (TPA), connect the Trace Port interface board to the Versatile/AB926EJ-S. See *Connecting the Trace Port Analyzer* on page 2-7.
4. If you are using a debugger, connect to the JTAG or USB debug port on the board. See *Connecting JTAG debugging equipment* on page 2-5.
5. If you are using an interface board, install it on the expansion connectors (J17/J18 and J25/J26). See Appendix C *Versatile/AB-IB1 Interface Board*, Appendix D *Versatile/AB-IB2 Interface Board*, and the documentation supplied with your interface board for more details.
6. Apply power to the Versatile/AB926EJ-S. See *Supplying power* on page 2-10.
7. If you are using the supplied Boot Monitor software to select and run an application, see *Using the Versatile/AB926EJ-S Boot Monitor* on page 2-11.

2.2 Selecting the boot memory type

The configuration switches S4-1 to S4-4 shown in Figure 2-1 determine boot memory type and the FPGA image as shown in Figure 2-1 and Table 2-1 on page 2-4.

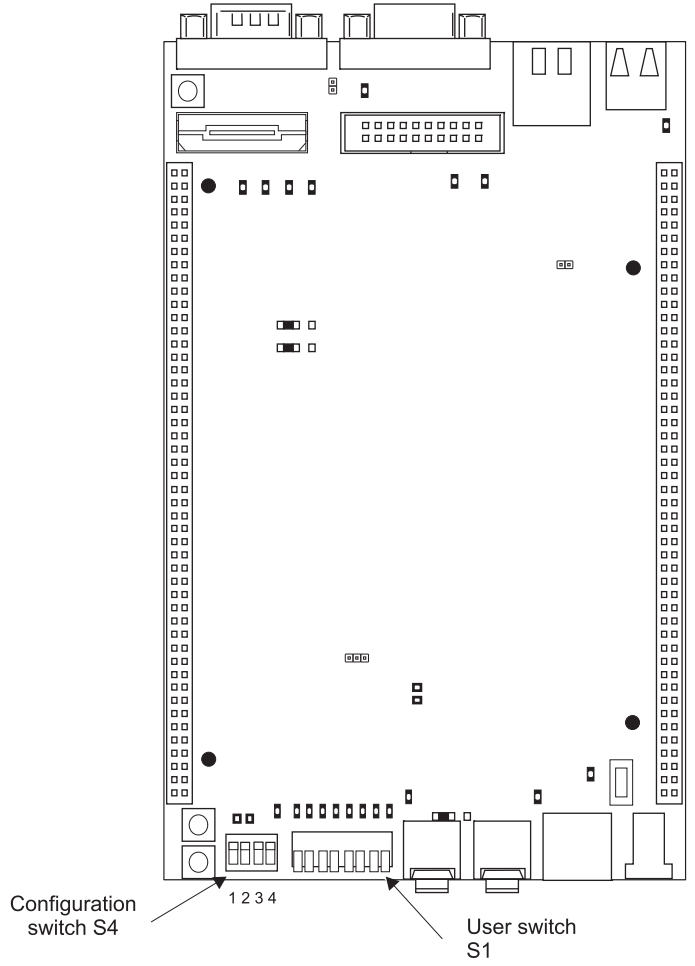

Figure 2-1 Location of configuration switches

Table 2-1 Selecting the boot device

S4-2	S4-1	Device
OFF	OFF	Disk on Chip, see *Booting from Disk on Chip* on page 4-15 (default position)
OFF	ON	NOR flash, see *Booting from NOR flash* on page 4-16
ON	OFF	Reserved
ON	ON	Reserved

Table 2-2 Selecting the FPGA image

S4-3	Device
OFF	FPGA image 0. This is the default image and is supplied with the board. Set both switches OFF for normal operation. **Note** Only one image is supplied with the Versatile/AB926EJ-S.
ON	Reserved for future use.

Note

Configuration switches S4-2 to S4-4 are not normally changed from their factory default position. For more information, see *Configuration control* on page 3-7.

Switch S4-4 is not used and should be left in the OFF position.

For information on the Boot Monitor selection switch S1, see *Using the Versatile/AB926EJ-S Boot Monitor* on page 2-11 and *Remapping of boot memory* on page 4-13.

For information on other configuration links and connectors and the function of the status LEDs, see *Test, configuration, and debug interfaces* on page 3-64.

2.3 Connecting JTAG debugging equipment

You can use JTAG debugging equipment and the JTAG connector, or the USB debug port, to:

- connect a debugger to the ARM926EJ-S core and download programs to memory and debug them
- program new configuration images into the configuration flash, FPGA, and PLDs on the board. (You cannot program the normal flash from configuration mode.)

The setup for using a JTAG interface with the Versatile/AB926EJ-S is shown in Figure 2-2.

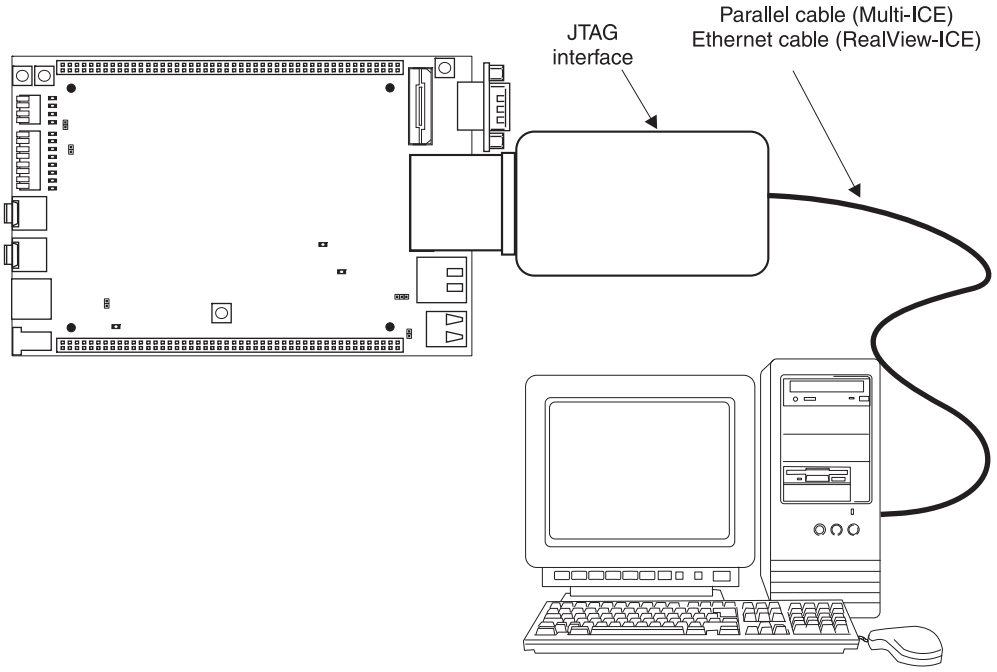

Figure 2-2 JTAG connection

Note

The high-density cable from the RealView ICE box requires a buffer board (the RVI LVDS probe) to connect to the JTAG connector on the Versatile/AB926EJ-S. The low-density cable can be used to connect the RealView ICE box directly to the JTAG

connector, but this interface operates at lower speed. The low-density JTAG cable from the RealView-ICE box can plug into either the target buffer board or the JTAG connector on the Versatile/AB926EJ-S.

The setup for using the USB debug port on the Versatile/AB926EJ-S is shown in Figure 2-3. The Versatile/AB926EJ-S contains logic that interfaces the USB debug port and the onboard JTAG signals.

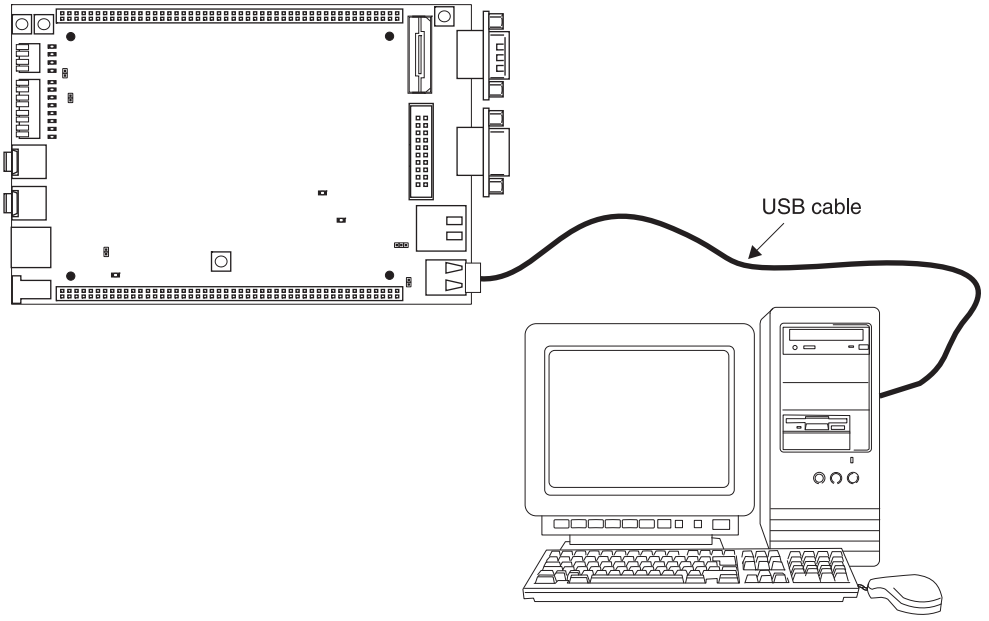

Figure 2-3 USB debug port connection

Note

For more details on JTAG debugging and selection between the JTAG and USB debug connector, see *JTAG and USB debug port support* on page 3-64. Refer to the documentation supplied with your debugger for details on configuring and using the USB debug port. If you are using the ARM RealView® Debugger, see Appendix G *Configuring a USB Debug Connection* for installation and configuration details.

2.4 Connecting the Trace Port Analyzer

The ARM926PXP development chip incorporates an *ARM9 Embedded Trace Macrocell* (ETM9). This enables you to carry out real-time debugging by connecting external trace equipment to the Versatile/AB926EJ-S. The ETM9 monitors the program execution and sends a compressed trace to the *Trace Port Analyzer* (TPA). The TPA buffers this information and transmits it to the debugger where it is decompressed and used to reconstruct the complete instruction flow. The trace size is 16-bit packets.

For MultiTrace, connect the TPA to the buffer board and plug the buffer board into the Versatile/AB926EJ-S as shown in Figure 2-4. MultiTrace requires a Multi-ICE JTAG unit.

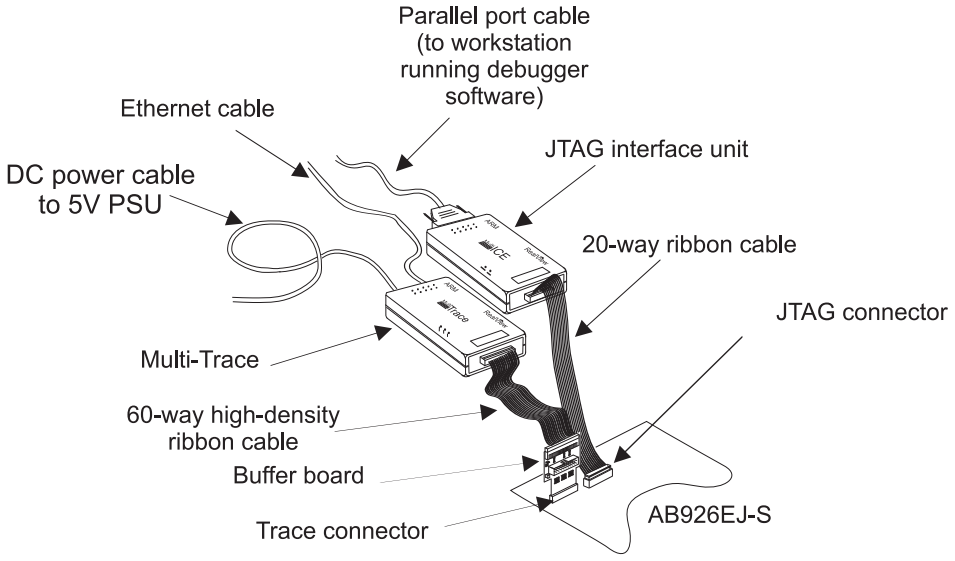

Figure 2-4 Example of MultiTrace and JTAG connection

For RealView Trace, connect the *Trace Port Analyzer* (TPA) to the adaptor board and plug the adaptor into the Versatile/AB926EJ-S as shown in Figure 2-4. RealView Trace requires a RealView-ICE JTAG unit. The Ethernet and power supply cables connect to the RealView-ICE unit.

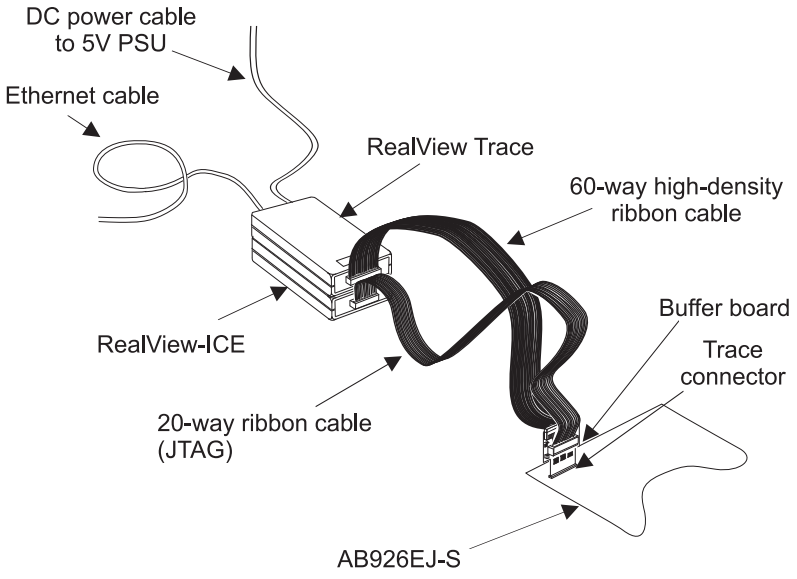

Figure 2-5 Example of RealView-ICE and RealView Trace

2.4.1 About using trace

The components used for trace capture are:

ETM The Embedded Trace Macrocell is part of the ARM926PXP development chip. It monitors the ARM core buses and outputs compressed information through the trace port to a trace connector. The on-chip ETM contains trigger and filter logic to control what is traced.

Trace connector and adaptor board

The trace connector enables you to connect a TPA to the Versatile/AB926EJ-S. The connector is a high-density AMP Mictor connector. The pinout for this connector is provided in *Test and debug connections* on page A-18.

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

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

Trace Port Analyzer

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

Debugger and Trace software

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

Note

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

ARM eXtended Debugger (AXD)

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

ARM RealView Debugger (RVD)

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

2.5 Supplying power

Connect the supplied brick power supply to power socket J23 as shown in Figure 2-6.

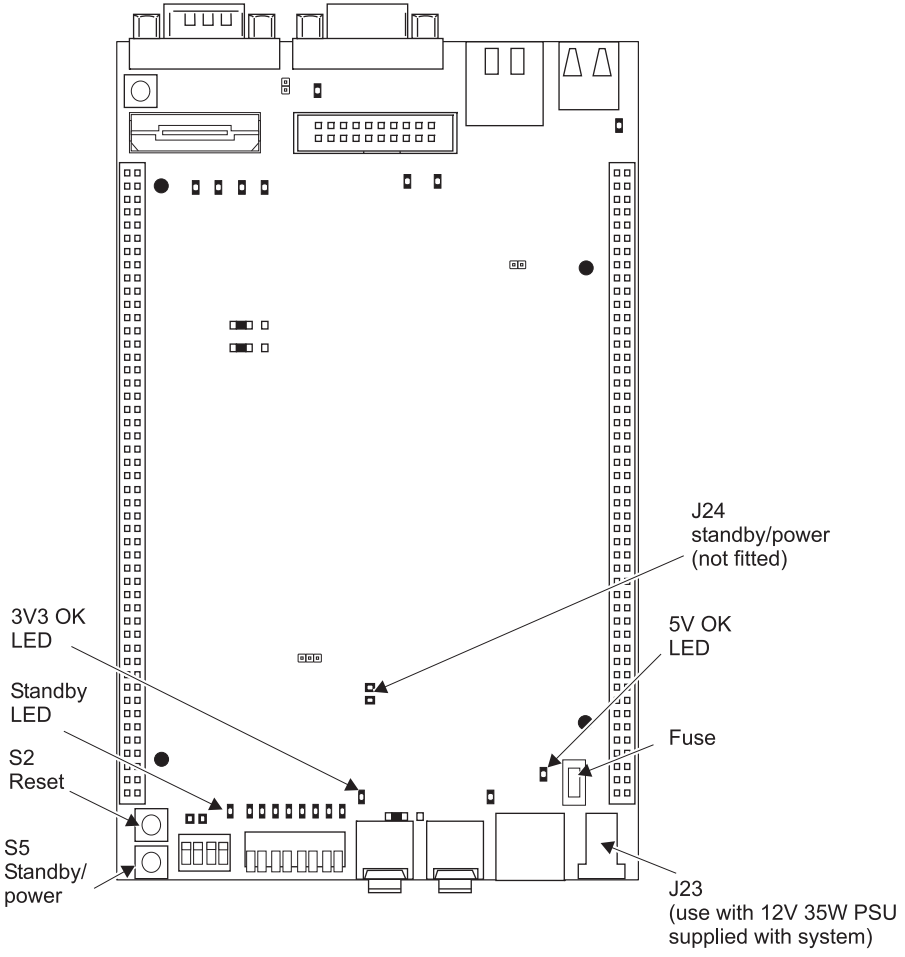

Figure 2-6 Power connector J23

The Standby/power pushbutton toggles the power on and off. Use connector J24 to attach a remote standby/power switch.

2.6 Using the Versatile/AB926EJ-S Boot Monitor

The Versatile/AB926EJ-S Boot Monitor is a collection of tools and utilities designed as an aid to developing applications on the Versatile/AB926EJ-S.

When the Boot Monitor starts on reset, the following actions are performed:

- clock dividers are loaded with appropriate values
- the memory controllers are initialized
- a stack is set up in memory
- Boot Monitor code is copied into SDRAM
- C library I/O routines are remapped and redirected
- the current bootscript, if any, is run.

The behavior at system reset is determined by S1-1:

S1-1 OFF A prompt is displayed enabling you to enter Boot Monitor commands.

S1-1 ON The Boot Monitor executes a boot script that was loaded into flash.

The boot script can execute any Boot Monitor commands. It typically selects and runs an image in application flash. You can store one or more code images in flash memory and use the boot script to start an image at reset. Use the `BOOTSCRIPT SET` command to enter a boot script from the Boot Monitor (see Table 2-3 on page 2-12).

Output of text from STDIO for both applications and Boot Monitor I/O depends on the setting of S1-3:

S1-3 ON STDIO is redirected to UART0. This occurs even under semihosting.

S1-3 OFF STDIO autodetects whether to use semihosting I/O or UART. If a debugger is connected and semihosting is enabled, STDIO is redirected to the debugger console window. Otherwise, STDIO goes to the UART.

S1-3 does not affect file I/O operations performed under semihosting. Semihosting operation requires a debugger and a JTAG interface device. See *Redirecting character output to hardware devices* on page 2-20 for more details on I/O.

Note

S1-2 is reserved for future use.

2.6.1 Running the Boot Monitor

To run Boot Monitor and have it display a prompt to a terminal connected to UART0, set switch S1-3 to ON and S4-1 to OFF, and reset the system. Standard input and output functions use UART0 by default. The default setting for UART0 is 38400 baud, 8 data bits, no parity, 1 stop bit. There is no hardware or software flow control.

Note

If the Boot Monitor has been accidently deleted from flash memory, it can be rebuilt and reloaded. See *Rebuilding the Boot Monitor* on page 2-16.

Boot Monitor commands

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

Note

Commands are accepted in uppercase or lowercase. The Boot Monitor also accepts abbreviations of commands if the meaning is not ambiguous. For example, for `QUIT`, you can type `QUIT`, `QUI`, `QU`, `Q`, `quit`, `qui`, `qu`, or `q`.

Copying images and files to a local directory that has a short path name (such as `C:\debug` for example) simplifies entering pathnames at the Boot Monitor command line.

Table 2-3 lists the commands for the Boot Monitor.

Table 2-3 Boot Monitor commands

Command	Action
`@ script_file`	Runs a script file.
`ALIAS alias commands`	Create an alias command *alias* for the string of commands contained in *commands*.
`CLEAR BOOTSCRIPT`	Clear the current boot script. The Boot Monitor will prompt for input on reset even if the S1-1 is set to ON to indicate that a boot script should be run.
`CONFIGURE`	Enter Configure subsystem. Commands listed in Table 2-4 on page 2-14 can now be executed.
`CONVERT BINARY binary_file LOAD_ADDRESS address [ENTRY_POINT address]`	Provides information to the system that is required by the RUN command in order to execute a binary file. A new file with name *binary_file* is produced, but with an `.exe` file extension.

Table 2-3 Boot Monitor commands (continued)

Command	Action
COPY *file1 file2*	Copy *file1* to *file2*. For example, to copy the `leds` code from the PC to the Disk-on-Chip enter: `COPY C:\software\projects\examples\rvds2.0\leds.axf leds.axf` **Note** Remote file access requires semihosting. Use a debugger connection to provide semihosting.
CREATE *filename*	Create a new file in the Disk-on-Chip by inputting text. Press Ctrl-Z to end the file.
DEBUG	Enter the debug subsystem. Commands listed in Table 2-6 on page 2-16 can now be executed.
DELETE *filename*	Delete *file* from Disk-on-Chip.
DIRECTORY [*directory*]	List the files in a Disk-on-Chip directory. Files only accessible from semihosting cannot be listed.
DISABLE CACHES	Disable both the I and D caches.
DISPLAY BOOTSCRIPT	Display the current boot script.
ECHO *text*	Echo *text* to the current output device.
ENABLE CACHES	Enable both the I and D caches.
EXIT	Exit the Boot Monitor. The processor is held in a tight loop until it is interrupted by a JTAG debugger.
FLASH	Enter the flash file system for the NOR flash on the Versatile/PB926EJ-S. See Table 2-5 on page 2-14 for flash commands.
HELP	List the Boot Monitor commands.
LOAD *name*	Load the Disk-on-Chip image *name* into memory and run it.
QUIT	Alias for `EXIT`. Exit the Boot Monitor.
RENAME *old_name new_name*	Rename Disk-on-Chip file named *old_name* to *new_name*.
RUN *image_name*	Load the Disk-on-Chip image *image_name* into memory and run it.
SET BOOTSCRIPT *script_file*	Specify *script_file* as the boot script. If the run boot script switch S1-1 is ON, *script_file* will be run at system reset.
TYPE *filename*	Display the Disk-on-Chip file *filename*.

Table 2-4 lists the commands for the configure subsystem.

Table 2-4 Boot Monitor Configure commands

Command	Action
`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`	List the configure commands.
`QUIT`	Alias for `EXIT`. Exit the Configure commands and return to standard Boot Monitor commands.
`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 2-5 lists the commands for the NOR flash subsystem.

Table 2-5 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. **Warning** This command can erase the Boot Monitor image if it is stored in NOR flash. See *Loading Boot Monitor into NOR flash* on page 2-17.
`EXIT`	Exit the flash commands and return to executing standard Boot Monitor commands.
`HELP`	List the flash commands.
`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.

Table 2-5 Boot Monitor NOR flash commands (continued)

Command	Action
`LOAD name`	Load the image *image_name* into memory.
`QUIT`	Alias for `EXIT`. Exit the NOR flash commands and return to standard Boot Monitor commands.
`RESERVE SPACE address size`	Reserve space in NOR flash. This space will not be used by the Boot Monitor. *address* is the start of the area and *size* is the size of the reserved area.
`RUN name`	Load the image *name* from flash and run it.
`UNRESERVE SPACE address`	Free the space starting at *address* in NOR flash. This space can be used by the Boot Monitor.
`WRITE BINARY file [NAME new_name] [FLASH_ADDRESS address] [LOAD_ADDRESS address] [ENTRY_POINT address]`	Write a binary file to flash. By default, the image is identified by its file name. Use `NAME new_name` to specify a name instead of using the default name. Use `FLASH_ADDRESS address` to specify where in flash the image is to be located. The optional `LOAD_ADDRESS` and `ENTRY_POINT` arguments enable you to specify the load address and the entry point. If an entry point is not specified, the load address is used as the entry point. **Note** Remote file access requires semihosting. Use a debugger connection to provide semihosting.
`WRITE IMAGE file [NAME new_name] [FLASH_ADDRESS address]`	Write an ELF image file to flash. By default, the image is identified by its file name. For example, t:\images\boot_monitor.axf is identified as boot_monitor. Use `NAME new_name` to specify a name instead of using the default name. Use `FLASH_ADDRESS address` to specify where in flash the image is to be located. If the image is linked to run from flash, the link address is used and *address* is ignored. **Note** Remote file access requires semihosting. Use a debugger connection to provide semihosting.

Table 2-6 lists the commands for the Debug subsystem.

Table 2-6 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	List the debug commands.
QUIT	Alias for EXIT. Exit the Debug commands and return to standard Boot Monitor commands.
START TIMER	Start a timer.
STOP TIMER	Stop the timer started with the START TIMER command and display the elapsed time.

2.6.2 Rebuilding the Boot Monitor

All firmware components are built using GNUmake, which is available for UNIX, Linux and for most Windows versions. To use GNUmake under windows Cygwin must be installed, for more information contact Redhat.

Because the platform library used by the Boot Monitor requires callout startup routines support specific to RVCT, the Boot Monitor (and any application that uses the platform library for directing STDIO) can only be rebuilt using RVCT tools.

To rebuild the Boot Monitor, set your default directory to *install_directory*/Firmware/Boot_Monitor and type make from a DOS command line.

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)
- DISKONCHIP=1/0, defines Disk-on-Chip support (Default 1, Disk-on-Chip support)

Note

The image must be build 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_NoDiskOnChip`.

After rebuilding the Boot Monitor, load it into either NOR flash or the NAND flash Disk-on-Chip, see *Loading Boot Monitor into NOR flash* and *Loading Boot Monitor into Disk-on-Chip* on page 2-18.

2.6.3 Loading Boot Monitor into NOR flash

If the flash becomes corrupt and the board no longer runs the Boot Monitor, the Boot Monitor must be reprogrammed into flash.

Note

The Boot Monitor is normally located in Disk-on-Chip flash instead of NOR flash. You can, however, load the Boot Monitor into NOR flash instead of Disk-on-Chip flash if this is requiredfor a specific application.

Because the debugger does not initialize SDRAM, the Boot Monitor image cannot be loaded and run directly. Use the scripts in the `BoardFiles` directory on the CD to setup the board:

1. Power off the board
2. Set switch S1-1 to ON to select booting from NOR flash

   Set all other S1 switches to OFF

   Set all S4 switches to OFF.
3. Connect a debug cable to either the JTAG or USB debug port.
4. Power on the board.
5. Connect the debugger to the target
   - For ARM eXtended Debugger, from the Command Line Interface enter:

     `Debug > Obey` *`path`* `VPB926ejs_SDRAM_Init_axd.li`
   - For RealView Debugger, select:from **Debug menu → Include Commands From File**

select **VPB926EJS_SDRAM_Init_rvd.li**

6. SDRAM 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.axf
      ```
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.

2.6.4 Loading Boot Monitor into Disk-on-Chip

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

1. Power off the board
2. Set all S1 and S4 switches to OFF.
3. Connect a debug cable to either the JTAG or USB debug port.
4. Power on the board.
5. Connect the debugger to the target
   - For ARM eXtended Debugger, from the Command Line Interface enter:
     `Debug > Obey path VPB926EJS_SDRAM_Init_axd.li`
   - For RealView Debugger:from the **Debug menu → Include Commands From File**
     select **VPB926EJS_SDRAM_Init_rvd.li**

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

   (See *Using the Disk-on-Chip configure utility program* on page 2-19 for more details on the configure utility.)
7. If required, use the utility to format the Disk-on-Chip. At the prompt enter:
   `Config> FORMAT`

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

Caution

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

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

   `Config> WRITE IPL path doc_ipl.axf`

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

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

   `Config> WRITE SPL path doc_spl.axf`

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

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

    `Config> WRITE BOOT path boot_monitor.axf`

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

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

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

    `Config> LIST`

13. Turn the board off and then on.

2.6.5 Using the Disk-on-Chip configure utility program

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

Table 2-7 lists the commands for the Configure utility.

Table 2-7 Configure utility commands

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

2.6.6 Redirecting character output to hardware devices

The redirection of character I/O is carried out within the Boot Monitor platform library routines in `retarget.c` and `boot.s`. During startup, the platform library executes a *SoftWare Interrupt* instruction (SWI). If the image is being executed without a debugger (or the debugger is not capturing semihosting calls) the value returned by this SWI is –1, otherwise the value returned is positive. The platform library uses the return value to determine how to redirect operations on the requested I/O channel. (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.`

The STDIO calls are redirected within `retarget.c`. Redirection depends on the semihosted flag set during image startup and the state of switch S1.

2.6.7 Rebuilding the platform library

All firmware components are built using GNUmake, which is available for UNIX, Linux and for most Windows versions. (To use GNUmake under windows Cygwin must be installed, for more information contact Redhat.)

To rebuild the platform library component, set your default directory to *install_directory*/Firmware/platform and type make from a DOS command line.

The platform library has a number of build options that can be specified with 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)
- VFP=1/0, defines VFP support (Default 0 no VFP support)
- DISKONCHIP=1/0, defines Disk-on-Chip support (Default 1, Disk-on-Chip support)

The build options define the directory that contains the compile and link output. The make file creates a directory called Builds if it is not already present. The Builds directory contains subdirectories for the specified make options (for example, Debug_ARM_Little_Endian). To delete the objects and images for all targets and delete the Builds directory, type make clean all.

2.6.8 Building an application with the platform library

The platform library on the CD provides all required initialization code to bring the Versatile/PB926EJ-S 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.

The platform library supports:

- remapping of boot memory
- SDRAM initialization
- UARTs
- Time-of-Year clock
- 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.

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.

   Define the image entry point to be `__main` and the region `__main` to be the first section in the execution region as the armlink command option:

   ```
   -entry __main -first __main
   ```

   Note

   If you are not using the `platform.a` library, you must provide your own initialization and I/O routines.

   See the `readme.txt` file in the `software` directory for a description of the contents of the CD. The `selftest` directory, for example, contains source files that can be used as a starting point for your own application.

To run the image from RAM, load the image with a debugger and execute as normal. The image uses the procedure described in *Redirecting character output to hardware devices* on page 2-20 to redirect standard I/O either to the debugger or to be handled by the application itself.

2.6.9 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 2-21 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 Boot Monitor 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 Boot Monitor `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 SDRAM for execution, the execution address must not be in the top 4MBytes of SDRAM since this is used by the Boot Monitor.

2. The image must be programmed into flash using the Boot Monitor. Flash support is implemented in the Boot Monitor image.

   Run the Boot Monitor image from the debugger and enter the flash subsystem, type `FLASH` at the prompt:

   ```
   >FLASH
   flash>
   ```

3. The command used to program the image depends on the type of image:

   - 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 image itself.

   - 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
     flash>
     ```

Note

`name` is a short name for the image. If the `NAME` option is not used at the command prompt, *`name`* will be derived from the file name.

4. The image is now in flash and can be run by the Boot Monitor. At the prompt, type:

   `flash> RUN` *`name`*

2.6.10 Running an image from Disk-on-Chip

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

1. Build and link the application as described in *Loading and running an application from NOR flash* on page 2-22.

   Note

   Images with multiple load regions are not supported.

   The image must have an execution region in RAM or SDRAM.

   The execution address must not be in the top 4MBytes of SDRAM since this is used by the Boot Monitor.

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

   Connect a debugger and use semihosting to load the file into NAND flash. For example, to copy an ELF file from your `software` directory on the `C:` drive to the Disk-on-Chip, enter:

   `>COPY C:\software\`*`elf_file_name file_name`*

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

   `>RUN `*`file_name`*

2.6.11 Using a boot script to run an image automatically

Use a boot script to run an image automatically after power-on:

1. Create a boot script from the Boot Monitor by typing:
   - If your image is in NAND flash:

     `> CREATE myscript.txt ; put any startup code here  RUN file_name`
   - If your image is in NOR flash, enter the flash subsystem before running:

     `> CREATE myscript.txt ; put any startup code here  FLASH RUN file_name`

2. Press **Ctrl-Z** to indicate the end of the boot script and return to the Boot Monitor prompt.

3. Verify the file was entered correctly by typing:

   `>TYPE myscript.txt`

   The contents of the file is displayed to the currently selected output device.

4. Specify the boot script to use at reset from the Boot Monitor by typing:

```
>SET BOOTSCRIPT myscript.txt
```

5. Set S1-1 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.

Chapter 3
Hardware Description

This chapter describes the Versatile/AB926EJ-S on-board hardware. It contains the following sections:

- *Smart card interface, SCI* on page 3-55
- *Synchronous Serial Port, SSP* on page 3-57
- *User switches and LEDs* on page 3-59
- *USB interface* on page 3-60
- *UART interface* on page 3-62
- *Test, configuration, and debug interfaces* on page 3-64.

Note

The Versatile/AB926EJ-S accepts interface boards (for example, the Versatile/AB-IB1 and Versatile/AB-IB2) that connect Versatile/AB926EJ-S memory and peripheral signals to expansion devices. For more details on the interface boards and their signals, see Appendix C *Versatile/AB-IB1 Interface Board*, Appendix D *Versatile/AB-IB2 Interface Board*, and *Peripheral expansion connector* on page A-4 and *Static memory expansion connector* on page A-7.

3.1 ARM926PXP development chip

The ARM926PXP development chip and its interfaces are described in the following sections:

- *ARM926PXP development chip overview*
- *Configuration control* on page 3-7
- *AHB bridges and the bus matrix* on page 3-8
- *Memory interface* on page 3-13
- *Reset controller* on page 3-18
- *GPIO interface* on page 3-48
- *CLCDC interface* on page 3-40
- *DMA* on page 3-43
- *UART interface* on page 3-62
- *Smart card interface, SCI* on page 3-55
- *Synchronous Serial Port, SSP* on page 3-57.

For more detail on using the ARM926PXP development chip components, see also:

- the *ARM926EJ-S Development Chip Reference Manual*
- Chapter 4 *Programmer's Reference*
- *ARM926PXP development chip clocks* on page 3-31.

3.1.1 ARM926PXP development chip overview

Figure 3-1 on page 3-4 shows the main blocks of the ARM926PXP development chip.

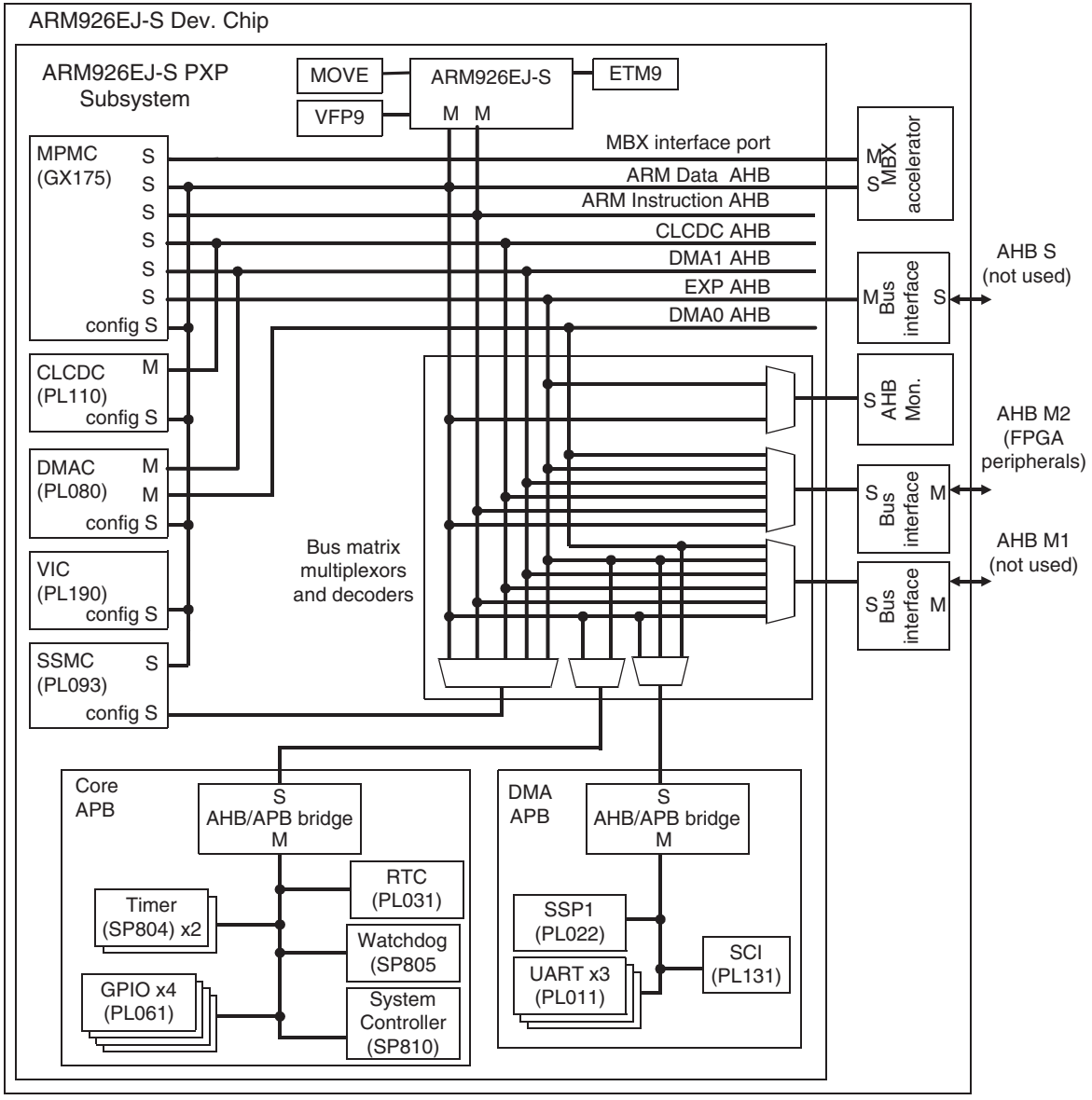

Figure 3-1 ARM926PXP development chip block diagram

The ARM926PXP development chip incorporates the following features:

ARM926EJ-S

The ARM926EJ-S CPU is a member of the ARM9 Thumb® family. The ARM926EJ-S (r0p3) macrocell is a 32-bit cached processor with ARMv5TE architecture that supports the ARM and Thumb instruction sets and includes features for direct execution of Java byte codes. Executing Java byte codes requires the *Java Technology Enabling Kit* (JTEK).

The ARM926EJ-S contains a *Memory Management Unit* (MMU), 32KB data and instruction caches, and 32KB of data and instruction *Tightly Coupled Memory* (TCM). The TCM operates with a single wait-state and provides higher data rates than external memory.

AHB buses and bus matrix

The ARM926EJ-S processor uses two separate AHB masters for instructions and data to maximize system speed. The DMA controller has two AHB masters. The CLCD controller has one AHB master.

There are also two expansion master buses (AHB M1 and AHB M2) and one expansion slave bus (AHB S). The expansion bus bridges are configurable to support different performance and complexity trade-offs.

The bus matrix inside the ARM926PXP development chip manages the multiple paths between each master and the peripherals and memory. The matrix enables different bus masters to access peripherals.

ETM9 The *Embedded Trace Macrocell* (ETM) provides signals for off-chip trace. The ETM transmits a 4, 8, or 16-bit packet to an external trace port analyzer where the signals can be stored and later analyzed to reconstruct the code flow.

VFP9 This high-performance, low-power *Vector Floating-Point* (VFP) coprocessor implements the VFPv2 vector floating-point architecture.

MOVE The MOVE coprocessor is a video encoding accelerator designed to accelerate *Motion Estimation* (ME) algorithms within block-based video encoding schemes such as MPEG4 and H.263. For more information on the MOVE coprocessor, see the *ARM MOVE Coprocessor Technical Reference Manual*. (The MOVE documentation is only available to licensees of this product.)

MBX This high-performance graphic accelerator operates on 3D scene data (as batches of triangles) sent from the main processor. Triangles are written directly to a tile accelerator so that the CPU is not stalled during

processing. For more information on the MBX coprocessor, see the *ARM MBX HR-S Graphics Core Technical Reference Manual*. (The MBX documentation is only available to licences of this product.)

Clock control

The ARM926PXP development chip contains deskew PLL that uses an external reference clock to generate internal clocks for the CPU, AHB bus, memory, and off-chip peripherals. Dividers in the chip are programmable and give considerable flexibility in clock rates for the CPU, bridges, and memory.

Memory controllers

The ARM926PXP development chip includes a multi-port memory controller (for dynamic memory) and a static memory controller. Both controllers have 32-bit interfaces to external memory. See *Memory interface* on page 3-13.

DMA controller

The PrimeCell DMAC enables peripheral-to-memory, memory-to-peripheral, peripheral-to-peripheral, and memory-to-memory transactions. See *DMA* on page 3-43.

Interrupt controller

The PrimeCell VIC provides an interface to the interrupt system and provides vectored interrupt support for high-priority interrupt sources from:

- peripherals in the ARM926PXP development chip
- peripherals in the FPGA (a secondary interrupt controller is present in the FPGA).

See *Interrupts* on page 3-49.

CLCD controller

The CLCDC provides a flexible display interface that supports a VGA monitor and color LCD displays. See *CLCDC interface* on page 3-40.

UARTs The UARTs perform serial-to-parallel conversion on data received from a peripheral device and parallel-to-serial conversion on data transmitted to the peripheral device. See *UART interface* on page 3-62.

Timer/counters

There are four 32-bit down counters that can be used to generate interrupts at programmable intervals. A Watchdog module can be used to trigger a system reset in the event of software failure. A Real-Time-Clock is fed with an external 1Hz signal.

Synchronous serial port

The SSP provides a master or slave interface for synchronous serial communications using Motorola SPI, TI, or National Semiconductor Microwire devices.

Smart Card interface

The Smart Card interface signals are programmable to enable support for a Smart Card, *Security Identity Module* (SIM) card, or similar module.

Watchdog A Watchdog module can be used to trigger an interrupt or system reset in the event of software failure.

3.1.2 Configuration control

The Versatile/AB926EJ-S uses configuration switches and the SYS_CFGDATAx registers in the FPGA to control configuration of the ARM926PXP development chip at power-up.

After reset, configuration can be modified by the system controller and the configuration registers in the FPGA. See *Status and system control registers* on page 4-19.

Note

In a production ASIC, the configuration signals are tied permanently HIGH or LOW. For the ARM926PXP development chip however, these signals can be modified by registers or configuration switches. This enables you to emulate different build options before finishing the design for the final product. The default configuration options are suitable for most application software development.

Configuration switches

The S4 boot option select switches are listed in Table 3-1. For more information on setting boot memory options, see *Selecting the boot memory type* on page 2-3 and *Configuration and initialization* on page 4-13. Switch S4 values determine the **BOOTCSSEL[3:0]** signals.

Table 3-1 Configuration switch S4

Switch	Description
S4-1 and S4-2	Controls the chip select signals for the static memory, see also *Selecting the boot memory type* on page 2-3.
S4-3	Selects the FPGA images to load on power up (see Table 2-2 on page 2-4)
S4-4	Reserved (leave in OFF position)

3.1.3 AHB bridges and the bus matrix

The ARM926PXP development chip is based on the ARM926EJ-S PrimeXSys Platform. The PrimeXSys Platform contains a multi-layer AHB bus matrix that routes the signals from the masters to a number of slaves. The masters are:

CPU-D ARM926PXP development chip data interface

CPU-I ARM926PXP development chip instruction interface

DMA-0 DMA port 0

DMA-1 DMA port 1

CLCD Color LCD controller

MBX The MBX graphic coprocessor

Expansion master

AHB bridge S on the ARM926PXP development chip. (This is not used on the Versatile/AB926EJ-S).

The internal slaves are:

MPMC Dynamic memory controller configuration bus and six interface buses

SSMC Static memory controller configuration and interface bus

DMAC The configuration bus for the DMA controller

VIC The configuration bus for the interrupt controller

MBX The configuration bus for the MBX graphic coprocessor

AHB-APB bridges

AHB bridges to the peripheral buses inside the ARM926PXP development chip

Expansion slaves

External slaves are connected to the AHBM2 port. The AHB M1 port is not used on the Versatile/AB926EJ-S.

See Figure 3-1 on page 3-4 for an overview of the bus interconnections.

Simultaneous access

Figure 3-2 on page 3-10 shows how the matrix allows multiple masters to use the buses at the same time:

- The ARM926EJ-S Data AHB master is accessing `0x10004000` and this decodes to the external AHB M2 bus (the CODEC interface in the FPGA).
- The ARM926EJ-S Instruction AHB master is accessing `0x02000000` and this decodes to dynamic memory on one of the MPMC slaves.
- The CLCDC master is accessing `0x70000000` and this decodes to dynamic memory on one of the MPMC slaves. The MPMC will manage the multiple accesses to the slave ports.
- The DMAC is doing a memory to peripheral transfer. DMA master 1 is accessing `0x38000000` which decodes to static memory (SRAM). DMA master 0 is accessing `0x101F0000` which is mapped to the DMA APB bus).

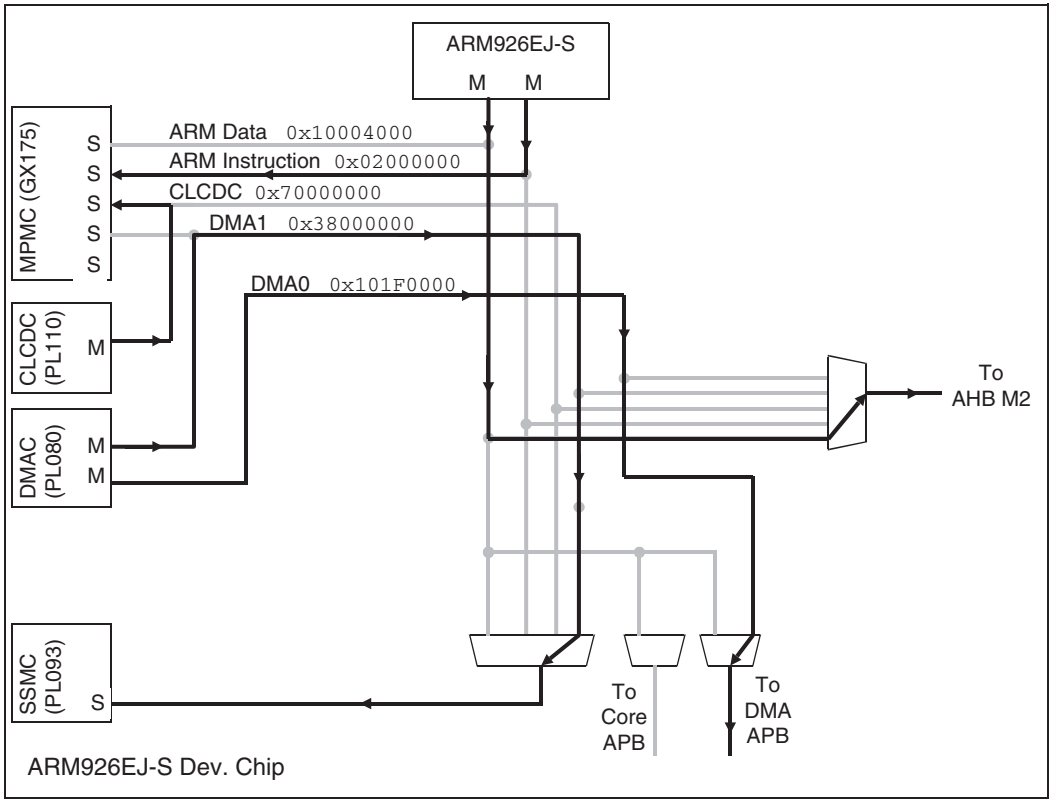

Figure 3-2 Multiple masters

The default memory map for each of the internal buses is slightly different as shown in Figure 3-3 on page 3-11 and Figure 3-4 on page 3-12.

Note

nMPMCDYCS[0] is the chip select for the dynamic memory present on the Versatile/AB926EJ-S. **nMPMCDYCS[3:1]** are not used. **STATIC_CS[2:0]** are the chip selects for the static memory present on the Versatile/AB926EJ-S. **STATIC_CS[7:3]** are for expansion static memory.

For more information on the system buses, see *Memory map* on page 4-3, *AHB buses used by the FPGA* on page 3-17, and the *ARM926EJ-S Development Chip Reference Manual*.

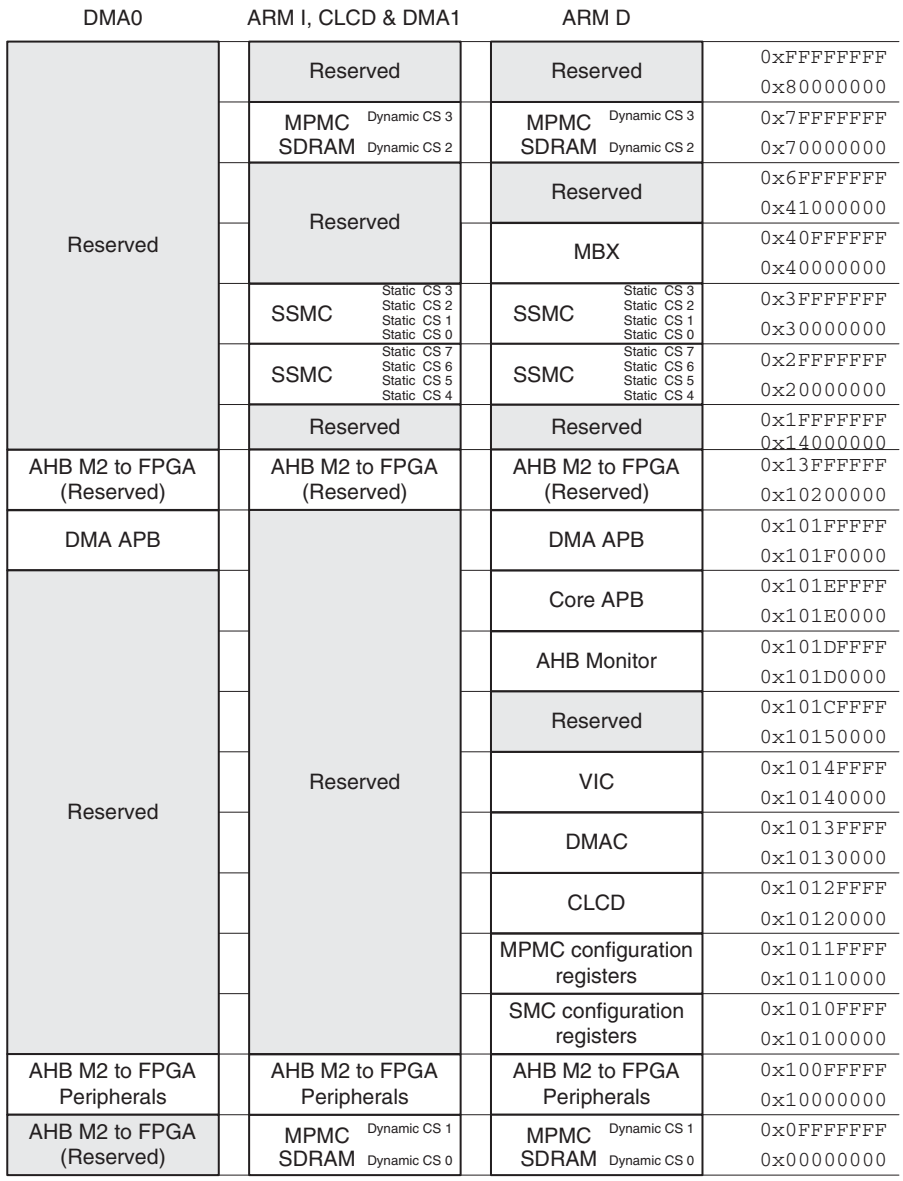

Figure 3-3 AHB map

The DMA and Core APB buses (0x101E0000–0x101FFFFF) shown in Figure 3-3 are shown in more detail in Figure 3-4 on page 3-12.

