Integrator/CM940T, CM920T, CM740T, and CM720T

User Guide

ARM

ARM DUI 0157A

Integrator/CM940T, CM920T, CM740T, and CM720T
User Guide

Release Information

Description	Issue	Change
26 April 2001	A	New document based on earlier core module manuals for ARM940T, ARM920T, ARM740T and ARM720T types. Incorporates the following changes: FCC statement revised. Section 3.3 includes a description on how the core module is configured for an AHB or ASB system bus. Section 3.7 added to better describe module ID selection. Section 3.9.1 (was 3.8.1) now contains a description of the JTAG clock signal path. Section 4.1 memory map now contains an alias of the on-board SSRAM. Appendix A now lists signals for AHB and ASB system buses. Appendix B now contains more detailed electrical and timing specifications.

Proprietary Notice

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

This equipment has been tested according to ISE/IEC Guide 22 and EN 45014. It conforms to the following product EMC specifications:

The product herewith complies with the requirements of EMC Directive 89/336/EEC as amended.

Confidentiality Status

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

Product Status

The information in this documents is Final (information on a developed product).

Web Address

`http://www.arm.com`

Contents
Integrator/CM940T, CM920T, CM740T, and CM720T User Guide

Preface

This preface introduces the ARM Integrator/CM940T, CM920T, CM740T, and CM720T core modules and their reference documentation. It contains the following sections:

About this document

This document describes how to set up and use your ARM Integrator core module.

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 module.

Chapter 2 ***Getting Started***

Read this chapter for a description of how to set up and start using the core module.

Chapter 3 ***Hardware Description***

Read this chapter for a description of the hardware architecture of the core module, including clocks, resets, and debug.

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

Read this chapter for a description of the core module memory map and registers.

Appendix A ***Signal Descriptions***

Refer to this appendix for a description of the signals on the HDRA and HDRB 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:

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

`type`writer Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.

`typewriter 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.

`typewriter 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 might provide additional information.

ARM publications

The following publications provide information about ARM cores and related ARM Integrator products:

- *ARM740T Technical Reference Manual* (ARM DDI 0008)
- *ARM720T Technical Reference Manual* (ARM DDI 0087)
- *ARM940T Technical Reference Manual* (ARM DDI 0087)
- *ARM920T Technical Reference Manual* (ARM DDI 0150)
- *ARM Integrator/AP User Guide* (ARM DUI 0144)
- *ARM Integrator/LM-XCV600E and LM-EP20K600E User Guide* (ARM DUI 0146)
- *ARM Integrator/AM User Guide* (ARM DUI 0133)

The following publications provide information about related ARM development tools:

- *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)
- *ARM Software Development Toolkit User Guide* (ARM DUI 0040)
- *ARM Software Development Toolkit Reference Guide* (ARM DUI 0041)
- *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).

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 on the ARM Integrator core module and on the documentation.

Feedback on this document

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

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

General suggestions for additions and improvements are also welcome.

Feedback on the ARM Integrator core module

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

- the product name
- an explanation of your comments.

Chapter 1
Introduction

This chapter introduces the ARM Integrator core module. It contains the following sections:

1.1 About the core module

The Integrator core module provides you with the basis of a flexible development system that can be used in several different ways. With power and a simple connection to a Multi-ICE debugger, it provides a basic development system. By mounting the core module onto a motherboard, you can build a realistic emulation of the system being developed. Through-board connectors allow up to four core modules to be stacked on one motherboard.

The core module can be used in the following ways:

- as a standalone development system
- mounted onto an ARM Integrator motherboard
- integrated into your own development or ASIC prototyping system.

