Integrator®/CP

Compact Platform Baseboard HBI-0086

User Guide

ARM®

ARM DUI 0159B

Integrator/CP
User Guide

Release Information

Description	Issue	Change
March 2002	A	First release
November 2002	B	Second release

Proprietary Notice

Confidentiality Status

Product Status

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

Web Address

`http://www.arm.com`

Conformance Notices

This section contains conformance notices.

Federal Communications Commission Notice

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

CE Declaration of Conformity

The system should be powered down when not in use.

The Integrator 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
Integrator/CP User Guide

Preface

This preface introduces the documentation for the Integrator/CP compact platform baseboard. It contains the following sections:

About this document

This document provides a guide to setting up and using the ARM Integrator/CP.

Intended audience

This document has been written for experienced hardware and software developers as an aid to using the ARM Integrator/CP platform to develop ARM-based products.

Organization

This document is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an introduction to the ARM Integrator/CP. This chapter identifies the main components and connectors.

Chapter 2 ***Getting Started***

Read this chapter for a description of how to set up and start using the Integrator/CP. This chapter describes how to attach modules to the Integrator/CP and how to apply power.

Chapter 3 ***System Architecture***

Read this chapter for a description of the system architecture. This includes the system buses, memory, and programmable devices.

Chapter 4 ***Peripherals and Interfaces***

Read this chapter for a description of the peripherals supported by the Integrator/CP.

Chapter 5 ***System Expansion***

Read this chapter for a description of the expansion hardware that you can use with your Integrator/CP.

Appendix A ***Porting Integrator/AP and IM-PD1***

Refer to this appendix for instructions on porting existing applications to an Integrator/CP system.

Appendix B ***Connector Pinouts***

Refer to this appendix for a description of the connector signals.

Appendix C ***Test Points***

Refer to this appendix for the location of test points.

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 may provide additional information.

ARM publications

The following publications provide information about related ARM Integrator modules:

- *ARM Integrator/AM User Guide* (ARM DDI 0133)
- *ARM Integrator/LM-XCV600E and LM-EP20K600E User Guide* (ARM DUI 0146)
- *ARM Integrator/LM-X2CV4000+ User Guide* (ARM DUI 0130).

The following publications provide information about ARM PrimeCell® devices that can be used to control the interfaces described in this manual:

- *ARM PrimeCell UART (PL011) Technical Reference Manual* (ARM DDI 0183)
- *ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual* (ARM DDI 0173).
- *ARM PrimeCell GPIO (PL061) Technical Reference Manual* (ARM DDI 0190)
- *ARM PrimeCell Color LCD Controller (PL110) Technical Reference Manual* (ARM DDI 0161).

- *ARM PrimeCell Smart Card Interface (PL130) Technical Reference Manual* (ARM DDI 0148)
- *ARM PrimeCell Multimedia Card Interface (PL181) Technical Reference Manual* (ARM DDI 0205).

The following publications provide reference information about the ARM architecture:

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

The following publication provides information about the ARM Firmware Suite:

- *ARM Firmware Suite Reference Guide* (ARM DUI 0102)

The following publications provides information about Multi-ICE® and MultiTrace®:

- *Multi-ICE User Guide* (ARM DUI 0048).
- *Trace Debug Tools User Guide* (ARM DUI 0118).
- *ARM MultiTrace User Guide* (ARM DUI 0150).

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 Integrator/CP:

- *CODEC with Sample Rate Conversion and 3D Sound* (LM4549) National Semiconductor, Santa Clara, CA.
- *DAC-Controlled Boost/Inverter LCD Bias Supply* (MAX686) Maxim Corporation, Sunnyvale, CA.
- *Digital Dimming CCFL Inverter Module* (LXM1611) LinFinity Microelectronics Inc., Garden Grove, CA. (This is an optional module that might be used with an external LCD panel.)
- *Dual 250mW Audio Power Amplifier* (LM4880) National Semiconductor Corporation, Santa Clara, CA.
- *MAX 3000A, MAX7000A Programmable Logic Device Family Data Sheets* Altera, San Jose, CA.
- *MicroClock OSCaR User Configurable Clock Data Sheet* (MDS525), MicroClock Division of ICS, San Jose, 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.
- *TFT-LCD Module* (LQ084V1DG21) Sharp Corporation, Osaka, Japan.
- *Three-In-One Fast Ethernet Controller* (LAN91C111) SMSC, Hauppauge, NY.
- *Touch Screen Controller* (ADS7843) Burr-Brown, Tucson, AZ.
- *Virtex Field Programmable Gate Arrays* (XCV600) Xilinx Corporation, San Jose, CA.

Feedback

ARM Limited welcomes feedback both on the ARM Integrator/CP, and on the documentation.

Feedback on the ARM Integrator/CP

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

- the product name
- an explanation of your comments.

Feedback on this document

If you have any comments about this document, please 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 Integrator/CP compact platform baseboard. It contains the following sections:

1.1 About the Integrator/CP

The Integrator/CP is a compact development platform that gives you a flexible environment to enable rapid development of ARM-based devices. It enables you to model your product, aid you with the development of hardware and software, and produce proof of concept prototypes.

The Integrator/CP baseboard is used as part of a two-board system:

- the baseboard provides interface clocks, power, boot memory, and interfaces
- the core module provides the ARM core, SDRAM, SSRAM, memory and core clocks, and an FPGA that implements peripheral devices.

Additional peripherals and interfaces can be added to your system by adding up to three logic modules.

Note

ARM interface modules such as the IM-PD1 can be used, but some system features are unavailable. If you use an interface module with an Integrator/CP system, for example, the CLCD and VGA interfaces on both CP and IM-PD1 boards use the same set of pins on HDRA for the display controller, but with different pin assignments. Therefore, the VGA/CLCD interfaces on the IM-PD1 would be unusable in this configuration.

Figure 1-1 on page 1-3 shows the layout of an assembled system.

Note

Multiple core modules cannot be used with the baseboard because the Integrator/CP uses a single-master AHB-Lite bus rather than a multimaster AHB bus.

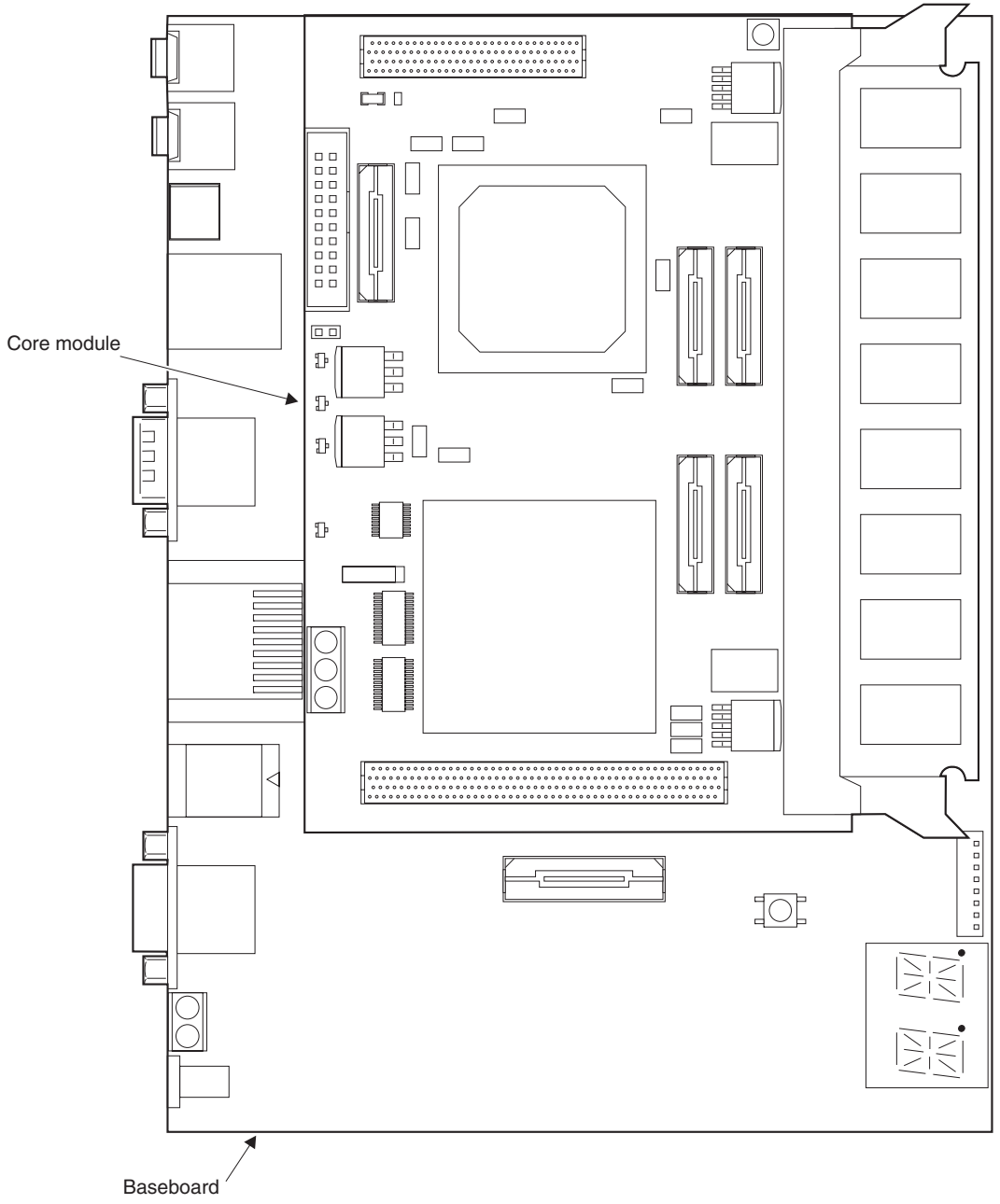


Figure 1-1 System layout

1.2 System description

This section provides an overview of the baseboard as follows:

- *System features*
- *System connectors* on page 1-5
- *Baseboard LEDs* on page 1-8.

1.2.1 System features

The core module and baseboard system provides the following:

- ARM microprocessor core.
- Volatile memory comprising 1MB SSRAM onboard and, optionally, 16 to 256MB of SDRAM plugged into the DIMM socket.
- 16MB flash memory.
- Video *Digital to Analog Converter* (DAC) with VGA interface.
- Audio CODEC.
- Touchscreen controller, keyboard, and mouse interface
- LAN91C111 Ethernet interface controller supporting 10BaseT and 100BaseTx.
- *MultiMedia Card Interface* (MMCI).
- Dual UART.
- Alphanumeric display, LEDs, switches.
- Programmable clock generators.
- System support PLD containing:
  - AHB host interface to the Ethernet controller
  - flash memory access controller
  - 8-bit GPIO interface.
- System controller FPGA incorporating the following functional blocks:
  - Dual-port SDRAM controller
  - ZBT SRAM controller
  - *Color Liquid Crystal Display* (CLCD) Controller
  - peripheral control logic
  - AHB/AHB and AHB/APB system bus bridges.
- System bus connectors.
- Multi-ICE debug connector.
- Logic analyzer connectors for local memory bus.
- Trace port connector (for those cores that have Embedded Trace Macrocell).

1.2.2 System connectors

The CP platform has a large number of connectors on the baseboard and core module.

Baseboard connectors

Table 1-1 lists the connectors on the Integrator/CP baseboard.

Table 1-1 Baseboard connectors

Connector	Name	Function
J1	HDRA	System bus connector to core module.
J2	HDRB	System bus and interface connector to core module.
J3	DC Input	Power supply input, 9 to 20V DC
J4	DC Input	Power supply input, 9 to 20V DC (12V DC adaptor supplied with product)
J5	Backlight	TFT panel backlight
J7	Ethernet	10BaseT/100BaseTx Ethernet connector
J8	Ethernet Debug	Debug connector for the ethernet interface
J9	MMC Debug	Debug connector for the MMC interface
J10	Line	Line in (bottom) and line out (top).
J11	MMC	MMC socket
J12	MIC IN	Microphone input
J13	Generic LCD	Connector to generic LCD display panel.
J14	TFT Panel	Connector to TFT LCD display.
J15	VGA	Connector to VGA type display.
J16	Touchscreen	Touchscreen interface
J17	Serial	Dual RS232 serial interfaces, port A top, port B bottom
J18	LED	Off-board LED interface
J19	KMI	Keyboard (bottom) and mouse (top)
J20	GPIO	Interface to 8-bit GPIO
J22	Debug	Debug connector for the Ethernet controller host signals

Figure 1-2 identifies the connectors on the Integrator/CP baseboard.

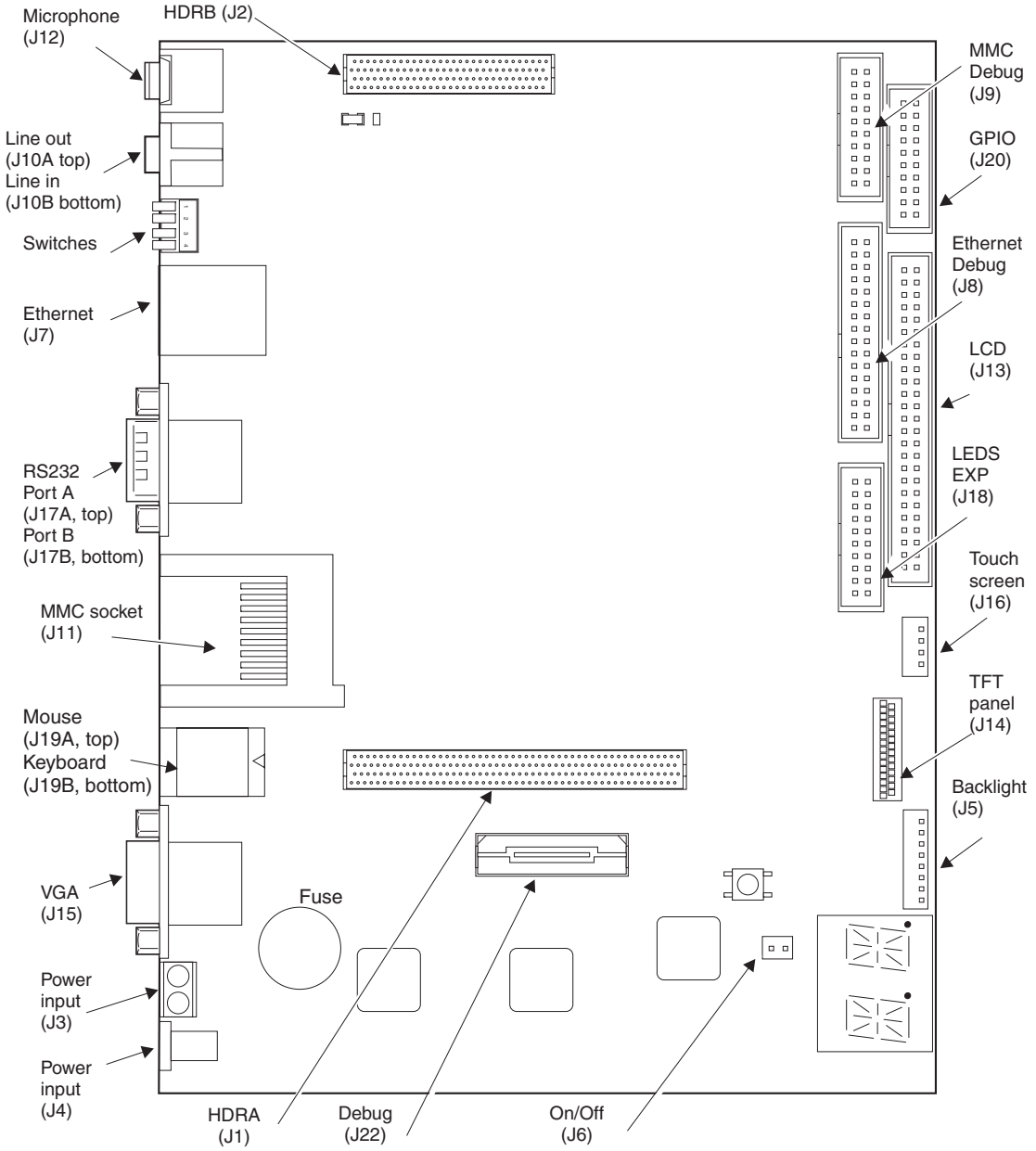


Figure 1-2 Baseboard layout

Core module connectors

Figure 1-3 shows the connectors for a typical core module. Other core modules have similar connectors, however, the number and location of trace connectors can vary.

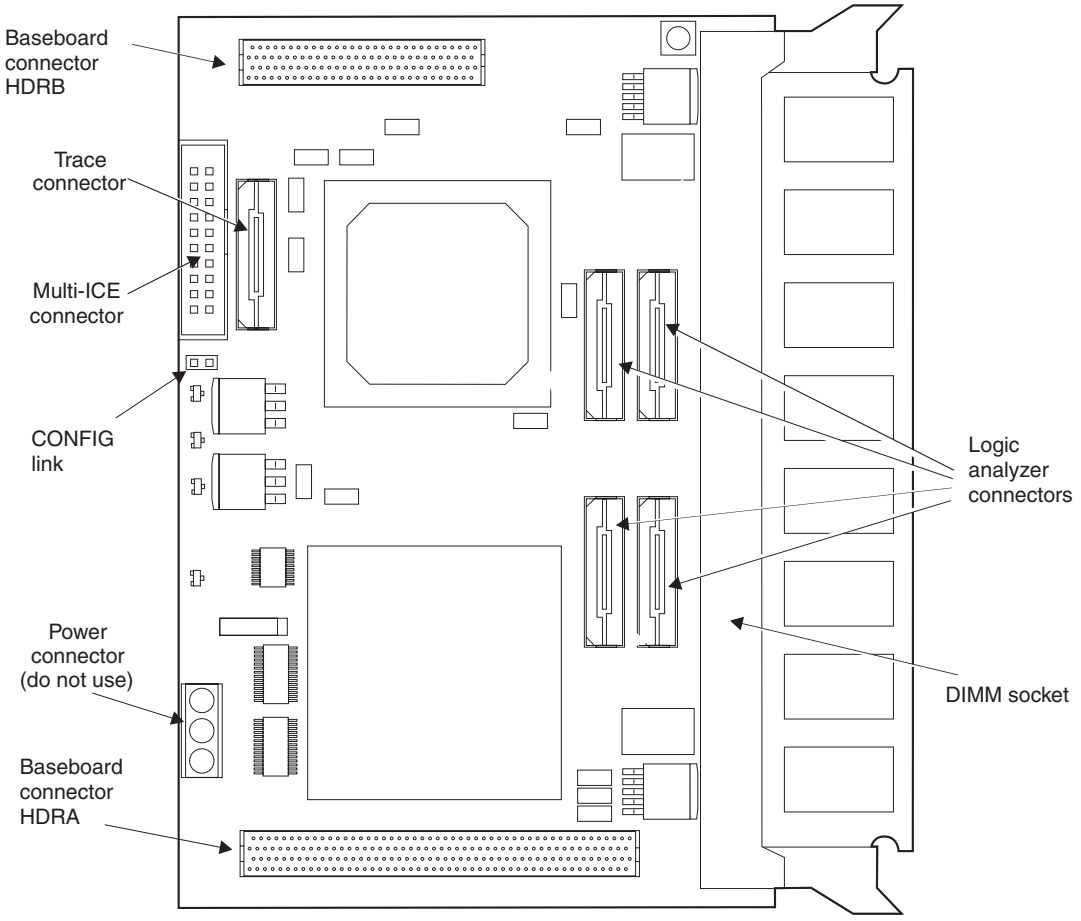


Figure 1-3 Core module connectors, CP966E-S

The CONFIG link selects between debug operation and configuration mode. If the CONFIG link is not fitted, the processor core and debuggable devices on other modules are accessible on the scan chain and you can download and run programs.

For more details on configuration see the documentation provided with your core module.

1.2.3 Baseboard LEDs

The Integrator/CP baseboard LEDs are shown in Figure 1-4.

Ethernet LEDs A and B are duplicated within the Ethernet connector. The function of the Ethernet LEDs can be configured from the Ethernet control registers, see *Ethernet interface* on page 4-4 and the datasheet for the Ethernet controller.)

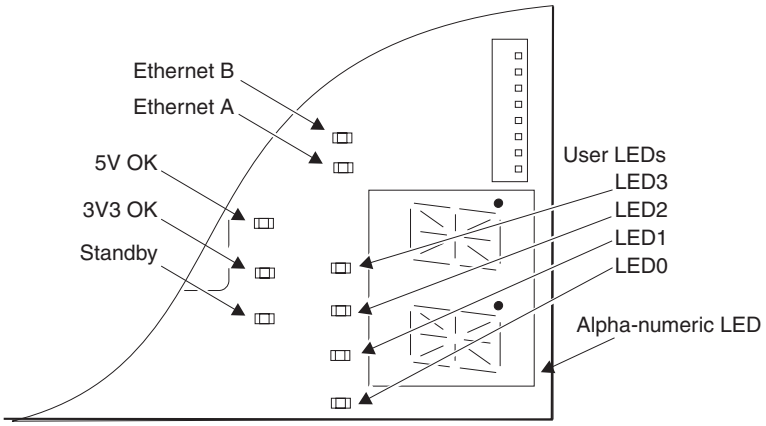


Figure 1-4 Baseboard LEDs

The function of the LEDs on the Integrator/CP baseboard is summarized in Table 1-2.

Table 1-2 Baseboard LED functional summary

LED		Color	Function
3V3 OK	(D9)	Green	Indicates that a 3.3V supply is available.
5V OK	(D10)	Green	Indicates that a 5V supply is available.
STANDBY	(D12)	Red	This LED illuminates when power is applied to the power connector to indicate that the power supply is in standby mode. To power on the Integrator/CP, press the POWER button.
LED0	(D16)	Green	This is a general purpose LED controlled by writing to bit 0 in the LED_LIGHTS register.

Table 1-2 Baseboard LED functional summary (continued)

LED		Color	Function
LED1	(D17)	Yellow	This is a general purpose LED controlled by writing to bit 1 in the LED_LIGHTS register.
LED2	(D18)	Red	This is a general purpose LED controlled by writing to bit 2 in the LED_LIGHTS register.
LED3	(D19)	Green	This is a general purpose LED controlled by writing to bit 3 in the LED_LIGHTS register.

Note

The Ethernet interface IC, LAN91C111, directly controls two LEDs on the Ethernet connector. These indicate activity on the interface.

1.3 Precautions

This section contains safety information and advice on how to avoid damage to the Integrator/CP baseboard and core module.

1.3.1 Ensuring safety

To power the system, connect the power supply provided with the Integrator/CP product package to baseboard connector J4 (or an equivalent supply to J3). The core module is powered by the baseboard.

Warning

To avoid a safety hazard, only *Safety Extra Low Voltage* (SELV) equipment must be connected to the Integrator/CP.

1.3.2 Preventing electrostatic damage

The Integrator system is intended for use within a laboratory or engineering development environment. Because it is supplied without an enclosure, it is sensitive to electrostatic discharges and electromagnetic emission.

Caution

To avoid damage to the boards you must observe the following precautions:

- Never subject the boards to high electrostatic potentials.
- Always wear a grounding strap when handling the boards.
- Only hold the boards by the edges.
- Avoid touching the component pins or any other metallic element.

Do not use the boards near equipment that could be:

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

1.3.3 Care of connectors

This section contains advice about how to prevent damage to the connectors on Integrator modules.

Caution

To prevent damage to your modules, observe the following precautions:

- When separating modules, take care not to damage the connectors. Do not apply a twisting force to the ends of the connectors. Loosen each connector first before pulling on both ends of the module at the same time.
- Use the Integrator system in a clean environment and avoid debris fouling the connectors on the underside of the PCB. Blocked holes result in damage to connectors. Visually inspect the module to ensure that connector holes are clear before mounting modules onto the Integrator.

1.3.4 Core module image

The CP baseboard requires a compatible image loaded into the FPGA of the core module. Ensure that your core module image is compatible with the CP baseboard. For more details on image selection, see the documentation provided with your core module.

Chapter 2
Getting Started

This section describes the steps required to prepare an Integrator/CP system for use:

- *Setting the DIP switches* on page 2-3
- *Connecting power* on page 2-4
- *Connecting Multi-ICE and Trace* on page 2-6
- *System expansion* on page 2-7.
- *Using the ARM Firmware Suite boot monitor* on page 2-9.

Figure 2-1 on page 2-2 shows the Integrator/CP system.