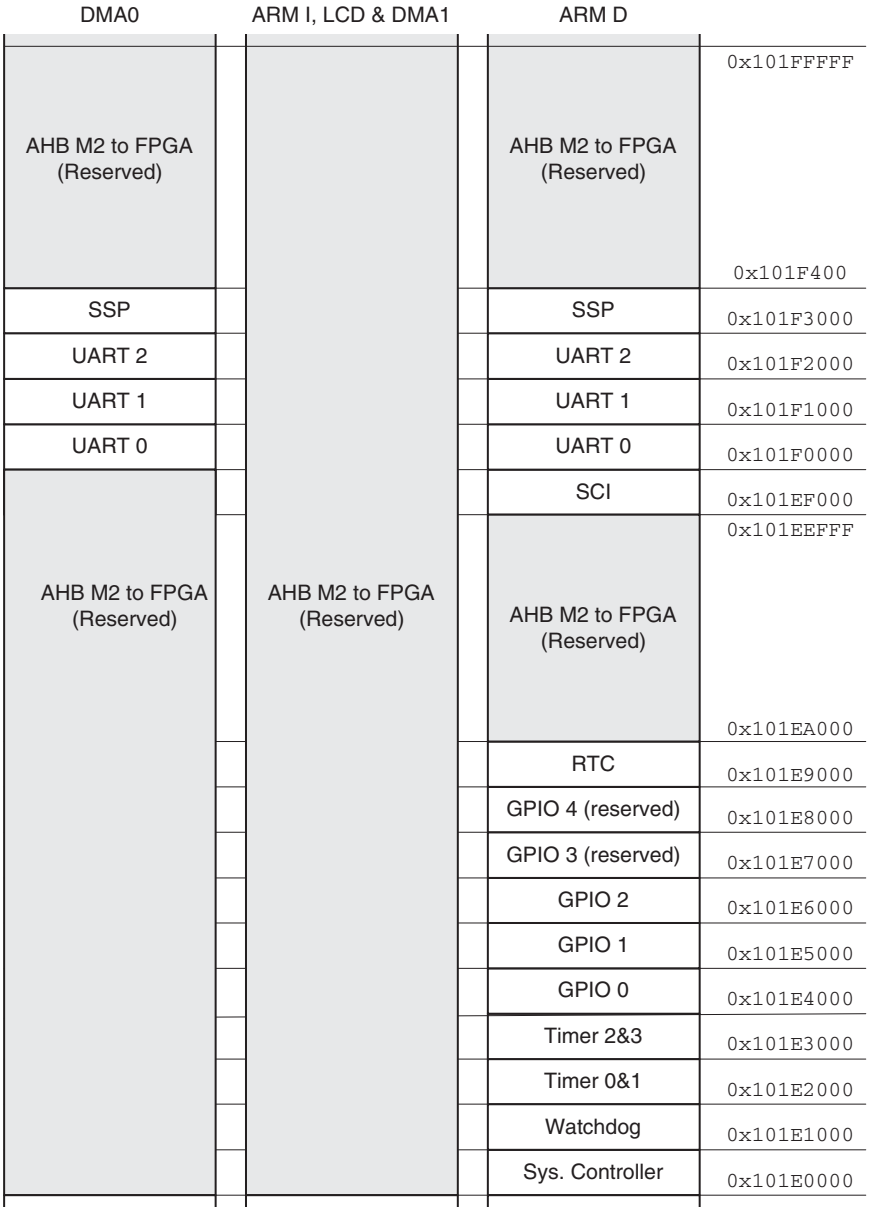

Figure 3-4 Core APB and DMA APB map

3.1.4 Memory interface

Memory access is provided by a MultiPort Memory Controller (MPMC) and a Static Memory Controller (SSMC). Both controllers are located in the ARM926PXP development chip. The static memory signals are also connected to the interface board.

The PLD decodes the STATIC_CS[7:0] chip select signals. (See *Memory aliasing at reset* on page 3-20 for a details on how the boot switches select the static memory to boot from.)

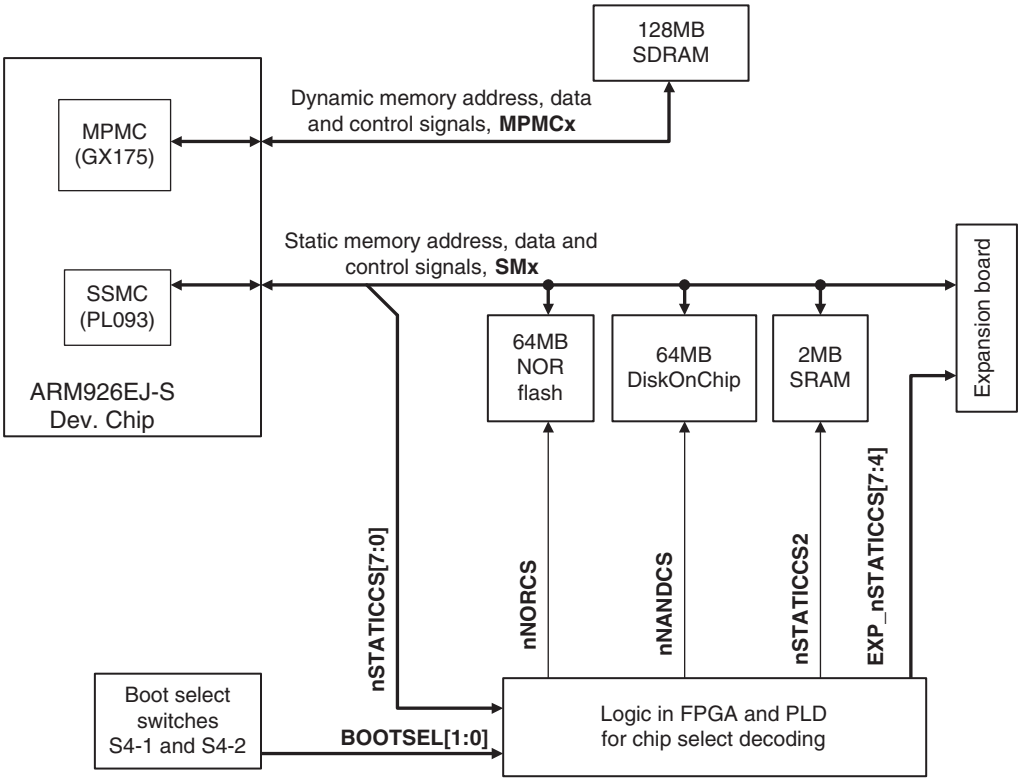

Figure 3-5 Memory devices

3.2 FPGA

Figure 3-6 shows the architecture of the FPGA on the Versatile/AB926EJ-S.

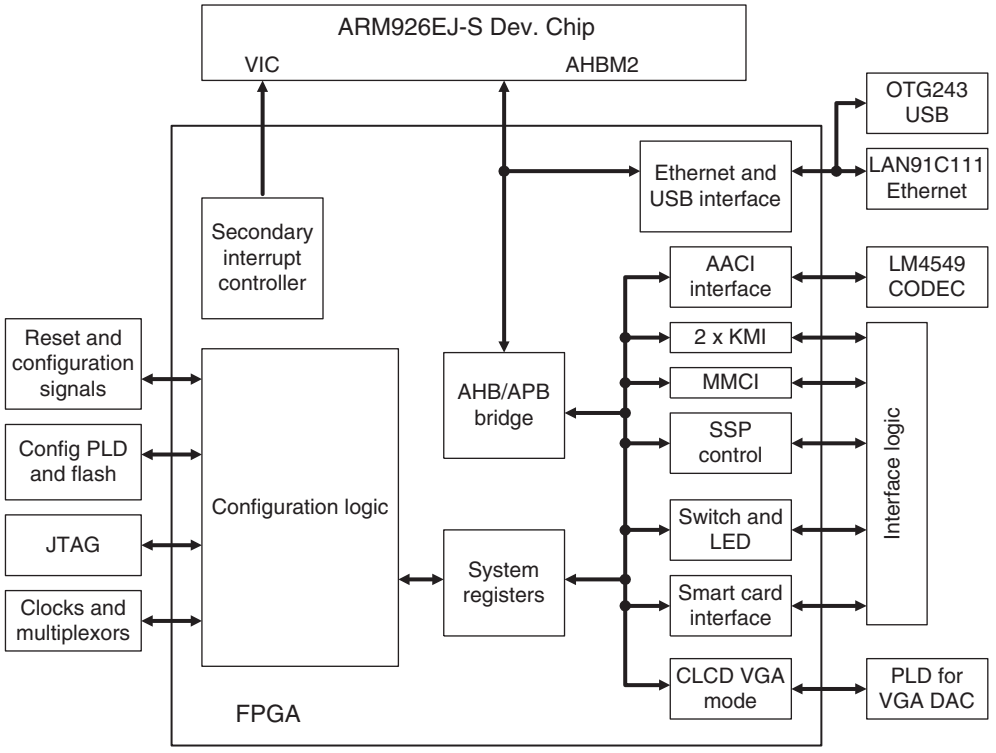

Figure 3-6 FPGA block diagram

For details on FPGA components, see:

- *FPGA configuration* on page 3-15
- *Reset controller* on page 3-18
- *Keyboard/Mouse Interface, KMI* on page 3-51
- *User switches and LEDs* on page 3-59
- *Advanced Audio Codec Interface, AACI* on page 3-38
- *SD/MultiMedia Card Interface, MCI* on page 3-52
- *Ethernet interface* on page 3-45
- *USB interface* on page 3-60

3.2.1 FPGA configuration

At power-up the FPGA loads its configuration data from a flash memory device. Parallel data from the flash memory is streamed by the configuration PLD into the configuration ports of the FPGA. Figure 3-8 on page 3-16 and Figure 3-7 show the FPGA configuration mechanism. The image loaded into the FPGA is determined by configuration switch S4-3 as listed in Table 3-2.

The reset and control signals are also described in Table 3-4 on page 3-25.

Table 3-2 FPGA image selection

S4-3	FPGA image	image Address
OFF	FPGA image 1 (this is the image supplied with the board)	`0x0`
ON	Reserved	Reserved

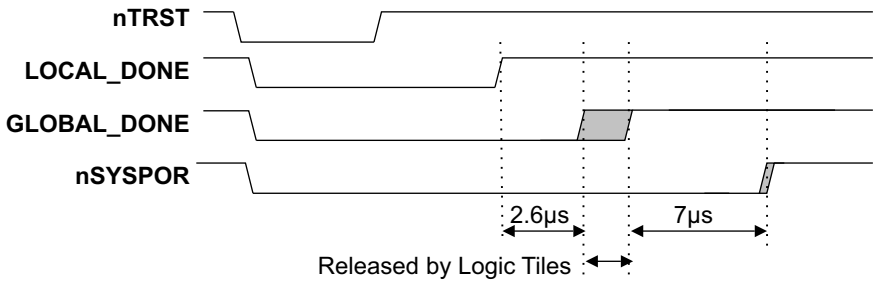

Figure 3-7 FPGA reload sequence

Note

The configuration flash can hold one FPGA image. The selection switch S4-3 must be in the off position.

The configuration flash is a separate device and not part of the user flash.

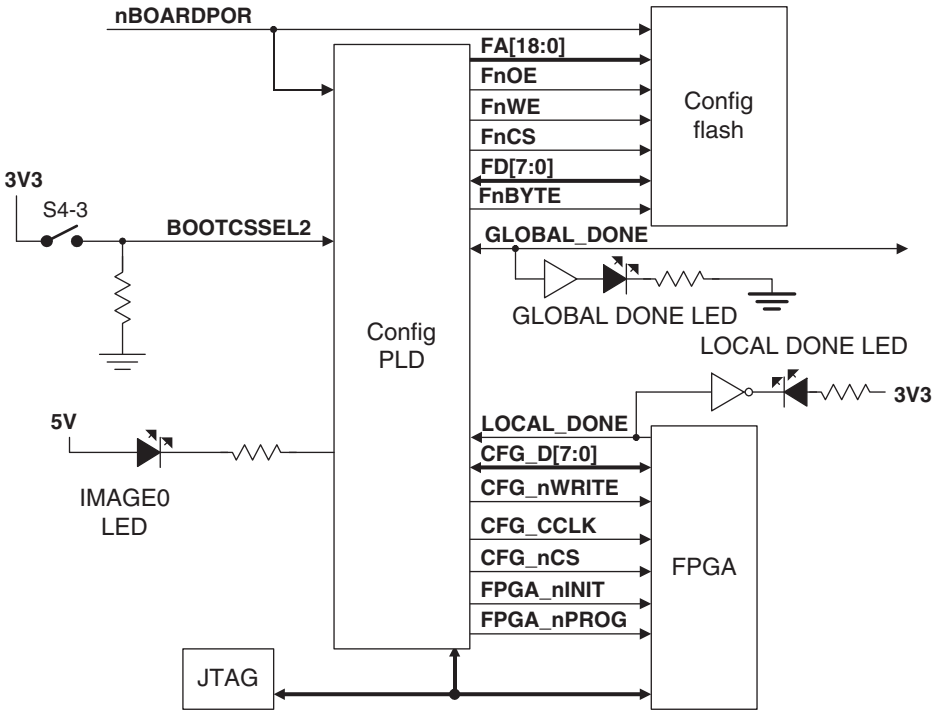

Figure 3-8 FPGA configuration

You can use a JTAG debugger or the ProgCards utility to reprogram the PLDs, FPGA, and flash if the Versatile/AB926EJ-S is placed in configuration mode. See also *JTAG and USB debug port support* on page 3-64.

The Versatile/AB926EJ-S is supplied with the configuration PLD and flash image already programmed. The information in this section is provided, however, in case of accidental erasure of the configuration PLD or flash image or if it is necessary for you to load a new configuration image to fix a defect in the image.

Caution

You are advised not to reprogram these device with any images other than those provided by ARM Limited.

Program the configuration PLD and configuration flash memory as follows:

1. Put the Versatile/AB926EJ-S into configuration mode by fitting the CONFIG link J22 on the board.

   Note

   The CONFIG link is a switch on some board versions. If present, turn the switch to ON,

2. Connect an interface cable to either the JTAG or USB debug port.
3. If you are using Multi-ICE, start the JTAG application and autoconfigure.

   If autoconfiguration fails, load the Multi-ICE configuration file (`.cfg`) for the board. For details on manual configuration, see the `readme.txt` file in the `boardfiles` directory on the CD.
4. Run the appropriate version of `the Progcards` utility from the `boardfiles` directory:

   Multi-ICE use `progcards.exe`

   USB debug port use `progcards_usb.exe`
5. Choose the required image for the configuration PLD, FPGA, and configuration flash.

3.2.2 AHB buses used by the FPGA

ARM926PXP development chip bus AHBM2 is connected to the FPGA.

Transactions in the addresses range `0x14000000–0x1FFFFFFF` are directed to AHBM2 by the ARM926PXP development chip, but do not select any of the slaves within the Versatile/AB926EJ-S FPGA.

3.3 Reset controller

The reset controller initializes the ARM926PXP development chip, the FPGA, and external controllers as a result of a reset. The Versatile/AB926EJ-S can be reset from the following sources:

- power failure
- reset button
- JTAG
- software.

Note

Use the RESET pushbutton, a JTAG reset, or a software reset to reset the ARM926EJ-S core. The current ARM926PXP development chip configuration settings are retained. (The effect of the RESET pushbutton can be modified by setting a reset level flags, see *Reset level* on page 3-22.)

3.3.1 Reset and reconfiguration logic

Figure 3-9 on page 3-19 shows the reset and reconfigure logic. (Not all JTAG reset signals are shown.) See Table 3-4 on page 3-25 for a description of the reset signals.

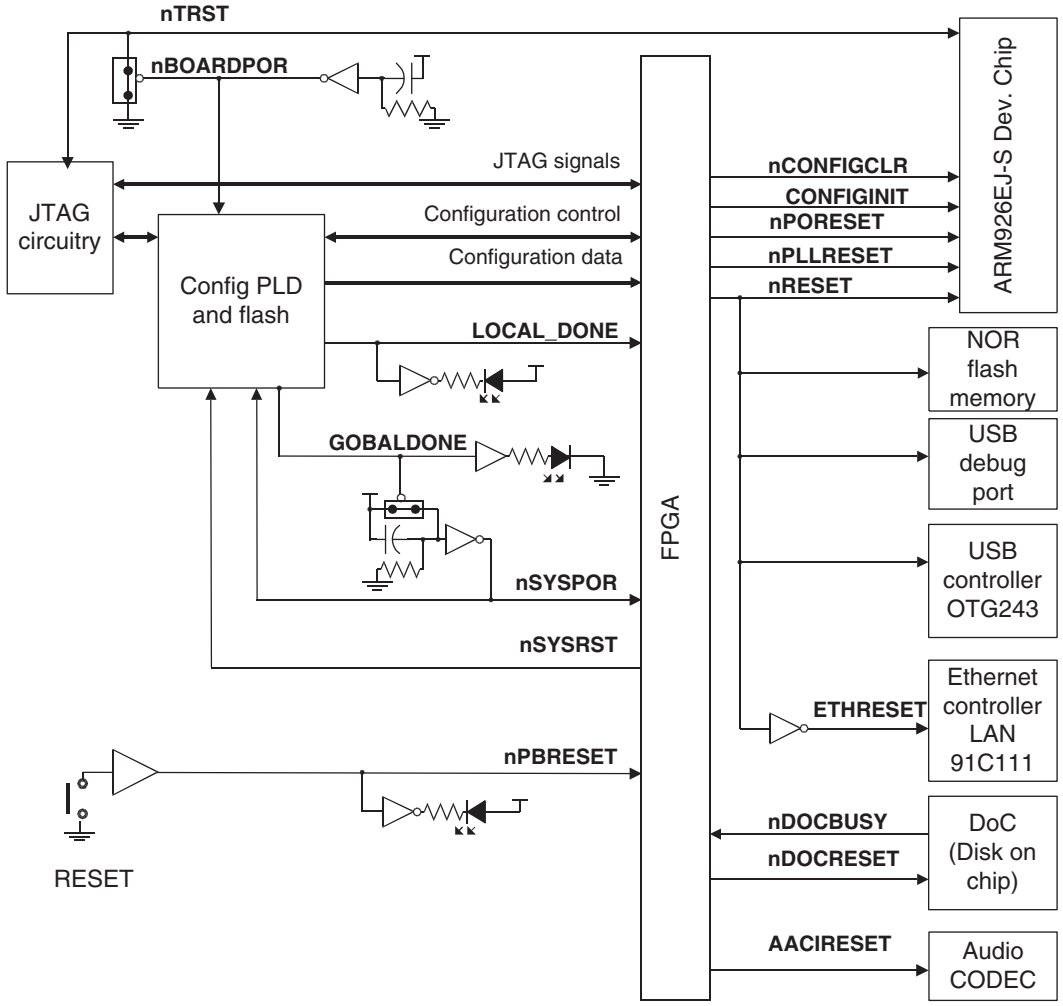

Figure 3-9 Reset logic

3.3.2 Memory aliasing at reset

Remapping the memory is done by changing how the chip select signals in the ARM926PXP development chip connect to the external chip select signals that control memory devices. Figure 3-10 on page 3-21 shows the two stage remapping process:

- If **DEVCHIP REMAP** signal is HIGH, from the system controller, it disables the **nMPMCDYCS0** signal (dynamic memory chip select) that is normally generated by accesses to memory region `0x00000000`–`0x03FFFFFF`.

  Accesses to memory region `0x00000000`–`0x03FFFFFF` are remapped to **nSTATICCS1** (static memory chip select).

  This remapping occurs inside the ARM926PXP development chip.

- If **FPGA_REMAP** is HIGH, from the SYS_MISC register, **nSTATICCS1** is remapped to:

  — **nDOCCS** (Disk-on-Chip) if **BOOTCSSEL[1:0]** is `b00`

  — **nNORCS** (NOR flash memory) if **BOOTCSSEL[1:0]** is `b01`.

  This remapping occurs inside the FPGA.

At reset, the **DEVCHIP REMAP** and **FPGA_REMAP** signals are both HIGH.

Which of **nDOCCS** or **nNORCS** memory is active at reset therefore depends on the value of the **BOOTCSSEL[1:0]** signals.

Note

If the size of the physical memory selected by **nDOCCS** or **nNORCS** is less than the address range of `0x00000000`–`0x03FFFFFF`, the physical memory is aliased and repeated to fill the address space.

The static expansion memory cannot be used by the boot monitor as boot memory. If **nNEXPCS** is remapped to low memory, the expansion memory will not be aliased at the high memory address and the jump to high memory from the boot monitor code will fail.

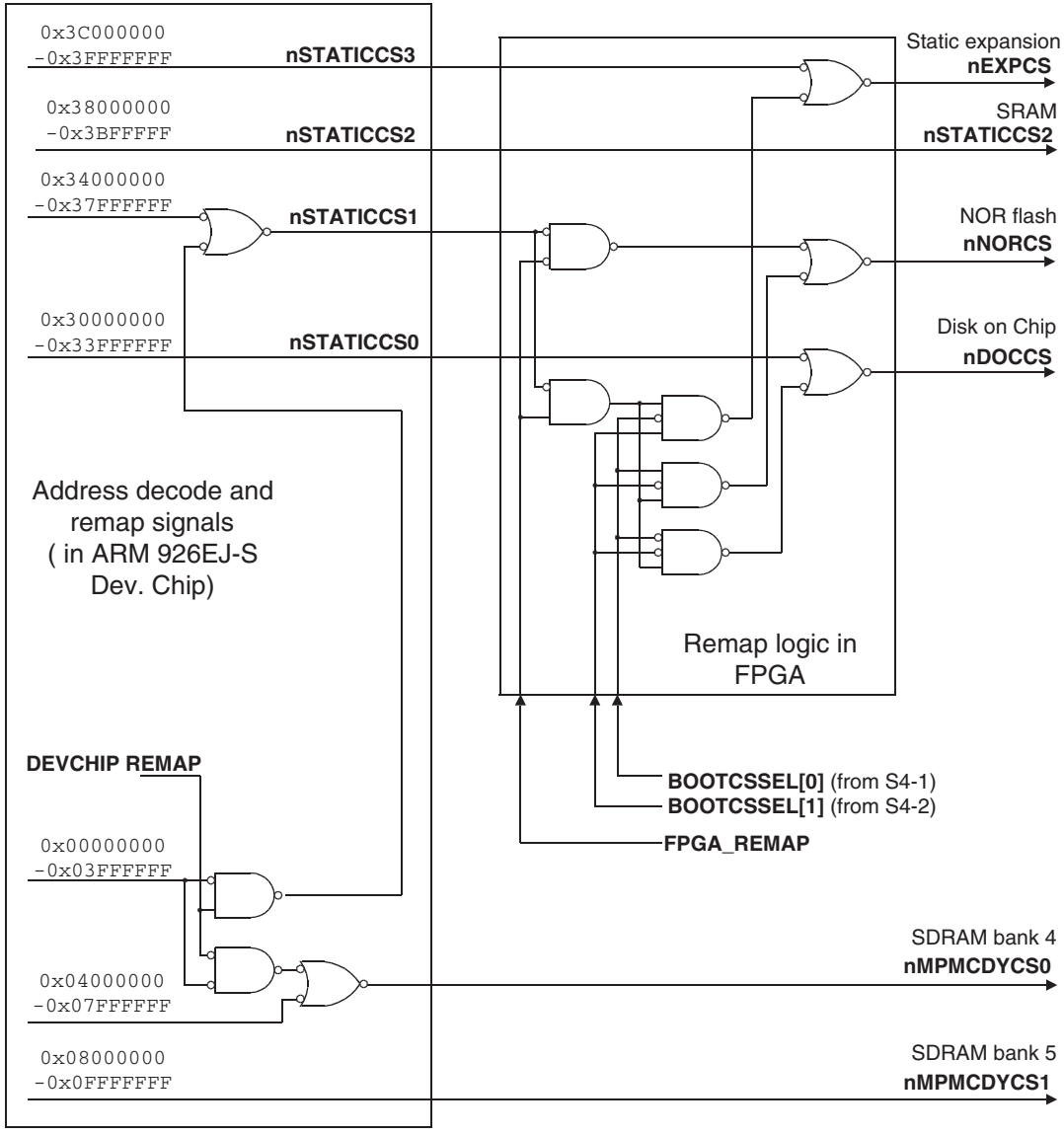

Figure 3-10 Boot memory remap logic

3.3.3 Reset level

Table 3-3 lists the default levels of reset that results from external sources.

Table 3-3 Reset sources and effects

External source	Reset level	Hardware nBOARDPOR generated	Reset generatedfor CPU, memory and peripherals
Power on	0	Yes	Yes
RESET pushbutton or software reset	6	No	Yes

Figure 3-11 on page 3-23 shows the activity on the reset signals at different levels of reset.

The level of reset that results from pressing the RESET pushbutton or generating a software reset can be configured by the SYS_RESETCTRL register. The ability to configure the reset level gives greater flexibility in designing applications, FPGA images, and RealView Logic Tile IP.

Set SYS_RESETCTRL[8] to generate a software reset.

The reset levels specified by SYS_RESETCTRL[2:0] are:

- `b000` is reserved
- `b001` resets to level 1, **CONFIGCLR**
- `b010` resets to level 2, **CONFIGINIT**
- `b011` resets to level 3, **DLLRESET** (DLL located in FPGA)
- `b100` resets to level 4, **PLLRESET** (located in ARM926PXP development chip)
- `b101` resets to level 5, **PORESET**
- `b110` resets to level 6, **DOCRESET**
- `b111` is reserved.

Note

The Versatile/AB926EJ-S does not differentiate between some of the reset levels. The reset programming for the Versatile/PB926EJ-S has six reset levels to enable more detailed configuration options. The same levels are used in the Versatile/AB926EJ-S to maintain code compatibility between the two platforms.

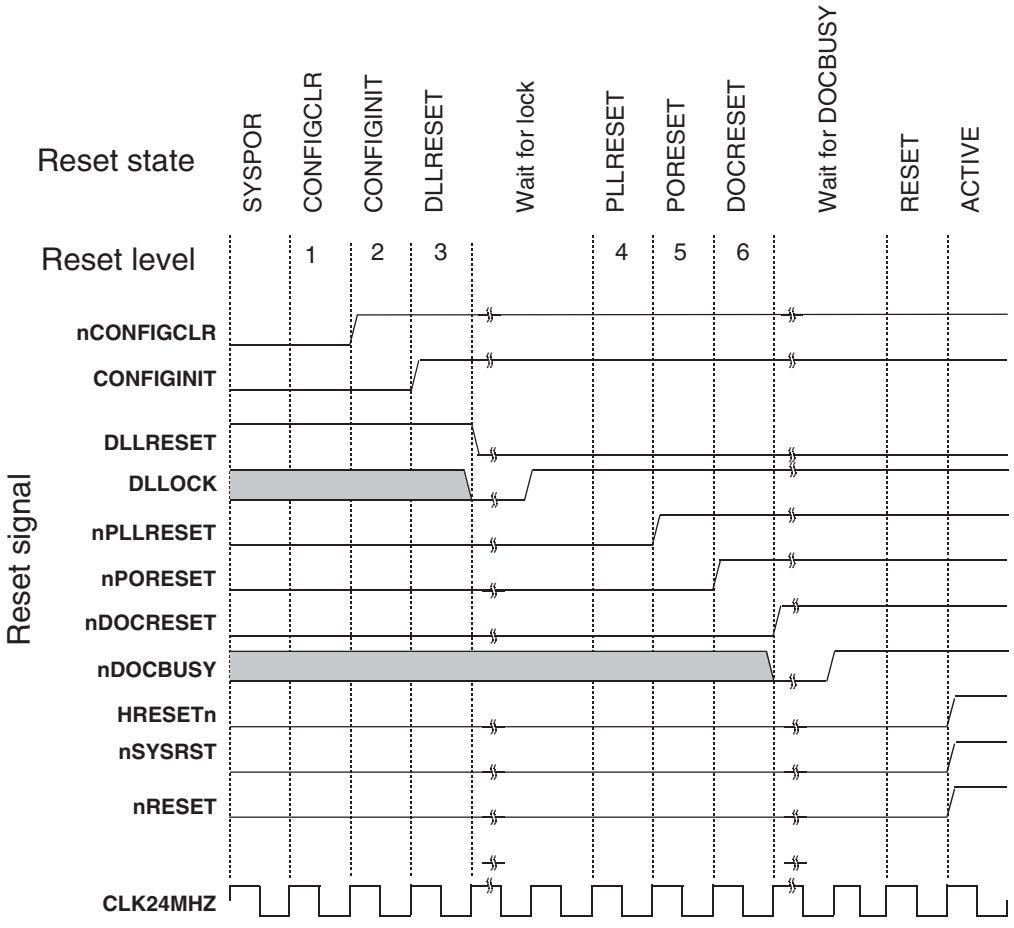

Figure 3-11 Reset signal sequence

A state machine in the FPGA (see Figure 3-12 on page 3-24) uses the value of SYS_RESETCTRL and the external reset signals to sequence the reset signals.

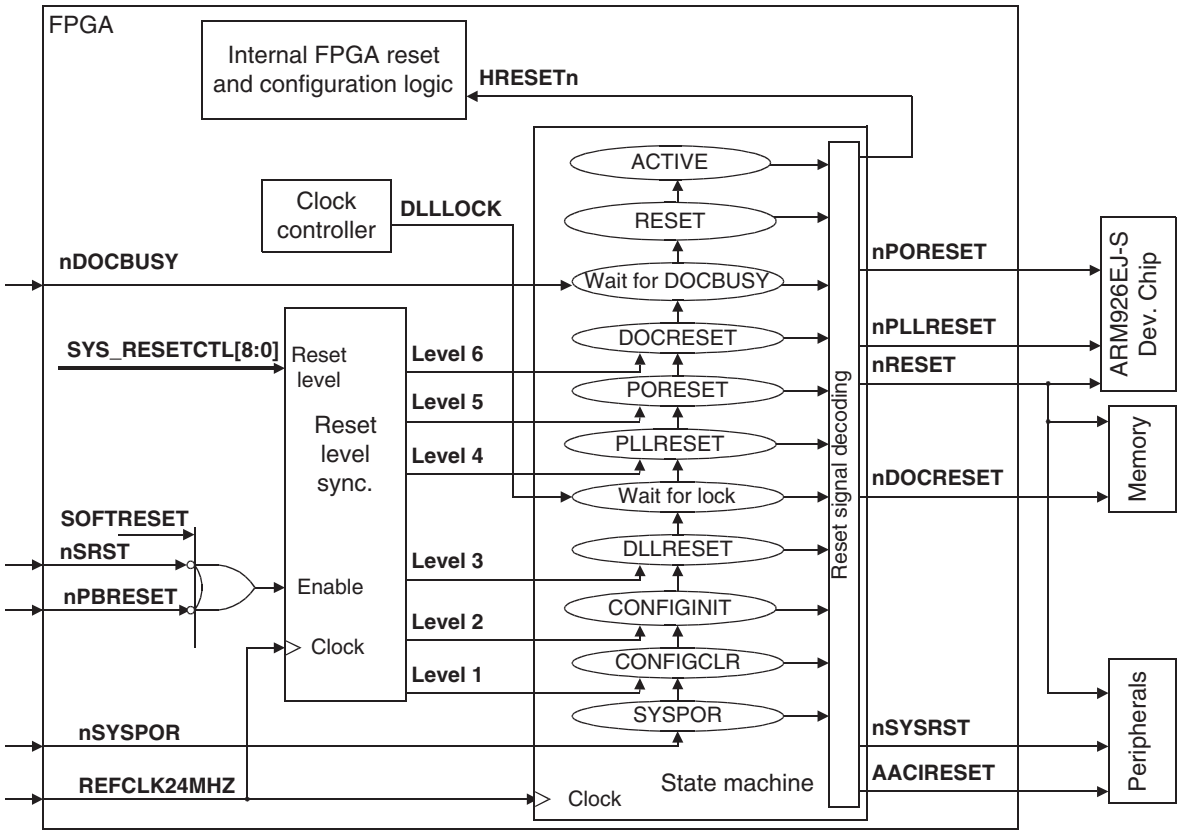

Figure 3-12 Programmable reset level

See Table 3-4 on page 3-25 for a description of the reset signals.

3.3.4 Reset signals

Table 3-4 describes reset signals.

Table 3-4 Reset signal descriptions

Name	Function
AACIRESET	System reset to audio codec.
nBOARDPOR	This signal is used to generate the **nTRST** pulse at power on. This pulse resets the configuration PLD and configuration flash.
EXPnSRST	JTAG open-collector reset signal (from **FPGAnINIT**) to the expansion connector. This signal is part of the configuration JTAG chain.
EXPnTRST	JTAG **TRST** signal to the expansion connector. This signal is part of the configuration JTAG chain.
nDOCBUSY	Memory status signal from the Disk-on-Chip. The DoC requires approximately 27ms after reset is applied in order to initialize its internal controllers.
nDOCRESET	System reset to Disk-on-Chip memory.
FPGA_nPROG	The **FPGA_nPROG** signal forces all FPGAs in the system to reconfigure. This signal enables the FPGAs to be reconfigured without powering-down the system.
GLOBAL_DONE	This is an open-collector configuration signal that goes HIGH when all FPGAs have finished configuring. The system is held in reset until this signal goes HIGH.
nPBRESET	Push-button reset signal to the FPGA. The signal is generated by pressing the reset button.
nPLLRESET	Reset for ARM926PXP development chip PLL clock circuit.
nPORESET	Power-on reset to development chip, configuration flash, and expansion memory. The CPU core, all system peripherals, and all system controller registers are reset and the ARM926PXP development chip configuration data is reloaded. For details on system registers reset at different reset levels, see Table 4-9 on page 4-19.

Table 3-4 Reset signal descriptions (continued)

Name	Function
nPWRFAIL	This signal is provided by the FPGA to the interrupt controller. User software can test this signal and shut down before a power loss causes a loss of data. **Note** This signal is not driven by any power-detection logic. It is provided so that custom implementations of the FPGA image have a signal that could be manipulated by a register. Creating such an FPGA image would enable testing of user software that implements a shutdown routine.
nRESET	Reset signal to the development chip and FPGA. The CPU core, all system peripherals, and some system controller registers are reset. This signal is synchronized with the system bus clock to provide AMBA compliance. For details on system registers reset at different reset levels, see Table 4-9 on page 4-19.
nSRST	**nSRST** is an active LOW open-collector signal that can be driven by the JTAG equipment to reset the board. Some JTAG equipment senses this line to determine when you have reset a board. This is also used in configuration mode to control the initialization of the FPGA.
nSYSPOR	Power-on reset signal that initializes the reset level state machine after **GLOBAL_DONE** goes HIGH.
nTRST	TAP controller reset (the board drives this signal with **nBOARDPOR**).

3.3.5 Reset timing

Figure 3-13 on page 3-27 shows the power-on reset sequence.

nBOARDPOR is generated at power-up and connected to the FPGA by the **nTRST** signal.

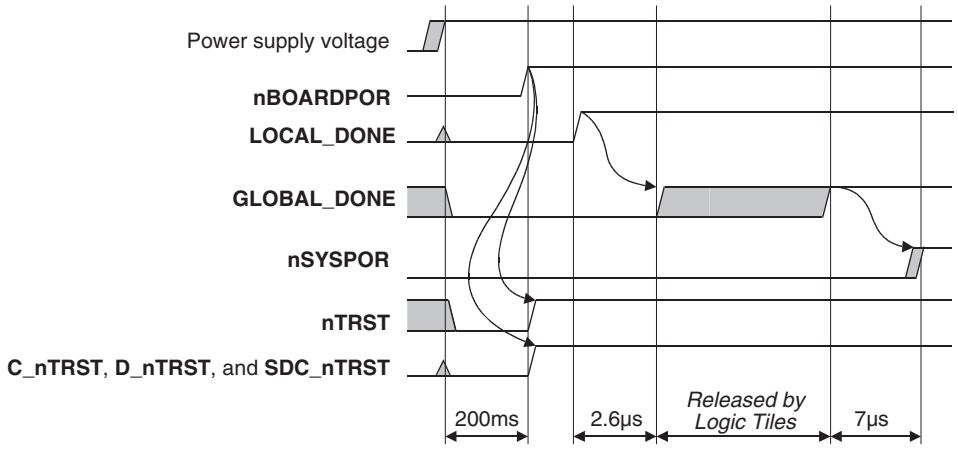

Figure 3-13 Power-on reset and configuration timing

On power up, **nTRST** is asserted to guarantee the embedded ICE macrocell is reset in the ARM926PXP development chip.

3.3.6 Standby power, battery pack, and automatic shutdown

The Versatile/AB926EJ-S is powered from the brick power supply and a nominal 12V level (**DCIN**) is supplied to the 5V and 3V regulators. If **DCIN**, drops too low, shutdown signals **nPOWERFAIL** and **nSHDN1** become active and power is switched off. The shutdown circuitry is shown in Figure 3-14 on page 3-28. The power supply can be toggled on and off by pressing the Power/Standby pushbutton.

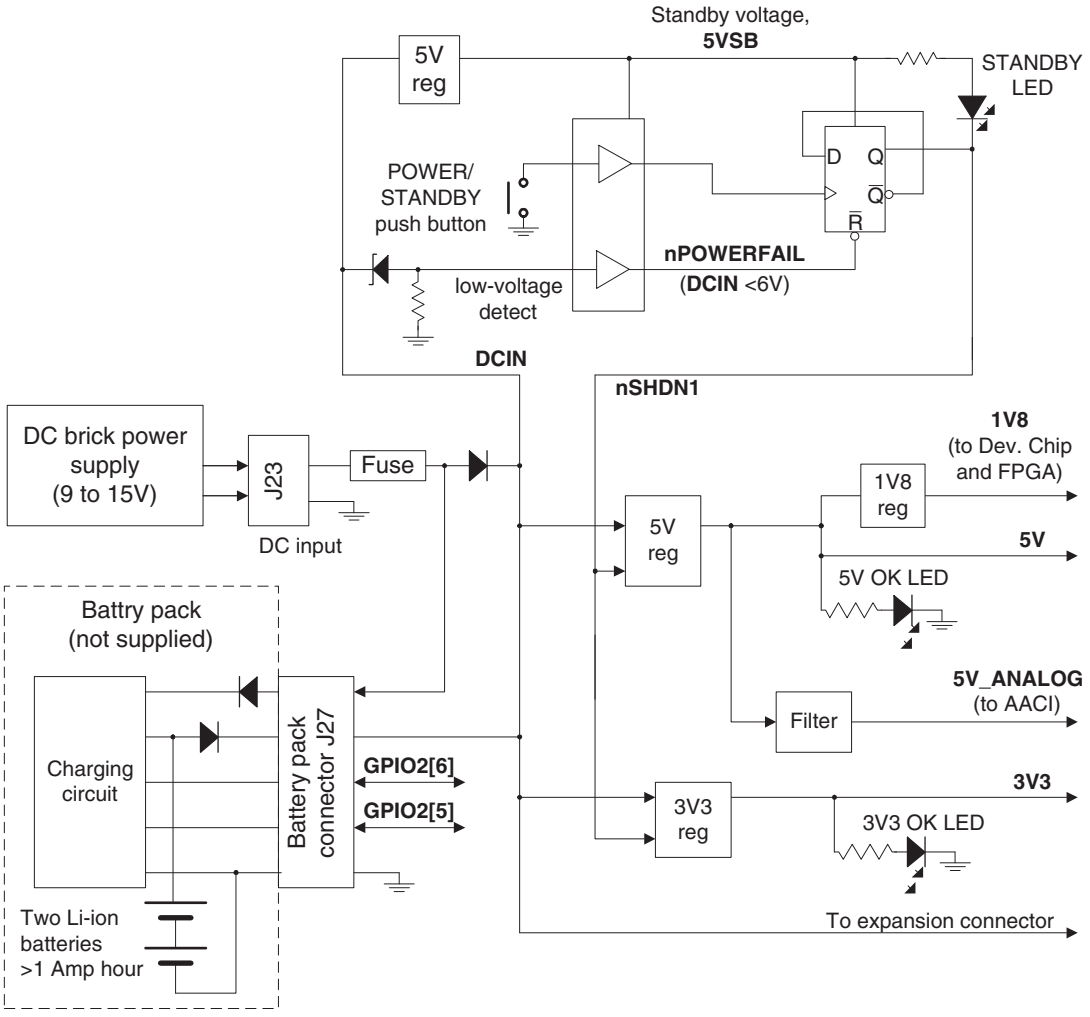

Figure 3-14 Standby switch and power-supply control

The battery pack connector (J27) enables connection to a custom battery pack for portable application development. Two GPIO pins (GPIO2[6:5) enable monitoring battery status from application software. This interface protocol for the monitor interface is not predefined, but typically will use a serial bus.

3.4 Clock architecture

The clock domains for the Versatile/AB926EJ-S are shown in Figure 3-15.

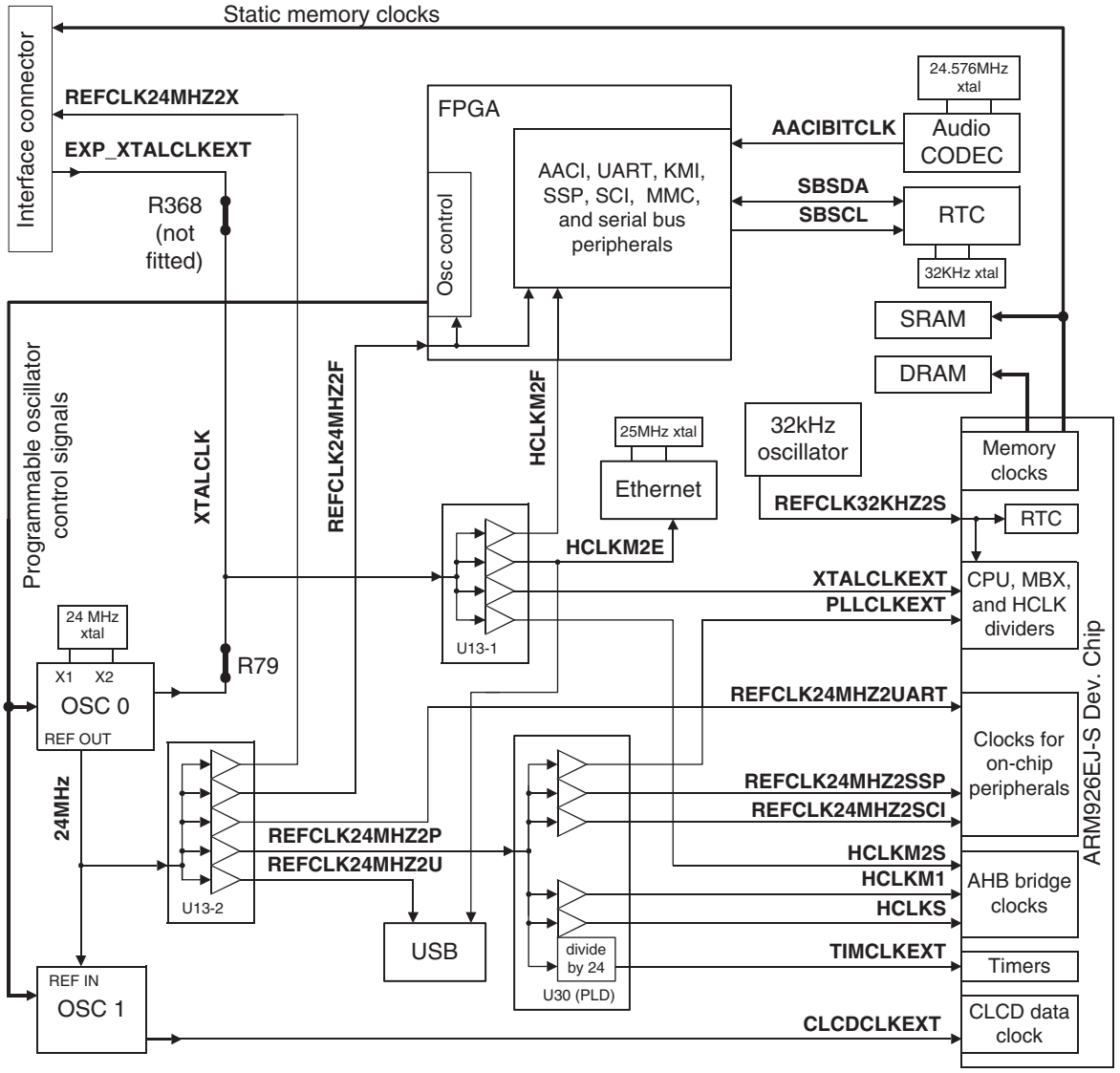

Figure 3-15 Clock distribution

The main reference for the system is the 24MHz crystal connected to the programmable oscillator OSC0. A 24MHz clock from the bypass output of the oscillator OSC0 is used for:

Reference for OSC0 programmable oscillator

The 24MHz clock is the reference for the OSC0 programmable oscillator. The programmable output of OSC0 is normally used as **XTALCLK** and distributed to the FPGA, Ethernet logic, and the ARM926PXP development chip.

The FPGA contains clock control logic that can set the frequency of the programmable clock generators in OSC0 and OSC1.

Reference for OSC1 programmable oscillator

The 24MHz clock is also the reference for the OSC1 programmable oscillator. The programmable output of OSC1 is used as the bit clock for the CLCD controller in the ARM926PXP development chip.

ARM926PXP development chip peripherals

The SSP and SCI peripherals inside the use the 24MHz reference directly. The Timer uses a 1MHz clock derived from the 24MHz reference.

Alternative ARM926PXP development chip PLL clock

The ARM926PXP development chip normally uses the programmable output of OSC0 as the reference clock for the CPU, bus, and memory clock dividers. The System Controller in the ARM926EJ-S development chip can select the 24MHz clock signal on **PLLCLKEXT**, however, as an alternative source.

USB The USB uses the 24MHz clock for bus and interface timing.

Ethernet A buffered version of **HCLKM2** is used as a reference frequency for the controller interface to the FPGA.

In addition to the 24MHz reference and the outputs from the programmable oscillators, the following reference clocks are also present:

24.576MHz output Audio CODEC

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

32kHz external RTC reference

There is an external real-time clock clocked by a dedicated 32kHz crystal oscillator. The external RTC interfaces with the FPGA over a serial bus.

32kHz oscillator module

A 32kHz oscillator modules outputs the **REFCLK32K** clock to the RTC inside the ARM926PXP development chip.

External XTALCLK from interface board

XTALCLK is normally sourced from OSC1, however resistor links R79 and R368 allow it to be sourced from the interface board.

25MHz Ethernet oscillator

The Ethernet controller has a 25MHz dedicated crystal oscillator for timing signals to and from the Ethernet connector.

6MHz USB debug oscillator

The USB debug interface has a dedicated 6MHz oscillator (not shown in *Clock distribution* on page 3-29).

The clocks and clock-related logic are described in the following sections:

- *ARM926PXP development chip clocks*
- *ICS307 programmable clock generators* on page 3-35
- *Peripheral clocks* on page 3-33

Note

The default values for clock selection and frequency are appropriate for most situations.

Changing the clock divider options requires modifying the HDL inside the FPGA.

3.4.1 ARM926PXP development chip clocks

This section describes the clocks used by the ARM926PXP development chip. Figure 3-16 on page 3-32 shows the clock circuitry inside the chip.

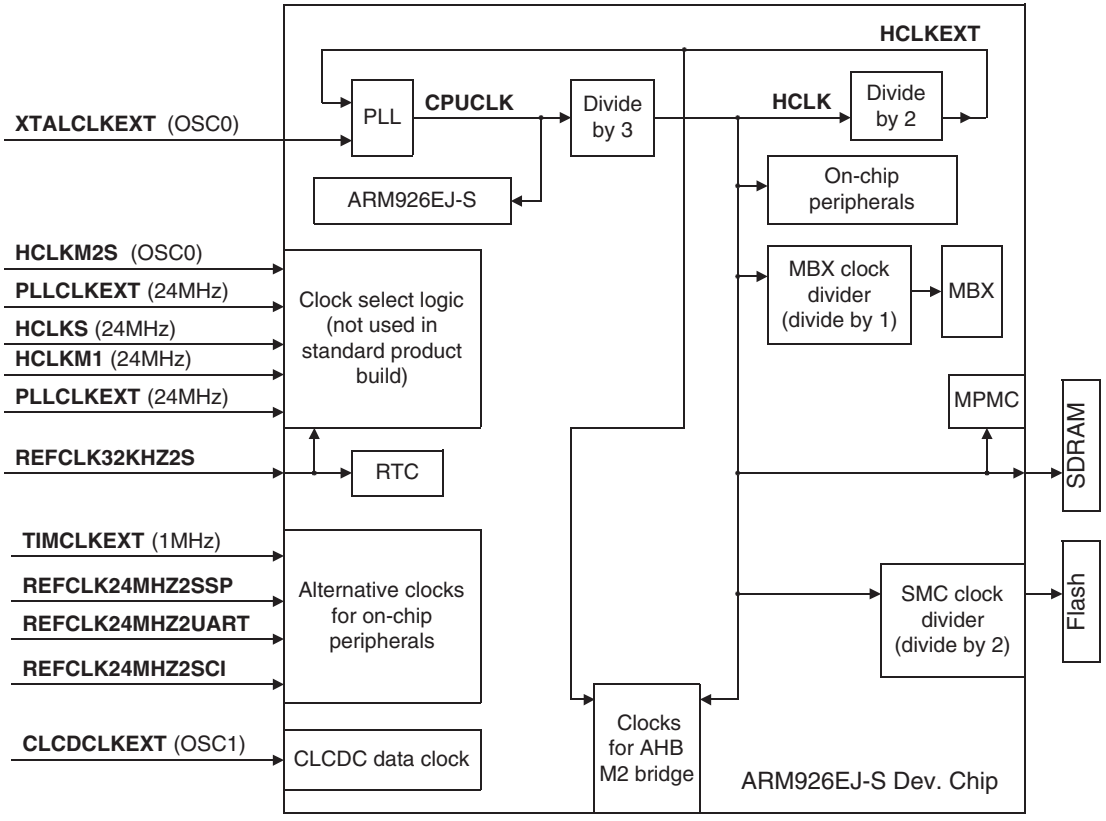

Figure 3-16 ARM926PXP development chip internal clock logic

The ARM926PXP development chip CPU clock is normally based on OSC0. Alternatively, the CPU can be clocked from an external clock from an oscillator on an interface board. (See the *ARM926-EJS Development Chip Reference Manual* for details of the clock circuitry inside the chip.) For details of the peripheral clocks, see *Peripheral clocks* on page 3-33.

The external part of AHB M2 bridge operates in synchronous mode and the bridge clocks are based on the CPU clock.

The default values for the clock sources and clock dividers are determined by the HDL in the FPGA image. For the default clock source and configuration:

- OSC0 provides the **XTALCLKEXT** input clock for the PLL in the ARM926PXP development chip.

- The PLL output **CPUCLK** is used as the CPU core clock and as the input to the **HCLK** divider.
- **HCLK** is **CPUCLK** divided by 3. **HCLK** is used as the SDRAM clock **MPMCCLK**, and as the inputs to the MBX and SMC clock dividers.
- **HCLKEXT** is **HCLK** divided by 2. **HCLKEXT** is the reference clock for the external part of the AMBA bridges M1, M2, and S. This clock is the feedback clock for the PLL, therefore the frequency of **HCLKEXT** is the same as that of **XTALCLKEXT**.

Note

The programmable output of OSC0 is normally the source for **XTALCLKEXT**. If the resistor links on the board are changed, **EXP_XTALCLKEXT** from the interface connector can be used as the source for **XTALCLKEXT**.

The clocks must also be within their operational limits, see *Clock rate restrictions* on page B-4.

The AHBM2 bridges only operates in synchronous mode with the FPGA and PLD images supplied with the product. The internal part of the AHB bridge is clocked by **HCLK** and external part of the bridge is clocked by **HCLKEXT**. **HCLKEXT** is the feedback to the PLL, so the **HCLKEXT** frequency is the same as the PLL reference frequency **XTALCLKEXT**.

CPUCLK is generated by multiplying the reference **XTALCLKEXT** by six. (Times three for the **HCLK** divider and times two for the **HCLKEXT** divider.)

Low-frequency clocks and power-saving mode

The system controller in the ARM926PXP development chip can switch the system into power-saving modes (slow, doze, and sleep). The Versatile/AB926EJ-S however, does not support power-saving modes.

Peripheral clocks

Table 3-5 on page 3-34 lists the memory and peripheral clocks on the Versatile/AB926EJ-S.

The UART, Smart Card Interface, and Synchronous Serial Port in the ARM926PXP development chip are clocked from a 24MHz reference clock. The clock is a buffered version of the OSC0 24MHz crystal reference.

The Dual Timer Counter modules in the ARM926PXP development chip are clocked by a 1MHz reference clock from the PLD. The clock is a buffered version of the OSC0 24MHz crystal reference that has been divided by 24 in the PLD.

For more detail on the clocking system, see the files in the `Schematics` directory of the CD supplied with the Versatile/AB926EJ-S.