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

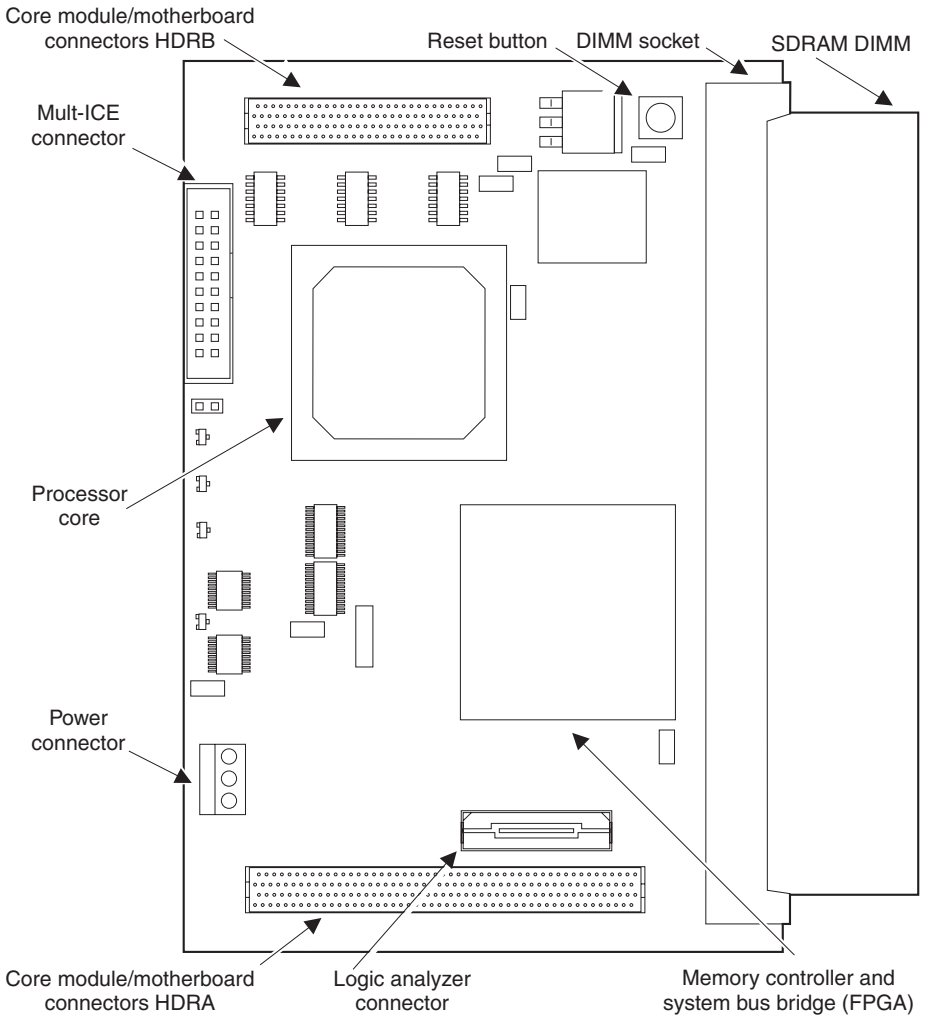


Figure 1-1 Core module layout

1.2 Core module architecture

The major components on the core module are as follows:

- ARM core
- core module FPGA that implements:
  - SDRAM controller
  - system bus bridge
  - reset controller
  - interrupt controller
  - status, configuration, and interrupt registers.
- volatile memory comprising:
  - up to 256MB of SDRAM (optional) connected to DIMM socket
  - 256KB SSRAM.
- SSRAM controller
- clock generator
- system bus connectors
- Multi-ICE debug connector and logic analyzer connector.

1.2.1 System architecture

Figure 1-2 illustrates the architecture of the core module.

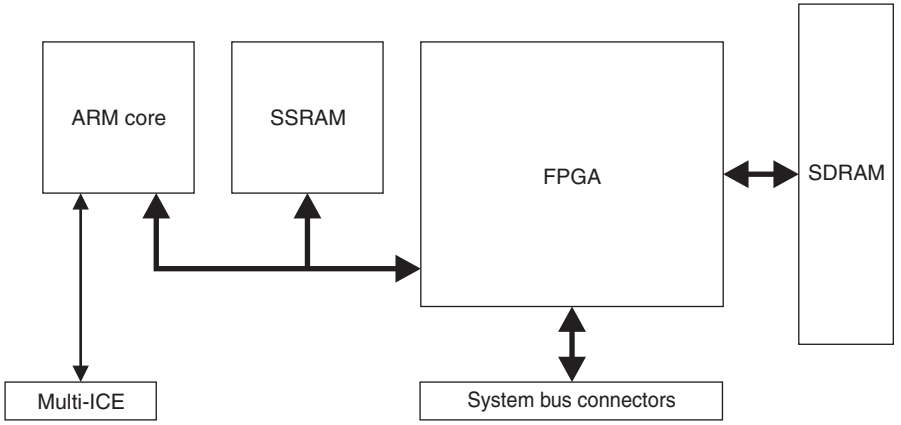


Figure 1-2 ARM Integrator/CM740T block diagram

1.2.2 ARM core

The core module is fitted with one of the following ARM core types:

- ARM940T™
- ARM920T™
- ARM740T™
- ARM720T™.

The Integrator system architecture and memory map remain the same whichever core is fitted. Modules fitted with these cores support:

- asynchronous and FastBus clocking
- little-endian and big-endian addressing when operated standalone.

They do not support:

- synchronous clocking
- big-endian addressing when mounted on an Integrator/AP.

Depending on core type, the clocking and addressing modes are selected as follows:

Clocking mode

ARM740T and ARM720T core types. Select the clocking mode by programming the FASTBUS bit in the CM_CTRL register on the core module (see *Core module control register* on page 4-12).

ARM940T and ARM920T. The clocking mode is selected by writing to coprocessor 15 register 1.

Addressing mode

ARM740T, ARM720T, ARM940T, and ARM920T core types. Select little-endian addressing mode by programming the B bit in coprocessor 15 register 1. For details about the addressing modes, refer to the *Technical Reference Manual* for the appropriate core.

1.2.3 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*.

SDRAM controller

The SDRAM controller is implemented within the FPGA. This provides support for *Dual In-line Memory Modules* (DIMMs) with a capacity of between 16 and 256MB. See *SDRAM controller* on page 3-7.

Reset controller

The reset controller initializes the core and allows the core module to be reset from five sources:

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

For information about the reset controller, see *Reset controller* on page 3-9.

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 allows the processor to access resources on the motherboard and on other modules. It also allows other masters to access the core module SDRAM (see *System bus bridge* on page 3-12). The system bus bridge can be configured at power up to operate with an AHB or ASB system bus (see *Core module FPGA* on page 3-5).

The memory bus and system bus operate asynchronously, enabling each to be run at the speed of its slowest device without compromising the performance of other buses in the system.

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:

- type of processor and whether it has a cache, MMU, or protection unit
- the position of the core module in a multi-module stack
- SDRAM size, address configuration, and CAS latency setup
- core module oscillator setup
- 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*.

1.2.4 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 ensure high 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 and by other system bus masters. The SSRAM can only be accessed by the local processor.

1.2.5 Clock generators

The core module uses four clock signals:

REFCLK A fixed frequency 24MHz signal supplied to the clock generators and to the FPGA.

CORECLK A programmable frequency clock input to the ARM core.

LCLK A programmable frequency clock for the local memory bus.

nLCLK An inverted version of **LCLK**.

The programmable clocks are supplied by two clock generator chips. The output frequencies of these are set using the oscillator control register within the FPGA. The reference clock is supplied to the two clock generators and to the FPGA enabling it to be used to generate real-time delays (see *Clock generators* on page 3-20).

