Integrator®/CM920T, CM920T-ETM, and CM940T

HBI-0070

User Guide

ARM®

ARM DUI 0149B

Integrator/CM920T, CM920T-ETM, and CM940T
User Guide

Release Information

Description	Issue	Change
21 February 2001	A	New document
19 November 2002	B	Second release

Proprietary Notice

Confidentiality Status

This document is Open Access. This document has no restriction on distribution.

Product Status

The information in this document is final (information on 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

CE

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/CM920T, CM920T-ETM, and CM940T User Guide

List of Tables
Integrator/CM920T, CM920T-ETM, and CM940T User Guide

List of Figures
Integrator/CM920T, CM920T-ETM, and CM940T User Guide

Preface

This preface introduces the *Integrator/CM920T, CM920T-ETM,and CM940T User Guide*. It contains the following sections:

- *About this document* on page xii
- *Feedback* on page xvi.

Note

This manual covers core modules that use the HBI-0070C PCB, and above. For information on CM720T and CM920T products using the older HBI-0047B PCB, see the ARM Integrator *CM720T User Guide* and *CM920T User Guide*.

About this document

This document describes how to set up and use the ARM Integrator/CM920T, CM920T-ETM, and CM940T core modules.

Intended audience

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

Organization

This document is organized into the following chapters:

Chapter 1 ***Introduction***

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

Chapter 2 ***Getting Started***

Read this chapter for a description of how to set up and start using the core module. This chapter describes how to connect the core module and how to apply power.

Chapter 3 ***Hardware Description***

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

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

Read this chapter for a description of the core module memory map and registers. This chapter describes how to use the registers to control the core module.

Chapter 5 ***Using Core Modules with an Integrator/AP***

Refer to this chapter for details on how to use the core module with an Integrator/AP motherboard.

Chapter 6 ***Using Core Modules with an Integrator/CP***

Refer to this chapter for details on how to use the core module with an Integrator/CP baseboard. An FPGA image that supports the CP is required. Current core modules contain images for both the AP and CP.

Appendix A ***Signal Descriptions***

Refer to this appendix for a description of the signals on the HDRA, HDRB, trace, and logic analyzer connectors.

Appendix B ***Specifications***

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

Typographical conventions

The following typographical conventions are used in this book:

`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.

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

bold Highlights interface elements, such as menu names and buttons. Also used for terms in descriptive lists, where appropriate.

`monospace bold`

Denotes language keywords when used outside example code and ARM processor signal names.

Further reading

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

ARM publications

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

- *ARM Integrator CM720T and CM920T User Guide* (ARM DDI 0157)
- *ARM920T Technical Reference Manual* (ARM DDI 0151)
- *ARM940T Technical Reference Manual* (ARM DDI 0144)
- *ARM Integrator/AP User Guide* (ARM DUI 0098)
- *ARM Integrator/CP User Guide* (ARM DUI 0159)
- *ARM ETM9 Technical Reference Manual* (ARM DDI 0157)
- *ARM Multi-ICE User Guide* (ARM DUI 0048)
- *AMBA Specification* (ARM IHI 0011)
- *ARM Architecture Reference Manual* (ARM DDI 0100)

- *ARM Firmware Suite Reference Guide* (ARM DUI 0102)
- *ADS Tools Guide* (ARM DUI 0067)
- *ADS Debuggers Guide* (ARM DUI 0066)
- *ADS Debug Target Guide* (ARM DUI 0058)
- *ADS Developer Guide* (ARM DUI 0056)
- *ADS CodeWarrior IDE Guid*e (ARM DUI 0065)
- *RealView Debugger User Guid*e (ARM DUI 0153)
- *RealView Compilers and Libraries Guid*e (ARM DUI 0205)
- *RealView Linker and Utilities Guid*e (ARM DUI 0206)
- *ARM PrimeCell UART (PL011) Technical Reference Manual* (ARM DDI 0183*)*
- *PrimeCell MultiMedia Card Interface (PL181) Technical Reference Manual* (ARM DDI 0205)
- *Application Note 101- Stacking Integrator Modules* (ARM DAI 0101A)
- *ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual* (ARM DDI 0173).

Other publications

The following publication provides information about the clock controller chip used on the Integrator modules:

- *MicroClock OSCaR User Configurable Clock Data Sheet* (MDS525), MicroClock Division of ICS, San Jose, CA.

The following publications provide information and guidelines for developing products for Microsoft Windows CE:

- *Standard Development Board for Microsoft® Windows® CE*, 1998, Microsoft Corporation
- *HARP Enclosure Requirements for Microsoft® Windows® CE*, 1998, Microsoft Corporation.

Further information on these topics is available from the Microsoft web site.

Feedback

ARM Limited welcomes feedback both on the ARM Integrator/CM920T, CM920T-ETM, and CM940T core modules and on the documentation.

Feedback on this document

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

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

General suggestions for additions and improvements are also welcome.

Feedback on the ARM Integrator/CM920T, CM920T-ETM, and CM940T

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

- the product name
- an explanation of your comments.

Chapter 1
Introduction

This chapter introduces the ARM Integrator/CM920T, CM920T-ETM, and CM940T core modules. It contains the following sections:

1.1 About the core module

The Integrator/CM920T, CM920T-ETM, and CM940T core modules provide you with a development system that can be used to develop products around the ARM920T™, ARM920T™-*Embedded Trace Macrocell* (ETM), and ARM940T™ cores.

The core module can be used in several different ways. With power and a connection to a Multi-ICE® unit, the core module provides a basic development system. By mounting the core module onto an Integrator motherboard (Integrator/AP or Integrator/CP, for example) or other Integrator modules, you can build a realistic emulation of the system being developed.

The core module can be used in the following ways:

- as a standalone software development system using Multi-ICE for program download
- mounted onto an ARM Integrator motherboard
- mounted onto an ARM logic module without a motherboard, with the logic module providing the system controller functions of a motherboard
- integrated into a third-party development or ASIC prototyping system.

Figure 1-1 on page 1-3 shows the layout of the core module.

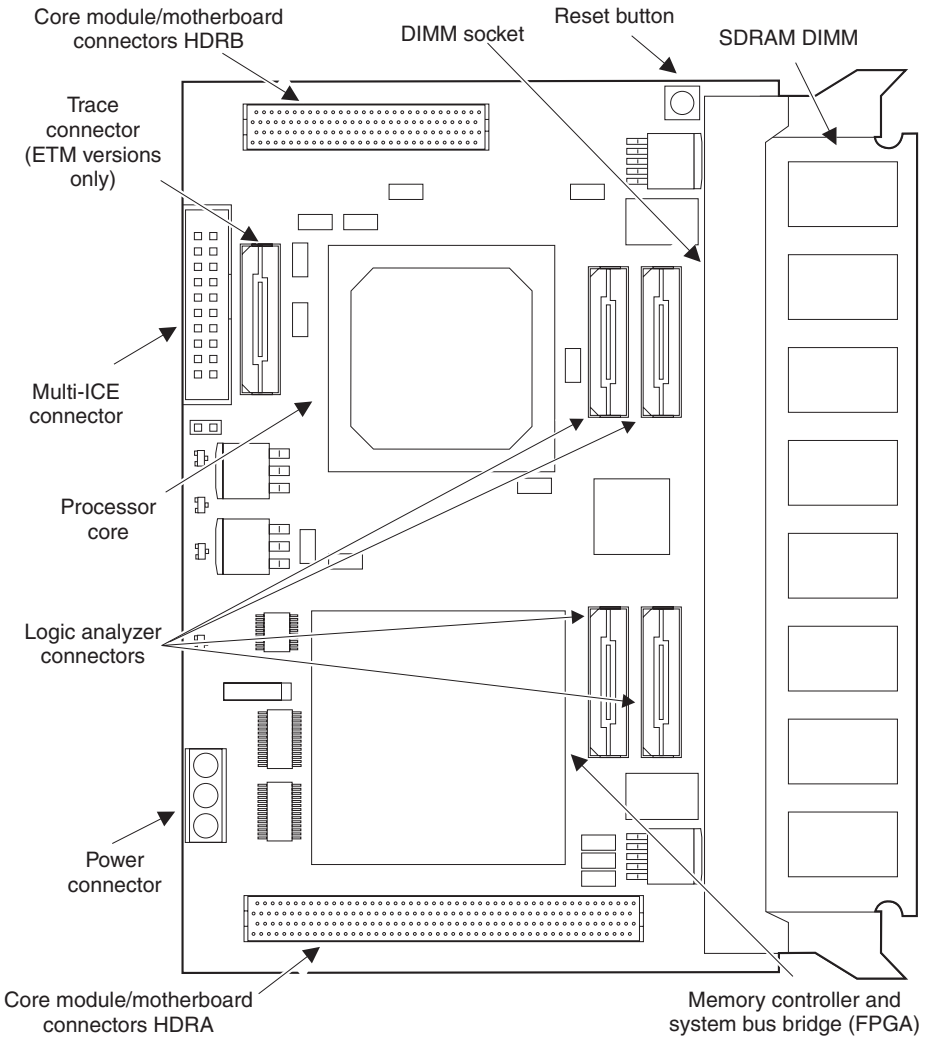


Figure 1-1 Core module layout

1.2 Core module architecture

The major components on the core module are:

- a microprocessor core
  - CM920T (ARM920T test chip)
  - CM920T-ETM (ARM920T with ETM)
  - CM940T (ARM940T test chip).

  For a brief description of these cores, see *ARM microprocessor test chip* on page 3-2.
- core module FPGA that implements:
  - SDRAM controller
  - system bus bridge
  - reset controller
  - interrupt controller
  - status, configuration, and interrupt registers.

  Note

  The FPGA image used with the Integrator/CP baseboard implements additional peripherals. See Chapter 6 *Using Core Modules with an Integrator/CP*.
- 1MB SSRAM
- up to 256MB of SDRAM (optional) plugged into the DIMM socket
- SSRAM controller
- clock generator
- system bus connectors
- logic analyzer connectors for AHB and Trace port.

Through-board connectors on the core module enable up to four core modules to be stacked on one Integrator/AP motherboard. One core module can be stacked on an Integrator/CP baseboard.

1.2.1 System architecture

Figure 1-2 on page 1-5 shows the architecture of the core module as used with an Integrator/AP motherboard.

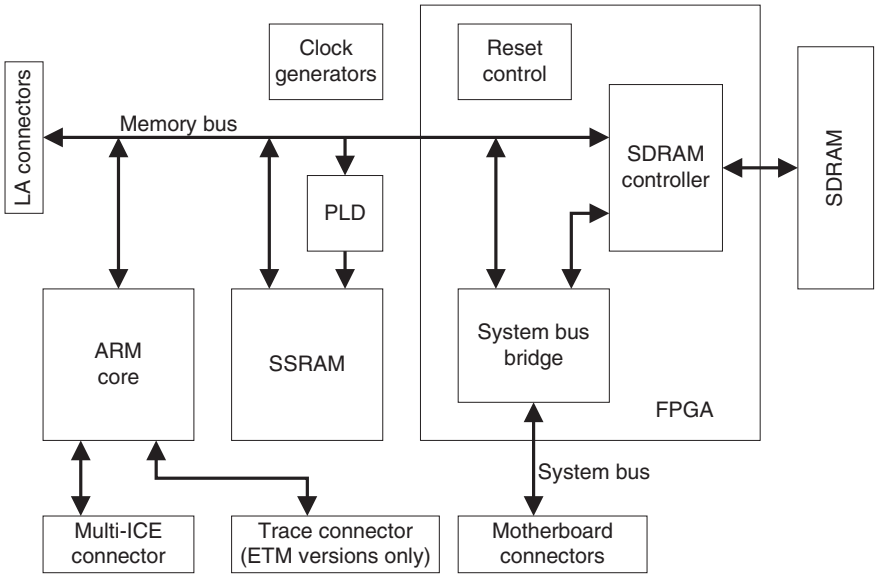


Figure 1-2 Core module block diagram

1.2.2 Core module FPGA

The FPGA provides system control functions for the core module, enabling it to operate as a standalone development system or attached to a motherboard. These functions are outlined in this section and described in detail in Chapter 3 *Hardware Description*.

Note

At power on, the FPGA is loaded with one of three images provided in the core module configuration flash. The functionality of these images is different. For more details see Chapter 5 *Using Core Modules with an Integrator/AP* and Chapter 6 *Using Core Modules with an Integrator/CP*.

SDRAM controller

The SDRAM controller is implemented within the FPGA. This provides support for *Dual In-line Memory Modules* (DIMMs) with a capacity of 16–256MB. See *SDRAM controller* on page 3-8.

Reset controller

The reset controller initializes the core. The core module can be reset from five sources:

- reset button
- motherboard
- other core modules or logic modules
- Multi-ICE
- software.

For information about the reset controller, see *Reset controller for AP* on page 5-18 and *Reset controller for CP* on page 6-21.

System bus bridge

The system bus bridge provides an AMBA interface between the memory bus on the core module and the system bus on a motherboard. It enables the processor to access resources on the motherboard and on other modules.

For the Integrator/AP, it also enables other masters to access the core module SDRAM (see *System bus bridge for AP* on page 5-10).

Status and configuration space

The status and configuration space contains status and configuration registers for the core module. These provide the following information and control:

- the manufacturer of the test chip
- the position of the core module in a multi-module stack
- SDRAM size, address configuration, and CAS latency setup
- core module clock speed and configuration
- interrupt control for the processor debug communications channel.

The status and control registers can only be accessed by the local processor. For more information about the status and control registers see Chapter 4 *Programmer's Reference*.

1.2.3 Volatile memory

The volatile memory system includes an SSRAM device, and a plug-in SDRAM memory module (referred to as *local* SDRAM when it is on the same core module as the processor). These areas of memory are closely coupled to the processor core to improve performance. The core module uses separate memory and system buses to avoid memory access performance being degraded by bus loading.

The SDRAM controller is implemented within the core module controller FPGA and a separate SSRAM controller is implemented with a *Programmable Logic Device* (PLD).

The SDRAM can be accessed by the local processor, by processors on other core modules, and by other system bus masters.

The SSRAM can only be accessed by the local processor.

1.2.4 Clock generator

The core module uses four clock signals:

REFCLK A fixed frequency 24MHz signal that can be used by the FPGA to generate real-time delays.

CORECLK A programmable frequency clock input to the ARM test chip.

LCLK A programmable frequency clock for the local memory bus.

AUXCLK A programmable frequency clock on the Integrator/CP. On the Integrator/AP, this is reserved for future use.

The programmable clocks are supplied by three clock generator chips. Their frequencies are selected in oscillator control registers within the FPGA. A reference clock is supplied to the clock generators and to the FPGA (see *Clock generators* on page 3-10).

If used with the Integrator/AP, the memory bus and system bus are asynchronous. This enables each to be run at the speed of its slowest device without compromising the performance of other buses in the system.

If used with the Integrator/CP, the memory bus and system bus are synchronous.

1.2.5 Multi-ICE connector

The Multi-ICE connector enables JTAG hardware debugging equipment, such as Multi-ICE, to be connected to the core module. It is possible to both drive and sense the system-reset line (**nSRST**), and to drive JTAG reset (**nTRST**) to the core from the Multi-ICE connector. See *Multi-ICE support* on page 3-16.

1.2.6 CP peripherals

The core module FPGA image for use with an Integrator/CP contains additional controllers for peripherals on the CP baseboard. For more information, see Chapter 6 *Using Core Modules with an Integrator/CP*.

1.3 Links and indicators

The core module provides one link and four surface-mounted LEDs. These are shown in Figure 1-3.

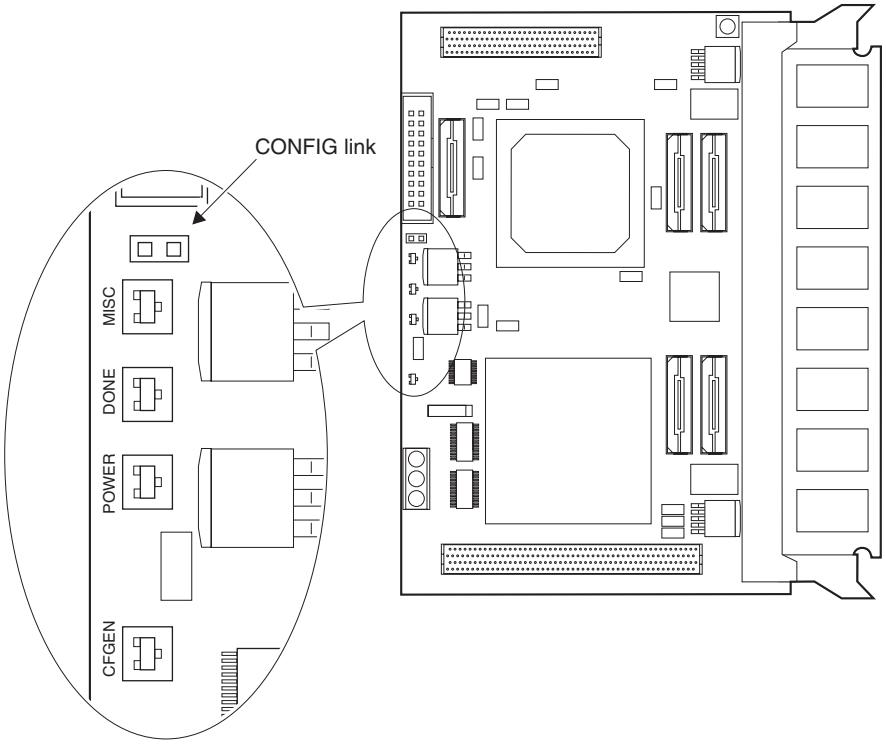


Figure 1-3 Links and indicators

1.3.1 CONFIG link

The core module has only one link, marked CONFIG. This is left open during normal operation. It is only fitted when downloading new FPGA and PLD configuration information.

1.3.2 LED indicators

The functions of the four surface-mounted LEDs are summarized in Table 1-1.

Table 1-1 LED functional summary

Name	Color	Function
MISC	Green	This LED is controlled using the control register (see *Core Module Control Register* on page 4-12)
DONE	Green	This LED illuminates when the FPGA has successfully loaded its configuration information following power-on
POWER	Green	This LED illuminates to indicate that a 3.3V supply is present
CFGLED	Orange	This LED illuminates to indicate that the CONFIG link is fitted

1.4 Test points

The core module provides test points and ground points to aid diagnostics. The most useful of these are shown in Figure 1-4.

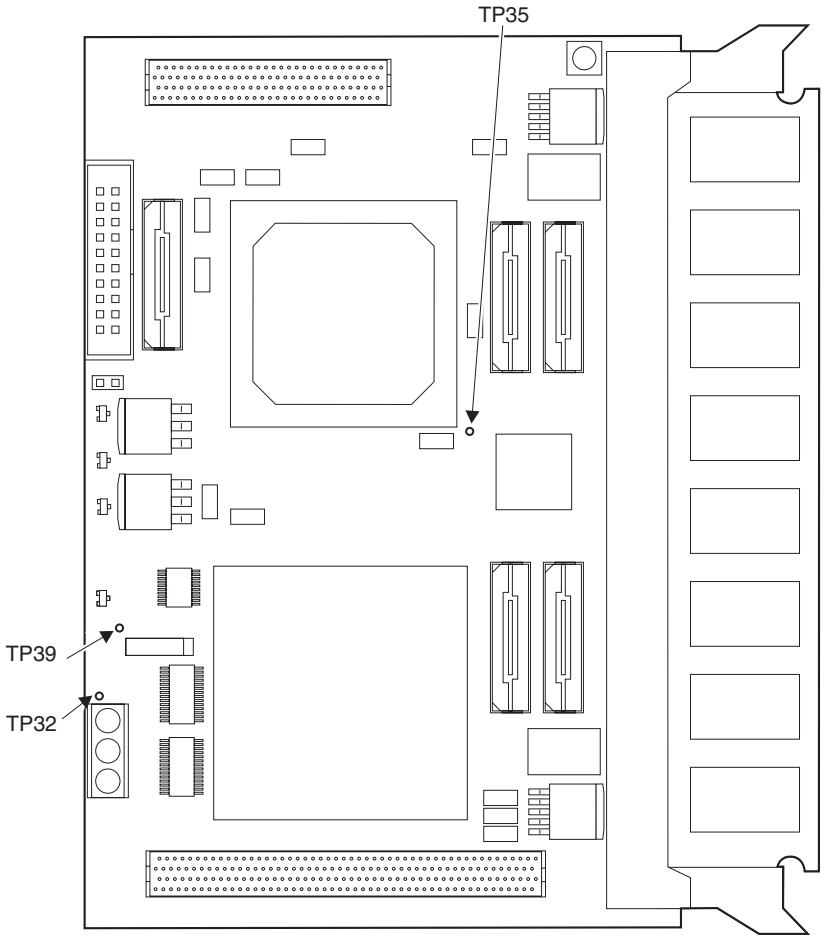


Figure 1-4 Test points

The functions of these test points are summarized in Table 1-2 on page 1-11. For information about setting the frequency of the core clock and auxiliary clock, see *Clock generators* on page 3-10.

Table 1-2 Test point functions

Test point	Name	Function
TP35	FCLKOUT	Clock output from the microprocessor core
TP32	AUXCLK	Auxiliary clock
TP39	REFCLK	Reference clock (24MHz)

1.5 Precautions

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

1.5.1 Ensuring safety

The core module is powered from 3.3V and 5V DC supplies.

Warning

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

1.5.2 Preventing damage

The core module is intended for use within a laboratory or engineering development environment. It is supplied without an enclosure and this leaves the board sensitive to electrostatic discharges and allows electromagnetic emissions.

Caution

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

- never subject the board to high electrostatic potentials.
- always wear a grounding strap when handling the board.
- only hold the board by the edges.
- avoid touching the component pins or any other metallic element.

Caution

Do not use the board near equipment that is:

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

Chapter 2
Getting Started

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

2.1 Setting up a standalone core module

To set up the core module as a standalone development system:

1. Optionally, fit an SDRAM DIMM.
2. Supply power.
3. Connect Multi-ICE.

2.1.1 Fitting an SDRAM DIMM

You can fit the following type of SDRAM module:

- PC66, PC100, or PC133- compliant 168-pin DIMM
- unbuffered
- 3.3V
- 16MB, 32MB, 64MB, 128MB or 256MB.

To install an SDRAM DIMM:

1. Ensure that the core module is powered down.
2. Open the SDRAM retaining latches outwards.
3. Press the SDRAM module into the edge connector until the retaining latches click into place.

Note

The DIMM edge connector has polarizing notches to ensure that it is correctly oriented in the socket.

2.1.2 Using the core module without SDRAM

You can operate the core module without SDRAM because it has 1MB of SSRAM permanently fitted. When using ADW or AXD, you can adjust the `top_of_memory` internal variable from its default value to `0x100000`.

For further information about ARM debugger internal variables, refer to the *ADS Debuggers Guide*.

2.1.3 Supplying power

When using the core module as a standalone development system, you must connect a bench power supply with 3.3V and 5V outputs to the power connector, as illustrated in Figure 2-1.

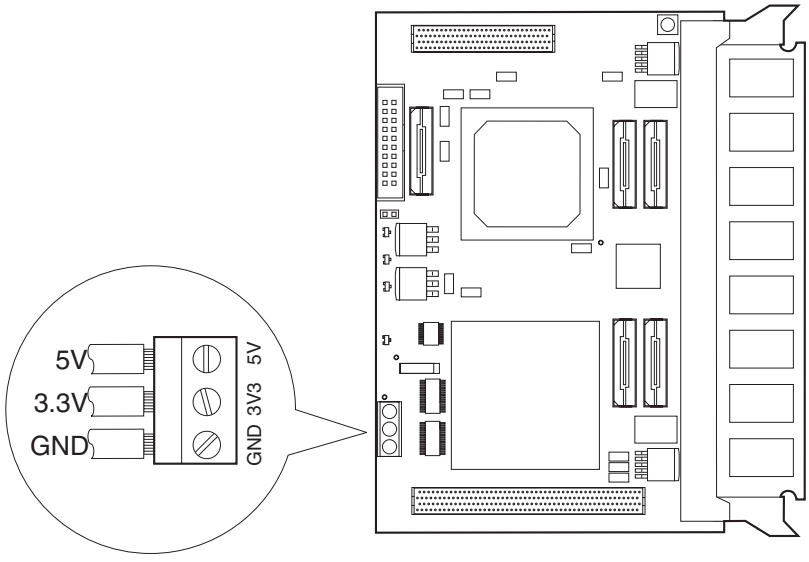


Figure 2-1 Power connector

Note

This power connection must not be used if the core module is fitted to a motherboard.

2.1.4 Connecting Multi-ICE and Trace

When you are using the core module as a standalone system, Multi-ICE debugging equipment can be used to download programs. The Multi-ICE setup for a standalone core module is shown in Figure 2-2.

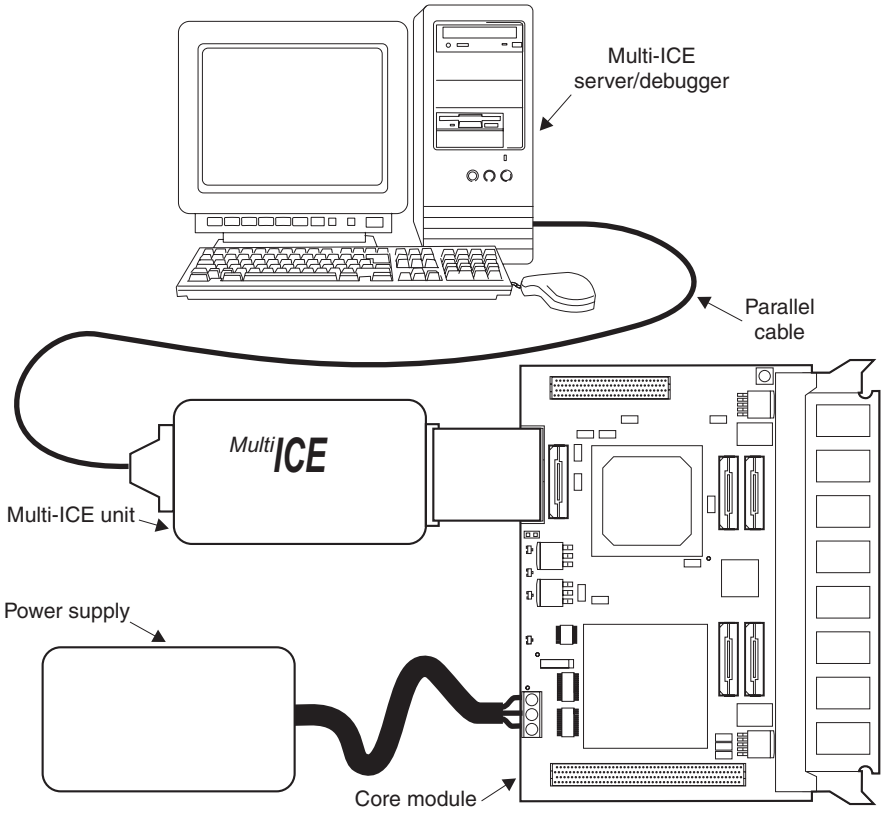


Figure 2-2 Multi-ICE connection to a core module

Caution

Because the core module does not provide nonvolatile memory, programs are lost when the power is removed.

You can also use Multi-ICE when a core module is attached to a motherboard. If more than one core module is attached, then the Multi-ICE unit must be connected to the module at the top of the stack. The Multi-ICE server and the debugger can be on one computer or on two networked computers.

If you are using Trace for the CM920T-ETM, connect the trace port adapter board to the core module as shown in Figure 2-3.

Note

The CM920T and CM940T core modules do not have an ETM or a trace connector fitted.

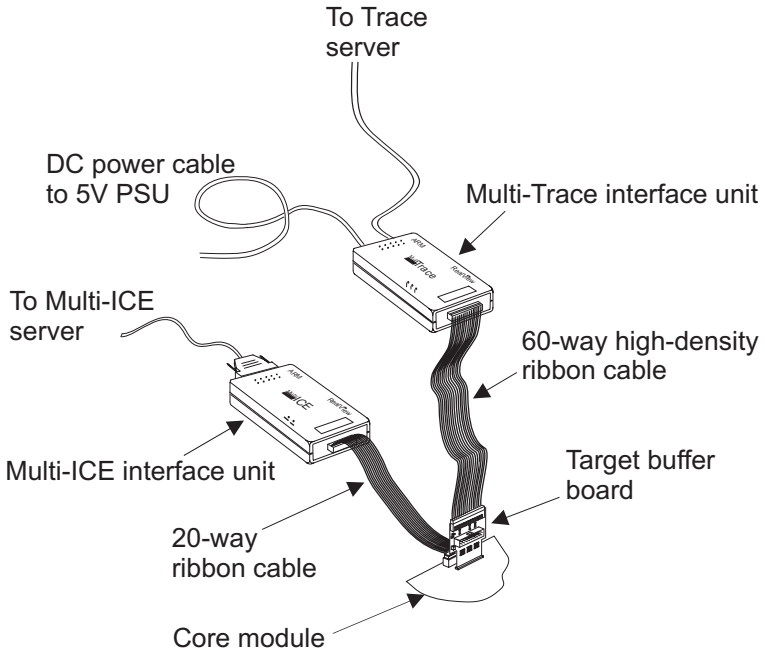


Figure 2-3 Connecting Trace

You can connect the Multi-ICE connector directly to the trace port adapter board as shown in Figure 2-4 on page 2-6, but be careful to avoid putting too much pressure on the trace port socket.