Table 3-5 Versatile/AB926EJ-S clocks and clock control signals

Clock signal	Frequency	Description	Source
TIMCLKEXT	1MHz	The dual timer modules in the ARM926PXP development chip are clocked from an external 1MHz clock derived from the 24MHz reference.	24MHz crystal and divider in PLD
AACIBITCLK	12.288MHz	This is the synchronization clock from the audio CODEC. The clock is an input to the PL041 AACI PrimeCell.	Crystal oscillator
CLCDCLKEXT	25–50MHz	The clock for PL110 CLCD Controller in the development chip.	ICS307 OSC1 programmable output
HCLKM2E	**XTALCLK**	This clock synchronizes data transfers between the external controller and the FPGA. This signal is at the same frequency as the HCLKM2S AHB bridge clock.	ICS307 OSC0 programmable output
SMCLK[2:0]	-	The static memory clocks from the SSMC in the development chip. The clock frequency is determined by **HCLK** and the programmed divider ratio (normally divide by two).	SSMC controller
MPMC[4:0]	-	The dynamic memory clocks from the MPMC in the development chip.	MPMC controller
REFCLK24MHZ2UART	24MHz	The clock for PL011 UART in the development chip can be derived from this input.	24MHz reference from ICS307 OSC0
REFCLK24MHZ2SCI	24MHz	The clock for PL131 SCI in the development chip can be derived from this input.	24MHz reference from ICS307 OSC0
REFCLK24MHZ2SSP	24MHz	The clock for PL022 SSP in the development chip can be derived from this input.	24MHz reference from ICS307 OSC0

3.4.2 ICS307 programmable clock generators

Two programmable (6–200 MHz) clocks are supplied to the FPGA by the programmable MicroClock ICS307 clock generators (OSC0 and OSC1):

OSC0 Reference clock for **XTALCLKEXT**. This is normally used as the external AHB clock and the reference for the PLL that generates **CPUCLK** inside the ARM926PXP development chip.

OSC1 This the reference for the CLCD controller (a buffered version of this clock is output to the ARM926PXP development chip as **CLCDCLKEXT**.

OSC0 uses a 24MHz crystal as its reference and outputs a fixed-frequency 24MHz signal (**REFOUT**) and a programmble frequency (**XCLKEXT**). The 24MHz **REFOUT** clock from OSC0 is used as a reference signal for:

- The reference divider input for programmable oscillator OSC1.
- the Ethernet controller clock (the Ethernet serial data clock is generated from a 25MHz crystal on the Ethernet controller).
- the USB controller clock
- the external clock on the expansion connector
- the external peripheral clocks for the SCI, UART, and SSP in the ARM926PXP development chip.
- the input to divide-by-24 logic in the PLD that produces the 1MHz reference clock for the timers.

The output frequencies of the ICS307s are controlled by divider values loaded into the serial data input pins on the oscillators. The divider values are defined by the SYS_OSC0 and SYS_OSC1 registers. The register format for the divider values is shown in Figure 3-17.

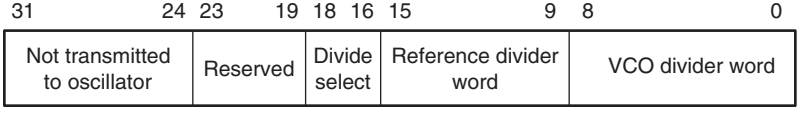

Figure 3-17 SYS_OSCx register format

Note

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

You can calculate the oscillator output frequency from the formula:

$$CLKA = \frac{48^* (VDW+8)}{(RDW+2)^*DIVIDE} \text{ MHz}$$

where:

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

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

DIVIDE Is the divide ratio (2 to 10) selected from SYS_OSCx[18:16]:

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

For more information on the ICS clock generator and a frequency calculator, see the ICS web site at `www.icst.com`. For details of the clock control registers, see *Status and system control registers* on page 4-19.

3.4.3 External real-time clock

An external DS1338 device provides a battery-backed real time clock. Use the serial bus to read the current time and date.

Note

The 1.5V cell battery provides the **VBATT** backup voltage to the external DS1338 time-of-year clock.

The battery is expected to last for approximately 10 years from manufacture of the Versatile/AB926EJ-S. To replace the battery:

1. Power on the Versatile/AB926EJ-S.

If the battery is removed while the board is powered down, the FPGA encryption keys will be erased. ARM might supply encrypted images in the future and if the keys are not present in the FPGA, the board must be returned to ARM to have the keys reloaded.

2. Remove the old battery.
3. Insert the new battery and ensure that the positive terminal is facing upwards in the holder.

3.5 Advanced Audio Codec Interface, AACI

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

Note

For a description of the audio CODEC signals, refer to the LM4549 datasheet available from the National Semiconductor website.

The AACI on the Versatile/AB926EJ-S connects to a National Semiconductor LM4549 audio CODEC. The audio CODEC is compatible with AC'97 Rev 2.1. Table 3-6 lists the specifications for the audio system.

Table 3-6 Audio system specification

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

Figure 3-18 on page 3-39 shows the architecture of audio interface.

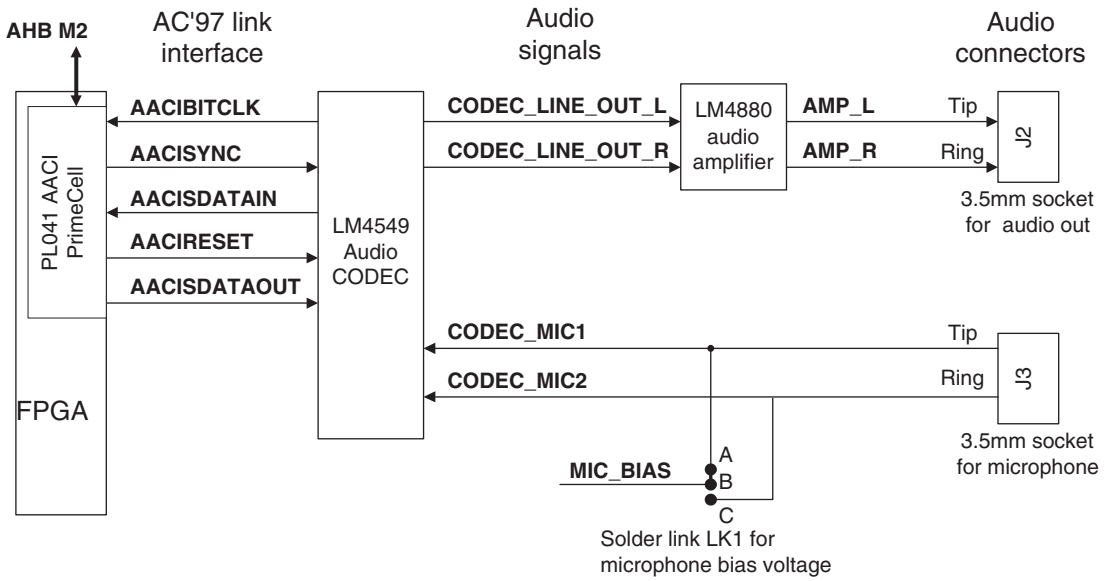

Figure 3-18 Audio interface

Two microphone inputs are present on J3. Only monophonic sound is supported, but microphone channel **CODEC_MIC1** or **CODEC_MIC2** can be selected in software. Solder link LK1 selects passive or active (electret) microphones:

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

This is the default configuration.

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

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

3.6 CLCDC interface

A PrimeCell CLCD controller is present in the ARM926PXP development chip.

The Versatile/AB926EJ-S provides a display interface with outputs to:

- a VGA connector for connecting a VGA or SVGA monitor
- the peripheral expansion connectors. The AB-IB1 interface board has a connector for an external CLCD kit with a 2.2, 3.8, or 8.4 inch display. The Versatile/AB-IB2 interface board has a built-in 2.5 inch CLCD display.

A PLD and a DAC convert the CLCDC data signals into VGA analog signals. The **LCDMODE[1:0]** signals select the mapping of CLCD video data to the RGB signals for different resolutions. The video PLD also manages the conversion of the CLCD data into 24 or 16-bit color depth for the VGA output.

Table 3-7 lists the display interface signals and Figure 3-19 on page 3-42 shows the architecture of the display interface.

See *Color LCD Controller, CLCDC* on page 4-34, Appendix E *LCD Kits*, and the *ARM926EJ-S Development Chip Reference Manual* for interface details.

Table 3-7 Display interface signals

Signal	Description
CLCD[23:0]	LCD panel data. This is the digital RGB signals and synchronization signals.
CLCP	LCD panel clock to the interface board.
CLLP	Line synchronization pulse (STN)/horizontal synchronization pulse (TFT) to the interface board.
CLFP	Frame pulse (STN)/vertical synchronization pulse (TFT) to the interface board.
CLAC	STN AC bias drive or TFT data enable output to the interface board.
CLLE	Line end signal to the interface board.
CLPOWER	LCD panel power enable to the interface board.
B[7:0]	Blue output signals to D/A converter and to the interface board.
G[7:0]	Green output signals to D/A converter and to the interface board.
R[7:0]	Red output signals to D/A converter and to the interface board.
RED, **GREEN**, **BLUE**	Analog output from D/A converter for red, blue, and green signals to VGA connector.
TSSCLK	Clock output to touchscreen controller.

Table 3-7 Display interface signals (continued)

Signal	Description
TSMOSI	Data from touchscreen controller.
TSMISO	Data to touchscreen controller.
TSnDAV	Touchscreen controller data available signal.
TSnPENIRQ	Touchscreen controller pen down interrupt to the secondary interrupt controller in the FPGA.
TSnKPADIRQ	Touchscreen controller key pressed interrupt to the secondary interrupt controller in the FPGA.
TSnSS	Touchscreen controller chip select.
Power control signals	The **nLCDIOON**, **CLPOWER**, **PWR3V5VSWITCH**, **VDDNEGSWITCH**, and **VDDPOSSWITCH** signals can be used by the interface board.
LCDID[4:0]	These signals are determined by resistor links on the interface board and indicate the type of display that is attached. The value of these signals can be read from the SYS_CLCD register.
LCDMODE[1:0]	These signals select the VGA display resolution. The signals from the FPGA register SYS_CLCD control remapping of the **CLCD[23:0]** data signals to the **B[7:0]**, **G[7:0]**, and **R[7:0]** signals to the video DAC.
VGA_CLK	The VGA clock synchronizes the conversion of the **B[7:0]**, **G[7:0]**, and **R[7:0]** signals into the **BLUE**, **GREEN**, and **RED** analog signals.
VGA_HSYNC	The VGA horizontal synchronization signal.
VGA_VSYNC	The VGA vertical synchronization signal.

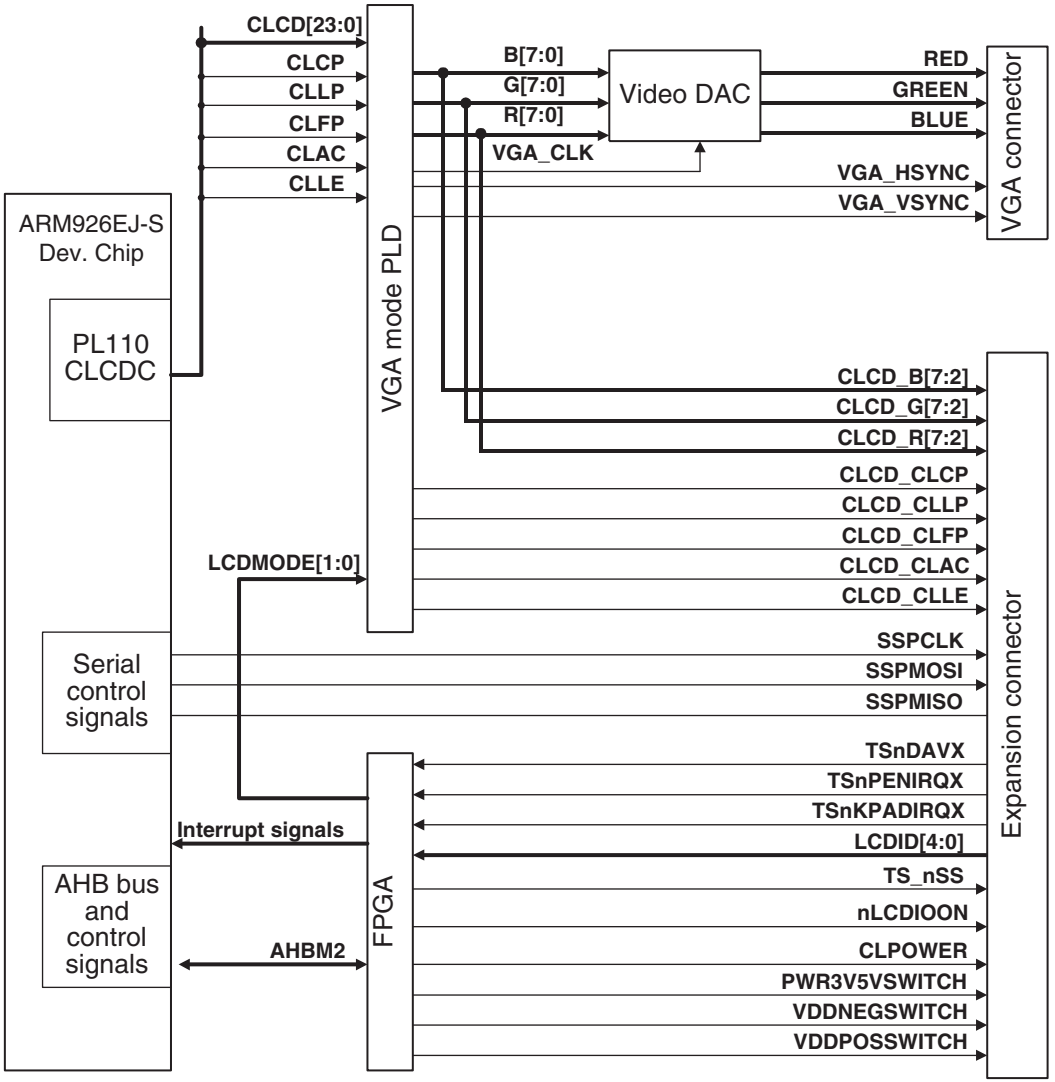

Figure 3-19 Display interface

3.7 DMA

On-chip peripherals in the ARM926PXP development chip use DMA channels 6–15.

DMA control signals 0–2 are passed to the peripherals in the FPGA. Figure 3-20 shows the DMA architecture.

See also *Direct Memory Access Controller* on page 4-40.

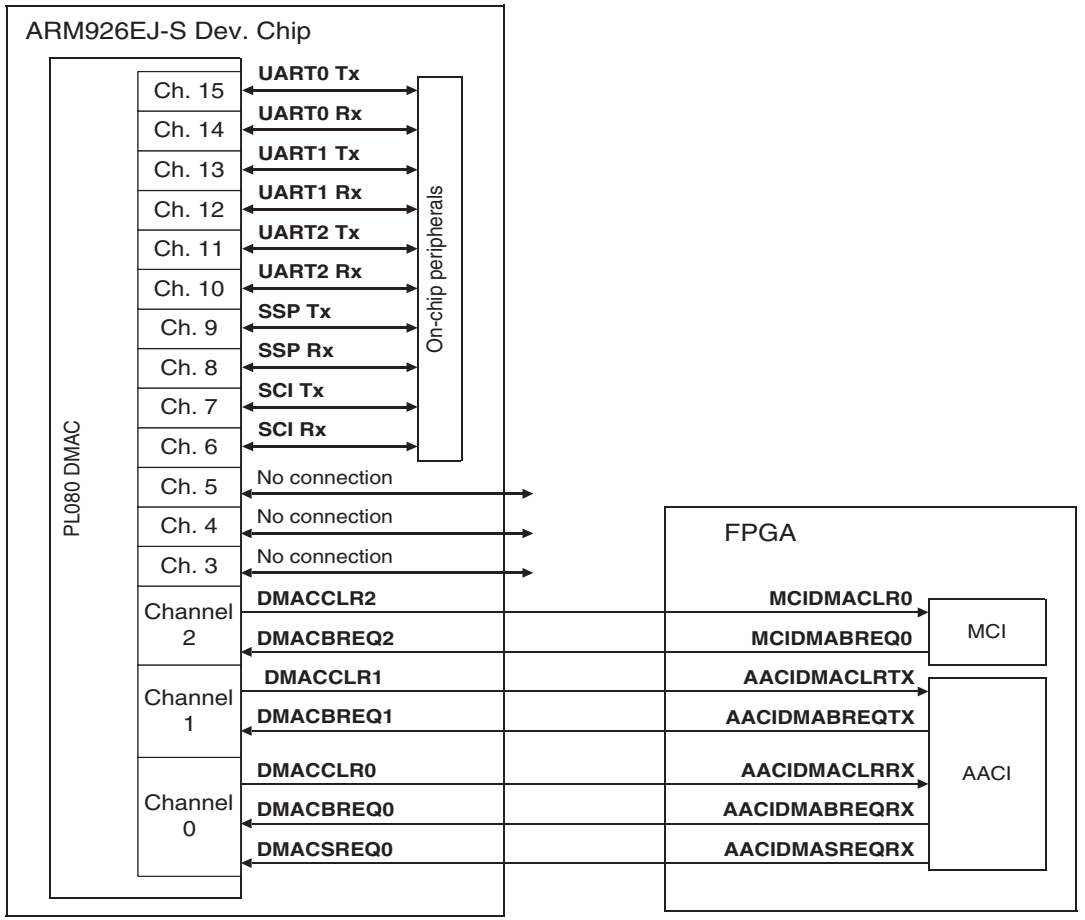

Figure 3-20 DMA channels

The DMA control signals for external devices are listed in Table 3-8 on page 3-44.

Note

None of the signals for DMA channels 3,4, or 5 are used.

The FPGA peripherals do not use all of the DMA control signals for channels 0, 1, and 2.

The names of DMA control signals change as they pass through the mapping logic in the FPGA.

Table 3-8 DMA signals for external devices

Signal	Description
DMACBREQ[2:0]	*Burst request* inputs to DMAC**DMACBREQ[1:0]** are used by the AACI**DMACBREQ[2]** is used by the MCI
DMACLBREQ[2:0]	*Last burst request* inputs to DMAC (not available for use)
DMACSREQ[2:0]	*Single request* inputs to DMAC **DMACSREQ[0]** is used by the AACI**DMACSREQ[2:1]** are not available for use
DMACLSREQ[2:0]	*Last single request* inputs to DMAC (not available for use)
DMACCLR[2:0]	*Clear* outputs from DMAC. These signals acknowledge the request from the corresponding **DMASREQ** or **DMABREQ** signals**DMACCLR[1:0]** are used by the AACI**DMACCLR[2]** is used by the MCI
DMACTC[2:0]	*Terminal count* outputs from DMAC (not available for use)

3.8 Ethernet interface

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

The internal registers of the LAN91C111 are memory-mapped onto the AHBM2 bus and occupy 16 word locations at `0x10010000`.

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

The architecture of the Ethernet interface is shown in Figure 3-21.

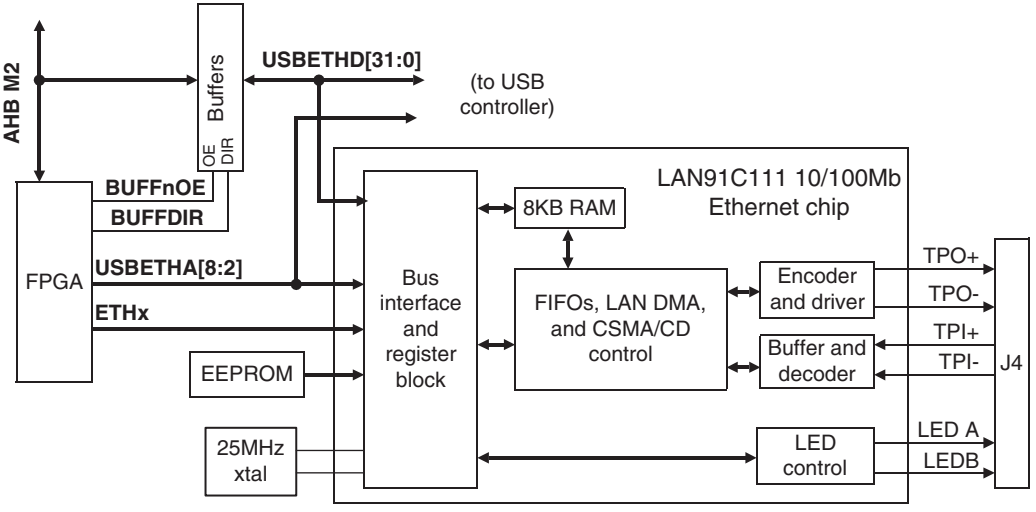

Figure 3-21 Ethernet interface architecture

Table 3-9 Ethernet signals

Signal	Description
USBETHD[31:0]	Data lines to USB and Ethernet controllers
USBETHA[8:2]	Address lines to USB and Ethernet controllers
ETHA[15:13]	Address lines to Ethernet controller
ETHnBE[3:0]	Byte-enable signals to Ethernet controller
TPO+, **TPO-**	Signal from controller to Ethernet interface
TPI+, **TPI-**	Signal from Ethernet interface to controller
LEDA, **LEDB**	Activity indicator LEDs. The function of the LEDs can be configured by writing to a LAN91C111 register.
ETHRESET	Reset signal to LAN91C111 (buffered and inverted **nRESET**)
ETHARDY	Asynchronous ready signal
ETHSRDY	Synchronous ready signal
ETHnRDYRTN	Signals to the controller to complete synchronous read cycles
ETHnADS	Latches address to controller
LCLK	Clock to controller interface
ETHnRD	Read signal for asynchronous interface
ETHnWR	Write signal for asynchronous interface
ETHnDATACS	Enables accesses to the controller data path
ETHnCYCLE	Used to control EISA bust mode synchronous cycles if LOW
ETHnLDEV	Asserted LOW if AEN is low and the address lines decode to the controller address programmed into the base address register
ETHWnR	Defines bus direction for synchronous accesses
ETHnVLBUS	This signal is connected to ground by a pull-down resistor. If LOW, the controller uses VL bus accesses. If HIGH, the controller uses EISA DMA accesses.

3.8.1 About the SMSC LAN91C111

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

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

The controller uses a local VL-Bus host interface with a bridge to the AHB bus provided by the FPGA. The FPGA generates the appropriate access control signals for the host side of the Ethernet controller. The VL-Bus is a synchronous bus that supports 32-bit accesses.

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

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

- the individual MAC address, that is, the Ethernet MAC address
- MII interface configuration
- register base address.

When the Versatile/AB926EJ-S is manufactured, an ARM value for the Ethernet MAC address and the register base address are loaded into the EEPROM. The register base address is 0. The MAC address is unique, but can be reprogrammed if required. Reprogramming of the EEPROM is done through Bank 1 (general and control registers).

3.9 GPIO interface

The GPIO signals **GP0[7:0]** and **GP1[7:0]** from the ARM926PXP development chip are connected to the interface connector as shown in Figure 3-22. This enables you to use the GPIO signals with custom logic you implement in an interface board.

See also *General Purpose Input/Output, GPIO* on page 4-43, the *ARM926EJ-S Development Chip Reference Manual* and the *ARM PrimeCell GPIO (PL061) Technical Reference Manual*.

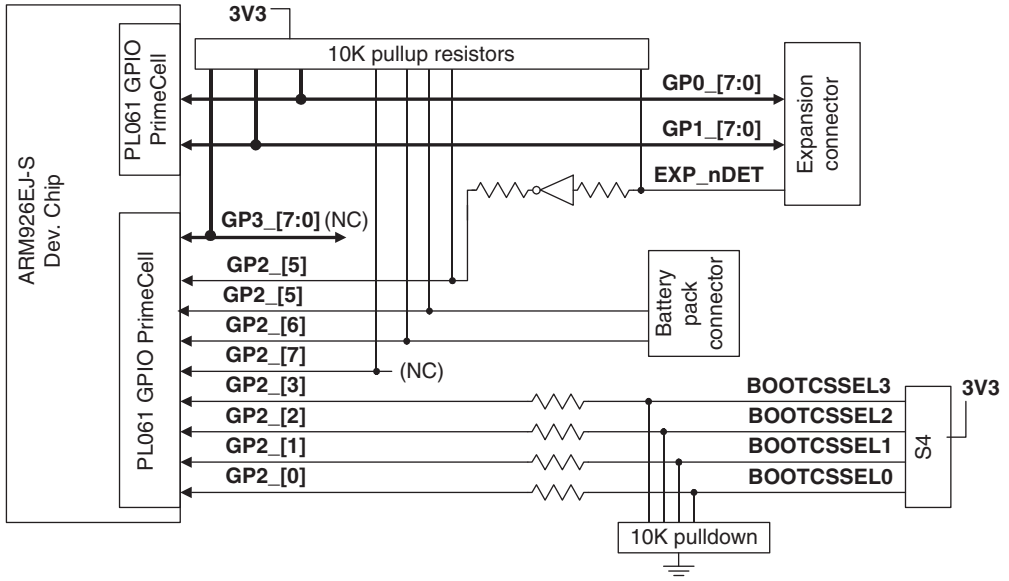

Figure 3-22 GPIO block diagram

Note

The GPIO signals **GP2_[3:0]** are connected to the boot select switch S4.

GP2_[4] is HIGH if an interface board is connected.

GP2_[6:5] go to the battery pack connector.

There are no external connections for **GP3** from the ARM926PXP development chip.

3.10 Interrupts

The ARM926PXP development chip contains the primary interrupt controller and a secondary interrupt controller is in the FPGA, see Figure 3-23.

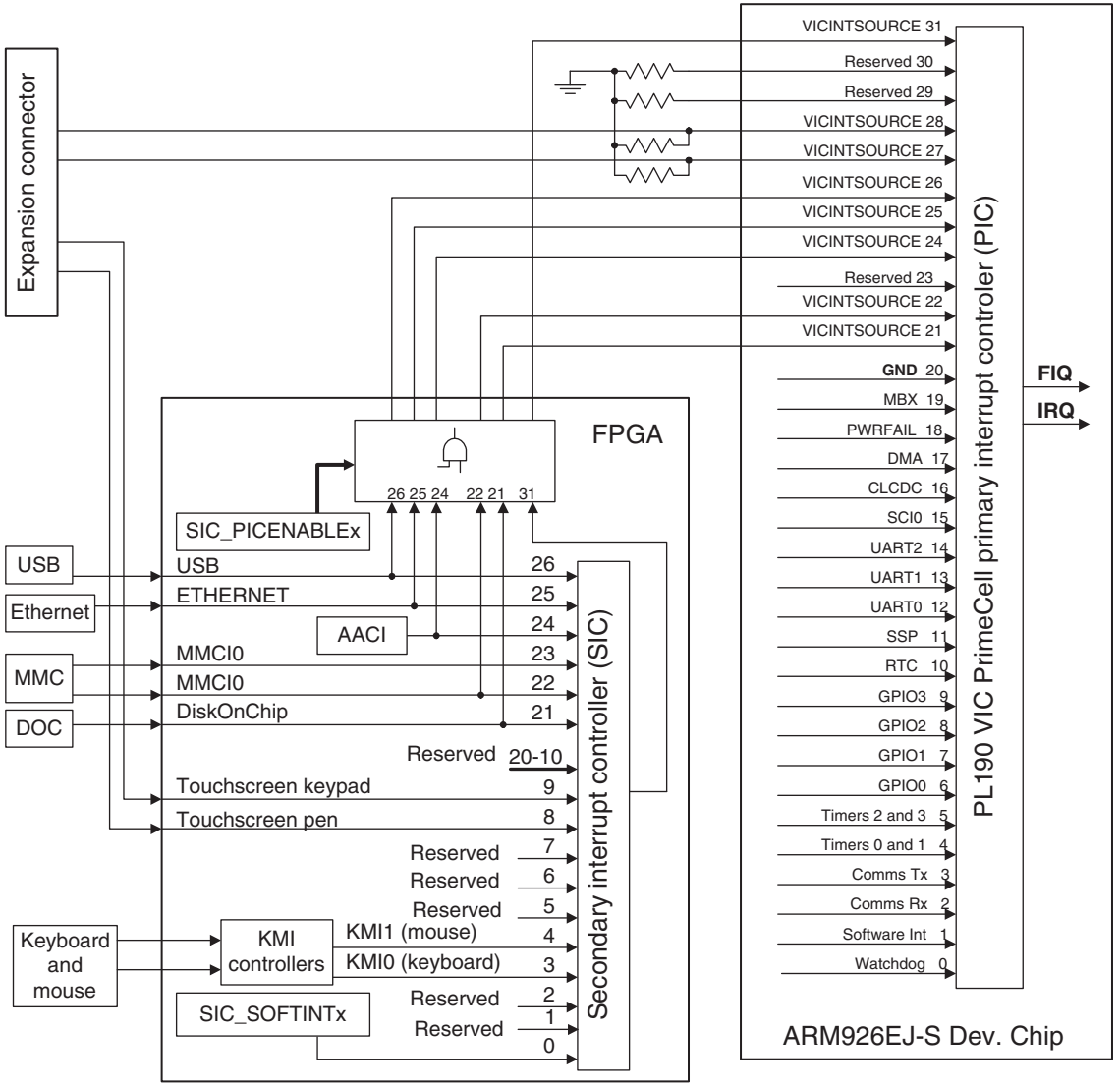

Figure 3-23 External and internal interrupt sources

The primary interrupt controller manages interrupts from internal devices and provides six pins for use by the external secondary interrupt controller present in the FPGA. **VICINTSOURCE31** is the output from the secondary controller. **VICINTSOURCE[26:21]** can be driven from individual interrupt signals from peripherals in the FPGA. **VICINTSOURCE[28:27]** can be driven from the interface connector.

For details on the programming model for the interrupt controllers, see:

- the *ARM926EJ-S Development Chip Reference* manual
- the *ARM PrimeCell Vector Interrupt Controller (PL190) Technical Reference Manual* manual
- *Primary interrupt controller* on page 4-45.

3.11 Keyboard/Mouse Interface, KMI

The *Keyboard and Mouse Interfaces* (KMI) are implemented with two PrimeCells incorporated into the system controller FPGA. This is shown in Figure 3-24.

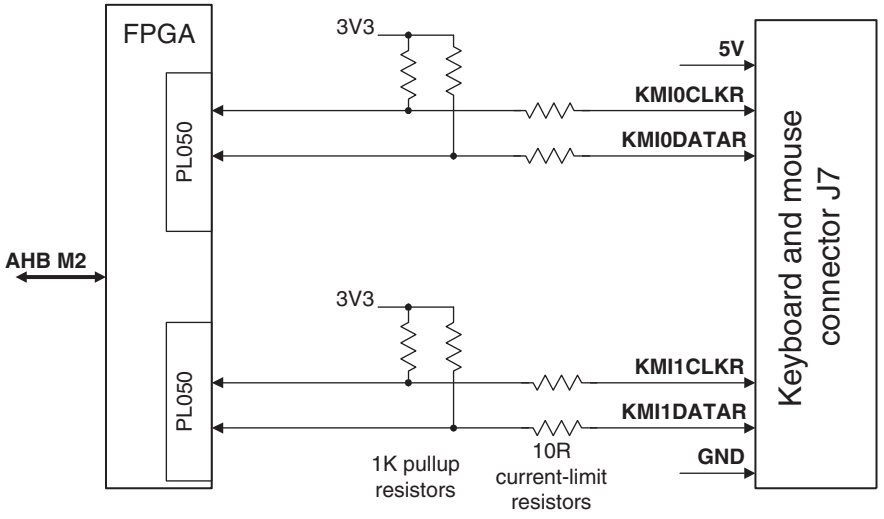

Figure 3-24 KMI block diagram

See also *Keyboard and Mouse Interface, KMI* on page 4-53 and the *ARM PrimeCell PS2 Keyboard Mouse Controller (PL050) Technical Reference Manual.*

Note

By default, the KMI connector will work directly with either a mouse or keyboard connected. The signals are connected to KMI0.

Plug a keyboard/mouse splitter into J7 to enable a mouse and keyboard to be used simultaneously.

3.12 SD/MultiMedia Card Interface, MCI

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

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

3.12.1 MMC or SD operation

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

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

The function of the interface signals depends on whether an MMC or SD card is fitted. Both card types default to MMC mode but the SD card has an additional operating mode called widebus mode. Table 3-10 shows the use of the signals for both modes of operation.

Table 3-10 MMC/SD interface signals

Signal	Widebus mode (SD only)	MMC mode (default for MMC and SD cards)
MCIDAT3	card detect/data (3)	chip select (active LOW)
MCICMD	command/response	command/response
MCICLK	clock	clock
MCIDAT0	data (0)	data
MCIDAT1	data (1)	not used
MCIDAT2	data (2)	not used
nMCICARDIN	card presence detect (active LOW)	card presence detect (active LOW)
WPROT	card write-protection detect	card write-protection detect
nMCIVDDO	power enable signal	power enable signal

3.12.2 Card insertion and removal

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

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

3.12.3 Card interface description

Figure 3-25 shows the MMC card interface.

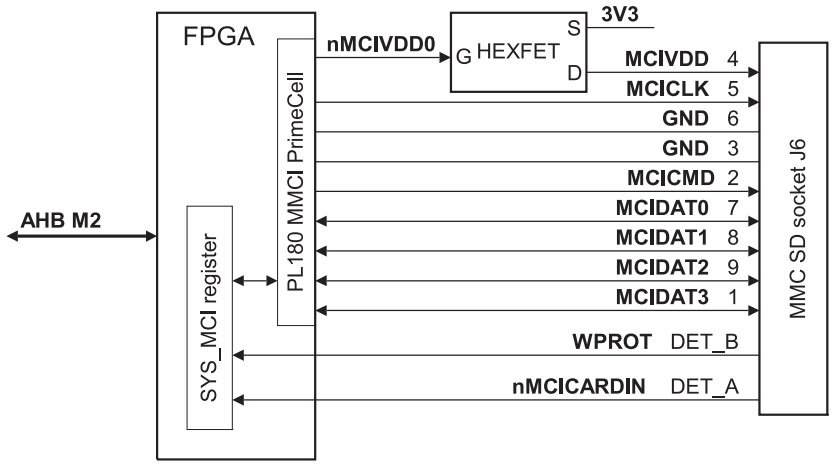

Figure 3-25 MMC interface

See *MMC and SD flash card interface* on page A-11 for details of the MMC/SD card socket and pin numbering. See also *MultiMedia Card Interface, MCI* on page 4-55 and the *ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual*.

Table 3-11 MMC signals

Signal	Description
MCIPWR	Enables supply voltage to card
MCICMD	Command selection
nCARDIN	Card detect signal. Read the current state from SYS_MCI.
MCIDAT[3:0]	Card data bus
WPROT	Write protection indication.
MCIMCLK	Clock to card

3.13 Serial bus interface

The FPGA implements a serial bus interface that is used to identify devices connected to the expansion connector and read and set the time-of-year clock.

Each device on the serial bus has its own slave address. The unique address for each slave on the serial bus is shown in Table 3-12.

Table 3-12 Serial bus addresses

Slave address (7-bit)	Slave device
b1010001	Static memory module (if present, this is located on the Versatile/AB-IB1 or Versatile/AB-IB2 interface board). See Appendix F *Static Memory Expansion Board* for details of the serial bus interface to the memory module EEPROM.
b1101000	Time-of-year clock

The block diagram of the interface is shown in Table 3-12.

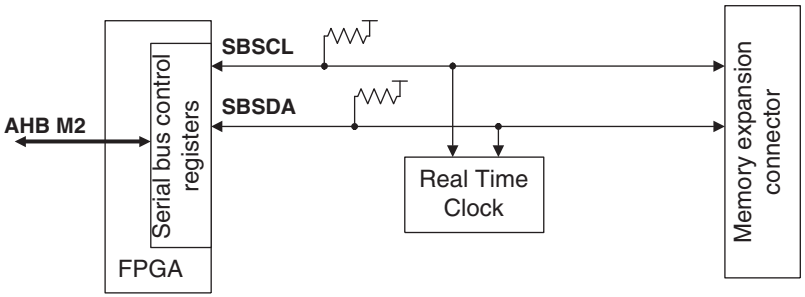

Figure 3-26 Serial bus block diagram

Table 3-13 Serial bus signals

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

3.14 Smart card interface, SCI

The ARM926PXP development chip contains a PrimeCell *Smart Card Interface* (SCI) controller.

A Smart Card connector (J10) is fitted to the underside of the board. There are solder pads that can be used for a smaller connector (J9) in place of the larger connector. Figure 3-27 shows the tristate buffers that are used to provide the interface between the SCI and the card.

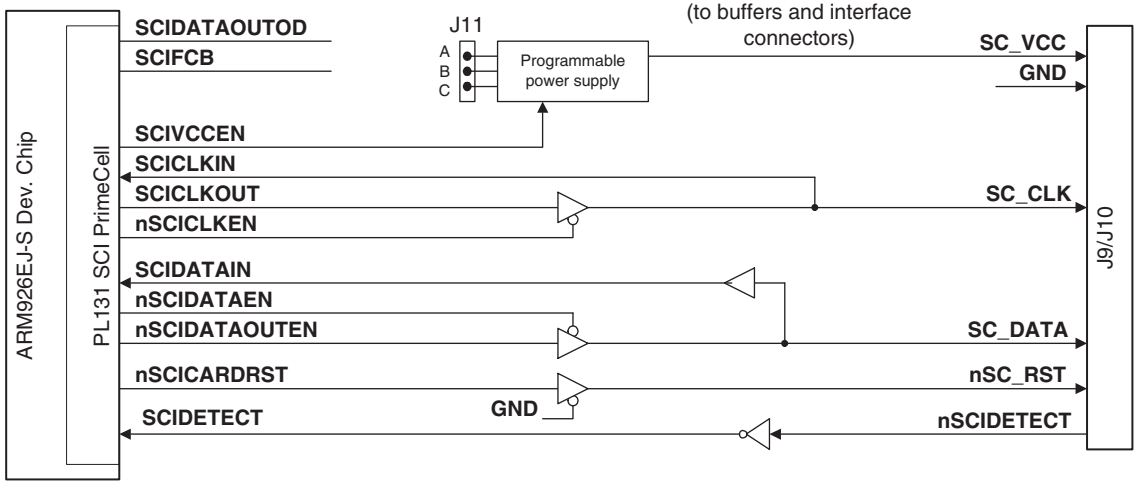

Figure 3-27 SCI block diagram

You can set the Smart Card interface voltage to operate at 5V, 3.3V or 1.8V by setting jumpers on J11.

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

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

Note

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

See also *Smart Card Interface, SCI* on page 4-61 and the *SCI PrimeCell PL131 Technical Reference Manual*.

Table 3-14 Smart Card interface signals

Signal	Description
SCICLKOUT	PrimeCell SCI clock.
nSCICLKEN	Tristate output buffer control for clock (active LOW).
SC_CLK	Clock output to card.
nSCIDATAEN	Tristate control for external off-chip buffer (active LOW).
SCIDATAIN	PrimeCell SCI serial data input.
nSCIDATAOUTEN	Data output (typically drives an open-drain buffer and functions as a data enable, active LOW).
nSC_RST	Reset to card (active LOW).
SCIFCB	Function code bit, not used in Versatile/AB926EJ-S but for other products it is typically used in conjunction with **nSCICARDRST**.
SCIDETECT	Card detect signal from card interface device (active HIGH).

3.15 Synchronous Serial Port, SSP

The ARM926PXP development chip contains a PrimeCell SSP controller that is accessible from the peripheral expansion connector as shown in Figure 3-28.

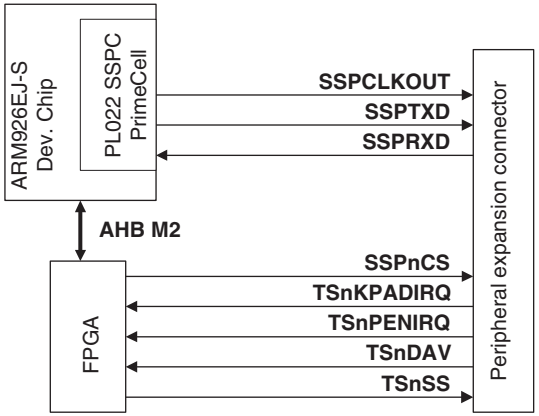

Figure 3-28 SSP block diagram

Table 3-15 SSP signal descriptions

Name	Description
SSPnCS	Chip select to external device connected to SSP controller (see *CLCD Control Register, SYS_CLCD* on page 4-28)
SSPCLKOUT	PrimeCell SSP clock output to peripheral interface board
SSPRXD	PrimeCell SSP receive data input.
SSPTXD	PrimeCell SSP transmit data output
TSnSS	Chip select to peripheral interface board (see *CLCD Control Register, SYS_CLCD* on page 4-28)
TSnKPADIRQX	Keypad interrupt signal from interface board
TSnPENIRQ	Pen interrupt signal from interface board
TSnDAV	Data available from interface board (see *CLCD Control Register, SYS_CLCD* on page 4-28)

The SSP functions as a master interface.See also *Synchronous Serial Port, SSP* on page 4-62 and the *ARM PrimeCell Synchronous Serial Port Controller (PL022) Technical Reference Manual.*

3.16 User switches and LEDs

The FPGA provides a switch and LED register that enables you to read the general-purpose pushbutton switch and user switches (S1) and light the user LEDs (located next to switch S1). See Figure 1-1 on page 1-3 for the location of the switches and LEDs. Figure 3-29 shows the interface.

Set bits [7:0] in the SYS_LED register at `0x10000008` to illuminate LEDs 7–0. The state of the user switches is present on bits [7:0] of the SYS_SW register at `0x10000004`. The state of the general-purpose pushbutton can be read from the SYS_MISC register.

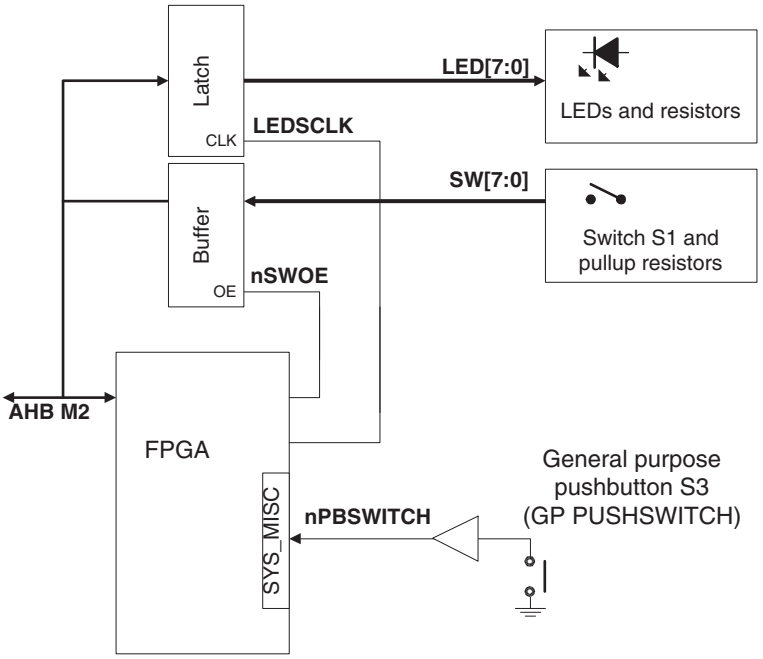

Figure 3-29 Switch and LED interface

3.17 USB interface

The FPGA provides the bus interface to an external OTG243 USB controller. A block diagram of the USB interface is shown in Figure 3-30.

The internal registers of the controller are memory-mapped onto the AHBM2 bus at `0x10020000`.

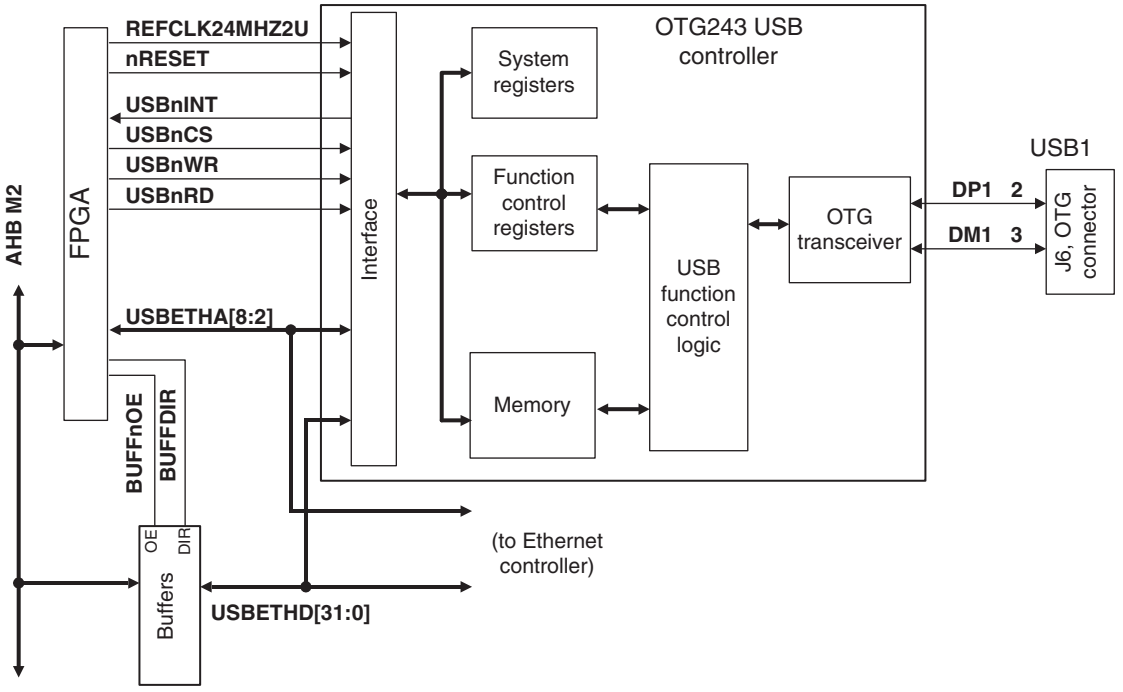

Figure 3-30 OTG243 block diagram

OTG243 USB ports 2 and 3 are not used on the Versatile/AB926EJ-S.

The signals associated with the USB interface is shown in Table 3-16.

Table 3-16 USB interface signal assignment

Signal name	Description
DPx	D+ data line
DMx	D– data line
USBETHD[31:0]	Data lines of USB controller
USBETHA[8:2]	Address lines of USB controller
USBOT[1:0]	DMA end of transfer for channel 1 and 0
USBnCS	Controller chip select
USBnRD	Read strobe to controller
USBnWR	Write strobe to controller
USBnINT	Controller interrupt out
nRESET	Controller reset
USBWAKEUP	FPGA drives this signal HIGH to wake up the controller
REFCLK24MHZ2U	24MHz reference clock to controller
nOC	Over current detect (disconnects power to USB2 and USB3)
USBEOT[0]	DMA end of transfer.
USBDRQ[0]	DMA request.
USBDACK[0]	DMA acknowledge.
nEXVBO	Connects additional power to the OTG (VBUS)
VBP	Connects additional power to the OTG (VBUS)
VBUS	If the OTG is in slave mode, this is the incoming 5V digital power supply from the cable.

Note

For a full description, refer to the datasheet for the TransDimension OTG243.

3.18 UART interface

Three UARTs (SER0, SER1, and SER2) are provided by the ARM926PXP development chip.

The UARTs have the following features:

- port function corresponds to the DTE configuration
- SER1 and SER2 (UART1 and UART2) have only **CTS**, **RTS**, **TXD** and **RXD** signals.
- SER0 (UART0) has **CTS**, **RTS**, **DCD**, **DSR**, **DTR**, and **RI** control signals
- programmable baud rates of up to 1.5Mbits per second (the line drivers however, are only guaranteed to 250kbps)
- 16-byte transmit FIFO
- 16-byte receive FIFO
- four programmable interrupts.

Signals from UART1 and UART2 go to the expansion connector. The signals from UART0 are converted from logic level to RS232 level by MAX3243E buffers as shown in Figure 3-31.

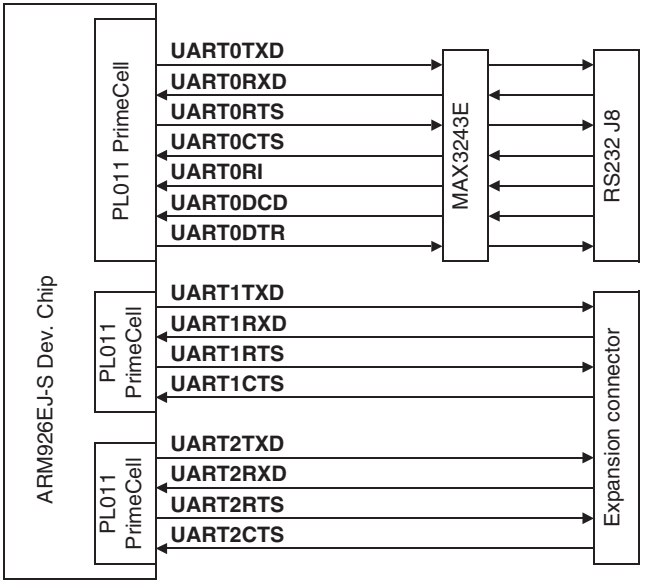

Figure 3-31 UARTs block diagram

See also *UART* on page 4-68 and the *ARM PrimeCell UART (PL011) Technical Reference Manual*.

The signals associated with the UART interface are shown in Table 3-17.

Table 3-17 Serial interface signal assignment

Signal	Description
UARTxTXD	Transmit data
UARTxRTS	Ready to send
UARTxDTR	Data terminal ready
UARTxCTS	Clear to send
UARTxDSR	Data set ready
UARTxDCD	Data carrier detect
UARTxRXD	Receive data
UARTxRI	Ring indicator

3.19 Test, configuration, and debug interfaces

The following debug methods are available for the Versatile/AB926EJ-S:

- The JTAG interface provides a connection to external JTAG debuggers such as RVI or Multi-ICE. If you are using an external JTAG debug tool, the embedded debug hardware is disabled.
- The onboard embedded USB debug port hardware enables a direct USB connection from the host PC to the Versatile/AB926EJ-S.

The following test and configuration interfaces are located on the Versatile/AB926EJ-S:

- JTAG, see *JTAG and USB debug port support*
- Trace, see *Embedded trace support* on page 3-71
- Configuration switches and status indicators, see *Configuration control* on page 3-7 and *User switches and LEDs* on page 3-59.
- Boot Monitor, see *Using the Versatile/AB926EJ-S Boot Monitor* on page 2-11.

Note

There are also test points and debug connectors for individual interface circuits. See *Test and debug connections* on page A-18.

3.19.1 JTAG and USB debug port support

The Versatile/AB926EJ-S supports debugging using embedded or external hardware. The debugging interface can be controlled by:

JTAG hardware

The RVI and AXD debuggers, for example, use an external interface box, such as RVI or Multi-ICE, to connect to the JTAG connector.

USB debug port

The USB debug port is embedded on the Versatile/AB926EJ-S. An application, Progcards or the RealView Debugger, for example, can control the JTAG signals from the USB port of the PC. The PC and the Versatile/AB926EJ-S are connected by a standard USB cable.

Note

The use of the RealView Debugger and the USB debug port is only supported by RVDS 2.1 and later.

The USB debug port is enabled unless the **nICEDETECT** signal (from pin 20 of the JTAG connector) is pulled LOW. ARM Multi-ICE and RVI ground pin 20 and disable the USB debug port automatically. (See Figure 3-33 on page 3-70.)

If you are using third-party debugging hardware, ensure that a ground is present on pin 20 of the JTAG connector.

The JTAG interface is described in the following subsections:

- *JTAG debug (normal) mode*
- *JTAG configuration mode*
- *Debug comms channel* on page 3-66
- *JTAG and USB debug port connections* on page 3-66
- *JTAG signals* on page 3-67.

JTAG debug (normal) mode

During normal operation and software development, the Versatile/AB926EJ-S operates in debug mode. The debug mode is selected by default (when a jumper is not fitted on the CONFIG link (or if a CONFIG switch is used, the switch in is the OFF position), see Figure 3-32 on page 3-67). In this mode, the ARM926EJ-S processor is accessible on the scan chain.

JTAG configuration mode

In configuration mode the debuggable devices are still accessible (through a TAP port on the processor however instead of the processor debug port that is used in debug mode) and, in addition, all FPGAs and PLDs in the system are added into the scan chain. This enables the board to be configured or upgraded in the field using JTAG equipment or the onboard USB debug port.

To select configuration mode, fit a jumper to the CONFIG link (see Figure 3-32 on page 3-67*)*. This has the effect of pulling the **nCFGEN** signal LOW. **nCFGEN** reroutes the JTAG scan path and illuminates the CFGEN LED on the Versatile/AB926EJ-S. The LED provides an indication that the development system is in the configuration mode.