1.2.6 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 3-24.

Note

JTAG test equipment supplied by other vendors can also be used.

1.2.7 Diagnostic connector

The logic analyzer connector enables you to gain access to a number of signals. These differ for each core module type described by this manual and you should refer to the schematics for pinout details.

1.3 Links and indicators

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

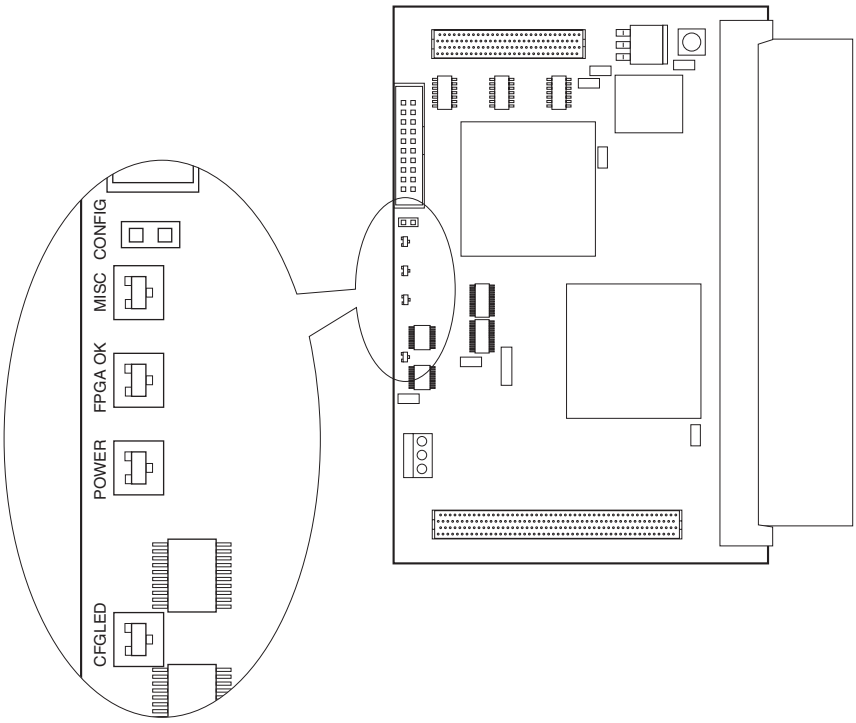


Figure 1-3 Links and indicators

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 registers* on page 4-8.
FPGA OK	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 two ground points, some signal test points, and a logic analyzer connector to an aid to debug. These are illustrated in Figure 1-4.

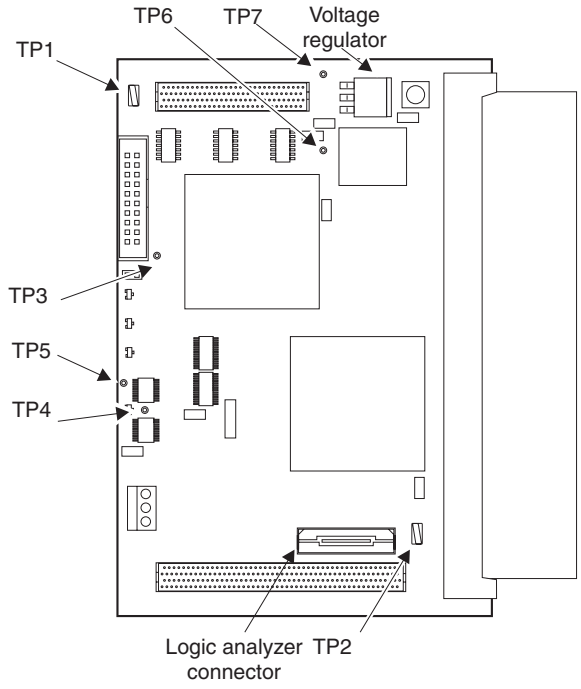


Figure 1-4 Test points

The functions of the test points are summarized in Table 1-2.

Table 1-2 Test point functions

Test point	Name	Function
TP1	GND	Ground
TP2	GND	Ground
TP3	**FCLKOUT**	Clock output form the core (ARM940T and ARM920T only)
TP4	**LBCLK**	Local memory bus clock

Table 1-2 Test point functions

Test point	Name	Function
TP5	**CORECLK**	Clock supplied to the microprocessor core
TP6	**VDD7x0T** or **VDD9x0T**	Output from voltage regulator
TP7	ADJ	ADJ pin of voltage regulator

The logic analyzer connector has a different pinout for each of the core module types. Refer to the schematics for the correct pinout for your core module.

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 JTAG interface.

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 which 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.

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

Fit the following type of SDRAM module:

- PC66, PC10, 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

The core module can be operated without SDRAM because it has 256KB of SSRAM permanently fitted. However, in order to operate the core module with an ARM debugger, you must change the internal variable `$top_of_memory` from the default setting of `0x00080000` (= 512KB) to `0x00040000` (= 256KB) before running programs that are linked with the standard libraries.

For further information about ARM debugger internal variables, refer to the *Software Development Toolkit Reference Guide*.