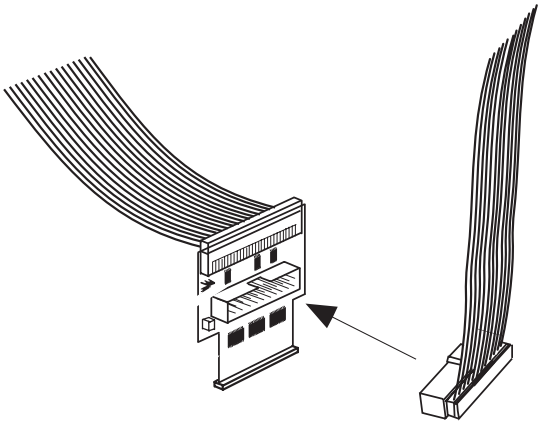


Figure 2-4 Connecting Multi-ICE to the trace port adapter board

2.2 Attaching the core module to an Integrator/AP motherboard

Attach the core module onto a motherboard (for example, the ARM Integrator/AP) by engaging the connectors HDRA and HDRB on the bottom of the core module with the corresponding connectors on the top of the motherboard. The lower side of the core module has sockets and the upper side of the core module has plugs to enable core modules to be mounted on top of one another. You can stack a maximum of four core modules on an Integrator/AP.

Figure 2-5 shows an example development system with four core modules attached to an ARM Integrator/AP motherboard.

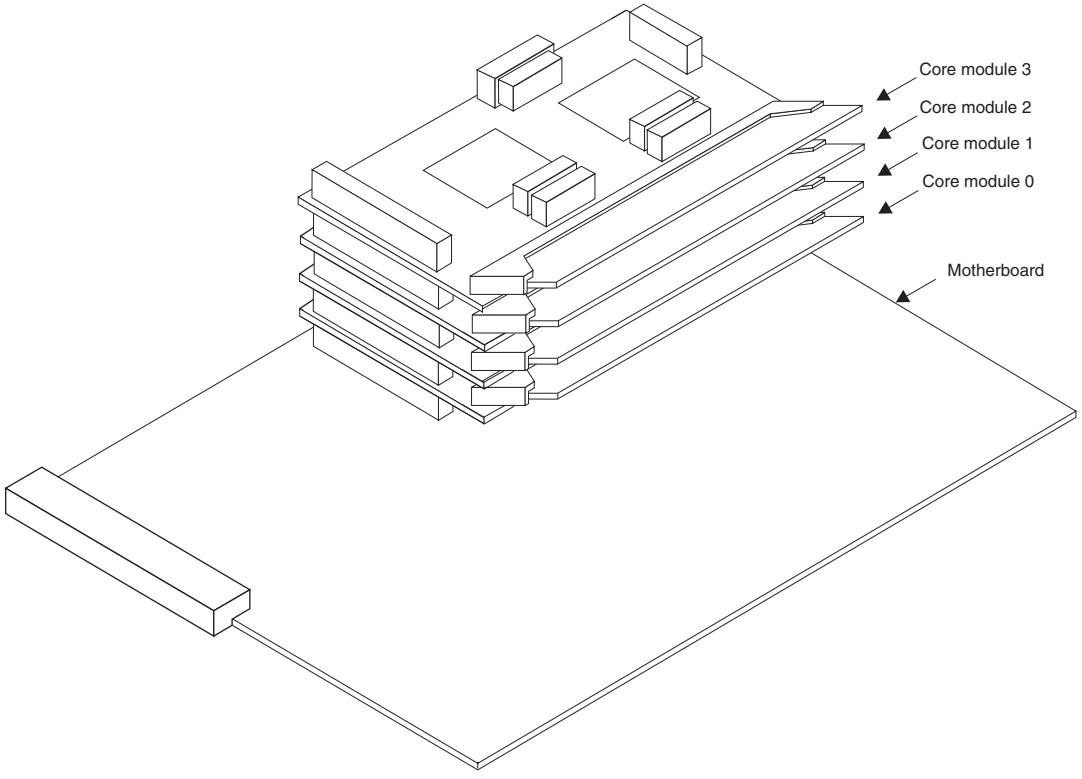


Figure 2-5 Assembled Integrator/AP system

Note

For correct operation of the core module, do not use the core module in the EXPA/EXPB stack position on the Integrator/AP.

2.2.1 Core module ID

The ID of the core module is configured automatically by the connectors (there are no links to set) and depends on its position in the stack:

- core module 0 is installed first
- core module 1 is installed next, and cannot be fitted without core module 0
- core module 2 is installed next, and cannot be fitted without core module 1
- core module 3 is installed next, and cannot be fitted without core module 2.

The ID of the core module also defines the ID of the microprocessor it carries and the system bus address of its SDRAM. The mechanism that controls the ID and mapping of the core module is described in *Module ID selection for AP* on page 5-3. The position of a core module in the stack can be read from the CM_STAT register (see *Core Module Status Register* on page 4-14).

2.2.2 SDRAM, Trace, and Multi-ICE

For details on adding expansion SDRAM, see *Fitting an SDRAM DIMM* on page 2-2.

For details on connecting a trace probe or Multi-ICE, see *Connecting Multi-ICE and Trace* on page 2-4.

2.2.3 Powering the assembled Integrator development system

Power the assembled Integrator development system by:

- connecting a bench power supply to the motherboard
- installing the motherboard in a card cage or an ATX-type PC case, depending on type.

For further information, refer to the *Integrator/AP User Guide*.

2.3 Attaching the core module to an Integrator/CP baseboard

Attach the core module onto an Integrator/CP baseboard by engaging the connectors HDRA and HDRB on the bottom of the core module with the corresponding connectors on the top of the baseboard. The lower side of the core module has sockets and the upper side of the core module has plugs to enable modules to be mounted on top of one another. Only one core module can be stacked on the baseboard, but up to three logic modules can be used.

Figure 2-6 shows an example development system.

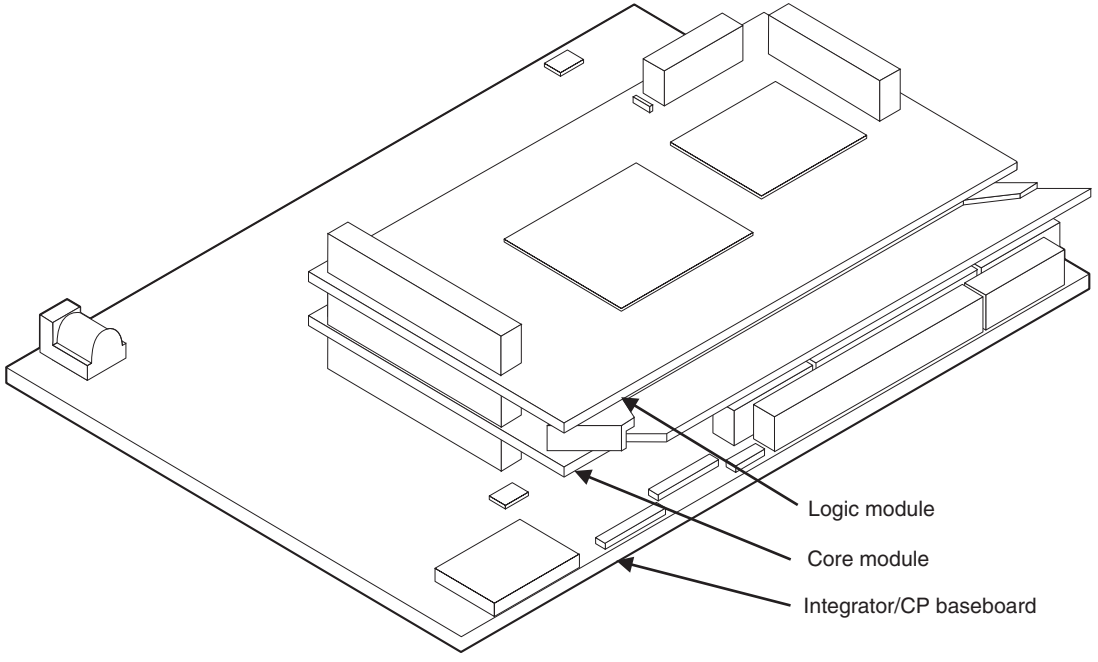


Figure 2-6 Assembled Integrator/CP development system

Note

To ensure reliable operation of the core module:

- Ensure that the core module loads a CP-compatible image to the FPGA. Current core modules are shipped with images for both the AP and CP boards and the correct image is automatically selected.
- Do not connect power to the core module. Apply power to the baseboard.

2.3.1 FPGA image

The core module hardware is the same for mounting the module on an Integrator/AP or Integrator/CP. The appropriate FPGA image for the Integrator/CP is loaded from the configuration flash. The FPGA image includes implementations of peripherals that drive hardware on the Integrator/CP baseboard.

Note

Some earlier versions of core modules do not contain the configuration image for the Integrator/CP. Contact your supplier for the updated FPGA image.

2.3.2 Core module ID

Only one core module can be placed on the Integrator/CP and it must be the first module in the stack. For the Integrator/CP, the core module addresses are fixed.

The position of a logic module in the stack determines its address range. The ID of the module is configured automatically by the connectors (there are no links to set).

Note

Core modules installed on the Integrator/CP are always core module 0. See Chapter 6 *Using Core Modules with an Integrator/CP* for more details.

2.3.3 SDRAM, Trace, and Multi-ICE

For details on adding expansion SDRAM, see *Fitting an SDRAM DIMM* on page 2-2.

For details on connecting a trace probe or Multi-ICE, see *Connecting Multi-ICE and Trace* on page 2-4.

2.3.4 Powering the assembled Integrator/CP development system

Power the assembled Integrator/CP development system by:

- connecting a bench power supply to the CP baseboard (not to the core module)
- connect the supplied power supply to the CP baseboard.

For further information, see the *Integrator/CP Baseboard User Guide*.

Chapter 3
Hardware Description

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

Note

This chapter describes the generic hardware and is independent of the FPGA image used. For details of register usage that is dependent on the FPGA image, see Chapter 5 *Using Core Modules with an Integrator/AP* and Chapter 6 *Using Core Modules with an Integrator/CP*.

3.1 ARM microprocessor test chip

The ARM920T, ARM920T-ETM, and ARM940T cores are members of the ARM9 Thumb family of processor cores. They incorporate the following features:

- ARM9TDMI 32-bit RISC core
- Harvard architecture
- Thumb 16-bit compressed instruction set for higher code density
- JTAG interface to EmbeddedICE logic
- AMBA bus interface.
- *System-on-Chip* (SoC) integration features for embedded production test.

The ARM9TDMI core executes both the 32-bit ARM and 16-bit Thumb instruction sets. This enables you to switch between high-performance operation and high code density. The ARM9TDMI™ is user code binary-compatible with ARM7TDMI™, ARM10TDMI™, and StrongARM® processors, and is supported by a wide range of tools, operating systems, and application software.

In addition to the ARM9TDMI core, the ARM920T, ARM920T-ETM, and ARM940T processors feature:

- separate instruction and data caches provide write-through or write-back operation under software control on a per-region basis
  - 16K/16K for the ARM920T and ARM920T-ETM
  - 4K/4K for the ARM940T
- write buffer
- memory access control
  - *Memory Management Unit* (MMU) for the ARM920T and ARM920T-ETM
  - *Memory Protection Unit* (MPU) for the ARM940T
- fast context switching and high vector extensions
- ETM for real-time debugging of applications (ARM920T-ETM only, the ETM size depends on the test chip fitted)
- AMBA bus interface.

The MMU on the ARM920T and ARM920T-ETM features separate instruction and data *Translation Lookaside Buffers* (TLBs). It supports tiny (1KB) page mapping to enable a large number of small objects to be implemented as required by some operating systems, such as WindowsCE. WindowsCE is also supported by fast context switching and high vector extensions.

For more information about the cores, refer to:

- *ARM920T Technical Reference Manual*
- *ARM940T Technical Reference Manual.*

3.2 SSRAM controller

The SSRAM controller is implemented in a Xilinx 9572XL PLD that enables the SSRAM to achieve single-cycle operation. In addition to controlling accesses to the SSRAM, the controller generates the processor response signals (**BWAIT**, **BERROR**, and **BLAST**) for all accesses to:

- SSRAM
- SDRAM
- status and configuration register space
- system bus bridge.

Note

The **BERROR** signal is generated by the PLD for accesses above `0x11000000` and no motherboard present. If a motherboard is present, the motherboard FPGA controls the generation of **BERROR**.

3.3 Core module FPGA

At power-up the FPGA loads its configuration data from a flash memory device. Parallel data from the flash is serialized by the *Programmable Logic Device* (PLD) into the configuration inputs of the FPGA. Figure 3-1 shows the FPGA configuration mechanism.

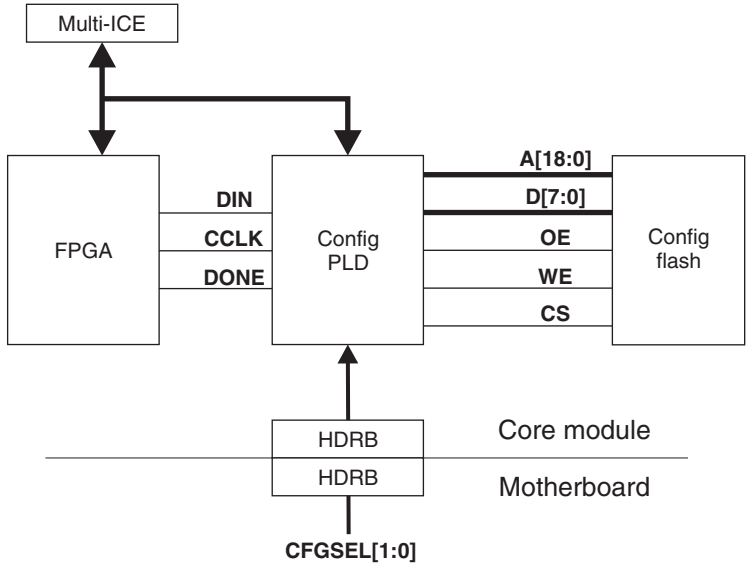


Figure 3-1 FPGA configuration

The following sections describe functional blocks implemented in the FPGA:

- *Core module control registers* on page 4-9
- *Core module debug comms interrupt registers* on page 4-22.

The FPGA image also has functional blocks specific to the development environment.

You can use Multi-ICE to reprogram the PLD, FPGA, and flash when the core module is placed in configuration mode (see *Multi-ICE support* on page 3-16).

Core modules might be fitted with either an XCV600 or XCV600E FPGA. Ensure that the appropriate FPGA image is loaded into the configuration flash.

3.3.1 Integrator/AP image

The FPGA provides sufficient functionality for the core module to operate as a standalone development system, although with limited capabilities.

For the Integrator/AP, the FPGA image also contains a system bus bridge. See *System bus bridge for AP* on page 5-10.

System bus arbitration, system interrupt control, and input/output resources are provided by the system controller FPGA on the motherboard. See the *Integrator/AP User Guide* for further information.

Figure 3-2 shows the function of the core module FPGA (for standalone or Integrator/AP use) and how it connects to the other devices in the system.

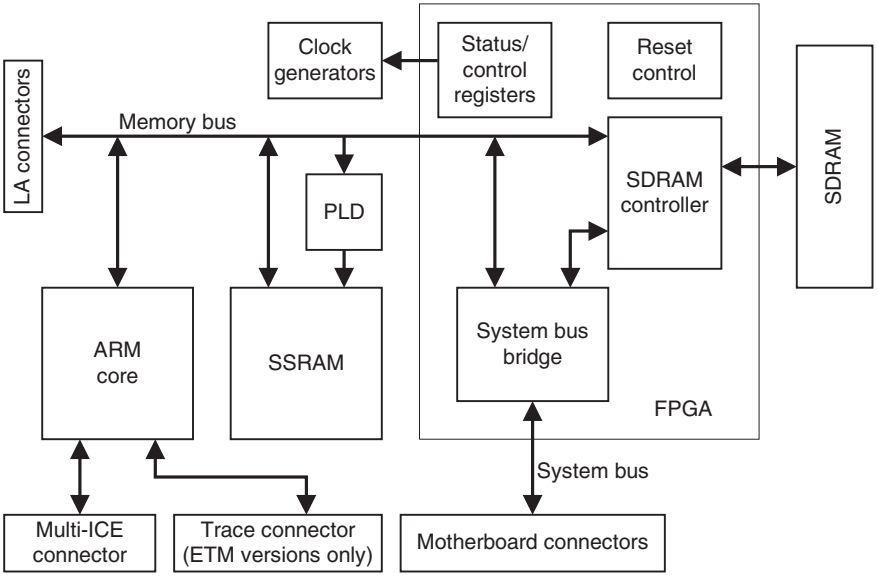


Figure 3-2 FPGA functional diagram

3.3.2 Integrator/CP image

For the Integrator/CP, the FPGA image contains additional peripherals, the LCD controller for example. See *Programmable logic on CP* on page 6-10 for more details.

A primary interrupt controller is provided by the FPGA on the core module. A secondary interrupt controller is provided by the baseboard PLD. See the *Integrator/CP User Guide* for further information.

Note

Using core modules with the Integrator/CP requires a compatible FPGA image. See Chapter 6 *Using Core Modules with an Integrator/CP* for more details.

3.3.3 FPGA image selection

The config flash contains multiple images that enable the FPGA to be configured to support an AHB or ASB motherboard. Image selection is controlled by the static configuration select signals **CFGSEL[1:0]** from the motherboard. The encoding of these signals is shown in Table 3-1.

Table 3-1 CFGSEL[1:0] encoding

CFGSEL[1:0]	Description
00	Little endian ASB Integrator/AP.
01	Reserved.
10	Little-endian AHB Integrator/AP, and bi-endian AHB standalone. If there is not a motherboard connected to the core module, pull-up and pull-down resistors on the core module select this as the default image selection code.
11	AHB Lite, bi-endian, Integrator/CP.

3.4 SDRAM controller

The core module provides support for a single 16, 32, 64, 128, or 256MB SDRAM DIMM.

3.4.1 SDRAM operating mode

The operating mode of the SDRAM devices is controlled with the mode set register within each SDRAM. These registers are set immediately after power-up to specify:

- a burst size of four for both reads and writes
- *Column Address Strobe* (CAS) latency of 2 cycles.

You can program the CAS latency and memory size using the SDRAM Control Register (CM_SDRAM) at address `0x10000020`. See *Core Module SDRAM Status and Control Register* on page 4-18.

Note

Before the SDRAM is used, it is necessary to read the SPD memory and program the CM_SDRAM Register with the parameters indicated in Table 4-8 on page 4-18. If these values are not correctly set then SDRAM accesses might be slow or unreliable.

3.4.2 Access arbitration

If used with the Integrator/AP, the SDRAM controller provides two ports to support reads and writes by the local processor core and by masters on the system bus. The SDRAM controller uses an alternating priority scheme to ensure that the processor core and other system bus masters have equal access (see *System bus bridge for AP* on page 5-10).

If used with the Integrator/CP, the SDRAM controller provides two ports to support reads and writes by the local processor core and reads by the CLCD controller in the core module FPGA. The CLCD controller has higher priority than the processor core.

3.4.3 Serial presence detect

JEDEC-compliant SDRAM DIMMs incorporate a *Serial Presence Detect* (SPD) feature. This comprises a 2048-bit serial EEPROM located on the DIMM with the first 128 bytes programmed by the DIMM manufacturer to identify the following:

- module type
- memory organization
- timing parameters.

The EEPROM clock (SCL) operates at 93.75kHz (24MHz divided by 256). The transfer rate for read accesses to the EEPROM is 100kbit/s maximum. The data is read out serially eight bits at a time, preceded by a start bit and followed by a stop bit. This makes reading the EEPROM a very slow process because it takes approximately 27ms to read all 256 bytes. However, during power-up the contents of the EEPROM are copied into a 64 x 32-bit area of memory (CM_SPD) within the SDRAM controller. The SPD flag is set in the SDRAM Control Register (CM_SDRAM) when the SPD data is available. This copy can be randomly accessed at `0x10000100–0x100001FC` (see *SDRAM SPD memory* on page 4-26).

Write accesses to the SPD EEPROM are not supported.

3.5 Clock generators

The clock generators provide three programmable clock sources and a fixed-frequency reference clock, **REFCLK**. These four clocks are not synchronous with each other.

If the core module is mounted on an Integrator/CP baseboard, the local memory bus clock generated in the core module is also used to clock the system bus.

If the core module is mounted on an Integrator/AP, the system bus is provided by the motherboard and is asynchronous to the local memory bus.

These clocks are discussed in the following subsections:

- *Clock generation functional overview* on page 3-11
- *Programming the processor core clock, CORECLK* on page 3-12
- *Programming the local memory bus clock, LCLK* on page 3-13.
- *Programming the auxiliary clock, AUXCLK* on page 3-14
- *FPGA reference clock, REFCLK* on page 3-15.

3.5.1 Clock generation functional overview

The architecture of the clock generators is shown in Figure 3-3.

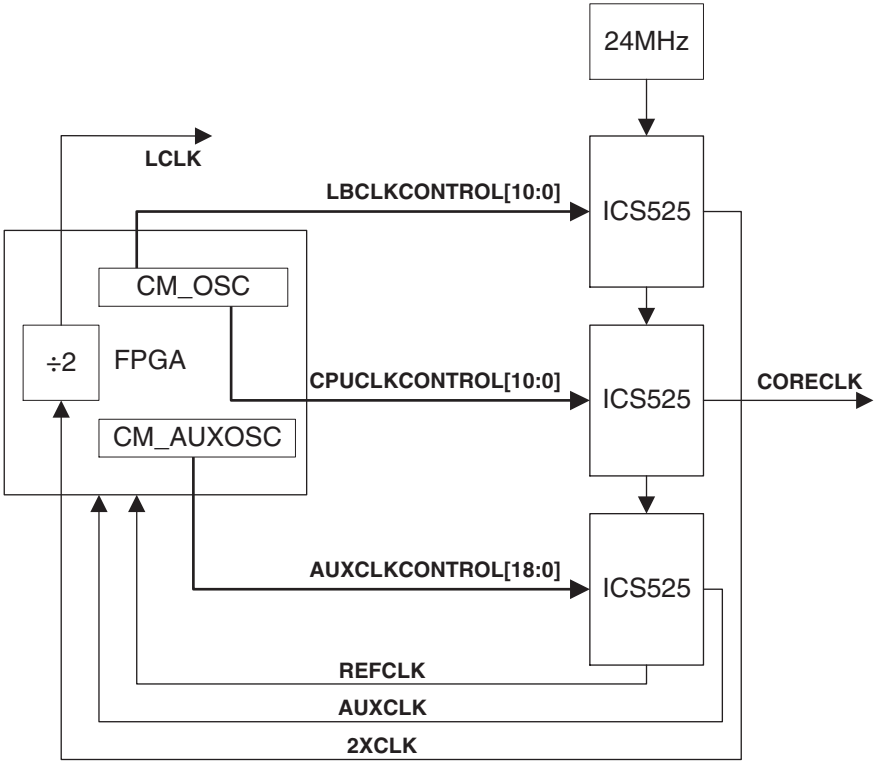


Figure 3-3 Core module clock generator

The core module has three Microclock OSCaR programmable clock generators. These are supplied with a reference clock by a 24MHz crystal oscillator and their output frequencies are controlled by *divider* input pins. They are able to produce a wide range of frequencies controlled by registers in the FPGA.

3.5.2 Programming the processor core clock, CORECLK

The frequency of **CORECLK** is controllable in 1MHz steps in the range 12MHz to 160MHz. This is achieved by setting the *Voltage Controlled Oscillator* (VCO) divider and output divider for the **CORECLK** generator in the CM_OSC Register. The VCO divider is controlled by the C_VDW bits and output divider is controlled by the C_OD bits. The reference divider value is fixed.

Figure 3-4 shows the values placed on the divider input pins and how the clock speeds are obtained.

C_RDW R[6:0]							C_VDW V[9:0]									C_OD S[2:0]		
0	0	1	0	1	1	0	0	C	C	C	C	C	C	C	C	C	C	C
22 (fixed value)							4 to 152									2 to 10		

Figure 3-4 CORECLK divider control

The bits marked:

- C are programmable in the CM_OSC Register
- 1 are tied HIGH
- 0 are tied LOW.

You can calculate the frequency of **CORECLK** using the formula:

$Freq_{MHz} = 2*((C\_VDW+ 8)/C\_OD)$

where:

C_VDW is the VCO divider word for the core clock.

C_OD is the output divider for the core clock.

Note

The output divider is not a straight forward binary representation of a decimal number. Its value is assigned as described in *Programming the output dividers* on page 3-15.

For details about programming the CM_OSC Register, see *Core Module Oscillator Register* on page 4-12.

Note

Values for C_VDW and C_OD can be calculated using the ICS525 calculator on the Microclock website.

The **CORECLK** is converted to the supply level for the test chip pads by the FPGA. The clock is series terminated with a 33Ω resistor and then drives a single load on the microprocessor core.

3.5.3 Programming the local memory bus clock, LCLK

The clock signal **2XCLK** is divided by 2 by the FPGA to produce **LCLK** that is distributed to the following devices:

- FPGA
- logic analyzer
- SDRAM DIMM
- SSRAM PLD
- the test chip (ARM920T, ARM920T-ETM, or ARM940T)

Note

The clock is fed back into the FPGA so that the internal version is not skewed with respect to the others.

The frequency of **LCLK** is controllable in 0.5MHz steps in the range 6MHz to 66MHz by programming the VCO and output divider bits for the **2XCLK** generator in the CM_OSC Register. The VCO divider is controlled by the L_VDW bits and the output divider is controlled by the L_OD bits. The reference divider is fixed. The default frequency for the core module is 40MHz. Figure 3-5 shows the values placed on the divider input pins and how the clock speeds are obtained.

L_RDW R[6:0]							L_VDW V[9:0]									L_OD S[2:0]		
0	0	1	0	1	1	0	0	L	L	L	L	L	L	L	L	L	L	L
22 (fixed value)							4 to 124									2 to 10		

Figure 3-5 2XCLK divider control

The bits marked:

- L are programmable in the CM_OSC Register
- 1 are tied HIGH
- 0 are tied LOW.

You can calculate the frequency of the **LCLK** using the simplified formula:

$Freq_{MHz} = (L\_VDW + 8)/L\_OD$

where:

L_VDW is the VCO divider word.

L_OD is the output divider.

Note

The output divider is not a straight forward binary representation of a decimal number. Its value is assigned as described in *Programming the output dividers* on page 3-15.

For details about programming L_VDW and L_OD, see *Core Module Oscillator Register* on page 4-12.

3.5.4 Programming the auxiliary clock, AUXCLK

The frequency of the **AUXCLK** is controlled by CM_AUXOSC (see *Core Module Auxiliary Oscillator Register* on page 4-16). The values for RDW, VDW, and OD are all programmable, as shown in Figure 3-6, to give frequencies in the range 1 to 160MHz with a better than 0.1% accuracy. The default frequency following a power-on reset is 32.369MHz.

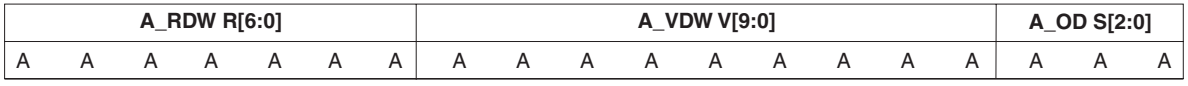


Figure 3-6 AUXCLK divider control

The clock frequency is controlled by the divider signals **AUXCLKCONTROL[18:0]** that are assigned to the control parameters as shown in Table 3-2 on page 3-15.

Table 3-2 Clock control signal assignment

Signals	Control parameter	Label
AUXCLKCONTROL[18:16]	Output divider	A_OD
AUXCLKCONTROL[15:9]	Reference divider	A_RDW
AUXCLKCONTROL[8:0]	VCO divider	A_VDW

The reference divider and VCO divider are used to calculate the output frequency using the following formula:

$$\text{Freq} = 48\text{MHz} \times \frac{(\text{A\_VDW} + 8)}{(\text{A\_RDW} + 2) \times \text{A\_OD}}$$

3.5.5 Programming the output dividers

The output dividers, C_OD, L_OD, and A_OD, are not straightforward binary representations of decimal values but have assigned values. The value assigned to each bit pattern is shown in Table 3-3.