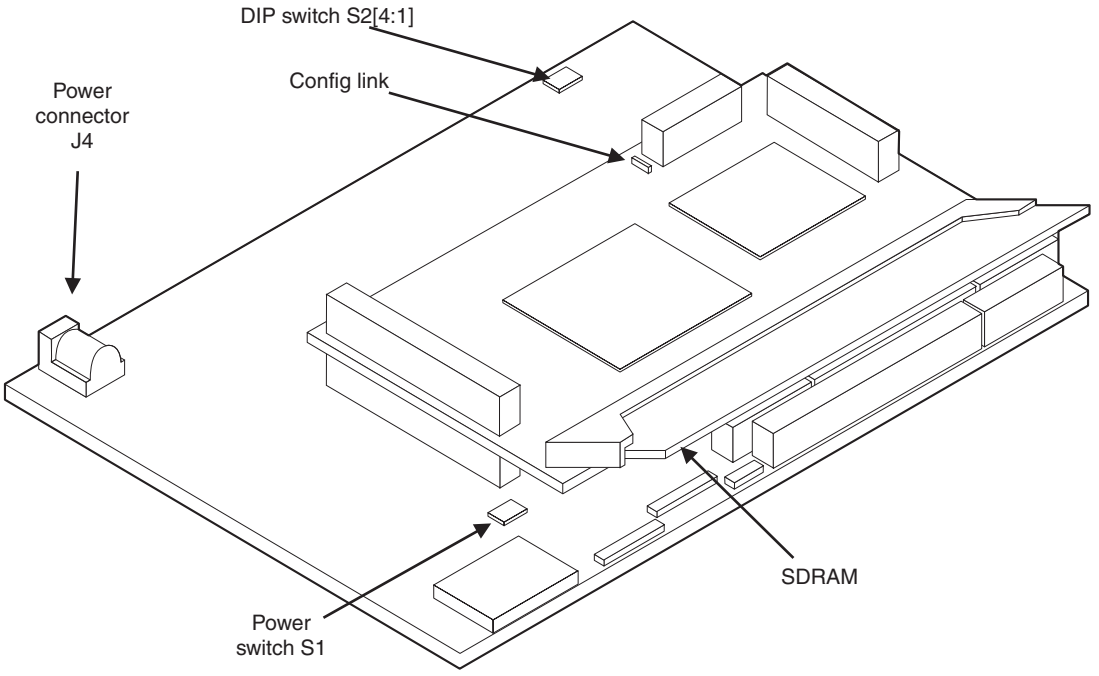


Figure 2-1 Integrator/CP system

2.1 Setting the DIP switches

The 4-pole DIP switch (S2) is used to control system startup. At reset, S2[1] is assigned a special meaning by the hardware. The switch settings can be read from the LED and switch register (LED_SWITCHES).

Caution

Check the documentation provided with your core module for CM-specific details on image selection and startup.

Switches S2[2] and S2[3] are not used to control startup. After the system has started, all of S2[4:1] can be assigned any meaning required by the system developer.

S2[1] is used to control where code execution begins (S2[1] is sampled by hardware). The boot switcher component of the ARM Firmware Suite then uses S2[4] to select whether to jump to the user area of the flash or to remain in a loop polling for serial input. The switch settings are shown in Table 2-1 (where x = *do not care*).

Table 2-1 DIP switch settings

S2[1]	S2[2]	S2[3]	S2[4]	Code execution start location following reset
ON	x	x	ON	The boot code area of the flash. The boot monitor banner is output on serial port A. The boot monitor can be controlled by a dumb terminal or terminal emulator.
ON	x	x	OFF	The boot code area of the flash following reset, then control jumps to the user code area of flash.
OFF	x	x	x	User code area of flash following reset.

To communicate with the boot monitor, connect a terminal using a null-modem cable to serial port A (J17). See *Using the ARM Firmware Suite boot monitor* on page 2-9. The boot monitor only runs if S2[1] is in the ON position. For information about the flash memory see *Flash memory* on page 3-11.

Note

Switches S2[3:2] might be used by programs such as the boot monitor for other configuration options. (For example, on some versions of boot monitor for the CP9x6E-S, switch S2[2] turns the PLL on or off.)

See the *ARM Firmware Suite Reference Guide* for more information on the boot monitor.

2.2 Connecting power

The Integrator/CP can be powered from the external *Power Supply Unit* (PSU) packaged with the system or from an alternative external D.C. supply. Figure 2-2 shows the power supply connectors.

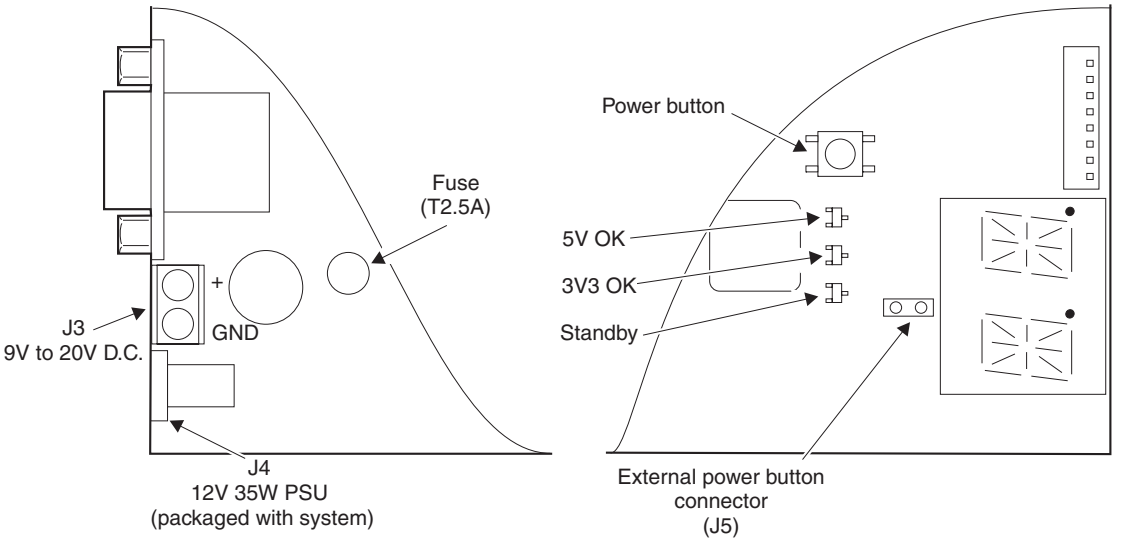


Figure 2-2 Powering the assembled Integrator/CP system

Connect either:

- the PSU packaged with your system to J4 on the baseboard. The PSU is capable of supplying 12V at 2.92A from an A.C. supply of between 100V and 240V. (This is the preferred supply.)
- an external supply, between 9V and 20V D.C., to connector J3.

Onboard voltage regulators provide 3.3V and 5V. The on-board regulators can supply the maximum currents shown in Table 2-2 on page 2-5 from a 9V to 20V input. This input is protected by a 2.5A time-delayed fuse and shuts down if the supply voltage dips below 9V. On some models, the fuse is in a different location, but it is easily recognized.

When you apply power to J3 or J4, the standby LED illuminates. To power the system up, press the power button (S1).

Caution

Do not connect power to the power connector on the core module.

Table 2-2 shows the approximate power requirements of a typical system and an estimate of the power required for an additional logic module. You must also make an allowance for any external display powered by the system.

Table 2-2 System power requirements

Rail	On-board PSU maximum output	Typical system	Logic module
5V	1.5A	100mA + display	500mA
3.3V	2A	600mA	500mA
LCD bias 11.5 to 20V d.c.	100mA	100mA	-
5V analog (for audio)	250mA	250mA	-

2.3 Connecting Multi-ICE and Trace

Multi-ICE debugging equipment can be used to download and debug programs. Connect the Multi-ICE unit to the module at the top of the stack. Integrator modules route the JTAG signals from this connector to other modules in the system. See the documentation for the core module for more details on using Multi-ICE.

During normal operation and software development, the core module operates in *debug* mode. You can use debug mode to download and run programs. Select debug mode by ensuring that the CONFIG link is not fitted. In debug mode, the processor core and debugable devices on other modules are accessible on the scan chain.

In *configuration* mode, all FPGAs and PLDs in the system are added into the scan chain. This enables the logic devices to be reconfigured or upgraded in the field.

If you are using Trace, connect the trace port adapter board to the target board as shown in Figure 2-3.

Note

Different boards can have different sizes (small, medium, or large) or there might be no ETM. Support for ETM is dependent on the test chip fitted in the core module. See the documentation provided with the core module and the trace product for more details.

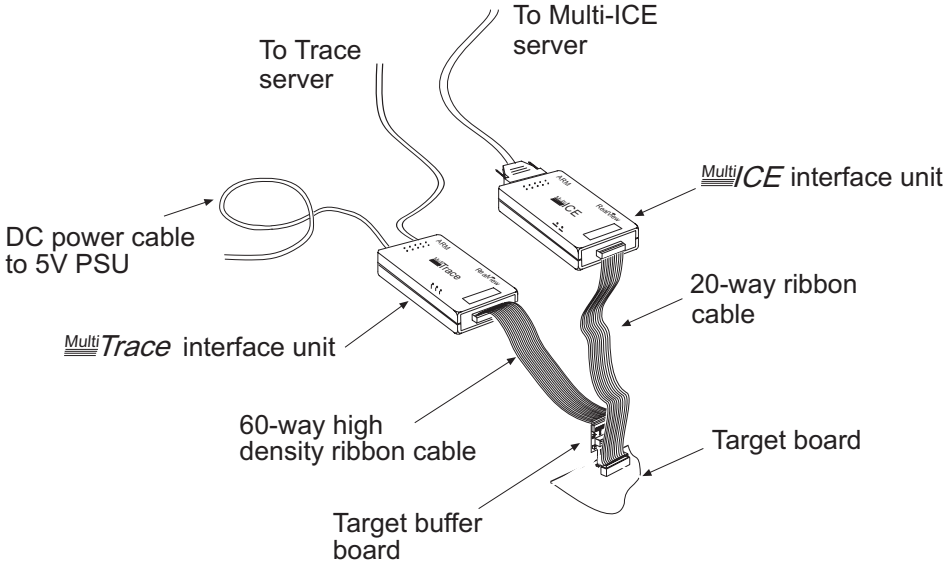


Figure 2-3 Connecting Trace

2.4 System expansion

You can expand the Integrator/CP baseboard by adding:

- a core module (the baseboard does not contain a processor)
- up to three Integrator logic modules
- an IM-LT1 interface module (for use with logic tiles)
- your own modules
- A different dynamic memory DIMM.

Caution

Always power OFF before adding or removing expansion modules.

2.4.1 Expanding your system with Integrator modules

There are restrictions on how you can expand your system with ARM Integrator modules:

- Signals on HDRA/HDRB must not clash.

  Note

  ARM interface modules such as the IM-PD1 can be used, but some system features are unavailable. If you use an interface module with an Integrator/CP system, for example, the CLCD and VGA interfaces on both CP and IM-PD1 boards use the same set of pins on HDRA for the display controller, but with different pin assignments. Therefore, the VGA/CLCD interfaces on the IM-PD1 would be unusable in this configuration.

- Loading for each line is limited.
- Interrupt assignment is position-dependent.
- Addressing is position-dependent (see *Module-assigned signals* on page 3-6).
- Inserting the Multi-ICE CONFIG jumper sets the mode for all the boards in the stack. Connect to the top module only.
- You cannot use multiple ARM Integrator core modules in the stack.
- The core module must have the appropriate interface controllers implemented in its FPGA image.

  The same core module hardware is used for operating standalone, with an Integrator/AP, or with the Integrator/CP. The FPGA image for use with the Integrator/CP is different from the one for standalone operation or use with an

Integrator/AP. ARM Integrator core modules select the FPGA image based on the **CFGSEL** signal, so the Integrator/CP compatible core modules can operate either standalone or with an Integrator/CP baseboard.

2.4.2 Expanding your system with your own modules

There are restrictions on how you can expand your system with your own modules:

- Some of the bus lines are reserved or used for special signals. Be careful that expansion modules do not use signals on the HDRA and HDRB that clash with signals carried between the core module and baseboard.
- Custom core modules cannot be added unless they have their own system bus and address space. That is, they must not use the bus signals on HDRA and HDRB. The module must have some dual-port memory, or some equivalent communication mechanism, in order to communicate with other modules. (Any additional custom core modules must operate in slave mode.)

  Caution

  Custom core module processors cannot use the header address or data lines. The custom module must appear as a memory region to the system.

- The appropriate connector type and spacing must be used.
- The new module must decode its own address space.
- Keep the current drawn by the system below the limits.

2.5 Using the ARM Firmware Suite boot monitor

The Integrator/CP is shipped with a boot monitor preprogrammed into the flash. For details, see the *ARM Firmware Suite Reference Guide* and the documentation supplied with your core module.

The monitor enables you to:

- load images into RAM and flash memory
- specify the flash image to boot
- run system self tests
- set system clock frequencies.

Use serial port A (J17 top) to communicate with the boot monitor. Connect a standard null modem cable between the system and a PC running a terminal emulator. The serial port settings are:

- 38400 baud
- no parity
- 8 bits
- 1 stop bit
- Xon/Xoff software handshaking.

2.5.1 Sample output

The actual output from the boot monitor depends on your core module. A sample output from an Integrator/CP with a CM920T is shown below.

```
boot Monitor > x dh
Core Modules
============
------ FPGA -------
CM Core       Arch  SSRAM  SDRAM  Bus   Type      Rev Build  Silicon ID
-- ----       ----  -----  -----  ---   ----      --- -----  ----------
0  ARM920       4T     1M   128M  ASB   XCV600     D    17      0x03

System
======
---- FPGA/PLD -----
Type  Endian        SSRAM  Flash  Bus   Type      Rev Build
----  ------        -----  -----  ---   ----      --- -----
CP    Either            0    16M  AHB   EPM7256AE  D    13

boot Monitor >
```

2.5.2 Possible effect on remap of using a debugger

When the Integrator system is powered up, the boot code area of flash is mapped at address `0x00000000`. Before you can access SSRAM at this address, you must remap the memory to replace the flash with the SSRAM. Occasionally however when using a debugger, the boot monitor does not run after power up and therefore does not remap the memory. (This might happen for example, if the debugger has set a breakpoint on the exception vectors.) If this happens, write to the appropriate core module register before loading code at address `0x00000000`.

2.5.3 Self-test program

The Integrator/CP is packaged with a CD that contains a suite of register level software tests for each of the Integrator/CP peripherals. The project and executable files for these tests are in *install*`\Platforms\CP\platform\software\selftest\build\ADS1.2\` (where *install* is the installation directory).

Note

The default location for the installation directory is `C:\Program Files\ARM\`. If another installation directory is used, replace *install* with your path.

If the default install directory is not used, then the executable file `selftest.axf` in *install*`\Platforms\CP\platform\software\selftest\build\ADS1.2\selftest_Data\Debug` must be rebuilt in order for the debugger to display the source code automatically.

If the project is built with the `Release` option, source symbols will not be included in the executable image file.

Running the basic tests requires a compatible color LCD screen, touch screen, stereo jack cable, serial cable, VGA monitor, and MultiMedia card. Running all tests requires custom loopback cables.

Connect the interfaces and load the `selftest.axf` image from *install*`\Platforms\CP\platform\software\selftest\build\ADS1.2\selftest_Data\Debug` into the debugger.

The following tests are available:

- AACI (Audio) requires a cable with two 3.5mm to 3.5mm stereo jack plugs to connect line level out to aux in.
- MMCI (MultiMedia card)
- CLCD (Color LCD)
- VGA
- LAN (Ethernet) requires a special RJ45/UTP Ethernet loopback cable

- UART A/B Interface, requires a three-wire null modem cable
- LEDs
- GPIO, requires a special loopback cable
- Keyboard, use either keyboard or loopback cable
- Mouse, use either mouse or loopback cable
- CLCD Bias, requires a voltmeter.

The source code for the tests are in *install*\Platforms\CP\platform\software\selftest\.

The selftest.mcp project file is in *install*\Platforms\CP\platform\software\selftest\build\ads1.2\.

Generic Integrator platform support files are in *install*\Platforms\CP\platform\software\selftest\Integrator_Library\.

The Integrator library project must have been built before the test suite is built. There is a prebuilt image on the CD. Details on constructing special loopback cables is provided on the CD.

2.5.4 PrimeCell Software

Included in the *install*\Platforms\CP\PrimeCell_drivers directory are generic PrimeCell driver source files and the test harness project file primecell.mcp is in *install*\Platforms\CP\PrimeCell_drivers\build\ads1.2\.

The Integrator/CP specific configuration files applatfm.h and apintcfg.h are in *install*\Platforms\CP\PrimeCell_drivers\apos\. The test instantiates the PrimeCell drivers and performs top level functionality tests. A MultiMedia Card (flash memory card) is required to complete all tests. PrimeCell drivers are provided for inclusion into target application software.

Note

Some MultiMedia cards do not support whole card erase, so this test might also fail.

2.5.5 System information block

In the *top sector of the flash memory on the baseboard is a System Information Block* (SIB) that holds the board-specific setup values. On the Integrator, the SIB is primarily used to store clock frequencies and program them when images are executed. If the program switches are set to run from boot monitor (S1[1]=ON and S1[4]=OFF), the clock speeds will be set from the SIB.

By default, the Integrator/CP powers up with a set of frequencies that are suitable for a wide range of processor cores. Use the boot monitor `DC` command to list the default frequencies. Use the `SC` command to change the SIB settings. Use the `CC` command to change the board frequencies immediately to the SIB settings.

Chapter 3
System Architecture

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

3.1 About the system architecture

Figure 3-1 shows the architecture of the Integrator/CP baseboard together with an ARM core module.

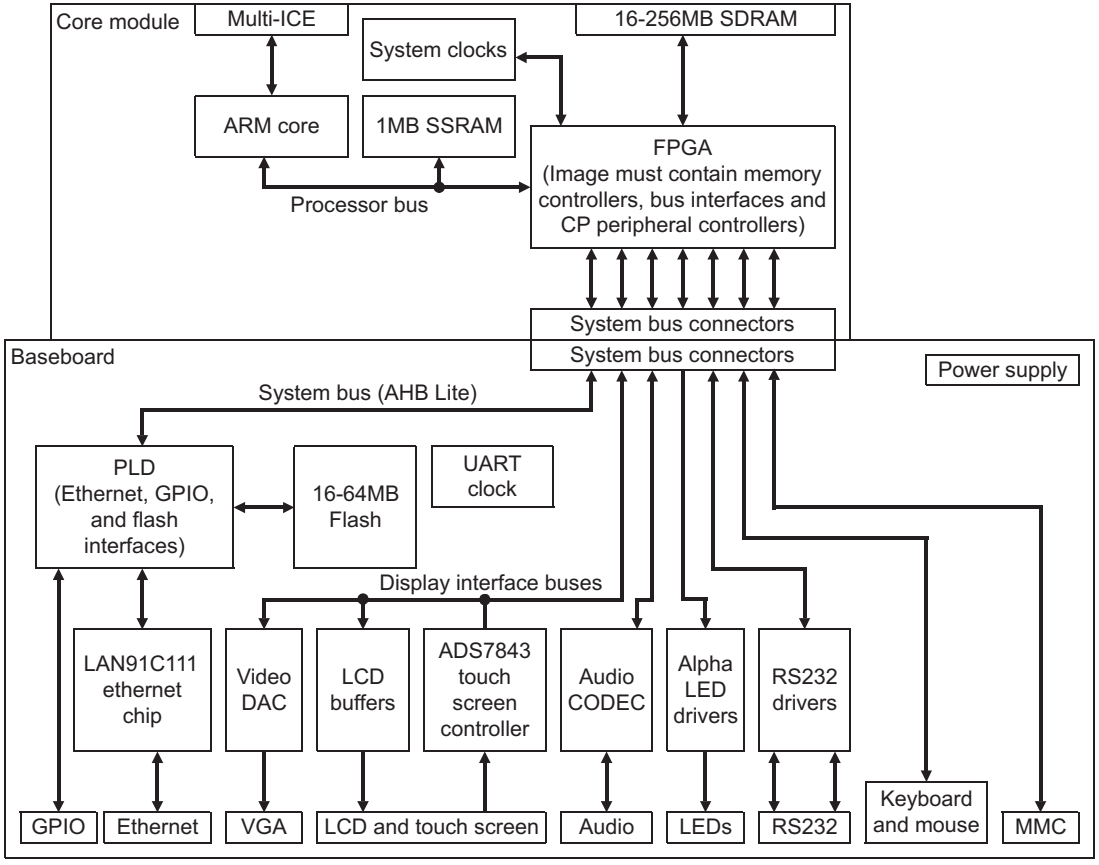


Figure 3-1 Integrator/CP system architecture

3.2 Baseboard AHB bus

The connections between the external busses and the Integrator/CP baseboard is described in the following subsections:

- *AHB-Lite bus protocol*
- *Bus connectors HDRA and HDRB* on page 3-4
- *Module-assigned signals* on page 3-6.

3.2.1 AHB-Lite bus protocol

The main system bus uses the AHB-Lite protocol. This is a version of the AHB system bus aimed at single-master system designs. The ARM core is the only master permitted. The system bus allows the processor to access resources on the baseboard and on other modules.

The AHB-Lite implementation on the Integrator/CP has the following main features:

- address and data-read multiplexors are not required
- the data bus has a tristate data bus rather than separate read and write busses
- a bus arbiter is not required
- slave SPLIT and RETRY capabilities are not supported
- standard AHB slaves that do not use SPLIT or RETRY are supported without modification.

The core module is an AHB-Lite master and has a similar signal interface to a full AHB master but differs as follows:

- it does not support **HBUSREQx** and **HGRANTx**
- it does not require **HRESP[1]** because it does not support SPLIT or RETRY that are signaled using the **HRESP[1]** signal.

The LOCK signal is tied to ground on the Integrator/CP. Lock is normally used to indicate to a slave that no other transfer can occur while the core requires the locked access. For the Integrator/CP however, there is only one core module present.

Because a number of signal paths are registered, the bridge interface can insert BUSY wait states during a transfer. The burst-read transaction takes a minimum of 2 clock cycles (data with at least one wait state per unit transfer). For the burst-write transaction, the number of clock cycles depends upon whether the transfer can be buffered. Non-buffered writes take a minimum of 3 clock cycles (data and at least two inserted wait states per unit transfer). Buffered writes complete in a single cycle when the pipeline is filled

Note

Table B-1 on page B-3 provides a full listing of the external system bus signals. More details on the AHB bus can be found on the ARM website at `www.arm.com\arm\amba`.

3.2.2 Bus connectors HDRA and HDRB

The system bus is connected between the baseboard, core module card, and additional logic modules using the HDRA and HDRB connectors.

Caution

The Integrator/CP supports only one ARM Integrator core module.

Custom core modules can be added, but they must operate in slave mode and communicate with an ARM core module using shared memory.

The signals carried by these connectors are described in Table 3-1 and the pinouts for the base board and core module are shown in Appendix B *Connector Pinouts*.

Table 3-1 System bus connector signal assignments for AHB-Lite

Connector	Pins	Function
HDRA	**A[31:0]**	This is the AHB address bus.
	B[31:0]	These carry the display interface signals.
	C[31:0]	These carry AHB bus control signals and various interface signals.
	D[31:0]	This is the AHB data bus. This uses a bidirectional bus **HDATA** rather than the separate unidirectional buses **HWDATA** and **HRDATA** described in the AMBA specification.
HDRB	**E[31:0]**	This bus carries the system bus control signals. These are mainly the signals associated with the Integrator system, such as interrupt requests and clocks.There are no arbitration signal connections on the baseboard. The clock signals, interrupt signals, module ID, and module presence signals are routed so that they rotate up through the stack (see *Module-assigned signals* on page 3-6). These pins correspond with the pins labeled **H[31:0]** on logic modules.
	F[31:0]	This set of pins is used to implement a variety of interface connections.
	G[16:0]	This set of pins is used to implement the Multi-ICE/JTAG signals and FPGA configuration control signals. These pins correspond with the **J[16:0]** pins on logic modules.

Note

There are no free pins. Expansion must use the EXPIM connector on a logic module.

3.3 Module-assigned signals

A number of signals are routed between the HDRA and HDRB plug and socket on the core module and logic modules so that they rotate up through the stack. The signal rotation allows interrupt and memory control based on the cards position in the stack without the requirement for changing jumpers on a board if its position in the stack is changed.

The signals that are rotated are:

nIRQ[3:0] These are the interrupt request signals from the logic modules. They are described in *Interrupt controllers* on page 3-14.

nFIQ[3:0] These are the interrupt request signals from the logic modules. They are described in *Interrupt controllers* on page 3-14.

nPPRES[3:0]

These are the module presence signals from the logic modules. They indicate to the address decoder that a logic module has been added to the system and is responsible for generating bus responses for its own address space (see *Register overview* on page 3-25). On the logic modules these signals correspond to the **nEPRES[3:0]** signals. On the Integrator/CP, three logic modules can be added, but only one core module is permitted. See *Logic module expansion memory space* on page 3-13.