2.1.3 Supplying power

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

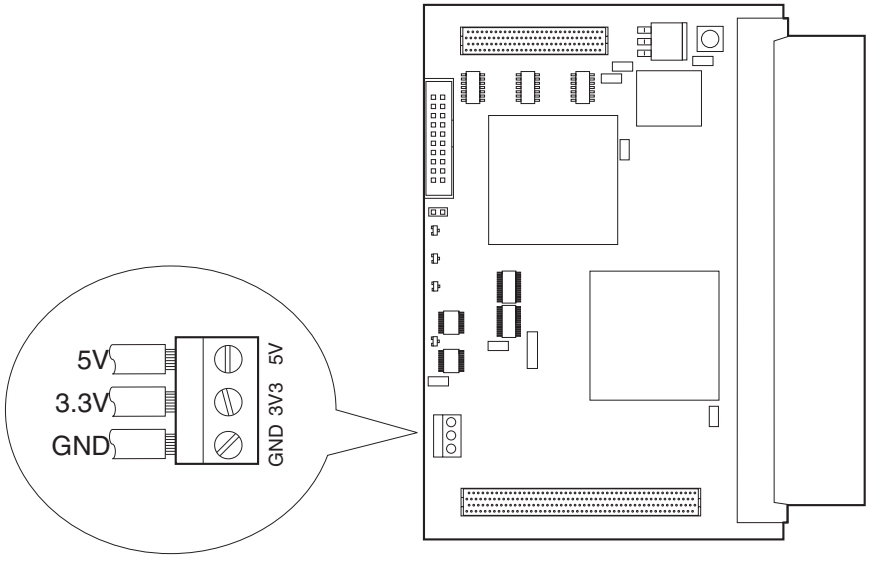


Figure 2-1 Power connector

Note

This power connection is not required when the core module is fitted to a motherboard.

2.1.4 Connecting Multi-ICE

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.

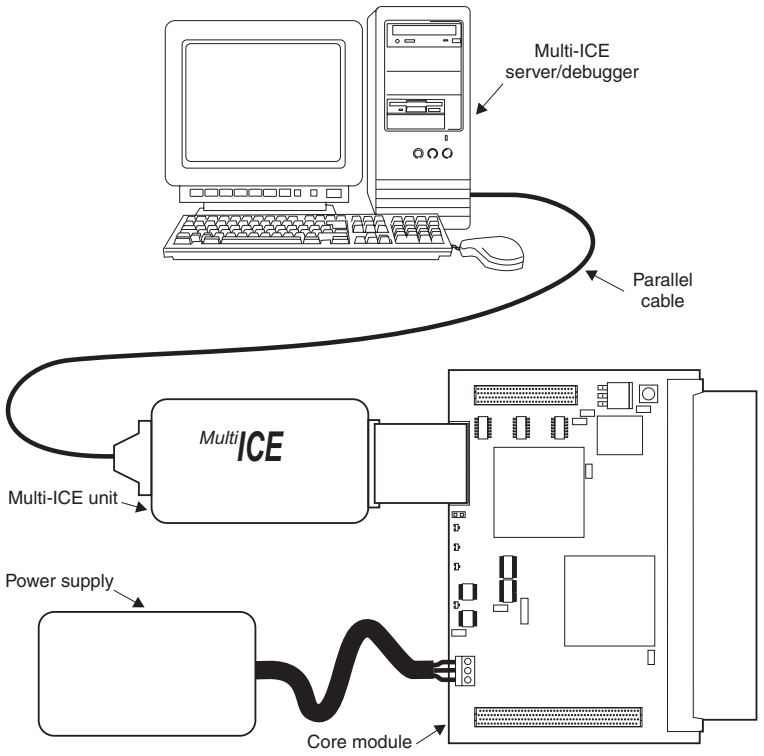


Figure 2-2 Multi-ICE connection to a core module

Caution

The core module does not provide nonvolatile memory meaning that programs are lost when the power is removed.

Multi-ICE can also be used when a core module is attached to a motherboard. If more than one core module is attached, then connect the Multi-ICE 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.

2.2 Attaching the core module to a 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 allow core modules to be mounted on top of one another. A maximum of four core modules can be stacked on a motherboard.

Figure 2-3 illustrates an example development system with four core modules attached to an ARM Integrator/AP motherboard.