Configuration mode enables the FPGA and PLD images to be updated as follows:

- The FPGAs are volatile. In normal mode, they load their configuration from nonvolatile flash memory. In configuration mode, they are initially loaded from the configuration flash memory but can be manually reloaded from JTAG.

The configuration flash memory does not have a JTAG port, but it can be programmed using JTAG to load a flash-loader design into the FPGAs and PLDs. The flash-loader design can then transfer data from the JTAG programming utility to the configuration flash.

- The PLDs are nonvolatile devices that can be programmed directly by JTAG.

After configuration you must:

1. Remove the CONFIG link.

   Note

   The CONFIG link is a switch on some board versions. If present, turn the switch to OFF,

2. Power cycle the development system.

Debug comms channel

The ARM926EJ-S processor incorporates EmbeddedICE logic and provides debug communications data registers that are used to pass data between the processor and JTAG equipment. See the *ARM926EJ-S Technical Reference Manual* for a description of the debug communications channel.

JTAG and USB debug port connections

Figure 3-32 on page 3-67 shows the CFGEN link, CONFIG LED, JTAG, USB debug port, and Trace Port connectors.

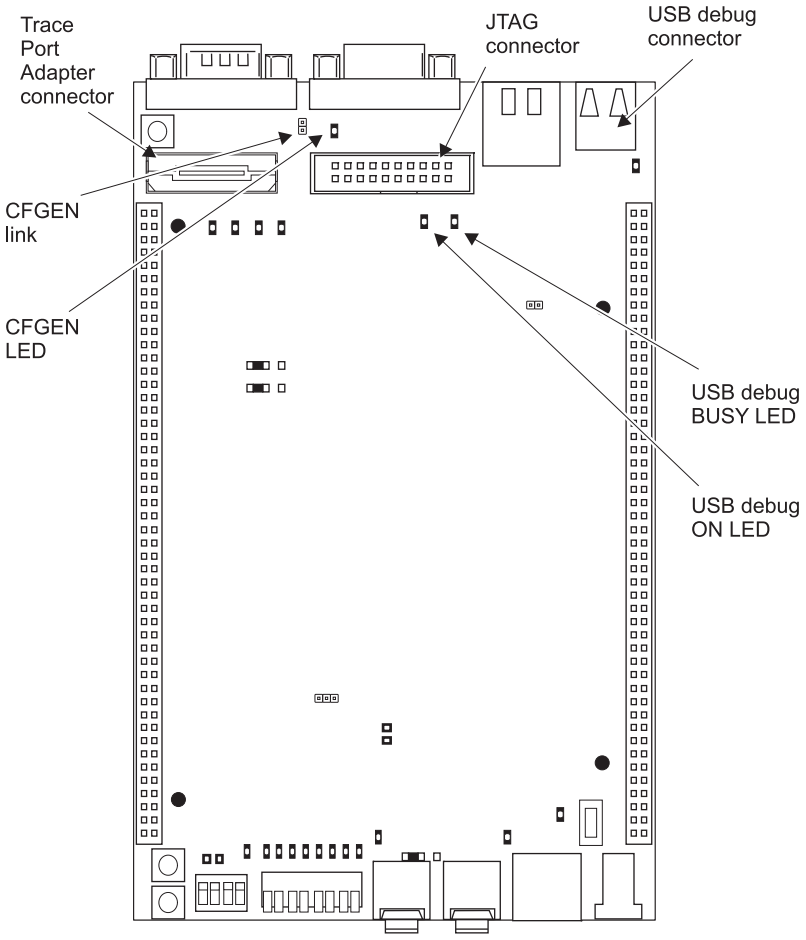

Figure 3-32 JTAG connector, CFGEN link, and LED

JTAG signals

Table 3-18 on page 3-68 provides a description of the JTAG and related signals.

Note

In the description in Table 3-18 on page 3-68, the term JTAG equipment refers to any hardware that can drive the JTAG signals to devices in the scan chain. Typically this is RVI, Multi-ICE, or the embedded USB debug logic.

Table 3-18 JTAG related signals

Name	Description	Function
TDI	Test data in (from JTAG equipment)	**TDI** and **TDO** connect each component in the scan chain.
TDO	Test data out (to JTAG equipment)	**TDO** is the return path of the data input signal **TDI**.
TMS	Test mode select (from JTAG equipment)	**TMS** controls transitions in the tap controller state machine.
TCK	Test clock (from JTAG equipment)	**TCK** synchronizes all JTAG transactions. **TCK** connects to all JTAG components in the scan chain. Series termination resistors are used to reduce reflections and maintain good signal integrity.
RTCK	Return **TCK** (to JTAG equipment)	Some devices sample **TCK** and delay the time at which a component actually captures data. Using a mechanism called *adaptive clocking*, the **RTCK** signal is returned by the core to the JTAG equipment, and the TCK is not advanced until the core has captured the data. In adaptive clocking mode, RVI or Multi-ICE waits for an edge on **RTCK** before changing **TCK**. In a multiple device JTAG chain, the **RTCK** output from a component connects to the **TCK** input of the next device in the chain.
nCFGEN	Configuration enable	**nCFGEN** is an active LOW signal used to put the boards into configuration mode. In configuration mode all FPGAs and PLDs are connected to the scan chain so that they can be configured by the JTAG equipment.
nSRST	System reset (bidirectional)	**nSRST** is an active LOW open-collector signal that can be driven by the JTAG equipment to reset the target board. Some JTAG equipment senses this line to determine when a board has been reset by the user. This is also used in configuration mode to control the initialization pin (**nINIT**) on the FPGAs. Though not a JTAG signal, **nSRST** is described because it can be controlled by JTAG equipment.
nTRST	Test reset (from JTAG equipment)	This active LOW open-collector is used to reset the JTAG port and the associated debug circuitry on the ARM926PXP development chip. It is asserted at power-up and can be driven by the JTAG equipment. This signal is also used in configuration mode to control the programming pin, **nPROG**, on FPGAs.

Table 3-18 JTAG related signals (continued)

Name	Description	Function
DBGRQ	Debug request (from JTAG equipment)	**DBGRQ** is a request for the processor core to enter the debug state. It is provided for compatibility with third-party JTAG equipment.
DBGACK	Debug acknowledge (to JTAG equipment)	**DBGACK** indicates to the debugger that the processor core has entered debug mode. It is provided for compatibility with third-party JTAG equipment.
GLOBAL_DONE	FPGA configured	**GLOBAL_DONE** is an open-collector signal that indicates when FPGA configuration is complete. Although this signal is not a JTAG signal, it does affect **nSRST**. The **GLOBAL_DONE** signal is routed between all FPGAs.

The JTAG path chosen depends on whether the system is in configuration mode or debug mode. The CONFIG link (or switch) controls the **nCFGEN** signal that is routed to the Versatile/AB926EJ-S, FPGAs, PLDs, and the expansion connectors. if **nCFGEN** is HIGH, only the ARM926EJ-S development chip is in the scan chain.

Figure 3-33 on page 3-70 and Figure 3-34 on page 3-71 show the JTAG signal routing. If an expansion board is present, the **TDI** and **TDO** JTAG signals are routed through the board. If not, the signals bypass the expansion connector.

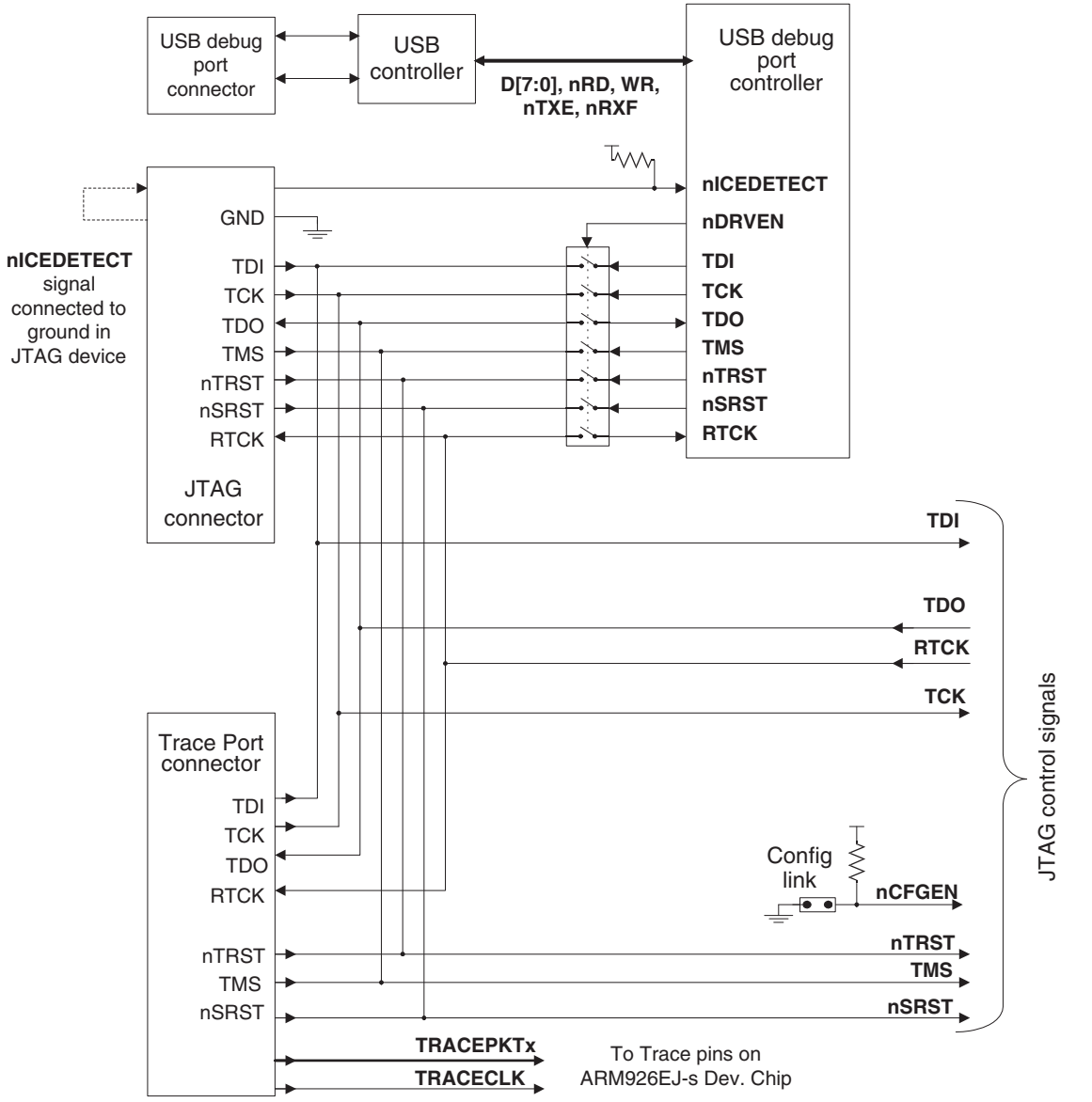

Figure 3-33 JTAG connector signals

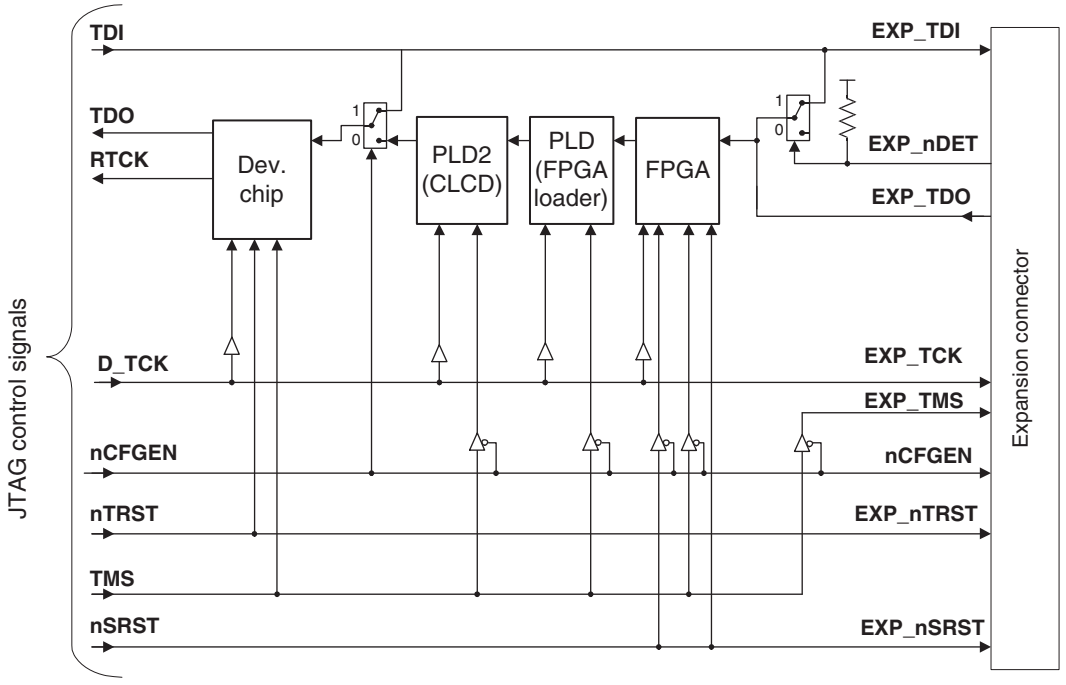

Figure 3-34 JTAG signal routing

3.19.2 Embedded trace support

The ARM926EJ-S processor incorporates an *ARM9 Embedded Trace Macrocell* (ETM9). This enables you to carry out real-time debugging by connecting external trace equipment to the Trace connector on the Versatile/AB926EJ-S. To trace program flow, the ETM broadcasts branch addresses, data accesses, and status information through the trace port. Later in the debug process, the complete instruction flow can be reconstructed (see *Connecting the Trace Port Analyzer* on page 2-7).

Note

The routing of the JTAG scan chain is described in *JTAG and USB debug port support* on page 3-64. Connection of the trace port analyzer is described in *Connecting the Trace Port Analyzer* on page 2-7.

Chapter 4
Programmer's Reference

This chapter describes the memory map and the configuration registers for the peripherals in the ARM926PXP development chip and the Versatile/AB926EJ-S FPGA. It contains the following sections:

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.

4.1 Memory map

The locations for memory, peripherals, and controllers are listed in Table 4-1.

There are multiple buses in the ARM926PXP development chip. Not all of the buses can access all of the memory regions. See *AHB bridges and the bus matrix* on page 3-8 and the *ARM926EJ-S Reference Manual* for details on the bus matrix and bus accesses.

Note

The MOVE and VFP coprocessors are not memory-mapped peripherals so they do not appear in the memory map listed in Table 4-1. See the appropriate technical reference manual for more detail on these devices.

Table 4-1 Memory map

Peripheral	Location	Interrupt[a] PIC and SIC	Address	Region size
MPMC Chip Select 0. Normally the bottom 64MB of SDRAM (During boot remapping, this can be NOR flash, Disk-on-Chip, or static expansion memory) **Note** Interrupts are for Disk-on-Chip only.	Board	PIC21, SIC21	0x00000000-0x03FFFFFF	64MB
MPMC Chip Select 0, top 64MB of SDRAM	Board	-	0x04000000-0x07FFFFFF	64MB
Reserved (MPMC Chip Select 1, dynamic expansion memory on Versatile/PB926EJ-S)	-	-	0x08000000-0x0FFFFFFF	128MB
System registers	FPGA	-	0x10000000-0x10000FFF	4KB
Reserved (PCI controller configuration registers on Versatile/PB926EJ-S)	-	-	0x10001000-0x10001FFF	4KB
Serial Bus Interface	FPGA	-	0x10002000-0x10002FFF	4KB
Secondary Interrupt Controller (SIC)	FPGA	PIC 31	0x10003000-0x10003FFF	4KB

Table 4-1 Memory map (continued)

Peripheral	Location	Interrupt[a] PIC and SIC	Address	Region size
Advanced Audio Codec Interface	FPGA	PIC24, SIC 24	0x10004000-0x10004FFF	4KB
Multimedia Card Interface 0 (MMCI0)	FPGA	MCI0A: PIC 22, SIC 22 MCI0B: SIC 1	0x10005000-0x10005FFF	4KB
Keyboard/Mouse Interface 0 (keyboard)	FPGA	SIC 3	0x10006000-0x10006FFF	4KB
Keyboard/Mouse Interface 1 (mouse)	FPGA	SIC 4	0x10007000-0x10007FFF	4KB
Reserved (character LCD Interface on Versatile/PB926EJ-S)	-	-	0x10008000-0x10008FFF	4KB
Reserved (UART3 Interface on Versatile/PB926EJ-S)	-	-	0x10009000-0x10009FFF	4KB
Reserved (Smart Card on Versatile/PB926EJ-S)	-	-	0x1000A000-0x1000AFFF	4KB
Reserved (MCI1 on Versatile/PB926EJ-S)	-	-	0x1000B000-0x1000BFFF	4KB
Reserved for future use	-	-	0x1000C000-0x1000FFFF	16KB
Ethernet Interface	FPGA	PIC 25, SIC 25	0x10010000-0x1001FFFF	64KB
USB Interface	FPGA	PIC 26, SIC 26	0x10020000-0x1002FFFF	64KB
Reserved for future use	-	-	0x10030000-0x100FFFFF	832KB (13 * 64KB)
Synchronous Static Memory Controller configuration registers	Dev. chip	-	0x10100000-0x1010FFFF	64KB
Multi-Port Memory Controller configuration registers	Dev. chip	-	0x10110000-0x1011FFFF	64KB
Color LCD Controller	Dev. chip	PIC 16	0x10120000-0x1012FFFF	64KB

Table 4-1 Memory map (continued)

Peripheral	Location	Interrupt[a] PIC and SIC	Address	Region size
DMA Controller	Dev. chip	PIC 17	0x10130000–0x1013FFFF	64KB
Vectored Interrupt Controller (PIC)	Dev. chip	-	0x10140000–0x1014FFFF	64KB
Reserved	-	-	0x10150000–0x101CFFFF	64KB
Reserved (AHB monitor on Versatile/PB926EJ-S)	-	-	0x101D0000–0x101DFFFF	64KB
System Controller	Dev. chip	-	0x101E0000–0x101E0FFF	4KB
Watchdog Interface	Dev. chip	PIC 0	0x101E1000–0x101E1FFF	4KB
Timer modules 0 and 1 interface (Timer 1 starts at 0x101E2020)	Dev. chip	PIC 4	0x101E2000–0x101E2FFF	4KB
Timer modules 2 and 3 interface (Timer 3 starts at 0x101E3020)	Dev. chip	PIC 5	0x101E3000–0x101E3FFF	4KB
GPIO Interface (port 0)	Dev. chip	PIC 6	0x101E4000–0x101E4FFF	4KB
GPIO Interface (port 1)	Dev. chip	PIC 7	0x101E5000–0x101E5FFF	4KB
GPIO Interface (port 2) This interface is dedicated to reading the value of the boot select switches SW4, the battery pack status signals, and detecting whether an interface board is present.	Dev. chip	PIC 8	0x101E6000–0x101E6FFF	4KB
Reserved (GPIO port 3, these signals are not present on the Versatile/AB926EJ-S connectors)	Dev. chip	PIC 9	0x101E7000–0x101E7FFF	4KB
Real Time Clock Interface	Dev. chip	PIC 10	0x101E8000–0x101E8FFF	4KB
Reserved for subsystem use	-	-	0x101E9000–0x101EFFFF	4KB

Table 4-1 Memory map (continued)

Peripheral	Location	Interrupt[a] PIC and SIC	Address	Region size
Smart Card 0 Interface	Dev. chip	PIC 15	0x101F0000-0x101F0FFF	4KB
UART 0 Interface	Dev. chip	PIC 12	0x101F1000-0x101F1FFF	4KB
UART 1 Interface	Dev. chip	PIC 13	0x101F2000-0x101F2FFF	4KB
UART 2 Interface	Dev. chip	PIC 14	0x101F3000-0x101F3FFF	4KB
Synchronous Serial Port Interface	Dev. chip	PIC 11	0x101F4000-0x101F4FFF	4KB
Reserved	-	-	0x101F5000-0x13FFFFFF	78MB
Reserved (Logic Tile expansion on Versatile/PB926EJ-S)	-	-	0x14000000-0x1FFFFFFF	176MB
SSMC Chip Selects 4–7, static expansion memory	Board	-	0x20000000-0x2FFFFFFF	256MB
SSMC Chip Select 0, Disk on Chip	Board	-	0x30000000-0x33FFFFFF	64MB
SSMC Chip Select 1, normally NOR flash (During boot remapping, this can be NOR flash, Disk-on-Chip, or static expansion memory)	Board	-	0x34000000-0x37FFFFFF	64MB
SSMC Chip Select 2, SRAM	Board	-	0x38000000-0x3BFFFFFF	64MB
MBX Graphics Accelerator Interface	Dev. chip	PIC 19	0x40000000-0x40FFFFFF	16MB
Reserved (PCI interface on Versatile/PB926EJ-S)	-	-	0x41000000-0x6FFFFFFF	752MB
Reserved (MPMC Chip Selects 2–3, expansion dynamic memory on Versatile/PB926EJ-S)	-	-	0x70000000-0x7FFFFFFF	256MB
Reserved (Logic Tile expansion on Versatile/PB926EJ-S)	-	-	0x80000000-0xFFFFFFFF	2GB

a. The primary interrupt controller is in the ARM926PXP development chip. The secondary controller is in the FPGA. See *Primary interrupt controller* on page 4-45 and *Interrupt controllers* on page 4-44.

Figure 4-1 on page 4-8 shows the ARM Data bus memory map.

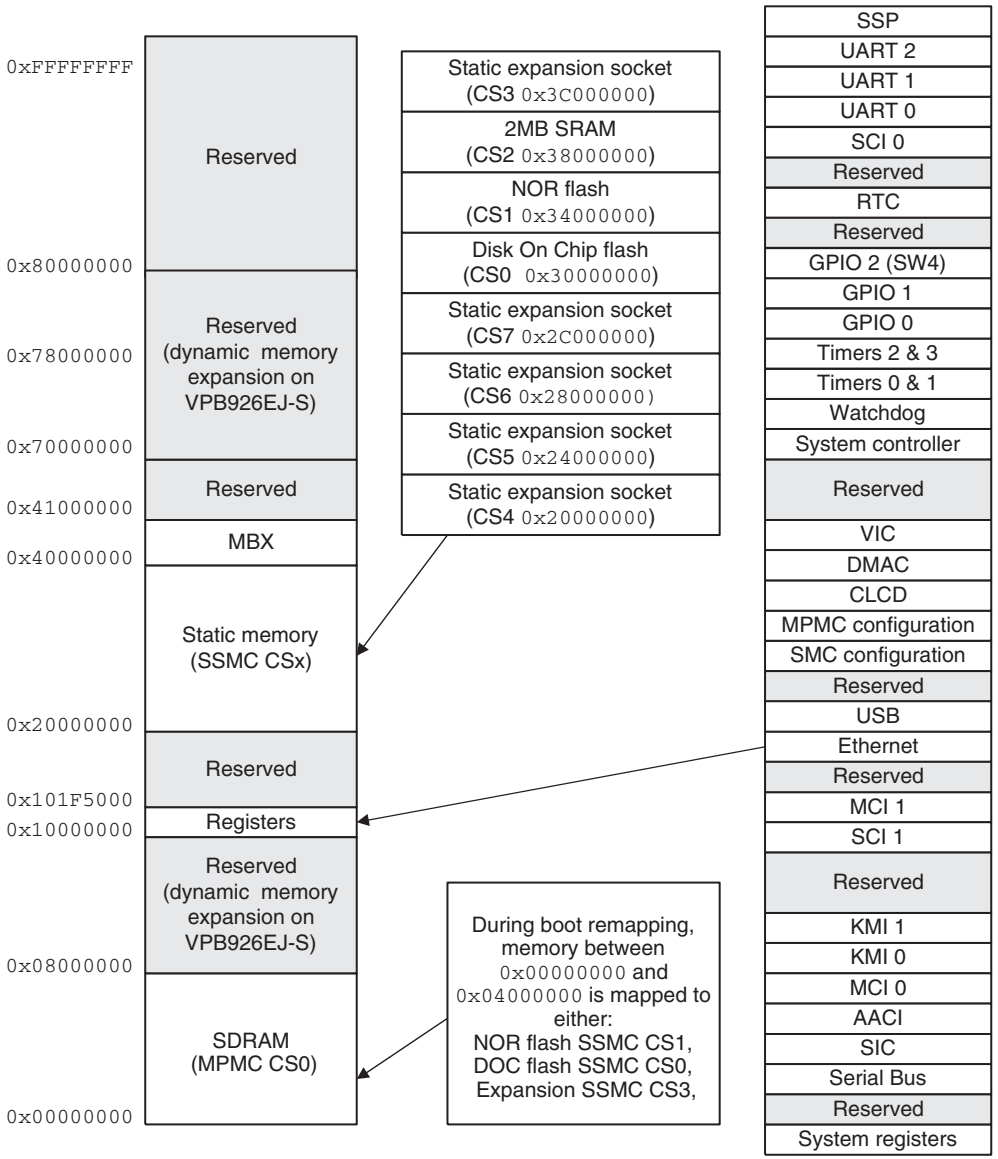

Figure 4-1 AHB Data bus memory map

4.1.1 Differences between the Versatile/PB926EJ-S and Versatile/AB926EJ-S

The Versatile/AB926EJ-S Development Board is a simplified version of the Versatile/PB926EJ-S Development Board. Both products have a similar memory map. There are differences. however, that affect migrating applications between the boards as listed in:

- *Memory map differences*
- *Interrupt differences* on page 4-11
- *DMA differences* on page 4-12.

Memory map differences

Some peripherals present on the Versatile/PB926EJ-S are not present on the Versatile/AB926EJ-S. Table 4-2 and Table 4-3 on page 4-10 list the differences resulting from the different functionality.

Table 4-2 Peripherals not present on Versatile/AB926EJ-S

Peripheral	Memory location	Details
Character LCD	0x10008000-0x10008FFF	There is no character LCD on the AB926EJ-S, however there is dummy character LCD interface to enable code that uses the character LCD should not need to be changed
UART3 interface	0x10009000-0x10009FFF	There are only three UARTs on the AB926EJ-S
SCI1 interface	0x1000A000-0x1000AFFF	There is only one SmartCard interface on AB926EJ-S
MCI1 interface	0x1000B000-0x1000BFFF	There is only one Multimedia card on AB926EJ-S
PCI interface	0x41000000-0x6FFFFFFF	Accesses to this location will cause an abort

Table 4-3 System registers

Register	Memory location	VPB926EJ-S	AB926EJ-S
SYS_ID	0x10000000	0x41007*xx*4 (where *xx* is the build number)	0x41008*xx*4 (where *xx* is the build number)
SYS_OSC1	0x10000010	HCLKM1 frequency	Reserved (Versatile/AB926EJ-S has only two oscillators, one for the Development Chip **XTALCLKEXT** and one for the **CLCDCLK**)
SYS_OSC2	0x10000014	HCLKM2 frequency	Reserved
SYS_OSC3	0x10000018	HCLKS frequency	Reserved
SYS_OSC4 SYS_OSC1	0x1000001c	CLCDCLK	CLCDCLK renamed SYS_OSC1
SYS_CFGDATA1	0x10000028	Development Chip and clock control configuration	Reserved
SYS_PCICTL	0x10000044	PCI control	Reserved
SYS_CLCDSER	0x10000054	CLCD 2.2 control	Reserved (no support for the 2.2inch Epson panel on Versatile/AB926EJ-S)
SYS_DMAPSR0	0x10000064	DMA mapping control 0	Reserved (no DMA mapping control on Versatile/AB926EJ-S)
SYS_DMAPSR1	0x10000068	DMA mapping control 1	Reserved
SYS_DMAPSR2	0x1000006C	DMA mapping control 2	Reserved
SYS_OSCRESET0	0x1000008C	Oscillator 0 reset value	Reserved
SYS_OSCRESET1	0x10000090	Oscillator 1 reset value	Reserved
SYS_OSCRESET2	0x10000094	Oscillator 2 reset value	Reserved
SYS_OSCRESET3	0x10000098	Oscillator 3 reset value	Reserved
SYS_OSCRESET4	0x1000009C	Oscillator 4 reset value	Reserved
SYS_TEST_OSC1	0x100000C4	Test Oscillator Reg 1	Reserved

Table 4-3 System registers (continued)

Register	Memory location	VPB926EJ-S	AB926EJ-S
SYS_TEST_OSC2	0x100000C8	Test Oscillator Reg 2	Reserved
SYS_TEST_OSC3	0x100000CC	Test Oscillator Reg 3	Reserved
SYS_TEST_OSC4	0x100000D0	Test Oscillator Reg 4	Reserved

Interrupt differences

The difference in functionality between the Versatile/PB926EJ-S and the Versatile/AB926EJ-S means that some interrupts are not used or have modified function as listed in Table 4-4 and Table 4-5.

Table 4-4 Primary interrupt mapping

Interrupt bit number	VPB926EJ-S	AB926EJ-S
30	PCI3	Reserved
29	PCI2	Reserved
28	PCI1	Expansion interrupt
27	PCI0	Expansion interrupt
23	MCI1A	Reserved

Table 4-5 Secondary interrupt mapping

Interrupt bit number	VPB926EJ-S	AB926EJ-S
30	PCI3	Reserved
29	PCI2	Reserved
28	PCI1	Expansion interrupt
27	PCI0	Expansion interrupt
23	MCI1A	Reserved

Table 4-5 Secondary interrupt mapping (continued)

Interrupt bit number	VPB926EJ-S	AB926EJ-S
7	Character LCD	Reserved
6	SCI1	Reserved
5	UART3	Reserved
2	MMCI1B	Reserved

DMA differences

The routing of the DMA control signals is fixed on the Versatile/AB926EJ-S. The Versatile/PB926EJ-S allows routing of these signals.

On Versatile/PB926EJ-S, the SYS_DMAPx registers allow AACI TX, AACI RX, USB A, USB B, MCI 0, MCI 1, UART3 TX, UART3 RX, and SCI0 to map to DMA channels 0, 1 or 2.

On Versatile/AB926EJ-S, the mapping is fixed and some DMA transfer request types are restricted as listed in Table 4-6.

Table 4-6 DMA functionality

DMA channel	Peripheral	Notes
0	AACI RX	Single and burst requests allowed
1	AACI TX	Burst requests only
2	MCI0	Burst requests only

4.2 Configuration and initialization

This section describes how the ARM926PXP development chip and external memory and peripherals are configured and initialized at power on. See *Status and system control registers* on page 4-19 for details on configuration operations that can be performed after the system has started. See also *Configuration control* on page 3-7.

4.2.1 Remapping of boot memory

On reset, the ARM926PXP development chip begins executing code at address `0x0`. This address is normally volatile SDRAM. Remapping allows non-volatile static memory to be decoded for accesses to low memory. Remapping of non-volatile memory to the boot region at `0x00000000`–`0x03FFFFFF` is done by the following signals:

BOOTSEL[1:0]

These signals (from configuration switches S4-1 and S4-2) select the non-volatile memory to use if remapping is active (**DEVCHIP REMAP** HIGH).

DEVCHIP REMAP

This signal (from the System Controller register at `0x101E0000`) in the ARM926PXP development chip redirects accesses to memory region `0x00000000`–`0x03FFFFFF` (normally decoded to dynamic memory) to non-volatile memory.

Depending on the state of **BOOTSEL[1:0]**, the non-volatile memory used for boot memory can be either NOR flash or Disk-on-Chip. At reset, the **DEVCHIP REMAP** signal is HIGH.

FPGA_REMAP

This signal (from the SYS_MISC register in the FPGA) redirects chip select 3 (normally `0x34000000`–`0x37FFFFFF`) to one of Disk-on-Chip (`0x30000000`), or NOR flash (`0x34000000`) depending on the state of **BOOTSEL[1:0]**. At reset, the **FPGA_REMAP** signal is HIGH.

Configuration switch S4 modifies the memory map of static memory at reset as listed in Table 4-7 on page 4-14.

A simplified version of the remap logic is shown in Figure 3-10 on page 3-21. S4-1 controls **BOOTCSSEL0** and S4-2 controls **BOOTCSSEL0**. If a switch is ON, the corresponding **BOOTCSSEL** signal is HIGH.

Table 4-7 Selecting the boot device

S4-2	S4-1	Device
OFF	OFF	Disk on Chip, see *Booting from Disk on Chip* on page 4-15
OFF	ON	NOR flash, see *Booting from NOR flash* on page 4-16
ON	OFF	Reserved, do not use this setting.
ON	ON	Reserved, do not use this setting.

A simplified version of the remap logic is shown in *Memory aliasing at reset* on page 3-20.

Removing boot remapping and enabling SDRAM at 0x0

The ARM926PXP development chip begins executing at `0x0` after a reset. The remapping logic is decoding this address to a chip select signal for static memory and executing boot code.

The boot code must perform the following actions on reset to remove the remapping and enable SDRAM at `0x0`:At reset,

1. The remap signals are both high, therefore static memory is remapped to address `0x0`. Perform any critical CPU initialization at this time. Ensure that you do not access SDRAM at this point as it has not been initialized.
2. For Disk-on-Chip (**nDOCCS**) or NOR flash (**nNORCS**), jump to a location in the range `0x34000000-0x37FFFFFF`. Jumping out of the range `0x000000000-0x03FFFFFF` means that the remapped memory at `0x0` is no longer needed and can be unmapped.

   The code jumps to `0x34000000-0x37FFFFFF` rather than the physical location of the boot memory because the boot code does not know which physical memory device it is located in and because the control registers for the other device selects are not installed.
3. Clear the **DEVCHIP REMAP** bit by writing a 1 to bit 8 of the System Controller register at `0x101E0000`.
4. Initialize the MPMC controller with the appropriate values for the type of dynamic RAM used.

5. Use the SDRAM at location `0x0` to hold additional boot code and the stack for the application.
6. Jump to the additional boot code that you have loaded into SDRAM.
7. Set up all static chip select control registers. If you are not booting from NOR flash, you must also set up the control register for **nSTATICCS1**.
8. Clear the **FPGA_REMAP** signal by writing a 0 to bit 2 of SYS_MISC register.

Note

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

Booting from Disk on Chip

The memory maps for S4-2 OFF (**BOOTSEL1** LOW) and S4-1 OFF (**BOOTSEL0** LOW) are shown in Figure 4-2 on page 4-16.

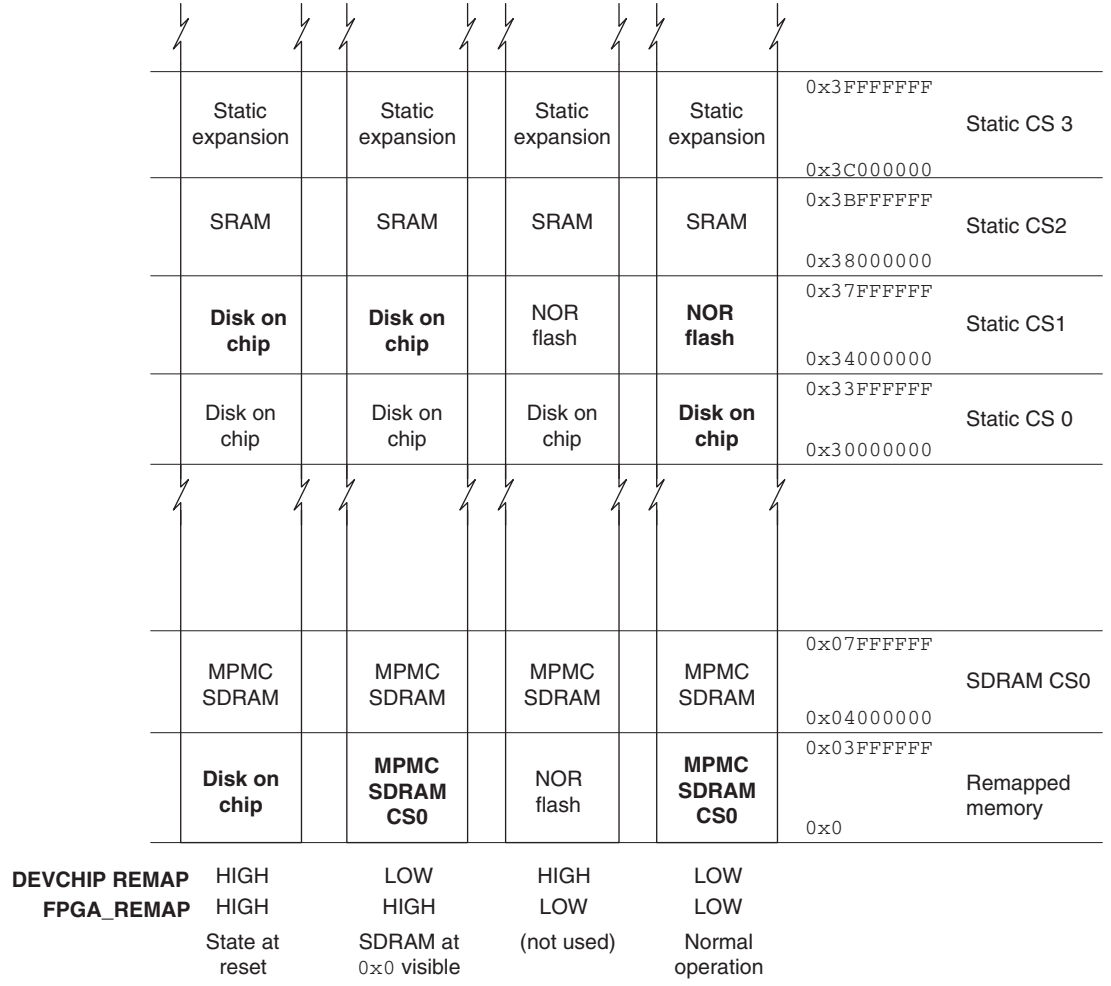

Figure 4-2 Booting from Disk on Chip

Booting from NOR flash

The memory maps for S4-2 OFF (**BOOTSEL1** LOW) and S4-1 ON (**BOOTSEL0** HIGH) are shown in Figure 4-3 on page 4-17.

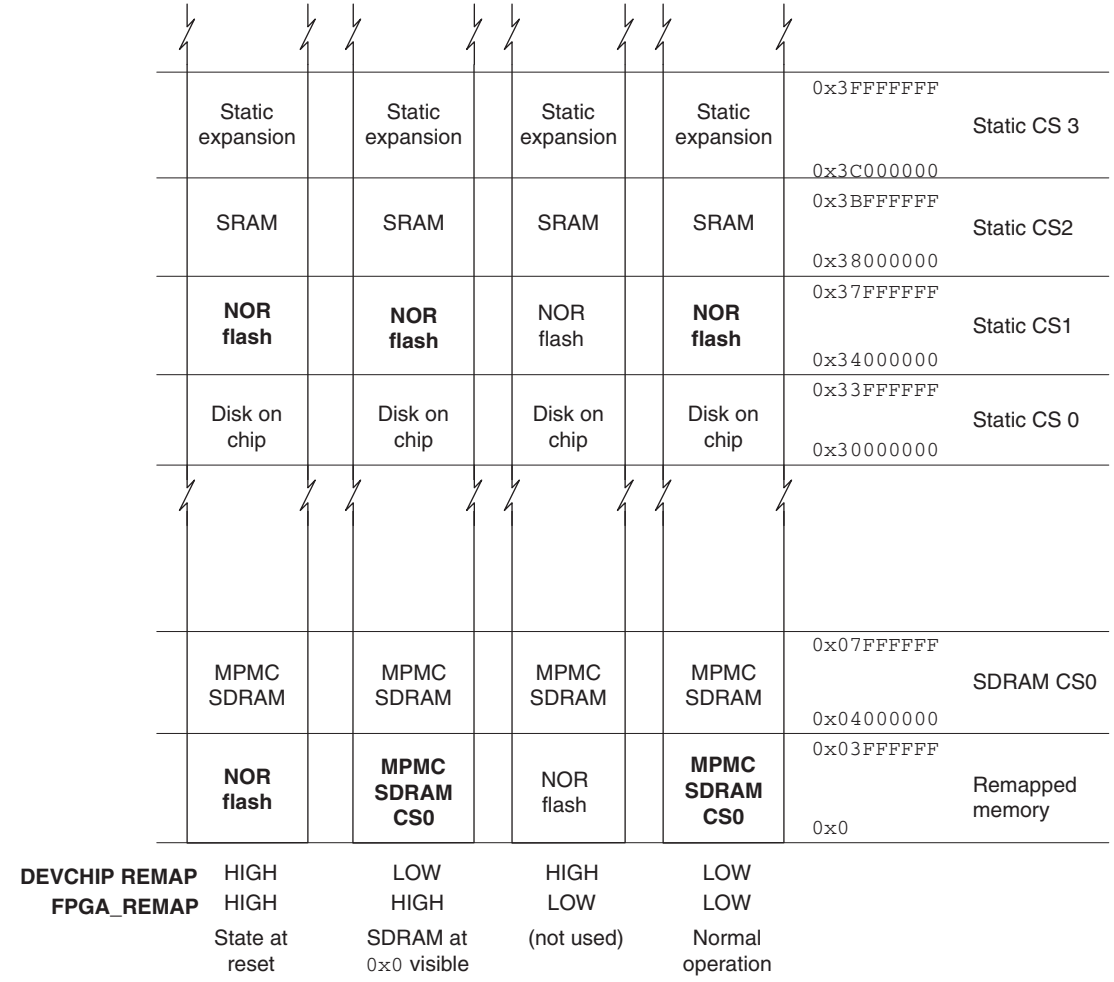

Figure 4-3 Booting from NOR flash

Memory characteristics

Table 4-8 on page 4-18 lists the controller memory banks, chip selects, and memory range.

Table 4-8 Memory chip selects and address range

Bank	Chip select	Address range	Device
MPMC bank 4	**nMPMCDYCS0**	0x00000000-0x07FFFFFF	SDRAM
SSMC bank 0	**nSTATICCS0**	0x30000000-0x33FFFFFF	Disk-on-Chip
SSMC bank 7	**nSTATICCS1**	0x34000000-0x37FFFFFF	NOR flash
SSMC bank 3	**nSTATICCS2**	0x38000000-0x3BFFFFFF	SRAM
SSMC bank 1	**nSTATICCS3**	0x3C000000-0x0000FFFF	Expansion static memory
SSMC bank 4	**nSTATICCS4**	0x20000000-0x23FFFFFF	Expansion static memory
SSMC bank 5	**nSTATICCS5**	0x24000000-0x27FFFFFF	Expansion static memory
SSMC bank 6	**nSTATICCS6**	0x28000000-0x2BFFFFFF	Expansion static memory
SSMC bank 1	**nSTATICCS7**	0x2C000000-0x0000FFFF	Expansion static memory

4.3 Status and system control registers

The Versatile/AB926EJ-S status and system control registers enable the processor to determine its environment and to control some on-board operations. The registers, listed in Table 4-9, are located from `0x10000000`.

See also the *ARM PrimeXSys System Controller (SP810) Technical Reference Manual* for details of control registers in the SP810 System Controller that is in the ARM926PXP development chip.

Note

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

In Table 4-9, the entry for **Reset level** indicates the highest reset level that modifies the register contents:

Level 6 This a programmable reset level that is triggered by a software reset, **nSRST**, or **nPBRESET**. See *Reset controller* on page 3-18 for a description of programmable reset levels and the reset signals.

Level 2 The system is reconfigured. The registers are loaded from a writable configuration register. (This reset level is identical to Level 1 for the Versatile/AB926EJ-S. On the Versatile/PB926EJ-S, the OSC0 clock values are loaded from SYS_OSCRESET0.)

Level 1 The FPGA is reconfigured.

Level 0 The system power on reset (nSYSPOR) is a level 0 reset and initializes all registers to their default value.

Table 4-9 Register map for system control registers

Name	Address	Access	Reset level	Description
SYS_ID	`0x10000000`	Read only	-	System Identity. See *ID Register, SYS_ID* on page 4-22.
SYS_SW	`0x10000004`	Read only	-	Bits [7:0] map to S1 (user switches)
SYS_LED	`0x10000008`	Read/Write	6	Bits [7:0] map to user LEDs (located next to S1)

Table 4-9 Register map for system control registers (continued)

Name	Address	Access	Reset level	Description
SYS_OSC0	0x1000000C	Read/Write Lockable	2	Settings for the ICS307 programmable oscillator chip OSC0. For the default resistor links, the programmable output from the oscillator provides **XTALCLKEXT**. See *Oscillator registers, SYS_OSCx* on page 4-24 and *ARM926PXP development chip clocks* on page 3-31.
Reserved	0x10000010	-	-	(SYS_OSC1 on the Versatile/PB926EJ-S)
Reserved	0x10000014	-	-	(SYS_OSC2 on the Versatile/PB926EJ-S)
Reserved	0x10000018	-	-	(SYS_OSC3 on the Versatile/PB926EJ-S)
SYS_OSC1	0x1000001C	Read/Write Lockable	2	Settings for the ICS307 programmable oscillator chip OSC1. This oscillator provides the **CLCDCLK** signal source. (This register was SYS_OSC4 on the Versatile/PB926EJ-S.)
SYS_LOCK	0x10000020	Read/Write	6	Write 0xA05F to unlock. See *Lock Register, SYS_LOCK* on page 4-24.
SYS_100HZ	0x10000024	Read only	0	100Hz counter.
Reserved	0x10000028	-	-	(SYS_CFGDATA1 on Versatile/PB926EJ-S)
Reserved	0x1000002C	-	-	(SYS_CFGDATA2 on Versatile/PB926EJ-S)
SYS_FLAGS	0x10000030	Read only	6	General-purpose flags (reset by any reset). See *Flag registers, SYS_FLAGx and SYS_NVFLAGx* on page 4-26.
SYS_FLAGSSET	0x10000030	Write	6	Set bits in general-purpose flags.
SYS_FLAGSCLR	0x10000034	Write	6	Clear bits in general-purpose flags.
SYS_NVFLAGS	0x10000038	Read only	0	General-purpose nonvolatile flags (reset only on power up).
SYS_NVFLAGSSET	0x10000038	Write	0	Set bits in general-purpose nonvolatile flags.
SYS_NVFLAGSCLR	0x1000003C	Write	0	Clear bits in general-purpose nonvolatile flags.
SYS_RESETCTL	0x10000040	Read/Write Lockable	0	The reset control register sets reset depth and programmable softreset. It can only be written if SYS_LOCK is unlocked.

Table 4-9 Register map for system control registers (continued)

Name	Address	Access	Reset level	Description
Reserved	0x10000044	-	-	(SYS_PCICTL on Versatile/PB926EJ-S)
SYS_MCI	0x10000048	Read only	-	Contains the "card present" and "write enabled" status for the MMCI0 card
SYS_FLASH	0x1000004C	Read/Write	6	Controls write protection of flash devices.
SYS_CLCD	0x10000050	Read/Write	6	Controls LCD power and multiplexing.
Reserved	0x10000054	-	-	(2.2 inch display on theVersatile/PB926EJ-S LCD adaptor).
Reserved	0x10000058	-	-	(Contains the settings of the boot switch on theVersatile/PB926EJ-S)
SYS_24MHz	0x1000005C	Read only	6	32-bit counter clocked at 24MHz.
SYS_MISC	0x10000060	Read only	6	See *Miscellaneous System Control Register, SYS_MISC* on page 4-30 for details.
Reserved	0x10000064	-	-	(SYS_DMAPSR0 on Versatile/PB926EJ-S)
Reserved	0x10000068	-	-	(SYS_DMAPSR1 on Versatile/PB926EJ-S)
Reserved	0x1000006C	-	-	(SYS_DMAPSR2 on Versatile/PB926EJ-S)
Reserved	0x1000008C	-	0	(SYS_OSCRESET0 on Versatile/PB926EJ-S)
Reserved	0x10000090	-	-	(SYS_OSCRESET1 on Versatile/PB926EJ-S)
Reserved	0x10000094	-	-	(SYS_OSCRESET2 on Versatile/PB926EJ-S)
Reserved	0x10000098	-	-	(SYS_OSCRESET3 on Versatile/PB926EJ-S)
Reserved	0x1000009C	-	-	(SYS_OSCRESET4 on Versatile/PB926EJ-S)
SYS_TEST_OSC0	0x100000C0	Read only	6	32-bit counter clocked from ISC307 clock 0.
Reserved	0x100000C4	-	-	(SYS_TEST_OSC1 on Versatile/PB926EJ-S)
Reserved	0x100000C8	-	-	(SYS_TEST_OSC2 on Versatile/PB926EJ-S)
Reserved	0x100000CC	-	-	(SYS_TEST_OSC3 on Versatile/PB926EJ-S)
Reserved	0x100000D0	-	-	(SYS_TEST_OSC4 on Versatile/PB926EJ-S)

4.3.1 ID Register, SYS_ID

The ID Register, SYS_ID, at `0x10000000` is a read-only register that identifies the board manufacturer, board type, and revision. Figure 4-4 shows the bit assignment of the register.

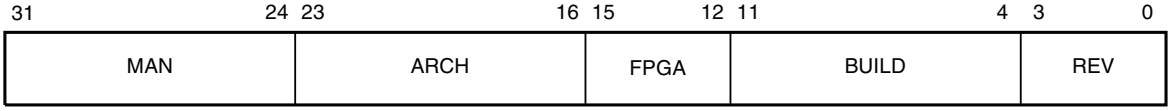

Figure 4-4 ID Register, SYS_ID

Table 4-10 describes the Versatile/AB926EJ-S ID Register assignment.

Table 4-10 ID Register, SYS_ID bit assignment

Bits	Name	Access	Function
[31:24]	MAN	Read	Manufacturer: `0x41` = ARM
[23:16]	ARCH	Read	Architecture: `0x00` = ARM926
[15:12]	FPGA	Read	FPGA type: `0x8` = XC2S300E
[11:4]	BUILD	Read	Build value (ARM internal use)
[3:0]	REV	Read	Major revision `0x4` = multilayer AHB

4.3.2 Switch Register, SW

Use the SYS_SW register at `0x10000004` to read the general purpose (user) switch S1. A value of 1 indicates that the switch is on.

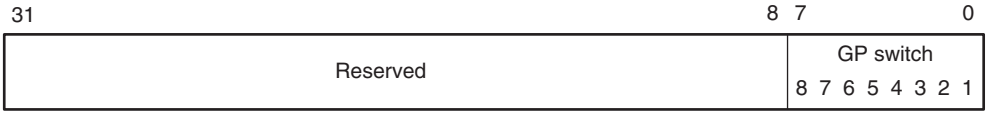

Figure 4-5 SYS_SW

4.3.3 LED Register, SYS_LED

Use the SYS_LED register at `0x10000008` to set the user LEDs. (The LEDs are located next to user switch S1.) Set the corresponding bit to 1 to illuminate the LED.

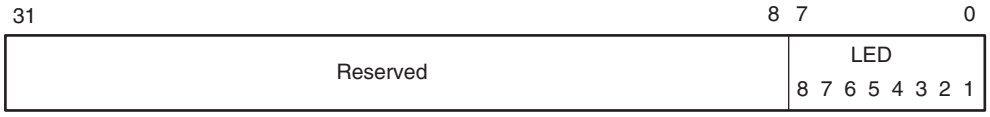

Figure 4-6 SYS_LED

4.3.4 Oscillator registers, SYS_OSCx

The Versatile/AB926EJ-S Oscillator Registers, SYS_OSC0 and SYS_OSC1, at `0x1000000C` and `0x1000001C` are read/write registers that control the frequency of the clocks generated by the ICS307 programmable clock generators. A serial interface transfers the values in the registers to the programmable oscillators when a reset occurs.

Figure 4-7 shows the bit assignment of the registers.

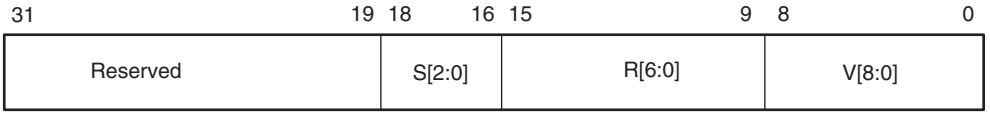

Figure 4-7 Oscillator Register, SYS_OSCx

Table 4-11 lists the details of the SYS_OSCx registers. For more detail on bit values, see *ICS307 programmable clock generators* on page 3-35 and *Clock rate restrictions* on page B-4.