SYSCLK[3:0]

These are the system bus clock signals rotated up through the stack to ensure even distribution and signal loading.

ID[3:0] The signals rotate up through the stack to indicate the position of a board in the stack of modules. The signals are only used by optional logic modules. Only one core module can be used and must always at the bottom of the stack.

3.4 Programmable logic

There are two main programmable logic parts on the Integrator/CP system. These are:

- *Integrator/CP baseboard system-support PLD*
- *Core module system controller FPGA* on page 3-8.

Note

Other programmable devices on the Integrator/CP system are:

- The SSRAM controller PLD (on the core module) is not used and contains a blank design)
- The Bitstreamer PLD (on the core module) used to load FPGA programming
- The alphanumeric LED PLD (normally on baseboard, but not present on the Rev A board).

3.4.1 Integrator/CP baseboard system-support PLD

The system support PLD provides AHB slave interfaces for the Ethernet and flash, an interrupt controller, and an 8-bit GPIO interface. Figure 3-2 on page 3-8 shows the internal architecture of the programmed PLD.

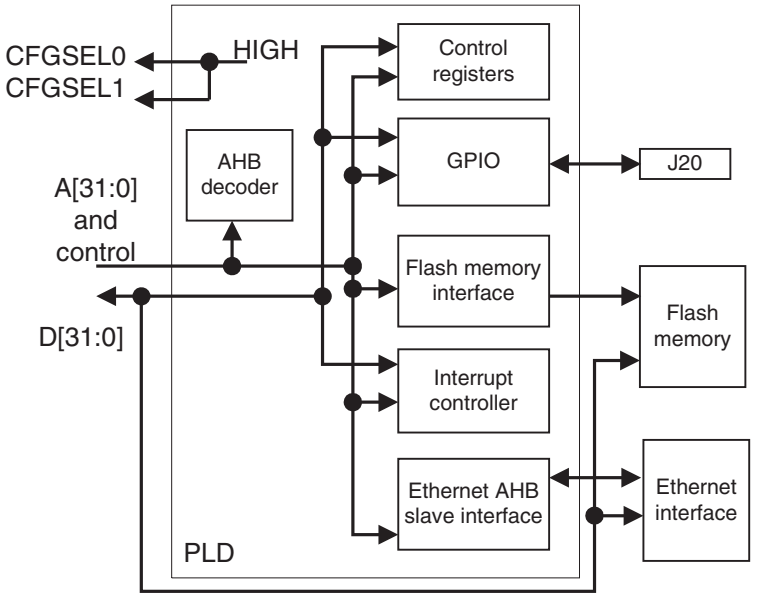


Figure 3-2 Baseboard PLD

The PLD is programmed during board manufacture and is not normally reprogrammed in the field. However, if required for system upgrade, the PLD can be reprogrammed using the Multi-ICE interface. This requires the progcards utility (version 2.30 or later).

3.4.2 Core module system controller FPGA

The system controller FPGA for the Integrator/CP is located on the core module. It contains the main bus bridges, memory controllers, and peripheral controllers.

Note

The core module FPGA must have an appropriate Integrator/CP-compatible image in order to control the peripherals on the Integrator/CP baseboard.

FPGA image selection

You can use Multi-ICE to reprogram the PLD, FPGA, and flash when the system is placed in configuration mode.

The configuration flash on the core module can store up to four images for the system controller FPGA. The images are selected by the **CFGSEL[1:0]** signals from the baseboard as shown in Table 3-2. For the Integrator/CP, the CP-AHB-Lite image is used.

Table 3-2 Image selection

CFGSEL[1]	CFGSEL[0]	Image
0	0	ASB
0	1	Reserved
1	0	AHB
1	1	CP-AHB-Lite

Note

The Integrator/CP drives the **CFGSEL[1:0]** signals high. The values for **CFGSEL[1:0]** are preprogrammed into the baseboard PLD.

For more details on image selection, see the documentation supplied with your core module.

Peripheral bus

Figure 3-3 shows the APB peripherals. These are implemented using PrimeCell or ADK devices synthesized into the FPGA on the core module.

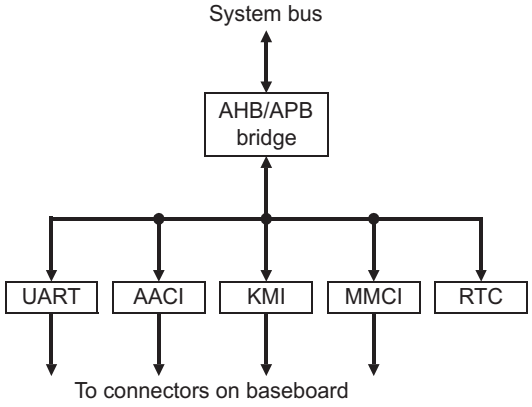


Figure 3-3 APB peripherals

The APB is an AMBA-compliant bus optimized for minimum power and reduced interface complexity. It is used to interface peripherals, such as the UARTs and the *Keyboard and Mouse Interface* (KMI), that do not require the high performance of the AHB.

The AHB-APB bridge is an AHB slave that provides an interface between the high-speed AHB domain and the low-power APB domain. Because the APB is not pipelined, wait states are added during transfers to and from the APB when the AHB is required to wait for the APB protocol.

3.5 Flash memory

The baseboard can be built with 16, 32, or 64MB of flash using 64 or 128Mb devices. (Typically, 16MB of flash is used.) Regardless of the size of flash memory, the top 256KB of the flash device is reserved for the system boot code. The remaining flash memory is available for your own code requirements.

The physical address of the boot code varies according to the amount of flash fitted as shown in Figure 3-4. The true start of flash is `0x24000000` and the flash extends to the size of the device fitted. For convenience, and to maintain compatibility with previous Integrator systems, a boot code alias region is provided at `0x20000000` and contains 256 aliased copies of the 256KB boot code area.

The bottom of the memory map is a configurable region. If the REMAP bit is 0, flash memory appears at `0x0`. See the documentation supplied with your core module for more information on REMAP.

Switch S2[1] selects boot or user code at `0x0`, see *Setting the DIP switches* on page 2-3.

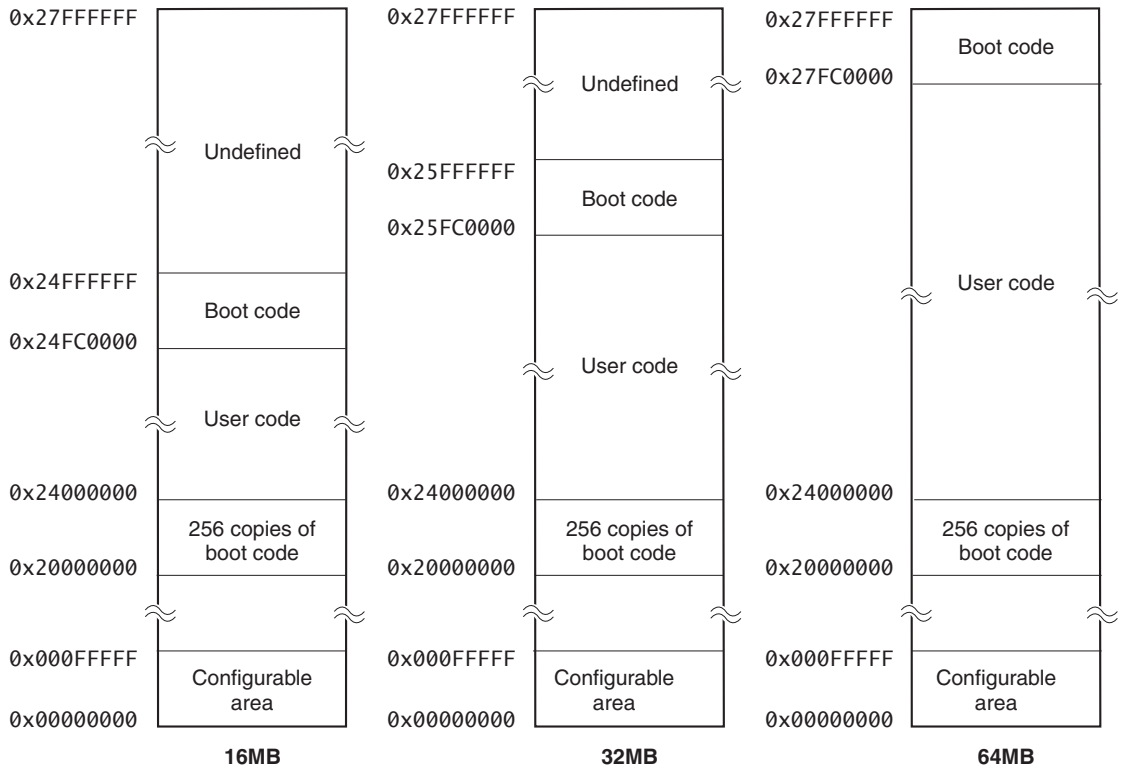


Figure 3-4 Flash memory mapping

Note

The copies of the boot code at 0x20000000 are aliases. The boot code can also be mapped to 0x00000000 by setting REMAP to 0 and setting SW2[1] to 1.

For more information on REMAP, see the documentation provided with the core module.

Table 3-3 shows the mapping of the flash memory for 16, 32, and 64MB configurations.

Table 3-3 Flash memory mapping

Memory	Address range	Content
16MB	0x00000000–0x000FFFFF	four copies of 0x24FC0000–0x24FFFFFF (REMAP= 0)
	0x20000000–0x23FFFFFF	256 copies of 0x24FC0000–0x24FFFFFF
	0x24000000–0x24FBFFFF	User code area
	0x24FC0000–0x24FFFFFF	Boot code area
	0x25000000–0x27FFFFFF	Not selected, reads return undefined values
32MB	0x00000000–0x000FFFFF	four copies of 0x25FC0000–0x25FFFFFF (REMAP= 0)
	0x20000000–0x23FFFFFF	256 copies of 0x25FC0000–0x25FFFFFF
	0x24000000–0x25FBFFFF	User code area
	0x25FC0000–0x25FFFFFF	Boot code area
	0x26000000–0x27FFFFFF	Not selected, reads return undefined values
64MB	0x00000000–0x000FFFFF	four copies of 0x27FC0000–0x27FFFFFF (REMAP= 0)
	0x20000000–0x23FFFFFF	256 copies of 0x27FC0000–0x27FFFFFF
	0x24000000–0x27FBFFFF	User code area
	0x27FC0000–0x27FFFFFF	Boot code area

The programmable logic device on the baseboard provides an AHB slave interface to the flash. Two links, EXTRABANK and FLASHSIZE are set at build time to indicate to the slave interface how many devices are fitted and their type (see Table 3-4).

Table 3-4 Flash configuration

Configuration	EXTRABANK	FLASHSIZE	Overall capacity
2 x 64Mb	0	0	16Mbyte
4 x 64Mb	1	0	32Mbyte
2 x 128Mb	0	1	32Mbyte
4 x 128Mb	1	1	64Mbyte

3.5.1 Logic module expansion memory space

The address decoder logic in the PLD provides a default slave response (**ERROR**) on the system bus for addresses allocated to any logic modules that are not present. In order to prevent the PLD from responding to addresses in the range `0xD0000000` to `0xFFFFFFFF`, the **nPPRES[3:0]** signals on the HDRB connector must be driven low. (**nPPRES[3:0]** are the names of the signal on HDRB, on the logic module itself, the signals are named **nEPRES[3:0]**.)

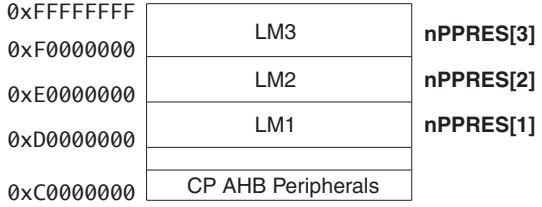


Figure 3-5 Memory space

Because of the rotation of these signals through the logic module stack however, modules are only required to drive **nEPRES[0]** and the appropriate line will be connected to the **nPPRES[3:0]** signals to the PLD depending on the position of module in the stack (see Figure 3-5).

Note

If the logic module is not present, the attempted memory access results in an abort. Some logic modules can be configured to drive more than one of the **nEPRES[3:0]** lines in order to claim address space that would normally be available for an absent module.

3.6 Interrupt controllers

There are three interrupt controllers for the Integrator/CP system:

- Primary interrupt controller (on core module)
- Secondary interrupt controller (on CP baseboard)
- Debug communications controller (on core module).

See the manual for your core module for more details of the primary and debug communications interrupt controllers. The secondary controller on the baseboard provides a set of registers to control and handle peripheral interrupts.

Figure 3-6 shows the interrupt control architecture for the Integrator/CP system.

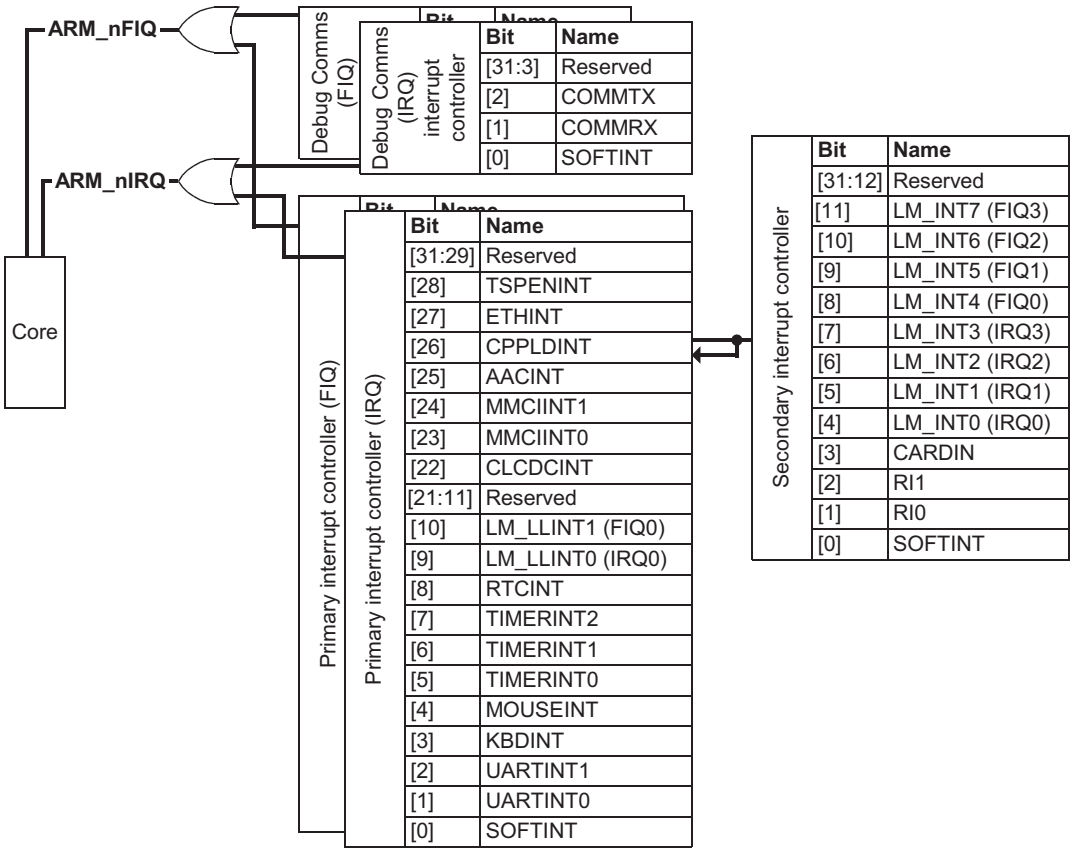


Figure 3-6 Interrupt architecture

The IRQ and FIQ controllers each provide three registers for controlling and handling interrupts. These are:

- status register
- raw status register
- enable register (accessed using the enable set and enable clear locations).

The way that the interrupt enable, clear, and status bits function for each interrupt is illustrated in Figure 3-7 and described in the following sections and the core module documentation. The illustration shows the control for one IRQ bit. The remaining IRQ bits and FIQ bits are controlled in a similar way.

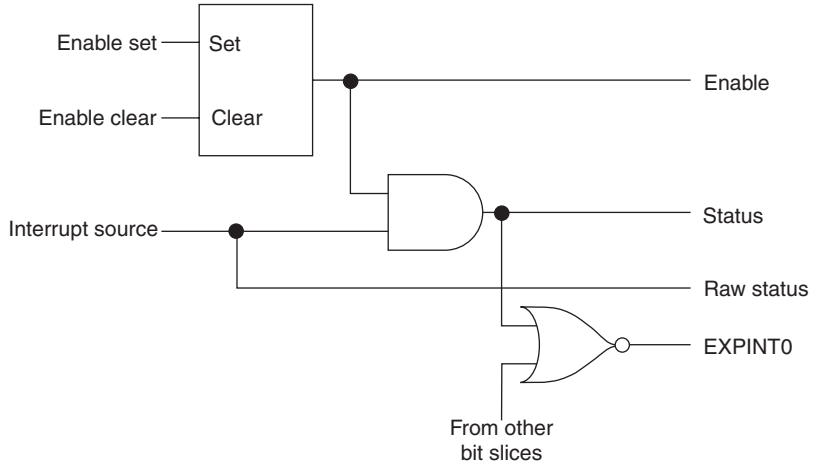


Figure 3-7 Interrupt control

3.6.1 Primary interrupt controller

The PIC is implemented within the system controller FPGA on the core module and handles the majority of interrupts from the system. A substantial number of interrupts are reserved, maintaining compatibility with other modules within the Integrator family.

There are separate controllers for IRQ and FIQ, allowing any interrupt source to be assigned to either. The PIC provides a set of registers to control and handle interrupts.

The Integrator/CP provides interrupt controllers for both IRQs and FIQs that maintain compatibility with other Integrator modules to ensure code portability. There are, however, some subtle differences (see Appendix A *Porting Integrator/AP and IM-PD1*).

An example of the primary interrupt control registers is provided in Table 3-5. The bit assignment for interrupts in the status, raw status, and enable registers for the IRQ and FIQ interrupt controllers is similar and is shown in Table 3-6.

Note

Refer to the documentation supplied with your core module for more information on the primary interrupt controller and registers.

Table 3-5 Example of primary interrupt register addresses

Address	Name	Type	Size	Function
0x14000000	PIC_IRQ_STATUS	Read	22	IRQ gated interrupt status
0x14000004	PIC_IRQ_RAWSTAT	Read	22	IRQ raw interrupt status
0x14000008	PIC_IRQ_ENABLESET	Read/write	22	IRQ enable set
0x1400000C	PIC_IRQ_ENABLECLR	Write	22	IRQ enable clear
0x14000010	PIC_INT_SOFTSET	Read/write	16	Software interrupt set
0x14000014	PIC_INT_SOFTCLR	Write	16	Software interrupt clear
0x14000020	PIC_FIQ_STATUS	Read	22	FIQ gated interrupt status
0x14000024	PIC_FIQ_RAWSTAT	Read	22	FIQ raw interrupt status
0x14000028	PIC_FIQ_ENABLESET	Read/write	22	FIQ enable set
0x1400002C	PIC_FIQ_ENABLECLR	Write-only	22	FIQ enable clear

Table 3-6 Primary interrupt register bit assignments

Bit	Name	Function
[31:29]	-	Reserved
[28]	TS_PENINT	Touchscreen pen-down event interrupt
[27]	ETH_INT	Ethernet interface interrupt
[26]	CPPLDINT	Interrupt from secondary interrupt controller, see *Secondary interrupt controller* on page 3-17.
[25]	AACIINT	Audio interface interrupt
[24]	MMCIINT1	MultiMedia card interface

Table 3-6 Primary interrupt register bit assignments (continued)

Bit	Name	Function
[23]	MMCIINT0	MultiMedia card interface
[22]	CLCDCINT	Display controller interrupt
[21:11]	-	Reserved
[10]	LM_LLINT1	Logic module low-latency interrupt 1
[9]	LM_LLINT0	Logic module low-latency interrupt 0
[8]	RTCINT	Real time clock interrupt
[7]	TIMERINT2	Counter-timer 2 interrupt
[6]	TIMERINT1	Counter-timer 1 interrupt
[5]	TIMERINT0	Counter-timer 0 interrupt
[4]	MOUSEINT	Mouse interrupt
[3]	KBDINT	Keyboard interrupt
[2]	UARTINT1	UART 1 interrupt
[1]	UARTINT0	UART 0 interrupt
[0]	SOFTINT	Software interrupt

3.6.2 Secondary interrupt controller

The SIC is implemented in the CP baseboard PLD and combines interrupts from MMC socket, the UART ring indicator bits, installed logic modules, and a software generated interrupt to the CPPLDINT input of the PIC.

The MMC interrupt on the SIC is generated by the card insertion detect switch and is different from the MMCI interrupts in the PIC generated by the MMCI PrimeCell. See *MMC interface* on page 4-27.

The secondary controller provides a set of registers to control and handle interrupts. The secondary interrupt control registers are listed in Table 3-7.

Table 3-7 Secondary interrupt register addresses

Address	Name	Type	Size	Function
0xCA000000	SIC_INT_STATUS	Read	22	SIC gated interrupt status
0xCA000004	SIC_INT_RAWSTAT	Read	22	SIC raw interrupt status
0xCA000008	SIC_INT_ENABLESET	Read/write	22	SIC enable set
0xCA00000C	SIC_INT_ENABLECLR	Write	22	SIC enable clear
0xCA000010	SIC_INT_SOFTSET	Read/write	16	Software interrupt set
0xCA000014	SIC_INT_SOFTCLR	Write	16	Software interrupt clear

The secondary interrupt controller bits are described in Table 3-8.

Table 3-8 Secondary interrupt register bit assignments

Bit	Name	Function
[31:12]	-	Reserved
[11:4]	LM_INT[7:0]	Interrupt sources from logic modules
[3]	CARDIN	Multimedia card insertion interrupt
[2]	RI1	UART1 ring indicator interrupt
[1]	RI0	UART0 ring indicator interrupt
[0]	SOFTINT	Software interrupt

3.6.3 Debug communications interrupts

The processor core incorporates EmbeddedICE hardware. This provides debug communications data read and write registers that are used to pass data between the processor and JTAG equipment. The processor accesses these registers as normal 32-bit read/write coprocessor registers and the JTAG equipment reads and writes the same registers using the relevant scan chain. For a description of the debug communications channel, see the *Technical Reference Manual* for your core.

You can use interrupts to signal when the debugger writes data into the read data register or reads from the write data register. These interrupts are supported by the CIC within the core module FPGA and can be enabled and cleared by accessing the interrupt registers.

3.6.4 Handling interrupts

This section describes interrupt handling and clearing in general. For examples of interrupt detection and handling, see the *ARM Firmware Suite User Guide*, the *ARM Developer Suite Developer Guide*, and the documentation supplied with your core module.

Enabling IRQ interrupts

The majority of peripheral interrupts are routed direct to the PIC. Each peripheral contains its own interrupt mask and clear registers. To enable interrupts, you must clear both the peripheral interrupt mask and the interrupt controller mask as well as clearing any previous interrupt flags:

1. Disable the primary interrupt by setting the appropriate bit in PIC_IRQ_ENCLR.
2. Clear the peripheral interrupt by setting the appropriate bit in the peripheral interrupt clear register.
3. Unmask the peripheral interrupt by clearing the appropriate bit in peripheral interrupt mask register.
4. Enable the primary interrupt by setting the appropriate bit in PIC_IRQ_ENSET.

The following C code stub demonstrates how the PIC UART0 CTS interrupt is cleared and re-enabled:

```
*PIC_IRQ_ENCLR    = PIC_UARTINT0;
*UART0_UARTICR    = UART_CTSINTR;
*UART0_UARTIMSC   &= ~UART_CTSINTR;
*PIC_IRQ_ENSET    |= PIC_UARTINT0;
```

The following C code stub demonstrates how the SIC MMCI CARDIN is cleared and re-enabled:

```
*PIC_IRQ_ENCLR   = PIC_CPPLDINT;
*CP_INTREG       = SIC_CARDIN;
*SIC_IRQ_ENSET   |= SIC_CARDIN;
*PIC_IRQ_ENSET   |= PIC_ CPPLDINT;
```

Note

The constants in these C code stubs must contain bit patterns necessary to set only the required interrupt mask bits. For example, `PIC_UARTINT0` must contain `0x02` to set only the UART0 bit in the PIC_IRQ_ENCLR register.

Determining and clearing IRQ interrupts

To determine an interrupt source, read the STATUS registers in the PIC and CIC to determine the interrupt controller that generated the interrupt. The sequence to determine and clear the interrupt is:

1. Determine the interrupt source by reading CM_IRQ_STATUS and PIC_STATUS.

   The interrupt handler is directed by the status register information to the particular peripheral that generated the interrupt. In the case of SIC interrupts the interrupt handler must also read the SIC STATUS register to determine the interrupt source.