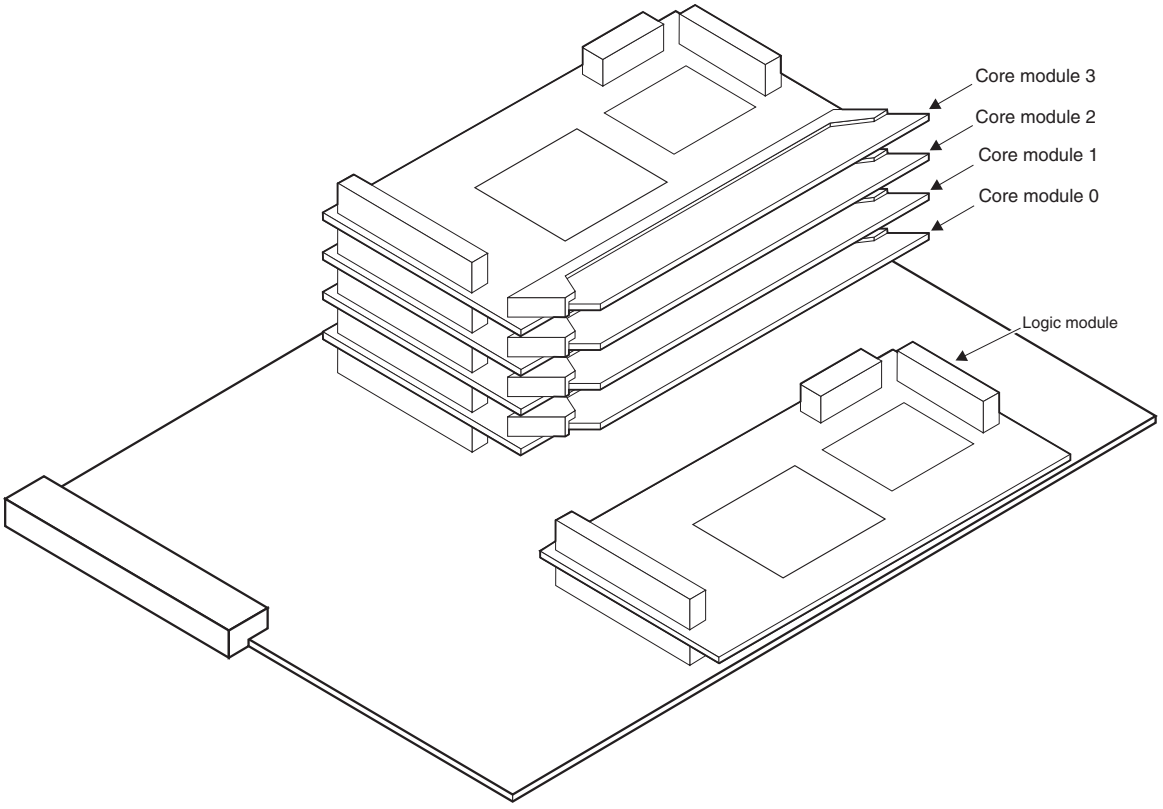


Figure 2-3 Assembled Integrator system

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 system bus address of its SDRAM. 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 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 user guide for the motherboard you are using.

Chapter 3
Hardware Description

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

3.1 Microprocessor core

This section provides a brief overview of the four core types that can be fitted to the core modules described in this manual.

3.1.1 ARM9™ Thumb® family

The ARM9 Thumb family cores are low-power, high-performance 16/32-bit RISC microprocessors built around the ARM9TDMI™ processor core. The ARM9 Thumb family incorporates the Thumb 16-bit instruction set, to improve code density while maintaining high levels of performance.

The ARM9 Thumb family includes the ARM940T and ARM920T cached processor macrocells, each of which has been developed to address different application requirements.

ARM940T macrocell

The ARM940T is targeted at applications such as network interface cards, printers, and automotive control devices, where no additional system software is needed.

The ARM940T macrocell combines an ARM9TDMI processor core with 4KB instruction cache and 4KB data cache. The macrocell includes a *Memory Protection Unit* (MPU) and an AMBA bus interface for *System-on-Chip* (SoC) integration. The protection units allow you to define eight regions in memory, each with independent cache, write buffer enable, and access permissions. The protection unit is configured using on-chip registers, to provide a simple programmer's model that removes the need for page-mapping tables to be stored in memory.

ARM920T macrocell

The ARM920T macrocell is designed to support operating systems such as EPOC, Linux, and WindowsCE. It is targeted at hand-held, battery-powered, wireless devices such as PDAs, smart phones, and internet appliances. It combines high performance and cache features that allow real-time functions such as soft modem and voice-recognition interfaces to be run on the same CPU as the operating system.

The ARM920T macrocell combines the ARM9TDMI processor core with 16KB instruction cache and 16KB data cache. These can be set to write-through or write-back mode. The core also includes a *Memory Management Unit* (MMU) that supports virtual addressing requirements for a operating system. a write buffer, an AMBA bus interface for SoC integration and an *Embedded Trace Macrocell* (ETM) interface, that allows for Real-Time Trace functionality.

3.1.2 ARM7™ Thumb family

The ARM7 Thumb family is a range of low-power 32-bit RISC microprocessor cores optimized for cost and power-sensitive consumer applications. The ARM7 Thumb family incorporates the Thumb 16-bit instruction set enabling you to have 32-bit performance at lower system cost.

The family includes the ARM720T and ARM740T cached processor macrocells, each of which has been developed to address different market requirements.

The ARM720T macrocell

Targeted at mobile information appliances such as PDAs and smartphones, the ARM720T macrocell is ideal for all applications running EPOC32, Linux, or WindowsCE. It combines the ARM7TDMI™ core with 8KB unified cache and a full MMU. The virtual memory features provided by the MMU make it possible to safely use code downloaded from a network, such as the Internet or from an independent developer.

The ARM740T macrocell

The ARM740T macrocell is targeted at high-performance, embedded consumer applications such as printers and digital still cameras that require a cache but not a full MMU.

The ARM740T macrocell combines an ARM7TDMI core with 8KB unified cache, write buffer and MPU. The MPU provides up to eight user-defined memory areas, with programmable access permissions for each area. The ARM740T offers the performance advantages of a cached system, but with smaller die size and a simpler programming model.

3.2 SSRAM controller

The SSRAM controller is implemented in a Xilinx 9572 PLD and manages all SSRAM accesses. 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.

3.3 Core module FPGA

The core module FPGA contains five main functional blocks:

- *SDRAM controller* on page 3-7
- *Reset controller* on page 3-9
- *System bus bridge* on page 3-12
- *Debug communications interrupts* on page 3-31
- *Core module registers* on page 4-8.

The FPGA provides sufficient functionality for the core module to operate as a standalone development system, although with limited capabilities. System bus arbitration, system interrupt control, and input/output resources are provided by the system controller FPGA on the motherboard. See the user guide for your motherboard for further information.

Figure 3-1 illustrates the function of the core module FPGA and shows how it connects to the other devices in the system.