Table 4-11 Oscillator Register, SYS_OSCx bit assignment

Bits	Name	Access	Function
[31:19]	Reserved	Use read-modify-write to preserve value.	
[18:16]	S	Read/write	Output divider select (OD)
[15:9]	R	Read/write	Reference divider word (RDW)
[8:0]	V	Read/write	VCO divider word (VDW)

Note

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

4.3.5 Lock Register, SYS_LOCK

The Lock Register, SYS_LOCK at `0x10000020`, locks or unlocks access to the following registers:

- Oscillator registers, SYS_OSCx
- Reset control register, SYS_RESETCTL

The control registers cannot be modified while they are locked. This mechanism prevents the registers from being overwritten accidently. The registers are locked by default after a reset. Figure 4-8 shows the bit assignment of the register.

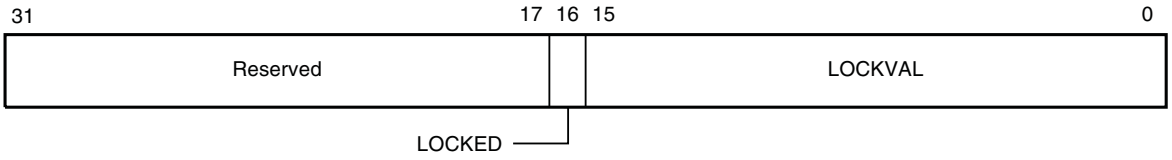

Figure 4-8 Lock Register, SYS_LOCK

Table 4-12 describes the Versatile/AB926EJ-S Lock Register bit assignment.

Table 4-12 Lock Register, SYS_LOCK bit assignment

Bits	Name	Access	Function
[31:17]	Reserved	Use read-modify-write to preserve value.	
[16]	LOCKED	Read	This bit indicates if the control registers are locked or unlocked: 0 = unlocked 1 = locked.
[15:0]	LOCKVAL	Read/write	Write the value `0xA05F` to unlock the control registers. Write any other value to this register to lock the registers.

4.3.6 100Hz Counter, SYS_100HZ

The 100Hz Counter Register, SYS_100HZ, at `0x10000024` is a 32-bit counter incremented at 100Hz. The 100Hz reference is derived from the 32KHz crystal oscillator. The register is set to zero by a reset.

4.3.7 Flag registers, SYS_FLAGx and SYS_NVFLAGx

The Versatile/AB926EJ-S Flag Registers shown in Table 4-13 provide two 32-bit registers containing general-purpose volatile and non-volatile flags. You can assign any meaning to the flags.

Table 4-13 Flag registers

Register name	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 Versatile/AB926EJ-S provides two distinct types of flag registers:

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

Flag and Nonvolatile Flag Registers

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

Flag and Nonvolatile Flag Set Registers

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

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

Flag and Nonvolatile Flag Clear Registers

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

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

4.3.8 Reset Control Register, SYS_RESETCTL

This register at `0x10000040` sets reset depth and programmable soft reset, see *Reset controller* on page 3-18 and *Reset level* on page 3-22. The function of the register bits are shown in Table 4-14. You must unlock the register (see *Lock Register, SYS_LOCK* on page 4-24) before the register contents can be modified.

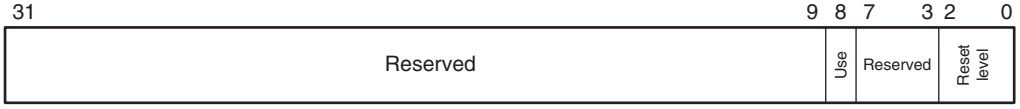

Figure 4-9 SYS_RESETCTL

Table 4-14 Reset level control

Bits	Access	LCD control signal description
[31:9]	Read write	Reserved. Use read-modify-write to preserve value.
[8]	Write	Set this bit to generate a reset at the level specified by bits [2:0]
[7:3]	Read write	Reserved. Use read-modify-write to preserve value.
[2:0]	Read write	Select reset level: `b001`–`b000` resets to level 1, CONFIGCLR`b010` resets to level 2, CONFIGINIT`b011` resets to level 3, DLLRESET`b100` resets to level 4, PLLRESET`b101` resets to level 5, PORRESET`b110`–`b111` resets to level 6, DOCRESET

4.3.9 MCI Register, SYS_MCI

The SYS_MCI register at `0x10000048` provides status information on the Multimedia card sockets. The function of the register bits are shown in Table 4-15 on page 4-28.

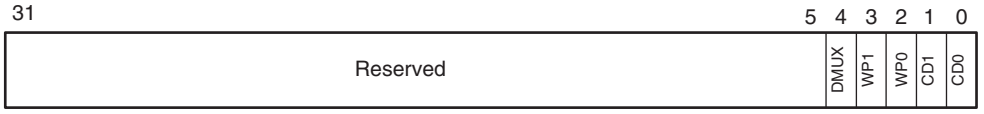

Figure 4-10 SYS_MCI

Table 4-15 MCI control

Bits	Access	Description
[31:5]	-	Reserved. Use read-modify-write to preserve value. (second MCI card on Versatile/PB926EJ-S)
[4]	-	Reserved (data multiplex)
[3]	-	Reserved (Write protect 1)
[2]	Read	Write protect 0
[1]	-	Reserved (Card detect 1)
[0]	Read	Card detect 0

4.3.10 Flash Control Register, SYS_FLASH

Bit 0 of the SYS_FLASH register at `0x1000004C` controls write protection of flash devices. The function of the register bits are shown in Table 4-16.

Table 4-16 Flash control

Bits	Access	Description
[31:1]	-	Reserved. Use read-modify-write to preserve value.
[0]	Read/write	Disables writing to flash if LOW (POR state is LOW)

4.3.11 CLCD Control Register, SYS_CLCD

This register at `0x10000050` controls LCD power and multiplexing and controls the interface to the touchscreen as listed in Table 4-17 on page 4-29. See Appendix E *LCD Kits* for details on the CLCD control signals.

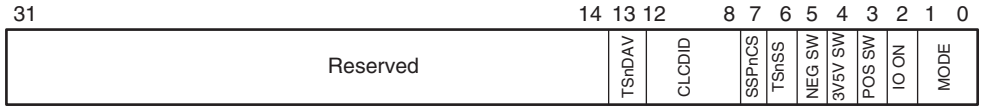

Figure 4-11 SYS_CLCD

Table 4-17 SYS_CLCD register

Bits	Access	Description
[31:14]	-	Reserved. Use read-modify-write to preserve value.
[13]	Read	TSnDAV indicates that data is available from the touchscreen controller.

[12:8]	Read	CLCDID[4:0] (returns the setting of the ID links on an interface board)
		`Value Display`
		`0 320x240 QVGA`
		`1 640x480 VGA`
		`2 220x176`
		`3-31 Reserved`

[7]	Read/write	SSP expansion (if SSPnCS is LOW)
[6]	Read/write	Touchscreen enable (TSnSS)
[5]	Read/write	VDDNEGSWITCH (power control for LCD interface)
[4]	Read/write	PWR3V5VSWITCH (power control for LCD interface)
[3]	Read/write	VDDPOSSWITCH (power control for LCD interface)
[2]	Read/write	nLCDIOON (power control for LCD interface)

[1:0]	Read/write	LCD Mode [1:0]
		`Bit 1 Bit 0 Display mode`
		`0 0 8:8:8`
		`0 1 5:5:5:1`
		`1 0 5:6:5, red LSB`
		`1 1 5:6:5, blue LSB`

4.3.12 24MHz Counter, SYS_24MHZ

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

4.3.13 Miscellaneous System Control Register, SYS_MISC

This register at `0x10000060` provides miscellaneous status and control signals as shown in Table 4-18.

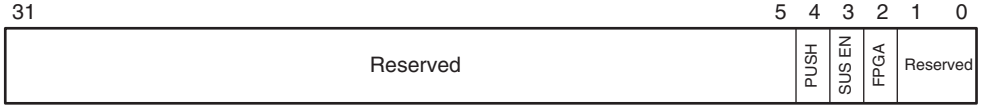

Figure 4-12 SYS_MISC

Table 4-18 SYS_MISC

Bits	Access	Description
[31:8]	-	Reserved. Use read-modify-write to preserve value.
[7:5]	-	Reserved. Use read-modify-write to preserve value.
[4]	Read	GP PUSHSWITCH state. If pressed, the value is 1.
[3]	Read/write	Suspend Enable (The **PWRFAIL** pin is not connected to any power-fail logic, but the pin can be used to test application code that must respond to a power failure. When set, this signal allows the GP PUSHSWITCH to toggle the **PWRFAIL** pin on ARM926PXP development chip)
[2]	Read/write	FPGA remap control (**FPGA_REMAP**)
[1:0]	-	Reserved. Use read-modify-write to preserve value (expansion detect on Versatile/PB926EJ-S).

4.3.14 Oscillator test registers, SYS_TEST_OSC0

The oscillator test register, SYS_TEST_OSC0, provides 32-bit count values. The count increments at frequency of ICS307 programmable oscillator OSC0. The registers are set to zero by a reset.

Table 4-19 Oscillator test registers

Name	Address	Access	Description
SYS_TEST_OSC0	0x100000C0	Read	Counter clocked from OSC0

4.4 Advanced Audio CODEC Interface, AACI

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

Table 4-20 AACI implementation

Property	Value
Location	FPGA (the CODEC is an external LM4549)
Memory base address	`0x10004000`
Interrupt	24 on secondary controller
DMA	Tx is DMA channel 1, Rx is DMA channel 0
Release version	ARM AACI PL041 r1p0 (modified to one channel and 256 FIFO depth in compact mode and 512 in non-compact mode)
Reference documentation	*ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual* and *National Semiconductor LM4549 Data Sheet*. (See also changed PrimeCell ID listed in Table 4-21 on page 4-33 and *Advanced Audio Codec Interface, AACI* on page 3-38. on page 3-38)

4.4.1 PrimeCell modifications

The AACI for the Versatile/AB926EJ-S has a different FIFO depth than the standard PL041. Therefore, the AACIPeriphID3 register contains the values listed in Table 4-21 on page 4-33.

Figure 4-13 AACI ID register

Table 4-21 Modified AACI PeriphID3 register

Bit		Description
[31:8]	-	not used
[7:6]	-	Reserved. Use read-modify-write to preserve value.
[5:3]	Read	FIFO depth in compact mode b000 8 b001 16 b010 32 b011 64 b100 128 b101 256 b110 512 b111 1024
[2:0]	Read	number of channels b000 4 b001 1 b010 2 b011 3 b100 4 b101 5 b110 6 b111 7

4.5 Color LCD Controller, CLCDC

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

Table 4-22 CLCDC implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x10120000` **Note** There are also a LCD power control register at `0x10000050`. See *CLCD Control Register, SYS_CLCD* on page 4-28.
Interrupt	16 on PIC
DMA	NA
Release version	ARM CLCDC PL110 r0p0-00alp0
Reference documentation	*ARM PrimeCell Color LCD Controller (PL110) Technical Reference Manual* (see also address modifications listed in *PrimeCell modifications* on page 4-35, *Display resolutions and display memory organization* on page 4-35, and *CLCDC interface* on page 3-40)

The following locations are reserved for future use and must not be used:

- locations at offsets `0x030`–`0x1FE`
- locations at offsets `0x400`–`0x7FF`.

4.5.1 PrimeCell modifications

The register map for the variant of the PL110 used in the ARM926PXP development chip is not the same as that listed for the standard PL110. The differences are listed in Table 4-23.

Table 4-23 PrimeCell CLCDC register differences

Address (Dev. Chip)	Reset value (Dev. Chip)	Description	Difference
0x10120018	0x0	LCDControl, LCD panel pixel parameters	listed as address0x1012001C in CLCDC TRM
0x1012001C	0x0	LCDIMSC, interrupt mask set and clear	listed as address 0x10120018 in CLCDC TRM
0x10120800–0x10120C2C	0x0	Not present	Hardware cursor registers from PL111 (see the *ARM926EJ-S Technical Reference Manual* for details)
0x10120FE0	0x93	CLCDPeriphID0	listed as 0x10 in CLCDC TRM
0x10120FE4	0x10	CLCDPeriphID1	listed as 0x11 in CLCDC TRM

4.5.2 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_OSC1 to define the display timings. Table 4-24 on page 4-36 lists the register and clock values for different display resolutions.

Note

Register values in Table 4-24 on page 4-36 are for external monitors connected to the VGA connector. Some resolutions, QVGA for example, do not fill the entire screen on the VGA monitor.

For clock settings for LCD display kits, see Appendix E *LCD Kits*.

Table 4-24 Values for different display resolutions

Display resolution for external monitors	CLCDCLK frequency and SYS_OSC1 register value	CLCD_TIM0 register at 0x10120000	CLCD_TIM1 register 0x10120004	CLCD_TIM2 register at 0x10120008
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

The pixel data in memory is mapped to the display in different ways depending on display mode. Table 4-25 lists software usage of memory bits and Table 4-26 on page 4-38 lists the correspondence between the hardware pins and the bits in memory.

Note

For resolutions based on 16 bits per pixel, two pixels (pixel0 and pixel1) are encoded in one 32-bit word.

Rx, Gx, and Bx in Table 4-24 and Table 4-26 on page 4-38 refer to bits used to set the red, green, and blue brightness.

Table 4-25 Assignment of display memory to R[7:0], G[7:0], and B[7:0]

Memory bit	8/8/8	1/5/5/5	5/6/5 red (lsb)	5/6/5 blue (lsb)
31	unused	pixel1 I (intensity)	pixel1 B5 (msb)	pixel1 R5 (msb)
30	unused	pixel1 B5 (msb)	pixel1 B4	pixel1 R4
29	unused	pixel1 B4	pixel1 B3	pixel1 R3

Table 4-25 Assignment of display memory to R[7:0], G[7:0], and B[7:0] (continued)

Memory bit	8/8/8	1/5/5/5	5/6/5 red (lsb)	5/6/5 blue (lsb)
28	unused	pixel1 B3	pixel1 B2	pixel1 R2
27	unused	pixel1 B2	pixel1 B1 (lsb)	pixel1 R1 (lsb)
26	unused	pixel1 B1 (lsb)	pixel1 G5 (msb)	pixel1 G5 (msb)
25	unused	pixel1 G5 (msb)	pixel1 G4	pixel1 G4
24	unused	pixel1 G4	pixel1 G3	pixel1 G3
23	B7 (msb)	pixel1 G3	pixel1 G2	pixel1 G2
22	B6	pixel1 G2	pixel1 G1	pixel1 G1
21	B5	pixel1 G1 (lsb)	pixel1 G0 (lsb)	pixel1 G0 (lsb)
20	B4	pixel1 R5 (msb)	pixel1 R5 (msb)	pixel1 B5 (msb)
19	B3	pixel1 R4	pixel1 R4	pixel1 B4
18	B2	pixel1 R3	pixel1 R3	pixel1 B3
17	B1	pixel1 R2	pixel1 R2	pixel1 B2
16	B0 (lsb)	pixel1 R1 (lsb)	pixel1 R1 (lsb)	pixel1 B1 (lsb)
15	G7 (msb)	pixel0 I (intensity)	pixel0 B5 (msb)	pixel0 R5 (msb)
14	G6	pixel0 B5 (msb)	pixel0 B4	pixel0 R4
13	G5	pixel0 B4	pixel0 B3	pixel0 R3
12	G4	pixel0 B3	pixel0 B2	pixel0 R2
11	G3	pixel0 B2	pixel0 B1 (lsb)	pixel0 R1 (lsb)
10	G2	pixel0 B1 (lsb)	pixel0 G5 (msb)	pixel0 G5 (msb)
9	G1	pixel0 G5 (msb)	pixel0 G4	pixel0 G4
8	G0 (lsb)	pixel0 G4	pixel0 G3	pixel0 G3
7	R7 (msb)	pixel0 G3	pixel0 G2	pixel0 G2
6	R6	pixel0 G2	pixel0 G1	pixel0 G1
5	R5	pixel0 G1 (lsb)	pixel0 G0 (lsb)	pixel0 G0 (lsb)

Table 4-25 Assignment of display memory to R[7:0], G[7:0], and B[7:0] (continued)

Memory bit	8/8/8	1/5/5/5	5/6/5 red (lsb)	5/6/5 blue (lsb)
4	R4	pixel0 R5 (msb)	pixel0 R5 (msb)	pixel0 B5 (msb)
3	R3	pixel0 R4	pixel0 R4	pixel0 B4
2	R2	pixel0 R3	pixel0 R3	pixel0 B3
1	R1	pixel0 R2	pixel0 R2	pixel0 B2
0	R0 (lsb)	pixel0 R1 (lsb)	pixel0 R1 (lsb)	pixel0 B1 (lsb)

Table 4-26 PL110 hardware playback mode

ARM926PXP development chip pin	TFT24bit 8/8/8 memory bit, color	TFT16bit 1/5/5/5 memory bit, color	TFT16bit 5/6/5 red LSB memory bit, color	TFT16bit 5/6/5 blue LSB memory bit, color
CLD23	23, B7	-	-	-
CLD22	22, B6	-	-	-
CLD21	21, B5	-	-	-
CLD20	20, B4	-	-	-
CLD19	19, B3	-	-	-
CLD18	18, B2	-	-	-
CLD17	17, B1	14/30, B5	14/30, B3	14/30, R3
CLD16	16, B0	13/29, B4	13/29, B2	13/29, R2
CLD15	15, G7	12/28, B3	12/28, B1	12/28, R1
CLD14	14, G6	11/27, B2	11/27, B0	11/27, R0
CLD13	13, G5	10/26, B1	10/26, G5	10/26, G5
CLD12	12, G4	15/31, I (B0)	15/31, B4	15/31, R4
CLD11	11, G3	9/25, G5	9/25, G4	9/25, G4
CLD10	10, G2	8/24, G4	8/24, G3	8/24, G3
CLD9	9, G1	7/23, G3	7/23, G2	7/23, G2

Table 4-26 PL110 hardware playback mode (continued)

ARM926PXP development chip pin	TFT24bit 8/8/8 memory bit, color	TFT16bit 1/5/5/5 memory bit, color	TFT16bit 5/6/5 red LSB memory bit, color	TFT16bit 5/6/5 blue LSB memory bit, color
CLD8	8, G0	6/22, G2	6/22, G1	6/22, G1
CLD7	7, R7	5/21, G1	5/21, G0	5/21, G0
CLD6	6, R6	15/31, I (G0)	15/31, B4	15/31, R4
CLD5	5, R5	4/20, R5	4/20, R4	4/20, B4
CLD4	4, R4	3/19, R4	3/19, R3	3/19, B3
CLD3	3, R3	2/18, R3	2/18, R2	2/18, B2
CLD2	2, R2	1/17. R2	1/17, R1	1/17, B1
CLD1	1, R1	0/16, R1	0/16, R0	0/16, B0
CLD0	0, R0	15/31, I (R0)	15/31, B4	15/31, R4

4.6 Direct Memory Access Controller

The PrimeCell *Direct Memory Access Controller* (DMAC) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited. The DMAC is located in the ARM926PXP development chip.

Table 4-27 DMAC implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x10130000` for DMAC (PL010)
Interrupt	17 on the primary controller
DMA	NA
Release version	ARM DMAC PL080 r1p0
Reference documentation	*ARM PrimeCell DMA (PL080) Technical Reference Manual* (see also *DMA* on page 3-43)

Sixteen peripheral DMA interfaces are provided by the PrimeCell DMAC, of which ten are used by the ARM926PXP development chip peripherals (UART0–3, SCI, and SSP) and three are made available for devices in the FPGA.

Note

The DMA controller cannot access the Tightly Coupled Memory in the ARM926EJ-S core. Other access limitations are:

- DMAC master 0 can always access the DMA APB and FPGA peripherals
- DMAC master 1 can always access dynamic and static memory and FPGA peripherals.
- Accesses to other regions are usually mapped to AHB M2.

See *AHB bridges and the bus matrix* on page 3-8.

Table 4-28 shows the DMA channel allocation.

Table 4-28 DMA channels

DMA channel	DMA Requester
15	UART0 Tx
14	UART0 Rx
13	UART1 Tx
12	UART1 Rx
11	UART2 Tx
10	UART2 Rx
9	SSP Tx
8	SSP Rx
7	SCI Tx
6	SCI Rx
5	Reserved
4	Reserved
3	Reserved
2	MCI (device in the FPGA)
1	AACI TX (device in the FPGA)
0	AACI RX (device in the FPGA)

4.7 Ethernet

The Ethernet interface is implemented in an external SMC LAN91C111 10/100 Ethernet single-chip MAC and PHY. The internal registers of the LAN91C111 are memory-mapped onto the AHBM2 bus and occupy 16 word locations at `0x10010000`.

Table 4-29 Ethernet implementation

Property	Value
Location	Board (LAN91C111 chip)
Memory base address	`0x10010000`
Interrupt	25 on both the primary and secondary controllers
DMA	No peripheral DMA support. Use memory to memory DMA to access the buffer memory. The master interface located in the LAN91C11 is not supported.
Release version	The FPGA contains a custom interface to the LAN91C111 chip
Reference documentation	*LAN91C111 Data Sheet* (see also *Ethernet interface* on page 3-45).

To access the PHY MII registers, you must implement a synchronous serial connection in software to control the management register in Bank 3. By default, the PHY is set to isolate in the control register. This disables the external interface. Refer to the LAN91C111 application note for additional information.

When manufactured, an ARM value for the Ethernet MAC address and the register base address are loaded into the EEPROM. The register base address is 0. The MAC address is unique, but can be reprogrammed if required. Reprogramming of the EEPROM is done through Bank 1 (general and control registers).

4.8 General Purpose Input/Output, GPIO

The PrimeCell *General Purpose Input/Output* (GPIO) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-30 GPIO implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x101E4000` for GPIO 0`0x101E5000` for GPIO 1`0x101E6000` for GPIO 2 (board status)`0x101E7000` for GPIO 3 (not used)
Interrupt	6 on primary controller for GPIO 0 7 on primary controller for GPIO 1 8 on primary controller for GPIO 2 9 on primary controller for GPIO 3
DMA	NA
Release version	ARM GPIO PL061 r1p0
Reference documentation	*ARM PrimeCell GPIO (PL061) Technical Reference Manual* (see also *GPIO interface* on page 3-48)

Note

Only GPIO 0 and GPIO 1 are available for general-purpose use.

GPIO 2 signals **GP2_[3:0]** are connected to the boot select switch S4 and **GP2_[4]** is LOW if an interface board is connected.

4.9 Interrupt controllers

The PrimeCell *Vectored Interrupt Controller* (VIC) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

The ARM926EJ-S has two interrupt signals:

- **FIQ** for fast, low latency interrupt handling
- **IRQ** for more general interrupts.

The VIC in the ARM926PXP development chip accepts interrupts from peripherals in the ARM926PXP development chip, FPGA, or an interface board and generates the FIQ and IRQ signals. The VIC provides a software interface to the interrupt system and functions as the *primary interrupt controller* (PIC).

Table 4-31 VIC Primary Interrupt Controller implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x10140000` for PL190 VIC (primary interrupt controller)
Interrupt	FIQ and IRQ
DMA	NA
Release version	ARM VIC PL190 r1p1
Reference documentation	*ARM PrimeCell Vector Interrupt Controller (PL190) Technical Reference Manual*, *Primary interrupt controller* on page 4-45, and *Interrupts* on page 3-49)

A secondary interrupt controller is implemented as a custom design in the FPGA. The output from the secondary controller is connected to the primary controller as **VICINTSOURCE31**.

Interrupt sources **VICINTSOURCE[30:21]** are used for interrupts from an interface board or from peripherals located in the FPGA.

Table 4-32 SIC implementation

Property	Value
Location	FPGA
Memory base address	`0x10003000`
Interrupt	31 on the primary interrupt controller

Table 4-32 SIC implementation (continued)

Property	Value
DMA	NA
Release version	custom logic
Reference documentation	*Secondary interrupt controller* on page 4-48 and *Interrupts* on page 3-49)

4.9.1 Primary interrupt controller

The primary interrupt control registers are listed in Table 4-33. For more detail on the primary interrupt controller, see the *ARMPL190 VIC Technical Reference Manual*.

To simplify programming, some registers have multiple addresses. For example:

PICIntEnable

This register contains the enable interrupt enable bits. A value written to this register replaces the original contents.

PICIntEnClear

A value written to this register resets the original contents on a bit by bit basis. If a bit is HIGH in PCIIntEnClear, the corresponding bit in PICIntEnable is set LOW. If a bit is LOW in PCIIntEnClear, the corresponding bit in PICIntEnable is not affected.

Table 4-33 Primary interrupt controller registers

Name	Address	Access	Description
PICIRQStatus	0x10140000	Read	IRQ status register
PICFIQStatus	0x10140004	Read	FIQ status register
PICRawIntr	0x10140008	Read	Raw interrupt status register
PICIntSelect	0x1014000C	Read/write	Interrupt select register
PICIntEnable	0x10140010	Read/write	Interrupt enable register
PICIntEnClear	0x10140014	Write	Interrupt enable clear register
PICSoftInt	0x10140018	Read/write	Software interrupt register
PICSoftIntClear	0x1014001C	Write	Software interrupt clear register

Table 4-33 Primary interrupt controller registers (continued)

Name	Address	Access	Description
PICProtection	0x10140020	Read/write	Protection enable register
PICVectAddr	0x10140030	Read/write	Vector address register
PICDefVectAddr	0x10140034	Read/write	Default vector address register
PICVectAddr0– PICVectAddr15	0x10140100–0x1014013C	Read/write	Vector address 0 register to Vector address 15 register
PICVectCntl0– PICVectCntl15	0x10140200–0x1014023C	Read/write	Vector control 0 register to Vector control 15 register
PICITCR, PICITIP1, PICITIP2, PICITOP1, PICITOP2,	0x10140300–0x10140310	Read/write	Test control registers
PICPeriphID0– PICPeriphID3	0x10140FE0–0x10140FEC	Read	Peripheral identification registers
PICPCellID0– PICPCellID3	0x10140FF0–0x10140FFC	Read	PrimeCell identification registers

The bit assignments for the primary interrupt controller are shown in Table 4-34. Each bit corresponds to an interrupt source. Use the bit to enable or disable the interrupt or to check the interrupt status.

Table 4-34 Interrupt signals to primary interrupt controller

Bit	Interrupt source[a]	Function
[31]	**VICINTSOURCE31**	External interrupt from secondary controller
[30]	-	Reserved
[29]	-	Reserved
[28]	**VICINTSOURCE28**	External interrupt signal from expansion connector
[27]	**VICINTSOURCE27**	External interrupt signal from expansion connector
[26]	**VICINTSOURCE26**	External interrupt signal from USB
[25]	**VICINTSOURCE25**	External interrupt signal from ETHERNET
[24]	**VICINTSOURCE24**	External interrupt signal from AACI
[23]	-	Reserved
[22]	**VICINTSOURCE22**	External interrupt signal from MCI0A

Table 4-34 Interrupt signals to primary interrupt controller (continued)

Bit	Interrupt source[a]	Function
[21]	**VICINTSOURCE21**	External interrupt signal from DiskOnChip flash device
[20]	GND	Reserved
[19]	MBX	Graphics processor on development chip
[18]	PWRFAIL	Power failure from FPGA
[17]	DMA	DMA controller in development chip
[16]	CLCD	CLCD controller in development chip
[15]	SCI0	Smart Card interface in development chip
[14]	UART2	UART2 on development chip
[13]	UART1	UART1 on development chip
[12]	UART0	UART0 on development chip
[11]	SSP	Synchronous serial port in development chip
[10]	RTC	Real time clock in development chip
[9]	GPIO3	GPIO controller in development chip
[8]	GPIO2	GPIO controller in development chip
[7]	GPIO1	GPIO controller in development chip
[6]	GPIO0	GPIO controller in development chip
[5]	Timer 2 or 3	Timers on development chip
[4]	Timer 0 or 1	Timers on development chip
[3]	Comms TX	Debug communications transmit interrupt. This interrupt indicates that the communications channel is available for the processor to pass messages to the debugger.

Table 4-34 Interrupt signals to primary interrupt controller (continued)

Bit	Interrupt source[a]	Function
[2]	Comms RX	Debug communications receive interrupt. This interrupt indicates to the processor that messages are available for the processor to read.
[1]	Software interrupt	Software interrupt. Enabling and disabling the software interrupt is done with the Enable Set and Enable Clear Registers. Triggering the interrupt however, is done from the Soft Interrupt Set register.
[0]	Watchdog	Watchdog timer

a. The **VICINTSOURCEx** signals are external to the FPGA. Other signals listed connect to the SIC internally only.

4.9.2 Secondary interrupt controller

The register map for the secondary interrupt controller is shown in Table 4-35 on page 4-49.

To simplify programming, some registers have multiple addresses. For example:

SIC_ENABLE

This register contains the enable interrupt enable bits. The register is read-only. The contents are changed by writing to the SIC_ENSET or SIC_ENCLR registers.

SIC_ENSET

A value written to this register sets the original contents on a bit by bit basis. If a bit is HIGH in SIC_ENSET, the corresponding bit in SIC_ENABLE is set HIGH. If a bit is LOW in SIC_ENSET, the corresponding bit in SIC_ENABLE is not affected.

SIC_ENCLR

A value written to this register resets the original contents on a bit by bit basis. If a bit is HIGH in SIC_ENCLR, the corresponding bit in SIC_ENABLE is set LOW. If a bit is LOW in SIC_ENCLR, the corresponding bit in SIC_ENABLE is not affected.

Table 4-35 Secondary interrupt controller registers

Register name	Address	Access	Description
SIC_STATUS	0x10003000	Read	Status of interrupt (after mask)
SIC_RAWSTAT	0x10003004	Read	Status of interrupt (before mask)
SIC_ENABLE	0x10003008	Read	Interrupt mask
SIC_ENSET	0x10003008	Write	Set bits in interrupt mask
SIC_ENCLR	0x1000300C	Write	Clear bits in interrupt mask
SIC_SOFTINTSET	0x10003010	Read/write	Set software interrupt
SIC_SOFTINTCLR	0x10003014	Write	Clear software interrupt
SIC_PICENABLE	0x10003020	Read	Read status of pass-through mask (allows interrupt to pass directly to the primary interrupt controller)
SIC_PICENSET	0x10003020	Write	Set interrupt pass through bits
SIC_PICENCLR	0x10003024	Write	Clear interrupt pass through bits

The bit assignments for the secondary interrupt controller are shown in Table 4-36 on page 4-50.

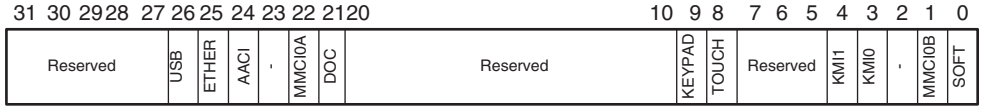

Figure 4-14 Secondary interrupt registers

Table 4-36 Interrupt signals to secondary interrupt controller

Bit	Interrupt source	Function
[31:27]	Reserved	NA
[26]	USB	USB controller ready for data or data available
[25]	ETHERNET	Ethernet controller ready for data or data available
[24]	AACI	Audio codec interface interrupt
[23]	Reserved	-
[22]	MMCI0A	Multimedia card 0A interrupt
[21]	DiskOnChip	Interrupt from DiskOnChip flash device
[20:10]	Reserved	-
[9]	Keypad	Key pressed on external display keypad
[8]	Touchscreen	Pen down on external display touchscreen
[7]	Reserved	-
[6]	Reserved	-
[5]	Reserved	-
[4]	KMI1	Activity on mouse port
[3]	KMI0	Activity on keyboard port
[2]	Reserved	-
[1]	MMCI0B	Multimedia card 0B interrupt
[0]	SOFTINT	Software interrupt from secondary controller (SIC_SOFTINT register)

4.9.3 Handling interrupts

This section describes interrupt handling and clearing in general. For examples of interrupt detection and handling, see the *ARM Developer Suite Developer Guide*, the *RealView Compilation Tools User Guide*, and the *ARM926EJ-S Technical Reference Manual*.

The majority of peripheral interrupts can be routed direct to the ARM926PXP development chip primary interrupt controller. Peripherals external to the development chip have their interrupts routed to the PIC through the SIC_PICEnable register or the SIC. Routing interrupts through the PIC_Enable register rather than the SIC provides a faster mechanism for reading external interrupts.

Note

Although the primary interrupt controller is a vectored interrupt controller (VIC), the examples in this section do not used vectored addresses.

To determine an interrupt source, read the STATUS registers in the PIC and SIC to determine the interrupt controller that generated the interrupt.

The sequence to determine and clear an interrupt is:

1. Determine the interrupt source by reading PIC_IRQStatus and SIC_STATUS.

   The interrupt handler must read PIC_IRQStatus first to determine if the interrupt was generated by a source that is connected directly to the PIC. If PIC_IRQStatus indicates that the interrupt source was the SIC, the SIC_STATUS register must be read identify the interrupting device.

2. Determine the nature of the interrupt by reading the peripheral masked interrupt status register.

3. Clear the peripheral interrupt by setting the appropriate bit in the peripheral interrupt clear register.

Each peripheral contains its own interrupt mask and clear registers that must be configured before an interrupt is enabled. The code segment in Example 4-1 shows how primary and secondary peripheral interrupts are enabled.

Example 4-1 shows an example of clearing and re-enabling the PIC SCI0 card out interrupt from the selftest program supplied on the CD.

Example 4-1 Clearing and re-enabling SCI0 card out interrupt

```
#define PIC_BASE          0x10140000
#define PIC_IntEnable     ((volatile unsigned int *)(PIC_BASE  + 0x10))
#define PIC_IntEnClear    ((volatile unsigned int *)(PIC_BASE  + 0x14))
#define PIC_SCI0          (1 << 15)   // Smart Card interrupt
#define SCI0_CARDOUTIM    0x002       // Card removed
#define SCI0_IMSC         ((volatile int *)(SCI0_BASE + 0x6C))
#define SCI0_ICR          ((volatile int *)(SCI0_BASE + 0x78))
*PIC_IntEnClear  = PIC_SCI0;         // Mask the PIC SCI0 interrupt
*SCI0_ICR        = SCI0_CARDOUTIM;   // Clear SCI0 card out flag
```

```
// ...
// code for managing SCI I/O
// ...
*SCI0_IMSC      |= SCI0_CARDOUTIM;   // Enable SCI0 card out interrupt
*PIC_IntEnable   = PIC_SCI0;         // Enable the PIC SCI0 interrupt
```

See the example code supplied on the CD for more details.

4.10 Keyboard and Mouse Interface, KMI

The ARM 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 Versatile/AB926EJ-S: KMI0 is used for keyboard input and KMI1 is used for mouse input.

Table 4-37 KMI implementation

Property	Value
Location	FPGA
Memory base address	`0x10006000` KMI 0 (keyboard)`0x10007000` KMI 1 (mouse)
Interrupt	3 on secondary controller KMI 0 4 on secondary controller KMI 1
DMA	NA
Release version	ARM KMI PL050 r1p0
Reference documentation	*ARM PrimeCell Keyboard Mouse Controller (PL050) Technical Reference Manual* (see also *Keyboard/Mouse Interface, KMI* on page 3-51)

4.11 MBX

The MBX Graphics Accelerator is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-38 MBX implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x40000000`
Interrupt	19 on primary controller
DMA	NA
Release version	MBX HR-S r1p2
Reference documentation	*ARM MBX Graphics Accelerator Technical Reference Manual* **Note** The MBX documentation is only available to licensees.

The ARM MBX HR-S contains a tile accelerator that operates on 3D scene data (sent as batches of triangles). The accelerator has a direct connection to the MPMC that allows rendered images to be saved directly into display memory.

4.12 MultiMedia Card Interface, MCI

The PrimeCell *Multimedia Card Interface* (MCI) is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited. The MCI provides the interface to MMC and SD cards.

Table 4-39 MCI implementation

Property	Value
Location	FPGA
Memory base address	`0x10005000`
Interrupt	22 on the primary and secondary controller
DMA	DMA channel 2, only DMACBREQ and DMACCLR are used.
Release version	ARM MCI PL180 r1p0
Reference documentation	*ARM PrimeCell Multimedia Card Interface (PL180) Technical Reference Manual* (see also *SD/MultiMedia Card Interface, MCI* on page 3-52)

4.13 MOVE video coprocessor

The MOVE coprocessor is a video encoding acceleration coprocessor designed to accelerate motion estimation algorithms within block-based video encoding schemes such as MPEG4 and H.263.

Details of the MOVE coprocessor function are only available to licensees. Contact ARM for information on licensing. The release version of the MOVE accelerator is MOVE r3p0-00bet0

Note

The MOVE documentation is only available to licences.

4.14 MultiPort Memory Controller, MPMC

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

Table 4-40 MPMC implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x10110000`
Interrupt	NA
DMA	The MPMC does not use interrupts or DMA. DMA transfers, however, can be set up to access memory controlled by the MPMC.
Release version	ARM MPMC GX175 r0p0-00alp2
Reference documentation	*ARM PrimeCell Multiport Memory Controller (GX175) Technical Reference Manual* (see also *Memory interface* on page 3-13)

The MPMC controls the dynamic memory on the Versatile/AB926EJ-S.

SyncFlash is supported on the dynamic memory bus but it cannot be selected as boot memory.

Sample programs that configure and use dynamic memory can be found on the CD that accompanies the Versatile/AB926EJ-S.

4.14.1 Register values

Table 4-41 on page 4-58 lists the register values for typical operation with 133MHz SDRAM. The HCLK frequency is 70MHz and the SDRAM is organized as four banks of 8MB x 16bit.

Note

The platform.a library contains memory setup routines. See *Building an application with the platform library* on page 2-21.

Table 4-41 SDRAM register values

Address offset	Register name	Value	Description
+0x000	MPMCControl	0x1	Enabled
+0x008	MPMCConfig	0x0	Little Endian
+0x020	MPMCDynamicControl	0x3	**MPMCCLKOUT** runs continuously, **CKE** high
+0x024	MPMCDynamicRefresh	0x22	544 cycles of HCLK between refreshes
+0x028	MPMCDynamicReadConfig	0x111	command delayed strategy, using MPMCCLKDELAY, datacapture on positive **HCLK** edge
+0x030	MPMCDynamictRP	0x2	42.86ns
+0x034	MPMCDynamictRAS	0x3	57.14ns
+0x038	MPMCDynamictSREX	0x5	85.71ns
+0x044	MPMCDynamictWR	0x4	71.43ns
+0x048	MPMCDynamictRC	0x5	85.71ns
+0x04c	MPMCDynamictRFC	0x5	85.71ns
+0x050	MPMCDynamictXSR	0x5	85.71ns
+0x054	MPMCDynamictRRD	0x1	28.57ns
+0x058	MPMCDynamictMRD	0x2	42.86ns
+0x05c	MPMCDynamictCDLR	0x1	28.57ns
+0x100	MPMCDynamicConfig0	0x5880	SDRAM32M16BRCX32
+0x104	MPMCDynamicRasCas0	0x202	CAS latency =2, RAS latency =2
+0x120	MPMCDynamicConfig1	0x5880	SDRAM32M16BRCX32
+0x124	MPMCDynamicRasCas1	0x202	CAS latency =2, RAS latency =2

Table 4-41 SDRAM register values (continued)

Address offset	Register name	Value	Description
+0x140	MPMCDynamicConfig2	0x5880	SDRAM32M16BRCX32
+0x144	MPMCDynamicRasCas2	0x202	CAS latency =2, RAS latency =2
+0x160	MPMCDynamicConfig3	0x5880	SDRAM32M16BRCX32
+0x164	MPMCDynamicRasCas3	0x202	CAS latency =2, RAS latency =2
+0x400	MPMCAHBControl0	0x0	-
+0x408	MPMCAHBTimeOut0	0x2	Timeout value

4.15 Real Time Clock, RTC

The 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 to provide 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 4-42 RTC implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x101E8000`
Interrupt	10 on the primary controller
DMA	NA
Release version	ARM RTC PL031 r1p0
Reference documentation	*ARM PrimeCell Real Time Clock Controller (PL031) Technical Reference Manual*

Note

There is also a separate Time-of-Year RTC implemented in an external DS1338U chip on the Versatile/AB926EJ-S. The external RTC can be accessed by the serial bus interface (see *Serial bus interface* on page 4-64). For details on the programming interface to the Time-of-Year RTC, see the datasheet for the Maxim DS1338 integrated circuit.

4.16 Smart Card Interface, SCI

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

Table 4-43 SCI implementation

Property	Value
Location	ARM926PXP development chip SCI0
Memory base address	`0x101F0000` for SCI0
Interrupt	15 on the primary controller (SCI0)
DMA	7 SCI0 transmit 6 SCI0 receive DMA channels for SCI1 are selectable as 0,1, or 2. See *Direct Memory Access Controller* on page 4-40.
Release version	ARM SCI PL131 r1p0
Reference documentation	*SCI PrimeCell PL131 Technical Reference Manual* (see also *Smart card interface, SCI* on page 3-55)

The following key parameters are programmable:

- Smart Card clock frequency and communication baud rate
- protocol convention
- card activation and deactivation time
- check for maximum time for first character of *Answer To Reset* (ATR) reception
- check for maximum duration of ATR character stream
- check for maximum time for receipt of first character of data stream
- check for maximum time allowed between characters
- character and block guard time
- transmit and receive character retry and FIFO level
- clock start and stop time and inactive level.

See the self-test software that is supplied on the CD accompanying the Versatile/AB926EJ-S for a example of detecting a SIM card response to a reset.

4.17 Synchronous Serial Port, SSP

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

Table 4-44 SSP implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x101F4000` for SSP.
Interrupt	11 on primary controller
DMA	9 for transmit 8 for receive
Release version	ARM SSP PL022 r1p2
Reference documentation	*ARM PrimeCell Synchronous Serial Port Controller (PL022) Technical Reference Manual* (see also *Synchronous Serial Port, SSP* on page 3-57)

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

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

In both master and slave configurations, the PrimeCell SSP performs:

- parallel-to-serial conversion on data written to a transmit FIFO
- serial-to-parallel conversion and FIFO buffering of received data.

Interrupts are generated to:

- request servicing of the transmit and receive FIFO
- inform the system that a receive FIFO over-run has occurred
- inform the system that data is present in the receive FIFO.

Note

Some SSP signals used for touchscreen expansion are controlled by the SYS_CLCD register (see *CLCD Control Register, SYS_CLCD* on page 4-28).

4.18 Synchronous Static Memory Controller, SSMC

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

Table 4-45 SSMC implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x10100000`
Interrupt	NA
DMA	NA
Release version	ARM SSMC PL093 r0p0-00rel0
Reference documentation	*ARM PrimeCell Static Memory Controller (PL093) Technical Reference Manual* and *Configuration and initialization* on page 4-13

The following key parameters are programmable for each SSMC memory bank:

- external memory width, 8, 16, or 32-bit
- burst mode operation
- write protection
- external wait control enable
- external wait polarity
- write WAIT states for static RAM devices
- read WAIT states for static RAM and ROM devices
- initial burst read WAIT state for burst devices
- subsequent burst read WAIT state for burst devices
- read byte lane enable control
- bus turn-around (idle) cycles
- output enable and write enable output delays.

4.19 Serial bus interface

A serial bus interface is implemented in the FPGA. The registers shown in Table 4-47 control the serial bus. The serial bus enables access to control signals for memory modules located on the interface boards and to the time-of-year clock implemented in the DS1338U chip.

Table 4-46 Serial bus implementation

Property	Value
Location	FPGA
Memory base address	`0x10002000`
Interrupt	NA
DMA	NA
Release version	Custom logic
Reference documentation	The datasheet for the Dallas Maxim DS1338 Real Time Clock.

Table 4-47 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**

Note

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

Software must manipulate the **SCL** and **SDA** bits directly to access the data in the devices. The pre-defined eight-bit device addresses are listed in Table 4-48. See the `\firmware\examples` directory on the CD for example code for reading the memory expansion EEPROM.

Table 4-48 Serial bus device addresses

Device	Write address	Read address
Static expansion E2PROM	`0xA2`	`0xA3`
Time-of-year clock	`0xD0`	`0xD1`

4.20 System Controller

The *ARM PrimeXSys Controller* is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited.

Table 4-49 System controller implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x101E0000`
Interrupt	NA
DMA	NA
Release version	ARM SYSCTRL SP810 r0p0-00ltd0
Reference documentation	*ARM PrimeXSys Controller (SP810) Technical Reference Manual*. See also *Status and system control registers* on page 4-19. **Note** Bit 8 of the System controller register at `0x101E000` controls remapping of static memory devices to address `0x0`. See also *Remapping of boot memory* on page 4-13.

The system controller in the ARM926PXP development chip provides an interface to control the operation of the chip.

The PrimeXSys System Controller supports the following functionality:

- a system mode control state machine
- 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.

Note

System controller registers for PLL frequency, AHB bridge asynchronous mode, and clock dividers are not supported for the Versatile/AB926EJ-S. The clock dividers have fixed sources and divide ratios.

4.21 Timers

The Dual-Timer module is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited. There are two Dual-Timer modules present in the ARM926PXP development chip.

Table 4-50 Timer implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x101E2000` for Timer 0 `0x101E2020` for Timer 1 `0x101E3000` for Timer 2 `0x101E3020` for Timer 3.
Interrupt	4 on primary controller for Timers 0 and 1 5 on primary controller for Timers 2 and 3
DMA	NA
Release version	ARM Dual-Timer SP804 r1p0-02ltd0
Reference documentation	*ARM PrimeCell Timer Module (SP804) Technical Reference Manual*

The features of the Dual-Timer module are:

- Two 32/16-bit down counters with free-running, periodic and one-shot modes.
- Common clock with separate clock-enables for each timer gives flexible control of the timer intervals.
- Interrupt output generation on timer count reaching zero.
- Identification registers that uniquely identify the Dual-Timer module. These can be used by software to automatically configure itself.

At reset, the timers are clocked by a 32KHz reference from an external oscillator module. Use the system controller to change the timer reference from 32KHz to 1MHz (see the *ARM926EJ-S Development Chip Reference Manual*).

4.22 UART

The PrimeCell UART is an AMBA compliant SoC peripheral that is developed, tested, and licensed by ARM Limited. There are three UARTs in the ARM926PXP development chip. The 24MHz reference clock to the UARTs come from the crystal oscillator that is part of OSC0.

Table 4-51 UART implementation

Property	Value
Location	ARM926PXP development chip for UART 0-2
Memory base address	`0x101F1000` for UART0`0x101F2000` for UART1`0x101F3000` for UART2
Interrupt	12 on PIC for UART0 13 on PIC for UART1 14 on PIC for UART2
DMA	15 UART0 Tx 14 UART0 Rx 13 UART1 Tx 12 UART1 Rx 11 UART2 Tx 10 UART2 Rx
Release version	ARM UART PL011 r1p3
Reference documentation	*ARM PrimeCell UART (PL011) Technical Reference Manual* (see also *UART interface* on page 3-62)

The following key parameters are programmable:

- communication baud rate, integer, and fractional parts
- number of data and stop bits
- parity mode
- FIFO enable (16 deep) or disable (1 deep)
- FIFO trigger levels selectable between 1/8, 1/4, 1/2, 3/4, and 7/8
- UART or IrDA protocol
- hardware flow control.

4.22.1 PrimeCell modifications

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

- receive FIFO trigger levels are 1/8, 1/4, 1/2, 3/4, and 7/8
- the internal register map address space, and the bit function of each register differ
- the deltas of the modem status signals are not available.
- 1.5 stop bits not available (1 or 2 stop bits only are supported)

- no independent receive clock.

4.23 USB interface

The USB interface is provided by an OTG243 controller that provides an *On-The-Go* (OTG) dual role device controller. The OTG can function as either a host or slave device.

Table 4-52 USB implementation

Property	Value
Location	Board (an OTG243 chip)
Memory base address	`0x10020000`, the registers in the OTG243 are memory-mapped onto the AHB M2 bus
Interrupt	26 on PIC and SIC
DMA	None
Release version	Custom interface in FPGA to external OTG243 controller
Reference documentation	*TransDimension OTG243 Data Sheet* (see also *USB interface* on page 3-60 and example programs supplied on the CD)

The OTG243 has the following features:

- fully compliant to the USG On-The-Go specification
- configurable number of downstream and upstream hosts or functions
- USB host is USB 2.0 compliant and supports 12Mb/s and 1.5Mb/s
- 4KB on-chip RAM.

The OTG243 register base addresses are shown in Table 4-53.

Table 4-53 USB controller base address

Address	Description
`0x10020000`	Chip-level register bank
`0x10020080`	Host controller register bank
`0x10020100`	Function controller register bank

4.24 Vector Floating Point, VFP9

The *VFP9-S coprocessor* is an implementation of the *Vector Floating-point Architecture version 2* (VFPv2). It provides low-cost floating-point computation that is fully compliant with the *ANSI/IEEE Std. 754-1985, IEEE Standard for Binary Floating-Point Arithmetic*. The VFP9-S coprocessor supports all addressing modes described in section 5 of the *ARM Architecture Reference Manual*.

Table 4-54 VFP9 implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	The VFP registers are not memory-mapped. Access is from the coprocessor instructions.
Interrupt	NA
DMA	NA
Release version	VFP9 r0p1
Reference documentation	*ARM VFP9 Coprocessor Technical Reference Manual*

Note

The following operations from the IEEE 754 standard are not supplied by the VFP9-S instruction set:

- remainder
- round floating-point number to integer-valued floating-point number
- binary-to-decimal conversions
- decimal-to-binary conversions
- direct comparison of single-precision and double-precision values.

Complete implementation of the IEEE 754 standard is achieved by support code that is provided with the ARM compilation tools.

The latest VFP support code can be obtained as part of *Application Note 98*. If you are using RealView Compilation Tools (RVCT), the appropriate code and documentation are provided within your installation. If you are using the ARM Developer Suite (ADS) 1.2, *Application Note 98* can be downloaded from `www.arm.com/support/`.

4.25 Watchdog

The PrimeCell Watchdog module is an AMBA compliant SoC peripheral developed, tested and licensed by ARM Limited. The Watchdog module consists of a 32-bit down counter with a programmable timeout interval that has the capability to generate an interrupt and a reset signal on timing out. It is intended to be used to apply a reset to a system in the event of a software failure.

Note

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

Table 4-55 Watchdog implementation

Property	Value
Location	ARM926PXP development chip
Memory base address	`0x101E1000`
Interrupt	0 on primary controller
DMA	NA
Release version	ARM WDOG SP805 r1p0-02ltd0
Reference documentation	*ARM PrimeCell Watchdog Controller (SP805) Technical Reference Manual*

The following Watchdog module parameters are programmable:

- interrupt generation enable/disable
- interrupt masking
- reset signal generation enable/disable
- interrupt interval.

Appendix A
Signal Descriptions

This appendix provides a summary of signals present on the Versatile/AB926EJ-S connectors. It contains the following sections:

For more information on connectors used on the Versatile/AB926EJ-S, see the parts list spreadsheet in the CD `schematics` directory.

A.1 Audio CODEC interface

The Versatile/AB926EJ-S jack connectors J2 and J3 enable you to connect to the microphone input and line level output on the CODEC. Figure A-1 shows the pinouts of the sockets.

Note

A link on the board enables bias voltage to be applied to the microphone (see *Advanced Audio Codec Interface, AACI* on page 3-38).

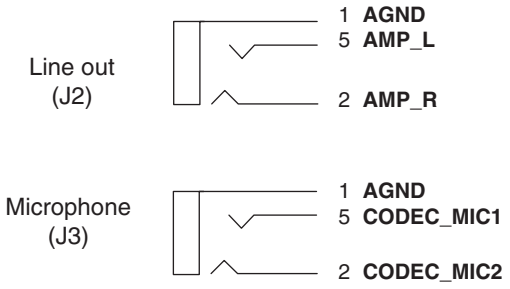

Figure A-1 Audio connectors

A.2 Ethernet interface

The RJ45 Ethernet connector J4 is shown in Figure A-2.

LEDA (green) and LEDB (yellow) are connected to the LAN91C111 controller. The function of the LEDs is determined by registers in the controller. Typical usage would be to monitor transmit activity and packet detection.

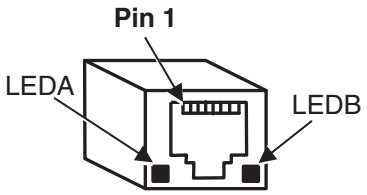

Figure A-2 Ethernet connector J4

The signals on the Ethernet cable are shown in Table A-1.

Table A-1 Ethernet signals

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

A.3 Peripheral expansion connector

The peripheral expansion connector signals are listed in Table A-2. The signals on this connector interface to peripheral devices (such as LCD displays) mounted on an attached expansion board. For details on expansion board signals, see Appendix C *Versatile/AB-IB1 Interface Board* and Appendix D *Versatile/AB-IB2 Interface Board*.