2. Determine the peripheral interrupt source by reading the peripheral masked interrupt status register.

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

The following pseudo code example demonstrates how the UART0 CTS interrupt is detected:

```
If CM_IRQ_STATUS flags set,
      . . .  CIC interrupt handler

If PIC_STATUS flags set,
      . . . PIC interrupt handler
     If PIC_CPPLDINT set,
           . . .  SIC interrupt handler
     If PIC_UARTINT0 set,
           . . . UART0 interrupt handler
           If UART0_UARTMIS, UART_CTSINTR flag set,
                   . . . act on interrupt then clear flag with UARTICR
     . . . Test other PIC flags
```

3.6.5 Interrupt routing between Integrator modules

Figure 3-8 on page 3-22 shows how the IRQ signals are routed from the logic modules down to the interrupt controllers. It highlights how the signals are routed so that, for example, the interrupt request from logic module 2 connects to **LM_INT2** (**IRQ[2]**) on

the baseboard. The same rerouting applies for logic module 1 and 3. The operation of the interrupts relies on the core module being mounted on the baseboard first with the logic modules on top.

To simplify the description, this document refers to the logic modules source interrupts to the SIC as **LM_INT[7:0]**. The other signal names used in this diagram are those that appear on the schematic diagrams for the different modules. The signal names are not relevant to the final destination (IRQ or FIQ) of the signal in the ARM core.

For maximum flexibility, you can connect the eight logic module interrupt lines (**LM_INT[7:0]**) to the logic module interrupts as appropriate sources for your system. The SIC allows any of the interrupt sources **LM_INT[7:0]** to activate the **CPPLDINT** input on the PIC.

Because the signals route through the SIC, determining the source of an interrupt might require interrogating first the primary and then the secondary controller. The time required for the extra instructions might cause a problem with interrupt latency in some situations. To improve latency, **FIQ[0]** and **IRQ[0]** (as **LM_LLINT[1:0]**) are also routed to the PIC as well.

Note

Ensure that the correct interrupt line is driven. See the routing pattern in Figure 3-8 on page 3-22.

Although the signal rotation scheme is identical, the logic module interrupt routing scheme used on the Integrator/CP is not the same as that used on the Integrator/AP. In an Integrator/AP system, the **nFIQ[3:0]** pins on a logic module are unused and each logic module typically outputs only one interrupt source on its **nIRQ[0]** line.

Modules produced for the Integrator/AP should work on the Integrator/CP without the need to re-assign the interrupt signal positions. Logic modules in positions 1, 2, and 3 appear on the Integrator/CP interrupt lines **LM_INT[1]** to **LM_INT[3]** and lines **LM_INT[7:4]** are not connected.

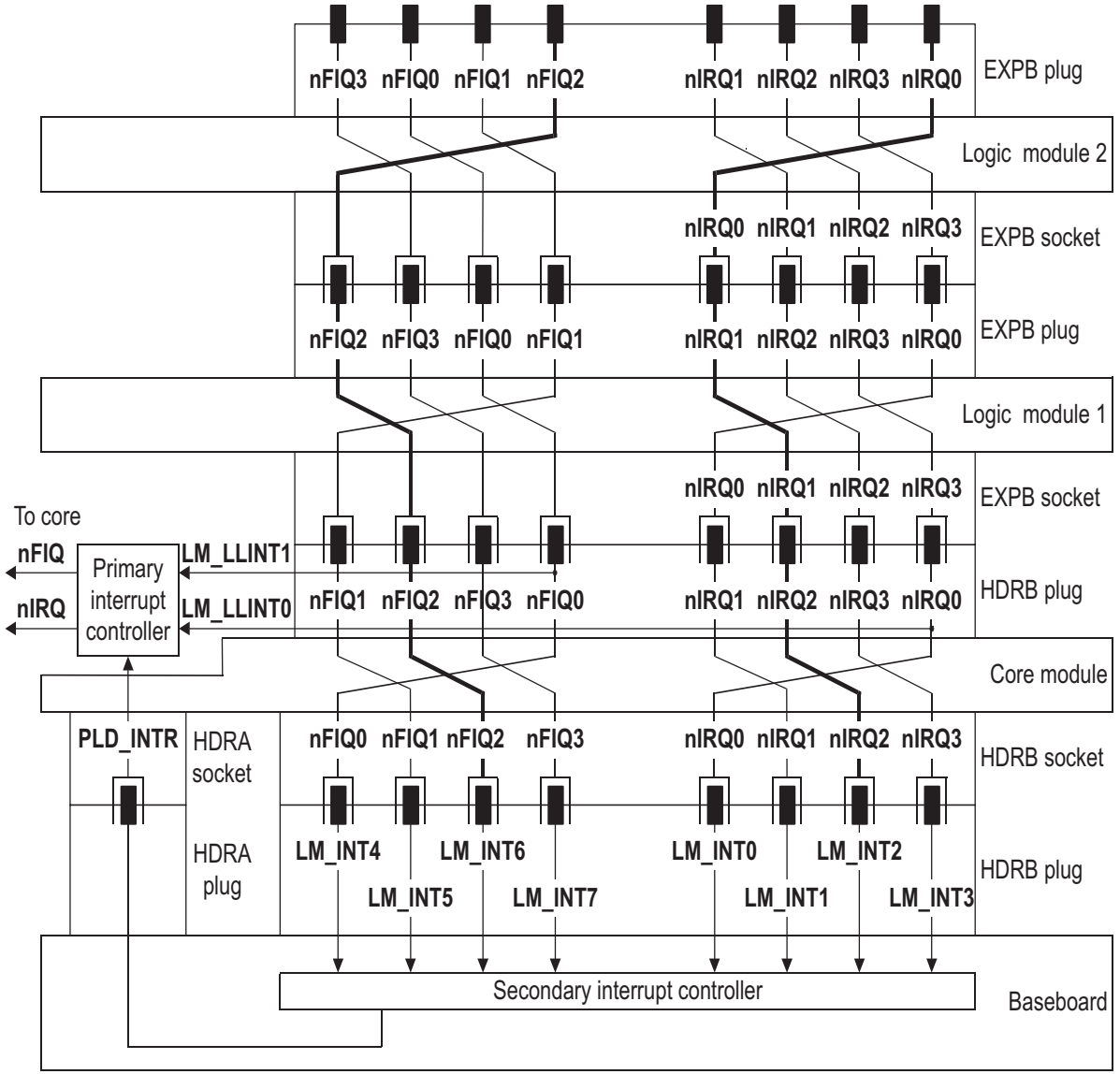


Figure 3-8 Interrupt signal routing

3.7 Clocks

Table 3-9 provides a summary of the clock signals on the CP baseboard.

Table 3-9 Clock signal descriptions

Signal name	Description
UARTCLK	This is a fixed-frequency 14.7456MHz clock generated by IC525 on the baseboard from a 25MHz crystal. It is supplied to the UARTs within the system controller FPGA to provide a 16x baud-rate clock, allowing baud rates of up to 460,800 to be selected in the UART registers. Although the UART is capable of baud rates up to 460kbps, the line drivers on the board are only guaranteed to operate to 120kbps
AACI_BIT_CLK	This is a fixed-frequency 12.288MHz clock generated by the audio CODEC from a 24.576MHz crystal on the baseboard. It also provides a reference clock for the AACI within the FPGA.
SYSCLK	This is the system bus clock. This clock has the same phase and frequency as the processor bus clock.
MCI_5	This clock signal is supplied by the MMCI in the system controller FPGA to the MMC socket on the baseboard. See *MMC interface* on page 4-27.

The clock architecture varies for each CP and core module combination (see the documentation for your core module for details).

3.7.1 Programmable clock control

The output clock frequencies from the programmable generators on the core module are set using registers instantiated into the core module FPGA.

For the CM9x0 core modules, use the CM_OSC register to set the operating frequency of the system bus clock. For the CM9x6 core modules, use the CM_INIT register to set the operating frequency of the system bus clock. For details on setting the clock frequencies, see the documentation supplied with the core module.

The Integrator/CP baseboard provides an additional IC525 clock generator to supply a UART clock. This is supplied with a 25MHz reference clock and has its divider inputs tied to provide a fixed frequency output of 14.7456MHz.

3.8 Configuring little or big-endian operation

By default, the Integrator/CP operates as a little-endian system. To change to big-endian operation, write to the appropriate register in CP15 on the ARM core.

There is a delay between changing the endian configuration of the core and the system functioning in the new endian mode. Changing endianess should only be done at the start of a debugging session. The change must have taken effect before you can perform any subword accesses.

Note

The supplied boot monitor, Ethernet controller, and LCD subsystem cannot operate in big-endian mode. For details on the interfaces, see Chapter 4 *Peripherals and Interfaces*.

3.9 Register overview

This section gives a overview of the register map for the entire Integrator/CP baseboard and core module system.

Note

The core module registers are included for information only. Refer to the documentation supplied with your core module for details on core module register addresses.

3.9.1 Register overview

Table 3-10 shows the map for an Integrator/CP baseboard and core module system.

Table 3-10 System control register map

Peripheral	Address range	Size	See
Core Module control registers	0x10000000–0x1000003F	64bytes	The documentation provided with your core module
Core Module interrupt controller	0x10000040–0x1000007F	64bytes	Core module documentation
Reserved	0x10000080–0x100000FF	128bytes	Core module documentation
Serial Presence Detect memory	0x10000100–0x100001FF	256bytes	Core module documentation
Counter/timers	0x13000000–0x13FFFFFF	16MB	*Counter/timer interface* on page 4-42
Primary interrupt controller registers	0x14000000–0x14FFFFFF	16MB	Core module documentation
Real-time clock	0x15000000–0x15FFFFFF	16MB	Core module documentation
UART0	0x16000000–0x16FFFFFF	16MB	*UART interface* on page 4-37
UART1	0x17000000–0x17FFFFFF	16MB	
Keyboard	0x18000000–0x18FFFFFF	16MB	*Keyboard and mouse interface* on page 4-33
Mouse	0x19000000–0x19FFFFFF	16MB	
Debug LEDs and DIP switch	0x1A000000–0x1AFFFFFF	16MB	*CP control registers* on page 3-26
Reserved	0x1B000000–0x1BFFFFFF	16MB	Core module documentation
Multimedia Card Interface	0x1C000000–0x1CFFFFFF	16MB	*MMC interface* on page 4-27
Advanced Audio CODEC Interface	0x1D000000–0x1DFFFFFF	16MB	*Audio interface* on page 4-20

Table 3-10 System control register map (continued)

Peripheral	Address range	Size	See
Touch Screen Controller Interface	0x1E000000–0x1EFFFFFF	16MB	*Touchscreen controller interface* on page 4-14
CLCD regs/palette	0xC0000000–0xC0FFFFFF	16MB	*LCD interfaces* on page 4-6
Ethernet	0xC8000000–0xC8FFFFFF	16MB	*Ethernet interface* on page 4-4
GPIO	0xC9000000–0xC9FFFFFF	16MB	*GPIO interface* on page 4-2
Secondary interrupt controller	0xCA000000–0xCAFFFFFF	16MB	*Secondary interrupt controller* on page 3-17
CP control registers	0xCB000000–0xCBFFFFFF	16MB	*CP control registers*

Note

Device registers are usually mapped repeatedly to fill their assigned spaces. However, to ensure correct operation on future product versions, it is advisable to only access the register at its true address.

3.9.2 CP control registers

The CP control registers return information about the Integrator/CP configuration and are used to control media interrupts. The registers are listed in Table 3-11 For more detail, see the documentation for your core module.

Table 3-11 Core Module status, control, and interrupt registers

Register Name	Address	Access	Reset	Description
CP_IDFIELD	0xCB000000	Read	Static	CP build information
CP_FLASHPROG	0xCB000004	Read	Static	Flash devices
CP_INTREG	0xCB000008	Read/Write	POR	Interrupt control
CP_DECODE	0xCB00000C	Read/Write	Reset	Fitted logic modules

CP_IDFIELD

The CP_IDFIELD register at `0xCB000000` returns build information.

Table 3-12 describes the ID register bits.

Table 3-12 CM_ID register bit descriptions

Bits	Name	Access	Function
[31:24]	MAN	Read	Manufacturer: `0x41` = ARM
[23:16]	ARCH	Read	`0x03` AHB-Lite system bus, bi-endian
[15:12]	FPGA	Read	`0x4` = EPM7256AE (Altera PLD on Integrator/CP)
[11:4]	BUILD	Read	Build value
[3:0]	REV	Read	Release revision `0x3` = Rev D

CP_FLASHPROG

The CP_FLASHPROG register at `0xCB000004` is used to identify the number of flash devices fitted and to enable the flash.

Table 3-13 describes the register bits.

Table 3-13 CP_FLASHPROG register

Bits	Name	Access	Function
[31:4]	Reserved	Use read-modify-write to preserve value.	
[3]	EXTRABANK	Read	Number of devices: 0 = 2 devices 1 = 4 devices.
[2]	FLASHSIZE	Read	size of devices: 0 = 64MBit 1 = 128MBit.
[1]	FLWREN	Read/write	Enable writing to flash (**nWE**).
[0]	FLVPPEN	Read/write	Enable flash programming voltage Vpp.

CP_INTREG

The CP_INTREG at `0xCB000008` is used to acknowledge and control interrupts related to the MMCI and UART ring indicator.

Table 3-14 describes the register bits.

Table 3-14 CP_INTREG register

Bits	Name	Access	Function
[31:5]	Reserved	Use read-modify-write to preserve value.	
[3]	CARDINSERT	Write	Signals that a MMC has been inserted. Write 1 to acknowledge and clear the card inserted interrupt.
[2]	RI1	Write	Signals that UART1 ring indicator signal (RI1) has been asserted. Write 1 to acknowledge and clear.
[1]	RI0	Write	Signals that UART0 ring indicator signal (RI0) has been asserted. Write 1 to acknowledge and clear.
[0]	WPROT	Read	This is the MMC write-protection status. This does not generate an interrupt.

CP_DECODE

Bits 7 to 6 of the CP_DECODE register at `0xCB00000C` indicates whether logic modules are fitted.

Table 3-15 describes the register bits.

Table 3-15 CP_DECODE register

Bits	Name	Function
[31:8]	Reserved	Use read-modify-write to preserve value.
[7]	LM3	This bit indicates that logic module 3 is fitted.
[6]	LM2	This bit indicates that logic module 2 is fitted.
[5]	LM1	This bit indicates that logic module 1 is fitted.
[4:0]	Reserved	Returns value `0b10001`.

Chapter 4
Peripherals and Interfaces

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

4.1 GPIO interface

An 8-bit *General Purpose Input/Output* (GPIO) controller is incorporated into the PLD. The PLD provides an AHB-Lite interface to the GPIO.

4.1.1 About the GPIO

Bits [7:4] can be individually programmed as an input or an output, but bits [3:0] are output-only. Bits in the data register are set and cleared using the GPIO_DATASET and GPIO_DATACLR registers. Bits are read and written to using the GPIO_DATAIN and GPIO_DATAOUT locations.

4.1.2 GPIO registers

The GPIO provides general-purpose input and output signals that are connected to J20. Each GPIO line has a 10KΩ pullup resistors to 3.3V. The GPIO registers are shown in Table 4-1.

Table 4-1 GPIO register summary

Address	Name	Type	Size	Function
0xC9000000	GPIO_DATASET	Write	8	Data output set
	GPIO_DATAIN	Read	8	Read data input pins
0xC9000004	GPIO_DATACLR	Write	8	Data register output clear
	GPIO_DATAOUT	Read	8	Read data output pins
0xC9000008	GPIO_DIRN	Read/write	8	Data direction

Data output set register

The GPIO_DATASET location is used to set individual GPIO output bits as follows:

- 1 = SET the associated GPIO output bit
- 0 = leave the associated GPIO bit unchanged.

Read data input register

Read the current state of the GPIO pins from this location.

Data register output clear

The GPIO_DATACLR location is used to clear individual GPIO output bits as follows:

- 1 = CLEAR the associated GPIO output bit
- 0 = leave the associated GPIO bit unchanged.

Read data output pins

Read the current state of the GPIO output register bits from this location.

Data direction

The GPIO_DIRN location is used to set the direction of each GPIO pin as follows:

- 1 = pin is an output
- 0 = pin is an input (default).

Figure 4-1 shows the data direction control for one GPIO bit.

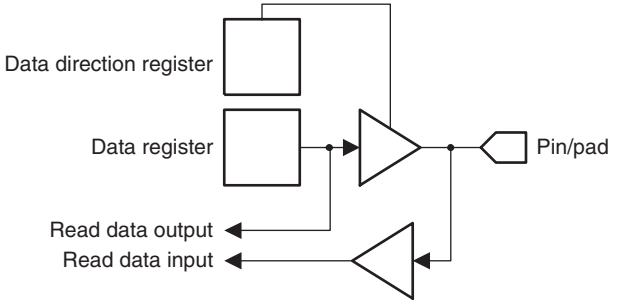


Figure 4-1 GPIO direction control (1 bit)

4.2 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 AHB Lite system bus by the PLD on the baseboard. The architecture of the Ethernet interface is shown in Figure 4-2.

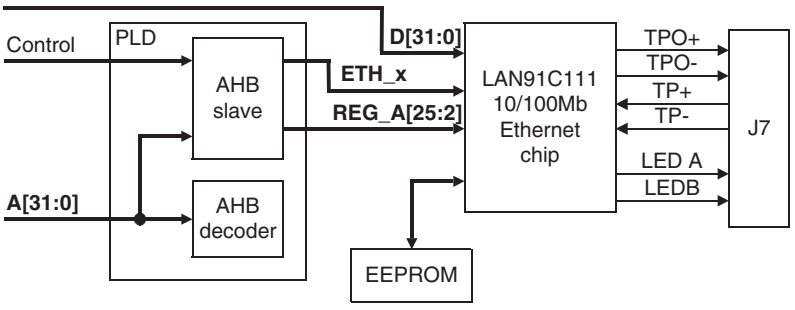


Figure 4-2 Ethernet interface architecture

J7 is an isolating RJ45 type of connector that incorporates two network status LEDs.The function of the LEDs can be set to indicate either link, activity, transmit, receive, full duplex, or 10/100 selection. See the data sheet for the LAN91C111 for more details on programming the registers.

4.2.1 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 autoconfiguration. When autoconfiguration 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 Lite bus provided by the PLD. The PLD 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 internal registers occupy 16 word locations. These are organized into four banks of 8 x 16-bit registers located at `0xC8000000` in the Integrator/CP memory map. Table 4-2 lists the internal registers.

Table 4-2 LAN91C111 internal registers

Offset	Bank 0	Bank 1	Bank 2	Bank 3
0	TCR	CONFIG	MMU COMMAND	MT0-1
2	EPH STATUS	BASE	PNR	MT2-3
4	RCR	IA0-1	FIFO PORT	MT4-5
6	COUNTER	IA2-3	POINTER	MT6-7
8	MIR	IA4-5	DATA	MGMT
A	RPCR	GENERAL	DATA	REVISION
C	RESERVED	CONTROL	INTERRUPT	ERCV
E	BANK	BANK	BANK	BANK

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 parameters to the LAN91C111 at reset. These parameters are:

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

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 desired. Reprogramming of the EEPROM is done through Bank 1 (general and control registers).

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

The LAN91C111 is provided with a 25MHz reference clock.

4.3 Display interface

The Integrator/CP provides a flexible display interface that provides support for two types of color LCD displays or a VGA display. These are described in the following subsections:

- *LCD interfaces*
- *LCD power control* on page 4-7
- *VGA display interface* on page 4-8
- *CM_CTRL and LCD control bits* on page 4-12.

The touchscreen controller that accompanies the LCD is described in *Touchscreen controller interface* on page 4-14. The selftest program supplied on the CD displays to the LCD or VGA monitor. See also the *ARM PrimeCell Color LCD Controller (PL110) Technical Reference Manual*.

4.3.1 LCD interfaces

Figure 4-3 shows the architecture of the CLD display interface and shows the signals used to provide pixel data and for buffer control.

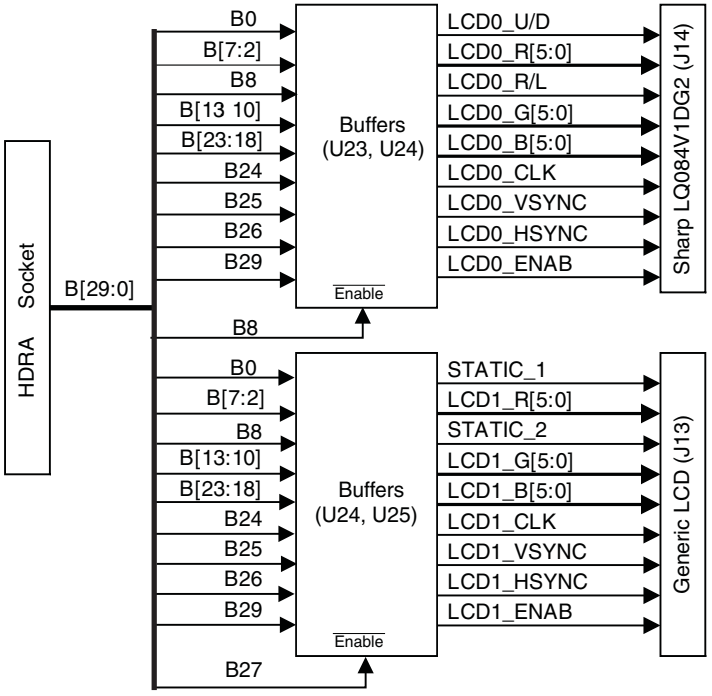


Figure 4-3 LCD interfaces

B27 is used to enable the signals on J13 (**LCD1_xxx**), and **B28** is used to enable the buffers for the Sharp display signals on J14 (**LCD0_xxx**). The enable LCD1 and enable LCD0 control bits are located in the CM_CTRL register at `0x1000000C`. See the manual for your core module for more details.

If you have a PrimeCell license and the HDL source, you can define the assignment of the generic LCD display interface signals to suit your particular application. However, Figure 4-3 on page 4-6 shows the default functional assignment.

4.3.2 LCD power control

Figure 4-4 shows the three Integrator/CP power outputs for the attached display:

- **LCD1_BIAS**
- **LCD1_3V3**
- **LCD0_3V3**.

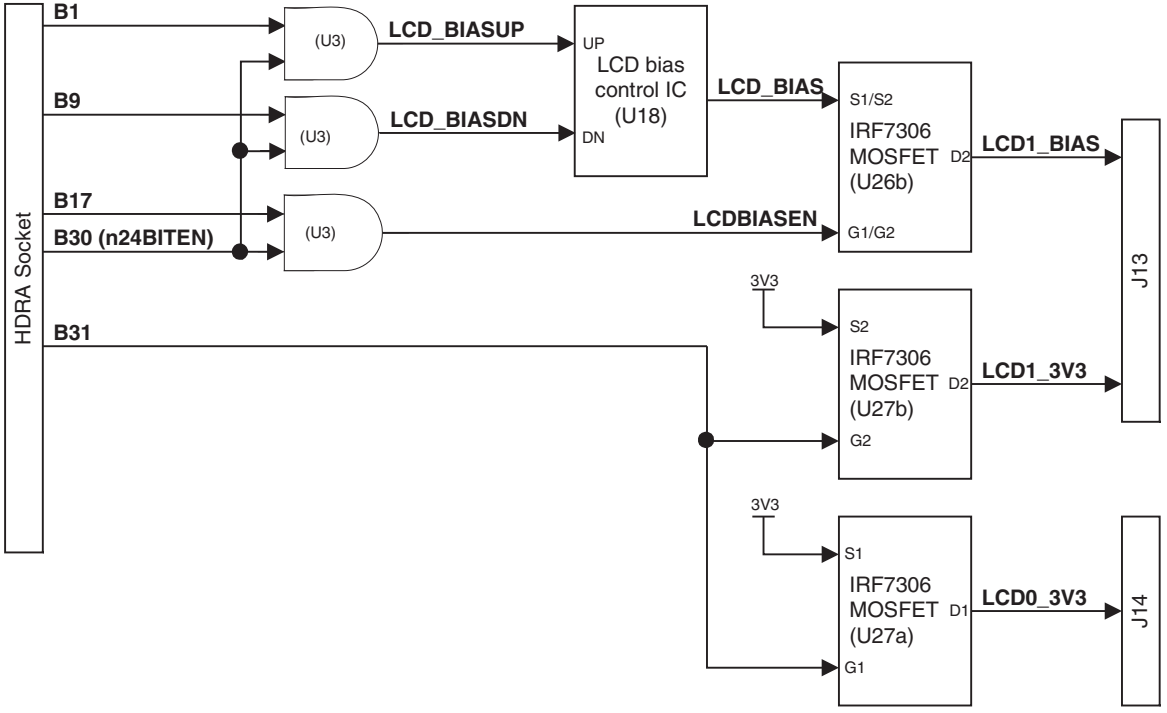


Figure 4-4 LCD power supply control

LCD1_BIAS is a variable supply controlled by the MAX686 (U6, DAC-controlled boost/Inverter LCD bias supply) and switched ON and OFF by the MOSFET switch (U26). **LCD1_BIAS** is varied between 11.5V and 24V in 64 steps using the inputs on pins B9 and B17 on the HDRA socket. These are edge-triggered inputs. The MAX686 is reset to the midpoint by a power-on reset. The MOSFET switch is controlled by the input from B1. The bias control signals are enabled when B30 is LOW and disabled when B30 is HIGH.