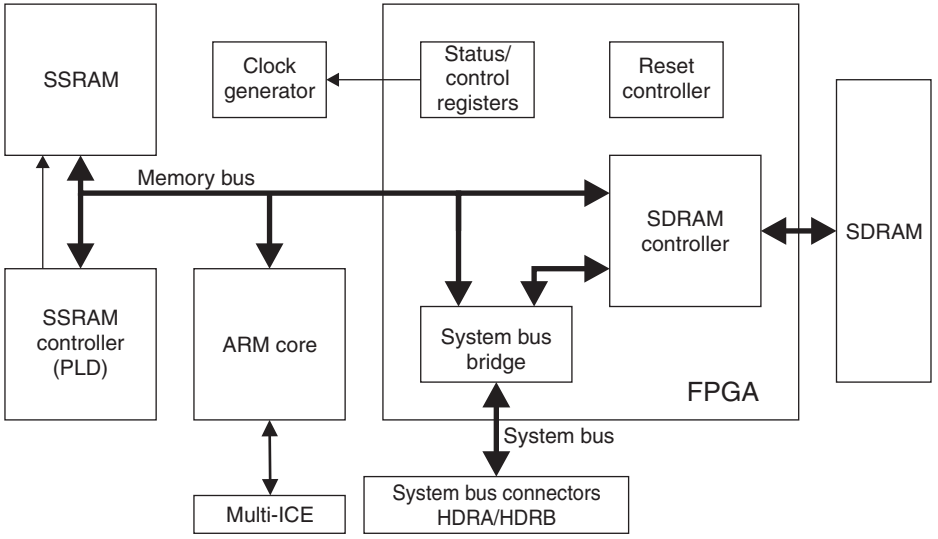


Figure 3-1 FPGA functional diagram

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 3-2 on page 3-6 shows the FPGA configuration mechanism.

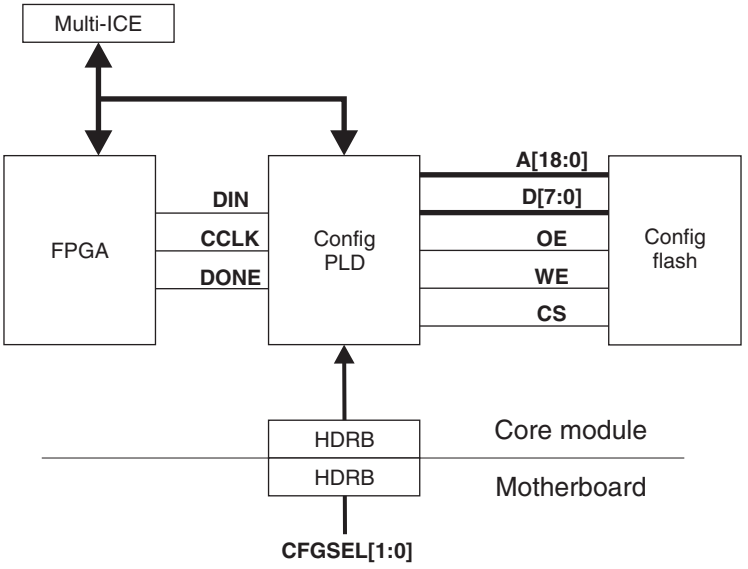


Figure 3-2 FPGA configuration

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
01	Reserved
10	Little endian AHB
11	Reserved

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 3-24.

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.

The CAS latency and memory size can be reprogrammed using the SDRAM control register (CM_SDRAM) at address `0x10000020` (see *Core module SDRAM status and control register* on page 4-15).

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 4-10 on page 4-19. If these values are not correctly set then SDRAM accesses may be slow or unreliable.

3.4.2 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* on page 3-12).

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 8 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` to `0x100001FC` (see *SDRAM SPD memory* on page 4-21).

Write accesses to the SPD EEPROM are not supported.

3.5 Reset controller

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 modules
- Multi-ICE
- software.

Figure 3-3 shows the architecture of the reset controller.

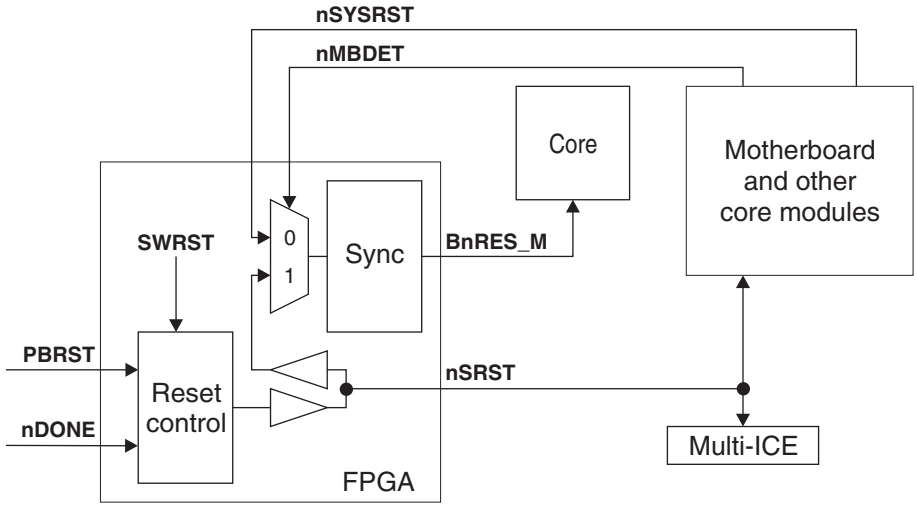


Figure 3-3 Core module reset controller

3.5.1 Reset signals

Table 3-2 describes the external reset signals.

Table 3-2 Reset signal descriptions

Name	Description	Type	Function
BnRES_M	Processor reset	Output	The **BnRES_M** 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 de-asserted 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** that 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 **BnRES_M** signal. When **nMBDET** is HIGH, **nSRST** is used to generate the **BnRES_M** 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 **BnRES_M** signal when the core module is attached to a motherboard. It is selected by the motherboard detect signal (**nMBDET**).

3.5.2 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** and this generates **nSRST** and resets the whole system (see *Core module control register* on page 4-12).

3.6 System bus bridge

The system bus bridge provides an asynchronous bus interface between the local memory bus and system bus connecting the motherboard and other modules.

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.