Table 3-3 Values for output divider

OD[2:0]	Value
001	2
011	4
100	5
111	6
101	7
010	8
110	9
000	10

3.5.6 FPGA reference clock, REFCLK

The **REFCLK** signal is used by the FPGA to generate the SDRAM refresh clock and SPD EEPROM clock.

For the Integrator/CP, REFCLK also clocks timers 1 to 3. Timer 0 is clocked by the local memory bus clock.

3.6 Multi-ICE support

The core module supports debugging using JTAG. This is described in the following subsections:

- *Multi-ICE connection*
- *JTAG scan paths* on page 3-17
- *JTAG connection modes* on page 3-20
- *JTAG signals* on page 3-21.

3.6.1 Multi-ICE connection

Figure 3-7 shows the Multi-ICE connector and the CONFIG link.

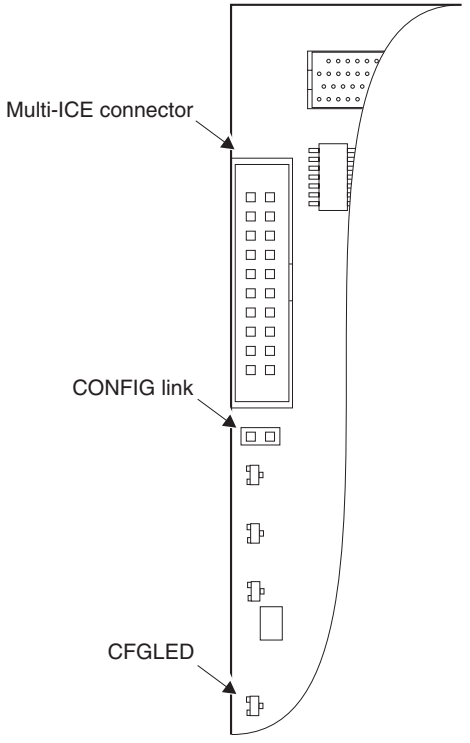


Figure 3-7 JTAG connector, CONFIG link, and LED

The CONFIG link is used to enable in-circuit programming of the FPGA and PLDs using Multi-ICE (see *JTAG connection modes* on page 3-20).

The Multi-ICE connector provides a set of JTAG signals that enable JTAG debugging equipment to be used (see *JTAG signals* on page 3-21). If you are debugging a development system with multiple core modules, connect the JTAG debugging equipment to the top core module.

3.6.2 JTAG scan paths

This section describes:

- *JTAG data path*
- *JTAG clock path* on page 3-18.

JTAG data path

Figure 3-8 shows a simplified diagram of the data path.

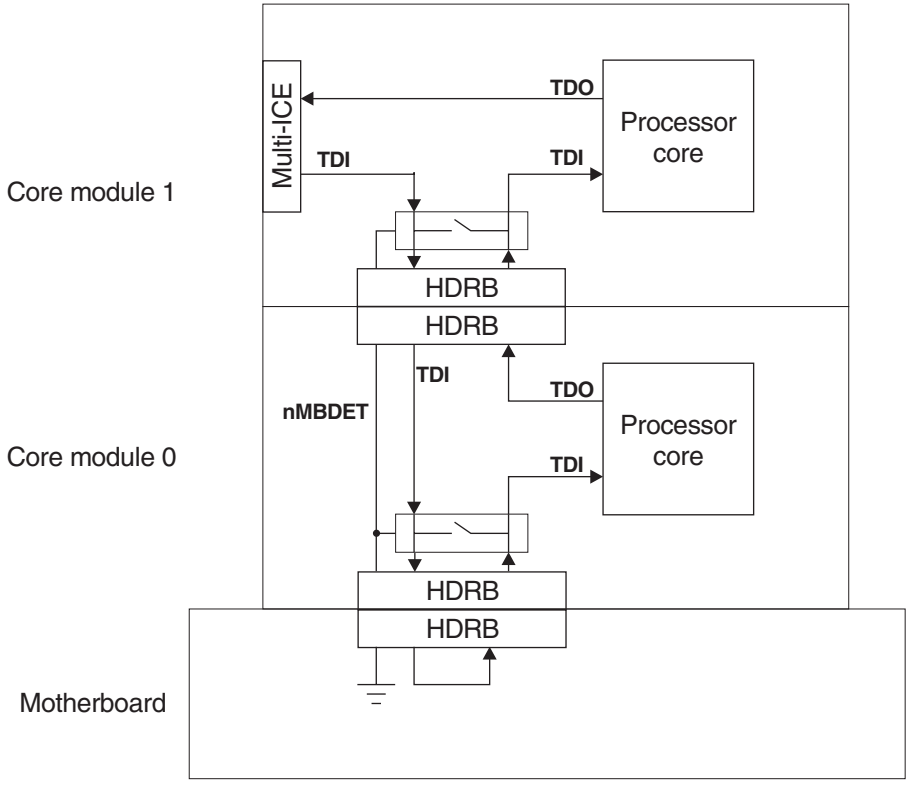


Figure 3-8 JTAG data path

Note

You can connect only one core module to the Integrator/CP baseboard. However, up to three logic modules can be added on top of the core module.

When you use the core module as a standalone development system, the data path is routed to the processor core and back to the Multi-ICE connector.

If the core module is attached to an Integrator motherboard, the **TDI** signal from the top core module is routed down through the HDRB connectors of any modules in the stack to the motherboard. From there the path is routed back up the stack through each core module, before being returned to the Multi-ICE connector as **TDO**. The motherboard detect signal **nMBDET** controls a switching circuit on the core module and, therefore, the routing of **TDI**.

The PLDs and FPGAs are included in the scan chain if the core module is in configuration mode, as described in *JTAG connection modes* on page 3-20.

JTAG clock path

The clock path is routed in a similar way to the data path, although in the opposite direction. Figure 3-9 on page 3-19 shows a simplified diagram of the clock path.

Several synthesized cores (for example, the ARM966E-S) sample **TCK**. This introduces a delay into the clock path. Cores of this type pass on the delayed clock signal as **RTCK**, which is fed to the **TCK** input of the next device in the chain. The **RTCK** signal at the Multi-ICE connector is used by Multi-ICE to regulate the advance of **TCK**, a mechanism called adaptive clocking (see the *ARM Multi-ICE User Guide*).

The routing of the **TCK/RTCK** signals through the stack is controlled by switches in a similar way to the data path. The routing of **RTCK** back up the stack is controlled by the signal **nRTCKEN** and an AND gate on the motherboard (the pullups on **nMBDET** are omitted for clarity).

The ARM920T, ARM920T-ETM, and ARM940T do not sample **TCK** but route the **TCK** signal straight through to the next board down the stack. If one or more modules in a stack drives **RTCK** (and so asserts **nRTCKEN**), you must ensure that the board at the bottom of the stack provides the necessary return path. All Integrator motherboards do so.

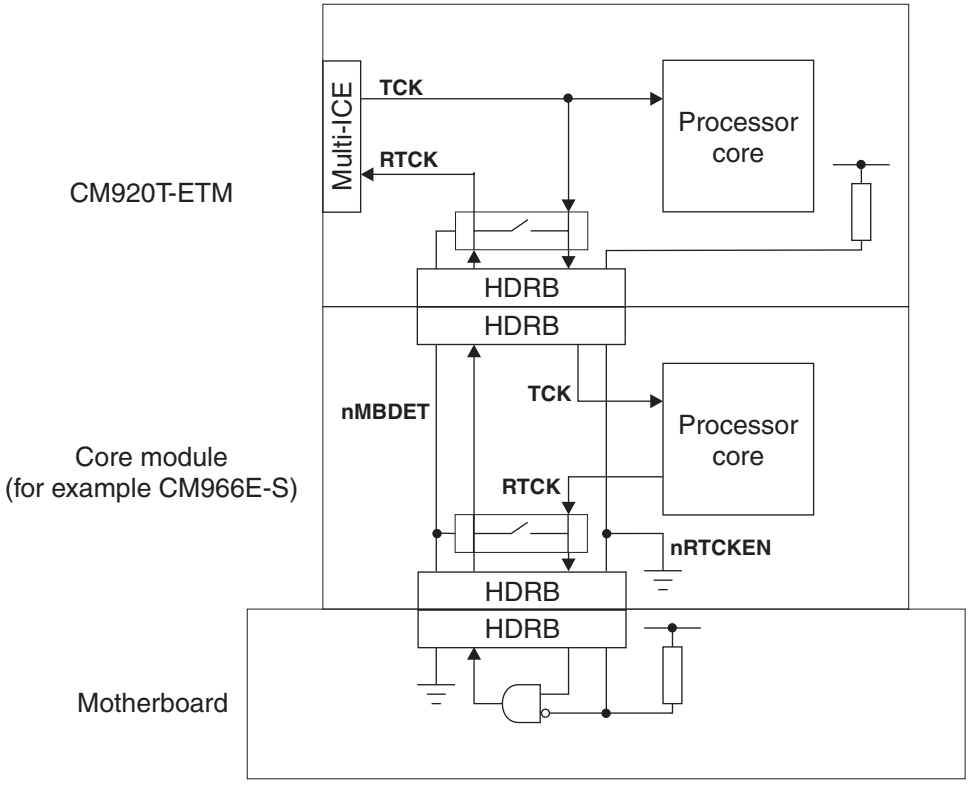


Figure 3-9 JTAG clock path

Note

You can connect only one core module to the Integrator/CP baseboard. However, up to three logic modules can be added on top of the core module.

3.6.3 JTAG connection modes

The core module is capable of operating in normal debug mode or configuration mode.

Normal debug mode

During normal operation and software development, the core module operates in debug mode. The debug mode is selected by default (when a jumper is *not* fitted at the CONFIG link, see Figure 3-7 on page 3-16). In this mode, the processor core and debuggable devices on other modules in the stack are accessible on the scan chain, as shown in Figure 3-8 on page 3-17.

Configuration mode

In configuration mode the debuggable devices are still accessible and, in addition, all FPGAs and PLDs in the system are added into the scan chain. This enables the board to be configured or upgraded in the field using Multi-ICE or other JTAG debugging equipment.

To select configuration mode, fit a jumper to the CONFIG link on the core module *at the top of the stack (*see Figure 3-7 on page 3-16*)*. This has the effect of pulling the **nCFGEN** signal LOW and illuminating the CFG LED (yellow) on each module in the stack and rerouting the JTAG scan path. The LED provides a warning that the development system is in the configuration mode.

Note

Configuration mode is guaranteed for a single core module attached to a motherboard but might be unreliable if more than one core module is attached. The larger loads on the **TCK** and **TMS** lines can cause unreliable operation.

After configuration or code updates you must:

1. Remove the CONFIG link.
2. Power cycle the development system.

The configuration mode enables FPGA and PLD code to be updated as follows:

- The FPGAs are volatile, but load their configuration from flash memory. Flash memory, which itself does not have a JTAG port, can be programmed by loading designs into the FPGAs and PLDs which handle the transfer of data to the flash using JTAG.
- The PLDs are nonvolatile devices that can be programmed directly by JTAG.

3.6.4 JTAG signals

Figure 3-10 shows the pinout of the Multi-ICE connector and Table 3-4 on page 3-22 provides a description of the JTAG and related signals.

Note

In the description in Table 3-4 on page 3-22, the term JTAG equipment refers to any hardware that can drive the JTAG signals to devices in the scan chain. Typically this is Multi-ICE, although hardware from other suppliers can also be used to debug ARM processors.

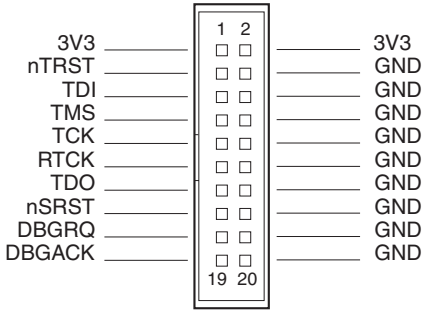


Figure 3-10 Multi-ICE connector pinout

Table 3-4 JTAG signal description

Name	Description	Function
DBGRQ	Debug request (from JTAG equipment)	**DBGRQ** is a request for the processor core to enter the debug state. It is provided for compatibility with third-party JTAG equipment.
DBGACK	Debug acknowledge (to JTAG equipment)	**DBGACK** indicates to the debugger that the processor core has entered debug mode. It is provided for compatibility with third-party JTAG equipment.
DONE	FPGA configured	**DONE** is an open-collector signal that indicates when FPGA configuration is complete. Although this signal is not a JTAG signal, it does affect **nSRST**. The **DONE** signal is routed between all FPGAs in the system through the HDRB connectors. The master reset controller on the motherboard senses this signal and holds all the boards in reset (by driving **nSRST** LOW) until all FPGAs are configured.
nCFGEN	Configuration enable (from jumper on module at the top of the stack)	**nCFGEN** is an active LOW signal used to put the boards into configuration mode. The **nCFGEN** signal is routed between all FPGAs in the system through the HDRB connectors. In configuration mode all FPGAs and PLDs are connected to the scan chain so that they can be configured by the JTAG equipment.
nRTCKEN	Return TCK enable (from core module to motherboard)	**nRTCKEN** is an active LOW signal driven by any core module that requires **RTCK** to be routed back to the JTAG equipment. If **nRTCKEN** is HIGH, the motherboard drives **RTCK** LOW. If **nRTCKEN** is LOW, the motherboard drives the **TCK** signal back up the stack to the JTAG equipment. The **nCFGEN** signal is routed between all FPGAs in the system through the HDRB connectors.
nSRST	System reset (bidirectional)	**nSRST** is an active LOW open-collector signal that can be driven by the JTAG equipment to reset the target board. Some JTAG equipment senses this line to determine when a board has been reset by the user. When the signal is driven LOW by the reset controller on the core module, the motherboard resets the whole system by driving **nSYSRST** LOW. This is also used in configuration mode to control the initialization pin, **nINIT** on the FPGAs. Though not a JTAG signal, **nSRST** is described because it can be controlled by JTAG equipment.

Table 3-4 JTAG signal description (continued)

Name	Description	Function
nTRST	TAP reset (from JTAG equipment)	This active LOW open-collector is used to reset the JTAG port and the associated debug circuitry on the ARM920T, ARM920T-ETM, or ARM940T processor. It is asserted at power-up by each module, and can be driven by the JTAG equipment. This signal is also used in configuration mode to control the programming pin, **nPROG** on FPGAs.
RTCK	Return **TCK** (to JTAG equipment)	Some devices sample **TCK** (for example a synthesizable core with only one clock), and this has the effect of delaying the time at which a component actually captures data. Using a mechanism called *adaptive clocking* , the **RTCK** signal is returned by the core to the JTAG equipment, and the clock is not advanced until the core has captured the data. In *adaptive clocking mode*, Multi-ICE waits for an edge on **RTCK** before changing **TCK**. In a multiple device JTAG chain, the **RTCK** output from a component connects to the **TCK** input of the next device in the chain. The **RTCK** signal on the module connectors HDRB returns **TCK** to the JTAG equipment. If there are no synchronizing components in the scan chain then it is unnecessary to use the **RTCK** signal and it is connected to ground on the motherboard.
TCK	Test clock (from JTAG equipment)	**TCK** synchronizes all JTAG transactions. **TCK** connects to all JTAG components in the scan chain. Series termination resistors are used to reduce reflections and maintain good signal integrity. **TCK** flows down the stack of modules and connects to each JTAG component. However, if there is a device in the scan chain that synchronizes **TCK** to some other clock, then all down-stream devices are connected to the **RTCK** signal on that component (see **RTCK**).
TDI	Test data in (from JTAG equipment)	**TDI** goes down the stack of modules to the motherboard and then back up the stack, labelled **TDO**, connecting to each component in the scan chain.
TDO	Test data out (to JTAG equipment)	**TDO** is the return path of the data input signal **TDI**. The module connectors HDRB have two pins labelled **TDI** and **TDO**. **TDI** refers to data flowing down the stack and **TDO** to data flowing up the stack. The JTAG components are connected in the return path so that the length of track driven by the last component in the chain is kept as short as possible.
TMS	Test mode select (from JTAG equipment)	**TMS** controls transitions in the tap controller state machine. **TMS** connects to all JTAG components in the scan chain as the signal flows down the module stack.

3.6.5 Debug communications interrupts

The processor core incorporates EmbeddedICE logic that contains a communications channel used for passing information between the core and the JTAG equipment. The debug communications channel is implemented as coprocessor 14.

The ARM920T, ARM920T-ETM, and ARM940T processor cores incorporate EmbeddedICE logic and provides a Debug Communications Data Register that is used to pass data between the processor and JTAG equipment. The processor accesses this register as a normal 32-bit read/write register and the JTAG equipment reads and writes the register using a scan chain. For a description of the debug communications channel, see the *ARM920T Technical Reference Manual* or the *ARM940T Technical Reference Manual*.

You can use interrupts to signal when data has been written into one side of the register and is available for reading from the other side. These interrupts are supported by the interrupt controller within the core module FPGA and can be enabled and cleared by accessing the interrupt registers (see *Core module debug comms interrupt registers* on page 4-22).

3.7 Embedded trace support

The ARM920T-ETM processor incorporates an *ARM9 Embedded Trace Macrocell* (ETM9). This enables you to carry out real-time debugging by connecting external trace equipment to the core module. To trace program flow, the ETM broadcasts branch addresses, data accesses, and status information through the trace port. Later in the debug process, the complete instruction flow can be reconstructed by the ARM *Trace Debug Tools* (TDT).

Note

You cannot reconstruct self-modifying code.

3.7.1 About using trace

Figure 3-11 shows a trace debugging setup with a CM920T-ETM.

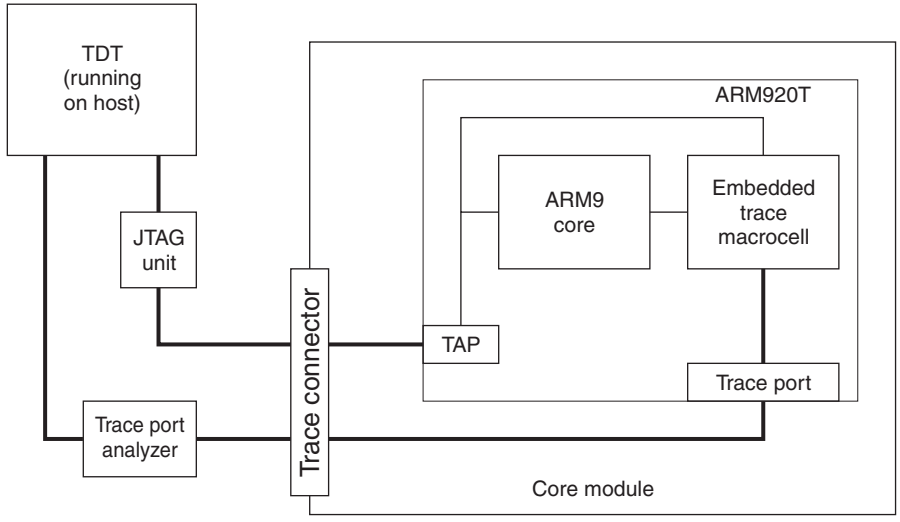


Figure 3-11 Trace connection

Note

The routing of the JTAG scan chain on the Integrator system is described in *Multi-ICE support* on page 3-16.

The components in the trace debug setup shown in Figure 3-11 on page 3-25 are as follows:

Embedded trace macrocell

The ETM monitors the ARM core buses and outputs compressed information through the trace port to a *Trace Port Analyzer* (TPA). The on-chip ETM contains trigger and filter logic to control what is traced.

Trace port analyzer

The TPA is an external device that stores information from the trace port.

JTAG unit

This is a protocol converter that converts debug commands from the debugger into JTAG messages for the ETM. The JTAG unit can be a separate device, such as Multi-ICE, or it might be incorporated within the TPA.

Trace debug tools

TDT is an optional component of the *ARM Developer Suite* (ADS) that runs on a host system. It is used to set up the filter logic, retrieve data from the analyzer, and reconstruct an historical view of processor activity. For further information, see the *ADS Trace Debug Tools User Guide*.

3.7.2 Core module trace configuration

The ETM configuration provided by the test chip is defined by its manufacturer. For example, different test chips might have different ETM sizes. See the release note for your test chip for details.

3.7.3 Trace interface description

The trace connector enables you to connect a TPA. This connector is a high-density AMP Mictor connector. The pinout for this connector is provided in *Trace connector pinout* on page A-13.

3.8 Stacking options

The core module provides two stacking options selected by the position of a surface mount link, LK11, that is set at manufacture.

Table 3-5 shows the link positions and the corresponding stacking option.

Table 3-5 Link positions

Position	Function
A:C	Normal operation. The core module to be used with a motherboard or standalone
B:C	Core Module at the bottom of the stack with no motherboard. At least one logic module must be placed on top of the stack. Refer to the appropriate logic module documentation for information about how to use it in this configuration.

Chapter 4
Programmer's Reference

This chapter describes the memory map and the status and control registers. It contains the following sections:

Note

This chapter describes the generic programming interface. For details of register usage that is dependent on the FPGA image, see Chapter 5 *Using Core Modules with an Integrator/AP* and Chapter 6 *Using Core Modules with an Integrator/CP*.

4.1 Memory organization

This section describes the memory map of the core module. For a standalone core module, the memory map is limited to local SSRAM, SDRAM, and core module registers. The memory map depends on whether the module is fitted to a motherboard and on the state of REMAP.

For the full memory map of an Integrator development system, which includes a motherboard, refer to the user guide for the motherboard.

4.1.1 Core module memory map

The core module has a fixed memory map that maintains compatibility with ARM Integrator motherboards and modules. Table 4-1 shows the memory map (x indicates that the signal can be HIGH or LOW without effect). The signal **nMBDET** is pulled LOW when the core module is fitted to an Integrator motherboard.

Table 4-1 Core module memory map

nMBDET	REMAP	Address range	Size	Description
0	0	0x00000000–0x000FFFFF	1MB	Boot ROM or flash on motherboard
0	1	0x00000000–0x000FFFFF	1MB	SSRAM
1	x	0x00000000–0x000FFFFF	1MB	SSRAM
x	x	0x00100000–0x0FFFFFFF	255MB	Local SDRAM
x	x	0x10000000–0x107FFFFF	8MB	Core module registers
x	x	0x10800000–0x10FFFFFF	8MB	SSRAM alias
0	x	0x11000000–0xFFFFFFFF	3824MB	System bus address space
1	x	0x11000000–0xFFFFFFFF	3824MB	Bus error response

4.1.2 Using REMAP

The SSRAM on the core module and the alias of the boot ROM or flash memory on an Integrator motherboard share the same locations within the Integrator memory map. Accesses these devices are controlled by the REMAP bit and the motherboard detect signal, **nMBDET**, that is permanently grounded by the motherboard. The effect on the memory map is shown in Figure 4-1.

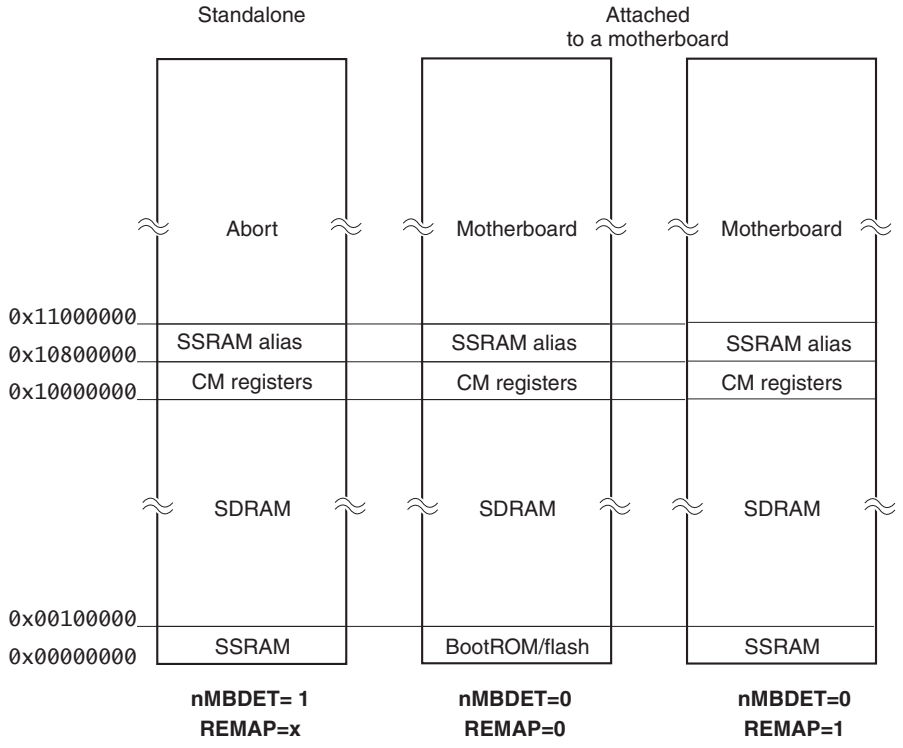


Figure 4-1 Effect of remap

The REMAP bit only has any effect if the core module is attached to a motherboard (**nMBDET**=0). It is controlled by bit 2 of the CM_CTRL Register at `0x1000000C` and functions as follows:

REMAP=0 Accesses to locations within the address range `0x00000000`–`0x000FFFFF` are to the boot ROM or flash on the motherboard.

If the core module is attached to a motherboard, REMAP is always 0 following a reset.

Note

You can set REMAP to 0 only if the core module is attached to a motherboard.

REMAP=1 Accesses to locations within the address range `0x00000000`–`0x000FFFFF` are to the SSRAM on the core module.

Note

Program execution normally starts at `0x00000000`. A switch on the motherboard determines whether the boot ROM or flash is aliased to this location. This enables you to boot from the boot ROM or from flash. Refer to the user guide for your motherboard for more information.

4.1.3 SDRAM mapping

The Integrator memory map provides two regions in which the SDRAM can be accessed:

Local region The local region can only be accessed by the local processor

Global region

If using an Integrator/AP, the global region can also be accessed by any master within the system.

Local SDRAM access

The local SDRAM access region enables access to the SDRAM only by the processor on the same core module at `0x00100000`–`0x0FFFFFFF`. You can fit an SDRAM of up to 256MB but you cannot access the lowest 1MB. This is because the lowest 1MB (`0x00000000`–`0x000FFFFF`) is hidden by the SSRAM (if REMAP=1), or by the boot ROM or flash (if REMAP=0). When a smaller sized SDRAM DIMM is fitted, it is mapped repeatedly to fill this space. Figure 4-2 shows an example of a 64MB DIMM mapped four times.

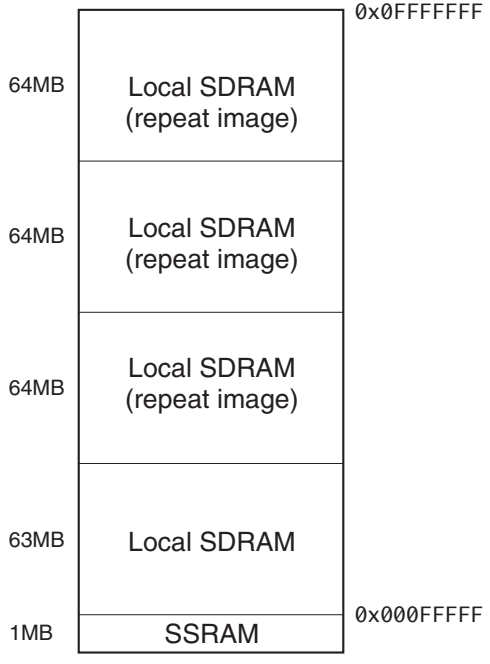


Figure 4-2 SDRAM repeat mapping for a 64MB DIMM

Global SDRAM access

The Integrator system memory map provides an alias memory region in which the SDRAM on core modules can be accessed. This region is only available if the core module is attached to a motherboard. Each core module is assigned to a 256MB space within this region and enables you to access the whole of a 256MB DIMM. The SDRAM can be accessed by all bus masters at its alias location.

The system bus address for a core module is automatically defined by its position in the stack (see *Core module ID* on page 2-8). Figure 4-3 shows the local and alias address of the SDRAM on four core modules.