LCD0_3V3 and **LCD1_3V3** are fixed-level power outputs that are controlled by MOSFET switches within U27. The switches are controlled by the signal on the pin B31 of the HDRA connector.

The power-control signal **B31** is controlled by bit 11 in the LCD PrimeCell register LCD_CTRL.

The LCD bias controls are disabled when the VGA display interface is in 24-bit mode, when **n24BITEN**is LOW. **n24BITEN**must be HIGH in order to adjust the LCD bias controls.

The signals from the HDRA connector in Figure 4-4 on page 4-7 are controlled by the CM_CTRL register in the system controller FPGA:

B1 Bit 9 is LCDBIASUP

B9 Bit 10 is LCDBIASDN

B17 Bit 8 is LCDBIASEN

B30 Bit 19 is n24BITEN.

4.3.3 Backlight power

Connector J5 can be used to supply power to an inverter for a cold-cathode LCD backlight. The backlight supply pins VIN on J5 are supplied for the unregulated, fused power supply input to the CP. The backlight inverter must consume less than 5W.

In addition to voltage and ground pins, the connector also supplies the brightness adjustment voltage (0 to 2.5V) and shutdown logic signal (generated from the on/off switch). The brightness is adjusted by a variable resistor, R26, located near J5.

4.3.4 VGA display interface

The Integrator/CP provides a VGA display interface implemented with a THS8134A triple DAC as shown in Figure 4-5 on page 4-9. The video DAC can be configured for 24-bit (3x8 bit) or 16-bit RGB operation. The blanking and synchronization input signals are tied high to 3.3V. Pixel data and the horizontal and vertical synchronization signals are supplied by the display interface within the FPGA on the core module.

The 24-bit and 16-bit modes do not use the color palette. In 24-bit mode, color information is organized 8-8-8 bits. In 16-bit mode, there is 1 bit for intensity, then 5-5-5 bits for color.

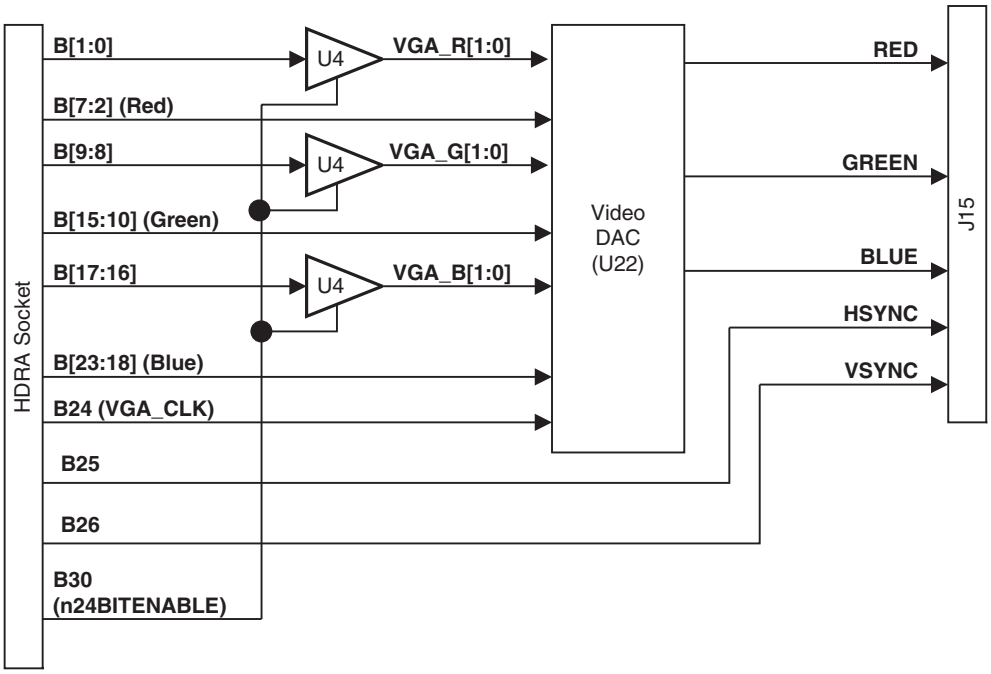


Figure 4-5 VGA interface architecture

The signal **n24BITENABLE** on B30 is used to enable the lower two bits on each of the three color channels. These are enabled when **n24BITENABLE** is LOW and disabled when **n24BITENABLE** is HIGH.

The timing for VGA and SVGA outputs are shown in Table 4-3.

Table 4-3 VGA and SVGA timing

Waveform	Part	VGA (640x480)	SVGA (800x600)
One line	Front porch	31 pixels	19 pixels
	Horizontal sync.	63 pixels	164 pixels
	Back porch	63 pixels	19 pixels
	Video	640 pixels	800 pixels
One field	Front porch	11 lines	5 lines
	Vertical sync.	24 lines	61 lines
	Back porch	8 lines	5 lines
	Video	480 lines	600 lines
Aux. Clock	Frequency	25MHz.	36MHz

4.3.5 Video frame buffer

The frame buffer is placed in SDRAM, for example at `0x00200000`. The amount of SDRAM used depends on the selected resolution and color depth.

4.3.6 Color LCD registers

This section describes the color LCD registers, for more detail see the *PrimeCell Color LCD controller (PL110).*

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

- locations `0xC0000030` through `0xC00001FC` are reserved for possible future extensions
- locations at offsets `0xC0000400` through `0xC00007FF` are reserved for test purposes.

The PrimeCell CLCDC registers are shown in Table 4-4.

Table 4-4 PrimeCell CLCDC register summary

Address	Type	Width	Reset value	Name	Description
0xC0000000	Read/write	32	0	LCDTiming0	Horizontal axis panel control
0xC0000004	Read/write	32	0	LCDTiming1	Vertical axis panel control
0xC0000008	Read/write	27	0	LCDTiming2	Clock and signal polarity control
0xC000000C	Read/write	17	0	LCDTiming3	Line end control
0xC0000010	Read/write	32	0	LCDUPBASE	Upper panel frame base address
0xC0000014	Read/write	32	0	LCDLPBASE	Lower panel frame base address
0xC0000018	Read/write	5	0	LCDINTRENABLE	Interrupt enable mask
0xC000001C	Read/write	16	0	LCDControl	LCD panel pixel parameters
0xC0000020	Read/write	5	0	LCDStatus	Raw interrupt status
0xC0000024	Read	5	0	LCDInterrupt	Final masked interrupts
0xC0000028	Read	32	X	LCDUPCURR	LCD upper panel current address value
0xC000002C	Read	32	X	LCDLPCURR	LCD lower panel current address value
0xC0000030 - 0xC00001FC	-	-	-	-	Reserved
0xC0000200 - 0xC00003FC	Read/write	32	-	LCDPalette	256 x 16-bit color palette

For VGA operation, perform the steps below and set the register values as shown in Table 4-5.

Table 4-5 Register values for VGA operation

Step number	Address	Value	Description
1	0x10000014	0xa05F	Write this value to the CM_LOCK register to enable changing the clock rate.
2	0x1000001C	0x12C11	Set AUXCLK (pixel clock) to 25MHz. This is the default value for AUXCLK.
3	0xC0000000	0x3F1F3F9C	Set the horizontal timing value in LCD_TIM0.
4	0xC0000004	0x080B61DF	Set the vertical timing value in LCD_TIM1.
5	0xC0000008	0x067F3800	Set other timing values in LCD_TIM2.
6	0xC0000010	FRAMEBASE	Set up the base address for the LCD frame buffer to, for example, 0x200000.
7	0xC000001C	0x1821 0x1823 0x1825 0x1827 0x1829 0x182B	Sets the bits per pixel: 1 2 4 8 16 24
8	0x1000000C	0x3e005 for 1,2,4,8, or 16 bits per pixel modes 0x33805 for 24 bits per pixel mode	Set the output multiplexor value in the CM_CTRL register to drive the VGA interface. Use read-modify-write to preserve the values of other bits (for example, REMAP) in the CM_CTRL register.

4.3.7 CM_CTRL and LCD control bits

In addition to the PrimeCell LCD registers, there are some LCD control bits in the CM_CTRL register at 0x1000000C that control:

- VGA color depth selection
- Up/down and left/right axis flip for Sharp LCD panels

- Buffer enable for **LCD0_xxx** on J14 and **LCD1_xxx** on J13
- Output multiplexor for PrimeCell CLDC
- LCD bias voltage control.

4.4 Touchscreen controller interface

The touchscreen interface is designed to connect to a 4-wire resistive touch screen. It is driven by the *TouchScreen Controller Interface* (TSCI) instantiated into the core module FPGA and is described in:

- *Touchscreen interface architecture*
- *Touchscreen controller interface registers* on page 4-17.

The selftest program supplied on the CD displays to the LCD or VGA monitor and receives input from the touchscreen.

4.4.1 Touchscreen interface architecture

Figure 4-6 shows the touchscreen interface. The signals to the touchscreen are routed to the 50-pin connector J13 and also to J16.

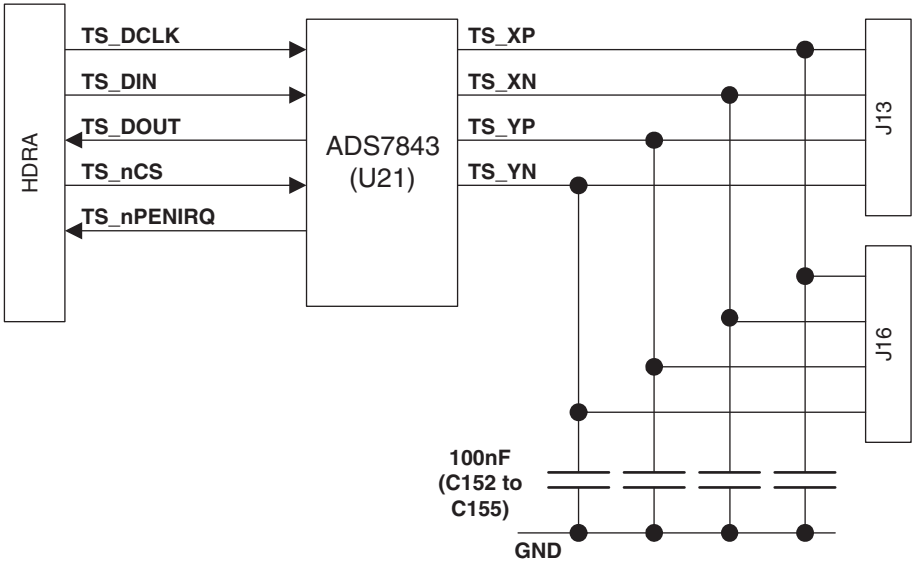


Figure 4-6 Touchscreen interface

Note

The values of C152 to C155 might require changing to match the characteristics of individual touch screens. The values fitted are satisfactory for most screen types.

The touchscreen interface uses an Analog Devices ADS7843 controller to provide an interface between a 4-wire resistive touch screen and the TSCI. The interface signals are shown in Table 4-1 on page 4-2.

Table 4-6 Touchscreen host interface signal assignment

Signal name	HDRB connector	Description
TS_DIN	F2	Serial data input to controller
TS_nCS	F1	Controller chip select
TS_DCLK	F0	Clock input to controller
TS_DOUT	F3	Data output from controller
TS_BUSY	Not connected	Busy indicator from controller
TS_nPENIRQ	F4	Interrupt from controller (drives the TSPENINT bit of the PIC)

Figure 4-7 shows the pinout of the touchscreen connector J16.

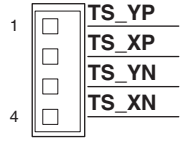


Figure 4-7 J16 pinout

The connection between J16 and the resistive elements of the touchscreen are shown in Figure 4-8 on page 4-16. 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.

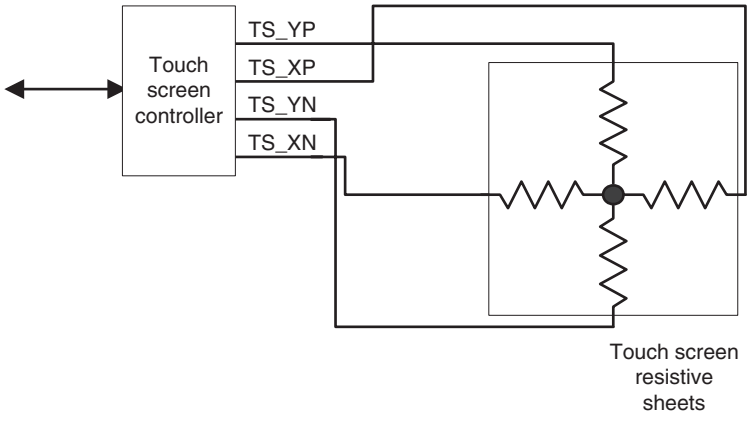


Figure 4-8 Touchscreen resistive elements

4.4.2 Touchscreen controller interface registers

The mapping of the TSCI registers is shown in Table 4-7.

Table 4-7 Touchscreen controller registers

Address	Name	Type	Function
0x1E000000	TS_CTRL	Read/write	Control register
0x1E000004	TS_DATA	Read	X Y data register
		Write	Write any value to acknowledge and clear the latched pendown interrupt

The interface consists of an 8-bit control register (write-only) and a 16-bit data read register (read-only). When a value is written to the control register, the controller starts a 24-bit transfer sequence:

- The first eight clocks write the control register value to the TSCI.
- Busy requires one clock.
- The next 12 clocks read data from the TSCI into the data register [11:0].
- The remaining three clocks complete the transfer.

The register bit mapping is shown in the tables below.

Table 4-8 TS_CTRL control register

Bit	Name	Function
7	S	Start bit and TSC/CS set low until after X2 read
6:4	A[2:0]	Enable Y (001) or X (101) plates for reading
3	MD	8/12 bit mode (always high, 12-bit resolution)
2	S/D	Single/Differential reference (always low, differential)
1:0	PD[1:0]	Power Down, use 11 for all readings except for last X reading that is 00.

Table 4-9 TS_DATA control register

Bit	Name	Function
15	RDY	The ready status bit indicates that the TSCI has completed processing of the last control value and is ready for a new value. While RDY is low, DATA[11:0] is invalid and the control byte must not be written.
14	PD	The controller PENIRQ generates an IRQ when the pen is down. The interrupt must not be cleared until the end of a complete pen reading sequence (the IRQ line is asserted several times by the reading process). At the end of each reading, PD is checked to see if the pen has been lifted. If so, the PENIRQ can be cleared ready for the next pen down event.
13:12	-	Not used
11:0	DATA[11:0]	Data indicating the X or Y position.

4.4.3 Pen reading sequence

Reading the pen position requires multiple operations:

1. Poll PD for pen down (or use interrupt PENIRQ).
2. Precharge Y (set /CS low).
3. Delay 1ms (delays are required to allow the touchscreen plate voltages to stabilize to an acceptable level). To avoid wasting time in delay loops, enable a 1ms tick timer at this point to service the touchscreen driver.
4. Read Y1 value. Precharge X.
5. Delay 1ms.
6. Read X1 value. Precharge Y.
7. Delay 1ms.
8. Read Y2 value. Precharge X.
9. Delay 1ms.
10. Read X2 value with power down. (XP is left charged). Set /CS high with low start bit.
11. Delay 1ms.

12. Check PD for pen up. If not pen up, check for large errors and average Y1, Y2 and X1, X2 values. If the pen is up, discard the readings and clear the interrupt by writing to TS_DATA.

    Note

    Always read X last to ensure correct operation of pendown detect.

13. After the pen up is detected, the 1ms tick timer can be disabled.

4.5 Audio interface

This section describes the audio interface in:

- *Hardware interface*
- *PrimeCell AACI functional overview* on page 4-21
- *AACI registers* on page 4-25.

4.5.1 Hardware interface

The baseboard provides a National Semiconductor LM4549 audio CODEC. The audio CODEC is compatible with AC'97 Rev 2.1 and features sample rate conversion and 3D sound. The CODEC is driven with a PrimeCell AACI (PL041) instantiated into the core module FPGA.

Figure 4-9 shows the architecture of audio interface. (Line out is the top connector.)

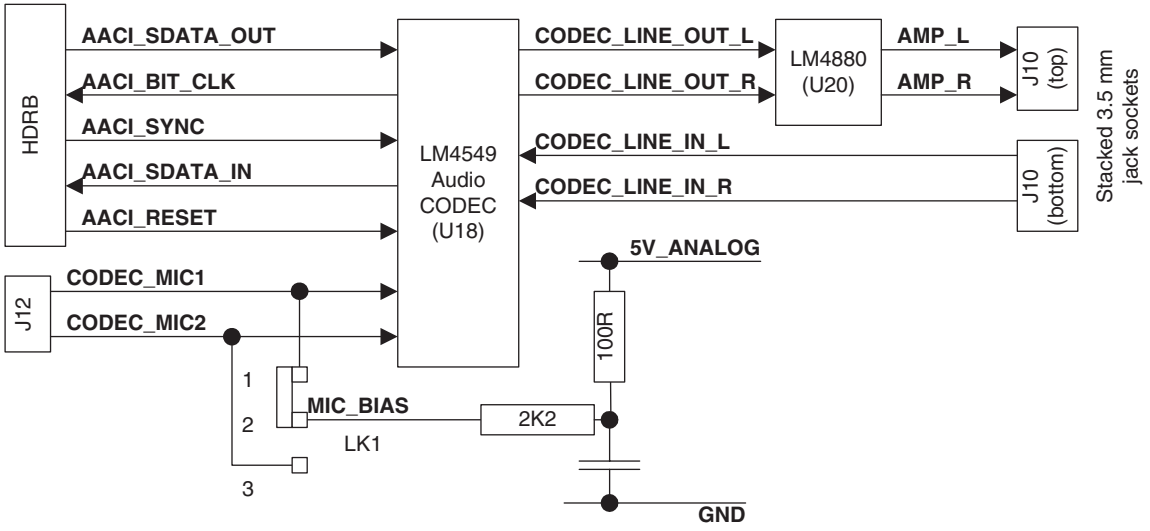


Figure 4-9 Audio interface

The CODEC is provided with a microphone input that can be configured by a link to support either passive or dynamic (electret) microphones:

- link 1 and 2 to support an active microphone with power on **MIC1** (tip)
- link 2 and 3 to support an active microphone with power on **MIC2** (ring 2)
- omit the link to support a passive microphone.

The default configuration supports a passive microphone. Only monophonic sound is supported, but microphone channel **MIC1** or **MIC2** can be selected in software.

The line output is provided with a 0.25W RMS unity-gain amplifier and enables you to connect 32Ω (or higher) headphones or speakers directly to J10.

The signals associated with the audio CODEC interface are assigned to the HDRB socket pins as shown in Table 4-10.

Table 4-10 Audio CODEC signal assignment

Signal name	HDRB	Description
AACI_SDATA_OUT	F7	Serial data from AACI to the CODEC
AACI_BIT_CLK	F6	Clock from the CODEC
AAC_SYNC	F8	Frame synchronization signal from the AACI
AACI_SDATA_IN	F5	Serial data from the CODEC to the AACI
AACI_RESET	F13	Reset signal from the PrimeCell AACI

Note

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

4.5.2 PrimeCell AACI functional overview

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

Figure 4-10 on page 4-22 shows a simplified block diagram of the PrimeCell AACI.

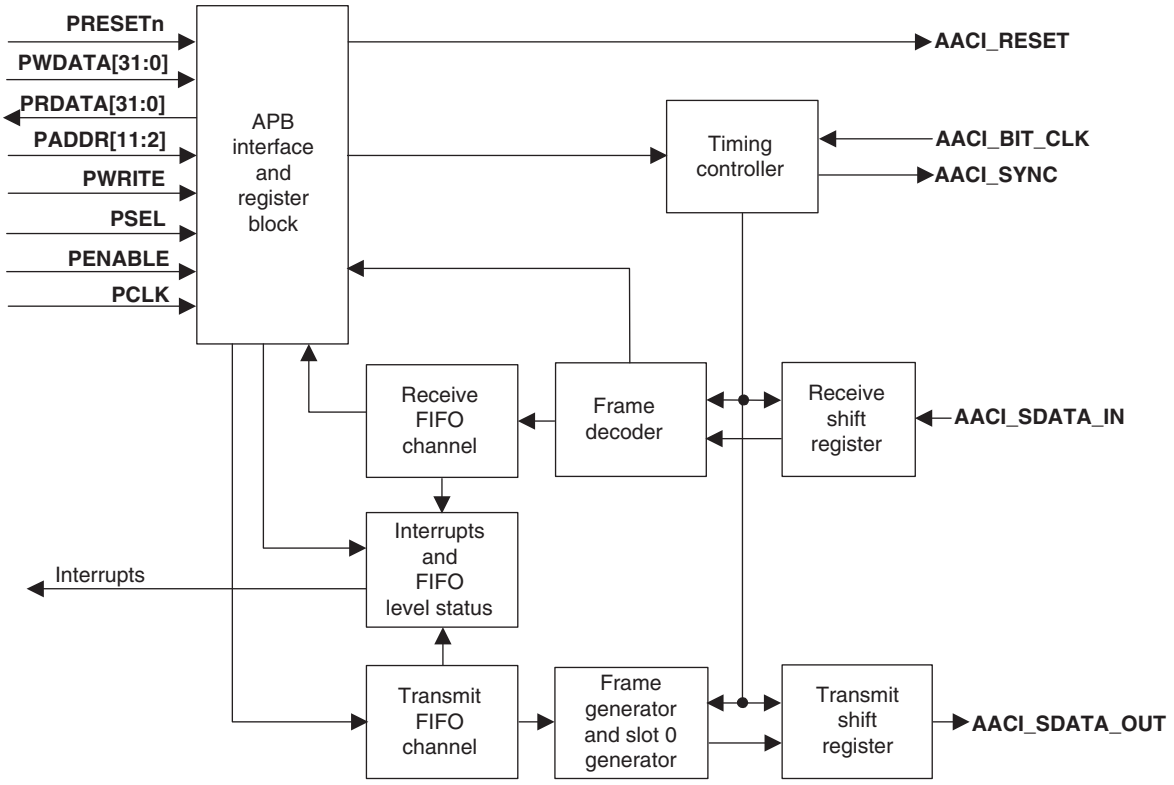


Figure 4-10 Simplified block diagram of the PrimeCell AACI

The AACI issues commands and audio data to the CODEC using the **AACI_SDATA_OUT** signal. It receives status and audio data from the CODEC using the **AACI_SDATA_IN** signal. Data is passed using 256-bit AC-link frames synchronized by **AACI_SYNC** and contained in two phases:

Tag phase The tag phase contains slot 0 that provides a 16-bit qualifier for the remaining slots in the frame.

Data phase The data phase contains 12 slots that provide commands or status and data to or from the audio channels in the CODEC. These slots are 20-bits long.

The start of an audio frame is signaled by the rising edge of **AACI_SYNC** signal that goes HIGH during the final bit of previous frame and remains HIGH for 16 **AACI_BIT_CLK** cycles.

The key features of communications using the AC-link are:

- all signals are synchronized to **AACI_BIT_CLK** running at 12.288MHz
- the start of a frame is signaled by the rising edge **AACI_SYNC**
- the **AACI_SYNC** signal is pulsed at the sample rate.

The AACI contains an 8-entry x 20-bit (standard) receive FIFO and an 8-entry x 20-bit transmit FIFO in which data phase slots are buffered.

For received data, each bit of a received frame on **AACI_SDATA_IN** is shifted into the receive shift register on the falling edge of **AACI_BIT_CLK**. Data is clocked out of the shift register at the end of each slot.

For transmitted data, each bit on **AACI_DATA_OUT** is clocked into the transmit shift register from the frame generator or slot 0 generator and is shifted out one bit at a time on the rising edge of **AACI_BIT_CLK**.

The slot 0 generator generates the slot 0 tag information and outputs it to the transmit shift register. The frame generator generates slots 1 to 12 and outputs them to the transmit shift register. (See Figure 4-11.)

The frame decoder qualifies the data of slots 1 to 12 that is output from the receive shift register, depending on data present in slot 0, and outputs it to the receive FIFO channel or slot receive registers.