3.6.1 Processor accesses to the system bus

The first FIFO supports read and write accesses by the local processor to the system bus.

Processor writes

The data routing for processor writes to the system bus is illustrated in Figure 3-4.

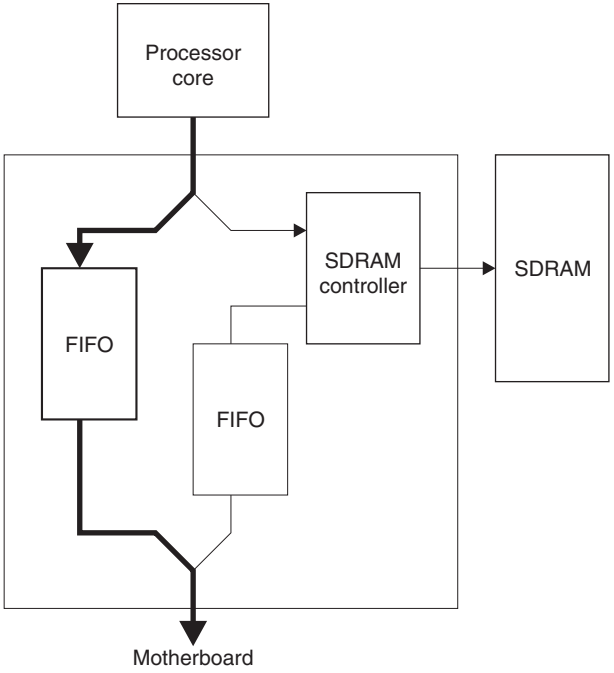


Figure 3-4 Processor writes to the system bus

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 3-5.

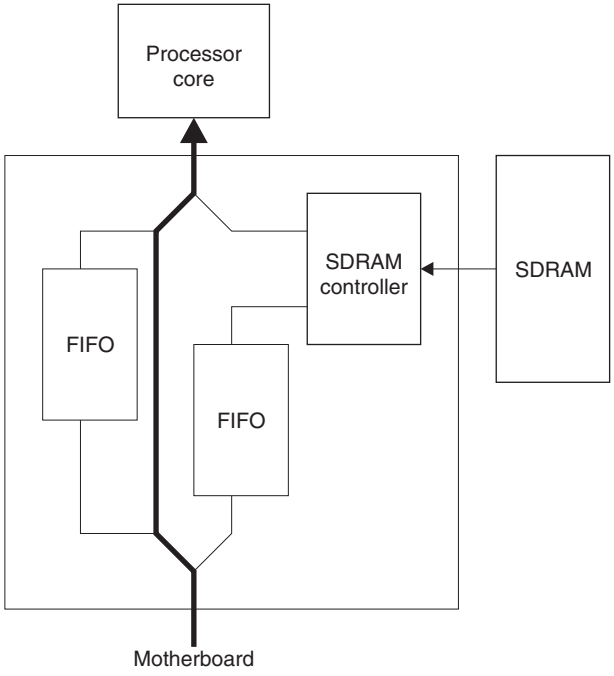


Figure 3-5 Processor reads from the system bus

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, allowed to complete 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 accesses* on page 4-4.

3.6.2 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.

System bus writes

The data routing for system bus writes to SDRAM is illustrated in Figure 3-6.

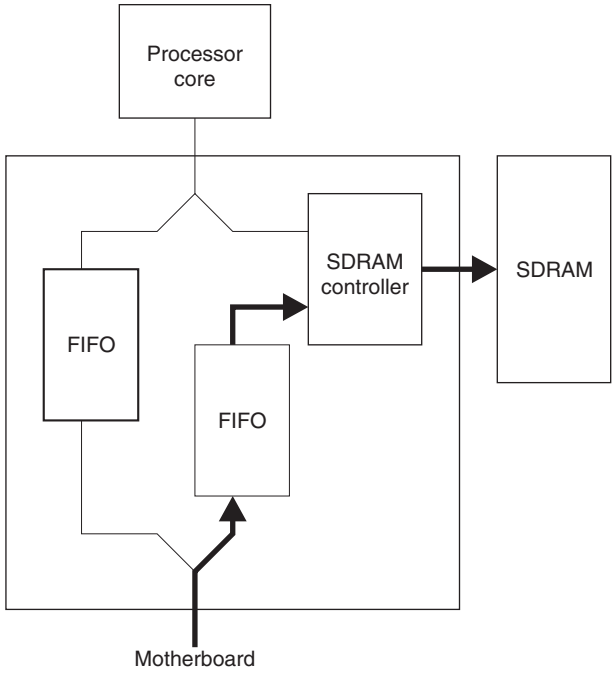


Figure 3-6 System bus writes to SDRAM

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 FIFO, and the transfer into the SDRAM completes when the SDRAM is available. If the FIFO is full, then the system bus master receives an ASB retract or AHB retry response indicating that the arbiter can 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 3-7.

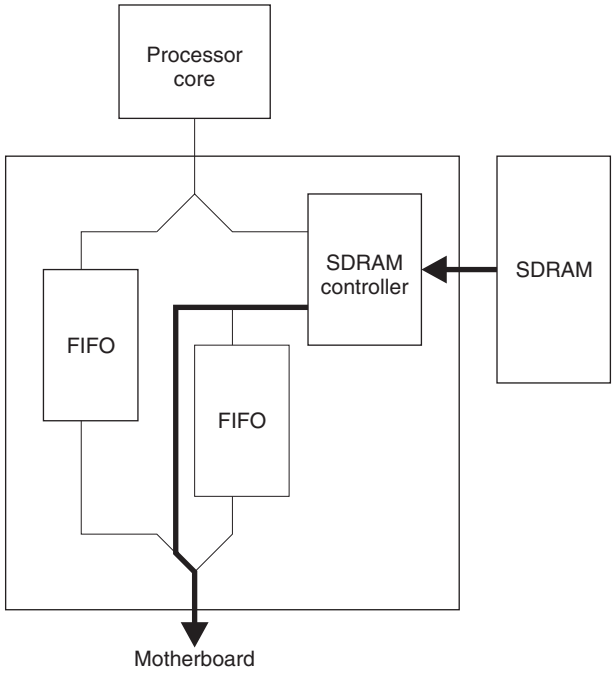


Figure 3-7 System bus reads from SDRAM

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.