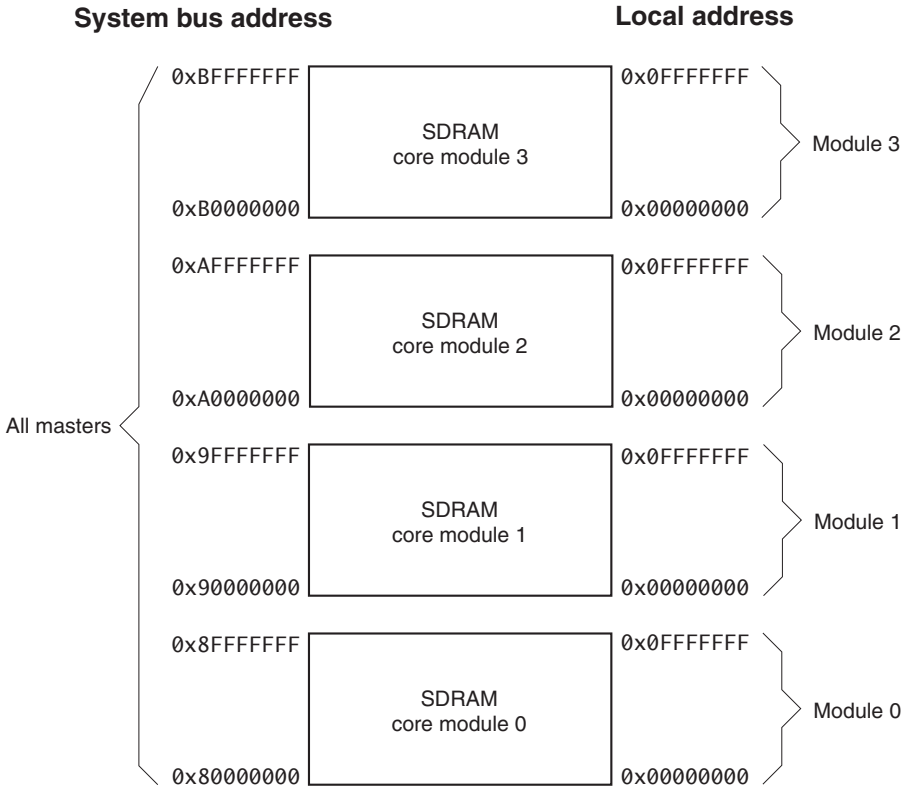


Figure 4-3 Core module local and alias addresses

Note

A processor can read from and write to its own SDRAM at the system bus address. However, these accesses are slower than local accesses because they are bridged to and from the system bus.

You can determine which core module a processor is on by reading the CM_STAT Register. Use the core module position to determine the alias location of the SDRAM on the same core module (see *Core Module Status Register* on page 4-14).

4.2 Exception vector mapping

The convention for ARM cores is to map the exception vectors to begin at address 0. However, the ARM920T and ARM940T cores enable the vectors to be moved to `0xFFFF0000` by setting the V bit in coprocessor 15 register 1. To maintain compatibility across all cores, the default reset value maps the vector to begin at address 0 (see the *ARM920T Technical Reference Manual* and *ARM940T Technical Reference Manual*).

You can use the CM_INIT Register to control the value of the V bit at reset (see *Core Module Initialization Register* on page 4-20).

To run applications with high vectors (at virtual address `0xFFFF0000`–`0xFFFF001C`), you must write the correct value to the coprocessor register. This virtual address must be mapped to a physical address that contains real memory, for example, core module SDRAM.

4.3 Core module control registers

The core module status and control registers enable the processor to determine its environment and to control some core module operations. The registers, listed in Table 4-2, are located at `0x10000000` and can only be accessed by the local processor.

Table 4-2 Core module status, control, and interrupt registers

Register name	Address	Access	Reset	Description
CM_ID	`0x10000000`	Read	Static	Core Module Identification Register
CM_PROC	`0x10000004`	Read	Static	Core Module Processor Register
CM_OSC	`0x10000008`	Read/write	POR	Core Module Oscillator Register
CM_CTRL	`0x1000000C`	Read/write	Reset	Core Module Control Register
CM_STAT	`0x10000010`	Read	Reset	Core Module Status Register
CM_LOCK	`0x10000014`	Read/write	Reset	Core Module Lock Register
CM_LMBUSCNT	`0x10000018`	Read	Reset	Core Module Local Memory Bus Cycle Counter (Integrator/AP only)
CM_AUXOSC	`0x1000001C`	Read/write	POR	Core Module Auxiliary Clock Oscillator Register
CM_SDRAM	`0x10000020`	Read/write	POR	Sdram Status And Control Register
CM_INIT	`0x10000024`	Read/write	POR	Core Module Initialization Register
CM_REFCNT	`0x10000028`	Read	Reset	Reference Clock Cycle Counter
CM_UNUSED1	`0x1000002C`	-	-	Reserved
CM_FLAGS	`0x10000030`	Read	Reset	Core Module Flag Register
CM_FLAGSET	`0x10000030`	Write	Reset	Core Module Flag Set Register
CM_FLAGSCLR	`0x10000034`	Write	Reset	Core Module Flag Clear Register
CM_NVFLAGS	`0x10000038`	Read	POR	Core Module Nonvolatile Flag Register
CM_NVFLAGSSET	`0x10000038`	Write	POR	Core Module Nonvolatile Flag Set Register
CM_NVFLAGSCLR	`0x1000003C`	Write	POR	Core Module Nonvolatile Flag Clear Register
CM_IRQ_STATUS	`0x10000040`	Read	Reset	Core Module Irq Status Register
CM_IRQ_RSTAT	`0x10000044`	Read	Reset	Core Module Irq Raw Status Register

Table 4-2 Core module status, control, and interrupt registers (continued)

Register name	Address	Access	Reset	Description
CM_IRQ_ENSET	0x10000048	Read	Reset	Core Module IRQ Enable Set Register
CM_IRQ_ENCLR	0x1000004C	Write	Reset	Core Module IRQ Enable Clear Register
CM_SOFT_INTSET	0x10000050	Read/write	Reset	Core Module Software Interrupt Set
CM_SOFT_INTCLR	0x10000054	Write	Reset	Core Module Software Interrupt Clear
CM_FIQ_STATUS	0x10000060	Read	Reset	Core Module FIQ Status Register
CM_FIQ_RSTAT	0x10000064	Read	Reset	Core Module FIQ Raw Status Register
CM_FIQ_ENSET	0x10000068	Read/write	Reset	Core Module FIQ Enable Set Register
CM_FIQ_ENCLR	0x1000006C	Write	Reset	Core Module FIR Enable Clear Register
CM_SPD	0x10000100–0x100001FC	Read	POR	SDRAM SPD memory

Note

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

4.3.1 Core Module ID Register

The Core Module ID Register, CM_ID, is a read-only register that identifies the board manufacturer, board type, and revision.

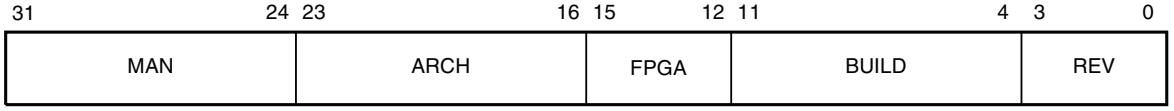


Table 4-3 describes the core module ID register bits.

Table 4-3 CM_ID Register bit descriptions

Bits	Name	Access	Function
[31:24]	MAN	Read	Manufacturer: 0x41 = ARM
[23:16]	ARCH	Read	Architecture: 0x00 = ASB system bus, ASB processor bus 0x03 = AHB-Lite system bus, bi-endian 0x08 = AHB system bus, ASB processor bus
[15:12]	FPGA	Read	FPGA type: 0x03 = XCV600 or XCV600E
[11:4]	BUILD	Read	Build value (ARM internal use)
[3:0]	REV	Read	Revision: 0x0 = Rev A 0x1 = Rev B 0x3 = Rev D (for CP image)

4.3.2 Core Module Processor Identification Register

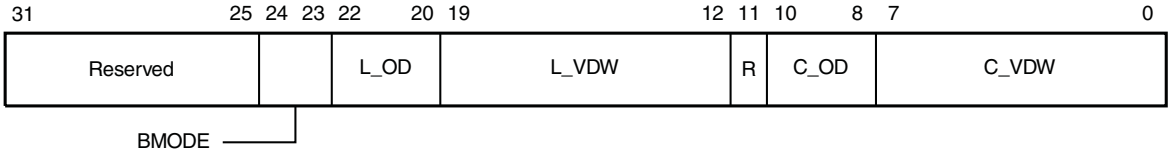
The Core Module Processor Identification Register, CM_PROC, is a read-only register that contains the value 0x00000000. This is provided for compatibility with processors that do not have a system control coprocessor (CP15). For the ARM920T and ARM940T, you can obtain information about the processor by reading coprocessor 15 register 0.

4.3.3 Core Module Control Register

The Core Module Control Register, CM_CTRL, is a read/write register that provides control of several user-configurable features of the core module. The function of CM_CTRL in the FPGA image is different for the Integrator/AP and Integrator/CP, see *Core Module Control Register for Integrator/AP* on page 5-8 and *Core Module Control Register, CP* on page 6-14.

4.3.4 Core Module Oscillator Register

The Core Module Oscillator Register, CM_OSC, is a read/write register that controls the frequency of the clocks generated by the two clock generators (see *Clock generators* on page 3-10). In addition, it provides information about processor bus mode setting.



Note

Before writing to CM_OSC, you must unlock it by writing the value `0x0000A05F` to CM_LOCK. After writing CM_OSC, relock it by writing any value other than `0x0000A05F` to CM_LOCK.

Table 4-4 describes the Core Module Oscillator Register bits.

Table 4-4 CM_OSC Register bit descriptions

Bits	Name	Access	Function
[31:25]	Reserved	Use read-modify-write to preserve value.	
[24:23]	BMODE	Read	This field contains 01 which indicates that the processor bus mode is selected by writing to coprocessor 15 register 1.
[22:20]	L_OD	Read/write	Memory clock output divider: 000 = divide by 10 001 = divide by 2 (default) 010 = divide by 8 011 = divide by 4 100 = divide by 5 101 = divide by 7 110 = divide by 9 111 = divide by 6.
[19:12]	L_VDW	Read/write	Processor bus clock VCO divider word. Defines the binary value of the V[7:0] pins of the clock generator (V[8] is tied LOW). `b00100000` = 20MHz (default with OD = 2).
[11]	Reserved	Use read-modify-write to preserve value.	
[10:8]	COREOD	Read/write	Core clock output divider: 000 = divide by 10 001 = divide by 2 (default) 010 = divide by 8 011 = divide by 4 100 = divide by 5 101 = divide by 7 110 = divide by 9 111 = divide by 6.
[7:0]	COREVCO	Read/write	Core clock VCO divider word. Defines the binary value of the V[7:0] pins of the clock generator (V[8] is tied LOW). `b00101010` = 50MHz (default with OD = 2).

4.3.5 Core Module Status Register

The Core Module Status Register, CM_STAT, is a read-only register that can be read to determine the SSRAM size, core type, and where in a multi-core module stack this core module is positioned.

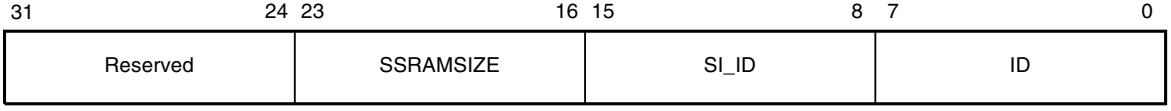


Table 4-5 describes the Core Module Status Register bits.

Table 4-5 CM_STAT Register bit descriptions

Bits	Name	Access	Function
[31:24]	Reserved	Use read-modify-write to preserve value.	
[23:16]	SSRAMSIZE	Read	SSRAM size. This contains `0x10` to indicate that 1MB is fitted.
[15:8]	SI_ID	Read	Silicon manufacturer identification. Identifies the manufacturer and type of core fitted to the module: `0x00` = unknown or socket fitted `0x01` = Cirrus ARM920T(0.25µ, 2V5 core and pads) `0x02` = Panasonic ARM920T (0.18µ, 1V8 core and 3V3 pads) `0x03` = Samsung ARM920T(0.25µ, 2V5 core and 3V3 pads) `0x04` - `0xFF` = reserved.
[7:0]	ID	Read	Card number in stack: `0x00` = core module 0 `0x01` = core module 1 `0x02` = core module 2 `0x03` = core module 3 `0xFF` = invalid or no motherboard attached.

4.3.6 Core Module Lock Register

The Core Module Lock Register, CM_LOCK, is a read/write register that is used to control access to the CM_OSC Register and enable it to be locked and unlocked. This mechanism prevents the CM_OSC Register from being overwritten accidently.

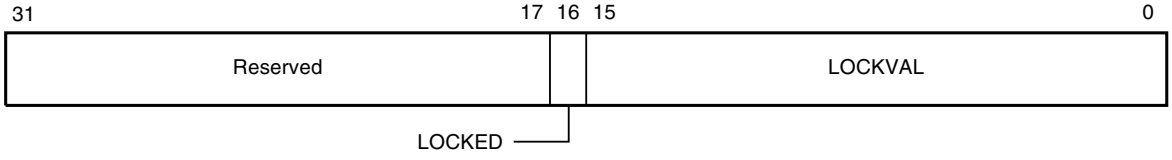


Table 4-6 describes the Core Module Lock Register bits.

Table 4-6 CM_LOCK Register bit descriptions

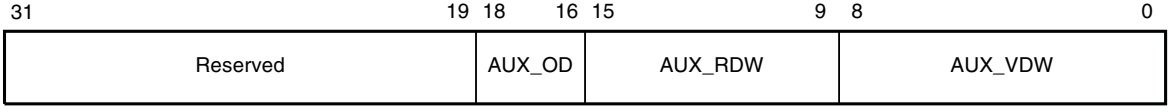
Bits	Name	Access	Function
[16]	LOCKED	Read	This bit indicates if the CM_OSC Register is locked or unlocked: 0 = unlocked 1 = locked.
[15:0]	LOCKVAL	Read/write	Write the value `0x0000A05F` to this register to enable write accesses to the CM_OSC Register. Write any other value to this register to lock the CM_OSC Register.

4.3.7 Core Module Local Memory Bus Cycle Counter Register

The Core Module Local Memory Bus Cycle Counter Register, CM_LMBUSCNT, provides a 16-bit count value. The count increments at the memory bus frequency and can be used as a cycle counter for performance measurement. The register is reset to zero by a reset. (This register is not present in the FPGA image for the Integrator/CP.)

4.3.8 Core Module Auxiliary Oscillator Register

The Core Module Auxiliary Oscillator Register, CM_AUXOSC, is a read/write register that controls the frequency of the clock generated by the clock generator for **AUXCLK** (see *Programming the auxiliary clock, AUXCLK* on page 3-14). This register enables you to set all three of the control inputs to the clock generator. The default setting of this register gives a 32.369MHz output.



Note

Before writing to the CM_AUXOSC Register, unlock it by writing the value `0x0000A05F` to the CM_LOCK Register. After writing the CM_AUXOSC Register, relock it by writing any value other than `0x0000A05F` to the CM_LOCK Register.

Table 4-7 describes the Core Module Auxiliary Oscillator Register bits.

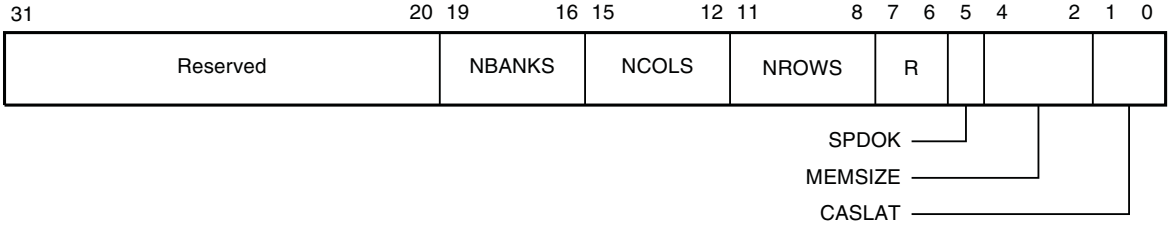
Table 4-7 CM_AUXOSC Register bit descriptions

Bits	Name	Access	Function
[31:19]	Reserved	Use read-modify-write to preserve value.	
[18:16]	AUX_OD	Read/write	Auxiliary output divider. Sets the binary code on the S[2:0] pins of the clock generator. The divider is encoded as follows: 000 = divide by 0 001 = divide by 2 010 = divide by 8 011 = divide by 4 100 = divide by 5 101 = divide by 7 110 = divide by 9 111 = divide by 6 (default).
[15:9]	AUX_RDW	Read/write	Auxiliary reference divider word. Defines the binary value on the R[6:0] pins of the clock generator. `b0111111` = 63 (default).
[8:0]	AUX_VDW	Read/write	Auxiliary clock VCO divider word. Defines the binary value on the V[8:0] pins of the clock generator. `b011111111` = 255 (default).

4.3.9 Core Module SDRAM Status and Control Register

The SDRAM Status and Control Register, CM_SDRAM, is a read/write register used to set the configuration parameters for the SDRAM DIMM. This control is necessary because of the variety of module sizes and types available.

Writing a value to this register automatically updates the mode register on the SDRAM DIMM.



Note

Before the SDRAM is used it is necessary to read the SPD memory and program the CM_SDRAM Register with the parameters indicated in Table 4-8. If these values are not correctly set then SDRAM accesses might be slow or unreliable. See *SDRAM SPD memory* on page 4-26.

Table 4-8 describes the SDRAM Status and Control Register bits.

Table 4-8 CM_SDRAM Register bit descriptions

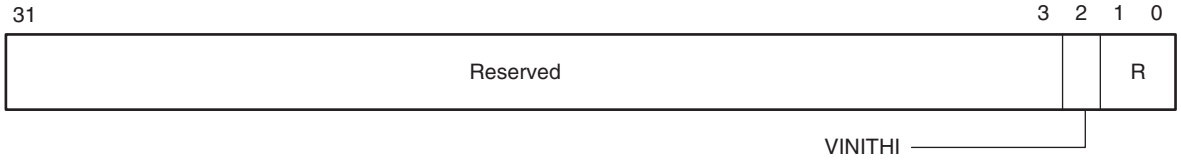
Bits	Name	Access	Function
[31:20]	Reserved	Use read-modify-write to preserve value.	
[19:16]	NBANKS	Read/write	Number of SDRAM banks. Must be set to the same value as byte 5 of SPD EEPROM.
[15:12]	NCOLS	Read/write	Number of SDRAM columns. Must be set to the same value as byte 4 of SPD EEPROM.
[11:8]	NROWS	Read/write	Number of SDRAM rows. Must be set to the same value as byte 3 of SPD EEPROM.
[7:6]	Reserved	Use read-modify-write to preserve value.	

Table 4-8 CM_SDRAM Register bit descriptions (continued)

Bits	Name	Access	Function
[5]	SPDOK	Read	This bit indicates that the automatic copying of the SPD data from the SDRAM module into CM_SPDMEM is complete: 1 = SPD data ready 0 = SPD data not available.
[4:2]	MEMSIZE	Read/write	These bits specify the size of the SDRAM module fitted to the core module. The bits are encoded as follows: 000 = 16MB 001 = 32MB 010 = 64MB (default) 011 = 128MB 100 = 256MB 101 = Reserved 110 = Reserved 111 = Reserved.
[1:0]	CASLAT	Read/write	These bits specify the CAS latency for the core module SDRAM. The bits are encoded as follows: 00 = Reserved 01 = Reserved 10 = 2 cycles (default) 11 = 3 cycles.

4.3.10 Core Module Initialization Register

The Core Module Initialization Register, CM_INIT, is used to control the mapping of the exception vector table for the test chip.



Note

Before writing to the CM_INIT Register, you must unlock it by writing the value `0x0000A05F` to the CM_LOCK Register. After writing the CM_INIT Register, relock it by writing any value other than `0x0000A05F` to the CM_LOCK Register.

Table 4-9 describes the Core Module Initialization Register bits.

Table 4-9 CM_INIT Register bit descriptions

Bits	Name	Access	Function
[31:3]	Reserved	Use read-modify-write to preserve value.	
[2]	VINITHI	Read/write	This bit controls the state of the VINTHI pin of the ARM920T and ARM940T cores. This value determines the initial value of the V bit in coprocessor 15 register 1 and, therefore, the location of the exception vector table. To use high vector mapping, set this bit to 1 and then reset the core module: 0 = vectors at 0 (default after POR) 1 = vectors at `0xFFFF0000`.
[1:0]	Reserved	Use read-modify-write to preserve value.	

4.3.11 Core Module Reference Clock Cycle Counter Register

The Core Module Reference Clock Cycle Counter Register, CM_REFCNT, provides a 32-bit count value. The count increments at the fixed reference clock frequency of 24MHz and can be used as a real-time counter. The register is reset to zero by a reset.

4.4 Core module flag registers

The core module flag registers provide you with two 32-bit register locations containing general purpose flags. You can assign any meaning to the flags. The flag registers are shown in Table 4-10.

Table 4-10 Core module flag registers

Register name	Address	Access	Reset by	Description
CM_FLAGS	0x10000030	Read	Reset	Core Module Flag Register
CM_FLAGSSET	0x10000030	Write	Reset	Core Module Flag Set Register
CM_FLAGSCLR	0x10000034	Write	Reset	Core Module Flag Clear Register
CM_NVFLAGS	0x10000038	Read	POR	Core Module Nonvolatile Flag Register
CM_NVFLAGSSET	0x10000038	Write	POR	Core Module Nonvolatile Flag Set Register
CM_NVFLAGSCLR	0x1000003C	Write	POR	Core Module Nonvolatile Flag Clear Register

The core module provides two distinct types of flag registers:

- FLAGS registers are cleared by a normal reset, such as a reset caused by pressing the reset button
- NVFLAGS registers retain their contents after a normal reset and are only cleared by a *Power-On Reset* (POR).

4.4.1 Core Module Flag and Core Module Nonvolatile Flag Registers

The status register contains the current state of the flags.

4.4.2 Core Module Flag and Core Module Nonvolatile Flag Set Registers

The flag set locations are used to set bits in the flag registers:

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

4.4.3 Core Module Flag and Core Module Nonvolatile Flag Clear Registers

The clear locations are used to clear bits in the flag registers:

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

4.5 Core module debug comms interrupt registers

The core module provides a 3-bit IRQ controller and 3-bit FIQ controller to support the debug communications channel used for passing information between applications software and the debugger. The interrupt control registers are shown in Table 4-11.

Table 4-11 Debug comms interrupt controller registers

Register name	Address	Access	Description
CM_IRQ_STAT	0x10000040	Read	Core Module IRQ Status Register
CM_IRQ_RSTAT	0x10000044	Read	Core Module IRQ Raw Status Register
CM_IRQ_ENSET	0x10000048	Read/write	Core Module IRQ Enable Set Register
CM_IRQ_ENCLR	0x1000004C	Write	Core Module IRQ Enable Clear Register
CM_SOFT_INTSET	0x10000050	Read/write	Core Module Software Interrupt Set
CM_SOFT_INTCLR	0x10000054	Write	Core Module Software Interrupt Clear
CM_FIQ_STAT	0x10000060	Read	Core Module FIQ Status Register
CM_FIQ_RSTAT	0x10000064	Read	Core Module FIQ Raw Status Register
CM_FIQ_ENSET	0x10000068	Read/write	Core Module FIQ Enable Set Register
CM_FIQ_ENCLR	0x1000006C	Write	Core Module FIR Enable Clear Register

Note

All registers are 32 bits wide and do not support byte writes. Write operations must be word-wide. The values of bits marked as *reserved* in the following sections must be written as 0s.

The Integrator/AP and Integrator/CP have additional interrupt controllers. See *Interrupt control for AP* on page 5-21 and *Interrupt control for CP* on page 6-23.

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 4-4 and described in the following subsections. The illustration shows the control for one IRQ bit. The remaining IRQ bits and FIQ bits are controlled in a similar way.

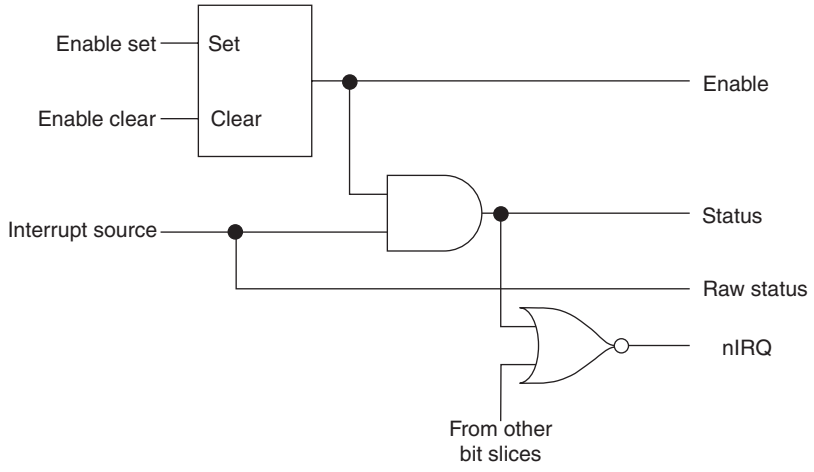


Figure 4-4 Interrupt control

4.5.1 Core Module IRQ/FIQ Status Registers

The Core Module Status Register contains the logical AND of the bits in the Core Module Raw Status Register and the Core Module Enable Register.

4.5.2 Core Module IRQ/FIQ Raw Status Registers

The Core Module Raw Status Register indicates the signal levels on the interrupt request inputs. A bit set to 1 indicates that the corresponding interrupt request is active.

4.5.3 Core Module IRQ/FIQ Enable Set Registers

The enable set locations are used to set bits in the Core Module Enable Registers:

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

Read the current state of the enable bits from the ENSET location.

4.5.4 Core Module IRQ/FIQ Enable Clear Registers

The clear set locations are used to set bits in the Core Module Enable Registers:

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

4.5.5 Core Module Debug Comms Interrupt Register bit assignment

The bit assignments for the IRQ and FIQ Status, Raw Status, and Enable Registers are shown in Table 4-12.

Table 4-12 IRQ and FIQ Debug Comms Interrupt Register bit assignment

Bits	Name	Function
[31:3]	Reserved	Write as 0. Reads undefined.
[2]	COMMTx	Debug communications transmit interrupt. This interrupt indicates that the communications channel is available for the processor to pass messages to the debugger.
[1]	COMMRx	Debug communications receive interrupt. This interrupt indicates to the processor that messages are available for the processor to read.
[0]	SOFT	Software interrupt.

4.5.6 Core Module Soft Interrupt Set and Core Module Soft Interrupt Clear Registers

The core module debug comms interrupt controller provides a register for controlling and clearing software interrupts. This register is accessed using the Core Module Software Interrupt Set Register and Core Module Software Interrupt Clear Register. The set and clear locations are used as follows:

- Set the software interrupt by writing to the CM_SOFT_INTSET location:
  - write a 1 to SET the software interrupt
  - write a 0 to leave the software interrupt unchanged.
- Read the current state of the of the Software Interrupt Register from the CM_SOFT_INTSET location. A bit set to 1 indicates that the corresponding interrupt request is active.
- Clear the software interrupt by writing to the CM_SOFT_INTCLR location:
  - write a 1 to CLEAR the software interrupt
  - write a 0 to leave the software interrupt unchanged.

The bit assignment for the Software Interrupt Register is shown in Table 4-13.

Table 4-13 IRQ Register bit assignment

Bits	Name	Function
[31:1]	Reserved	Write as 0. Reads undefined.
[0]	SOFT	Software interrupt.

Note

The *software interrupt* described in this section is used by software to generate IRQs or FIQs. It must not be confused with the ARM SWI software interrupt instruction. See the *ARM Architecture Reference Manual*.

4.6 SDRAM SPD memory

This area of memory contains a copy of the SPD data from the SPD EEPROM on the DIMM. Because accesses to the EEPROM are very slow, the data is copied to this memory during board initialization to enable faster random access to the SPD data (see *Serial presence detect* on page 3-9). The SPD memory contains 256 bytes of data. The most important of these are as shown in Table 4-14.

Table 4-14 SPD memory contents

Byte	Contents
2	Memory type
3	Number of row address lines
4	Number of column address lines
5	Number of chip-select banks
31	Module bank density (MB divided by 4)
18	CAS latencies supported
63	Checksum
64:71	Manufacturer
73:90	Module part number

Check for valid SPD data as follows:

1. Add together all bytes 0–62.
2. Logically AND the result with `0xFF`.
3. Compare the result with byte 63.

If the two values match, then the SPD data is valid.

Note

Several SDRAM DIMMs do not comply with the JEDEC standard and do not implement the checksum byte. The Integrator is not guaranteed to operate with non-compliant DIMMs.

The code segment shown in Example 4-1 on page 4-27 can be used to correctly setup and remap the SDRAM.