Note

The peripheral expansion connector consists of two 48-pin connectors placed end to end. The connector has one position (pin 35) blocked so that the interface board cannot be installed incorrectly.

Table A-2 Peripheral expansion connector J25/J26

Pin	Signal	Pin	Signal
1	**3V3**	2	**3V3**
3	**CLCD_B2**	4	**CLCD_B3**
5	**CLCD_B4**	6	**CLCD_B5**
7	**CLCD_B6**	8	**CLCD_B7**
9	**CLCD_G2**	10	**CLCD_G3**
11	**CLCD_G4**	12	**CLCD_G5**
13	**CLCD_G6**	14	**CLCD_G7**
15	**CLCD_R2**	16	**CLCD_R3**
17	**CLCD_R4**	18	**CLCD_R5**
19	**CLCD_R6**	20	**CLCD_R7**
21	**GND**	22	**GND**
23	**CLLE**	24	**CLAC**
25	**CLCP**	26	**CLLP**
27	**CLFP**	28	**CLPOWER**
29	**nLCDIOON**	30	**PWR3V35VSWITCH**
31	**VDDPOSSWITCH**	32	**VDDNEGSWITCH**

Table A-2 Peripheral expansion connector J25/J26 (continued)

Pin	Signal	Pin	Signal
33	**LCDID0**	34	**LCDID1**
35	Location pin	36	**LCDID2**
37	**LCDID3**	38	**LCDID4**
39	**GND**	40	**GND**
41	**GP0 0**	40	**GP0 1**
43	**GP0 2**	42	**GP0 3**
45	**GP0 4**	44	**GP0 5**
47	**GP0 6**	46	**GP0 7**
49	**GP1 0**	40	**GP1 1**
51	**GP1 2**	42	**GP1 3**
53	**GP1 4**	44	**GP1 5**
55	**GP1 6**	46	**GP1 7**
57	**GND**	58	**GND**
59	**nUART1RTS**	60	**nUART1TXD**
61	**nUART1CTS**	62	**nUART1RXD**
63	**nUART2RTS**	64	**nUART2TXD**
65	**nUART2CTS**	66	**nUART2RXD**
67	**SSPCLKOUT**	68	**SSPTXD**
69	**SSPRXD**	70	**SSPnCS**
71	**TSnSS**	72	**TSnDAVX**
73	**TSnPENIRQX**	74	**TSnKPADIRQX**
75	**REFCLK24MHZ2E**	76	**EXP_XTALCLK**
77	**GND**	78	**GND**
79	**EXP_TDI**	80	**EXP_TDO**
81	**EXP_TCK**	82	**EXP_TMS**

Table A-2 Peripheral expansion connector J25/J26 (continued)

Pin	Signal	Pin	Signal
83	**EXP_nTRST**	84	**EXP_nSRST**
85	**EXP_nCFGEN**	86	**GLOBALDONE**
87	**VICINTSOURCE27**	88	**VICINTSOURCE28**
89	**EXP_nDET**	90	**nSYSPOR**
91	**5V**	92	**5V**
93	**GND**	94	**GND**
95	**DCIN**	96	**DCIN**

A.4 Static memory expansion connector

The memory connector signals are listed in Table A-3. The signals on this connector interface to memory devices mounted on an attached expansion board.

Note

The memory expansion connector consists of two 48-pin connectors placed end to end. The connector has one position (pin 35) blocked so that the interface board cannot be installed incorrectly.

If a Versatile/AB-IB1 or Versatile/AB-IB2 interface board is installed, a memory module can be connected to the interface board.

Table A-3 Static memory connector J1/4

Pin	Signal	Pin	Signal
1	3V3	2	3V3
3	**EXP_SMDATA0**	4	**EXP_SMDATA1**
5	**EXP_SMDATA2**	6	**EXP_SMDATA3**
7	**EXP_SMDATA4**	8	**EXP_SMDATA5**
9	**EXP_SMDATA6**	10	**EXP_SMDATA7**
11	**EXP_SMDATA8**	12	**EXP_SMDATA9**
13	**EXP_SMDATA10**	14	**EXP_SMDATA11**
15	**EXP_SMDATA12**	16	**EXP_SMDATA13**
17	**EXP_SMDATA14**	18	**EXP_SMDATA15**
19	**EXP_SMDATA16**	20	**EXP_SMDATA17**
21	**EXP_SMDATA18**	22	**EXP_SMDATA19**
23	**EXP_SMDATA20**	24	**EXP_SMDATA21**
25	**EXP_SMDATA22**	26	**EXP_SMDATA23**
27	**EXP_SMDATA24**	28	**EXP_SMDATA25**
29	**EXP_SMDATA26**	30	**EXP_SMDATA27**
31	**EXP_SMDATA28**	32	**EXP_SMDATA29**

Table A-3 Static memory connector J1/4 (continued)

Pin	Signal	Pin	Signal
33	**EXP_SMDATA30**	34	**EXP_SMDATA31**
35	Location pin	36	**GND**
37	**EXP_SMADDR0**	38	**EXP_SMADDR1**
39	**EXP_SMADDR2**	40	**EXP_SMADDR3**
41	**EXP_SMADDR4**	42	**EXP_SMADDR5**
43	**EXP_SMADDR6**	44	**EXP_SMADDR7**
45	**EXP_SMADDR8**	46	**EXP_SMADDR9**
47	**EXP_SMADDR10**	48	**EXP_SMADDR11**
49	**EXP_SMADDR12**	50	**EXP_SMADDR13**
51	**EXP_SMADDR14**	52	**EXP_SMADDR15**
53	**EXP_SMADDR16**	54	**EXP_SMADDR17**
55	**EXP_SMADDR18**	56	**EXP_SMADDR19**
57	**EXP_SMADDR20**	58	**EXP_SMADDR21**
59	**EXP_SMADDR22**	60	**EXP_SMADDR23**
61	**EXP_SMADDR24**	62	**EXP_SMADDR25**
63	**GND**	64	**GND**
65	**EXP_SMCLK0**	66	**EXP_SMCLK1**
67	**SBSCL**	68	**SBSDA**
69	**EXPnSTATICCS4**	70	**EXPnSTATICCS5**
71	**EXPnSTATICCS6**	72	**EXPnSTATICCS7**
73	**EXP_nEXPCS**	74	**Nreset**
75	**EXP_nSMBLS0**	76	**EXP_nSMBLS1**
77	**EXP_nSMBLS2**	78	**EXP_nSMBLS3**
79	**EXP_SMADDRVALID**	80	**EXP_SMBAA**
81	**EXP_nSMWEN**	82	**EXP_SMOEN**

Table A-3 Static memory connector J1/4 (continued)

Pin	Signal	Pin	Signal
83	**EXP_SMBUFDIR**	84	**EXP_nSMBUFOE**
85	**EXPCSWIDTH1**	86	**EXPCSWIDTH0**
87	**SMWAIT**	88	**nSMBURSTWAIT**
89	**SMCANCELWAIT**	90	**nFLWP**
91	**nDOCBUSY**	92	**nDOCRESET**
93	**nDOCIRQ**	94	**nDOCDMARQ**
95	**GND**	96	**GND**

A.5 Keyboard and mouse interface

The pinout of the KMI connector J7 is shown in Figure A-3.

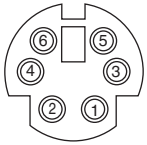

Figure A-3 KMI connector J7

Note

A standard keyboard cable will operate correctly if plugged into the KMI connector. To use both a keyboard and a mouse, use a splitter cable.

Table A-4 shows signals on the KMI connector.

Table A-4 Mouse and keyboard port signal descriptions

Pin	Signal	Function
1	**MDATA**	Mouse data
2	**KDATA**	Keyboard data
3	**GND**	Ground
4	**5V**	5V
5	**MCLK**	Mouse clock
6	**KCLK**	Keyboard clock

A.6 MMC and SD flash card interface

The MMC/SD card socket J6 provides nine pins that connect to the card when it is inserted into the socket. Figure A-4 shows the pin numbering and signal assignment. In addition, the socket contains switches that are operated by card insertion and provide signaling on the **CARDINx** and **MCI_WPROT** signals.

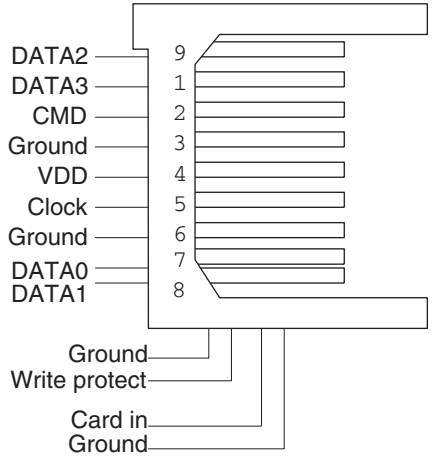

Figure A-4 MMC/SD card socket J6 pin numbering

The MMC card uses seven pins, and the SD card uses all nine pins. The additional pins are located as shown in Figure A-4 with pin 9 next to pin 1 and pins 7 and 8 spaced more closely together than the other pins. Figure A-5 shows an MCI card, with the contacts face up.

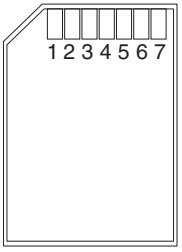

Figure A-5 MMC card

Table A-5 lists the signal assignments.

Table A-5 Multimedia Card interface signals

Pin	Signal	Function SD widebus mode	Function MCI
1	**MCI0DATA3**	Data	Chip select
2	**MCI0CMD**	Command/response	Data in
3	**GND**	Ground	Ground
4	**MCIVDD0**	Supply voltage	Supply voltage
5	**MCICLK0**	Clock	Clock
6	**GND**	Ground	Ground
7	**MCI0DATA0**	Data 0	Data out
8	**MCI0DATA1**	Data 1	NC
9	**MCI0DATA2**	Data 2	NC
10 (DET A)	**CARDIN0**	Card insertion detect	Card insertion detect
11 (DET B)	**WPROT0**	Write protect status	Write protect status

MMC connector is J6 on the bottom side of the board.

Insert and remove the card as follows:

Insertion The connector is on the bottom of the board. Insert the card into the socket with the contacts face up as viewed from the top of the board. Cards are normally labeled on the top surface with an arrow to indicate correct insertion.

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

A.7 Smart card interface

The Versatile/AB926EJ-S contains a Smart Card SIM socket J10 mounted on the bottom of the board. There are pads on the board for a different layout connector (J9) that can be used instead of connector J10.

Socket J10 includes a switch for card detection. The alternate connector, J9, does not have a card detect switch.

The signals associated with the SCI are shown in Table A-6.

Table A-6 Smartcard connector signal assignment

Pin	Signal	Description
1	**SC_VCC0**	Card power (1.8V, 3.3V, or 5V)
2	**SC_RST0**	Reset to card
3	**SC_CLK0**	Clock to or from card
4	NC	Not present on J10, NC on J9.
5	GND	Ground
6	**SC_VCC0**	Card power (1.8V, 3.3V, or 5V)
7	**SC_DATA0**	Serial data to or from the card
8	NC	Not present on J10, NC on J9.
SW1	**nSCIDETECT0**	Card detect signal from switch in socket (not present on J25 and J26)

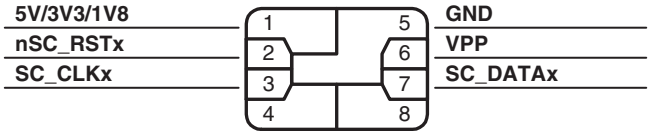

Figure A-6 Smartcard contacts assignment

Figure A-6 shows the signal assignment of a smartcard. Pins 4 and 8 are not connected and are omitted on some cards.

The SIM card is inserted of the SIM card sockets with the contacts face up.

A.8 UART interface

The Versatile/AB926EJ-S provides three serial transceivers. UART0 is connected to serial connector J8 and UART1 and UART2 are connected to the expansion connector.

Figure A-7 shows the pin numbering for the 9-pin D-type male connector (J8) used on the Versatile/AB926EJ-S and Table A-7 shows the signal assignment for the connectors.

UART0 (J8) is configured as a *Data Terminal Equipment* (DTE) device. Figure A-7 shows the view looking into the pins on the male connector mounted on the board.

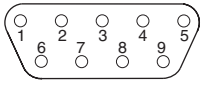

Figure A-7 Serial connector J8

Table A-7 Serial plug signal assignment

Pin	Direction	UART0 J8
1	Input	**SER0_DCD**
2	Input	**SER0_RX**
3	Output	**SER0_TX**
4	Output	**SER0_DTR**
5	-	**SER0_GND**
6	Input	**SER0_DSR**
7	Output	**SER0_RTS**
8	Input	**SER0_CTS**
9	Input	**SER0_RI**

A.9 USB interface

Connector J5 provides an OTG interface.

Note

For a full description of the USB signals refer to the datasheet for the TransDimension OTG243.

The USB debug port connector is dedicated to the JTAG debug system and cannot be used by the system for general-purpose I/O.

Figure A-8 shows the OTG USB connector.

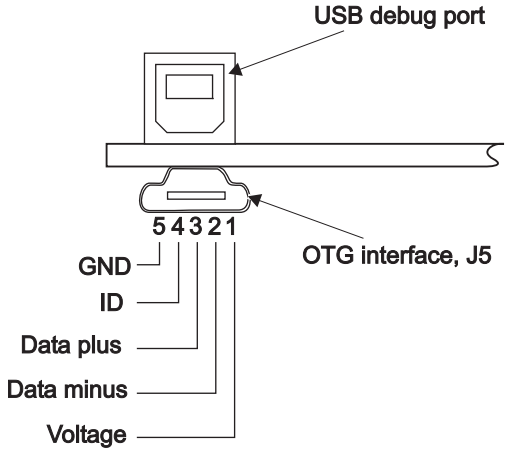

Figure A-8 OTG socket J5

A.10 VGA display interface

The VGA connector (J1) is shown in Figure A-9. The connector is a DB15 female connector.

The connector signals are listed in Table A-8. The analog VGA signals are generated from the digital CLCD data and synchronization signals.

Table A-8 VGA connector signals

Pin	Description
1	**RED**
2	**GREEN**
3	**BLUE**
4	NC
5	**GND**
6	**GND**
7	**GND**
8	**GND**
9	NC
10	**GND**
11	NC
12	NC
13	**HSYNC**
14	**VSYNC**
15	NC

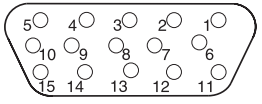

Figure A-9 VGA connector J1

A.11 Battery connector

The battery connector enables the connection of a custom battery pack and charger (not included with the Versatile/AB926EJ-S). The battery connector pinout is shown in Figure A-10.

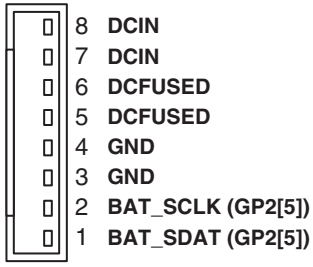

Figure A-10 Battery connector

A.12 Test and debug connections

The Versatile/AB926EJ-S provides test points, ground points, and connectors to aid diagnostics as shown in Figure A-11. The test point numbers in parenthesis indicate that the test point is on the bottom of the board.

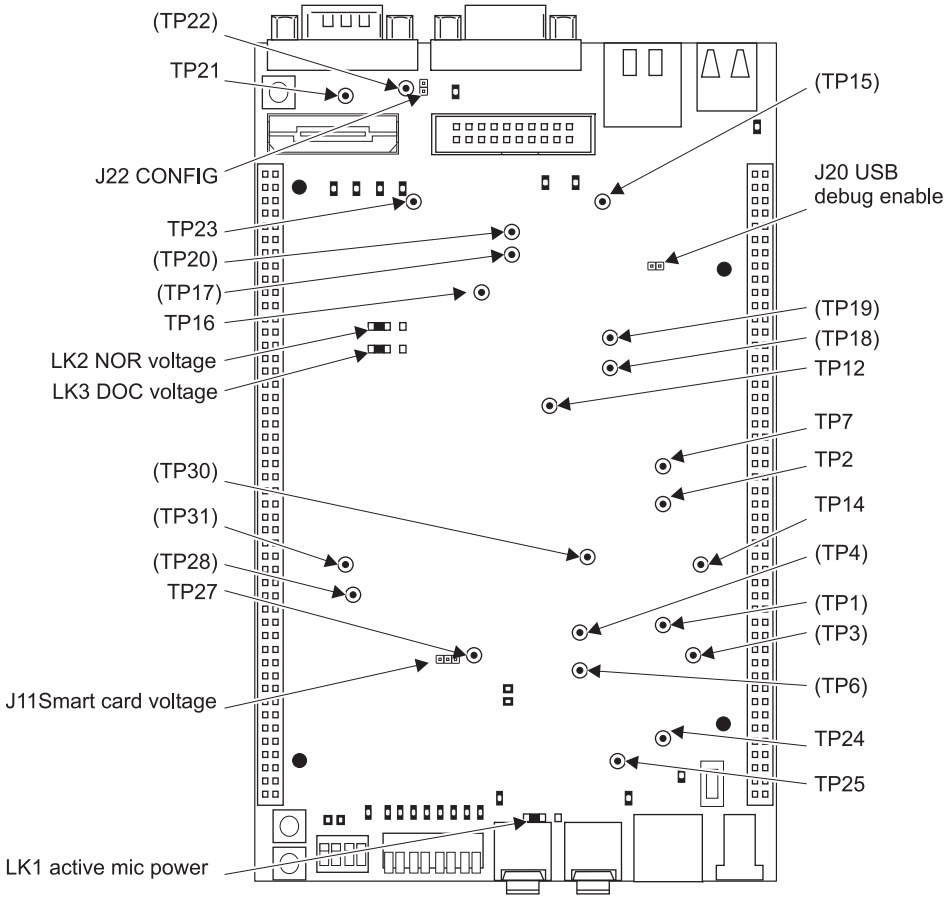

Figure A-11 Test points and debug connectors

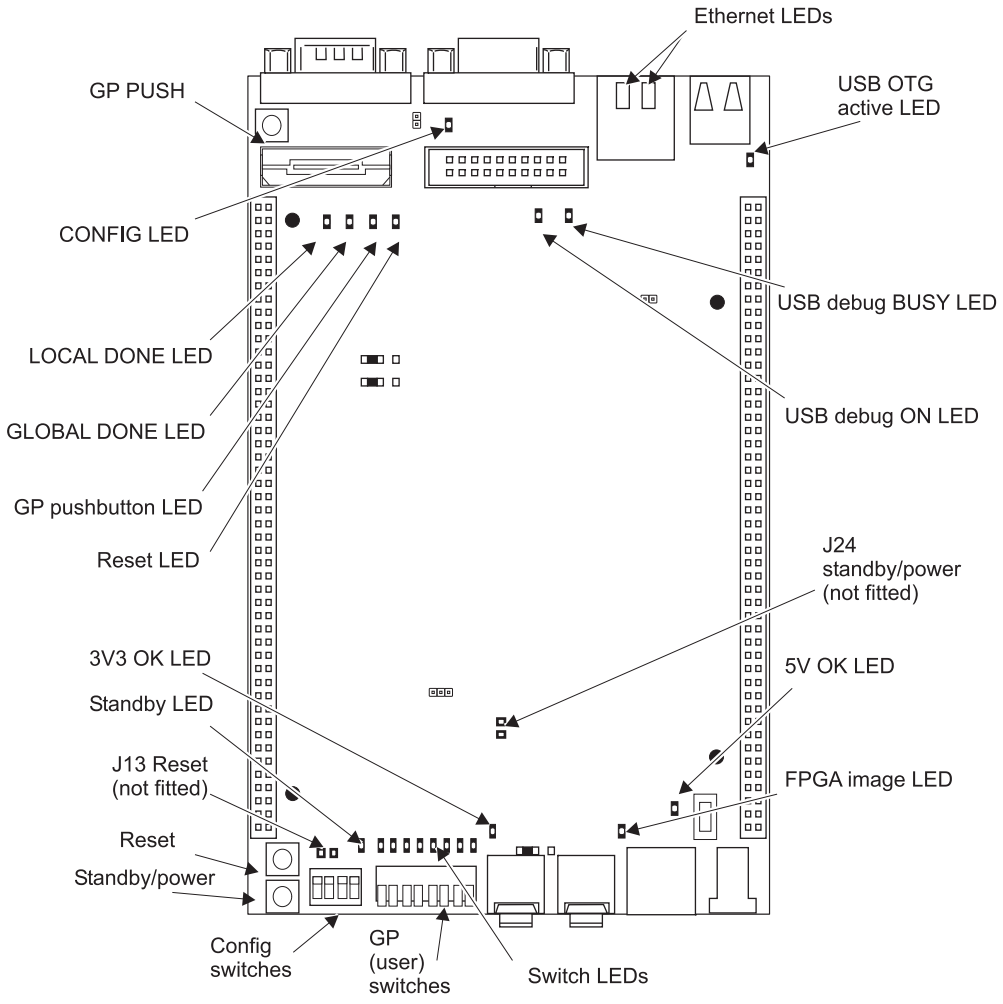

Figure A-12 Switches and LEDs

This section contains the following subsections:

- *Overview of links and test points* on page A-20
- *JTAG* on page A-21
- *USB debug port* on page A-22
- *Trace connector pinout* on page A-22.

A.12.1 Overview of links and test points

The functions of the test points on the Versatile/AB926EJ-S are summarized in Table A-9.

Table A-9 Test point functions

Test point	Signal	Description
TP1	**VBATT**	1.5V cell battery voltage for FPGA encryption key and RTC
TP2	**XTALCLK**	Output from programmable oscillator 0
TP3	**OSC1**	LCDCLK, output from programmable oscillator 1
TP4	**HCLKTP**	Buffered version of XTALCLK from oscillator 0
TP5	**nSRST**	JTAG reset
TP6	**REFCLK24MHZ**	24MHz output from oscillator 0 crystal reference
TP7	**REFCLK32K**	32kHZ output from oscillator module
TP12	**3V3**	3.3V supply voltage
TP14	**1V8**	1.8V supply voltage
TP15	**IntCLK**	Clock signal from USB debug interface
TP16	**EnRST**	USB debug reset signal
TP17	**REFCLK1**	Output from ICS525 oscillator
TP18	**SPARE1**	Spare test signal from USB debug interface controller
TP19	**SPARE1**	Spare test signal from USB debug interface controller
TP20	**CLK**	Output from ICS525 oscillator
TP21	**SDC_TDI**	JTAG data input to ARM926EJ-S development chip
TP22	**PLD_TDO**	JTAG data output to PLD
TP23	**FPGA_TDI**	JTAG input data from FPGA
TP24	**DCPROT**	DC input voltage (after fuse)
TP25	**5V ANALOG**	Filtered 5V voltage to audio CODEC
TP27	**AHBMONITOR33**	AHB monitor clock

Table A-9 Test point functions (continued)

Test point	Signal	Description
TP28	**HTRANSM2**	M2 bus control signal
TP30	**MPMCCLK3**	SDRAM memory clock
TP31	**HADDRM2**	M2 address signal

Table A-10 Links

Link	Description
LK1	Microphone power for CODEC
LK2	NOR flash voltage
LK3	USB power
J11	Voltage select for Smart Card
J13	Parallel connection to RESET pushbutton
J14	Parallel connection to GP PUSH SWITCH
J20	Port enable for USB debug logic
J22	CFGEN jumper
J24	Parallel connection to POWER pushbutton

A.12.2 JTAG

Figure A-13 on page A-22 shows the pinout of the JTAG connector J21 and Table 3-18 on page 3-68 provides a description of the JTAG and related signals. All JTAG active HIGH input signals have pull-up resistors (DGBRQ is active LOW and has a pull-down resistor).

Note

The term JTAG equipment refers to any hardware that can drive the JTAG signals to devices in the scan chain. Typically this is RVI or Multi-ICE, although hardware from other suppliers can also be used to debug ARM processors.

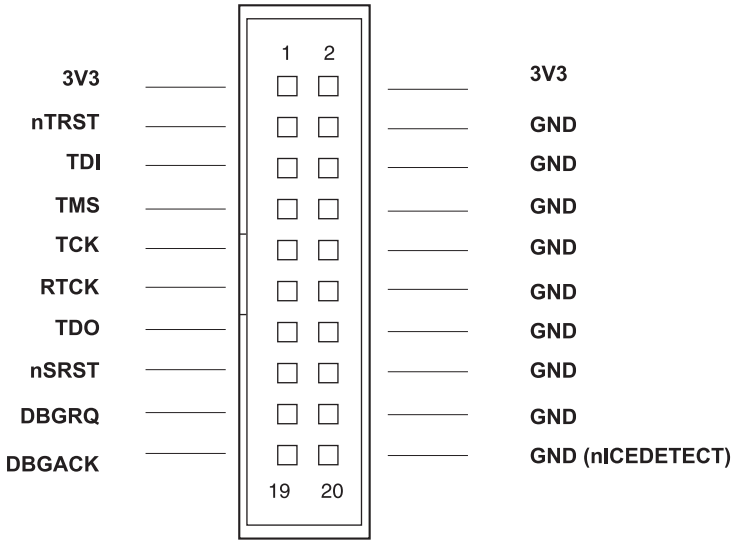

Figure A-13 Multi-ICE JTAG connector J21

A.12.3 USB debug port

Figure A-14 shows the signals on the USB debug connector J19. **USBDP** and **USBDM** are the positive and negative USB data signals.

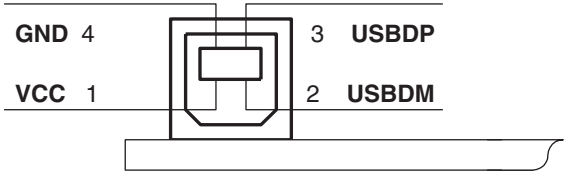

Figure A-14 USB debug connector J19

A.12.4 Trace connector pinout

Table A-11 on page A-23 lists the pinout of the trace connector J12. The Mictor connector is shown in Figure A-15 on page A-23.

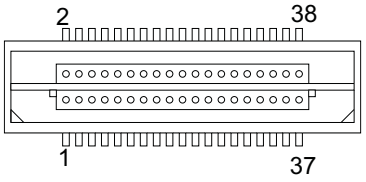

Figure A-15 Mictor connector J12

Table A-11 Trace signals

Channel	Pin	Pin	Channel
Not connected	1	2	Not connected
Not connected	3	4	Not connected
GND	5	6	**TRACECLK**
DBGRQ	7	8	**DBGACK**
nSRST	9	10	**EXTTRIG**
TDO	11	12	**3V3**
RTCK	13	14	**3V3**
TCK	15	16	**TRACEPKT7**
TMS	17	18	**TRACEPKT6**
TDI	19	20	**TRACEPKT5**
nTRST	21	22	**TRACEPKT4**
TRACEPKT15	23	24	**TRACEPKT3**
TRACEPKT14	25	26	**TRACEPKT2**
TRACEPKT13	27	28	**TRACEPKT1**
TRACEPKT12	29	30	**TRACEPKT0**
TRACEPKT11	31	32	**PIPESTAT3**
TRACEPKT10	33	34	**PIPESTAT2**
TRACEPKT9	35	36	**PIPESTAT1**
TRACEPKT8	37	38	**PIPESTAT0**

Appendix B
Specifications

This appendix contains the specification for the Versatile/AB926EJ-S. It contains the following sections:

B.1 Electrical specification

This section provides details of the voltage and current characteristics for the Versatile/AB926EJ-S.

B.1.1 Bus interface characteristics

Table B-1 shows the Versatile/AB926EJ-S electrical characteristics.

Table B-1 Versatile/AB926EJ-S electrical characteristics

Symbol	Description	Min	Max	Unit
DC IN	DC input voltage	6	15	V
V_{IH}	High-level input voltage	2.0	3.6	V
V_{IL}	Low-level input voltage	0	0.8	V
V_{OH}	High-level output voltage	2.4	-	V
V_{OL}	Low-level output voltage	-	0.4	V
C_{IN}	Capacitance on any input pin	-	20	pF

B.1.2 Current requirements

This section lists the current requirements of the Versatile/AB926EJ-S.

Table B-2 shows the current requirements at room temperature and nominal voltage powered from the DC IN connector. These measurements include the current drawn by Multi-ICE, approximately 160mA at 3.3V.

Table B-2 Current requirements from DC IN (12V)

System	DC IN typical	DC IN max
Standalone	0.7A	3A
with interface board	0.9A	3A

Loading on supply voltage rails

Table B-3 lists the maximum current load that can be placed on the supply voltage rails.

Table B-3 Maximum current load on supply voltage rails

System	5V	3.3V	1.8V
Supplied from DC IN (12V at 3A)	1A	2A	1A

Note

The AB-IB2 Interface Board has an additional 3.8V regulator to supply the GSM module and the prototyping area. The maximum load on the 3.8V supply is 1A.

B.2 Clock rate restrictions

The clock rates specified in Table B-4 are the maximum values that can be used for reliable operation.

Caution

The ICS307 programmable oscillators OSC0 and OSC1 can be programmed to deliver very high clock signals (200MHZ). The only ARM926PXP development chip clock input that can function at this frequency is **PLLCLKEXT**.

Also, the settings for VCO divider, output divider, and output select values are interrelated and must be set correctly. Some combinations of settings do not result in stable operation. For more information on the ICS clock generator and a frequency calculator, see the ICS web site at `www.icst.com`.

Reliable use of DMA might require lowering the bus frequency. See the `readme.txt` file for more information on timing. For timing information for ARM926PXP development chip signals, see the *ARM926PXP development chip Reference Manual*.

Table B-4 Maximum clock rates

Function	Signal	Description	Maximum frequency
CPU core	**CPUCLK**	This internal clock is normally generated from the PLL by multiplying up the **XTALCLKEXT** frequency. The ARM926PXP development chip can, however, be configured to use **PLLCLKEXT** as **CPUCLK**.	210MHz
SDRAM	**MPMCCLK**	This clock, together with the delay settings in the MPMC, determines the access time for the dynamic memory.	70MHz
Internal AMBA bus	**HCLK**	This is an internal clock generated from **CPUCLK** and the HCLK divider.	70MHz
External AMBA bus	**HCLKEXT**	This internal clock is generated from **HCLK** and the HCLKEXT divider. It clocks the output part of the AMBA M2 bridges in synchronous mode (with **HCLKM2**). This clock is at the same frequency as **XTALCLKEXT**.	35MHz
MBX	**MBXCLK**	This is an internal clock generated from **HCLK** and the MBX clock divider. The maximum frequency for **MBXCLK** is limited by the SDRAM memory devices.	70MHz
Flash memory	**SMCLK**	This clock is generated from **HCLK** and the SMC clock divider. It is used to time accesses to static memory.	54MHz

B.3 Mechanical details

Figure B-1 shows the mechanical outline of the Versatile/AB926EJ-S.

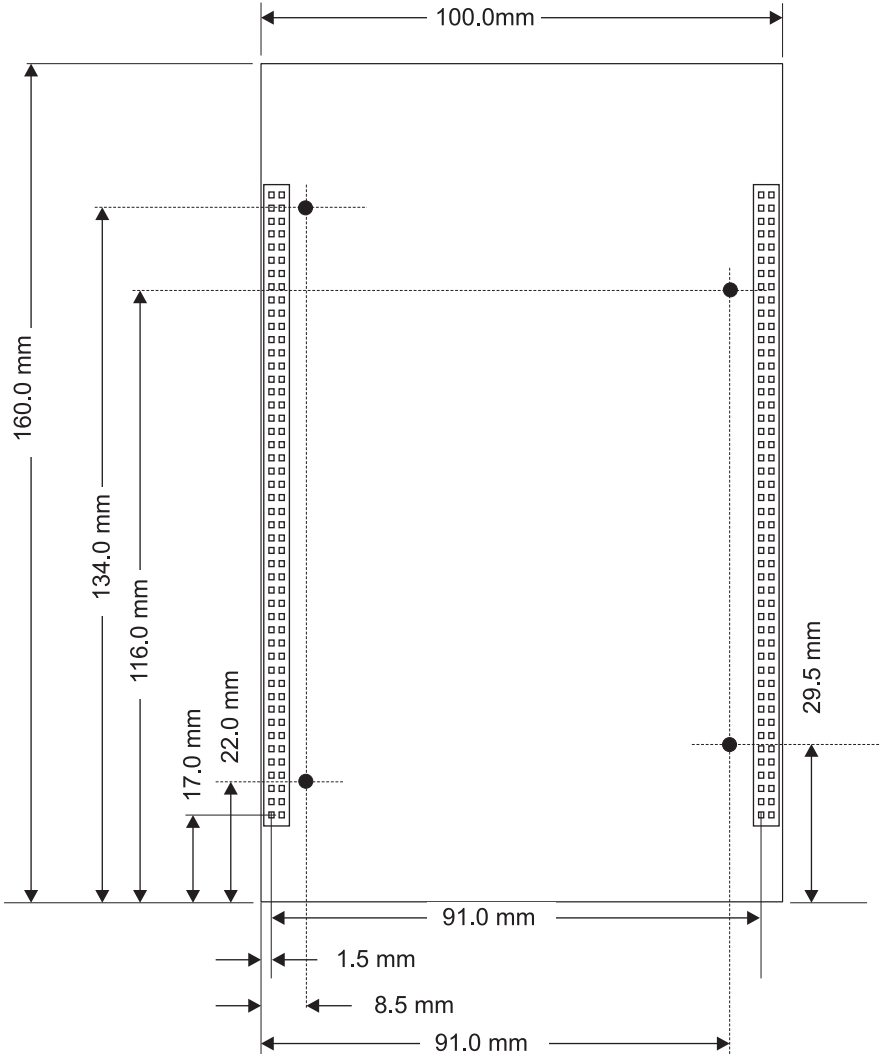

Figure B-1 Versatile/AB926EJ-S mechanical details

Appendix C
Versatile/AB-IB1 Interface Board

This appendix describes the Versatile/AB-IB1 (Application Baseboard Interface Board 1) for the Versatile/AB926EJ-S. It contains the following sections:

C.1 Introduction

The Versatile/AB-IB1 interface board provides connectors that enable you to access signals on the Versatile/AB926EJ-S:

- two serial connectors
- a synchronous serial port expansion connector
- static memory expansion connector
- GPIO expansion connector
- Interrupt signals **VICINTSOURCE27**, **VICINTSOURCE28**, **TSnKPADIRQX**, and **TSPENIRQX**
- connector for an external CLCD kit with a 2.2, 3.8, or 8.4 inch display.

Figure C-1 shows the layout of the board.

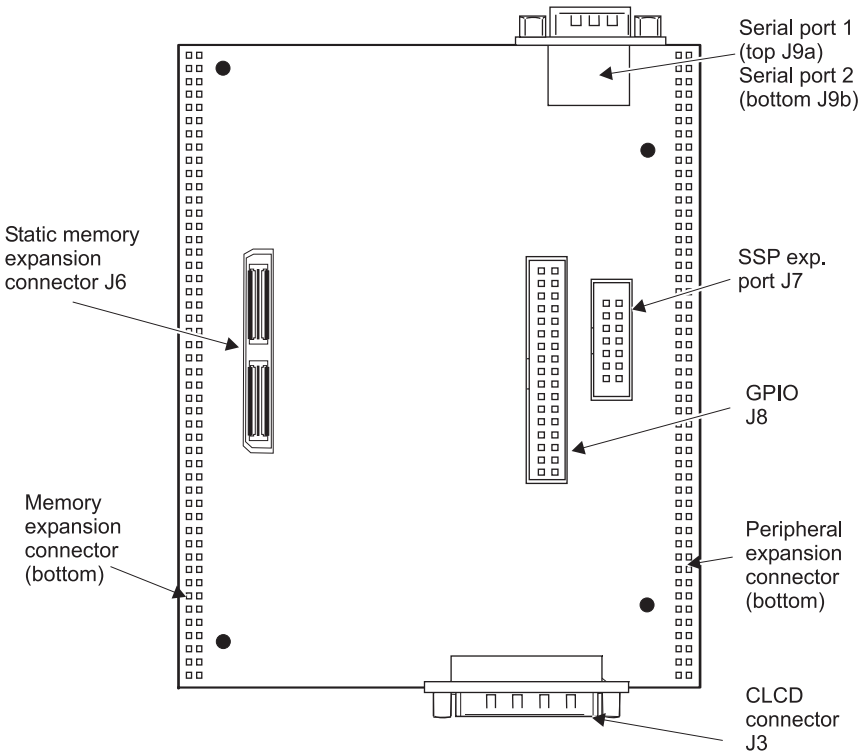

Figure C-1 Versatile/AB-IB1 mechanical details

C.2 Fitting the Versatile/AB-IB1 board to the Versatile/AB926EJ-S

Mount the AB-IB1 board onto the Versatile/AB926EJ-S as shown in Figure C-2.

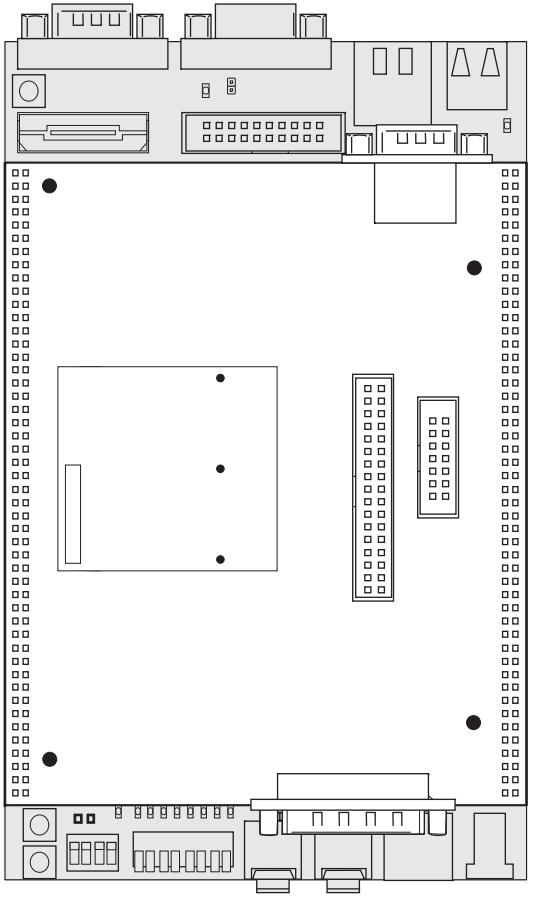

Figure C-2 Installing the Versatile/AB-IB1

If required, connect the CLCD cable, serial devices, or static memory module to the appropriate connectors.

C.3 Connectors

This section lists the signals on the interface board connectors.

C.3.1 Serial ports

The UART1 and UART2 signals from the ARM926PXP development chip are available on interface board serial connector J9.

Figure C-3 shows the pin numbering for the 9-pin D-type male connector used on the and Table C-1 shows the signal assignment for the connectors.

The pinout shown in Figure C-3 is configured as a *Data Communications Equipment* (DCE) device.

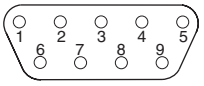

Figure C-3 Serial connector

Table C-1 Serial plug signal assignment

Pin	UART1 J9B (bottom)	UART2 J9A (top)
1	NC	NC
2	**UART1_RX**	**UART2_RX**
3	**UART1_TX**	**UART2_TX**
4	**UART1_DTR**[a]	**UART2_DTR**[a]
5	**UART1_GND**	**UART2_GND**
6	**UART1_DSR**	**UART2_DSR**
7	**UART1_RTS**	**UART2_RTS**
8	**UART1_CTS**	**UART2_CTS**
9	NC	NC

a. The signals **UART1_DTR** and **UART2_DTR**, and are connected to the corresponding **UART1_DSR** and **UART2_DSR** signals. These signals cannot be set or read under program control.

C.3.2 CLCD expansion

The CLCD adaptor board connector (J3) is shown in Figure C-4 on page C-7. The connectorsignals are listed in Table C-2.

Table C-2 CLCD adaptor board connector J3

Pin	Signal	Pin	Signal
1	**B0**	35	**B1**
2	**B2**	36	**B3**
3	**B4**	37	**B5**
4	**B6**	38	**B7**
5	**G0**	39	**G1**
6	**G2**	40	**G3**
7	**G4**	41	**G5**
8	**G6**	42	**G7**
9	**R0**	43	**R1**
10	**R2**	44	**R3**
11	**R4**	45	**R5**
12	**R6**	46	**R7**
13	**CLLE**	47	**GND**
14	**CLAC**	48	**GND**
15	**CLCP**	49	**GND**
16	**CLLP**	50	**GND**
17	**CLFP**	51	**GND**
18	**TSnKPADIRQ**	52	**GND**
19	**TSnPENIRQ**	53	**GND**
20	**TSnDAV**	54	**LCDID0**
21	**TSSCLK**	55	**LCDID1**
22	**TSnSS**	56	**LCDID2**

Table C-2 CLCD adaptor board connector J3 (continued)

Pin	Signal	Pin	Signal
23	**TSMISO**	57	**LCDID3**
24	**TSMOSI**	58	**LCDID4**
25	**LCDXWR**	59	**GND**
26	**LCDSD0**	60	**GND**
27	**LCDXRD**	61	**GND**
28	**LCDXCS**	62	**3V3**
29	**LCDDATnCOM**	63	**3V3**
30	**LCDSD0OUTnIN**	64	**5V**
31	**CLPOWER**	65	**5V**
32	**nLCDIOON**	66	**VLCD**
33	**PWR3V5VSWITCH**	67	**VLCD**
34	**VDDPOSSWITCH**	68	**VDDNEGSWITCH**

Note

The **R[7:0]**, **G[7:0]**, and **B[7:0]** signals are digital CLCD signals.

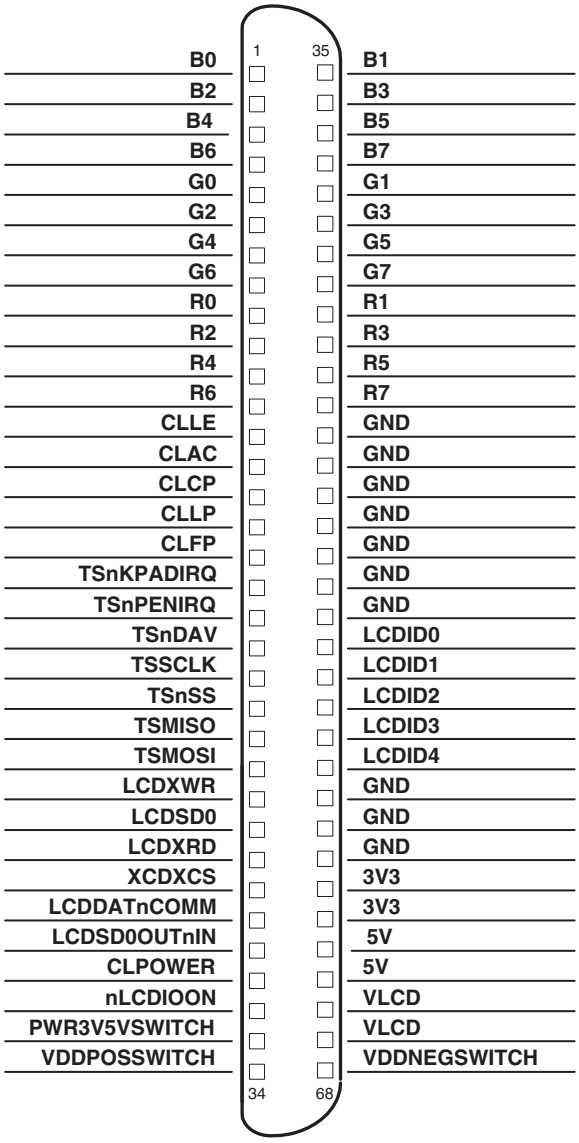

Figure C-4 CLCD Interface connector J3

C.3.3 SSP expansion

Figure C-5 on page C-8 shows the signals on the expansion SSP interface connector J7.

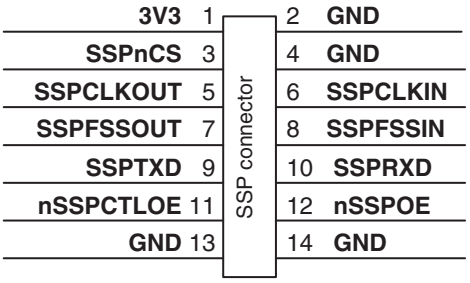

Figure C-5 SSP expansion interface J7

The signals associated with the SSP are shown in Table C-3.

Note

Some of the SSP signals are also connected to the touchscreen logic on the interface board.

Also, some signals that are used on the Versatile/PB926EJ-S are not connected on the Versatile/AB926EJ-S. Figure C-5 and Table C-3 lists all the signals.

Table C-3 SSP signal assignment

Description	Signal	Pin		Signal	Description
Supply voltage	**3V3**	1	2	**GND**	Ground
Chip select	**SSPnCS**	3	4	**GND**	Ground
Clock output from controller	**SSPCLKOUT**	5	6	**SSPCLKIN**[a]	Clock input to controller
Frame sync output	**SSPFSSOUT**[a]	7	8	**SSPFSSIN**[a]	Frame sync input
Ground	**SSPTXD**	9	10	**SSPRXD**	Data input
Control output enable control	**nSSPCTLOE**[a]	11	12	**nSSPOE**[a]	Data output enable control
Ground	**GND**	13	14	**GND**	Ground

a. Not connected on the Versatile/AB926EJ-S.

C.3.4 GPIO expansion

Two eight-bit *General Purpose Input/Output* (GPIO) controllers are incorporated into the ARM926PXP development chip. The signals are available on GPIO connector J8 as shown in Figure C-6.

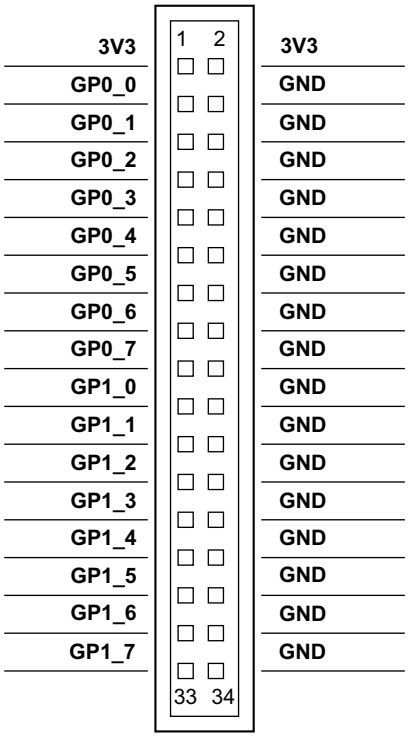

Figure C-6 GPIO connector J8

Note

Each GPIO data pin has an 10KΩ pullup resistor (on the Versatile/AB926EJ-S) to 3.3V.

C.3.5 Static memory expansion

The static memory signals from the SSMC in the ARM926PXP development chip are connected to the expansion connector. This enables an expansion memory module to be installed on the interface board. The static memory board uses a 120-way Samtec connector as shown in Figure C-7 on page C-10.

Note

The numbering of pins on the connectors is for the connectors as viewed from below.

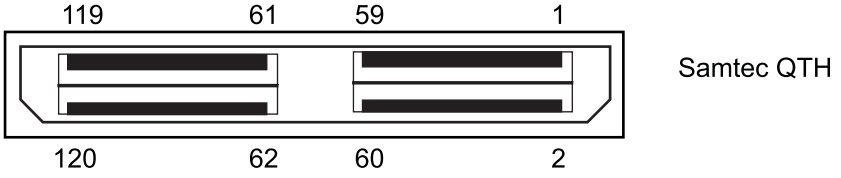

Figure C-7 Samtec connector J6

See Appendix F *Static Memory Expansion Board* for details on memory modules.

Caution

Use only 3.3V memory expansion modules.

Appendix D
Versatile/AB-IB2 Interface Board

This appendix describes the Versatile/AB-IB2 (Application Baseboard Interface Board 2) for the Versatile/AB926EJ-S. It contains the following sections:

D.1 Introduction

The AB-IB2 interface board adds the following devices to the Versatile/AB926EJ-S:

- keyboard with 19 keys and a joystick
- 2.5 inch CLCD display
- static memory expansion connector
- Prototyping area
- GSM interface

Figure D-1 shows a block diagram of the AB-IB2.

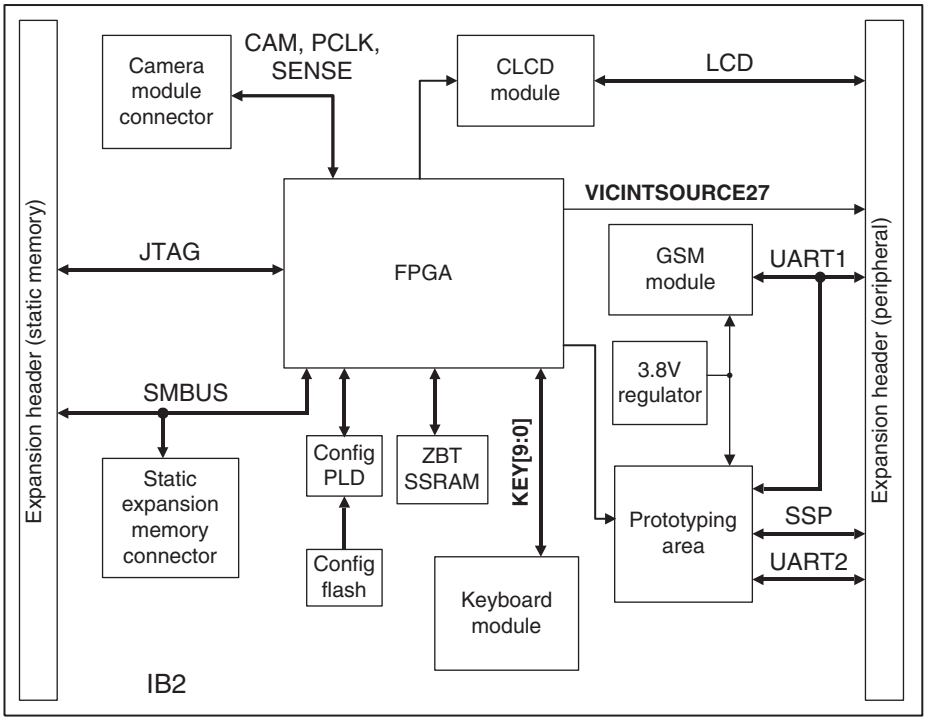

Figure D-1 AB-IB2 block diagram

Figure D-2 on page D-3 shows the layout of the interface board, camera module, and keyboard module.

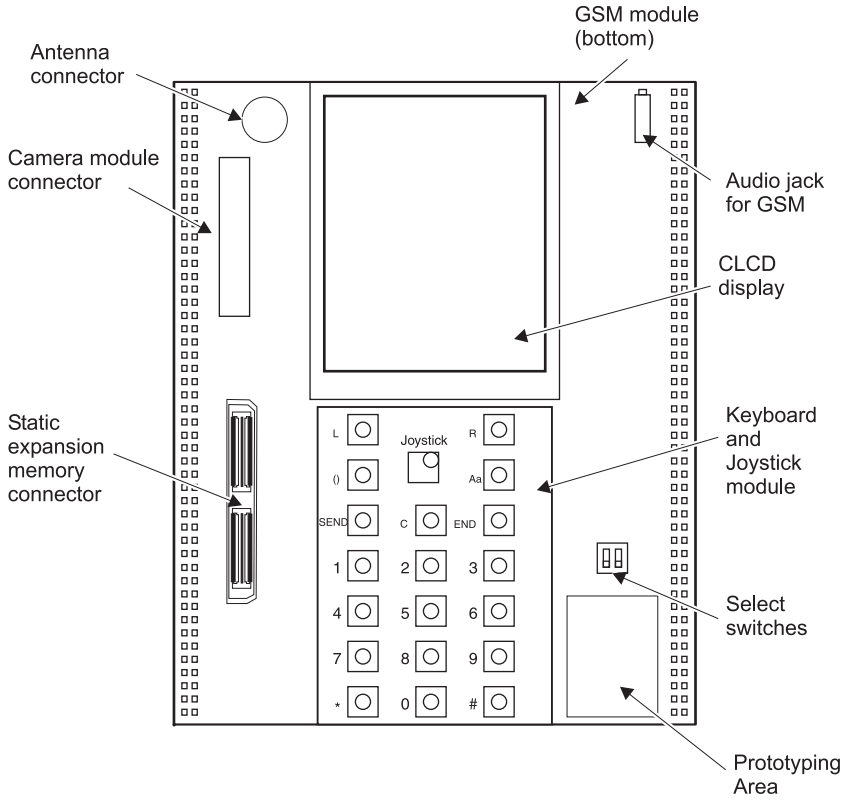

Figure D-2 AB-IB2 layout

Note

The S1 select switch controls two signals (**SEL1** and **SEL2**) connected to the PLD,.

These signals are reserved for future use.