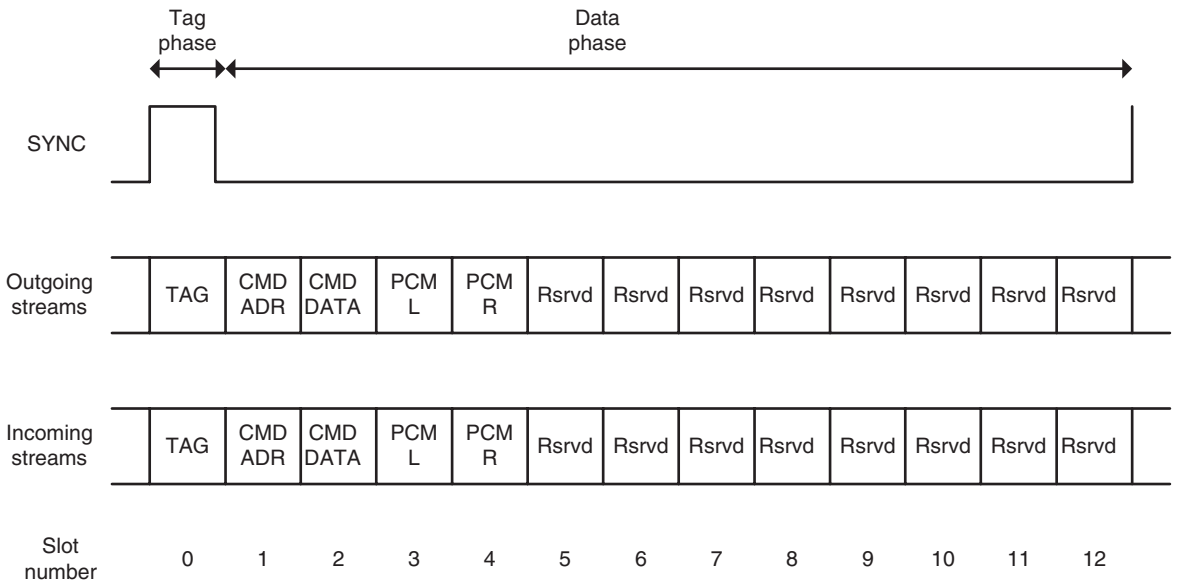


Figure 4-11 AC97 bidirectional audio frame

When the AC-link is active, the timing controller drives the **AACI_SYNC** signal based on the **AACI_BIT_CLK** signal from the off-chip CODEC. When the AC-link is inactive, the timing controller drives the internal register value on the **AACI_SYNC**.

Individual maskable, active HIGH interrupts are generated by the PrimeCell AACI and a combined interrupt output is also generated as an OR function of the individual interrupt requests.

The PrimeCell AACI uses two clock signals, the APB bus clock **PCLK** and **AACI_BIT_CLK**. The frequency of **AACI_BIT_CLK** is fixed at 12.288MHz.

4.5.3 AACI registers

The PrimeCell AACI registers are listed in Table 4-11. For a functional description of the registers, see *ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual*.

Table 4-11 PrimeCell AACI register summary

Address	Name	Type	Size	Description
0x1D000000	AACI_RXCR1	Read/write	29	Control register for receive FIFO1
0x1D000004	AACI_TXCR1	Read/write	17	Control register for transmit FIFO1
0x1D000008	AACI_SR1	Read	12	Status register channel 1
0x1D00000C	AACI_ISR1	Read	7	Interrupt status channel 1
0x1D000010	AACI_IE1	Read/write	7	Interrupt enable channel 1
0x1D000014–0x1D00004C				Reserved (unused AACI channels)
0x1D000050	AACI_SL1RX	Read	20	Data received on slot 1
0x1D000054	AACI_SL1TX	Read/write	20	Data transmitted on slot 1
0x1D000058	AACI_SL2RX	Read	20	Data received on slot 2
0x1D00005C	AACI_SL2TX	Read/write	20	Data transmitted on slot 2
0x1D000060	AACI_SL12RX	Read	20	Data received on slot 12
0x1D000064	AACI_SL12TX	Read/write	20	Data transmitted on slot 12
0x1D000068	AACI_SLFR	Read/write	14	Slot flag register
0x1D00006C	AACI_SLISTAT	Read	8	Slot interrupt status register
0x1D000070	AACI_SLIEN	Read/write	9	Slot interrupt enable register
0x1D000074	AACI_INTCLR	Write	13	Interrupt clear register
0x1D000078	AACI_MAINCR	Read/write	12	Main control register
0x1D00007C	AACI_RESET	Read/write	1	RESET control register
0x1D000080	AACI_SYNC	Read/write	1	SYNC control register
0x1D000084	AACI_ALLINTS	Read	28	All FIFO interrupt status register
0x1D000088	AACI_MAINFR	Read	2	Main flag register
0x1D000090–0x1D0000AC	AACI_DR1	Read/write	32	Data read or written, from or to FIFO1

Table 4-11 PrimeCell AACI register summary (continued)

Address	Name	Type	Size	Description
0x1D0000B0–0x1D00017F	-	-	-	Reserved
0x1D000180–0x1D00018C	-	-	-	Reserved for test purposes
0x1D000190–0x1D000FDC	-	-	-	Reserved
0x1D000FE0	AACI_PERIPHID0	Read	8	Identification register, bits [7:0]
0x1D000FE4	AACI_PERIPHID1	Read	8	Identification register, bits [15:8]
0x1D000FE8	AACI_PERIPHID2	Read	8	Identification register, bits [23:16]
0x1D000FEC	AACI_PERIPHID3	Read	8	Identification register bits, [31:24]
0x1D000FF0	AACI_PCELLID0	Read	8	PrimeCell identification register, bits [7:0]
0x1D000FF4	AACI_PCELLID1	Read	8	PrimeCell identification register, bits [15:8]
0x1D000FF8	AACI_PCELLID2	Read	8	PrimeCell identification register, bits [23:16]
0x1D000FFC	AACI_PCELLID3	Read	8	PrimeCell identification register, bits [31:24]

4.6 MMC interface

This section discusses the MMC interface in:

- *Card interface description*
- *PrimeCell MMCI functional overview* on page 4-30
- *MMCI registers* on page 4-31.

4.6.1 Card interface description

Figure 4-12 shows the card interface.

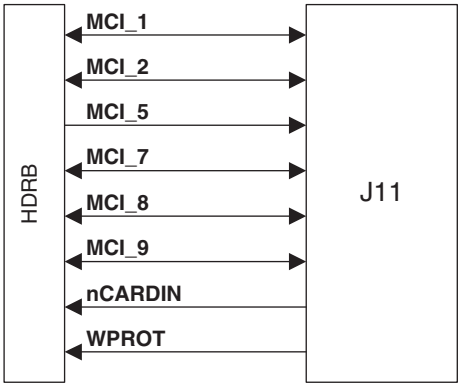


Figure 4-12 MMC interface description

Table 4-12 shows the assignment of the MMCI signals.

Table 4-12 MMC interface signals

Signal name	HDRB	Function
nMCI_ON	F15	Controls card power: LOW = power ON HIGH = power OFF
MCI_1	F14	Reserved
MCI_2	F10	CMD
MCI_5	F16	CLK
MCI_7	F11	DATA
MCI_8	F12	not used

Table 4-12 MMC interface signals (continued)

Signal name	HDRB	Function
MCI_9	F13	not used
nCARDIN	Routed to PLD	Card insertion interrupt
WPROT	Routed to PLD	Write protect status of MMC.

The MMC socket (J11) provides nine pins that connect with contacts on the card when it is inserted into the socket. Figure 4-13 shows the pin numbering and signal assignment, with pin 9 next to pin 1 and pins 7 and 8 spaced more closely together than the other pins. (The 9-way socket is compatible with SD cards, however MMC only uses 7 of the 9 pins.) In addition, 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.

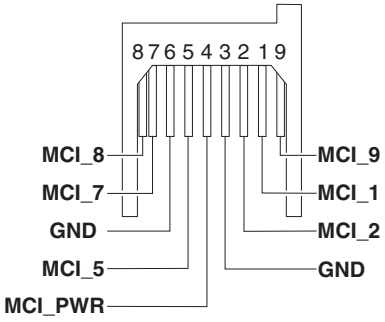


Figure 4-13 MMC card socket pin numbering

MMC cards use seven pins. Figure 4-14 shows an MMC card, with the contacts face up.

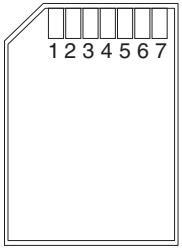


Figure 4-14 MMC card

Insert and remove the card as follows:

Insertion Insert the card into the socket with the contacts face down. Cards are normally labeled on the top surface with an arrow to indicate correct insertion.

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

The connector J9 enables you to access the signals for debugging or to route them to an off-PCB card socket. The pinout of J9 is shown in Figure 4-15.

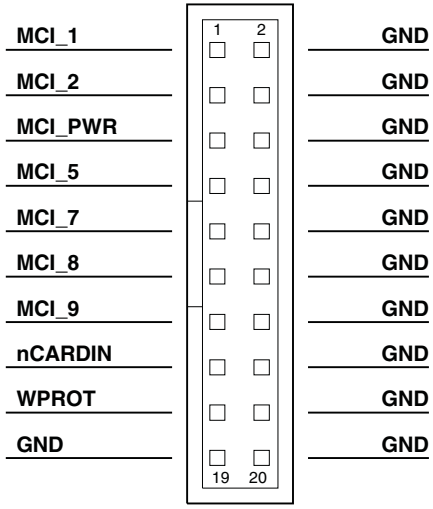


Figure 4-15 J9 pinout

4.6.2 PrimeCell MMCI functional overview

The PrimeCell *MultiMedia Card Interface* (MMCI) is an APB peripheral that provides an interface between the MMC and the APB. This section provides a brief overview of the MMCI. For detailed information, see the *MultiMedia Card Interface (PL181) Technical Reference Manual*.

Figure 4-16 shows a simplified block diagram of the PrimeCell MMCI.

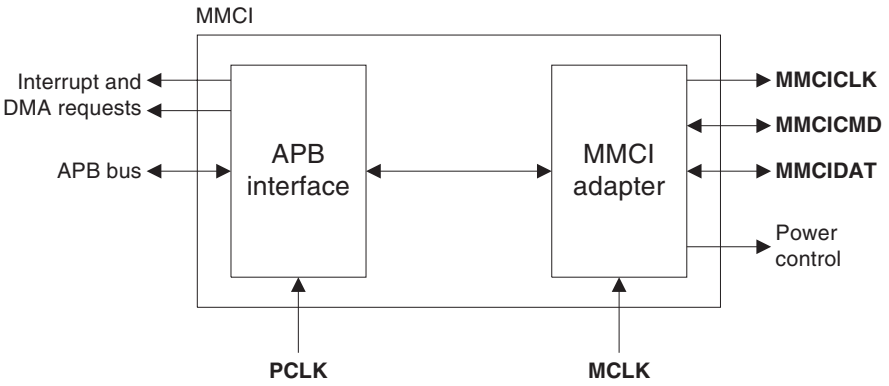


Figure 4-16 PrimeCell MMCI block diagram

The PrimeCell MMCI provides an interface between the APB and a multimedia card. It consists of two parts:

- The PrimeCell MMCI adapter block. This provides the functions specific to the multimedia card. This includes clock generation, power management, and command and data transfer.
- The APB interface accesses the PrimeCell MMCI adapter registers, and generates interrupt request signals.

4.6.3 MMCI registers

The PrimeCell MMCI registers listed shown in Table 4-13. For a functional description of the registers, see *ARM PrimeCell Multimedia Card Interface (PL181) Technical Reference Manual*.

Table 4-13 PrimeCell MMCI register summary

Address	Name	Type	Width	Description
0x1C000000	MMCI_Power	Read/write	8	Power control register
0x1C000004	MMCI_Clock	Read/write	11	Clock control register
0x1C000008	MMCI_Argument	Read/write	32	Argument register
0x1C00000C	MMCI_Command	Read/write	11	Command register
0x1C000010	MMCI_RespCmd	Read	6	Response command register
0x1C000014	MMCI_Response0	Read	32	Response register
0x1C000018	MMCI_Response1	Read	32	Response register
0x1C00001C	MMCI_Response2	Read	32	Response register
0x1C000020	MMCI_Response3	Read	31	Response register
0x1C000024	MMCI_DataTimer	Read/write	32	Data timer
0x1C000028	MMCI_DataLength	Read/write	16	Data length register
0x1C00002C	MMCI_DataCtrl	Read/write	8	Data control register
0x1C000030	MMCI_DataCnt	Read	16	Data counter
0x1C000034	MMCI_Status	Read	22	Status register
0x1C000038	MMCI_Clear	Write only	11	Clear register
0x1C00003C	MMCI_Mask0	Read/write	22	Interrupt 0 mask register
0x1C000040	MMCI_Mask1	Read/write	22	Interrupt 1 mask register
0x1C000044	Reserved	-	-	-
0x1C000048	MMCI_FifoCnt	Read	15	FIFO counter
0x1C00004C-0x07C	Reserved	-	-	-
0x1C000080-0x0BC	MMCI_FIFO	Read/write	32	Data FIFO register
0x1C000FE0	MMCI_PeriphID0	Read	8	Peripheral identification register bits 7:0

Table 4-13 PrimeCell MMCI register summary (continued)

Address	Name	Type	Width	Description
0x1C000FE4	MMCI_PeriphID1	Read	8	Peripheral identification register bits [15:8]
0x1C000FE8	MMCI_PeriphID2	Read	8	Peripheral identification register bits [23:16]
0x1C000FEC	MMCI_PeriphID3	Read	8	Peripheral identification register bits [31:24]
0x1C000FF0	MMCI_PCellID0	Read	8	PrimeCell identification register bits [7:0]
0x1C000FF4	MMCI_PCellID1	Read	8	PrimeCell identification register bits [15:8]
0x1C000FF8	MMCI_PCellID2	Read	8	PrimeCell identification register bits [23:16]
0x1C000FFC	MMCI_PCellID3	Read	8	PrimeCell identification register bits [31:24]

4.7 Keyboard and mouse interface

The *Keyboard and Mouse Interfaces* (KMI) are described in:

- *About the KMI interface*
- *KMI functional overview* on page 4-34
- *KMI interrupts* on page 4-34
- *KMI registers* on page 4-35.

4.7.1 About the KMI interface

The keyboard and mouse controllers are implemented with two PrimeCell KMIs that are incorporated into the system controller FPGA. This is shown in Figure 4-17.

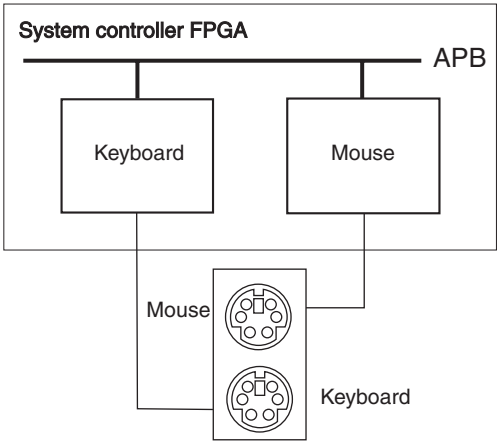


Figure 4-17 KMI block diagram

Each KMI contains the following functional blocks:

- APB interface and register block
- transmit block
- receive block
- controller block
- timer/clock divider blocks
- synchronization logic.

4.7.2 KMI functional overview

This section provides a functional overview of the *Keyboard and Mouse Interface* (KMI). For detailed information about the KMI, see *KMI (PL050) Technical Reference Manual*.

The APB interface and register block provides the interface with the APB and control registers. All control register bits are synchronized to the main clock for KMI logic before they are used in the KMI core. This block also generates individual transmit and receive interrupts.

The transmit block converts the parallel transmit data into a serial bit stream with a rate dependent on the incoming KMI clock signal. This block performs odd parity generation, data framing, and parallel-to-serial conversion. This block operates on **KMIREFCLK** (that on the baseboard is supplied by **CLK24MHz**) with the incoming clock signal **KMICLKIN** providing the bit-rate information. The data is shifted out on the falling edge of the **KMICLKIN** input.

The receive block performs serial-to-parallel conversion on the serial data stream received on the **KMIDATAIN** input pin. The falling edge on the synchronized and sampled **KMICLKIN** input signal is used to sample the **KMIDATAIN** input line. The **KMIDATAIN** input is synchronized to the **KMIREFCLK** clock.

The controller controls transmit and receive operations. If simultaneous requests for transmission and reception occur, the transmit request is given priority.

4.7.3 KMI interrupts

The KMI generates the following interrupts:

Transmit interrupt

This is asserted to indicate that a byte can be written to the data register KMIDATA for transmission.

Receive interrupt

This asserted to indicate that a byte has been received and can be read from the data register KMIDATA.

A combined interrupt is also asserted if either the transmit or receive interrupt is asserted. This combined interrupt is used by the system interrupt controller.

4.7.4 KMI registers

The KMI registers are summarized in Table 4-14. For more detailed information, refer to the *KMI (PL050) Technical Reference Manual*.

Table 4-14 KMI register summary

KeyboardAddress	MouseAddress	Name	Type	Description
0x18000000	0x19000000	KMI_CR	Read/write	Control register
0x18000004	0x19000004	KMI_STAT	Read	Status register
0x18000008	0x19000008	KMI_DATA	Read	Received data register, bits [7:0]
			Write	Transmit data register, bits [7:0]
0x18000010	0x19000010	KMI_IR	Read	Interrupt identification register bit [0] =1 for interrupt

Table 4-15 Keyboard and mouse control registers

Bit	Control
[0]	Force clock low, 0 = tristate (default), 1 = low
[1]	Force data low, 0 = tristate (default), 1 = low
[2]	Enable interface, 0 = disabled (default), 1 = enabled
[3]	Enable Transmit interrupt, 0 = disabled (default), 1 = enabled
[4]	Enable Receive interrupt, 0 = disabled (default), 1 = enabled
[5]	Interface type, 0 = standard (default), 1 = no line control

Table 4-16 Keyboard and mouse status registers

Bit	Status
[0]	Read data input line
[1]	Read clock input line
[2]	Read parity for last character, 1 = odd, 0 = even
[3]	Receiver busy, 0 = idle, 1 = busy receiving

Table 4-16 Keyboard and mouse status registers (continued)

Bit	Status
[4]	Receive register full, 0 = empty, 1 = full
[5]	Transmitter busy, 0 = idle, 1 = busy receiving
[6]	Transmit register empty, 0 = full, 1 = empty

4.8 UART interface

The serial interfaces are provided by two PrimeCell UARTs and are described in:

- *About the serial interface*
- *UART functional overview* on page 4-38
- *UART interrupts* on page 4-38
- *Baud rate selection* on page 4-39
- *Overview of UART registers* on page 4-40.

4.8.1 About the serial interface

The serial interface is implemented with two PrimeCell UARTs incorporated into the system controller FPGA. This is illustrated in Figure 4-18.

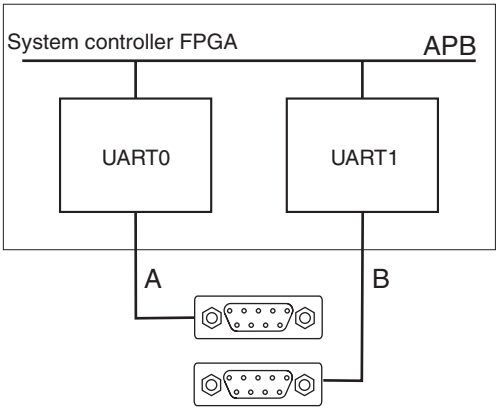


Figure 4-18 Serial interface

The UARTs are functionally similar to standard 16C550 devices. Each UART provides the following features:

- port function corresponds to the DTE configuration
- modem control inputs CTS, DCD, DSR, and RI
- modem output control signals RTS and DTR
- programmable baud rates of up to 460,800 bits per second (the line drivers however, are only guaranteed to 120kbps)
- 16-byte transmit FIFO
- 16-byte receive FIFO
- four interrupts.

4.8.2 UART functional overview

This section provides a functional overview of the UARTs. For detailed information about the UART, see the *UART (PL011) Technical Reference Manual.*

Data for transmission is written into a 16-byte transmit FIFO and the UART starts transmitting data frames with the parameters defined in the UART line control register. Transmission continues until the FIFO is emptied.

On the receive side, the UART begins sampling after it receives a start bit (LOW level input). When a complete word has been received, it is stored in the receive FIFO together with any error bits associated with that word. See *Overview of UART registers* on page 4-40 for details of the read FIFO bits.

You can disable the FIFOs. In this case, the UART provides 1-byte holding registers for the transmit and receive channels. The overrun bit in the UART_RSR register is set and an interrupt is generated if a byte is received before the previous one has been read.

A feature of the UART means that the FIFOs are not physically disabled but are bypassed. This means that if an overrun error occurs, the excess data is still stored in the FIFO and must be read out to clear the FIFO.

You set the baud rate of the UART by programming the bit rate divisor registers UART_LCRM and UART_LCRL.

4.8.3 UART interrupts

Each UART generates four interrupts. These are:

Modem status interrupt

This is asserted when any of the status lines (DCD, DSR, and CTS) change. It is cleared by writing to the UART_ICR register.

UART disabled interrupt

This is asserted when the UART is disabled and a start bit (low level) is detected on the receive line. It is cleared if the UART is enabled or if the receive line goes HIGH.

Rx interrupt

This is asserted when one of the following events occur:

- the receive FIFO is enabled and the FIFO is half or more than half full (contains eight or more bytes)
- the receive FIFO is not empty and there has been no data for more than a 32-bit period

- the receive FIFO is disabled and data is received.

The Rx interrupt is cleared by reading contents of the FIFO.

Tx interrupt

This is asserted when one of the following events occur:

- the transmit FIFO is enabled and the FIFO is half or less than half full
- the transmit FIFO is disabled and the holding buffer is empty.

The Tx interrupt is cleared by filling the FIFO to more than half full or writing to the holding register.

4.8.4 Baud rate selection

The baud rate generator in each UART uses a 16-bit divisor stored in the UART_LCRM and UART_LCRL registers to determine the bit period. The baud rate generator contains a 16-bit down counter that is clocked at 14.7456MHz by the **UARTCLK** signal.

4.8.5 Overview of UART registers

The UART registers are listed in Table 4-17. For more detailed information, refer to the *ARM PrimeCell UART (PL011) Technical Reference Manual*. The base addresses for the UARTs are:

`0x16000000` UART0, the top port on the connector, also called serial A.

`0x17000000` UART1, the bottom port on the connector, also called serial B.

Table 4-17 UART register summary

Address (UART 0)[a]	Type	Width	Reset value	Name	Description
0x16000000	Read/write	12/8	0x---	UARTDR	Data read or written from the interface. It is 12 bits wide on a read, and 8 on a write. This is also the UART base address.
0x16000004	Read/write	4/0	0x0	UARTRSR/ UARTECR	Receive status register (read)/ error clear register (write).
0x16000008- 0x1600014	-	-	-	-	Reserved.
0x16000018	Read	9	b-10010---	UARTFR	Flag register (read only).
0x1600001C	-	-	-	-	Reserved.
0x16000020	Read/write	8	0x00	UARTILPR	IrDA low-power counter register.
0x16000024	Read/write	16	0x0000	UARTIBRD	Integer baud rate divisor register.
0x160000028	Read/write	6	0x00	UARTFBRD	Fractional baud rate divisor register.
0x1600002C	Read/write	8	0x00	UARTLCR_H	Line control register, HIGH byte.
0x16000030	Read/write	16	0x0300	UARTCR	Control register.
0x16000034	Read/write	6	0x12	UARTIFLS	Interrupt FIFO level select register.
0x16000038	Read/write	11	0x000	UARTIMSC	Interrupt mask set/clear.
0x1600003C	Read	11	0x00-	UARTRIS	Raw interrupt status.
0x16000040	Read	11	0x00-	UARTMIS	Masked interrupt status.
0x16000044	Write	11	-	UARTICR	Interrupt clear register.
0x16000048	Read/write	3	0x00	UARTDMACR	DMA control register.