Example 4-1 SDRAM setup and remap

```
CM_BASE     EQU    0x10000000     ; base address of Core Module registers
SPD_BASE    EQU    0x10000100     ; base address of SPD information

lightled
           ; turn on header LED and remap memory
           LDR    r0, =CM_BASE    ; load register base address
           MOV    r1,#5           ; set remap and led bits
           STR    r1,[r0,#0xc]    ; write the register
           ; setup SDRAM

readspdbit
           ; check SPD bit is set
           LDR    r1,[r0,#0x20]   ; read the status register
           AND    r1,r1,#0x20     ; mask SPD bit (5)
           CMP    r1,#0x20        ; test if set
           BNE    readspdbit      ; branch until the SPD memory has been read

setupsdram
           ; work out the SDRAM size
           LDR    r0, =SPD_BASE  ; point at SPD memory
           LDRB   r1,[r0,#3]     ; number of row address lines
           LDRB   r2,[r0,#4]     ; number of column address lines
           LDRB   r3,[r0,#5]     ; number of banks
           LDRB   r4,[r0,#31]    ; module bank density
           MUL    r5,r4,r3       ; calculate size of SDRAM (MB divided by 4)
           MOV    r5,r5,ASL#2    ; size in MB
           CMP    r5,#0x10       ; is it 16MB?
           BNE    not16          ; if no, move on
           MOV    r6,#0x2        ; store size and CAS latency of 2
           B      writesize

not16
           CMP    r5,#0x20       ; is it 32MB?
           BNE    not32          ; if no, move on
           MOV    r6,#0x6        ; store size and CAS latency of 2
           B      writesize

not32
           CMP    r5,#0x40       ; is it 64MB?
           BNE    not64          ; if no, move on
           MOV    r6,#0xa        ; store size and CAS latency of 2
           B      writesize
```

```
not64
            CMP     r5,#0x80     ; is it 128MB?
            BNE     not128       ; if no, move on
            MOV     r6,#0xe      ; store size and CAS latency of 2
            B       writesize

not128
            ; if it is none of these sizes then it is either 256MB, or
            ; there is no SDRAM fitted so default to 256MB.
            MOV     r6,#0x12     ; store size and CAS latency of 2

writesize
            MOV      r1,r1,ASL#8        ; get row address lines for SDRAM register
            ORR      r2,r1,r2,ASL#12    ; OR in column address lines
            ORR      r3,r2,r3,ASL#16    ; OR in number of banks
            ORR      r6,r6,r3           ; OR in size and CAS latency
            LDR      r0, =CM_BASE       ; point at module registers
            STR      r6,[r0,#0x20]      ; store SDRAM parameters
```


Chapter 5
Using Core Modules with an Integrator/AP

This chapter contains the instructions for using an ARM Integrator core module with an Integrator/AP baseboard. It contains the following sections:

5.1 About the AP system architecture

Figure 5-1 shows the function of the core module FPGA and shows how it connects to the other devices in the system.

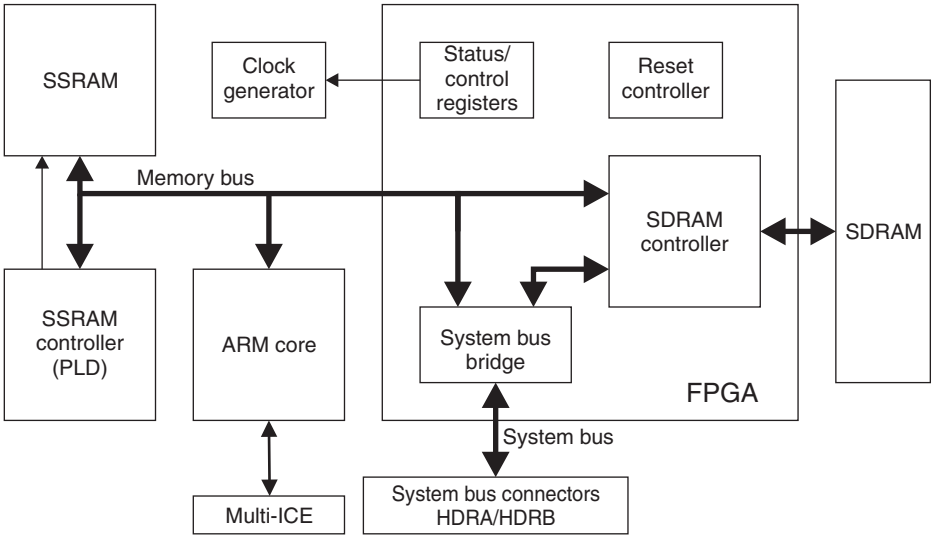


Figure 5-1 FPGA functional diagram

5.1.1 Configuring little or big-endian operation

The core module can be configured to operate as a big-endian or 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 must only be done at the start of a debugging session. The change must have taken effect before you can perform any subword accesses.

Caution

The FPGA image provided for the Integrator/AP is for little-endian operation. If you wish to use the Integrator/AP system in big-endian mode, you must provide your own FPGA images. Some of the peripherals on the Integrator/AP motherboard might not operate correctly in big-endian mode.

5.2 Module ID selection for AP

Several 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 enables 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 position of a core module in the HDRA/HDRB stack is used to determine its ID, and from this its address in the alias memory region (see the user guide for your motherboard) and the interrupts that it responds to.

Note

The core module cannot be damaged by connecting it onto the EXPA/EXPB position on the Integrator/AP motherboard, but fitting it in this position prevents reliable operation.

5.2.1 Module address decoding

The Integrator system implements a distributed address decoding system. This means that each core or logic module must decode its own area of the memory map. The central decoder in the system controller FPGA (on the motherboard) responds with an error code for all areas of the address space that are not occupied by a module. This default response is disabled for a memory region occupied by a module that is fitted.

The signals **nPPRES[3:0]** (core module present) and **nEPRES[3:0]** (logic module present) are used to signal the presence of modules to the central decoder. Only one signal in each group is pulled LOW for each module. The signals **ID[3:0]** indicate to the module its position in the stack and the address range that its own decoder must respond to. These signals rotate as they pass up the stack, as described in *System bus signal routing* on page 5-15.

The alias SDRAM address of a core module is determined in hardware, although a module can determine its own position by reading a decoded version of **ID[3:0]** from the CM_STAT Register (see *Core module control registers* on page 4-9 and *Core Module Control Register for Integrator/AP* on page 5-8). Table 5-1 shows alias addresses for a core module fitted to a motherboard HDRA/HDRB stack.

Table 5-1 Core module address decode

ID[3:0]	Module ID	Address range	Size
1101	3 (top)	0xB0000000–0xBFFFFFFF	256MB
1011	2	0xA0000000–0xAFFFFFFF	256MB
0111	1	0x90000000–0x9FFFFFFF	256MB
1110	0 (bottom)	0x80000000–0x8FFFFFFF	256MB

5.3 Top-level AP memory map

Figure 5-2 shows the top-level memory map of an Integrator/AP system. The memory map is largely compatible with other platforms in the Integrator product family. This simplifies porting code and enables you to expand the system with additional Integrator logic modules.

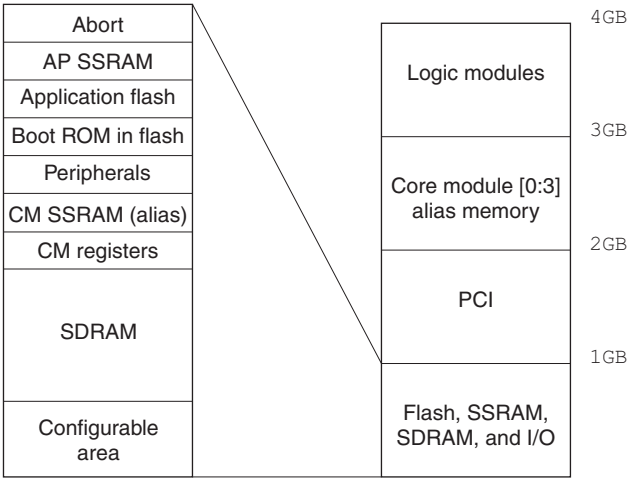


Figure 5-2 Top-level memory map

See the *Integrator/AP User Guide* and Chapter 4 *Programmer's Reference* for more details on memory mapping and register function. The FPGA image for the Integrator/AP provides different register function for the CM Control Register than the function provided by the FPGA image for the Integrator/CP. See *Core Module Control Register for Integrator/AP* on page 5-8 for details.

Note

The memory map below `0x10000000` depends on the setting for REMAP.

5.3.1 Global SDRAM access

If the core module is mounted on an Integrator/AP motherboard, the SDRAM appears within a 256MB space in the *alias* memory region of the overall Integrator system memory map (see the user guide for your motherboard). The SDRAM can be accessed by all bus masters within this region.

The alias address for a core module SDRAM is automatically controlled by its position in the stack (see *Module ID selection for AP* on page 5-3). Figure 5-3 shows the alias address of the SDRAM on four core modules.

A processor can determine which core module it is on and, therefore, the alias location of its own SDRAM by reading the CM_STAT Register, (see *Core Module Status Register* on page 4-14).

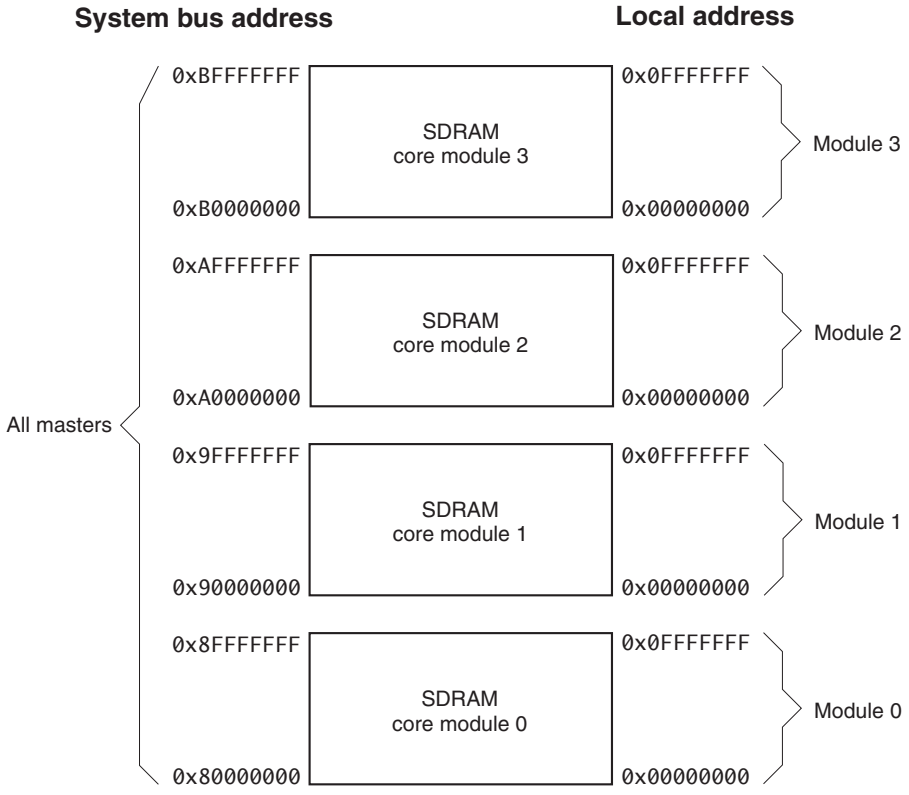


Figure 5-3 Core module local and alias addresses

5.3.2 Access arbitration

The SDRAM controller provides two ports to support reads and writes by the local processor core and by masters on the motherboard. The SDRAM controller uses an alternating priority scheme to ensure that the processor core and motherboard have equal access (see *System bus bridge for AP* on page 5-10).

5.4 Register and memory overview for AP

Table 5-2 shows a map for a typical Integrator/AP system. See the *Integrator/AP User Guide* for more details on register function.

Note

The memory map below `0x10000000` depends on the settings for REMAP.

Table 5-2 System control register map

Peripheral or device	Address range	Size
Boot flash. Mapped at this address only at power ON, and then can be disabled under software control to allow access to SSRAM. **Note** The memory map below `0x10000000` depends on the setting for REMAP.	`0x00000000–0x000FFFFF`	1MB
SDRAM.	`0x00010000–0x0FFFFFFF`	255MB
Core Module control registers.	`0x10000000-0x1000003F`	64 bytes
Core Module interrupt controller.	`0x10000040–0x1000007F`	64 bytes
Reserved	`0x10000080–0x100000FF`	128 bytes
Serial Presence Detect memory.	`0x10000100–0x100001FF`	256bytes
AP System Controller Registers.	`0x11000000–0x11FFFFFF`	16MB
AP EBI Configuration Registers.	`0x12000000–0x12FFFFFF`	16MB
AP Counter/timer Registers.	`0x13000000–0x13FFFFFF`	16MB
AP Interrupt Controller Registers.	`0x14000000–0x14FFFFFF`	16MB
AP Real time clock.	`0x15000000–0x15FFFFFF`	16MB
AP UART0.	`0x16000000–0x16FFFFFF`	16MB
AP UART1.	`0x17000000–0x17FFFFFF`	16MB
AP Keyboard.	`0x18000000–0x18FFFFFF`	16MB
AP Mouse.	`0x19000000–0x19FFFFFF`	16MB

Table 5-2 System control register map (continued)

Peripheral or device	Address range	Size
AP Debug LEDs and Boot switch.	0x1A000000–0x1AFFFFFF	16MB
AP GPIO.	0x1B000000–0x1BFFFFFF	16MB
Spare.	0x1C000000–0x1FFFFFFF	64MB
AP External bus interface (Boot ROM, application flash, and AP SSRAM)	0x20000000–0x2FFFFFFF	256MB
Reserved	0x30000000–0x3FFFFFFF	256MB
AP PCI memory space	0x40000000–0x7FFFFFFF	1GB
Core module 0 alias memory.	0x80000000–0x8FFFFFFF	256MB
Core module 1 alias memory.	0x90000000–0x9FFFFFFF	256MB
Core module 2 alias memory.	0xE0000000–0xEFFFFFFF	256MB
Core module 3 alias memory.	0xF0000000–0xFFFFFFFF	256MB
Logic module 0 memory.	0xC0000000–0xCFFFFFFF	256MB
Logic module 1 memory.	0xD0000000–0xDFFFFFFF	256MB
Logic module 2 memory.	0xE0000000–0xEFFFFFFF	256MB
Logic module 3 memory.	0xF0000000–0xFFFFFFFF	256MB

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.

5.4.1 Core Module Control Register for Integrator/AP

The Core Module Control Register, CM_CTRL, at 0x1000000C is a read/write register that provides control of several user-configurable features of the core module. The functionality in this section is for the FPGA image for use with a standalone core module or with the core module connected to an Integrator/AP motherboard. See the *Integrator/AP User Guide* and Chapter 4 *Programmer's Reference* for more details on memory mapping and register function.

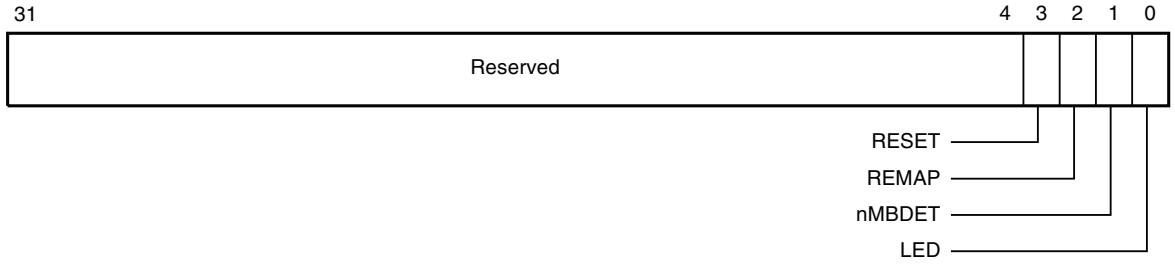


Figure 5-4 Core Module Control Register

Table 5-3 and Figure 5-4 describe the Core Module Control Register bits

Table 5-3 CM_CTRL Register bit descriptions

Bits	Name	Access	Function
[31:4]	Reserved	Use read-modify-write to preserve value.	
[3]	RESET	Write	This is used to reset the core module, the motherboard on which it is mounted, and any modules in a stack. A reset is triggered by writing a 1. Reading this bit always returns a 0 and enables you to use read-modify-write operations without masking the RESET bit.
[2]	REMAP	Read/write	This only has affect when the core module is mounted on a motherboard. The function of remap is described in *Using REMAP* on page 4-3.
[1]	nMBDET	Read	This bit indicates whether or not the core module is mounted on a motherboard: 0 = mounted on motherboard 1 = standalone.
[0]	LED	Read/write	This bit controls the green MISC LED on the core module: 0 = LED OFF 1 = LED ON.

5.5 System bus bridge for AP

The system bus bridge provides an asynchronous bus interface between the local memory bus and system bus connecting the motherboard and other modules. This section describes the following details of the system bus bridge:

- *System bus bridge FIFOs*
- *Processor accesses to the system bus* on page 5-11
- *Motherboard accesses to SDRAM* on page 5-12
- *Multiprocessor support* on page 5-14
- *System bus signal routing* on page 5-15
- *Bus operating modes* on page 5-17.

5.5.1 System bus bridge FIFOs

Inter-module accesses are supported by two 16 x 74-bit FIFOs. Each of the 16 entries in the FIFOs contains:

- 32-bit data used for write transfers
- 32-bit address used for reads and writes
- 10-bit transaction control used for reads and writes.

5.5.2 Processor accesses to the system bus

The first FIFO supports read and write accesses by the local processor to the system bus. The bus extends accesses onto the motherboard and other modules.

Processor writes

The data routing for processor writes to the system bus is illustrated in Figure 5-5.

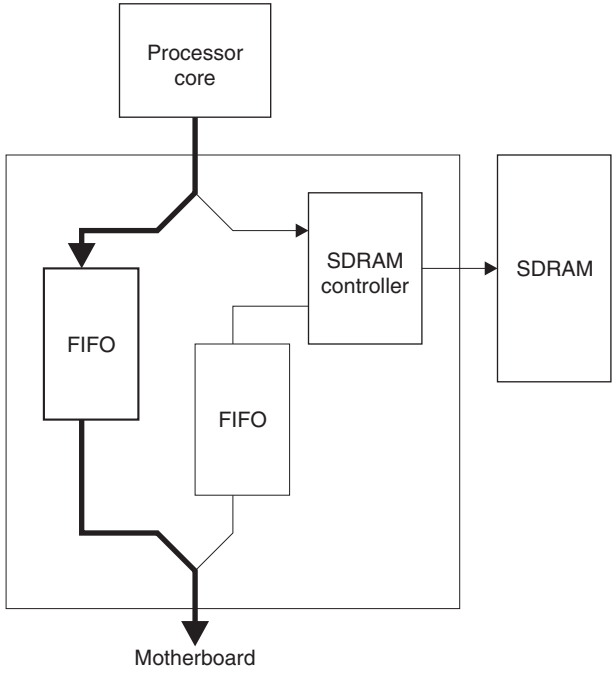


Figure 5-5 Processor writes to the system bus

Write transactions from the processor to the system bus normally complete on the local memory bus in a single cycle. The data, address, and control information associated with the transfer are posted into FIFO, and the transfer on the system bus occurs some time later when that bus is available. This means that system bus error responses to write transfers are not reported back to the processor as Data Aborts. If the FIFO is full, the processor receives a wait response until space becomes available.

Processor reads

The data routing for processor reads from the system bus is illustrated in Figure 5-6 on page 5-12.

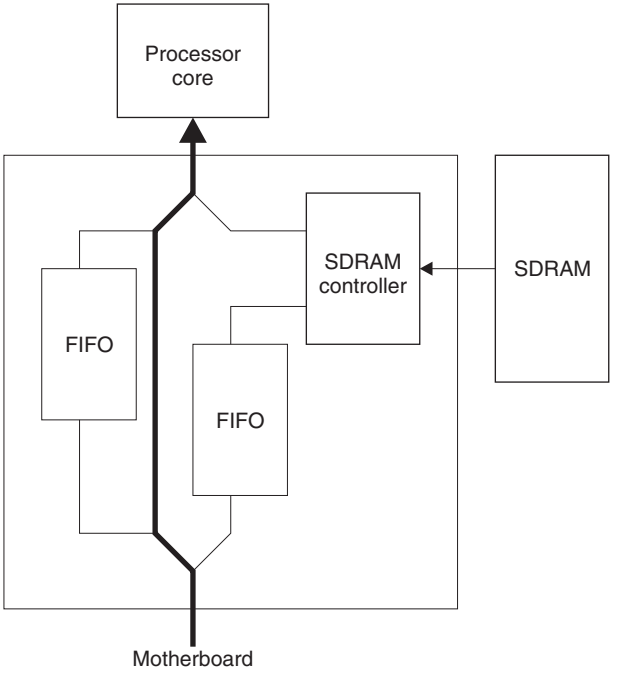


Figure 5-6 Processor reads from the system bus

For reads from the system bus, the address and control information also pass through the FIFO. The returned data from the system bus bypasses the FIFO.

The order of processor transactions is preserved on the system bus. Any previously posted writes are drained from the FIFO (that is, completed on the system bus) before the read transfer is performed. The processor receives a wait response until the read transfer has completed on the system bus, when it receives the data and any associated bus error response from the system bus. For information about SDRAM addresses, see *SDRAM mapping* on page 4-5.

You can use a system bus read to drain the FIFO of writes that might affect a subsequent operation. For example, to ensure that a write by an IRQ or FIQ handler clears an interrupt source and prevents the same interrupt being taken again by the core, the handler follows a write to the interrupt source with a read from the system bus.

5.5.3 Motherboard accesses to SDRAM

The second FIFO supports read and write accesses by system bus masters on the motherboard and other core modules to the local core module memory.

This section gives an overview of SDRAM access (see also *SDRAM mapping* on page 4-5.)

System bus writes

The data routing for system bus writes to SDRAM is illustrated in Figure 5-7.

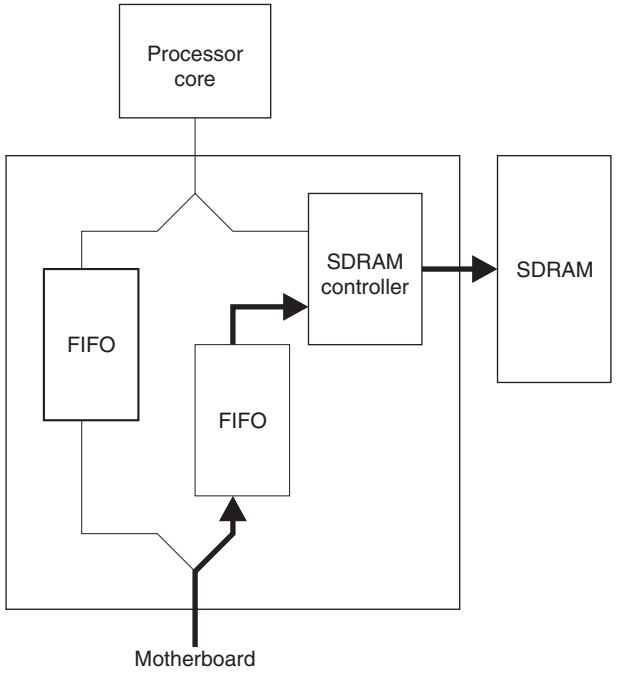


Figure 5-7 System bus writes to SDRAM

Write transactions from the system bus to the SDRAM normally complete in a single cycle on the system bus. The data, address, and control information associated with the transfer are posted into the FIFO, and the transfer into the SDRAM completes when the SDRAM is available. If the FIFO is full, then the system bus master receives a retract response indicating that the arbiter may grant the bus to another master and that this transaction must be retried later.

System bus reads

The data routing for system bus reads from SDRAM is illustrated in Figure 5-8.

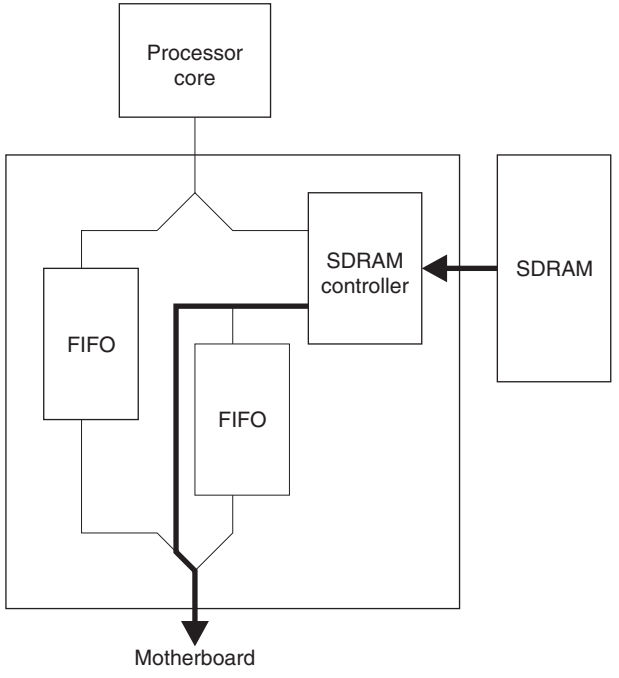


Figure 5-8 System bus reads from SDRAM

For system bus reads, the address and control information also pass through the FIFO, but the returned data from the SDRAM bypasses the FIFO.

The order of transactions on the system bus and the memory bus is preserved. Any previously posted write transactions are drained from the FIFO (that is, writes to SDRAM are completed) before the read transfer is performed.

5.5.4 Multiprocessor support

The two FIFOs operate independently, as described above, and can be accessed at the same time. This makes it possible for a local processor to read local SDRAM over the system bus (through both FIFOs). This feature can be used to support multiprocessor systems that share data in SDRAM because the processors can all access the same DRAM locations at the same addresses.

5.5.5 System bus signal routing

The core module is mounted onto a motherboard using the connectors HDRA and HDRB. As well as carrying all signal connections between the boards, these provide mechanical mounting (see *Attaching the core module to an Integrator/AP motherboard* on page 2-7).

HDRA

The signals on the HDRA connectors are tracked between the socket on the underside and the plug on the top so that pin 1 connects to pin 1, and pin 2 to pin 2 for example. That is, the signals are routed straight through.

HDRB

Several signals on the HDRB connectors are rotated in groups of four between the connectors on the bottom and top of each module. This ensures that each processor (or other bus master device) on a module connects to the correct signals according to whether it is bus master 0, 1, 2, or 3. The ID for the bus master on a module is determined by the position of the module in the stack.

This signal rotation scheme is illustrated in Figure 5-9.

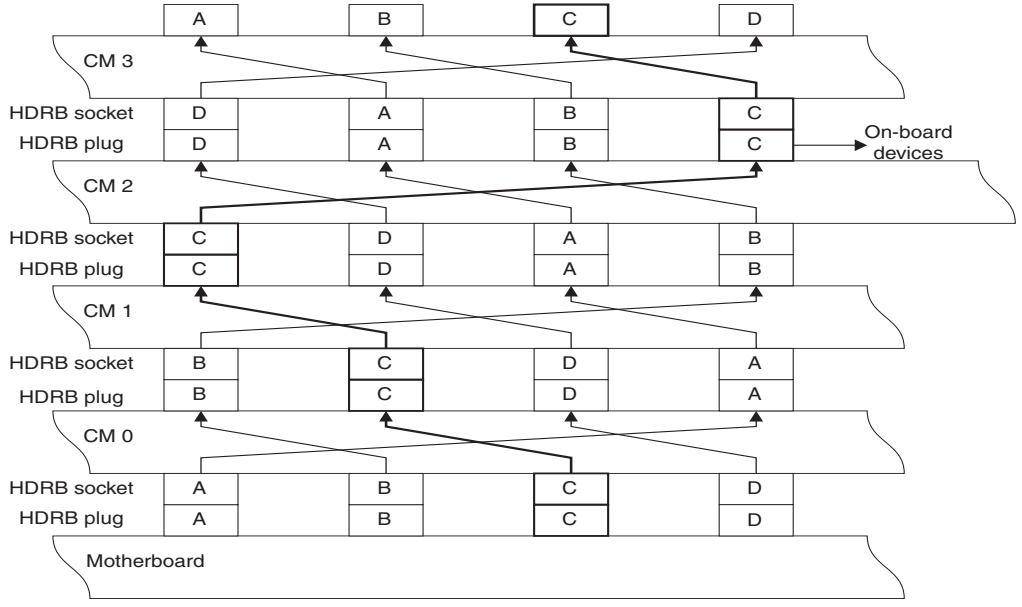


Figure 5-9 Signal rotation on HDRB, Integrator/AP

The example in Figure 5-9 on page 5-15 shows how a group of four signals (labeled A, B, C, and D) are routed through the stack. Signal C is rotated as it passes up through the stack and only utilized on module 2.

All four signals are rotated and utilized in a similar way, as follows:

- signal A on core module 0
- signal B on core module 1
- signal C used on core module 2
- signal D used on core module 3.

For details of the signals on the HDRB connectors, see *HDRB* on page A-5.

Note

The JTAG signals are described in *Multi-ICE support* on page 3-16.

5.5.6 Bus operating modes

The bus operating modes are programmed by writing to coprocessor 15 register 1 within the microprocessor core.

The Integrator/CM920T, CM920T-ETM, and CM940T support:

- asynchronous and FastBus clocking
- little-endian addressing.

The Integrator/CM920T, CM920T-ETM, and CM940T do not support:

- synchronous clocking
- big-endian addressing.

For details of how to set the bus operating parameters, refer to the *ARM920T Technical Reference Manual* or *ARM940T Technical Reference Manual*.