D.2 Fitting the Versatile/AB-IB2 board to the Versatile/AB926EJ-S

Mount the AB-IB2 board onto the Versatile/AB926EJ-S as shown in Figure D-3.

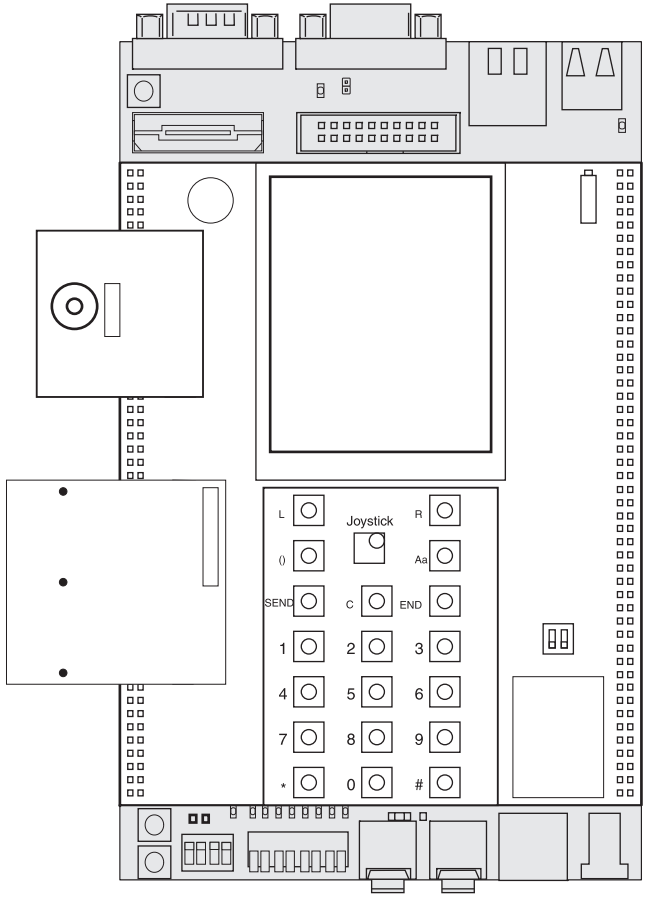

Figure D-3 Installing the IB2

The 2.5 inch CLCD module and keyboard module are fitted at manufacture. If required, connect the camera module or static memory module to the appropriate connectors.

See the selftest software on the CD provided with the product for more information on using the peripherals present on the AB-IB2 board.

D.3 Software interface

The FPGA on the AB-IB2 is memory-mapped onto the static-memory interface and uses **nSTATICCS5**. The register organization is listed in Table D-1.

Table D-1 Memory map

Address region on DATA AHB bus	Function
0x24000000–0x241FFFFF	Camera image memory bank:0x24000000–0x241FFFFF for single-frame capture0x24000000–0x240FFFFF multi-frame capture bank 00x24100000–0x24FFFFFF multi-frame capture bank 1
0x25000000	Keyboard data register
0x26000000	Interrupt enable register (for **VICINTSOURCE27**)
0x26000004	Interrupt status register
0x27000004	Status register
0x27000000	Control register

D.3.1 Control register

This register signals controls the modules on the interface board.

Table D-2 Control register

Bit	Name	Description	Reset Value
[31:8]	RESERVED	Always write as 0	0
[7]	CTRL_TEST	Test Mode (always write as 0)	0
[6]	CTRL_ICENB	Camera Image Capture Enable	0
[5]	CTRL_IMODE	Camera Image Capture Mode, 0 for multi-frame capture 1 for single-frame mode	0
[4]	CTRL_LEDEN	LED Enable, 1 for LED (D1) ON, 0 for LED OFF	0
[3]	CTRL_GSMEN	GSM Enable	0

Table D-2 Control register (continued)

Bit	Name	Description	Reset Value
[2]	CTRL_BTRST	Bluetooth Reset (to prototyping area)	0
[1]	CTRL_LCDSD	LCD Shutdown	0
[0]	CTRL_LCDBE	LCD Backlight Enable	0

Camera module signals

The camera image capture sub-system can operate in either a single-frame (snapshot) capture mode at high resolution or a multi-frame (movie) capture mode at lower resolutions. The selection of the capture mode is made using the CTRL_IMODE bit in the Control register:

Single-frame capture

In single frame capture mode an image is captured into memory and capture ceases at the end of a single captured frame. This can be detected by polling for the STAT_ACTIVE bit in the Status register to become LOW. Another image capture cannot be started until the CRTL_ICENB bit in the Control register is reset and then set again.

Multi-frame capture

In multi-frame capture mode the first frame is captured into memory bank 1 and the next frame is captured into memory bank 0. The capture bank number will then toggle every frame until the end of the frame following capture disable (disable is done by clearing the CTRL_ICENB bit in the Control register). The bank currently being written by the image capture sub-system can be monitored by reading the STAT_WBANK bit in the Status register.

End-of-frame events can be programmed to generate an interrupt by setting the appropriate bit in the interrupt enable register (IER), or can be detected by polling the STAT_VSYNC bit in the Status register.

Note

If the amount of data per frame received from the camera exceeds the size of a memory bank in this mode, image capture overwrites the contents of the inactive bank.

Image capture is enabled by setting the CTRL_ICENB bit in the Control register and stopped by clearing this bit.Image capture begins at the start of the frame following the setting of CTRL_ICENB. The image capture progress can be monitored by reading the STAT_ACTIVE bit in the Status register.

D.3.2 Status register

The status register returns the status of the camera module. A write to this register performs no function and should not be attempted

Table D-3 Status register

Bit	Name	Function
[31:16]	UNDEFINED	Undefined
[15:8]	FPGA_REV	FPGA revision number
[7:5]	UNDEFINED	Undefined
[4]	STAT_WBANK	Capture Memory Bank. Indicates the Memory bank currently being written to by the image capture sub-system
[3]	STAT_VSYNC	Camera VSYNC active. Set to 1 when a vsync is active (end of frame).
[2]	STAT_DETECT	Camera Presence Detect, Set to 1 when a camera is present
[1]	STAT_ORIENT	Camera Orientation. Set to 1 for 'Conference Mode' (image is wider than tall) Set to 0 for 'Camera Mode' (image is taller than wide)
[0]	STAT_ACTIVE	Image capture is active. Set to 1 when Capture to ZBT RAM is active. This bit will become set immediately after CTRL_ICENB is set and will become reset after a frame has been captured if in single-frame mode, or after capture of the last frame (following CTRL_ICENB reset) in multi-frame capture mode.

A camera can be attached in either of two orientations, and the current orientation can be detected by reading the STAT_ORIENT bit in the Status register.

The presence of a camera can be detected by reading the STAT_DETECT bit in the Status register. The bit is HIGH if a compatible camera is attached.

D.3.3 Interrupt status register

This read/write register provides the states of the pending interrupt sources (before the enable mask register). Set the appropriate status bit in this register to clear an active interrupt from that source. For example, write `0x02` to clear the End of Frame interrupt and `0x01` to clear the Keyboard interrupt.

Table D-4 Interrupt status register

Bit	Description
[31:2]	Reserved
[1]	End of frame from camera
[0]	Keyboard pressed event

D.3.4 Interrupt enable register

Each bit of this register enables a specific interrupt as detailed in the table below. Set a bit to enable the interrupt and reset the bit to disable it. The state of a disabled interrupt can be interrogated using the Interrupt status register. A read from this register returns the current value of the Interrupt enable register.

Table D-5 Interrupt enable register

Bit	Description
[31:2]	Reserved (write as 0)
[1]	End of frame
[0]	Keyboard event

D.3.5 Keypad controller register

This register reflects the current state of all the keys, any bit set to 1 indicates that the corresponding key is pressed. (See *Keyboard module* on page D-16 for details of key mapping.)

Table D-6 Keyboard data register

Bit	Description
[31:25]	Reserved
[24:0]	Key state (bit 24 is state of key 24, bit 0 is state of key 0) There is not a key fitted to position 19 on the module.

D.4 Expansion devices

This section lists the signals on the AB-IB2 connectors and modules.

For details of the signals on the static memory expansion connector and the peripheral expansion connector, see *Static memory expansion connector* on page A-7 and *Peripheral expansion connector* on page A-4.

D.4.1 CLCD module

The signals on interface connector (J11) for the 2.5 inch CLCD are listed in Table D-7 on page D-11. Use the Control register to modify the backlight (**BL**) and shutdown signals (**SD**) (see *Control register* on page D-5).

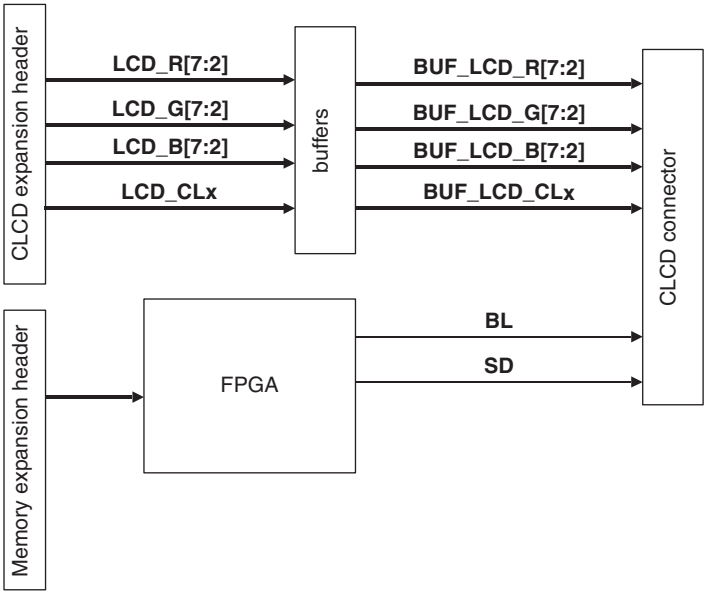

Figure D-4 CLCD block diagram

The connectors match a Sanyo ALR252RGT 2.5 inch quad-VGA color display. The CLCD module signals and function are listed in Table D-7.

Table D-7 CLCD module signals

Pin	Signal	Pin	Signal
1	**BUF_CLCP** (dot clock)	2	NC
3	**GND**	4	NC
5	**GND**	6	NC
7	**BUF_CLLP** (horiz. sync)	8	NC
9	**BUF_CLFP** (vert. sync)	10	**GND**
11	**GND**	12	**GND**
13	**BUF_CLAC** (data output enable)	14	**VCC** (back light 5V)
15	**BUF_R7** (red data)	16	**VCC** (back light 5V)
17	**BUF_R6**	18	**BL** (back light control)
19	**BUF_R5**	20	**VDD** (2.8V)
21	**BUF_R4**	22	**VDD** (2.8V)
23	**BUF_R3**	24	**GND**
25	**BUF_R2**	26	**EN16** (16-bit mode enable, tied to **GND**)
27	**BUF_G7** (green data)	28	**SD** (shut down)
29	**BUF_G6**	30	**BUF_B2** (blue data)
31	**BUF_G5**	32	**BUF_B3**
33	**BUF_G4**	34	**BUF_B4**
35	**BUF_G3**	36	**BUF_B5**
37	**BUF_G2**	38	**BUF_B6**
39	**GND**	40	**BUF_B7**

Configuration

The timing values for the display are listed in Table D-8.

Table D-8 Values for the QVGA TFT display

Manufacturer	Display resolution	CLCDCLK frequency and SYS_OSC1 register value	CLCD_TIM0 register at 0x10120000	CLCD_TIM1 register 0x10120004	CLCD_TIM2 register at 0x10120008
Sanyo ALR252 2.5 inch portrait QVGA Color TFT	240x320	5.4MHz, 0x2C13	0x141E0A38	0x0102093F	0x04EF3800

D.4.2 Static memory

The AB-IB2 has a 120-way Samtec QSH connector as shown in Figure D-5. The static memory board uses a corresponding 120-way Samtec QTH connector.

Note

The numbering of pins on the connectors is for the connectors as viewed from below.

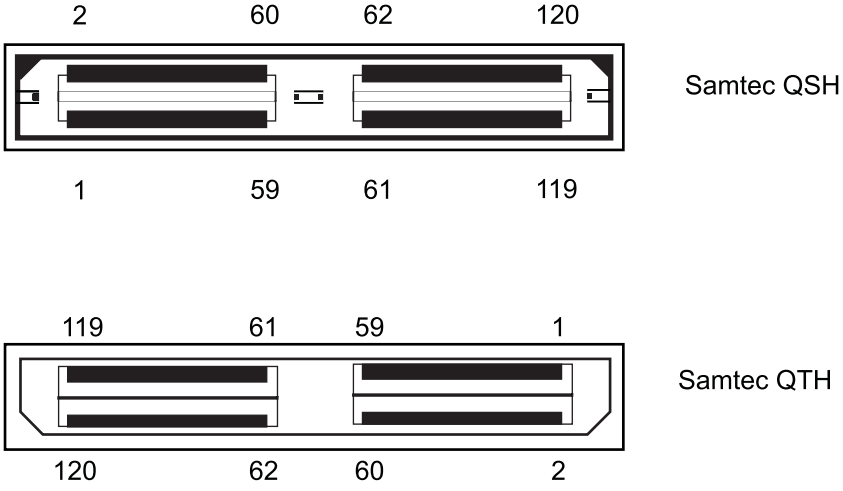

Figure D-5 Samtec connector J6

See Appendix F *Static Memory Expansion Board* for connector signals.

Caution

Use only 3.3V memory expansion modules.

D.4.3 Camera module

The 28-pin camera module connector (see Table D-9) enables the camera module to be inserted facing upward or downward. The **SENSE[1:0]** signals are set by the camera module and determine the function of the **CAM[13:0]** signals as listed in Table D-10 on page D-14.

Table D-9 Camera module connector J3

Signal	Pin	Pin	Signal
CLK_24MHZ	1	2	NC
3V3	3	4	**CAM0**
1V8	5	6	**CAM1**
GND	7	8	**CAM2**
SENSE0	9	10	**CAM3**
PCLK	11	12	**CAM4**
CAM7	13	14	**CAM5**
CAM8	15	16	**CAM6**
CAM9	17	18	**PCLK**
CAM10	19	20	**SENSE1**
CAM11	21	22	**GND**
CAM12	23	24	**1V8**
CAM13	25	26	**3V3**
NC	27	28	**CLK_24MHZ**

For more information on the camera unit, see the datasheet for the OmniVision OV9640 Concept Camera Module at www.ovt.com.

Table D-10 Camera module reverse function

Camera signal	SENSE0 active	SENSE1 active
CAM0	SIO_D	RESET
CAM1	SIO_C	SHDWN
CAM2	VSYNC	Y9
CAM3	HREF	Y8
CAM4	Y2	Y7
CAM5	Y3	Y6
CAM6	Y4	Y5
CAM7	Y5	Y4
CAM8	Y6	Y3
CAM9	Y7	Y2
CAM10	Y8	HREF
CAM11	Y9	VSYNC
CAM12	SHDWN	SIO_C
CAM13	RESET	SIO_D

The camera itself plugs into a 24-pin connector on the camera module. The camera signals are listed in Table D-11 on page D-15. The camera must not be removed from the camera module.

Table D-11 Camera signals

Pin	Module signal
1	NC
2	**GND**
3	**SIO_D**
4	**2.5V** (generated by regulator on camera module)
5	**SIO_C**
6	**RESET**
7	**VSYNC**
8	**SHDWN**
9	**HREF**
10	**1.8V**
11	**3.3V**
12	**Y9**
13	**XCLK1**
14	**Y8**
15	**GND**
16	**Y7**
17	**PCLK**
18	**Y6**
19	**Y5**
20	**Y4**
21	**Y3**
22	**Y2**
23	NC
24	NC

D.4.4 Keyboard module

The keyboard module signals are listed in Table D-13 on page D-17.

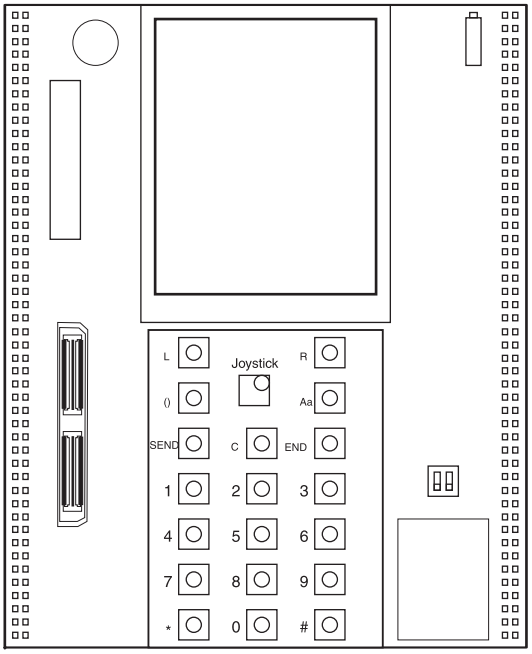

Figure D-6 Keyboard module

The keyboard uses row and column scanning. The matrix organization is listed in Table D-12.

Table D-12 Keyboard coding

	KCOL0	KCOL1	KCOL2	KCOL3	KCOL4
KROW0	SW0 (SEND)	SW1, (1)	SW2 (4)	SW3 (7)	SW4 (*)
KROW1	SW5 (C)	SW6 (2)	SW7 (5)	SW8 (8)	SW9 (0)
KROW2	SW10 (END)	SW11 (3)	SW12 (6)	SW13 (9)	SW14 (#)
KROW3	SW15 (FnL)	SW16 (UP/DOWN)	SW17 (FnR)	SW18 (Aa)	SW19 (not used)
KROW4	SW20 (JOYSTICK DOWN)	SW21 (JOYSTICK UP)	SW22 (JOYSTICK PRESS)	SW23 (JOYSTICK LEFT)	SW24 (JOYSTICK RIGHT)

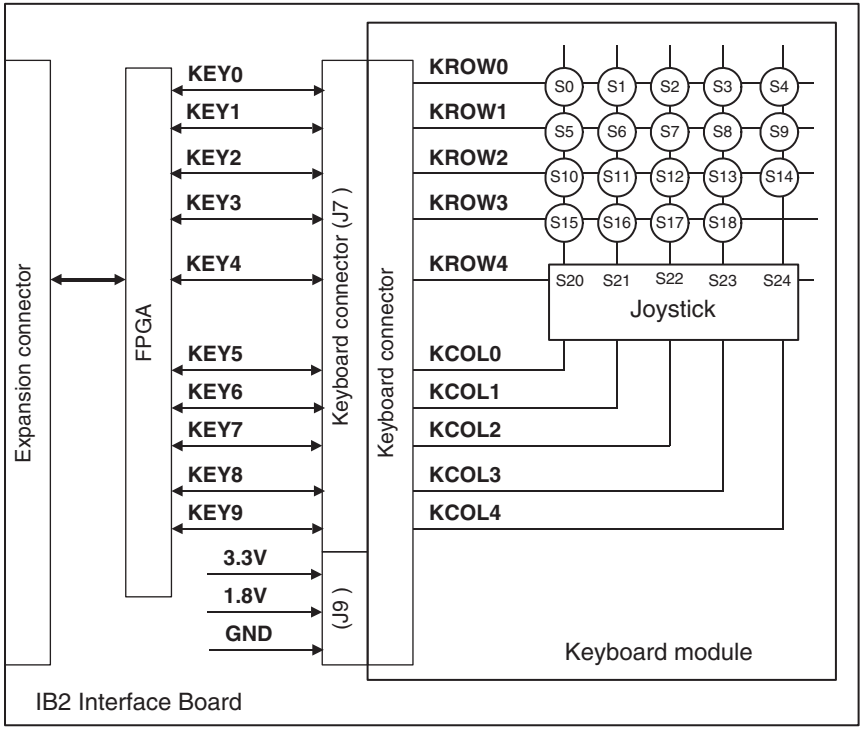

Figure D-7 Keyboard block diagram

Table D-13 Keyboard module connector

Pin	Signal
1	**KROW0**
2	**KROW1**
3	**KROW2**
4	**KROW3**
5	**KROW4**
6	**KCOL0**
7	**KCOL1**

Table D-13 Keyboard module connector (continued)

Pin	Signal
8	**KCOL2**
9	**KCOL3**
10	**KCOL4**
11	-
12	-
13	**GND**

D.4.5 Prototyping area

The PCB has prototyping holes that can be used for testing GSM modems or adding a Bluetooth interface. A Bluetooth module is not supplied as part of the product. The signals on the pads are shown in Table D-14 on page D-19.

Caution

The UART signals on the prototyping pins are at logic levels. Do not connect RS-232 level signals to the pins.

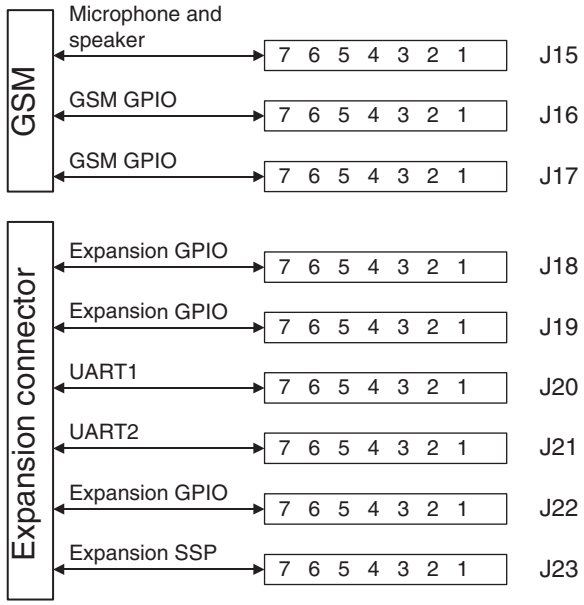

Figure D-8 Prototyping area block diagram

Table D-14 Prototyping area signals

Connector (row)	Pin (column)	Signal	Description
J15	1	**VMIC**	Microphone supply voltage
J15	2	**INT_MIC+**	Microphone input
J15	3	**INT_MIC-**	Microphone input
J15	4	**GND**	Power and signal ground
J15	5	**INT_SPK+**	Speaker output
J15	6	**INT_SPK-**	Speaker output
J15	1	**GND**	Power and signal ground
J16	1	**GSM_GPIO6**	GPIO from GSM module

Table D-14 Prototyping area signals (continued)

Connector (row)	Pin (column)	Signal	Description
J16	2	**GSM_GPIO7**	GPIO from GSM module
J16	3	**GSM_GPIO8**	GPIO from GSM module
J16	4	**GSM_ADC1**	ADC input to GSM module
J16	5	**GSM_ADC2**	ADC input to GSM module
J16	6	**GSM_DAC**	DAC output from GSM module
J16	7	**GND**	Power and signal ground
J17	1	**GSM_GPIO1**	GPIO from GSM module
J17	2	**GSM_GPIO2**	GPIO from GSM module
J17	3	**GSM_GPIO3**	GPIO from GSM module
J17	4	**GSM_GPIO4**	GPIO from GSM module
J17	5	**GSM_GPIO4**	GPIO from GSM module
J17	6	**GSM_GPIO5**	GPIO from GSM module
J17	7	**GND**	Power and signal ground
J18	1	**GND**	Power and signal ground
J18	2	**GPIO4**	GPIO from expansion header
J18	3	**GPIO5**	GPIO from expansion header
J18	4	**GPIO6**	GPIO from expansion header
J18	5	**GPIO7**	GPIO from expansion header
J18	6	**GND**	Power and signal ground
J18	7	**GND**	Power and signal ground
J19	1	**GND**	Power and signal ground
J19	2	**BTRESET**	Reset signal from FPGA
J19	3	**GPIO4**	GPIO from expansion header
J19	4	**GPIO5**	GPIO from expansion header

Table D-14 Prototyping area signals (continued)

Connector (row)	Pin (column)	Signal	Description
J19	5	**GPIO6**	GPIO from expansion header
J19	6	**GPIO7**	GPIO from expansion header
J19	7	**GND**	Power and signal ground
J20	1	**UART1TXD**	Transmit data
J20	2	**UART1RXD**	Receive data
J20	3	**UART1CTS**	Handshake signal
J20	4	**UART1RTS**	Handshake signal
J20	5	**3V3**	3.3V supply
J20	6	**3V8**	3.8V supply (maximum load 1A)
J20	7	**5V**	5V supply
J21	1	**UART2TXD**	Transmit data
J21	2	**UART2RXD**	Receive data
J21	3	**UART2CTS**	Handshake signal
J21	4	**UART2RTS**	Handshake signal
J21	5	**3V3**	3.3V supply
J21	6	**3V8**	3.8V supply (maximum load 1A)
J21	7	**5V**	5V supply
J22	1	**GND**	Power and signal ground
J22	2	**GSM_TOGGLE**	Radio power control input to GSM module
J22	3	**GSM_GPIO0**	GPIO from expansion header
J22	4	**GSM_GPIO1**	GPIO from expansion header
J22	5	**GSM_GPIO2**	GPIO from expansion header
J22	6	**GSM_GPIO3**	GPIO from expansion header
J22	7	**GND**	Power and signal ground

Table D-14 Prototyping area signals (continued)

Connector (row)	Pin (column)	Signal	Description
J23	1	**GND**	Power and signal ground
J23	2	**SSPTXD**	SPI_MOSI signal on module
J23	3	**SSPRXD**	SPI_MISO signal on module
J23	4	**SSPCLKOUT**	SPI_CLK signal on module
J23	5	**SSPnCS**	SPI_CSB signal on module
J23	6	**GND**	Power and signal ground
J23	7	**GND**	Power and signal ground

D.4.6 GSM module

J4 on the bottom of the PCB enables the connection of a quad-band GSM module. The signals on the connector are shown in Table D-15 on page D-24.

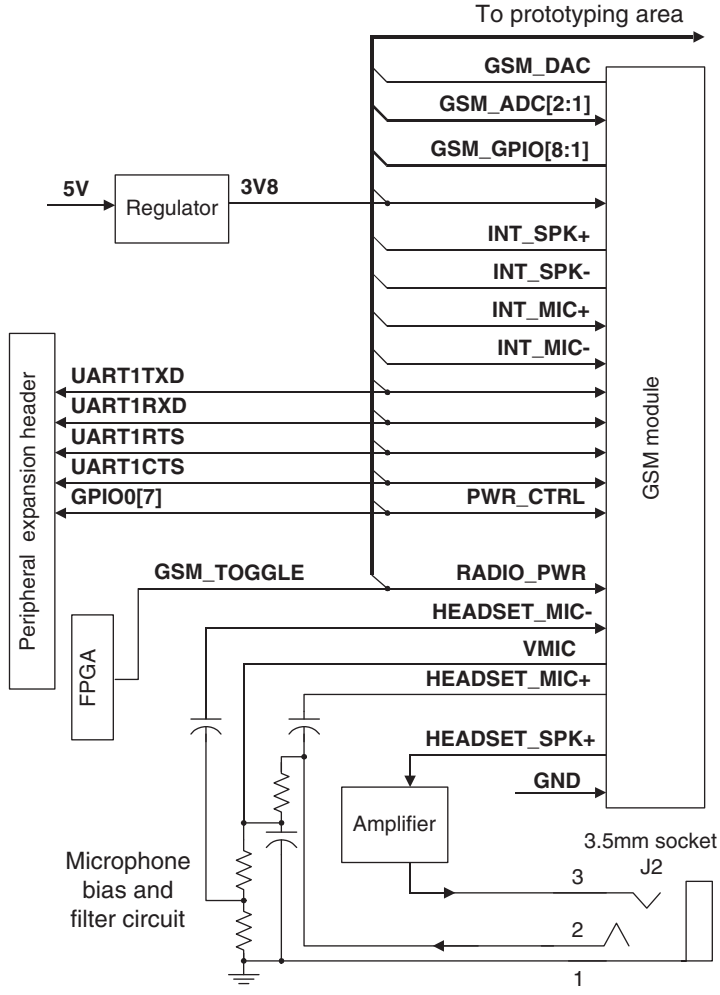

Figure D-9 GSM block diagram

Warning

The GSM module emits radio frequency radiation. Keep at least 20cm (7.9 inches) separation distance between the antenna and the human body.

Follow all safety instructions in this manual, any warning notices on the product, and any applicable legal requirements and safety regulations (see also the *Enfora Enabler IIG GSM/GPRS Radio Modem Integration Guide* GSM0108-01 at the Enfora website at `www.enfora.com`.)

The transmitter and antenna must not be collocated or operating in conjunction with any other antenna or transmitter. Failure to observe this warning could produce an RF exposure condition.

Enfora certifies that its Enfora Enabler II-G GSM Radio Module complies with the RF hazard requirements applicable to broadband PCS equipment operating under the authority of 47 CPR Part 24, Subpart E of the FCC Rules and Regulations. This certification is contingent upon installation, operation, and use of the Enfora Enabler II-G GSM Module in accordance with all instructions provided to the OEM and the end user. When installed and operated in a manner consistent with the instructions provided, the Enfora Enabler II-G meets the maximum permissible exposure (MPE) limits for general population/uncontrolled exposure as defined in Section 1.1310 of the FCC Rules and Regulations.

The antenna supplied with the Versatile/AB-IB2 has a gain of 0dBi. Modifications and/or additions to the Enfora Enabler II-G GSM transceiver, including use of antennas with higher than 3dBi gain, are prohibited. Mobile devices are limited to 2W EIRP under Part 24. The nominal RF output power of the GSM transceiver is 1.0W. The requirement for less than 2W EIRP can therefore be met if there are no modifications to the transceiver or antenna.

Table D-15 GSM connector signals

Pin	Signal	Description
1-6	**3.8V**	Power to GSM module
7-8	**GND**	Power and signal ground
8	**GND**	Power and signal ground
9	**INT_SPK-**	Earphone (to prototyping area)
9	**INT_SPK-**	Earphone (to prototyping area)
10	**3.8V**	Power to GSM module
11	**GND**	Power and signal ground
12	**3.8V**	Power to GSM module

Table D-15 GSM connector signals (continued)

Pin	Signal	Description
13	**INT_SPK+**	Earphone (to prototyping area)
14	**GSM_GPIO1**	GPIO signal from GSM module (use AT commands to read or write)
15	**GND**	Power and signal ground
16	**GND**	Power and signal ground
17	**VMIC**	Microphone bias voltage
18	**GSM_GPIO5**	GPIO signal from GSM module
19	**GND**	Power and signal ground
20	-	Not connected
21	**INT_MIC-**	Microphone (to prototyping area)
22	**GSM_TOGGLE**	On/off control
23	**GND**	Power and signal ground
24	**PWR_CTRL**	Power control, set HIGH to enable power to GSM
25	**INT_MIC+**	Microphone (to prototyping area)
26	-	Not connected
27	**GSM_GPIO3**	GSM DAC output (to prototyping area)
28	**GSM_GPIO2**	GPIO signal from GSM module
29	-	Not connected
30	**GSM_GPIO4**	GPIO signal from GSM module
31	-	Not connected
32	**GSM_GPIO6**	GPIO signal from GSM module
33	**GND**	Power and signal ground
34	**GSM_GPIO7**	GPIO signal from GSM module
35	**GSM_DAC**	GSM DAC output (to prototyping area)
36	**3.8V**	Power to GSM module
37	-	Not connected

Table D-15 GSM connector signals (continued)

Pin	Signal	Description
38	**HEADSET_SPK+**	GSM output to amplifier and earphone
39	**GSM_GPIO8**	GSM GPIO output (to prototyping area)
40	**HEADSET_MIC-**	GSM input from microphone
41	**GND**	Power and signal ground
42	**HEADSET_MIC+**	GSM input from microphone
43	**UART1RXD**	Data from GSM module
44	**GND**	Power and signal ground
45	**GSM_DCD**	Connected to GPIO 6 output from expansion header (also connected to prototyping area)
46	**GSM_ADC2**	GSM ADC input (to prototyping area)
47	**GSM_DCD**	Connected to GPIO 5 output from expansion header (also connected to prototyping area)
48	-	Not connected
49	**GSM_RI**	Connected to GPIO 4 output from expansion header (also connected to prototyping area). This signal can be used to trigger an interrupt when the GSM module receives a call.
50	**GND**	Power and signal ground
51	**UART1TXD**	Data to GSM module
52	-	Not connected
53	**nUART1RTS**	Handshake signal to GSM module
54	-	Not connected
55	**nUART1CTS**	Handshake signal to GSM module
56	**GSM_ADC1**	GSM ADC input (to prototyping area)
57	**GSM_DTR**	GSM handshake (connected to ground)
58	-	Not connected
59	**GND**	Power and signal ground
60	**GND**	Power and signal ground

Appendix E
LCD Kits

This appendix describes the optional external color LCD kits that can be used with the Versatile/AB-IB1. It contains the following sections:

E.1 About the CLCD display and adaptor board

The CLCD interface board provides multiple sockets for different types of CLCD displays and touchscreens. It connects to the AB-IB1 by a single cable.

The design of the interface board enables you to choose the CLCD display that is appropriate for your application. The CLCD displays and touchscreens supported by the AB-IB1 board are:

- Sanyo 3.8 inch QVGA color TFT with touchscreen and fluorescent backlight
- Sharp 8.4 inch VGA color TFT with touchscreen and fluorescent backlight.

Note

The QCIF 2.2 inch display that is available for the Versatile/PB926EJ-S is not supported on the Versatile/AB926EJ-S.

See also Appendix D *Versatile/AB-IB2 Interface Board* for details on the 2.5 inch display that is part of the Versatile/AB-IB2 board.

Six pushbutton switches are mounted on the interface board below the 3.8 inch display. The state of the switches can be read from the touchscreen controller interface.

The touchscreen interface on the CLCD interface board is described in *Touchscreen controller interface* on page E-11. The selftest program supplied on the CD reads the position of a pen on the touchscreen and displays it on the CLCD or VGA display connected to the board.

The 3.8 inch CLCD display and the adaptor board is mounted in a small enclosure as shown in Figure E-1 on page E-3.

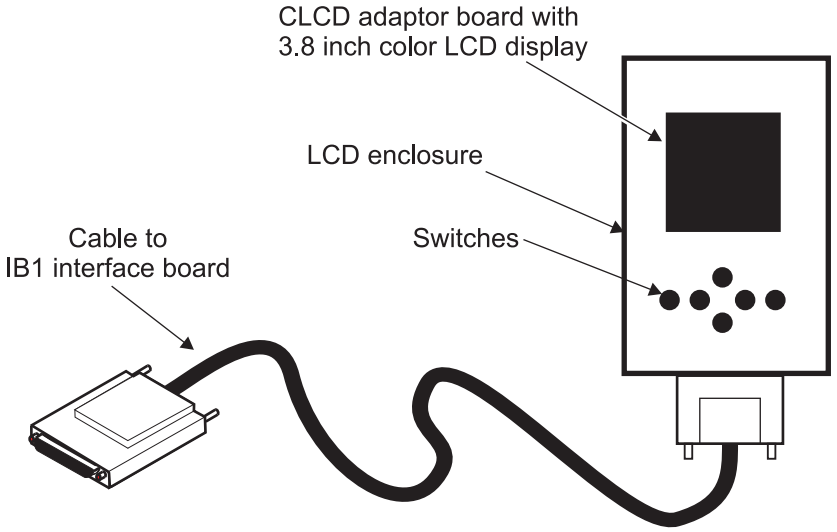

Figure E-1 Small CLCD enclosure

Note

The 2.2 inch display kit is not supported for the AB-IB1. You can, however, use the AB-IB2 interface board that has a 2.5 inch display fitted as standard.

The 8.4 inch display is mounted into a large enclosure that has two connectors: one for a keypad and one for the AB-IB1. (See Figure E-2 on page E-4.)

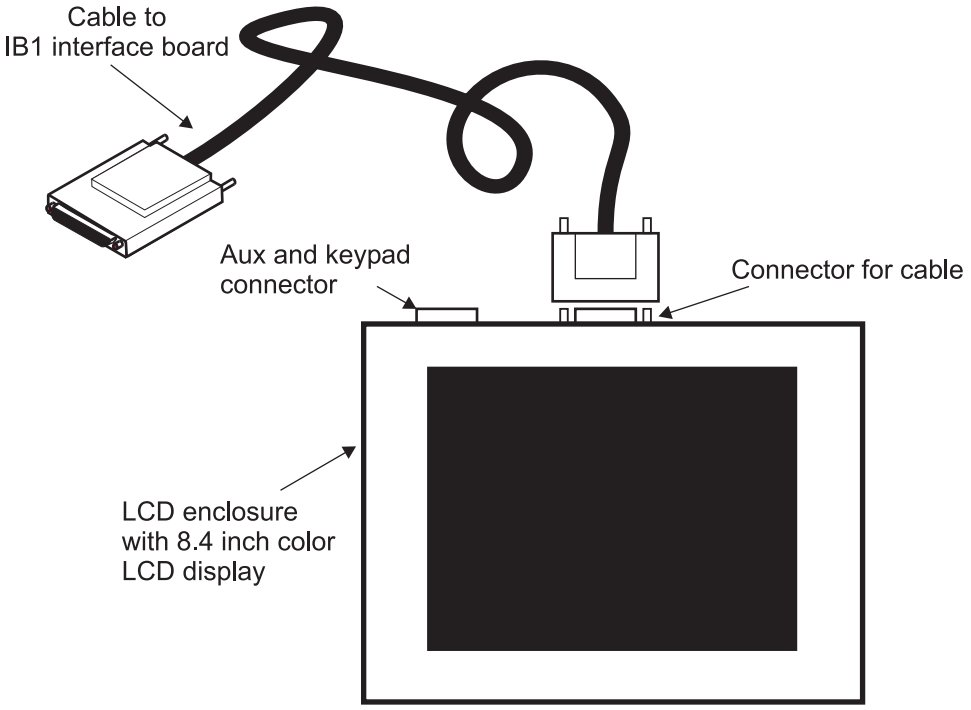

Figure E-2 Large CLCD enclosure

The 3.8 inch CLCD display is mounted on the top side of the adaptor board as shown in Figure E-3 on page E-5.

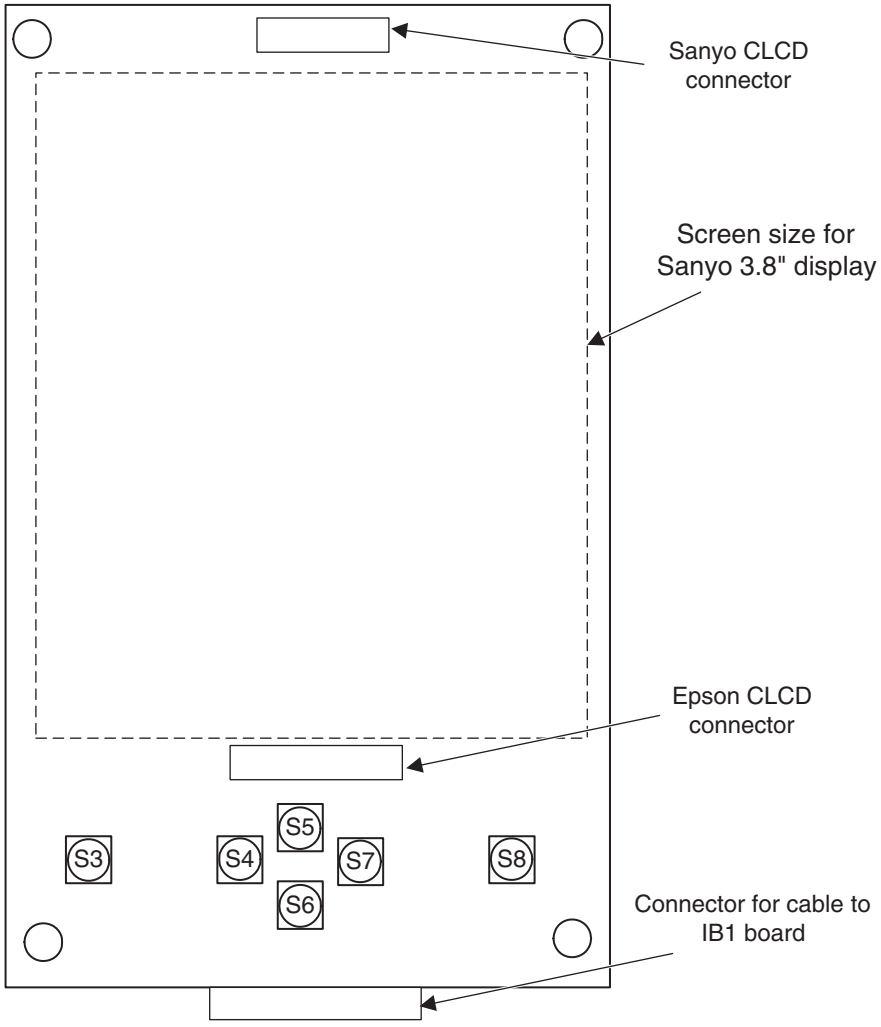

Figure E-3 3.8 display mounted directly onto top of adaptor board.

E.2 Installing the CLCD display

To install the CLCD display:

1. Connect one end of the CLCD expansion cable to the CLCD adaptor board.
2. Connect the other end of the cable to the Versatile/AB-IB1 CLCD expansion connector.
3. If required, program the CLCD control registers SYS_CLCD to sequence the power to the LCD display and specify the bit format. See the *CLCD Control Register, SYS_CLCD* on page 4-28.

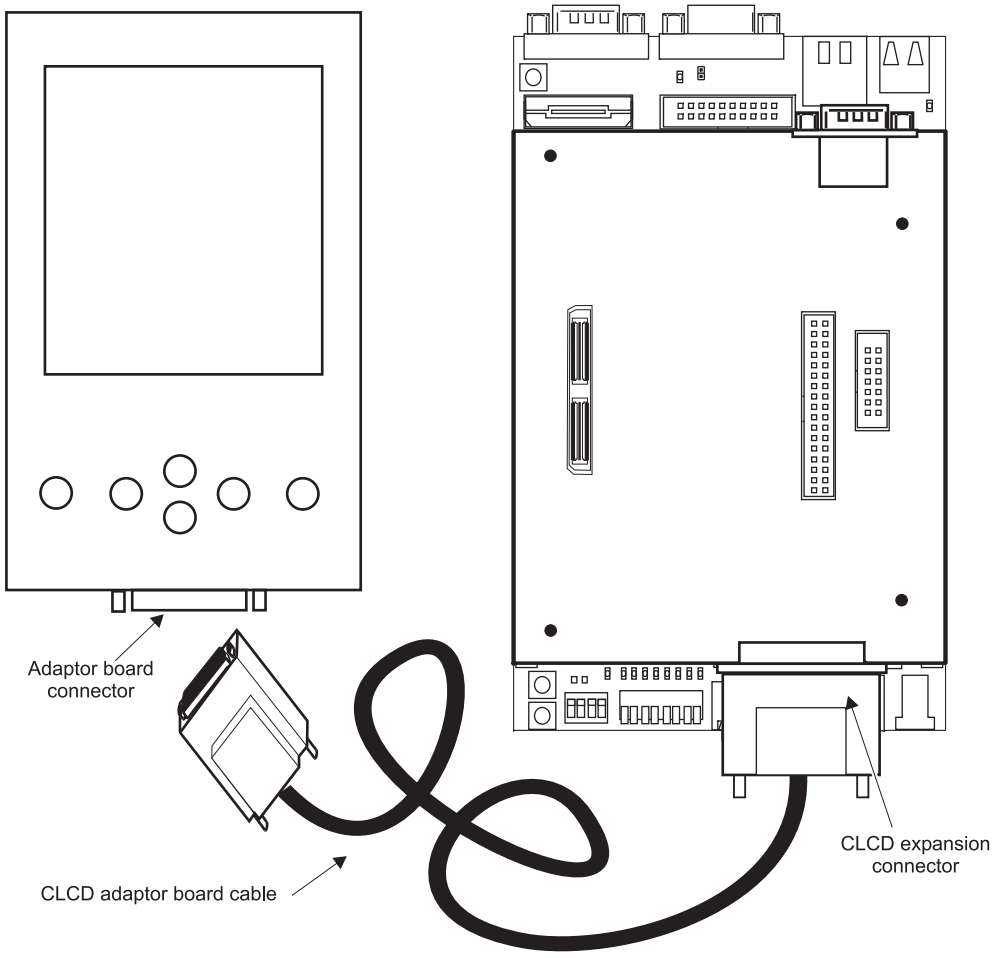

Figure E-4 CLCD adaptor board connection

E.2.1 Configuration

The CLCD adaptor board contains factory-installed links that identify the type of display. The display matching the identification links settings are listed in Table E-1. The value of the bits **CLCDID[4:0]** in the SYS_CLCD register can be read from software to determine the display in use with the board. The timing values for the different displays are listed in Table E-2.

Table E-1 Displays available with adaptor board

LCD_ID[4:0]	Manufacturer	Backlight inverter	Touchscreen	Display
b00000	Sanyo TM38QV67A02A	TDK CXA-0341	Part of display	3.8 inch landscape QVGA Color TFT
b00001	Sharp LQ084V1DG21	TDK CXA-L0612VJL	DynaPro/3M 95643	8.4 inch VGA color TFT
b00010-b01110	Reserved	-	-	-
b01111	No display fitted	-	-	-

Table E-2 Values for different TFT resolutions

LCD_ID[4:0]	Display resolution	CLCDCLK frequency and SYS_OSC1 register value	CLCD_TIM0 register at 0x10120000	CLCD_TIM1 register 0x10120004	CLCD_TIM2 register at 0x10120008
b00000	320x240 Quad VGA	10MHz, 0x2C2A	0x0505054C	0x050514EF	0x05EF1800
b00001	640x480 VGA	25MHz, 0x2C77	0x3F1F3F9C	0x090B616F	0x067F1800

E.2.2 LCD power control

The LCD adaptor board accommodates a wide range of LCDs. Displays can require from 1 to 4 power supplies that can either be turned on/off simultaneously or need to be switched on/off in a certain order. System control register SYS_CLCD and the CLCD PrimeCell system register control power switching. The voltage supplies on the board are:

Vin This is permanently on and is not switched. This provides power to the board (nominal 12V) for the backlight converter.

1V8 This supply is permanently on. It is generated from 5V.

SWITCHED_FIXED

The supply is generated from the 1.8V, 3.3V or 5V supply. It can be enabled by **PWR3V5VSWITCH** in SYS_CLCD or permanently enabled by link 13.

SWITCHED_CLPWR

This supply is generated from 5V. It can be enabled by the **CLPOWER** signal in the CLCD PrimeCell control register or permanently enabled by link 15.

SWITCHED_VDD_NEG

This –5V to –28V supply is generated from 5V. It can be enabled by **VDDNEGSWITCH** in SYS_CLCD or permanently enabled by link 14.

SWITCHED_VDD_POS

This 11V to 28V supply is generated from 5V. It can be controlled by the touchscreen D/A converter or manually with a pot. It can be enabled by **VDDPOSSWITCH** in SYS_CLCD or permanently enabled by link 11. This supply is used to generate the STN bias voltage.

LCD_IO_VDD and Buffer I/O voltage

This is the voltage to the interface logic on the adaptor board and the display. Link 16 selects the adaptor board interface level as **CLPWR** or **FIXED**. Link 3 selects the display interface level as **SWITCHED_FIXED** or **SWITCHED_CLPWR**.

Caution

Link 3 and link 16 must be set to use the same power source.

INV_IO This is the voltage to the interface logic on the prototype board. Link 2 selects the level as 5V or 3.3V.

Note

The I/O signals to the CLCD adaptor board pass through tri-state buffers. The buffers must be powered from the same IO voltage as that required by the CLCD. This enables the translation of the IO signals from the 3V3 signal levels present on the AB-IB1. The buffers are enabled by **LCDIOON** in SYS_CLCD.

Figure E-5 on page E-10 shows the block diagram of the adaptor board power-control circuitry.

Table E-3 shows the power configuration for the three displays. For additional information on configuring the CLCD displays, see the selftest code provided on the CD.

Table E-3 Power configuration

Voltage control	Sanyo 3.8"	Sharp 8.4"
Buffer IO	CLPOWER	CLPOWER
SWITCHED_VDD_POS	15V	Software control
SWITCHED_VDD_NEG	–10V	–10V
SWITCHED_CLPOWER	3.3V	3.3V
SWITCHED_FIXED	5V	5V
INV_IO	3.3V	5V
Buffer enabled (software control)	Always on	Set from CLPOWER register in ARM926PXP development chip

Caution

The links for power control are set during manufacture. Do not modify the links unless you are producing a new custom display board.

Use connector J4 to supply power to an inverter for a backlight. The backlight pins **VIN** are provide a nominal 12V supply. The backlight inverter must consume less than 5W. The I/O voltage level **INV_IO** is also present on J4. **INV_IO** can be link selected to be 5V or 3.3V.

In addition to voltage and ground pins, the connector also supplies the brightness adjustment voltage (0 to **INV_IO** voltage). The brightness is adjusted by a variable resistor, VR4, located near J4.

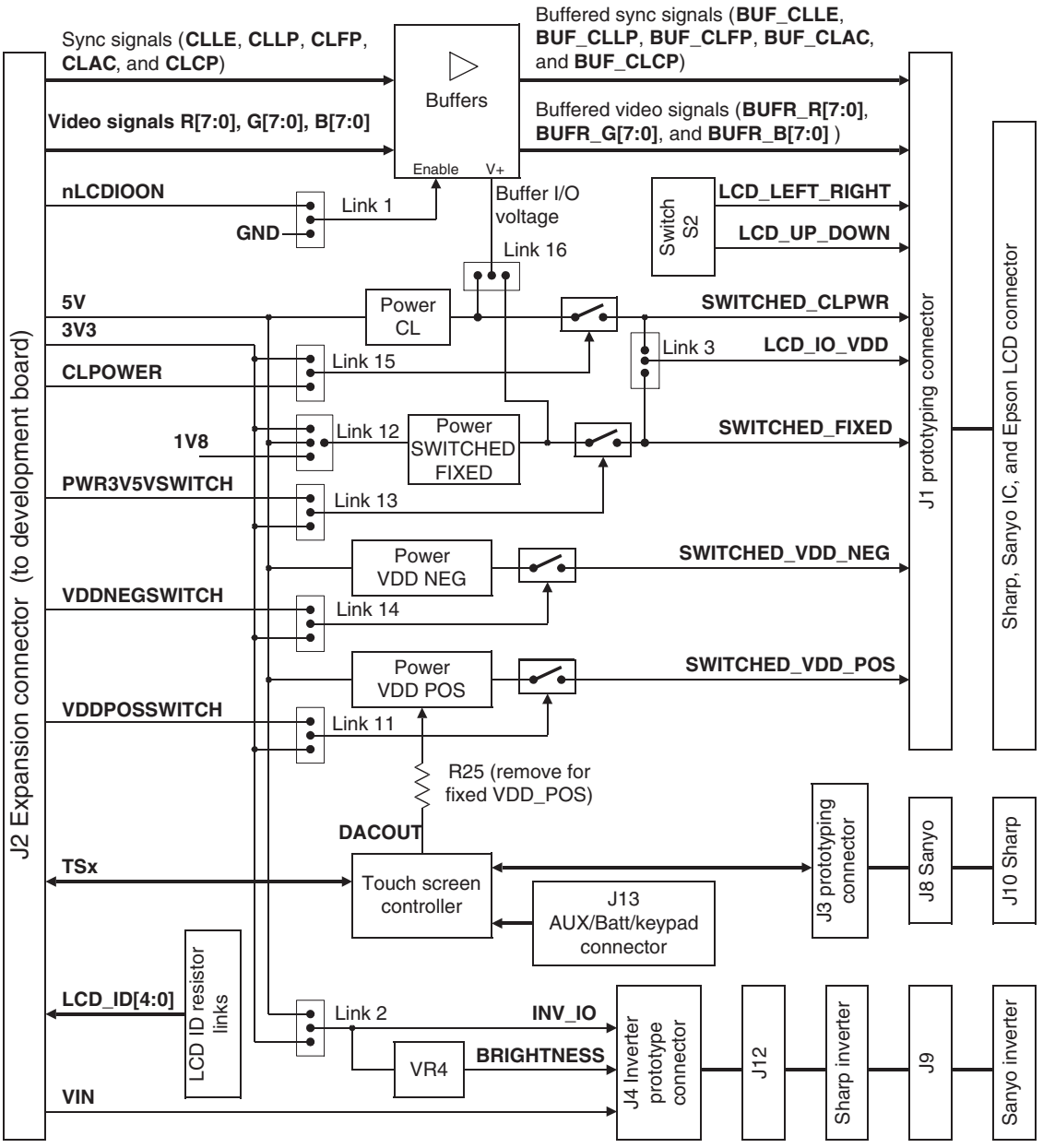

Figure E-5 CLCD buffer and power supply control links

E.3 Touchscreen controller interface

The touchscreen interface is designed to connect to a four-wire resistive touchscreen. It is driven by the TouchScreen controller TSC2200 and described in:

- *Touchscreen interface architecture*
- *Touchscreen controller programmer's interface* on page E-13.