Table 4-17 UART register summary (continued)

Address (UART 0)[a]	Type	Width	Reset value	Name	Description
0x1600004C-1600007C	-	-	-	-	Reserved.
0x16000080-0x16000008C	-	-	-	-	Reserved (for test purposes).
0x16000090-0x16000FCC	-	-	-	-	Reserved.
0x16000FD0-0x16000FDC	-	-	-	-	Reserved for future ID expansion.
0x16000FE0	Read	8	0x11	UARTPeriphID0	Peripheral identification register bits [7:0].
0x16000FE4	Read	8	0x10	UARTPeriphID1	Peripheral identification register bits [15:8].
0x16000FE8	Read	8	0x14	UARTPeriphID2	Peripheral identification register bits [23:16].
0x16000FEC	Read	8	0x00	UARTPeriphID3	Peripheral identification register bits [31:24].
0x16000FF0	Read	8	0x0D	UARTPCellID0	PrimeCell identification register bits [7:0].
0x16000FF4	Read	8	0xF0	UARTPCellID1	PrimeCell identification register bits [15:8].
0x16000FF8	Read	8	0x05	UARTPCellID2	PrimeCell identification register bits [23:16].
0x16000FFC	Read	8	0xB1	UARTPCellID3	PrimeCell identification register bits [31:24].

a. For UART 1, use 0x17000000, 0x17000004, and so forth.

4.9 Counter/timer interface

The system controller FPGA provides three counter/timers. These are discussed in:

- *Counter/timer operation*
- *Counter/timer registers* on page 4-43.

4.9.1 Counter/timer operation

Figure 4-19 shows one counter/timer. Each comprises:

- a 32-bit down counter with selectable prescale (can operate as a 16-bit timer)
- two load registers
- a control register
- interrupt control registers.

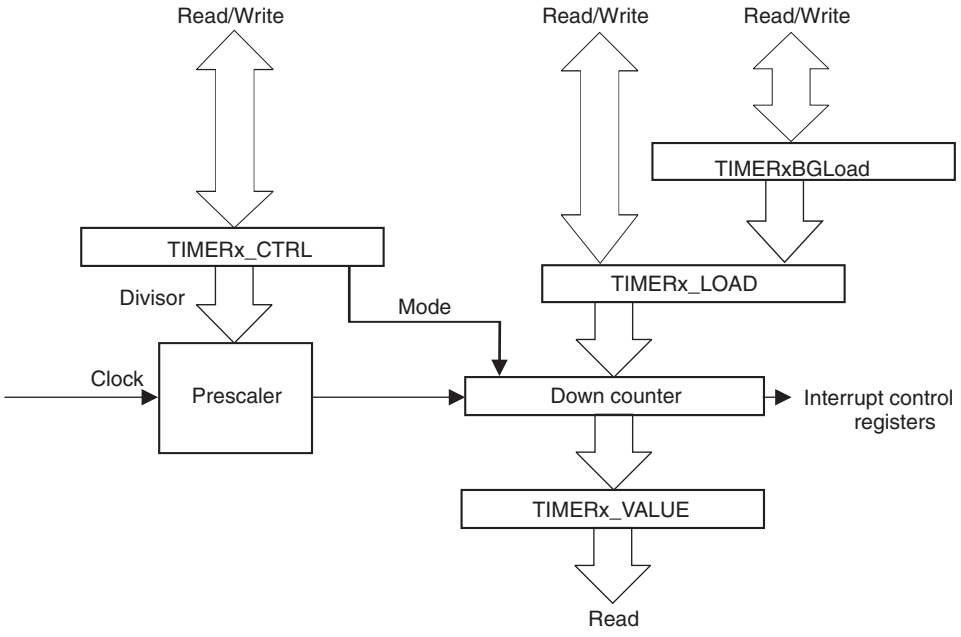


Figure 4-19 Counter/timer block diagram

Timer 0 is clocked by the system bus clock. Timer 1 and Timer 2 are clocked at a fixed frequency of 1MHz. The counters can be clocked directly or with a divide by 16 or 256 clock. The timers provide three operating modes:

Free running

The timer counts down to zero and then wraps around and continues to count down from its maximum value.

Periodic The counter counts down to zero and then reloads the period value held in the load register and continues to decrement.

One shot The counter counts down to zero and does not reload a value.

The prescale divisor and operating modes are selected by programming the control register TIMERx_CTRL.

The timer is enabled by setting a bit in the TIMERx_CTRL register. This register also contains the prescale selection bits and mode control bit.

A timer is loaded by writing to the load register TIMERx_LOAD. If the timer is enabled it begins a down count. When it reaches zero it generates an interrupt request. Interrupts are cleared by writing to the TIMERx_CLR register. The current value can be read at any time from the TIMERx_VALUE register.

For the one-shot mode, the counter generates an interrupt once only. When the counter reaches zero, it halts until reprogrammed by the user. This can be achieved by either clearing the One Shot Count bit in the control register (in which case the count will proceed according to the selection of Free-running or Periodic mode), or by writing a new value to the Load Value register.

4.9.2 Counter/timer registers

The counter/timer registers control the three counter/timers. There are seven registers for each of the counter/timers, as shown in Table 4-18.

Table 4-18 Timer register summary

Address	Type	Width	Reset value	Name	Description
0x13000000	Read/write	32	0x00000000	Timer0Load	Load value for Timer 0
0x13000004	Read	32	0xFFFFFFFF	Timer0Value	The current value for Timer 0
0x13000008	Read/write	8	0x20	Timer0Control	Timer 0 control register
0x1300000C	Write	-	-	Timer0IntClr	Timer 0 interrupt clear
0x13000010	Read	1	0x0	Timer0RIS	Timer 0 raw interrupt status
0x13000014	Read	1	0x0	Timer0MIS	Timer 0 masked interrupt status
0x13000018	Read/write	32	0x00000000	Timer0BGLoad	Background load value for Timer 0
0x13000100	Read/write	32	0x00000000	Timer1Load	Load value for Timer 1
0x13000104	Read	32	0xFFFFFFFF	Timer1Value	The current value for Timer 1

Table 4-18 Timer register summary (continued)

Address	Type	Width	Reset value	Name	Description
0x13000108	Read/write	8	0x20	Timer1Control	Timer 1 control register
0x1300010C	Write	-	-	Timer1IntClr	Timer 1 interrupt clear
0x13000110	Read	1	0x0	Timer1RIS	Timer 1 raw interrupt status
0x13000114	Read	1	0x0	Timer1MIS	Timer 1 masked interrupt status
0x13000118	Read/write	32	0x00000000	Timer1BGLoad	Background load value for Timer 1
0x13000200	Read/write	32	0x00000000	Timer2Load	Load value for Timer 2
0x13000204	Read	32	0xFFFFFFFF	Timer2Value	The current value for Timer 2
0x13000208	Read/write	8	0x20	Timer2Control	Timer 2 control register
0x1300020C	Write	-	-	Timer2IntClr	Timer 2 interrupt clear
0x13000210	Read	1	0x0	Timer2RIS	Timer 2 raw interrupt status
0x13000214	Read	1	0x0	Timer2MIS	Timer 2 masked interrupt status
0x13000218	Read/write	32	0x00000000	Timer2BGLoad	Background load value for Timer 2

Timer x load register

This is a 32-bit register containing the value from which the counter is to decrement. This is the value used to re-load the counter when Periodic mode is enabled, and the current count reaches zero.

When this register is written to directly, the current count is immediately reset to the new value at the next rising edge of TIMCLK which is enabled by TIMCLKEN.

The value in this register is also overwritten if the TimerXBGLoad register is written to, but the current count is not immediately affected.

If values are written to both the TimerXLoad and TimerXBGLoad registers before an enabled rising edge on TIMCLK, the following occurs:

- on the next enabled TIMCLK edge, the value written to the TimerXLoad value replaces the current count value
- each time the counter reaches zero, the current count value is reset to the value written to TimerXBGLoad.

Reading from the TimerXLoad register at any time after the two writes have occurred will retrieve the value written to TimerXBGLoad. That is, the value read from TimerXLoad is always the value that will take effect for Periodic mode after the next time the counter reaches zero.

Timer x background load

This is a 32-bit register containing the value from which the counter is to decrement. This is the value used to reload the counter when Periodic mode is enabled, and the current count reaches zero.This register provides an alternative method of accessing the TimerXLoad register. The difference is that writes to TimerXBGLoad will not cause the counter immediately to restart from the new value.Reading from this register returns the same value returned from TimerXLoad.

Timer x current value register

The timer value register contains the current count value for the timer. The upper 16 bits (of 32 bits) are undefined.

Timer x control register

The timer control registers are 8-bit read/write registers that control the operation of their associated counter/timers. The format of these three registers is similar.

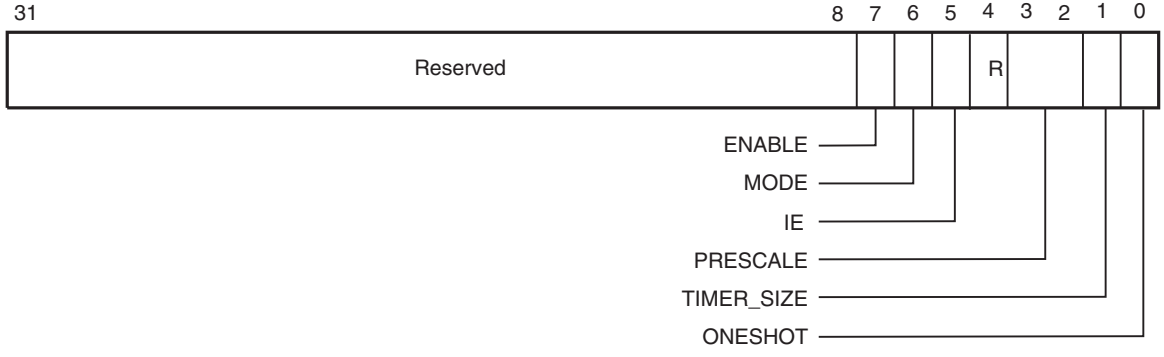


Figure 4-20 Timer control register

Table 4-19 describes the timer control register bits.

Table 4-19 TIMERx_CTL register

Bit	Name	Function
[7]	ENABLE	Timer enable: 0 = disabled 1 = enabled.
[6]	MODE	Timer mode: 0 = free running, counts once and then wraps to `0xFFFF` 1 = periodic, reloads from load register at the end of each count.
[5]	IE	Interrupt enable
[4]	R	Unused, always write as 0s.
3:2	PRESCALE	Prescale divisor: 00 = none 01 = divide by 16 10 = divide by 256 11 = undefined.
1	TimerSize	Selects 16/32 bit counter operation: 0 = 16-bit counter (default) 1 = 32-bit counter For 16-bit mode, write the high 16 bits of the 32-bit value as 0.
0	OneShot Count	Selects one-shot or wrapping counter mode: 0 = wrapping mode (default) 1 = one-shot mode

Timer x clear register

The timer clear register is a write-only location that does not have a storage element. Writing any value to this location clears the interrupt for the associated counter/timer.

Timer x raw interrupt status register

Bit 0 of this register indicates the raw interrupt status from the counter. This value is ANDed with the timer interrupt enable bit from the control register to create the masked interrupt that is passed to the interrupt output pin. Table 4-20 shows the bit assignments for the TimerXRIS register.

Table 4-20 TimerXRIS register

Bit	Name	Type	Function
[0]	Raw Timer Interrupt	Read	Raw interrupt status from the counter

Timer x interrupt status register

Bit 0 of this register indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupt status with the timer interrupt enable bit from the control register, and is the same value that is passed to the interrupt output pin. Table 4-21 shows the bit assignments for the TimerXMIS register.

Table 4-21 TimerXMIS register

Bit	Name	Type	Function
[0]	Timer Interrupt	Read	Enabled interrupt status from the counter

4.10 Debug LEDs and DIP switch interface

These registers shown in Table 4-22 are used to control the alphanumeric display and LEDs, and to read the 4-input DIP switch.

Table 4-22 LED control and switch registers

Address	Name	Type	Size	Function
0x1A000000	LED_ALPHA	Read/write	32	Alphanumeric characters register
0x1A000004	LED_LIGHTS	Read/write	4	LED control register for individual lights
0x1A000008	LED_SWITCHES	Read	4	DIP switch register

4.10.1 Alphanumeric characters register

This register is used to write characters to the alphanumeric display. The system controller FPGA manages the transfer of character data into a shift register within the alphanumeric display. You must allow each transfer to complete before writing new characters to the display using the following sequence:

1. Check that the display status is IDLE (bit 0 = 0) in the LED_ALPHA register.
2. Write 2 characters LED_ALPHA register (as a single 32-bit write to bits [31:1]).

Note

The data from the LED_ALPHA register is transferred as a serial bit-stream into the alphanumeric display at the same time as the LED control bits in LED_LIGHTS register.

Table 4-23 describes LED_ALPHA register bits.

Table 4-23 LED_ALPHA bit assignment

Bit	Name	Function
[30:1]	CHARACTERS	Bit patterns that form characters on the alphanumeric display. See Table 4-24 on page 4-49 for segment and bit assignments. Ensure that the display is idle before writing each new character.
[0]	STATUS	This is a read-only bit that returns the status of the alphanumeric display. For compatibility with the Integrator/AP, this always returns 0.

The digits and segments are identified in Figure 4-21.

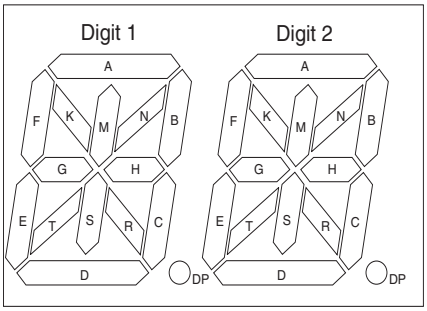


Figure 4-21 Identifying the alphanumeric display segment s

The bit assignments for the alphanumeric display segments in the LED_ALPHA register are shown in Table 4-24.

Table 4-24 LED_ALPHA bit-to-segment mapping

Digit 1	Segment	Digit 2
29	DP	30
28	T	14
27	S	13
26	R	12
25	N	11
24	M	10
23	K	9
22	H	8
21	G	7
20	F	6
19	E	5
18	D	4

Table 4-24 LED_ALPHA bit-to-segment mapping (continued)

Digit 1	Segment	Digit 2
17	C	3
16	B	2
15	A	1

4.10.2 LED control register

The LED_LIGHTS register controls four of the LEDs on the Integrator/CP. The LEDs are attached to bit ports provided by the alphanumeric display. Accesses to these bit ports are managed by the FPGA. Application programs write appropriate values into the LED_LIGHTS register, and the FPGA carries out the necessary transfer to turn the LEDs ON or OFF.

You must allow each transfer to the alphanumeric display to complete before writing new data to the display. Use the following sequence:

1. Check the display status is IDLE (bit 0 = 0) in the LED_ALPHA register.
2. Write a value to the LED_LIGHTS register.

Note

The data from the LED_LIGHTS register is transferred as a serial bit-stream into the alphanumeric display at the same time as the character data in LED_ALPHA.

The bit assignments for the LED_LIGHTS register are shown in Table 4-25.

Table 4-25 LED_LIGHTS register

Bit	Type	Function
3	Read/write	LED3: 0 = OFF 1 = ON
2	Read/write	LED2: 0 = OFF 1 = ON
1	Read/write	LED1: 0 = OFF 1 = ON
0	Read/write	LED0: 0 = OFF 1 = ON

See *Baseboard LEDs* on page 1-8 for the location and function of the LEDs.

4.10.3 DIP switch register

The DIP switch register is used to read the 4-pole DIP switch S2. The bit assignments for this register are shown in Table 4-26. A bit reads as 1 when the associated switch is ON and 0 when the switch is OFF.

Table 4-26 LED_SWITCH register

Bit	Type	Function
[3]	Read	Switch pole 4 (S2-4)
[2]	Read	Switch pole 3 (S2-3)
[1]	Read	Switch pole 2 (S2-2)
[0]	Read	Switch pole 1 (S2-1)

Chapter 5
System Expansion

This chapter describes standard and customized hardware expansion modules for the system. It contains the following sections:

5.1 Expanding your system with additional Integrator logic modules

There are restrictions on how you can expand your system:

- Signals on HDRA/HDRB must not clash. Interface modules that use normally free bus pins (for example, the IM-AD1 or IM-PD1) cannot be used with the Integrator/CP because some bus pins are reused for display control.

  Note

  ARM interface modules such as the IM-PD1 can be used, but some system features are unavailable. For example, the CLCD and VGA interfaces on both CP and IM-PD1 boards use the same set of pins on HDRA for the display controller, but with different pin assignments. Therefore, the VGA/CLCD interfaces on the IM-PD1 would be unusable in this configuration.

- Loading for each line is limited (see Chapter 2 *Getting Started*).
- Interrupt assignment is position-dependent on the board organization.
- Addressing is position-dependent (see *Module-assigned signals* on page 3-6).
- Inserting the Multi-ICE CONFIG jumper sets the mode for all the boards in the stack. Connect to the top module only.
- You cannot add additional ARM core modules to the stack.

5.2 Expanding your system with your own modules

There are restrictions on how you can expand your system with your own modules:

- Some of the bus lines are reserved or used for special signals. Be careful that expansion modules do not use signals on the HDRA and HDRB that clash with signals carried between the core module and baseboard. The standard logic module design for the IM-PD1, for example, clashes with the CP signals.
- Custom core modules cannot be added unless they have their own system bus and address space. That is, they must not use the bus signals on HDRA and HDRB. The module must have some dual-port memory, or some equivalent communication mechanism, in order to communicate with other modules. (Any additional core modules must operate in slave mode.)

  Note

  Custom core modules cannot use the header address or data lines.

- The appropriate connector type and spacing must be used.
- The custom module must decode its own address space.
- Keep the current drawn by the system below the limits defined in Table 2-2 on page 2-5.
- It is recommended that you do not separate the core module and baseboard. The core module will, however, operate stand-alone if it contains an appropriate FPGA image.

Appendix A
Porting Integrator/AP and IM-PD1

This appendix describes the how to port application software written for an Integrator/AP product fitted with an IM-PD1 board to the Integrator/CP with an attached core module. It contains the following sections:

A.1 Address map and interrupts

Most of the peripheral interrupts are routed direct to the primary interrupt controller (PIC). The Integrator/AP and IM-PD1 combination uses two interrupt controllers, the Integrator/AP (IC) and the IM-PD1 (VIC). Therefore the format of the interrupt handler will have to be modified to reflect the Integrator/CP layout. See the core module documentation for details on enabling, detecting and clearing interrupts.

Table A-1 shows the relationship between interrupts and peripheral addresses.

Table A-1 Peripheral and address map

Peripheral	CP module address	CP module IRQ	AP address	IM-PD1 address	IRQ	Notes
UART0	0x16000000	PIC 1	0x16000000	-	IC 1	PL011
UART1	0x17000000	PIC 2	0x17000000	-	IC 2	PL011
MMCI	0x1C000000	PIC 23/24	-	0xC0700000	VIC 7/8	-
KYBD	0x18000000	PIC 3	0x18000000	-	IC 3	-
MOUSE	0x19000000	PIC 4	0x19000000	-	IC 4	-
AACI	0x1D000000	PIC 25	-	0xC0800000	VIC 9	-
TSCI	0x1E000000	PIC 28	-	0xC0300000	VIC 3	Not Equivalent
CLCD	0xC0000000	PIC 22	-	0xC1000000	VIC 11	-
GPIO	0xC9000000	-	-	0xC0400000	VIC 4	Not Equivalent
LEDs	0x1A000000	-	0x1A000000	-	-	-
Timers	0x13000000	PIC 5/6/7	0x13000000	-	IC 5/6/7	1MHz ADK timers
PIC	0x14000000	-	0x14000000	0xC3000000	-	-

A.2 Registers

The following registers have been replaced or modified:

- All Integrator/AP system controller (SC_) registers have been removed. These are partly replaced by CP_ID and CM_CTRL (on the core module).
- The Integrator/AP CM_LMBUSCNT has been removed. This can be achieved with one of the new 32bit ADK timers.
- The IM-PD1 CLCD control register (LM_CONTROL) is now merged with CM_CTRL (on the core module).
- New registers at `0xCB000000` for CP_ID, CP_FLASH, CP_INTREG, CP_DECODE. See section *Register overview* on page 3-25 for further details on these registers.

A.3 Other changes

The following miscellaneous changes should be considered:

- The Integrator/AP timers have been replaced with ADK timers and are now clocked at 1MHz (not 24MHz). See section *Counter/timer interface* on page 4-42 for further details on ADK timers.
- The Integrator/AP dual UARTs have been replaced with PL011 PrimeCells (Not PL010). See section *UART interface* on page 4-37 for further details on UARTS.
- The TSCI interface has been redesigned. Although the interface uses the ADS7843 driver, serial communications to the driver is performed by a dedicated serial interface. See section *Touchscreen controller interface* on page 4-14.
- The CLCD frame buffer has moved (for example, SDRAM `0x00200000`). This used the SSRAM in the LM.
- A secondary interrupt controller (SIC) has been added at `0xCA000000`.
- The CTS and RTS signals were not supported on the PL010 PrimeCell (the signals were accessed from the SC_CTRL register.) These signals are supported on the PL011 PrimeCell used in the Integrator/CP (access is from the PrimeCell registers).
- MMCI CARDIN and WPROT raw status is read through the secondary interrupt controller (SIC) or CP_INTREG. CARDIN is latched and must be cleared by writing a one to this bit in CP_INTREG.

Appendix B
Connector Pinouts

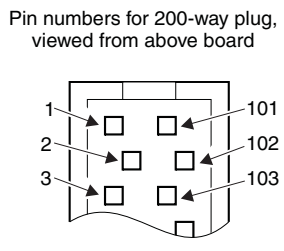
This appendix describes the Integrator/CP interface connectors and signal connections. It contains the following sections:

B.1 Header connectors

This section describes the header connectors located on the baseboard:

B.1.1 Baseboard connector HDRA

Figure B-1 shows the pin numbering of the connector HDRA on the baseboard.



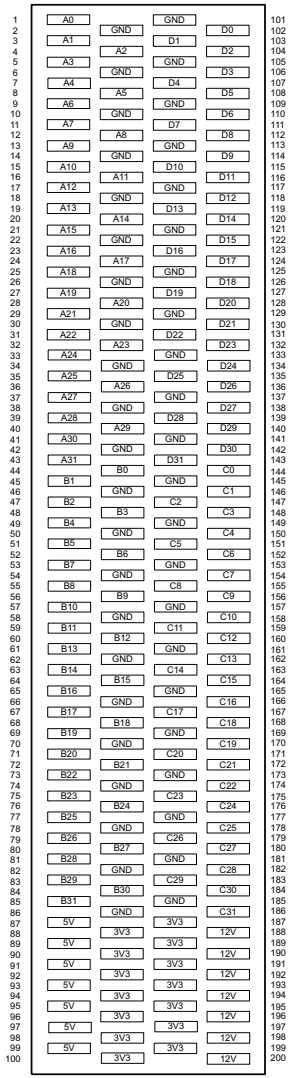


Figure B-1 Connector pin numbering

The signals on the pins labeled A[31:0], B[31:0], C[31:0], and D[31:0] are listed in Table B-1. A more in depth description of these buses in provided in *Baseboard AHB bus* on page 3-3. The signal names are the standard AHB names. Some pins have different functions under AHB-Lite or when used on the Integrator/CP.

Table B-1 HDRA signals descriptions

Pin label	AHB signal name	CP specific[a]	Description
A[31:0]	**HADDR[31:0]**	-	System address bus
B[31:0]	Peripheral control	Yes	Peripheral connections between base board and core module
C[31:16]	Peripheral control	Yes	Peripheral connections between base board and core module
C15	Reserved	Yes	Locked transaction On AHB-Lite there is only one master, therefore arbitration is not required.
C14	Reserved	Yes	Not used on Integrator/CP core modules
C13	**HRESP0**	-	Slave response
C12	**HREADY**	-	Slave wait response
C11	**HWRITE**	-	Write transaction
C10	**HPROT2**	-	Transaction protection type
C[9:8]	**HPROT[1:0]**	-	Transaction protection type
C[7:5]	**HBURST[2:0]**	-	Transaction burst size
C4	**HPROT[3]**	-	Transaction protection type
C[3:2]	**HSIZE[1:0]**	-	Transaction width The AHB specification defines a 3-bit bus for HSIZE, but 2 bits is sufficient to describe transfers of up to 64-bits wide which is why a 2-bit bus is sufficient on Integrator.
C[1:0]	**HTRAN[1:0]**	-	Transaction type
D[31:0]	**HDATA[31:0]**	-	System data bus

a. The use of these pins is specific to the Integrator/CP and might not be compatible with other modules that use this pins for other functions.

B.1.2 Baseboard connector HDRB

Figure B-2 shows the pin numbers of the connector HDRB on the baseboard.

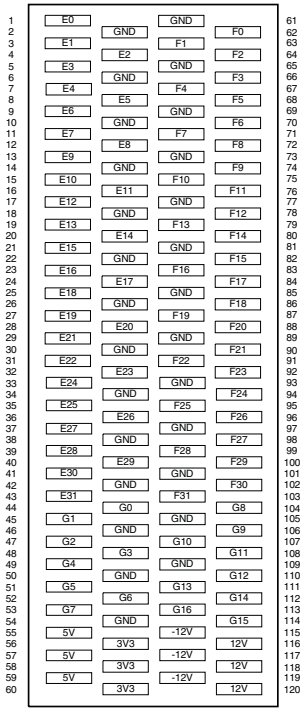


Figure B-2 HDRB pin numbering

Table B-2 on page B-5 describes the signals on the pins labeled E[31:0], F[31:0], and G[16:0] for AMBA AHB-Lite system bus. The signal names are the standard AHB names. Some pins have different functions under AHB-Lite or when used on the Integrator/CP.