5.6 Reset controller for AP

The core module FPGA incorporates a reset controller that enables the core module to be reset as a standalone unit or as part of an Integrator development system. The core module can be reset from five sources:

- reset button
- motherboard
- other core or logic modules
- Multi-ICE
- software.

Figure 5-10 shows the architecture of the reset controller.

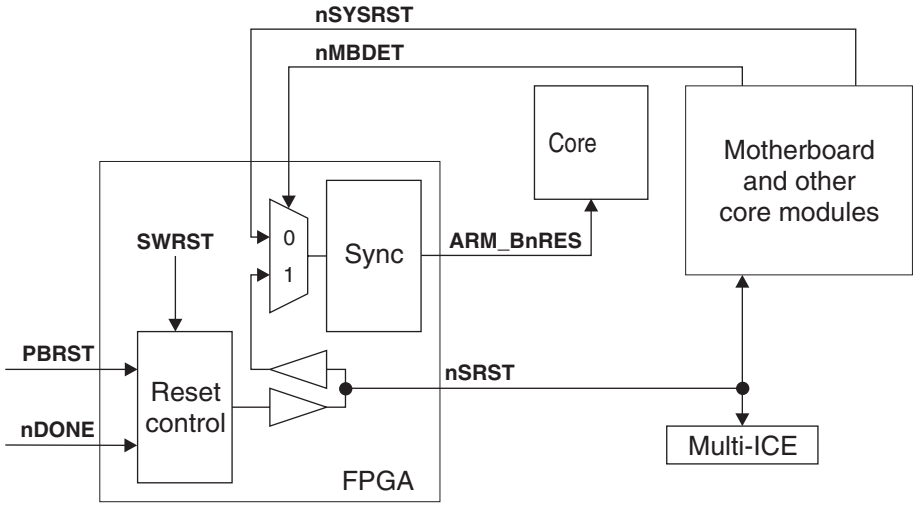


Figure 5-10 Core module reset controller

5.6.1 Reset control signals

Table 5-4 describes the external reset signals.

Table 5-4 Reset signal descriptions

Name	Description	Type	Function
ARM_BnRES	Processor reset	Output	The **ARM_BnRES** signal is used to reset the processor core. It is generated from **nSRST** LOW when the core module is used standalone, or **nSYSRST** LOW when the core module is attached to a motherboard. It is asserted as soon as the appropriate input becomes active. It is deasserted synchronously from the falling edge of the processor bus clock.
nDONE	FPGA configured	Input	The **nDONE** signal is an inversion of the open collector signal **FPGADONE** and is generated by all FPGAs when they have completed their configuration. The **FPGADONE** signal is routed round the system through the HDRB connectors to the inputs of all other FPGAs in the system. The signal **nSRST** is held asserted until **nDONE** is driven LOW.
nMBDET	Motherboard detect	Input	The **nMBDET** signal is pulled LOW when the core module is attached to a motherboard and HIGH when the core module is used standalone. When **MBDET** is LOW, **nSYSRST** is used to generate the **ARM_BnRES** signal. When **nMBDET** is HIGH, **nSRST** is used to generate the **ARM_BnRES** signal.
PBRST	Push-button reset	Input	The **PBRST** signal is generated by pressing the reset button.
nSRST	System reset	Bidirectional	The **nSRST** open collector output signal is driven LOW by the core module FPGA when the signal **PBRST** or software reset (**SWRST**) is asserted. As an input, **nSRST** can be driven LOW by Multi-ICE. If there is no motherboard present, the **nSRST** signal is synchronized to the processor bus clock to generate the **BnRES-M** signal.
nSYSRST	System reset	Input	The **nSYSRST** signal is generated by the system controller FPGA on the motherboard. It is used to generate the **ARM_BnRES** signal when the core module is attached to a motherboard. It is selected by the motherboard detect signal, **nMBDET**.

5.6.2 Software resets

The core module FPGA provides a software reset that can be triggered by writing to the reset bit in the CM_CTRL Register. This generates the internal reset signal **SWRST** that generates **nSRST** and resets the whole system (see *Core Module Control Register* on page 4-12).

5.7 Interrupt control for AP

Figure 5-11 on page 5-22 shows the interrupt control architecture for the Integrator/AP system.

The system controller FPGA on the motherboard incorporates interrupt controllers that route the various interrupts from around the system onto the **nFIQ** and **nIRQ** pins of up to four processors. The interrupts that a core module receives are determined by the position of the core module within the stack, as shown in Table 5-5.

Table 5-5 Core module interrupts

Module ID	Interrupt	Fast interrupt
3 (top)	**nIRQ3**	**nFIQ3**
2	**nIRQ2**	**nFIQ2**
1	**nIRQ1**	**nFIQ1**
0 (bottom)	**nIRQ0**	**nFIQ0**

The interrupt signals are routed to the core module using pins on the HDRB connectors (see *HDRB* on page A-5).

The interrupts and fast interrupts are enabled and handled using the interrupt control registers on the motherboard (see the user guide for your motherboard).

The Integrator/AP FPGA provides interrupt controllers to handle IRQs and FIQs from around the system. See the *Integrator/AP User Guide* for more details.

The debug comms interrupts registers are described in *Core module debug comms interrupt registers* on page 4-22.

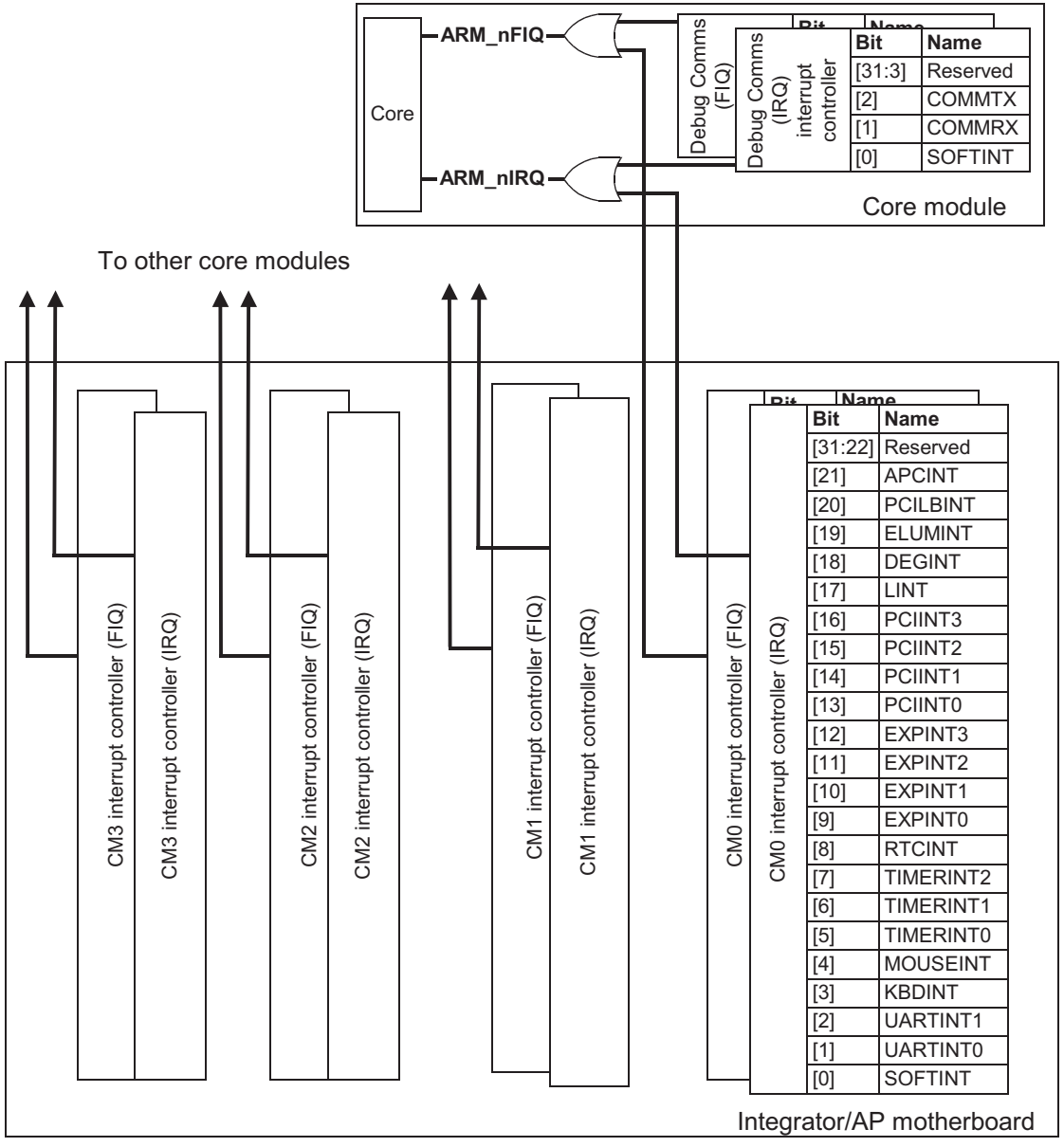


Figure 5-11 Interrupt architecture, AP image

Chapter 6
Using Core Modules with an Integrator/CP

This chapter contains the instructions for using an ARM Integrator core module with an Integrator/CP baseboard. It contains the following sections:

6.1 About the CP system architecture

Figure 6-1 shows the architecture of a system consisting of an Integrator/CP baseboard and a core module.

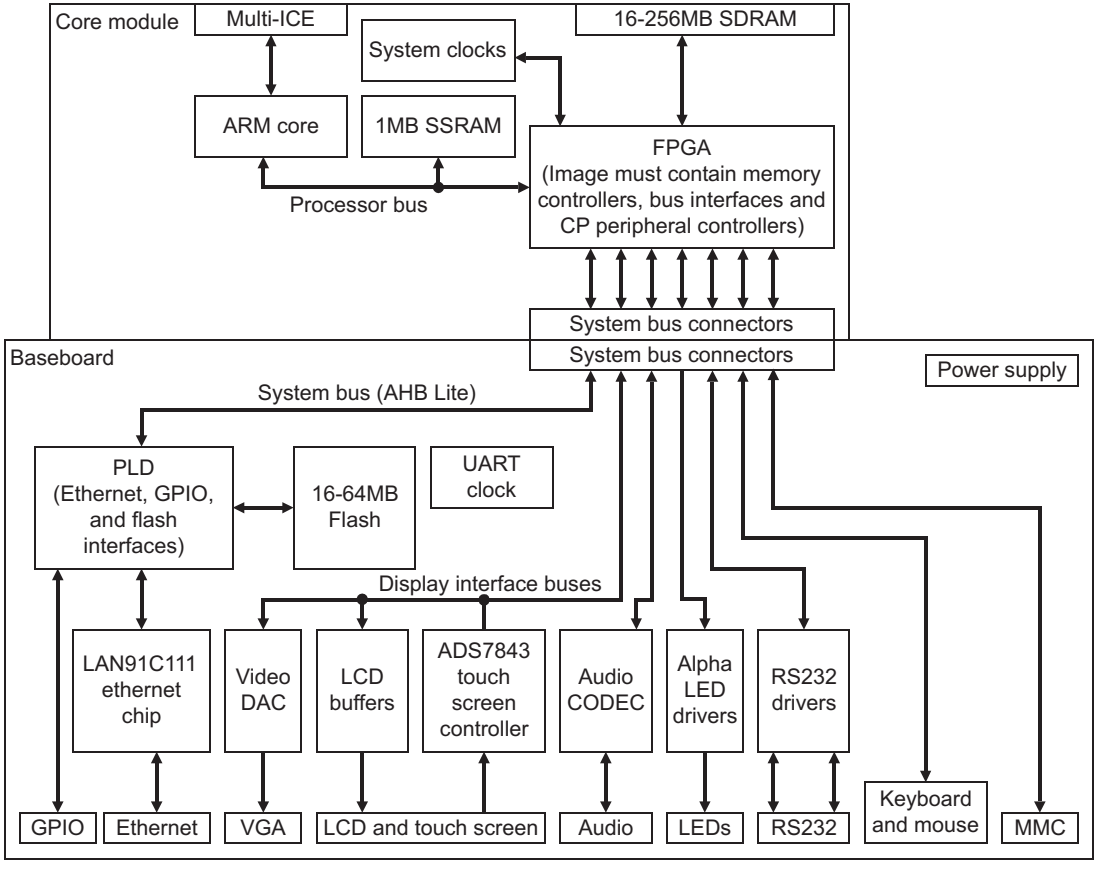


Figure 6-1 Integrator/CP system architecture

6.1.1 Integrator/CP system buses

Figure 6-2 on page 6-3 shows how the external system bus and the internal busses connect the various boards together.

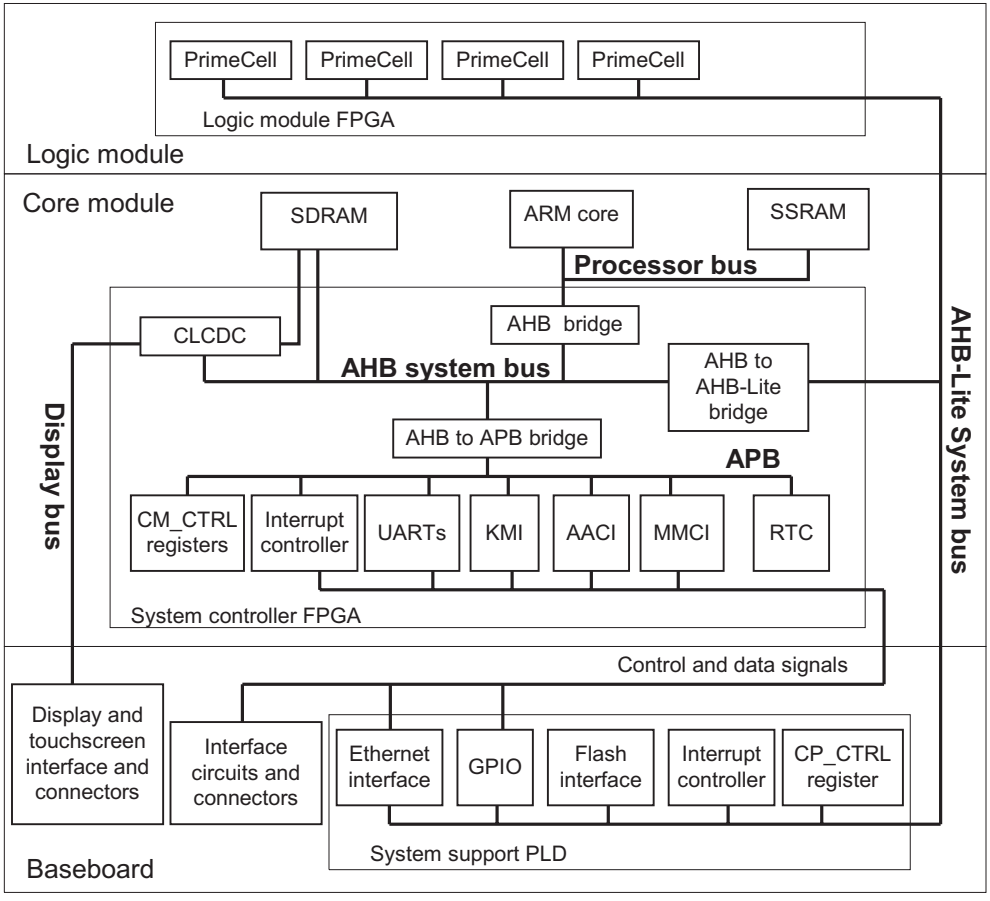


Figure 6-2 Bus routing between baseboard, core module, and logic module

System bus routing and bus interfaces

The Integrator/CP and core module use an AMBA system bus comprised of:

AHB system bus

This is the processor bus within the core module FPGA. The bus bridges connect it to the AHB-Lite external system bus and to the APB peripheral bus.

AHB-Lite system bus

This is the external bus present on the HDRA and HDRB connectors. It connects the baseboard, core module, and optional logic modules. The AHB-Lite system bus is a single-master bus with the processor core as sole bus master.

Peripheral bus

This is the APB bus present within the core module FPGA. It connects the system bus to the APB peripherals and control registers.

APB peripheral bus

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. The APB is not pipelined. Wait states are added during transfers between the APB and AHB when the AHB is required to wait for the APB protocol.

Figure 6-3 shows some example APB peripherals that are implemented using PrimeCell or ADK devices synthesized into the FPGA on the core module.

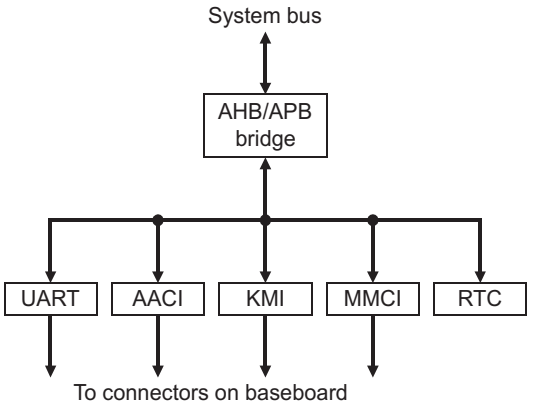


Figure 6-3 APB peripherals

System bus control

For the Integrator/AP, the system controller FPGA on the baseboard normally provides control for the system bus (bus arbitration and address decoding down to module level). For the Integrator/CP baseboard however, the external bus is AHB-Lite with only one master (the core module). The **SREQn** and **SGNTn** signals are not used to determine the bus master.

The core module is at a fixed location but logic modules added to the system must each provide address decoding for their assigned region in the memory map.

6.1.2 Module ID selection and interrupt routing

Several signals are routed between the HDRB plug and socket on the core module and logic modules so that they rotate up through the stack. The signal rotation eliminates the requirement changing jumpers on a board if its position in the stack is changed. The interrupt and memory control is based on the cards position in the stack and the signal rotation.

The position of a logic module in the stack is used to determine its ID. The address in the alias memory region and the interrupts that it responds to are based on this ID. (See *Module ID selection for AP* on page 5-3 and *Interrupt routing between Integrator modules* on page 6-25.)

The Integrator/CP can only have one core module (immediately on the baseboard), but it can have up to three logic modules. The core module ID is always 0. Logic module IDs are 1 to 3, depending on position.

6.1.3 Configuring little or big-endian operation

The Integrator/CP can be configured to operate as a big-endian or 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 must only be done at the start of a debugging session. The change must have taken effect before you can perform any subword accesses.

Caution

The Integrator/CP display system and Ethernet controller cannot operate in big-endian mode.

6.2 Top-level CP memory map

Figure 6-4 shows the top-level memory map of an Integrator/CP system. The memory map maintains compatibility with other platforms in the Integrator product family. This ensures code portability and enables you to expand the system with additional Integrator logic modules.

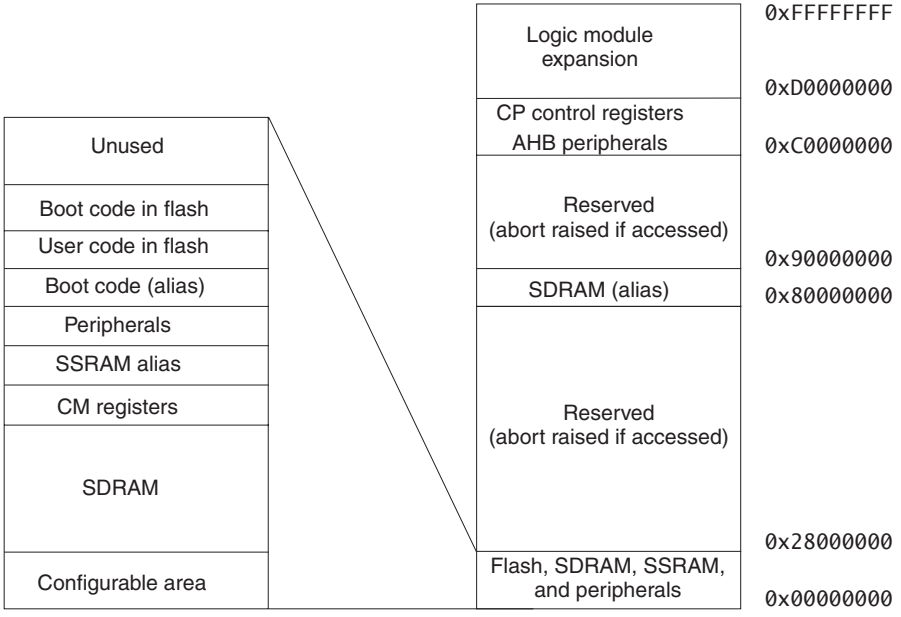


Figure 6-4 Top-level memory map

The memory map contains areas for the SSRAM, SDRAM, flash memory, and peripherals. It also contains an area reserved for expansion at `0xD0000000`–`0xFFFFFFFF`.

6.2.1 Physical location of memory chips

Figure 6-4 on page 6-6 shows how the different memory types are physically located in an Integrator/CP system.

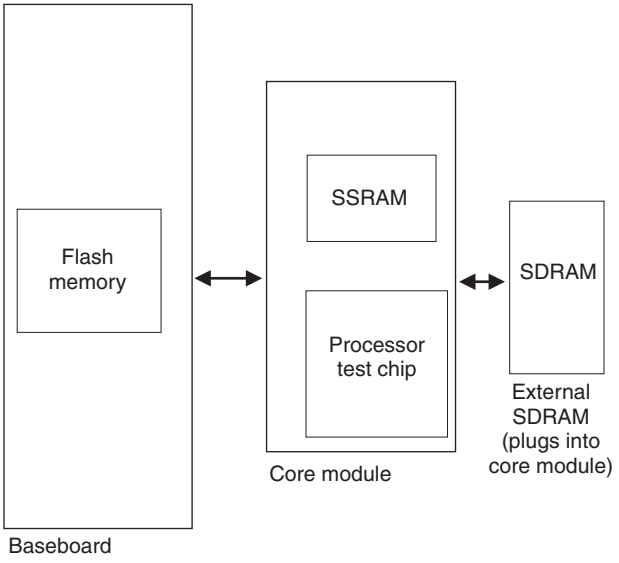


Figure 6-5 Top-level memory map

6.2.2 Configurable area of memory map

The addressing of memory is partially configurable. Reasons for configuring memory addresses include:

- The flash memory is located at `0x24000000`. The memory area at `0x0` can be configured to be either RAM or flash (REMAP =0). Placing flash at `0x0` enables you to use boot code in the flash at startup, and placing RAM at `0x0` provides a mechanism for loading custom interrupt vectors and vector routines.
- Some applications require large amounts of memory, or require large amounts during the development phase. The core modules can be extended with SDRAM when the core module RAM is not sufficient. A SDRAM DIMM is supplied with the core module.

Table 6-1 shows a typical memory map and the effect of REMAP.

Table 6-1 REMAP operation

Address range	Size	Description
0x00000000–0x000FFFFF	1MB	(REMAP=1) SSRAM or (REMAP=0) boot code area of flash
0x00100000–0x0FFFFFFF	256MB	SDRAM (repeats physical memory to fill space) Part of first SDRAM image is masked by SSRAM or flash
0x10000000–0x107FFFFF	8MB	Core module registers
0x10800000–0x11FFFFFF	8MB	On-board SSRAM (repeats physical memory to fill space)
0x24000000–0x27FFFFFF	64MB	Baseboard flash memory (contains user code and boot code)
0x11000000–0x1FFFFFFF	256MB	Peripherals
0x20000000–0x23FFFFFF	64MB	Boot code alias
0x28000000–0x7FFFFFFF	1.4GB	Reserved (abort if accessed)
0x80000000–0x8FFFFFFF	256MB	SDRAM (alias)
0x90000000–0xBFFFFFFF	768MB	Reserved (abort if accessed)
0xC0000000–0xCFFFFFFF	256MB	CP control registers and APB peripherals
0xD0000000–0xFFFFFFFF	768MB	Logic module address space

6.2.3 Baseboard 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 flash size, 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 See also the *Integrator/CP User Guide*.

6.2.4 Memory timing

Accesses to memory require a different number of cycles depending on the type of memory as shown in Table 6-2.

Table 6-2 Wait states for memory access

Memory type	CP920T	CP940T
SSRAM read or write	0	0
SDRAM read typical (no bus conflict)	1 to 6	0–5
SDRAM read maximum (write in progress before read)	32	31
Flash read	7	6

6.3 Programmable logic on CP

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

- *Baseboard system support PLD*
- *System controller FPGA* on page 6-11.

6.3.1 Baseboard system support PLD

The system support PLD is located on the baseboard. It provides AHB slave interfaces for the Ethernet and flash, an interrupt controller, and an 8-bit GPIO interface. Figure 6-6 shows the internal architecture of the programmed PLD.

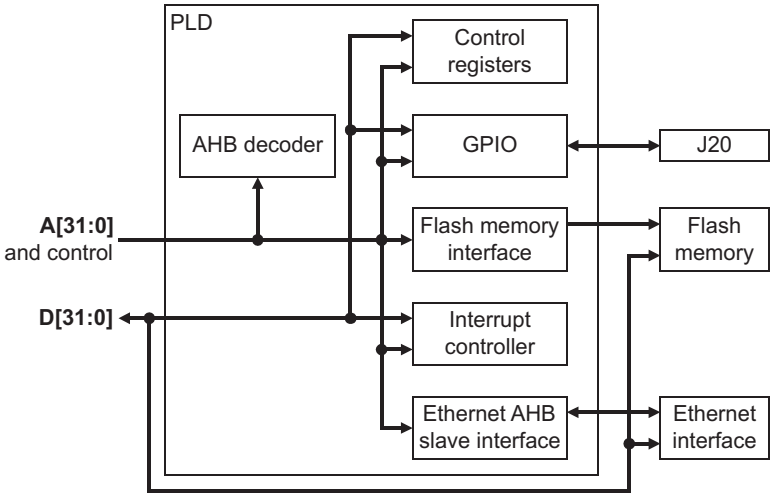


Figure 6-6 Baseboard PLD

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).

6.3.2 System controller FPGA

The system controller FPGA for an Integrator/CP system is located on the core module. It contains the main bus bridges, memory controllers, and peripheral controllers. Figure 6-7 shows a simplified block diagram of a system controller FPGA. (The details of the system controller differ with different processor cores.)

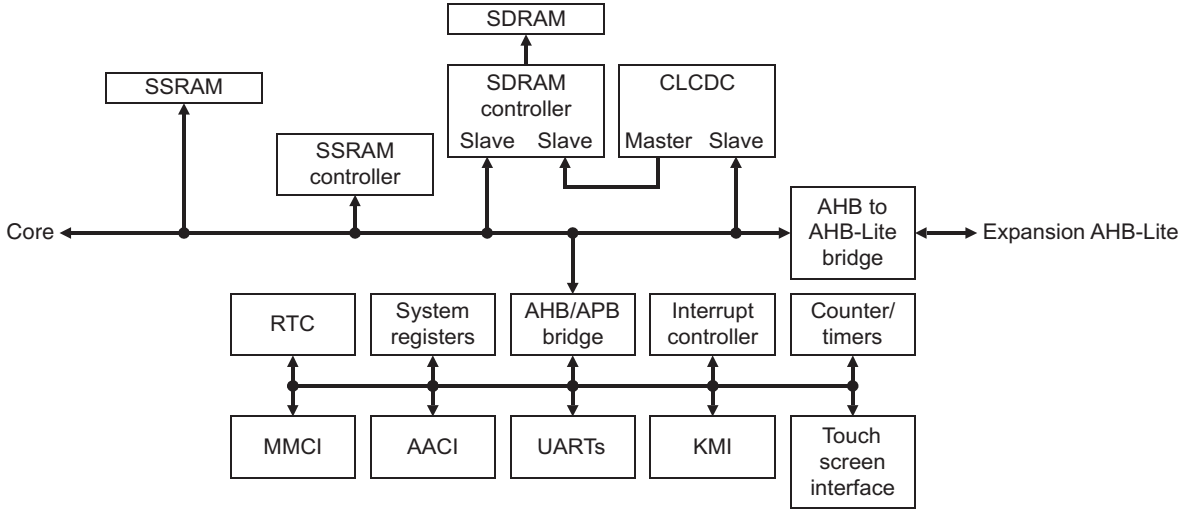


Figure 6-7 System controller FPGA block diagram

Note

The core module FPGA must have an appropriate Integrator/CP-compatible image to control the peripherals on the Integrator/CP baseboard. Current core modules contain an Integrator/CP FPGA image. For earlier core modules, contact your supplier for an updated image.

The FPGA on the core module, and on logic modules added to the system, share the open-collector signal **GLOBAL_DONE**. When the system is powered on, the FPGA loads a configuration from the configuration flash memory on the core module into the configuration inputs of the FPGA. The **GLOBAL_DONE** signal is used by the FPGAs to signal that configuration is complete and system reset can be completed.

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

The configuration flash can store up to four images. The images are selected by the **CFGSEL[1:0]** signals from the baseboard as shown in Table 6-3.

Table 6-3 Image selection

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

6.3.3 HDL files

The core module image provides the CP functional blocks as a set of HDL files. These are described in Table 6-4.