The `Selftest` program supplied on the CD demonstrates how to communicate with the touchscreen controller. The program uses the interface code to plot the touchscreen X and Y coordinates on the LCD or VGA screen.

Connectors J8 and J10 are used for the standard touchscreens provided with the CLCD assembly. Prototyping connector J3 enables the use of other resistive touchscreens, see *Touchscreen prototyping connector* on page E-17.

E.3.1 Touchscreen interface architecture

Figure E-6 on page E-12 shows the touchscreen interface. Table E-4 lists the touchscreen control signals. The signals to the touchscreen are routed to connector J13.

Table E-4 Touchscreen host interface signal assignment

Signal name	Description
TSMOSI	Serial data input to controller
TS_nSS	Chip select
TSSCLK	Clock input
TSMISO	Data output
TSnDAV	Data available
TSnPENIRQ	Pen down interrupt
TSnKPADIRQ	Keypad interrupt
VBAT[2:1] and **AUX[2:1]**	External voltage to analog to digital converter in touchscreen controller. These are reserved for expansion for external devices connected to the AD and keypad connector J13.
R[4:1] and **C[4:1]**	Row and column scan signals for a keyboard. The interface board switches S3 to S8 currently use eight positions on the scan matrix, but additional switches can be fitted using the AD and keypad connector J13.

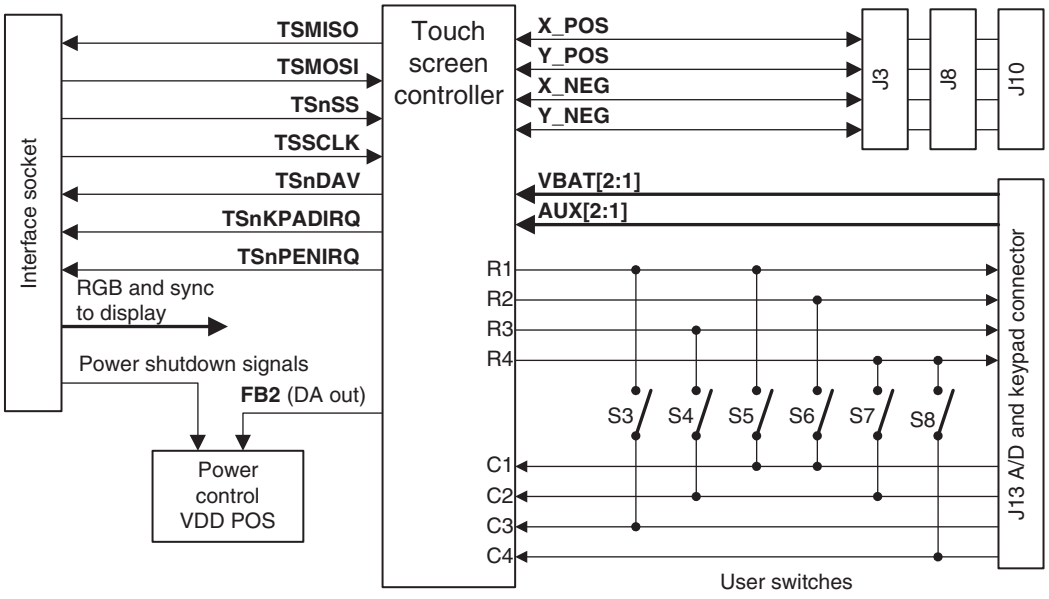

Figure E-6 Touchscreen and keypad interface

The connection between the resistive elements of the touchscreen and J3, J8, or J10 is shown in Figure E-7. When the pen is down, the two resistive elements touch and form a four-resistor network. Measuring the voltages at the two dividers indicates the X and Y positions.

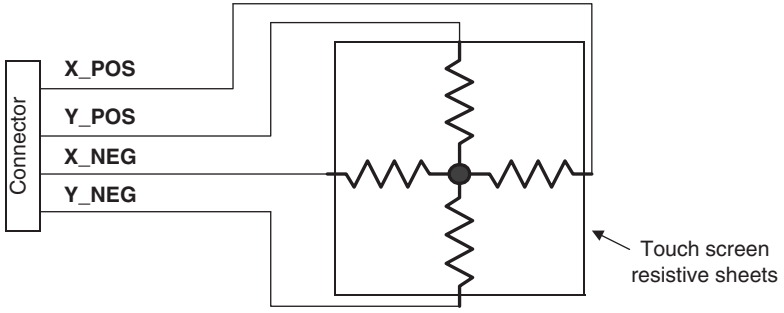

Figure E-7 Touchscreen resistive elements

E.3.2 Touchscreen controller programmer's interface

The LCD *Touch Screen Controller Interface* (TSCI) is based on a TSC2200 PDA analogue interface circuit. Use the ARM926PXP development chip SSP interface to configure and read the touch screen. For information on the touch screen registers, see the TSC2200 data sheet.

The TSC2200 also incorporates a sixteen key keypad interface and two 12bit analogue inputs that are available through the LCD expansion header J13. With the 3.8 inch Sanyo and 2.2 inch Epson build options, six keypad push buttons are mounted on the LCD board.

SSP and TSCI Configuration

The SSP interface is controlled by the SSP PrimeCell and the SSP TSCI chip select is enabled through SYS_CLCD **TSnSS** signal. Configuration of the SSP to TSCI interface requires the data format, phase, size, and clock to be set correctly. Example configuration code is given in the selftest (TSCI) software on the CD, a code fragment from this is shown in Example E-1.

Example E-1 SSP to TSCI interface setup

```
// Set serial clock rate (/3), Phase (SPH), Format (MOT), data size (16bit)
*SSPCR0 = SSPCR0_SCR_DFLT | SSPCR0_SPH | SSPCR0_FRF_MOT | SSPCR0_DSS_16;
// Clock prescale register (/8), with SCR gives 0.78MHz SCLK: 24MHz / 8*(1+3)
*SSPCPSR = SSPCPSR_DFLT;
// Enable serial port operation
*SSPCR1 = SSPCR1_SSE;
```

The TSC2200 TSCI controller registers must be configured through the SSP interface to enable correct touch screen operation.

After the TSCI is configured, conversion of touch screen X/Y values is fully automated by the TSCI controller and the application code simply reads the converted values. Use either the pen down flag in the touch screen controller interface or SIC interrupt 8 to detect the current pen state. The pseudo code in Example E-2 on page E-14 shows the sequence for configuring and reading the TSCI interface.

Read and write functions are used in the selftest code to transfer data to and from the TSCI registers TSCI_RTSC and TSCI_WTSC. The selftest example configures the TSCI for 12bit operation and 16 data averages with minimum precharge and sense times. This gives high accuracy and fast reading of the current pen position.

Example E-2 Configuring and reading the TSCI interface

```
Configure the SSP interface
Configure the TSCI registers
Enable the touch screen pendown interrupt (on SIC)
  ...
On touch screen pendown interrupts
    ...  touch screen interrupt handler
    Enable the touch screen event timer (TIMER 1-4) for approx. 2mS intervals
  ...
On touch screen timer events
   ...   touch screen reading
   If (pendown flag (PSM) is cleared)
        Disable the touch screen event timer
        Clear and re-enable the touch screen interrupt
   Else
        Read the pen X/Y values
        Draw the pen position on the screen
```

Note

The selftest example provided on the CD uses a simple polled system to determine pen down and timer events.

The pseudo code in Example E-2 is recommended for OS ports as they typically require interrupt-driven device drivers.

E.4 Connectors

This section describes the connectors present on the CLCD adaptor board.

E.4.1 Interface connector

The signals on the CLCD interface connector J2 are shown in Table E-5.

Table E-5 CLCD interface connector J2

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	**B0**	2	**B2**	35	**B1**	36	**B3**
3	**B4**	4	**B6**	37	**B5**	38	**B7**
5	**G0**	6	**G2**	39	**G1**	40	**G3**
7	**G4**	8	**G6**	41	**G5**	42	**G7**
9	**R0**	10	**R2**	43	**R1**	44	**R3**
11	**R4**	12	**R6**	45	**R5**	46	**R7**
13	**CLLE**	14	**CLAC**	47	**GND**	48	**GND**
15	**CLCP**	16	**CLLP**	49	**GND**	50	**GND**
17	**CLFP**	18	**TSnKPADIRQ**	51	**GND**	52	**GND**
19	**TSnPENIRQ**	20	**TSnDAV**	53	**GND**	54	**LCDID0**
21	**TSSCLK**	22	**TSnSS**	55	**LCDID1**	56	**LCDID2**
23	**TSMISO**	24	**TSMOSI**	57	**LCDID3**	58	**LCDID4**
25	**LCDXWR**	26	**LCDSD0**	59	**GND**	60	**GND**
27	**LCDXRD**	28	**LCDXCS**	61	**GND**	62	**3V3**
29	**LCDDATAnCOMM**	30	**LCDSD0DIR**	63	**3V3**	64	**5V**
31	**CLPOWER**	32	**nLCDIOON**	65	**5V**	66	**VIN**
33	**PWR3V5VSWITCH**	34	**VDDPOSSWITCH**	67	**VIN**	68	**VDDNEGSWITCH**

E.4.2 LCD prototyping connector

The signals on the LCD prototyping connector J1 are shown in Table E-6.

Table E-6 LCD prototyping connector J1

Signal	Pin	Pin	Signal
BUF_CLLP	1	2	**BUF_G2**
GND	3	4	**BUF_G3**
CLCP0	5	6	**GND**
GND	7	8	**BUF_G4**
BUF_CLFP	9	10	**BUF_G5**
GND	11	12	**GND**
BUF_CLAC	13	14	**BUF_G6**
GND	15	16	**BUF_G7**
BUF_CLLE	17	18	**GND**
GND	19	20	**BUF_R0**
BUF_B0	21	22	**BUF_R1**
BUF_B1	23	24	**GND**
GND	25	26	**BUF_R2**
BUF_B2	27	28	**BUF_R3**
BUF_B3	29	30	**GND**
GND	31	32	**BUF_R4**
BUF_B4	33	34	**BUF_R5**
BUF_B5	35	36	**GND**
GND	37	38	**BUF_R6**
BUF_B6	39	40	**BUF_R7**
BUF_B7	41	42	**SWITCHED_FIXED**
GND	43	44	**LCD LEFT_RIGHT**

Table E-6 LCD prototyping connector J1 (continued)

Signal	Pin	Pin	Signal
BUF_G0	45	46	**LCD UP_DOWN**
BUF_G1	47	48	**SWITCHED_VDD_POS**
SWITCHED_CLPWR	49	50	**SWITCHED_VDD_NEG**

E.4.3 Touchscreen prototyping connector

The signals on the touchscreen prototyping connector J3 are shown in Table E-7.

Table E-7 Touchscreen prototyping connector J3

Signal	Pin	Pin	Signal
GND	1	2	**GND**
X_POS	3	4	**GND**
Y_NEG	5	6	**GND**
X_NEG	7	8	**GND**
Y_POS	9	10	**GND**

E.4.4 Inverter prototyping connector

The signals on the inverter prototyping connector J4 are shown in Table E-8.

Table E-8 Inverter prototyping connector J4

Signal	Pin	Pin	Signal
VIN	1	2	**VIN**
VIN	3	4	**VIN**
GND	5	6	**GND**
BRIGHTNESS	7	8	**GND**
GND	9	10	**INV_IO**

E.4.5 A/D and keypad connector

The signals on the connector J13 are shown in Table E-8 on page E-17.

This connector enables the connection of an external keypad (**R[4:1]** are the keypad row scan output signals and **C[4:1]** are the column detect input signals). There are also connections to the analog to digital converter inputs on the CLCD adaptor board (**AUX[2:1]** and **VBAT[2:1]**).

Table E-9 A/D and keypad J13

Signal	Pin	Pin	Signal
3V3	1	20	**3V3**
AUX1	2	19	**GND**
AUX2	3	18	**GND**
VBAT1	4	17	**GND**
VBAT2	5	16	**GND**
R1	6	15	**C1**
R2	7	14	**C2**
R3	8	13	**C3**
R4	9	12	**C4**
GND	10	11	**GND**

E.5 Mechanical layout

Shows the board layout and location of the CLCD, switches, and mounting holes.

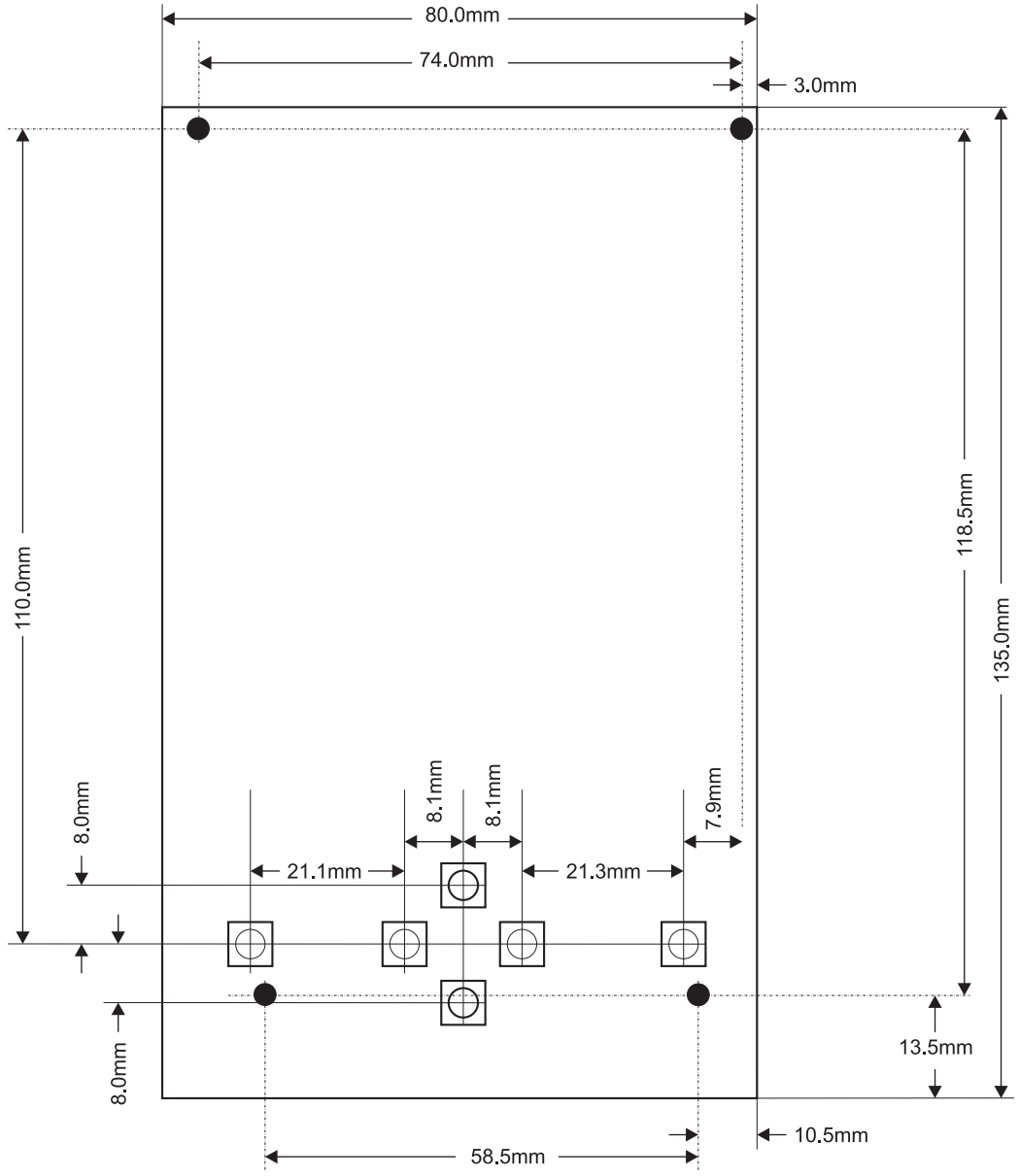

Figure E-8 CLCD adaptor board mechanical layout

Appendix F
Static Memory Expansion Board

This appendix describes the static memory expansion board for the Versatile/AB926EJ-S. It contains the following sections:

F.1 About memory expansion

You can fit a static expansion board to the Versatile/AB-IB1 or Versatile/AB-IB2 interface boards for the Versatile/AB926EJ-S. The block diagram for a typical memory board is shown in Figure F-1.

Note

There are five chip select signals available on the static expansion board. Each of these can select 64MB of SRAM. If you are using the IB2 interface board, one of the chip select signals is used by the FPGA and the maximum memory supported is therefore four times 64MB or 256MB.

Figure F-1 is an example only. Different expansion boards might have different features. For example, the links selecting which chip select to use might be omitted.See the documentation provided with your memory board for details on signals and link options.

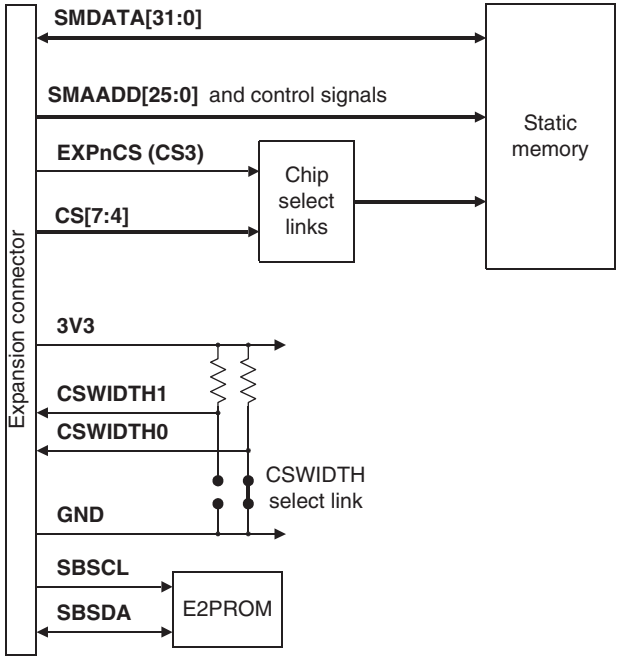

Figure F-1 Static memory board block diagram

F.1.1 Operation without expansion memory

You can operate the Versatile/AB926EJ-S without a memory expansion board because it has 2MB of SSRAM, 128MB SDRAM, 64MB NOR flash, and 64MB NAND flash permanently fitted.

You can use the expansion boards, however, for applications that require more static memory or to prototype or develop memory devices that are not available on the Versatile/AB926EJ-S.

F.1.2 Memory board configuration

The E2PROM on the memory board can be read from the Versatile/AB926EJ-S to identify the type of memory on the board and how it is configured. This information can be used by the application or operating system to initialize the memory space.

Memory width selection on the static memory board

The memory width on the memory board is encoded into the **CSWIDTH[1:0]** signals as shown in Table F-1.

Table F-1 Memory width encoding

CSWIDTH[1:0]	Width
00	8 bit
01	16 bit
10	32 bit (default)
11	No memory present

Note

Additional configuration information is present in the E2PROM on the expansion board, see *EEPROM contents* on page F-5.

F.2 Fitting a memory board

To install a memory expansion board:

1. Ensure that the Versatile/AB926EJ-S is powered down.
2. Align the memory board with the connectors on the interface board as shown in Figure F-2.
3. Press the memory board into the connector.

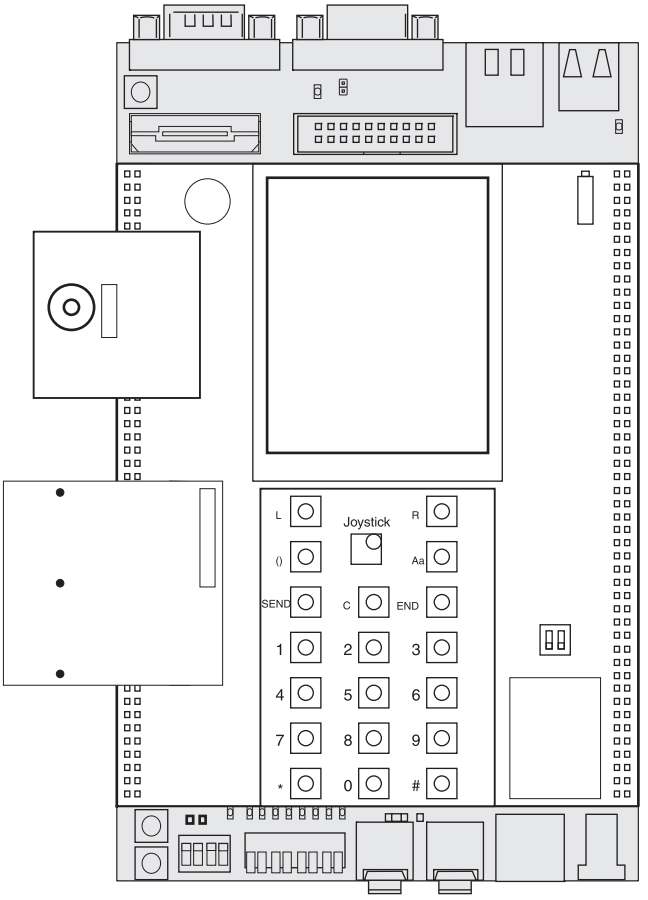

Figure F-2 Memory board installation

Note

The memory connector orientation for the AB-IB1 is reversed relative to the AB-IB2.

F.3 EEPROM contents

There are two devices on the Versatile/AB926EJ-S serial bus:

- Static Memory Expansion EEPROM at `0xA2` for write, `0xA3` for read
- Real Time Clock (Time of Year) at `0xD0` for write, `0xD1` for read

The Static Memory Expansion EEPROM is 256 bytes in size and contains 5 chip select information blocks, a manufacturer string and a checksum.

Each chip select information block contains details about the memory devices accessed with the corresponding chip select signal. The organization of a chip select information block is listed in table Table F-2.

Table F-2 Chip Select information block

Function	Address offset	Value

Memory Type	`0x0`	`0x0= Reserved`
		`0x1= Static Disk On Chip`
		`0x2= Static NOR flash`
		`0x3= Static SRAM`
		`0x4-0xFE = Reserved (used by dynamic memory expansion modules)`
		`0xFF = Not fitted.`

Memory Width	`0x01`	Bits [3:0] indicate the chip-select width: `0= byte wide1= 16-bit wide2= 32-bit wide3= Reserved.` Bits [7:4] indicate the device memory width: `0= byte wide1= 16-bit wide2= 32-bit wide3= Reserved.`
Access time	`0x02`	Two bytes containing the access time (tACC) decoded as a binary number of 100ps. Location 2 contains the LSB and location 3 contains the MSB. For example, a flash device with 120ns access is 1200 * 0.1ns. The decimal value is 1200 and the hex value is `0x04B0`, therefore location 2 contains `0xb0` and location 3 contains `0x04`.
Size	`0x04`	Four bytes containing the size of the memory in bytes location 4 is the LSB and location 7 is MSB.
Reserved	`0x08-0x0F`	Eight bytes reserved for future expansion
Device string	`0x10-0x2F`	Null terminated string of up to 32 characters (31 characters + null character) containing the manufacturer name and part number.

The base address of the information block is determined by the device chip select used.

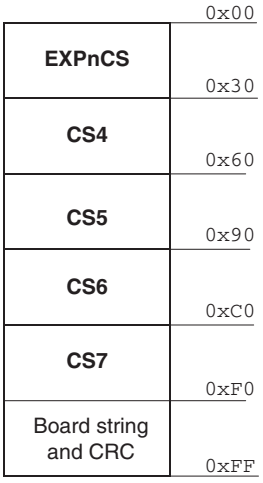

Figure F-3 Chip select information block

The contents of a typical static memory expansion EEPROM with devices on **EXPnCS** and **CS4** is listed in Table F-3. Unused chip select blocks are filled with `0xFF`.

Table F-3 Example contents of a static memory expansion EEPROM

Address offset	Contents	Contents
`0x00`	EXPnCS memory type	`0x02` = Static NOR flash
`0x01`	EXPnCS memory width	`0x12` - 32 bit chip select width, 16-bit device memory width
`0x02`	EXPnCS access time in 0.1ns (LSB)	`0xb0` - LSB (of 1200 which 1200 * 0.1ns = 120ns access time)
`0x03`	EXPnCS access time in 0.1ns (MSB)	`0x04` - MSB (of 1200 which 1200 * 0.1ns = 120ns access time)
`0x04`	EXPnCS memory size in bytes (LSB)	`0x00`
`0x05`	EXPnCS memory size in bytes	`0x00`
`0x06`	EXPnCS memory size in bytes	`0x00`
`0x07`	EXPnCS memory size in bytes (MSB)	`0x04` (`0x04000000` Bytes = 64MBytes)
`0x8-0xF`	Reserved	`0xFF`

Table F-3 Example contents of a static memory expansion EEPROM (continued)

Address offset	Contents	Contents
0x10-0x2F	EXPnCS memory device string	"Intel GE28F256K3C120" + null character
0x30	CS4 memory type	0x01 = Static SRAM
0x31	CS4 memory width	0x02 - 32 bit wide
0x32	CS4 access time in 0.1ps (LSB)	0x26 - LSB (of 550 which 550 * 0.1ns = 55ns access time)
0x33	CS4 access time in 0.1ps (MSB)	0x02 - MSB (of 550 which 550 * 0.1ns = 55ns access time)
0x34	CS4 memory size in bytes (LSB)	0x00
0x35	CS4 memory size in bytes	0x00
0x36	CS4 memory size in bytes	0x00
0x37	CS4 memory size in bytes (MSB)	0x20 (0x00200000 Bytes = 2MBytes)
0x38-0x3F	Reserved	0xFF
0x40-0x5F	CS4 memory device string	"Samsung K6F8016U6A-F55" + null character
0x60	CS5 memory type	0xFF - not fitted
0x61	CS5 memory width	0xFF
0x62	CS5 access time in 0.1ps (LSB)	0xFF
0x63	CS5 access time in 0.1ps (MSB)	0xFF
0x64	CS5 memory size in bytes (LSB)	0xFF
0x65	CS5 memory size in bytes	0xFF
0x66	CS5 memory size in bytes	0xFF
0x67	CS5 memory size in bytes (MSB)	0xFF
0x68-0x6F	Reserved	0xFF
0x70-0x8F	CS5 memory device string	0xFF
0x90	CS6 memory type	0xFF - not fitted
0x91	CS6 memory width	0xFF
0x92	CS6 access time in 0.1ps (LSB)	0xFF

Table F-3 Example contents of a static memory expansion EEPROM (continued)

Address offset	Contents	Contents
0x93	CS6 access time in 0.1ps (MSB)	0xFF
0x94	CS6 memory size in bytes (LSB)	0xFF
0x95	CS6 memory size in bytes	0xFF
0x96	CS6 memory size in bytes	0xFF
0x97	CS6 memory size in bytes (MSB)	0xFF
0x98-0x9F	Reserved	0xFF
0xA0-0xBF	CS6 memory device string	0xFF
0xC0	CS7 memory type	0xFF - not fitted
0xC1	CS7 memory width	0xFF
0xC2	CS7 access time in 0.1ps (LSB)	0xFF
0xC3	CS7 access time in 0.1ps (MSB)	0xFF
0xC4	CS7 memory size in bytes (LSB)	0xFF
0xC5	CS7 memory size in bytes	0xFF
0xC6	CS7 memory size in bytes	0xFF
0xC7	CS7 memory size in bytes (MSB)	0xFF
0xC8-0xCF	Reserved	0xFF
0xD0-0xEF	CS7 memory device string	0xFF
0xF0-0xFE	Board manufacturer string	"ARM HBI0124A"+ null character
0xFF	Checksum Byte	The LSB of the sum of bytes 0x00-0xFE

F.4 Connector pinout

The static memory board uses a 120-way Samtec QSH style connector as shown in Figure F-4. The AB-IB1 and AB-IB2 use the corresponding 120-way Samtec QTH connector on the interface board. The connector pinout for the static memory board is shown in Table F-4.

Note

The numbering of pins on the connector is for the connectors as viewed from below.

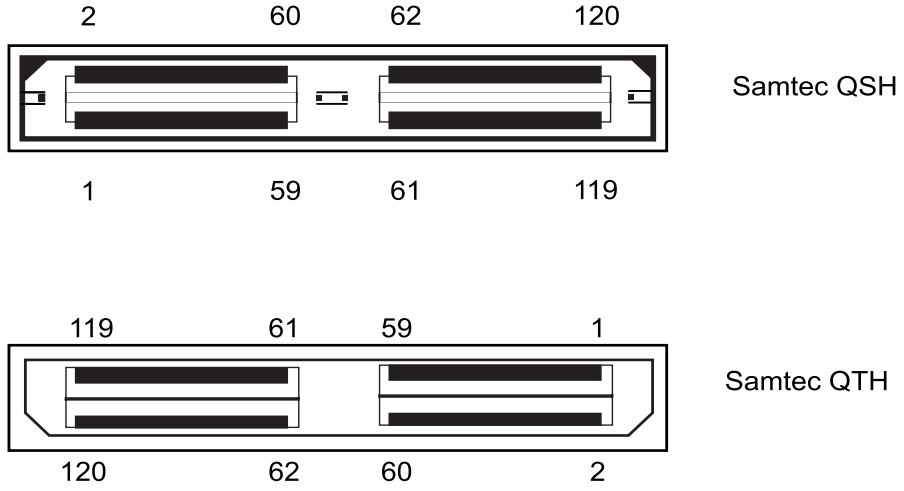

Figure F-4 Samtec connectors

Table F-4 Static memory connector signals

Pin No.	Signal	Pin No.	Signal
1	**DATA[0]**	2	**3V3**
3	**DATA[1]**	4	**3V3**
5	**DATA[2]**	6	**3V3**
7	**DATA[3]**	8	**3V3**
9	**DATA[4]**	10	**VDDIO**[a]
11	**DATA[5]**	12	**VDDIO**[a]

Table F-4 Static memory connector signals (continued)

Pin No.	Signal	Pin No.	Signal
13	**DATA[6]**	14	**VDDIO**[a]
15	**DATA[7]**	16	**VDDIO**[a]
17	**DATA[8]**	18	**1V8**
19	**DATA[9]**	20	**1V8**
21	**DATA[10]**	22	**1V8**
23	**DATA[11]**	24	**1V8**
25	**DATA[12]**	26	NC
27	**DATA[13]**	28	Reserved, do not drive
29	**DATA[14]**	30	Reserved, do not drive
31	**DATA[15]**	32	Reserved, do not drive
33	**DATA[16]**	34	**5V**
35	**DATA[17]**	36	**5V**
37	**DATA[18]**	38	**5V**
39	**DATA[19]**	40	**5V**
41	**DATA[20]**	42	Reserved, do not drive
43	**DATA[21]**	44	Reserved, do not drive
45	**DATA[22]**	46	Reserved, do not drive
47	**DATA[23]**	48	Reserved, do not drive
49	**DATA[24]**	50	Reserved, do not drive
51	**DATA[25]**	52	Reserved, do not drive
53	**DATA[26]**	54	Reserved, do not drive
55	**DATA[27]**	56	Reserved, do not drive
57	**DATA[28]**	58	Reserved, do not drive
59	**DATA[29]**	60	Reserved, do not drive

Table F-4 Static memory connector signals (continued)

Pin No.	Signal	Pin No.	Signal
61	**DATA[30]**	62	**SBSCL**, E2PROM serial interface clock (3.3V signal level)
63	**DATA[31]**	64	**SBSDA**, E2PROM serial interface data (3.3V signal level)
65	**ADDR[0]**	66	**nRESET**
67	**ADDR[1]**	68	**nBOARDPOR**, asserted on hardware power cycle
69	**ADDR[2]**	70	**nFLWP**, flash write protect. Drive HIGH to write to flash.
71	**ADDR[3]**	72	**nEARLYRESET**, Reset signal. Differs from **nRESET** in that it is not delayed by **nWAIT**.
73	**ADDR[4]**	74	**nWAIT**, Wait mode input from external memory controller. Pull HIGH if not used.
75	**ADDR[5]**	76	**nBURSTWAIT**, Synchronous burst wait input. This is used by the external device to delay a synchronous burst transfer if LOW. Pull to HIGH if not used.
77	**ADDR[6]**	78	**CANCELWAIT**, If HIGH, this signal enables the system to recover from an externally waited transfer that has taken longer than expected to finish. Pull LOW if not used.
79	**ADDR[7]**	80	**nCS[4]**
81	**ADDR[8]**	82	**nCS[3]**
83	**ADDR[9]**	84	**nCS[2]**
85	**ADDR[10]**	86	**nCS[1]**
87	**ADDR[11]**	88	Reserved, do not drive
89	**ADDR[12]**	90	Reserved, do not drive
91	**ADDR[13]**	92	Reserved, do not drive

Table F-4 Static memory connector signals (continued)

Pin No.	Signal	Pin No.	Signal
93	**ADDR[14]**	94	Reserved, do not drive
95	**ADDR[15]**	96	**nCS[0]**
97	**ADDR[16]**	98	**nBUSY**, Indicates that memory is not ready to be released from reset. If LOW, this signal holds **nRESET** active.
99	**ADDR[17]**	100	**nIRQ**
101	**ADDR[18]**	102	**nWEN**
103	**ADDR[19]**	104	**nOEN**
105	**ADDR[20]**	106	**nBLS[3]**, Byte Lane Select for bits [31:24]
107	**ADDR[21]**	108	**nBLS[2]**, Byte Lane Select for bits [23:16]
109	**ADDR[22]**	110	**nBLS[1]**, Byte Lane Select for bits [15:8]
111	**ADDR[23]**	111	**nBLS[0]**, Byte Lane Select for bits [7:0]
113	**ADDR[24]**	114	**CSWIDTH[0]**, Indicates bus width for fitted part. Do not route through stackable boards.
115	**ADDR[25]**	116	**CSWIDTH[1]**, Indicates bus width for fitted part. Do not route through stackable boards.
117	**ADDRVALID**, Indicates that the address output is stable during synchronous burst transfers	118	**CLK[1]**
119	**BAA**, Burst Address Advance. Used to advance the address count in the memory device	120	**CLK[0]**

a. VDDIO is the data voltage to the host. Do not route through on stackable boards

F.5 Mechanical layout

Figure F-5 shows the static memory board (viewed from above).

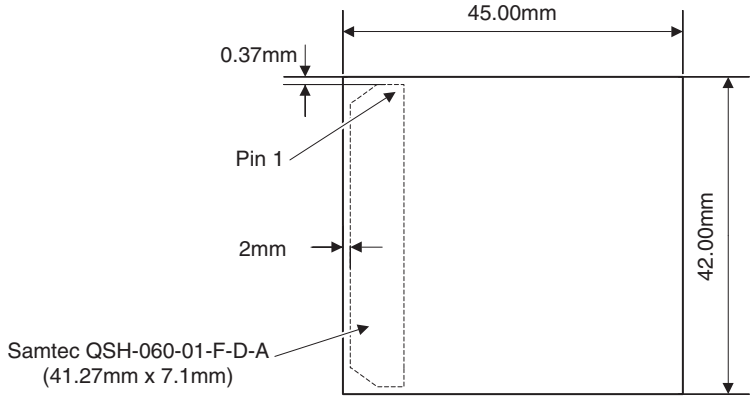

Figure F-5 Static memory board layout

Appendix G
Configuring a USB Debug Connection

When you install the RealView® ICE Micro Edition software that is provided with RVDS version 2.1 or higher, various features are added to the RealView® Debugger. This appendix tells you how to use these additional features to configure the Versatile/AB926EJ-S USB debug port connection, and how to connect RealView Debugger to the Versatile/AB926EJ-S. It contains the following sections:

Note

This chapter assumes that you are familiar with how to use RealView Debugger to connect to a target, and to configure a connection. For details, refer to the RealView Debugger documentation suite (see the *RealView Debugger v1.7 Target Configuration Guide*).

G.1 Installing the RealView ICE Micro Edition driver

The first time you connect a USB cable between the USB debug port on the Versatile/AB926EJ-S and you computer, the Windows operating system Plug and Play manager detects the unit and launches the Add New Hardware Wizard to install the RealView ICE Micro Edition driver. If the wizard does not appear, you can run it manually from the Control Panel.

The installation process varies depending on the operating system you are using. See the following sections:

- *Installing the RealView ICE Micro Edition driver on Windows 98SE*
- *Installing the RealView ICE Micro Edition driver on Windows 2000* on page G-3
- *Installing the RealView ICE Micro Edition driver on Windows XP Professional* on page G-3.

G.1.1 Installing the RealView Developer Suite

The basic components of RVDS 2.1 (or higher) and the RVI-ME component of RVD 1.7 (or higher) must already be present on your workstation before you begin configuring the USB debug port software. To install the RVDS software:

1. See the installation instructions provided with RVDS for details on installing that product.
2. After you have installed RVDS using the standard installation procedure, rerun the RVDS installation, but select **Custom** installation instead of **Typical** installation.
3. From the displayed list of items that can be installed, select only the RVI-ME software and click **OK**.
4. Continue the installation as described in the documentation supplied with RVDS.

G.1.2 Installing the RealView ICE Micro Edition driver on Windows 98SE

To install the RealView ICE Micro Edition driver on Windows 98SE:

1. Ensure that no RealView Debugger component is running.
2. Connect a USB cable between the USB debug port and your computer. The Add New Hardware Wizard is launched, and tells you that Windows has found the RealView ICE Micro Edition device.
3. Click **Next**. Select **Search for the best driver for your device**.

4. Click **Next**. Specify where you want Windows to search for the driver files:
   a. Select **Specify a location**.
   b. Click the **Browse...** button and navigate to the installation directory you selected for the RVI-ME software in *Installing the RealView Developer Suite* on page G-2.
   c. Click **OK**.
5. Click **Next**. The Add New Hardware Wizard locates the driver.
6. Click **Next**. Windows installs the driver.
7. Click **Finish** to close the wizard.

G.1.3 Installing the RealView ICE Micro Edition driver on Windows 2000

To install the RealView ICE Micro Edition driver on Windows 2000:

1. Ensure that no RealView Debugger component is running.
2. Connect a USB cable between the USB debug port and your computer. The Found New Hardware Wizard is launched, and displays a welcome message.
3. Click **Next**. The Install Hardware Device Drivers window is opened. Select **Search for a suitable driver for my device**.
4. Click **Next**. The Locate Driver Files window is opened. Specify where you want Windows to search for the driver files:
   a. Select **Specify a location**.
   b. Click the **Browse...** button and navigate to the installation directory you selected for the RVI-ME software in *Installing the RealView Developer Suite* on page G-2.
   c. Click **OK**.
5. Click **Next**. The Driver Files Search Results window is opened.
6. Click **Next**. The Completing the Found New Hardware Wizard window is opened.
7. Click **Finish** to finish the installation and close the wizard.

G.1.4 Installing the RealView ICE Micro Edition driver on Windows XP Professional

To install the RealView ICE Micro Edition driver on Windows XP Professional:

1. Ensure that no RealView Debugger component is running.

2. Connect a USB cable between the USB debug port and your computer. The Add New Hardware Wizard is launched, and displays a welcome message.
3. Click **Next**. Specify how you want Windows to find the required files:
   - Select **Install from a list of specific locations** and check **Search for the best driver in these locations**.
   - Click the **Browse...** button and navigate to the installation directory you selected for the RVI-ME software in *Installing the RealView Developer Suite* on page G-2.
   - Click **OK**.
4. Click **Next**. A message is displayed informing you that the device you are installing has not passed Windows Logo testing to verify its compatibility with Windows XP. Click **Continue Anyway**.
5. Windows completes installation of the driver. Click `Finish` to finish the installation and close the wizard.

G.2 Changes to RealView Debugger

When you install the RealView ICE Micro Edition software, it adds the following capabilities to RealView Debugger:

- New nodes in the **Connection Control** window:
  - an `ARM-ARM-DIR` target vehicle node at the top level lists the direct connection devices.
  - an `VPB926EJ-S USB` access provider node that appears at the second level, for connecting to and configuring the USB debug port
  - target nodes that appear at the third level, for establishing a debugging connection to the ARM926PXP development chip.

  These nodes are shown in Figure G-1.

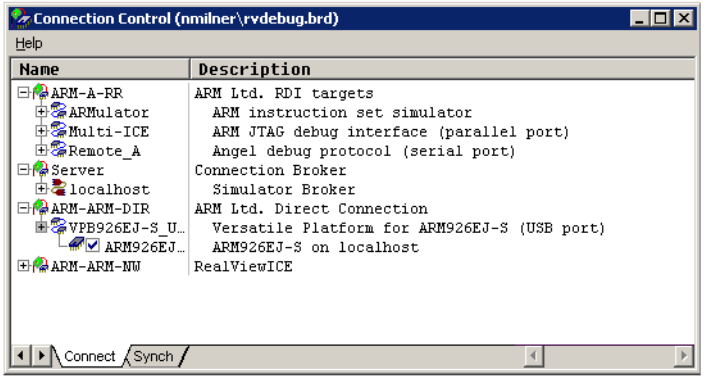

Figure G-1 Nodes added to Connection Control window

- New tabs in the **Register** pane of the Code window, in addition to the **Core** tab that is present for all targets. The additional tabs include:
  - a **CP15** tab that displays and sets the values of registers in coprocessor 15 (the System Control coprocessor)
  - a **Cache Operations** tab that you can use to perform operations on the cache for the target
  - a **TLB Operations** tab that you can use to perform operations on the *translation look-aside buffer* (TLB) for the target
  - a **Debug** tab that controls various internal debugger settings, many of which are specific to the USB debug port.

  The **CP15**, **Cache Operations**, and **TLB Operations** tabs control features of the target hardware. These features are described in the *ARM926EJ-S Technical Reference Manual*.

G.3 Using the USB debug port to connect RealView Debugger

To connect to the Versatile/AB926EJ-S using the USB debug port, you use the same RealView Debugger features that you use for any other target.

For more information about connecting RealView Debugger to targets, refer to the RealView Debugger documentation suite.

G.3.1 Configuration

To use the RealView Debugger with the Versatile/AB926EJ-S:

1. Start the RealView Debugger.
2. Connect a USB cable between the PC and the USB debug port on the Versatile/AB926EJ-S.
3. Display the RealView Debugger **Connection Control** window in one of the following ways:
   - Click on the blue hyperlink in the File Editor window, if available.
   - Select **File** → **Connection** → **Connect to Target** from the Code window.
   - Use the keyboard shortcut Alt+0 with the Code window active.

   The **Connection Control** window appears, as shown in Figure G-2.

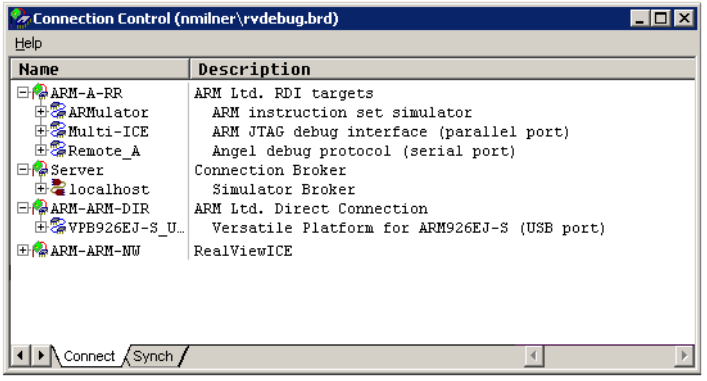

Figure G-2 The Connection Control window

4. Click on **VPB926EJ-S USB** in the **Connection Control** window. If the debugger is able to connect to the Versatile/AB926EJ-S, the **Connection Control** window displays the connection to the ARM926PXP development chip as shown in Figure G-3 on page G-7.

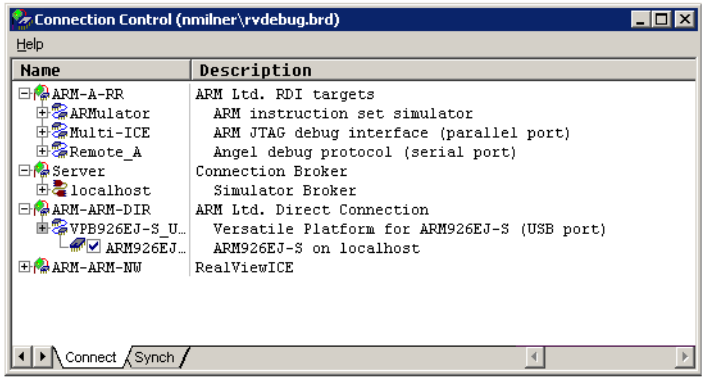

Figure G-3 ARM926PXP development chip detected

Note

If the there is not a **VPB926EJ-S USB** entry in the **Connection Control** window, the RealView ICE Micro Edition software is not installed. Close the RealView Debugger and install the software as described in *Installing the RealView ICE Micro Edition driver* on page G-2.

You might see one of the following errors:

- If the RealView Debugger detects any unpowered devices, it displays the error shown in Figure G-4. If you see this error, ensure that power is supplied to all your devices.

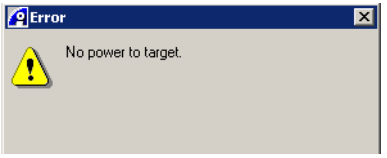

Figure G-4 Error shown when unpowered devices are detected

- If RealView Debugger does not detect any devices, it displays the error shown in Figure G-5. If you see this error, ensure that the USB cable is properly attached.

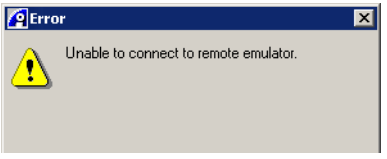

Figure G-5 Error shown when no devices are detected

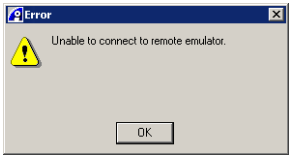

Figure G-6 Error shown when the USB debug port is not functioning

5. Right-click on **VPB926EJ-S USB** in the **Connection Control** window and select **Connection Properties** from the context menu that appears. The **Connection Properties** window is displayed as shown in Figure G-7.

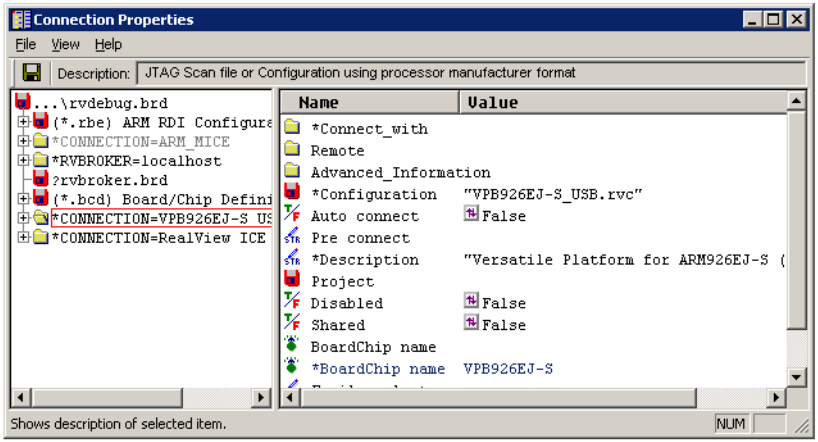

Figure G-7 Connection Properties window

6. You can change connection properties by selecting controls in the **Name** column.

Note

The default values for the connection do not typically require changing.

7. Close the **Connection Properties** window and return to the Code window in the RealView Debugger.

   You can now use the RealView Debugger to download programs to the Versatile/AB926EJ-S and debug them.

G.4 Using the Debug tab of the RealView Debugger Register pane

When you install the RealView ICE Micro Edition software and connect to a Versatile/AB926EJ-S, a **Debug** tab is added to the **Register** pane of the RealView Debugger Code window. This controls various internal debugger registers, many of which are specific to using the USB debug port. To use this tab, you must first connect RealView Debugger to your Versatile/AB926EJ-S, as described in *Using the USB debug port to connect RealView Debugger* on page G-6. A typical setting window is shown in Figure G-8.

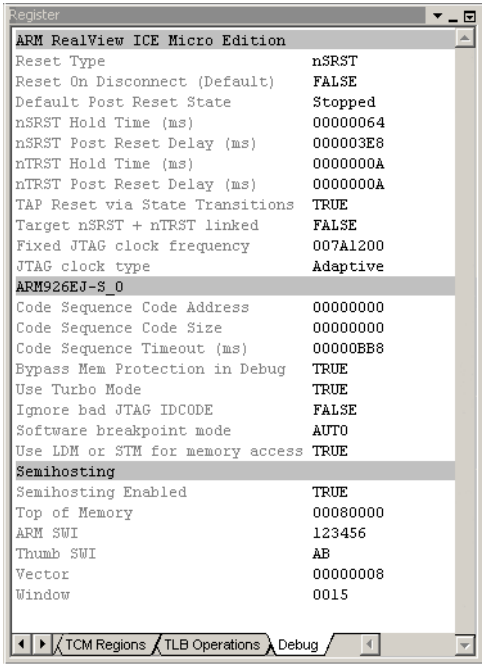

Figure G-8 The Debug tab of the Register pane

There are three groups of settings:

- *Global Properties*
- *Device Properties* on page G-10
- *Semihosting Properties* on page G-11.

G.4.1 Global Properties

The **Global Properties** area of the **Debug** tab contains settings that control the behavior of the USB debug port when it resets the target hardware.

Table G-1 shows:

- the name of each setting
- the name of the corresponding RealView Debugger internal register
- for enumerated settings, the name and value of each possible enumerator.

Table G-1 Reset behavior register names and values

Setting	Register name	Enumerator	Value
Reset Type	RESETOPERATION	**nSRST**	0x0
		nTRST	0x1
		nSRST + nTRST	0x2
		Fake	0x3
Reset On Disconnect (Default)	RESETONDISCONNECT	**FALSE**	0x0
		TRUE	0x1
Default Post Reset State	POSTRESETSTATE	**Running**	0x0
		Stopped	0x1
nSRST Hold Time (ms)	RESETHOLDTIME	-	-
nSRST Post Reset Delay (ms)	POSTRESETDELAY	-	-
nTRST Hold Time (ms)	NTRSTHOLDTIME	-	-
nTRST Post Reset Delay (ms)	NTRSTPOSTRESETTIME	-	-
TAP Reset via State Transitions	DOSOFTTAPRESET	**FALSE**	0x0
		TRUE	0x1
Target nSRST + nTRST linked	LINKED_SRST_TRST	**FALSE**	0x0
		TRUE	0x1
JTAG Clock Type	JTAGCLOCKTYPE	**Fixed**	0x0
		Adaptive	0x1

G.4.2 Device Properties

The settings in the ***Device*** **Properties** area of the **Debug** tab of the RealView Debugger register pane control the connected device. Table G-2 on page G-11 shows:

- the name of each device property setting

- the name of the corresponding RealView Debugger internal register
- for enumerated settings, the name and value of each possible enumerator.

Table G-2 Device property register names and values

Setting	Register name	Enumerator	Value
Code Sequence Code Address[a]	CODESEQ_CODE_ADDR	-	-
Code Sequence Code Size	CODESEQ_CODE_SIZE	-	-
Code Sequence Timeout (ms)	CODESEQ_TIMEOUT	-	-
Bypass Mem Protection in Debug[bc]	BYPASS_MEMPROT_IN_DBG	**FALSE**	0x0
		TRUE	0x1
Enable Turbo Mode	USE_TURBO_MODE	**FALSE**	0x0
		TRUE	0x1
Ignore Bad JTAG IDCODE	IGNORE_BAD_JTAG_IDCODE	**FALSE**	0x0
		TRUE	0x1
Software Breakpoint Mode	SOFTWARE_BREAKPOINT_MODE	**Auto**	0x0
		None	0x1
		Watchpoint	0x2
		Breakpoint	0x3
Use LDM or STM for Memory Access	USE_LDM_STM	**FALSE**	0x0
		TRUE	0x1

a. You must configure the **Code Sequence…** settings in the **Debug** tab before caching has been enabled. If you cannot halt the target before its caches are enabled, you must instead configure these settings before connecting (as described in *Configuration* on page G-6).
b. You must configure the **Bypass Mem Protection in Debug** setting in the **Debug** tab before memory protection has been enabled. If you cannot halt the target before its memory protection is enabled, you must instead configure these settings before connecting (as described in *Configuration* on page G-6).
c. The **Bypass Mem Protection in Debug** setting does not take effect until the next time that you enter debug state.

G.4.3 Semihosting Properties

The settings in the **Semihosting Properties** area of the **Debug** tab in the RealView Debugger **Register** pane are the same as those used for other debug targets. For details of these settings, see the *RealView Debugger User Guide*.

Index

N

O

P

R

S

T

U

V

W