Table B-2 HDRB signal description

Pin label	Signal Name	CP specific	Description
E[31:28]	**HCLK[3:0]**	-	System clock to the core module.
E[27:24]	**nPPRES[3:0]**	Yes	Core Module present. Each core module ties **nPPRES[0]** LOW and leaves **nPPRES[3:1]** open circuit. These signals rotate as they move up or down the stack so that there is a connection between each module and one of these signals at the system controller on the baseboard. Only one core module however, is permitted in the Integrator/CP system.
E[23:20]	**nIRQ[3:0]**	-	Interrupt request to processor.
E[19:16]	**nFIQ[3:0]**	-	Fast interrupt requests to processor.
E[15:12]	**ID[3:0]**	Yes	Core Module stack position indicator. Only one core module however, is permitted in the Integrator/CP system.
E[11:8]	**HLOCK[3:0]**	Yes	System bus lock from processor, but only a single core module is permitted.
E[7:4]	**HGRANT[3:0]**	Yes	System bus grant to processor, but only a single core module is permitted.
E[3:0]	**HBUSREQ[3:0]**	Yes	System bus request from processor, but only a single core module is permitted.
F[31:16] and F[9:8]	-	Yes	A variety of interface connections
F[15:10]	-	Yes	MMC interface signals
F[13] and F[7:5]	-	Yes	Audio codec signal
F[4:0]	-	Yes	Touchscreen host interface signals
G16	**nRTCKEN**	-	**RTCK** AND gate enable.
G[15:14]	**CFGSEL[1:0]**	-	FPGA configuration select.
G13	**nCFGEN**	-	Sets system into configuration mode.
G12	**nSRST**	-	Multi-ICE reset (open collector).
G11	**FPGADONE**	-	Indicates when FPGA configuration is complete.

Table B-2 HDRB signal description (continued)

Pin label	Signal Name	CP specific	Description
G10	**RTCK**	-	Returned JTAG test clock.
G9	**nSYSRST**	-	Buffered system reset.
G8	**nTRST**	-	JTAG reset.
G7	**TDO**	-	JTAG test data out.
G6	**TDI**	-	JTAG test data in.
G5	**TMS**	-	JTAG test mode select.
G4	**TCK**	-	JTAG test clock.
G[3:1]	**HMAST[2:0]**	Yes	Master ID. Binary encoding of the master currently performing a transfer on the bus. Corresponds to the module ID and to the **HBUSREQ** and **HGRANT** line numbers. However on AHB-Lite, only one master is permitted.
G0	**nMBDET**	-	Baseboard detect. This signal is tied LOW on the CP. When a module is attached to the baseboard, it detects that **nMBDET** is LOW and routes **TDI** and **TCK** down to the baseboard where they are looped back onto **TDO** and **RTCK**. Also, core modules pass addresses above `0x11000000` on to the system bus where they are decoded by the baseboard or by other modules. For the Integrator/CP, however, one core module is permitted.

B.2 Peripheral connectors

This section describes the peripheral connectors located on the baseboard.

B.2.1 Serial interface connectors

The pinout of the serial ports is shown in Figure B-3 and Table B-3.

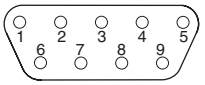


Figure B-3 Serial interface connector pinout

Table B-3 Serial interface signal descriptions

Pin	Signal	Type	Function
1	**DCD**	Input	Data carrier detect
2	**Rx**	Input	Receive
3	**Tx**	Output	Transmit
4	**DTR**	Output	Data terminal ready
5	**GND**	-	Ground
6	**DSR**	Input	Data set ready
7	**RTS**	Output	Ready to send
8	**CTS**	Input	Clear to send
9	**RI**	Input	Ring indicator

Note

The serial interfaces signals operate at RS232 signal levels.

Serial port functionality corresponds to the DTE configuration.

B.2.2 Keyboard and mouse connectors

The pinout of the KMI connectors is shown in Figure B-4 on page B-8.

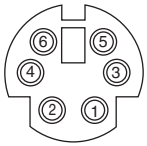


Figure B-4 KMI connector pinouts

Table B-4 shows signals on the KMI connectors.

Table B-4 Mouse and keyboard port signal descriptions

Pin	Keyboard (Lower)		Mouse (Top)	
	Signal	Function	Signal	Function
1	**KDATA**	Keyboard data	**MDATA**	Mouse Data
2	NC	Not connected	NC	Not connected
3	**GND**	Ground	**GND**	Ground
4	**5V**	5V	**5V**	5V
5	**KCLK**	Keyboard clock	**MCLK**	Mouse clock
6	NC	Not connected	NC	Not connected

B.2.3 LCD connectors

The pinout for the generic LCD connector is shown in Table B-5 on page B-9 and Figure B-5 on page B-9.

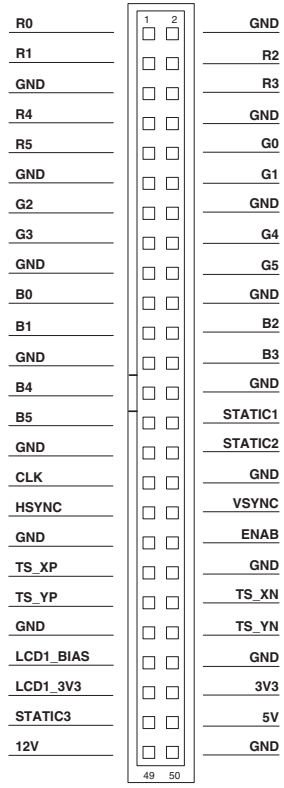


Figure B-5 Generic LCD, J13 pinout

Table B-5 Generic LCD, J13

Signal	Pin	Pin	Signal
RO	1	2	**GND**
R1	3	4	**R2**
GND	5	6	**R3**
R4	7	8	**GND**
R5	9	10	**G0**
GND	11	12	**G1**
G2	13	14	**GND**
G3	15	16	**G4**

Table B-5 Generic LCD, J13 (continued)

Signal	Pin	Pin	Signal
GND	17	18	**G5**
B0	19	20	**GND**
B1	21	22	**B2**
GND	23	24	**B3**
B4	25	26	**GND**
B5	27	28	**STATIC1**
GND	29	30	**STATIC2**
CLK	31	32	**GND**
HSYNC	33	34	**VSYNC**
GND	35	36	**ENAB**
TS_XP	37	38	**GND**
TS_YP	39	40	**TS_XN**
GND	41	42	**TS_YN**
BIAS	43	44	**GND**
3V3	45	46	**3V3**
STATIC3	47	48	**5V**
12V	49	50	**GND**

The signals on the Sharp TFT VGA connector are shown in *** 'Sharp connector, J14' on page 11 *** and Figure B-6 on page B-11. The signals on the standard VGA connector are shown in *** 'VGA connector, J15' on page 12 *** and Figure B-7 on page B-12. The signals for the CCFL power supply connector are shown in Table B-8 on page B-13 and Figure B-8 on page B-13.

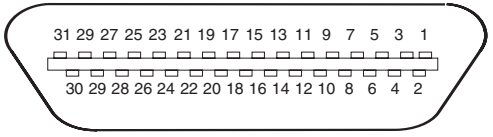


Figure B-6 Sharp connector, J14

Table B-6 Sharp LQ084 TFT VGA signals, J14

Pin	Signal
1	**GND** (shielding case is also connected to GND)
2	**CLK**
3	**H_SYNC**
4	**V_SYNC**
5	**GND**
6 to 11	**Red**, R0 to R5
12	**GND**
13 to 18	**Green**, G0 to G5
19	**GND**
20 to 25	**Blue**, B0 to B5
26	**GND**
27	**Enable**
28	**3V3**
29	**3V3**
30	**RL**, Right/left reverse
31	**UD**, Up/down reverse

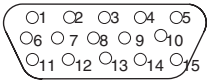


Figure B-7 VGA connector, J15

Table B-7 VGA connector, J15

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

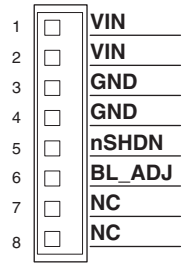


Figure B-8 Backlight power connector, J5

Table B-8 CCFL power connector, J5

Pin	Signal
1	**VIN** (power supply to inverter, for example, LinFinity LXM1611)
2	**VIN**
3	**GND**
4	**GND**
5	**nSHDN** (set high for normal operation)
6	**BL_ADJ** (0 to 2.5V brightness adjustment)
7	NC
8	NC

B.2.4 Touchscreen connector

The pinout for the touchscreen connector is shown in Figure B-9 on page B-14. The signals routed through the HDRB connector to the touch screen controller are shown in Table B-9.

Table B-9 Touchscreen host interface signal assignment

Signal name	HDRB connector	Description
TS_DIN	F2	Serial data input to controller
TS_nCS	F1	Controller chip select
TS_DCLK	F0	Clock input to controller

Table B-9 Touchscreen host interface signal assignment (continued)

Signal name	HDRB connector	Description
TS_DOUT	F3	Data output from controller
TS_BUSY	Not connected	Busy indicator from controller
TS_nPENIRQ	F4	Interrupt from controller

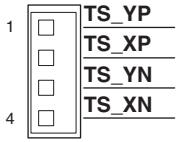


Figure B-9 J16 pinout

B.2.5 GPIO connector

The signals on the GPIO connector are shown in Table B-10 on page B-15 and Figure B-10 on page B-15.

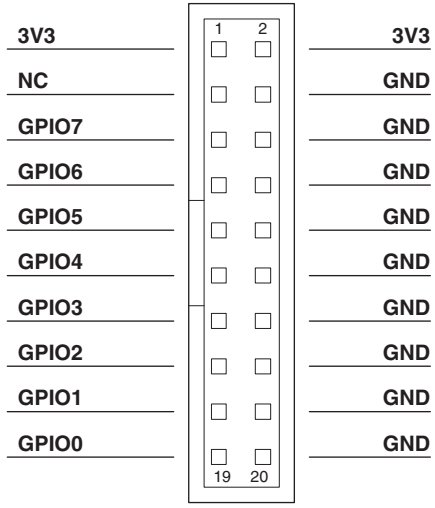


Figure B-10 GPIO, J20

Table B-10 GPIO, J20

Signal	Pin	Pin	Signal
3V3	1	2	**3V3**
NC	3	4	**GND**
GPIO7	5	6	**GND**
GPIO6	7	8	**GND**
GPIO5	9	10	**GND**
GPIO4	11	12	**GND**
GPIO3	13	14	**GND**
GPIO2	15	16	**GND**
GPIO1	17	18	**GND**
GPIO0	19	20	**GND**

B.2.6 Ethernet debug connector

The pinout for the Ethernet debug connector is shown in Table B-11 and Figure B-11 on page B-17. The Ethernet output connector is a standard RJ45 connector.

Table B-11 Ethernet debug, J8

Signal	Pin	Pin	Signal
RXDO	1	2	**3V3**
RXD1	3	4	**TX_EN**
RXD2	5	6	**CRS**
RXD3	7	8	**COL**
TXD0	9	10	**RX_ER**
TXD1	11	12	**MDIO**
TXD2	13	14	**MCLK**
TXD3	15	16	**RX_CLK**
RX_DV	17	18	**TX_CLK**
RBIAS	19	20	**nLNK**
nCNTRL	21	22	**nCSOUT**
REG_A25	23	24	**X25OUT**
ENEEP	25	26	**LBK**
EEDO	27	28	**3V3**
EEDI	29	30	**3V3**
EESK	31	32	**BIGEND**
EECS	33	34	**GND**

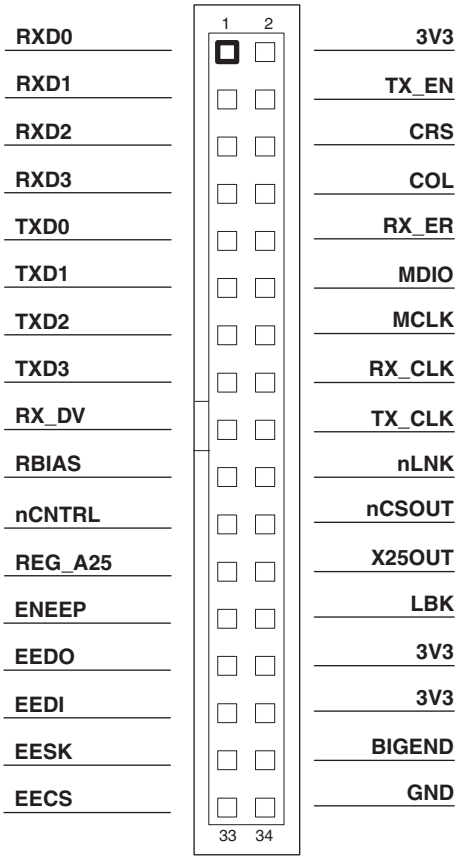


Figure B-11 Ethernet debug, J8 pinout

B.2.7 LED debug connector

The pinouts for the LED debug connector are shown in Table B-12 on page B-18 and Figure B-12 on page B-18.

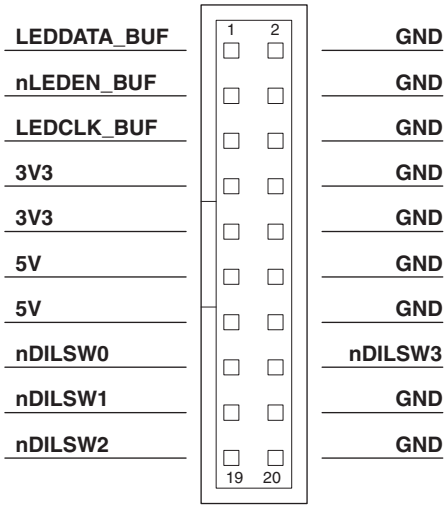


Figure B-12 LED, J18

Table B-12 Led debug, J18

Signal	Pin	Pin	Signal
LEDDATA_BUF	1	2	**GND**
nLEDEN_BUF	3	4	**GND**
LEDCLK_BUF	5	6	**GND**
3V3	7	8	**GND**
3V3	9	10	**GND**
5V	11	12	**GND**
5V	13	14	**GND**
nDILSW0	15	16	**nDILSW3**
nDILSW1	17	18	**GND**
nDILSW2	19	20	**GND**

B.2.8 MMC connector

Figure B-13 shows the pin numbering and signal assignment for the MMC card. Pin 9 is next to pin 1 and pins 7 and 8 are spaced more closely together than the other pins. Socket J9 provides test access to the MMC pins (see Figure B-15 on page B-20).

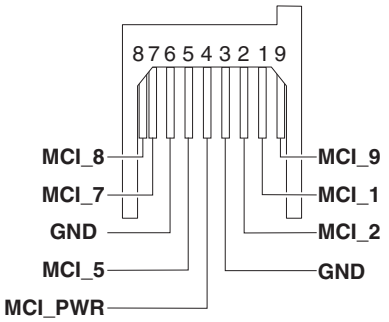


Figure B-13 MMC card socket pin numbering, J11

MMC cards use seven pins. Figure B-14 shows an MMC card, with the contacts face up.

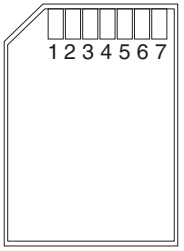


Figure B-14 MMC card

Connector J9 enables you to access the signals for debugging or to route them to an off-PCB card socket. The pinout of J9 is shown in Figure B-15 on page B-20.

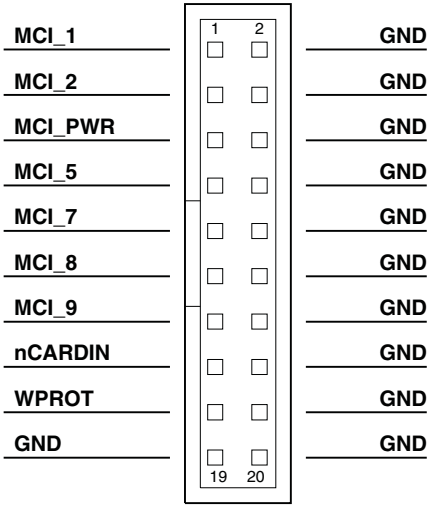


Figure B-15 J9 pinout

B.2.9 Audio

The pinouts for the audio connectors to the codec are shown in Figure B-16 on page B-21, Table B-13 and Table B-14 on page B-21. Line out is the top socket on the connector assembly. See *Audio interface* on page 4-20 for more details.

Table B-13 Line in/out, stacked stereo jack J10

Pin	Signal	Function
1	**CODEC_LINE_IN_R**	Line in, right
2	**CODEC_LINE_IN_L**	Line in, left
3	**GND**	Ring
4	**AMP_R**	Line out, right
5	**AMP_L**	Line out, left
6	**GND**	Ring

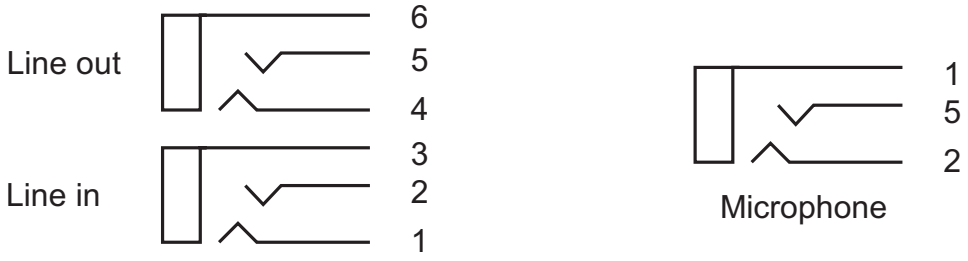


Figure B-16 Audio connectors

Table B-14 Microphone, 3.5mm jack J12

Pin	Signal	Function
1	**GND**	Ring
2	**CODEC_MIC1**	Microphone 1
5	**CODEC_MIC2**	Microphone 2

Note

Only one microphone line can be selected at a time.

Appendix C
Test Points

This appendix describes the test points on the CP baseboard. It contains the following section:

- *Baseboard test points* on page C-2.

See also Appendix B *Connector Pinouts* for test signals connected to pins in connectors. See the documentation provided with your core module for details on core module test points.

C.1 Baseboard test points

Figure C-1 shows the test points on the baseboard.

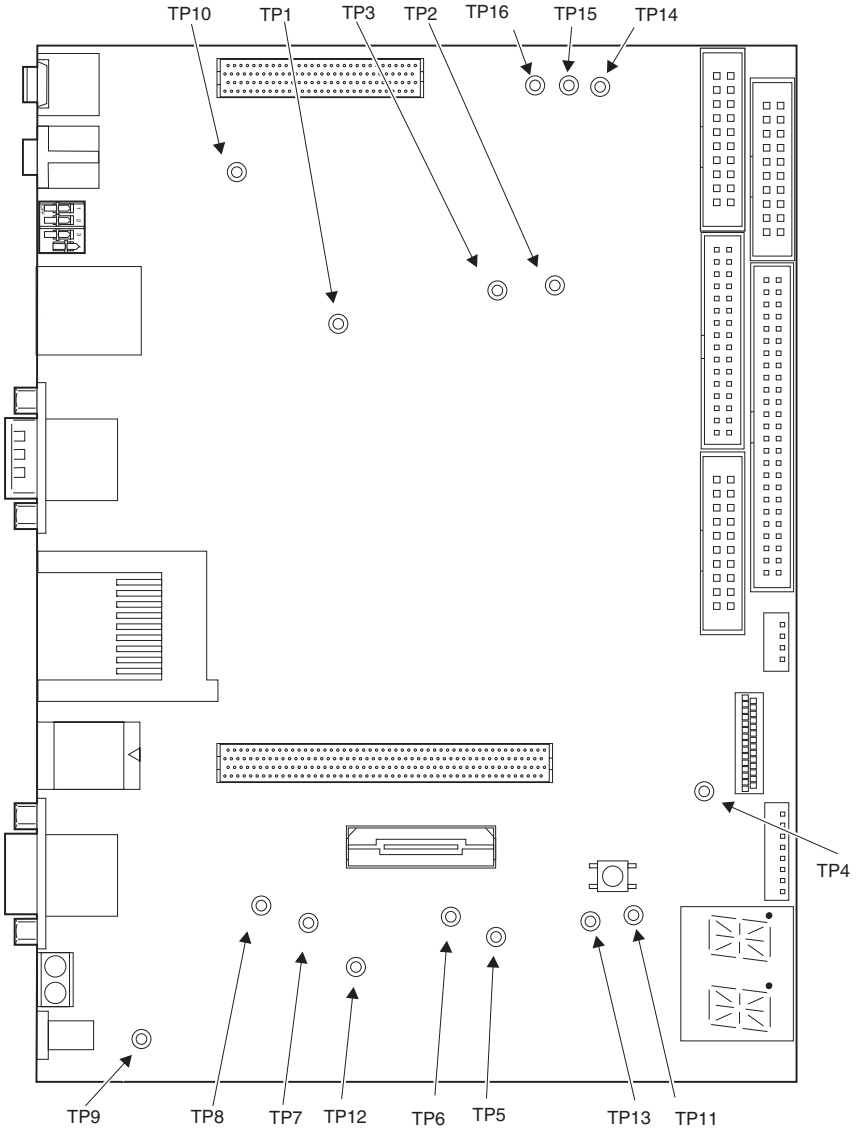


Figure C-1 CP test points

The functions of the test points are summarized in Table C-1.

Table C-1 Test point functions (baseboard)

Test point	Signal	Function
TP1	**HCLK_MON**	Clock
TP2	**UARTCLK**	UART reference clock
TP3	**REF25MHz**	25MHz clock
TP4	**VIN_REF**	Reference voltage input
TP5	**5V**	5 Volts
TP6	**5V**	5 Volt current sense
TP7	**3V3_WL**	3.3 Volts
TP8	**3V3**	3.3 Volt current sense
TP9	**VIN**	Supply voltage
TP10	**5V_ANALOG**	5 Volts Analog
TP11	**5VSB**	5 Volts Standby
TP12	**nSHDN**	Shutdown signal for LCD bias supply
TP13	**LCD_BIAS**	12 to 25 Volt LCD bias voltage
TP14	**12V**	12 Volts
TP15	**-12V**	-12 Volts (not supplied by on-board DC-DC converters)
TP16	**GND**	0 Volts

Glossary

This glossary lists all the abbreviations used in the Integrator/CP User Guide.

AACI Advanced Audio CODEC Interface.

ADC Analog to Digital Converter. A device that an analog signal into converts digital data.

ADK AMBA Design Kit. The ADK comprises the building blocks required to create an example system based on the low-power, generic design methodology of the Advanced Microcontroller Bus Architecture (AMBA).

AHB Advanced High-performance Bus. The ARM open standard for on-chip buses.

AMBA Advanced Microcontroller Bus Architecture.

APB Advanced Peripheral Bus. The ARM open standard for peripheral buses. This design is optimized for low power and minimal interface complexity

DAC Digital to Analog Converter. A device that converts digital data into analog level signals.

DCC Debug Communications Controller.

DCE Data Communications Equipment a specification for serial ports on communications adaptors such as modems (see also DTE).

DTE	Data Terminal Equipment, a specification for serial ports on terminals (see also DCE).
ETM	Embedded Trace Macrocell.
FIFO	First In First Out memory buffer.
FPGA	Field Programmable Gate Array.
GPIO	General Purpose Input/Output.
JTAG	Joint Test Action Group. The committee which defined the IEEE test access port and boundary-scan standard.
KMI	Keyboard and Mouse Interface.
LCD	Liquid Crystal Display. A flat-screen display.
MMC	Multimedia card. Used to store files containing audio or images.
Multi-ICE	Multi-ICE is a system for debugging embedded processor cores using a JTAG interface.
PCB	Printed Circuit Board.
PIC	Primary Interrupt Controller.
PLD	Programmable Logic Device.
PLL	Phase-Locked Loop, a type of programmable oscillator.
POR	Power On Reset.
RTC	Real-Time Clock.
SIC	Secondary Interrupt Controller.
UART	Universal Asynchronous Receiver/Transmitter.
VGA	Video Graphics Adaptor. A display standard.
TCM	Tightly-Coupled Memory.
TSCI	Touch Screen Controller Interface.
VL-bus	VESA Local Bus.
VIC	Vectored Interrupt Controller.

Index

The items in this index are listed in alphabetical order, with symbols and numerics appearing at the end. The references given are to page numbers.

U

W