Table 6-4 CM image functional block HDL file descriptions

Block	Description
`AHBDecoder`	The decoder block provides the high-speed peripherals with select lines. These are generated from the address lines and the module ID (position in stack) signals from the baseboard. The decoder blocks also contain the default slave peripheral to simplify the example structure. The Integrator family of boards uses a distributed address decoding system.
`AHBMuxS2M`	This is the AHB multiplexor that connects the read data buses from all of the slaves to the AHB master(s).
`AHB2APB`	This is the bridge block required to connect APB peripherals to the high-speed AMBA AHB bus. It produces the peripheral select signals for each of the APB peripherals.
`CMRegs`	The APB register peripheral provides memory-mapped registers that you can use to: • configure the two clock generators (protected by the LM_LOCK Register) • write to the user LEDs • read the user switch inputs.
`CMIntcon`	The APB interrupt controller contains all of the standard interrupt controller registers and has an input port for four APB interrupts. (The example only uses one of them. The remaining three are set inactive in the AHBAPBSys block.) Four software interrupts are implemented.

6.4 Register and memory overview for CP

Table 6-5 shows a map for a typical Integrator/CP system. See the *Integrator/CP User Guide* for more details on register function.

Note

The memory map below `0x10000000` depends on the settings for SSRAM mode and REMAP.

Table 6-5 System control register map

Peripheral or device	Address range	Size
Boot flash. Mapped at this address only at power ON, and then can be disabled under software control to allow access to SSRAM. **Note** The memory map below `0x10000000` depends on the setting REMAP.	`0x00000000–0x000FFFFF`	1MB
SDRAM.	`0x00010000–0x0FFFFFFF`	255MB
Core Module registers.	`0x10000000–0x107FFFFF`	8MB
SSRAM alias	`0x10800000–0x10FFFFFF`	8MB
Counter/timers.	`0x13000000–0x13FFFFFF`	16MB
Primary Interrupt Controller Registers.	`0x14000000–0x14FFFFFF`	16MB
Real time clock.	`0x15000000–0x15FFFFFF`	16MB
UART0.	`0x16000000–0x16FFFFFF`	16MB
UART1.	`0x17000000–0x17FFFFFF`	16MB
Keyboard.	`0x18000000–0x18FFFFFF`	16MB
Mouse.	`0x19000000–0x19FFFFFF`	16MB
Debug LEDs and DIP switch.	`0x1A000000–0x1AFFFFFF`	16MB
Reserved.	`0x1B000000–0x1BFFFFFF`	16MB
Multimedia Card Interface.	`0x1C000000–0x1CFFFFFF`	16MB

Table 6-5 System control register map (continued)

Peripheral or device	Address range	Size
Advanced Audio CODEC Interface.	0x1D000000–0x1DFFFFFF	16MB
Touch Screen Controller Interface.	0x1E000000–0x1EFFFFFF	16MB
Default.	0x28000000–0xBFFFFFFF	2416MB
CLCD regs/palette.	0xC0000000–0xC0FFFFFF	16MB
Default.	0xC1000000–0xC7FFFFFF	6MB
Ethernet.	0xC8000000–0xC8FFFFFF	16MB
GPIO.	0xC9000000–0xC9FFFFFF	16MB
Secondary interrupt controller.	0xCA000000–0xCAFFFFFF	16MB
CP Control Registers.	0xCB000000–0xCBFFFFFF	16MB
Default.	0xCC000000–0xFFFFFFFF	816MB

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.

6.4.1 Core Module Control Register, CP

This section describes the CM_CTRL Register that is modified in the CP image. The other CM registers are unchanged (see *Core module control registers* on page 4-9).

The Core Module Control Register, CM_CTRL, at 0x1000000C is a read/write register that controls several user-configurable features of the core module and the display interface on the baseboard.

Figure 6-8 on page 6-15 shows the Core Module Control Register bits.

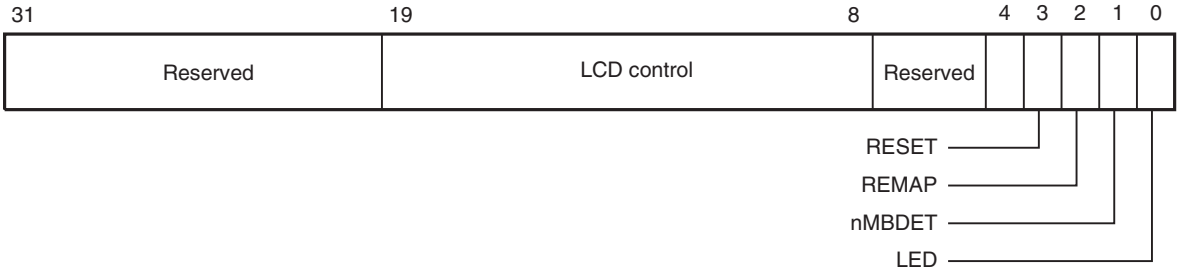


Figure 6-8 CM_CTRL Register as modified for CP

Table 6-6 describes the Core Module Control Register bits.

Table 6-6 CM_CTRL Register bit description

Bits	Name	Access	Function
[31:20]	Reserved	Use read-modify-write to preserve value.	
[19]	n24BITEN	Write	Select VGA depth: 0 = 24bit VGA 1 = 18bit VGA Must be 1 to enable BIAS control
[18]	STATIC	Write	No connection on Sharp panel
[17]	STATIC2	Write	Up/down axis flip on Sharp panel
[16]	STATIC1	Write	Right/left axis flip on Sharp panel
[15]	Enable LCD1	Write	Enable, active HIGH
[14]	Enable LCD0	Write	Enable, active HIGH
[13:11]	LCDMUXSEL	Write	001 = Generic LCD connector, 24-bit mode 011 = Sharp LCD panel 100 = Sharp LCD panel 111 = 24-bit VGA
[10]	LCDBIASDN	Write	Low to high transition decreases LCD bias voltage (dimmer)
[9]	LCDBIASUP	Write	Low to high transition increases LCD bias voltage (brighter)
[8]	LCDBIASEN	Read/write	Enable LCD bias supply

Table 6-6 CM_CTRL Register bit description (continued)

Bits	Name	Access	Function
[7:4]	Reserved	Use read-modify-write to preserve value.	
[3]	RESET	Write	This is used to reset the core module, the baseboard on which it is mounted, and any modules in a stack. A reset is triggered by writing a 1. Reading this bit always returns a 0 allowing you to use read-modify-write operations without masking the RESET bit.
[2]	REMAP	Read/write	This bit is used to control REMAP: 0 = flash at address 0 1= SRAM at address 0.
[1]	nMBDET	Read	This bit indicates whether or not the core module is mounted on a baseboard: 0 = mounted on baseboard 1 = reserved.
[0]	LED	Read/write	This bit controls the green MISC LED on the core module: 0 = LED OFF 1 = LED ON.

6.5 Peripherals and interfaces for CP

This section provides a brief description of the peripherals on the Integrator/CP:

- *Clock control*
- *Counter/timers*
- *Real-time clock* on page 6-18
- *UARTs* on page 6-18
- *Keyboard and mouse interface* on page 6-18
- *MMC interface* on page 6-18
- *Audio interface* on page 6-19
- *Touchscreen controller interface* on page 6-19
- *Display interface* on page 6-19
- *Ethernet interface* on page 6-20.

For more detail, see the *Integrator/CP User Guide* and the documentation for the PrimeCell peripherals.

6.5.1 Clock control

The baseboard has two Micrologic IC525 clock generators that provide:

- the 12.288MHz AACI bit clock
- the 14.7456MHz UART clock
- the 25MHz Ethernet clock.

See also *Core Module Oscillator Register* on page 4-12.

6.5.2 Counter/timers

The core module FPGA provides three 32-bit counter/timers that can operate in three 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.

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

6.5.3 Real-time clock

The *Real-Time Clock* (RTC) comprises the following elements:

- a 32-bit counter
- a 32-bit match register
- a 32-bit comparator.

The 32-bit counter increments on successive rising edges of a 1Hz clock generated by the core module FPGA. You load a start value by writing to the load register RTC_LR and read the current value of the counter from the data register RTC_DR.

You program a match register by writing to the match register RTC_MR and you can read the current value at any time. When the counter and match register contents are identical, an interrupt request is asserted.

6.5.4 UARTs

The serial interface is implemented with two PrimeCell UARTs instantiated into the core module FPGA. Transceivers and connectors for the serial interfaces are provided on the Integrator/CP baseboard and signals between the core module and base board are routed through the HDRB connectors.

The UARTs are functionally similar to standard 16C550 devices.

For detailed information about the UART, see the *UART (PL011) Technical Reference Manual* and the *Integrator/CP User Guide*.

6.5.5 Keyboard and mouse interface

The keyboard and mouse controllers are implemented with two PrimeCell *Keyboard and Mouse Interfaces* (KMIs) instantiated into the core module FPGA. Connectors for these interfaces are provided on the baseboard and signals between the core module and baseboard are routed through the HDRB connectors. The KMI generates an interrupt when a byte can be written or read from the data registers.

6.5.6 MMC interface

The PrimeCell *MultiMedia Card Interface* (MMCI) is instantiated into the core module FPGA. A card connector is provided by the baseboard and the interface signals are routed between the core module and baseboard using the HDRB connectors.

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

6.5.7 Audio 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 and signals the core module and baseboard are routed through the HDRB connectors.

Note

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

For detailed information on the AACI, see *ARM PrimeCell Advanced Audio CODEC Interface (PL041) Technical Reference Manual*.

6.5.8 Touchscreen controller interface

The touchscreen interface driven by the *TouchScreen Controller Interface* (TSCI) instantiated into the core module FPGA and the baseboard provides external connectors. Interface signals are routed between the core module and baseboard using the HDRB connectors.

6.5.9 Display interface

The touchscreen controller that accompanies the LCD is described in *Touchscreen controller interface*. See also the *ARM PrimeCell Color LCD Controller (PL110) Technical Reference Manual*.

The Integrator/CP provides a flexible display interface that provides support for two types of color LCD displays or a VGA display. The core module provides a PrimeCell *Color LCD Controller* (CLCDC) instantiated into the FPGA. Display interface signals are routed using the B bus signals on the HDRA connectors.

Note

In addition to the PrimeCell LCD registers, there are some LCD control bits in the CM_CTRL Register at `0x1000000C`. For more information, see *Core module control registers* on page 4-9 and *Core Module Control Register, CP* on page 6-14.

6.5.10 Ethernet interface

The Ethernet interface is implemented with a SMC LAN91C111 10/100 Ethernet single-chip MAC and PHY on the baseboard. This is provided with a slave interface to the system bus by the FPGA on the baseboard. The Ethernet interface is described in the *Integrator/CP Baseboard User Guide*.

6.6 Reset controller for CP

The core module FPGA incorporates a reset controller that enables the core module to be reset as a standalone unit or as part of an Integrator development system. The core module can be reset from four sources:

- reset button
- logic modules
- Multi-ICE
- software.

Figure 6-9 shows the architecture of the reset controller.

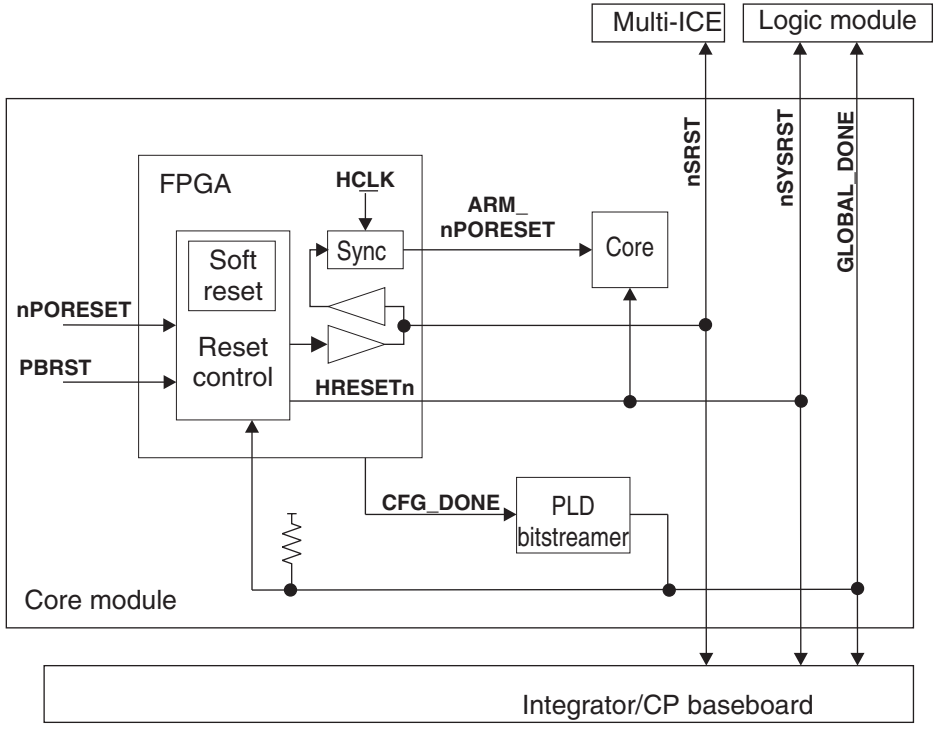


Figure 6-9 Core module reset controller

6.6.1 Reset control signals

Table 6-7 describes the external reset signals.

Table 6-7 Reset signal descriptions

Name	Description	Type	Function
ARM_nPORESET	Processor power-on reset	Output	The **ARM_nPORESET** signal is used to reset the processor core at power on. It is generated from **nSRST** LOW and synchronized to **HCLK**.
PBRST	Push-button reset	Input	The **PBRST** signal is generated by pressing the reset button.
nSRST	System reset	Bidirectional	The **nSRST** open collector output signal is driven LOW by the core module FPGA when the signal **PBRST** or software reset (**SWRST**) is asserted. As an input, **nSRST** can be driven LOW by Multi-ICE. The **nSRST** signal is synchronized to the processor bus clock to generate the **ARM_nPORESET** signal.
nSYSRST	System reset	Output	The **nSYSRST** signal is generated by the system controller FPGA on the core module. It is used to generate **HRESETn** to the test chip and the system.

6.6.2 Software resets

The core module FPGA provides a software reset that can be triggered by writing to the reset bit in the CM_CTRL Register. This generates the internal reset signal **SWRST** that generates **nSRST** and resets the whole system.

6.7 Interrupt control for CP

Figure 6-10 shows the interrupt control architecture for the Integrator/CP system. The interrupts are described in the following sections:

- *Interrupt controllers* on page 6-24
- *Interrupt routing between Integrator modules* on page 6-25
- *CP image interrupt control registers* on page 6-27
- *Handling interrupts* on page 6-30.

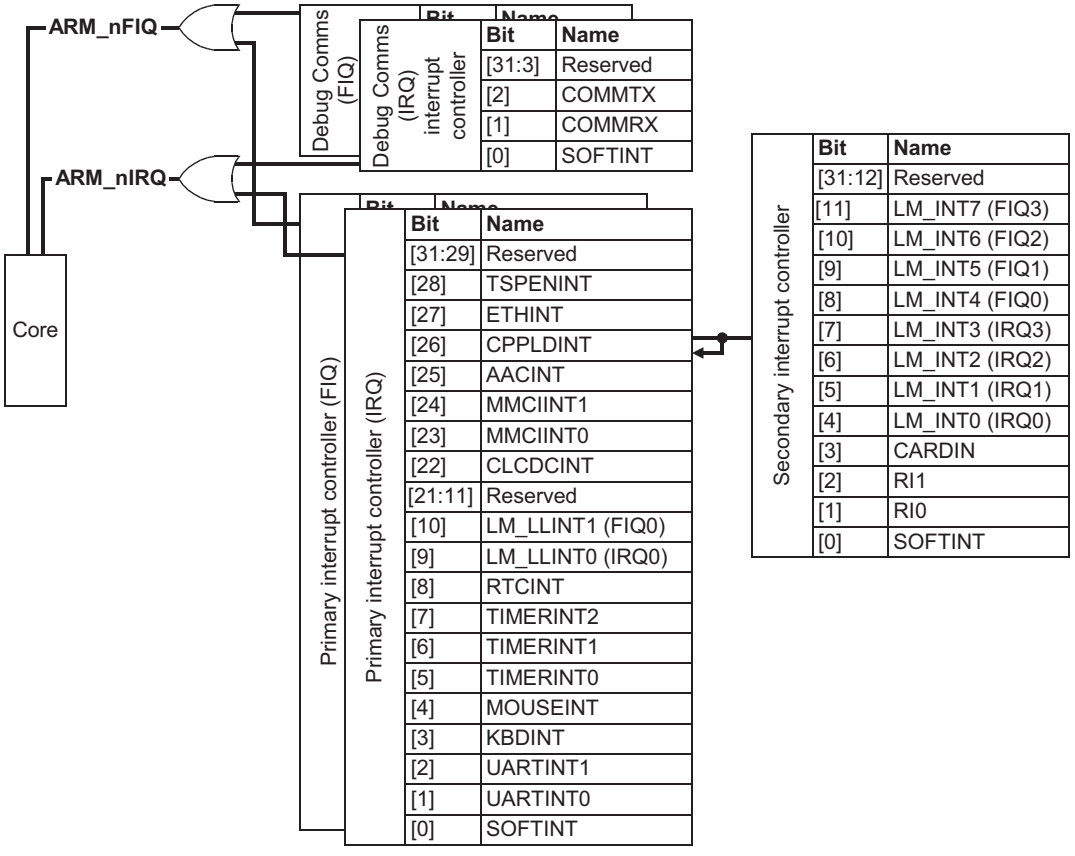


Figure 6-10 Interrupt architecture, CP image

6.7.1 Interrupt controllers

Integrator/CP system interrupts are generated by the *Primary Interrupt Controller* (PIC), the *Secondary Interrupt Controller* (SIC), and the *Communications Interrupt Controller* (CIC).

Detecting and clearing interrupts requires that each interrupt controller be correctly initialized as well as the interrupt control register in the individual peripheral.

Primary interrupt controller

The PIC is implemented within the core module FPGA and handles the majority of interrupts from the system. A substantial number of interrupts are reserved to maintain compatibility with other modules within the Integrator family. The PIC provides a set of registers to control and handle interrupts. These are described in *Primary Interrupt Controller Registers* on page 6-27.

Secondary interrupt controller

The SIC is implemented in the 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.

The secondary controller provides a set of registers to control and handle interrupts. These are described in the *Integrator/CP Baseboard User Guide*.

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. For a description of the debug communications channel, see the Technical Reference Manual for your core.

6.7.2 Interrupt routing between Integrator modules

Figure 6-11 shows how the IRQ signals are routed from modules to the interrupt controllers.

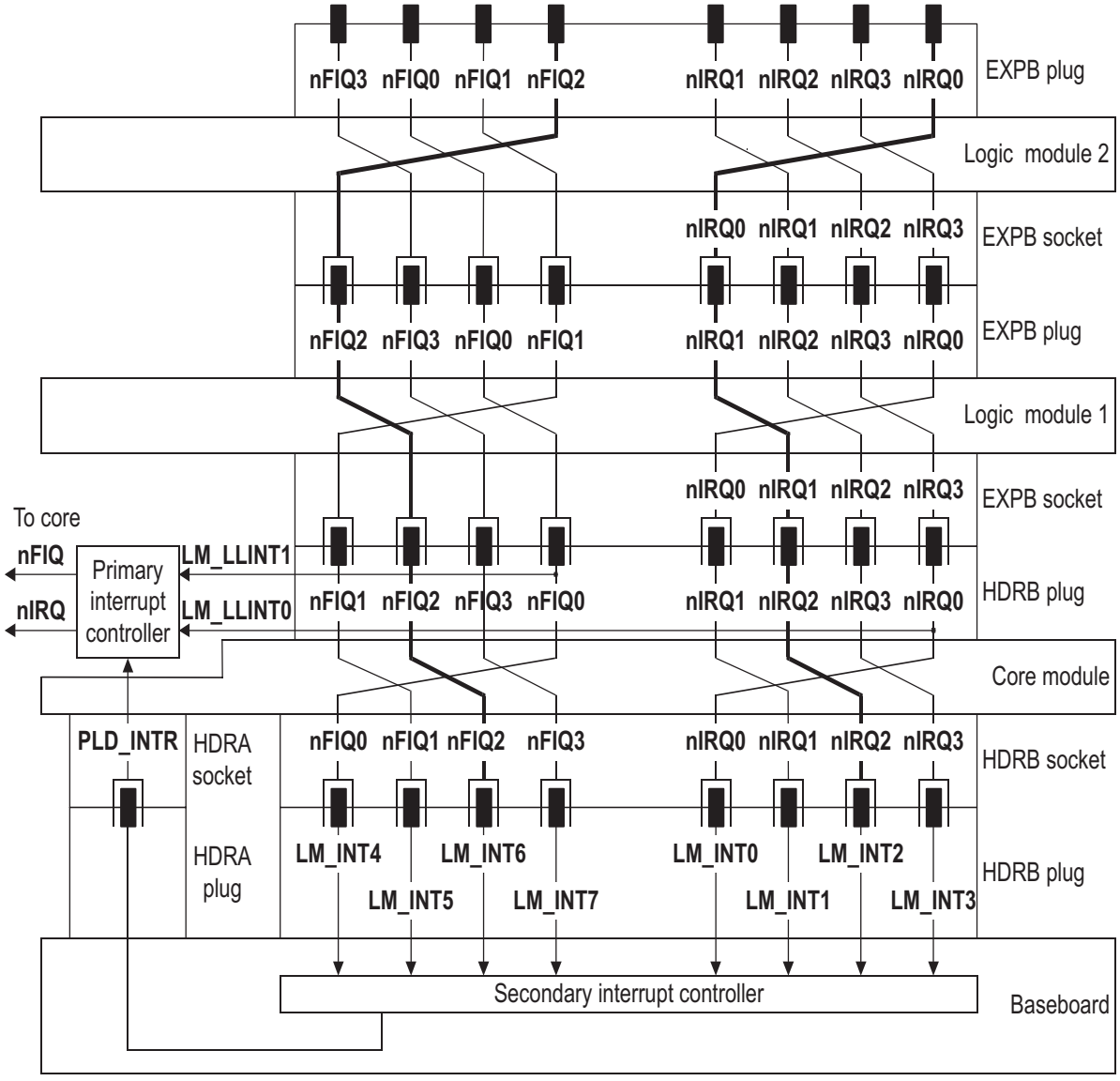


Figure 6-11 Interrupt signal routing

The diagram shows 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. A similar routing applies for logic module 1 and 3. The operation of the interrupts relies on the core module being mounted on the baseboard first with any logic modules on top.

The signals route through the SIC. Determining the source of an interrupt requires 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.

Note

Ensure that the correct interrupt line is driven. See the routing pattern in Figure 6-11 on page 6-25.

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.

6.7.3 CP image interrupt control registers

The baseboard FPGA provides an interrupt controller to handle IRQs and FIQs from around the system. These are in addition to the CM interrupt controller (see *Core module control registers* on page 4-9), and are described in:

- *Primary Interrupt Controller Registers*
- *Secondary Interrupt Controller Registers* on page 6-29
- *Debug Communications Interrupt Registers* on page 6-29
- *Soft Interrupt Set Register and Soft Interrupt Clear Register* on page 6-29.

Primary Interrupt Controller Registers

The CP image for the FPGA provides interrupt controllers for both IRQs and FIQs that maintain compatibility with other Integrator modules to ensure code portability. The Primary Interrupt Control Registers are listed in Table 6-8.

Table 6-8 Primary Interrupt Controller 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-only	22	FIQ gated interrupt status
0x14000024	PIC_FIQ_RAWSTAT	Read-only	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

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 6-9.

Table 6-9 Primary Interrupt Controller Register bit descriptions

Bits	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 Registers* on page 6-29.
[25]	AACIINT	Audio interface interrupt
[24]	MMCIINT1	MultiMedia card interface
[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

Debug Communications Interrupt Registers

The Debug Comms Interrupt Registers are described in *Core module debug comms interrupt registers* on page 4-22.

Secondary Interrupt Controller Registers

The Secondary Interrupt Controller Registers are listed in Table 6-10

Table 6-10 Secondary Interrupt Controller 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

For more details, see the *Integrator/CP Baseboard User Guide*.

Soft Interrupt Set Register and Soft Interrupt Clear Register

The primary, secondary, and comms interrupt controllers provide a register for controlling and clearing software interrupts.

Use the set and clear procedure described in *Core Module Soft Interrupt Set and Core Module Soft Interrupt Clear Registers* on page 4-25 for registers PIC_INT_SOFTSET, PIC_INT_SOFTCLR, SIC_INT_SOFTSET, and SIC_INT_SOFTCLR.

6.7.4 Handling interrupts

This section describes interrupt handling and clearing in general. For examples of interrupt detection and handling, see the *install_directory*\platform\software\selftest directory on the CD, the *ARM Firmware Suite User Guide*, the *ARM Developer Suite Developer Guide*, and the technical reference manual for your processor.

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. For 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
```


Appendix A
Signal Descriptions

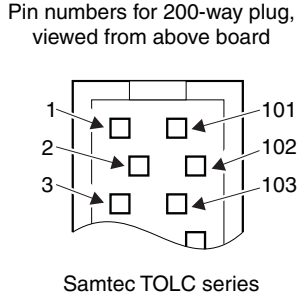
This index provides a summary of signals present on the core module main connectors. It contains the following sections:

Note

For the Multi-ICE connector pinout and signal descriptions see *JTAG signals* on page 3-21.

A.1 HDRA

Figure A-1 shows the pin numbers of the HDRA plug and socket. All pins on the HDRA socket are connected to the corresponding pins on the HDRA plug.



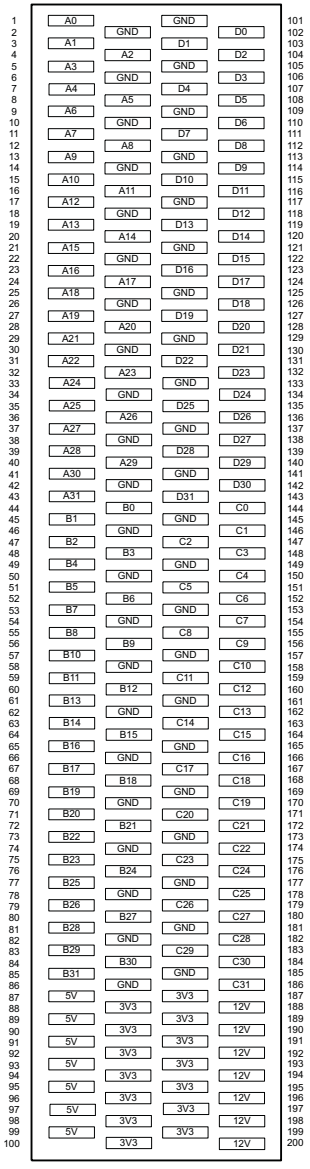


Figure A-1 HDRA plug pin numbering

For the Integrator/AP, the signals present on the pins labeled A[31:0], B[31:0], and C[31:0] are described in Table A-1.

Table A-1 Bus bit assignment, Integrator/AP

Pin label	AHB signal name	ASB signal name	Description
A[31:0]	**HADDR[31:0]**	**BA[31:0]**	System address bus
B[31:0]	Not used	Not used	-
C[31:16]	Not used	Not used	-
C15	**HMASTLOCK**	**BLOK**	Locked transaction
C14	**HRESP1**	**BLAST**	Slave response
C13	**HRESP0**	**BERROR**	Slave response
C12	**HREADY**	**BWAIT**	Slave wait response
C11	**HWRITE**	**BWRITE**	Write transaction
C10	**HPROT2**	Not used	Transaction protection type
C[9:0]	**HPROT[1:0]**	**BPROT[1:0]**	Transaction protection type
C[7:5]	**HBURST[2:0]**	Not used	Transaction burst size
C4	**HPROT[3]**	Not used	Transaction protection type
C[3:2]	**HSIZE[1:0]**	**BSIZE[1:0]**	Transaction width
C[1:0]	**HTRAN[1:0]**	**BTRAN[1:0]**	Transaction type
D[31:0]	**HDATA[31:0]**	Not used	System data bus

For the Integrator/CP, the signals present on the pins labeled A[31:0], B[31:0], and C[31:0] are described in Table A-2.

Table A-2 HDRA signals descriptions, Integrator/CP

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

Table A-2 HDRA signals descriptions, Integrator/CP (continued)

Pin label	AHB signal name	CP specific[a]	Description
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.

A.2 HDRB

The HDRB plug and socket have slightly different pinouts, as described in:

- *Through-board signal connections* on page A-6
- *HDRB socket pinout* on page A-6
- *HDRB plug pinout* on page A-8
- *HDRB signal descriptions* on page A-9.

A.2.1 Through-board signal connections

The signals on the pins labeled E[31:0] are cross-connected between the plug and socket so that the signals are rotated through the stack in groups of four. For example, the first block of four are connected as shown in Table A-3.

Table A-3 Example of signal cross-connections

Plug		Socket
E0	connects to	E1
E1	connects to	E2
E2	connects to	E3
E3	connects to	E0

For details about the signal rotation scheme, see *Module ID selection for AP* on page 5-3 and *Module ID selection and interrupt routing* on page 6-5.

The signals on the pins labeled F[31:0] are connected so that pins on the socket are connected to the corresponding pins on the plug.

The signals on G[16:8] and G[5:0] are connected so that pins on the socket are connected to the corresponding pins on the plug.

Pins G[7:6] carry the JTAG **TDI** and **TDO** signals. The signal **TDO** is routed through devices on each board as it passes up through the stack (see *JTAG signals* on page 3-21).

A.2.2 HDRB socket pinout

Figure A-2 on page A-7 shows the pin numbers of the socket HDRB on the underside of the core module, viewed from above the core module.

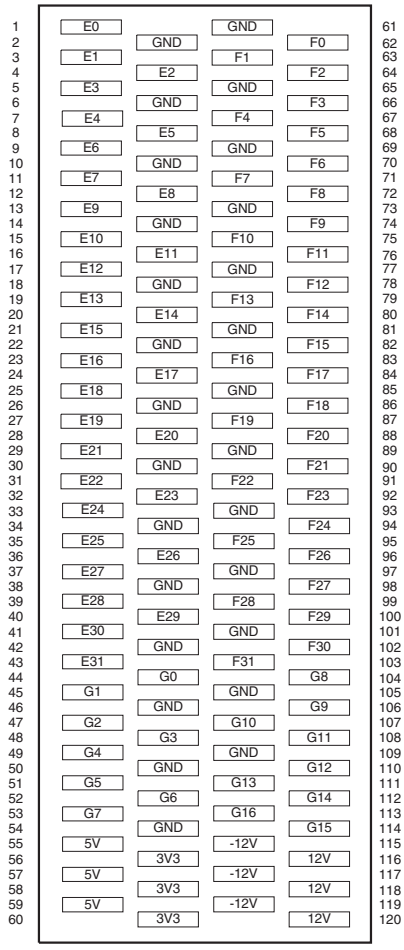


Figure A-2 HDRB socket pin numbering

A.2.3 HDRB plug pinout

Figure A-3 shows the pin numbers of the HDRB plug on the top of the core module.

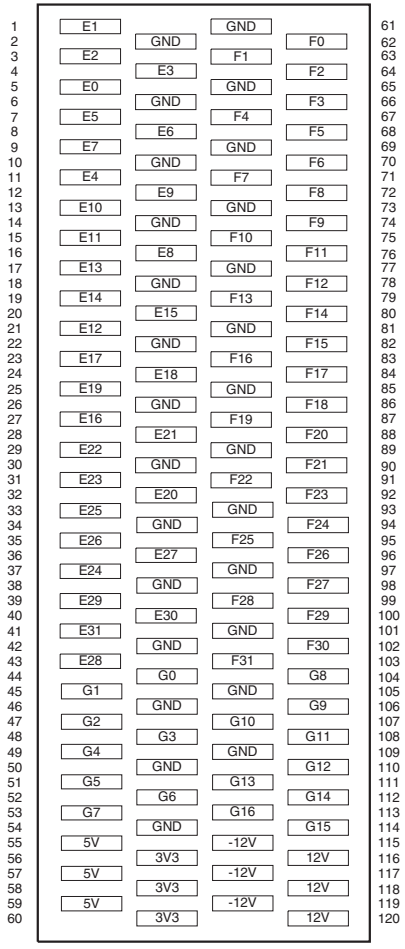


Figure A-3 HDRB plug pin numbering

A.2.4 HDRB signal descriptions

Table A-4 describes the signals on the pins labeled E[31:0], F[31:0], and G[16:0] for Integrator/AP and AMBA AHB system bus.

Table A-4 HDRB signal description, Integrator/AP AHB

Pin label	Name	Description
E[31:28]	**SYSCLK[3:0]**	System clock to each core module/expansion card.
E[27:24]	**nPPRES[3:0]**	Processor present.
E[23:20]	**nIRQ[3:0]**	Interrupt request to processors 3, 2, 1, and 0 respectively.
E[19:16]	**nFIQ[3:0]**	Fast interrupt requests to processors 3, 2, 1, and 0 respectively.
E[15:12]	**ID[3:0]**	Core module stack position indicator.
E[11:8]	**HLOCK[3:0]**	System bus lock from processor 3, 2, 1, and 0 respectively.
E[7:4]	**HGRANT[3:0]**	System bus grant to processor 3, 2, 1, and 0 respectively.
E[3:0]	**HBUSREQ[3:0]**	System bus request from processors 3, 2, 1, and 0 respectively.
F[31:0]	-	Not connected.
G16	**nRTCKEN**	**RTCK** AND gate enable.
G[15:14]	**CFGSEL[1:0]**	FPGA configuration select.
G13	**nCFGEN**	Sets motherboard into configuration mode.
G12	**nSRST**	Multi-ICE reset (open collector).
G11	**FPGADONE**	Indicates when FPGA configuration is complete (open collector).
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.

Table A-4 HDRB signal description, Integrator/AP AHB (continued)

Pin label	Name	Description
G4	**TCK**	JTAG test clock.
G[3:1]	**MASTER[2:0]**	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.
G0	**nMBDET**	Motherboard detect pin.

Table A-5 describes the signals on the pins labeled E[31:0], F[31:0], and G[16:0] for Integrator/AP and AMBA ASB system bus.

Table A-5 HDRB signal description, Integrator/AP ASB

Pin label	Name	Description
E[31:28]	**SYSCLK[3:0]**	System clock to the core module.
E[27:24]	**nPPRES[3:0]**	Processor present.
E[23:20]	**nIRQ[3:0]**	Interrupt request to processors 3, 2, 1, and 0 respectively.
E[19:16]	**nFIQ[3:0]**	Fast interrupt requests to processors 3, 2, 1, and 0 respectively.
E[15:12]	**ID[3:0]**	Core module stack position indicator.
E[11:8]	Reserved	-
E[7:4]	**SGRNT[3:0]**	System bus grant to processors 3, 2, 1, and 0 respectively.
E[3:0]	**SREQ[3:0]**	System bus request from processors 3, 2, 1, and 0 respectively.
F[31:0]	-	Not connected.
G16	**nRTCKEN**	**RTCK** AND gate enable.
G[15:14]	**CFGSEL[1:0]**	FPGA configuration select.
G13	**nCFGEN**	Sets motherboard into configuration mode.
G12	**nSRST**	Multi-ICE reset (open collector).
G11	**FPGADONE**	Indicates when FPGA configuration is complete (open collector).
G10	**RTCK**	Returned JTAG test clock.
G9	**nSYSRST**	Buffered system reset.

Table A-5 HDRB signal description, Integrator/AP ASB (continued)

Pin label	Name	Description
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]	**MASTER[2:0]**	Master ID. Binary encoding of the master currently performing a transfer on the bus. Corresponds to the module ID and to the **SREQ** and **SGRANT** line numbers.
G0	**nMBDET**	Motherboard detect pin.

Table A-6 describes the signals on the pins labeled E[31:0], F[31:0], and G[16:0] for Integrator/CP and AHB-Lite bus.

Table A-6 HDRB signal description, Integrator/CP

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.

Table A-6 HDRB signal description, Integrator/CP (continued)

Pin label	Signal Name	CP specific	Description
F[31:16] F[9:8]	-	Yes	A variety of interface connections
F[15:10]	-	Yes	MMC interface signals
F[13] 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.
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.

A.3 Trace connector pinout

Table A-7 shows the pinout of the Trace connector.

Table A-7 Trace connector pinout

Channel	Pin	Pin	Channel
No connect	1	2	No connect
No connect	3	4	No connect
GND	5	6	**TRACECLK**
DBGRQ	7	8	**DBGACK**
nSRST	9	10	**EXTTRIG**
TDO	11	12	**VDD** (3.3V)
RTCK	13	14	**VDD** (3.3V)
TCK	15	16	**TRACEPKT7**
TMS	17	18	**TRACEPKT6**
TDI	19	20	**TRACEPKT5**
nTRST	21	22	**TRACEPKT4**
TRACEPKT15	23	24	**TRACEPKT3**
TRACEPKT14	25	26	**TRACEPKT2**
TRACEPKT13	27	28	**TRACEPKT1**
TRACEPKT12	29	30	**TRACEPKT0**
TRACEPKT11	31	32	**TRACESYNC**
TRACEPKT10	33	34	**PIPESTAT2**
TRACEPKT9	35	36	**PIPESTAT1**
TRACEPKT8	37	38	**PIPESTAT0**

A.4 Logic analyzer connectors

A logic analyzer connects to the local memory bus on the core module using high-density AMP Mictor connectors. There are four logic analyzer connectors labelled:

- *Logic analyzer connector BA* on page A-15
- *Logic analyzer connector CONTROL* on page A-16
- *Logic analyzer connector BD* on page A-17
- *Logic analyzer connector SPARE* on page A-18.

The signals names include a prefix that identifies the source device.

The logic analyzer connection is a high-density AMP Mictor connector. The connectors carries 32 signals and 2 clocks or qualifiers. Figure A-4 shows the connector and identification of pin 1.

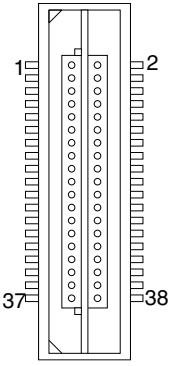


Figure A-4 AMP Mictor connector

Note

Agilent (formerly HP) and Tektronix label these connectors differently, but the assignments of signals to physical pins is appropriate for both systems and pin 1 is always in the same place. The schematic is labelled according to the Agilent pin assignment.

A.4.1 Logic analyzer connector BA

Table A-8 shows the pinout of the connector BA.

Table A-8 Connector BA pinout

Channel	Pin	Pin	Channel
No connect	1	2	No connect
GND	3	4	No connect
ARM_BTRAN1	5	6	**LA_BCLK1**
ARM_BA31	7	8	**ARM_BA15**
ARM_BA30	9	10	**ARM_BA14**
ARM_BA29	11	12	**ARM_BA13**
ARM_BA28	13	14	**ARM_BA12**
ARM_BA27	15	16	**ARM_BA11**
ARM_BA26	17	18	**ARM_BA10**
ARM_BA25	19	20	**ARM_BA9**
ARM_BA24	21	22	**ARM_BA8**
ARM_BA23	23	24	**ARM_BA7**
ARM_BA22	25	26	**ARM_BA6**
ARM_BA21	27	28	**ARM_BA5**
ARM_BA20	29	30	**ARM_BA4**
ARM_BA19	31	32	**ARM_BA3**
ARM_BA18	33	34	**ARM_BA2**
ARM_BA17	35	36	**ARM_BA1**
ARM_BA16	37	38	**ARM_BA0**

A.4.2 Logic analyzer connector CONTROL

Table A-9 shows the pinout of the CONTROL connector.

Table A-9 Connector CONTROL

Channel	Pin	Pin	Channel
No connect	1	2	No connect
GND	3	4	No connect
nSYSRST	5	6	No connect
SD_nWE	7	8	**SD_nCAS**
SD_CKE1	9	10	**SD_nRAS**
SD_CKE0	11	12	**SRAM_nCKE**
ARM_BnRES	13	14	**SRAM_RnW**
ARM_nIRQ	15	16	**SRAM_nAD**
ARM_nFIQ	17	18	**SRAM_RAMAT0**
SD_nCS0	19	20	**SRAM_nCE**
ARM_BLAST	21	22	**SRAM_nOE**
ARM_BERROR	23	24	**BRIDGE_WRWAIT**
ARM_BWAIT	25	26	**BRIDGE_RDWAIT**
ARM_BLOCK	27	28	**SDRAM_WRWAIT**
ARM_BSIZE1	29	30	**SDRAM_RDWAIT**
ARM_BSIZE0	31	32	**ARM_BURST1**
ARM_BPROT1	33	34	**ARM_BURST0**
ARM_BPROT0	35	36	**ARM_BTRAN1**
ARM_BWRITE	37	38	**ARM_BTRAN0**

A.4.3 Logic analyzer connector BD

Table A-10 shows the pinout of the BD connector.

Table A-10 Connector BD pinout

Channel	Pin	Pin	Channel
No connect	1	2	No connect
GND	3	4	No connect
ARM_BWAIT	5	6	**LA_BCLK2**
ARM_BD31	7	8	**ARM_BD15**
ARM_BD30	9	10	**ARM_BD14**
ARM_BD29	11	12	**ARM_BD13**
ARM_BD28	13	14	**ARM_BD12**
ARM_BD27	15	16	**ARM_BD11**
ARM_BD26	17	18	**ARM_BD10**
ARM_BD25	19	20	**ARM_BD9**
ARM_BD24	21	22	**ARM_BD8**
ARM_BD23	23	24	**ARM_BD7**
ARM_BD22	25	26	**ARM_BD6**
ARM_BD21	27	28	**ARM_BD5**
ARM_BD20	29	30	**ARM_BD4**
ARM_BD19	31	32	**ARM_BD3**
ARM_BD18	33	34	**ARM_BD2**
ARM_BD17	35	36	**ARM_BD1**
ARM_BD16	37	38	**ARM_BD0**

A.4.4 Logic analyzer connector SPARE

Table A-11 shows the pinout of the spare connector.

Table A-11 SPARE (J10)

Channel	Pin	Pin	Channel
No connect	1	2	No connect
GND	3	4	No connect
PLD_SPARE22	5	6	No connect
PLD_SPARE21	7	8	**PLD_SPARE5**
PLD_SPARE20	9	10	**PLD_SPARE4**
PLD_SPARE19	11	12	**PLD_SPARE3**
PLD_SPARE18	13	14	**PLD_SPARE2**
PLD_SPARE17	15	16	**PLD_SPARE1**
PLD_SPARE16	17	18	**PLD_SPARE0**
PLD_SPARE15	19	20	**ARM_EDBGRQ**
PLD_SPARE14	21	22	**ARM_DBACK**
PLD_SPARE13	23	24	**ARM_DBGEN**
PLD_SPARE12	25	26	**ARM_DEWPT**
PLD_SPARE11	27	28	**ARM_IEBKPT**
PLD_SPARE10	29	30	**ARM_TRACK**
PLD_SPARE9	31	32	**ARM_CHSEX1**
PLD_SPARE8	33	34	**ARM_CHSEX0**
PLD_SPARE7	35	36	**ARM_CHSE1**
PLD_SPARE6	37	38	**ARM_CHSE0**

Appendix B
Specifications

This appendix contains the specifications for the ARM Integrator/CM920T, CM920T-ETM, and CM940T core modules. It contains the following sections:

B.1 Electrical specification

This section provides details of the voltage and current characteristics for the core module.

B.1.1 Bus interface characteristics

Table B-1 shows the core module electrical characteristics for the system bus interface. The core module uses 3.3V and 5V sources. The 12V inputs are supplied by the Integrator/AP motherboard but not used by the core module. The Integrator/CP baseboard does not supply 12V.

Table B-1 Core module electrical characteristics

Symbol	Description	Min	Max	Unit
3V3	Supply voltage (interface signals)	3.1	3.5	V
5V	Supply voltage	4.75	5.25	V
V_{IH}	High-level input voltage	2.0	3.6	V
V_{IL}	Low-level input voltage	0	0.8	V
V_{OH}	High-level output voltage	2.4	-	V
V_{OL}	Low-level output voltage	-	0.4	V
C_{IN}	Input capacitance	-	20	pF

B.1.2 Current requirements

Table B-2 shows the maximum current requirements measured at room temperature and nominal voltage. These measurements include the current drawn by Multi-ICE (approximately 160mA at 3.3V).

Table B-2 Current requirements

System	At 3.3V	At 5V
Standalone core module	1A	100mA
Motherboard and one core module	1.5A	500mA

An Integrator/AP with additional core or logic modules draws more current, and future core modules might require more current. For these reasons, provision is made to power the system with an ATX-type power supply.

B.2 Timing specification

This section is a reference for designers adding modules on to an Integrator system. The timing information presented here is representative only. Specific modules and FPGA revisions will deviate from these numbers, but they provide some guidance when constraining FPGA designs.

The following sections detail the timing parameters for a typical ASB and AHB modules and motherboards.

B.2.1 Core module timing and the AMBA Specification

The parameters listed are those specified in the *AMBA Specification* with the following important differences:

- only output valid and input setup times are quoted
- the required input hold time (T_{ih}) is always less than or equal to 0ns
- the output hold time (T_{oh}) is always greater than 2ns.

Each version and revision of the FPGA has subtly different timing. The typical figures are those you can expect under nominal conditions and must be used as a guideline when designing your own motherboards and modules. The typical figures have been rounded to simplify timing analysis and constraints.

B.2.2 AHB system bus timing parameters

Table B-3 shows the clock and reset timing parameters.

Table B-3 Clock and reset parameters

Parameter	Description	Time (ns)	Notes
Tclk	**HCLK** minimum clock period	30	Representative of worst case maximum frequency
Tisrst	**HRESETn** deasserted setup time before HCLK	15	Applies to modules only
Tovrst	**HRESETn** deasserted valid time before **HCLK**	15	Applies only to the module or motherboard implementing the reset source

Table B-4 shows the AHB slave input parameters.

Table B-4 AHB slave input parameters

Parameter	Description	Time (ns)	Notes
Tissel	**HSELx** setup time before **HCLK**	N/a	**HSELx** are internally generated, not visible at the pins
Tistr	Transfer type setup time before **HCLK**	5	–
Tisa	**HADDR[31:0]** setup time before **HCLK**	10	–
Tisctl	**HWRITE**, **HSIZE[2:0]** and **HBURST[2:0]** control signal setup time before **HCLK**	5	–
Tiswd	Write data setup time before **HCLK**	5	–
Tisrdy	Ready setup time before **HCLK**	5	–
Tismst	Master number setup time before **HCLK** (SPLIT-capable only)	N/a	Applies to modules with split capable slaves only
Tismlck	Master locked setup time before **HCLK** (SPLIT-capable only)	N/a	Applies to modules with split capable slaves only

Table B-5 shows the AHB slave output parameters.

Table B-5 AHB slave output parameters

Parameter	Description	Time (ns)	Notes
Tovrsp	Response valid time after **HCLK**	15	–
Tovrd	Data valid time after **HCLK**	15	–
Tovrdy	Ready valid time after **HCLK**	15	–
Tovsplt	Split valid time after **HCLK** (SPLIT-capable only)	N/a	Applies to modules with split capable slaves only

Table B-6 shows the bus master input timing parameters.

Table B-6 Bus master input timing parameters

Parameter	Description	Time (ns)	Notes
Tisgnt	**HGRANTx** setup time before **HCLK**	5	Modules implementing masters only
Tisrdy	Ready setup time before **HCLK**	5	–
Tisrsp	Response setup time before **HCLK**	5	–
Tisrd	Read data setup time before **HCLK**	5	–

Table B-7 shows bus master output timing parameters.

Table B-7 Bus master output timing parameters

Parameter	Description	Time (ns)	Notes
Tovtr	Transfer type valid time after **HCLK**	15	–
Tova	Address valid time after **HCLK**	15	–
Tovctl	Control signal valid time after **HCLK**	15	–
Tovwd	Write data valid time after **HCLK**	15	–
Tovreq	Request valid time after **HCLK**	15	Modules implementing masters only
Tovlck	Lock valid time after **HCLK**	15	Modules implementing masters only

Table B-8 shows the AHB arbiter input parameters. Applies only to the module or motherboard implementing the arbiter.

Table B-8 AHB arbiter input parameters

Parameter	Description	Time (ns)
Tisrdy	Ready setup time before **HCLK**	5
Tisrsp	Response setup time before **HCLK**	5
Tisreq	Request setup time before **HCLK**	10
Tislck	Lock setup time before **HCLK**	10

Table B-8 AHB arbiter input parameters

Parameter	Description	Time (ns)
Tissplt	Split setup time before **HCLK**	10
Tistr	Transfer type setup time before **HCLK**	5
Tisctl	Control signal setup time before **HCLK**	5

Table B-9 shows the AHB arbiter output parameters. Applies only to the module or motherboard implementing the arbiter.

Table B-9 AHB arbiter output parameters

Parameter	Description	Time (ns)
Tovgnt	Grant valid time after **HCLK**	15
Tovmst	Master number valid time after **HCLK**	15
Tovmlck	Master locked valid time after **HCLK**	15

B.2.3 ASB system bus timing parameters

Table B-10 shows the clock and reset parameters.

Table B-10 Clock and reset parameters

Parameter	Description	Time (ns)	Notes
Tclk	**BCLK** minimum clock period	40	Representative of worst case maximum frequency
Tclkl	**BCLK** LOW time	20	–
Tclkh	**BCLK** HIGH time	20	–
Tisnres	**BnRES** deasserted setup to rising **BCLK**	15	Applies to modules only
Tovnres	**BnRES** deasserted valid after rising **BCLK**	15	Applies only to the module or motherboard implementing the reset source

Table B-11 shows the ASB slave input parameters. Applies only to the module or motherboard implementing the arbiter.

Table B-11 ASB slave input parameters

Parameter	Description	Time (ns)	Notes
Tisdsel	**DSEL** setup to falling **BCLK**	n/a	**DSEL** is internally generated, not visible at the pins
Tisa	**BA[31:0]** setup to falling **BCLK**	10	Path through decoder is up to 30ns
Tisctl	**BWRITE** and **BSIZE[1:0]** setup to falling **BCLK**	10	–
Tisdw	For write transfers, **BD[31:0]** setup to falling **BCLK**	10	–

Table B-12 shows the ASB slave output parameters.

Table B-12 ASB slave output parameters

Parameter	Description	Time (ns)
Tovresp	**BWAIT**, **BERROR** and **BLAST** valid after falling **BCLK**	10
Tovdr	For read transfers, **BD[31:0]** valid after rising **BCLK**	30

Table B-13 shows the bus master input timing parameters

Table B-13 Bus master input parameters

Parameter	Description	Time (ns)	Notes
Tisresp	**BWAIT**, **BERROR** and **BLAST** setup to rising **BCLK**	15	–
Tisdr	For read transfers, **BD[31:0]** setup to falling **BCLK**	10	–
Tisagnt	**AGNT** setup to rising **BCLK**	10	Modules implementing masters only

Table B-14 shows the bus master output timing parameters.

Table B-14 Bus master output parameters

Parameter	Description	Time (ns)
Tovtr	**BTRAN** valid after rising **BCLK**	10
Tova	**BA[31:0]** valid after rising **BCLK**, all transfer types	10
Tovctl	**BWRITE**, **BSIZE[1:0]**, and **BPROT[1:0]** valid after rising BCLK, all transfer types	10
Tovdw	**BD[31:0]** valid after rising **BCLK**, all transfer types	30
Tovlok	**BLOK** valid after rising **BCLK**	10
Tovareq	**AREQ** valid after rising **BCLK**	10

Table B-15 shows the ASB decoder input parameters.

Table B-15 ASB decoder input parameters

Parameter	Description	Time (ns)
Tistr	**BTRAN** setup to falling **BCLK**	10
Tisresp	**BWAIT**, **BERROR** and **BLAST** setup to rising **BCLK**	15

Table B-16 ASB decoder output parameters.

Table B-16 ASB decoder output parameters

Parameter	Description	Time (ns)	Notes
Tovresp	**BWAIT**, **BERROR** and **BLAST** valid after falling **BCLK**	10	–
Tovdsel	**DSEL** valid after rising **BCLK**	N/a	**DSEL** is internally generated, not visible at the pins

Table B-17 shows the ASB arbiter input parameters.Applies only to the module or motherboard implementing the arbiter.

Table B-17 ASB arbiter input parameters

Parameter	Description	Time (ns)
Tisareq	**AREQ** setup to falling **BCLK**	10
Tisresp	**BWAIT** setup to rising **BCLK**	10

Table B-18 ASB arbiter output parameters. applies only to the module or motherboard implementing the arbiter.

Table B-18 ASB arbiter output parameters

Parameter	Description	Time (ns)
Tovagnt	**AGNT** valid after falling **BCLK**	10

Table B-19 shows the ASB arbiter combinatorial parameters.

Table B-19 ASB arbiter combinatorial parameters

Parameter	Description	Time (ns)	Notes
Tlokagnt	Delay from valid **BLOK** to valid **AGNT**	N/a	Not applicable, arbiter samples all inputs

B.2.4 Notes on FPGA timing analysis

The system bus on all Integrator boards is routed between FPGAs. These FPGAs are routed with timing constraints similar to those shown in the tables in *AHB system bus timing parameters* on page B-3 and *ASB system bus timing parameters* on page B-6. The exact performance of a system depends on the timing parameters of the motherboard and all modules in the system. Some allowance also must be made for clock skew, routing delays, and number of modules (that is, loading).

Not all FPGAs meet the ideal timing parameters, because of the complexity of the design or routing congestion within the device. For this reason, the PLL clock generators on Integrator default to a safe low value that all modules can achieve.

A detailed timing analysis involves calculating the input/output delays between modules for all timing parameters. In general, the simplest approach to determine the maximum operating frequency is to increase the frequency of the clock generators until the system becomes unstable. The maximum stable operating frequency for your board combination is likely to be a few MHz lower.

ARM processors and core module FPGAs do not dissipate large amounts of heat. However, to be sure of stable operation, run the test program for a few minutes. Experiments show that the FPGAs, when operating at maximum system bus frequency, slowly increase in temperature, but that the maximum is typically less than 35°C.

B.3 Mechanical details

The core module is designed to be stackable on several different motherboards. Its size enables it to be mounted onto a CompactPCI motherboard while enabling the motherboard to be installed in a card cage.

Figure B-1 shows the mechanical outline of the core module.

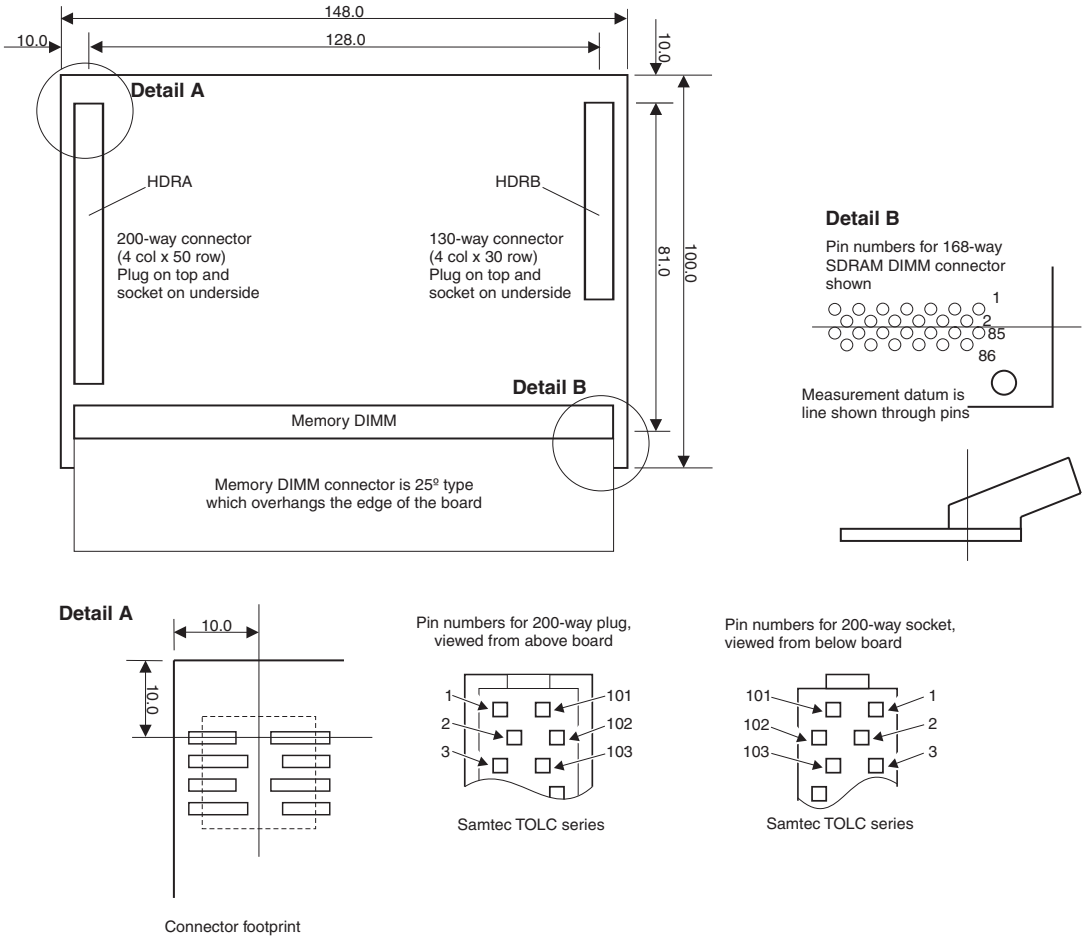


Figure B-1 Board outline

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.

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