3.6.3 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.

3.6.4 System bus signal routing

The core module is mounted onto a motherboard using the connectors HDRA and HDRB.These carry all signal connections between the boards, and provide mechanical mounting (see *Attaching the core module to a motherboard* on page 2-5).

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, pin 2 to pin 2 and so on. That is, the signals are routed straight through.

HDRB

Some signals on the HDRB connectors are assigned to specific modules. These are rotated in groups of four between the connectors on the bottom and top of each module to ensure that each module connects to a specific signal according to its ID and, in some cases, the stack it is mounted on. The ID for the bus master on a module is determined by the position of the module in the stack, see *Module ID selection* on page 3-18. This signal rotation scheme is illustrated in Figure 3-8 on page 3-17.

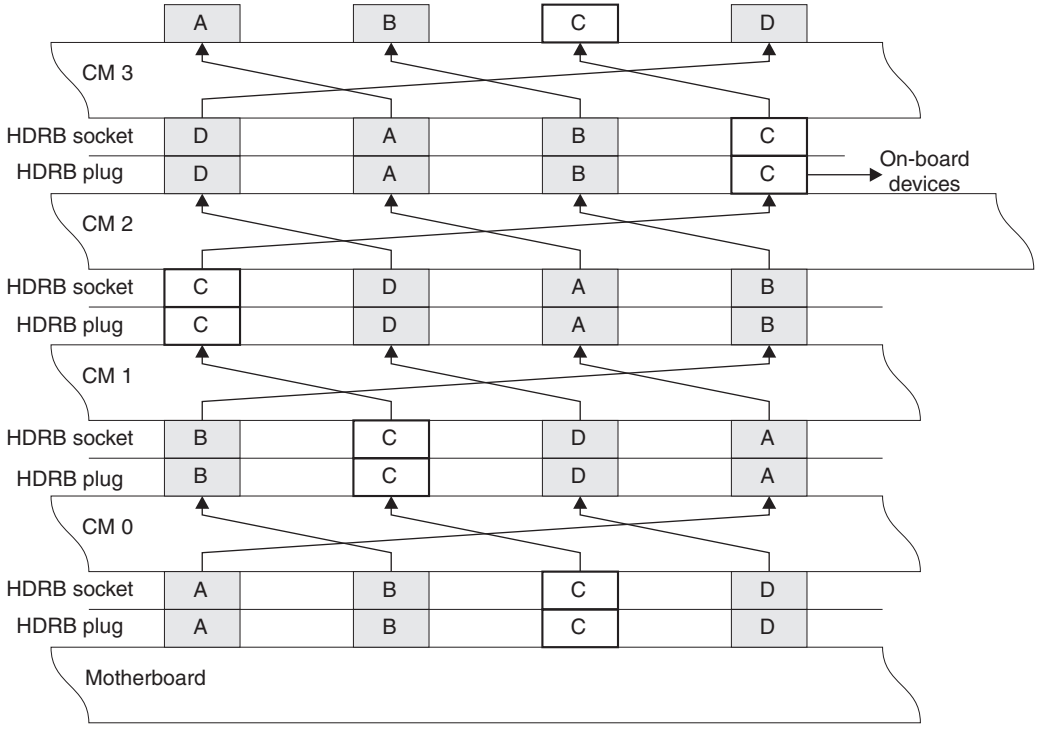


Figure 3-8 Signal rotation on HDRB

The example in Figure 3-8 illustrates how a group of four signals (labelled A, B, C, and D) are routed through a group of four connector pins up through the stack. It highlights how signal C is rotated as it passes up through the stack and is only utilized on module 2.

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

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

For details of the signals on the HDRB connectors, see *HDRB* on page A-4.

Note

The JTAG signals are described in Table 3-5 on page 3-29.

3.7 Module ID selection

The position of a core module in the HDRA/HDRB stack is used to determine:

- its ID
- its address in the alias memory region (see the user guide for your motherboard)
- 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 correct operation.

3.7.1 Module address decoding

The Integrator system implements a distributed address decoding system. These 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 response 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. 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. On the motherboard the **ID[3:0]** signals are tied to give the bit pattern 1110 and rotate as as they pass up the stack, as described in *System bus signal routing* on page 3-16.

The alias SDRAM address of a core module is determined in hardware, although a module can determine its own position by reading the state of **ID[3:0]** from the CM_STAT register (see *Core module status register* on page 4-14). Table 3-3 shows alias addresses for a core module fitted to a the motherboard on the HDRA/HDRB stack.

Table 3-3 Core module address decode

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

3.7.2 Interrupts

The system controller FPGA on the motherboard incorporates an interrupt controller that routes 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 3-4.

Table 3-4 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-4).

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

3.8 Clock generators

The core module provides its own clock generators and operates asynchronously with the motherboard. The clock generator provides three programmable clocks:

- processor core clock **CORECLK**
- processor local memory bus clocks **LCLK** and **nLCLK**.

In addition, a fixed frequency reference clock **REFCLK** is supplied to the FPGA. These clocks are supplied by two MicroClock ICS525 devices and by the SSRAM controller PLD, as illustrated in Figure 3-9.

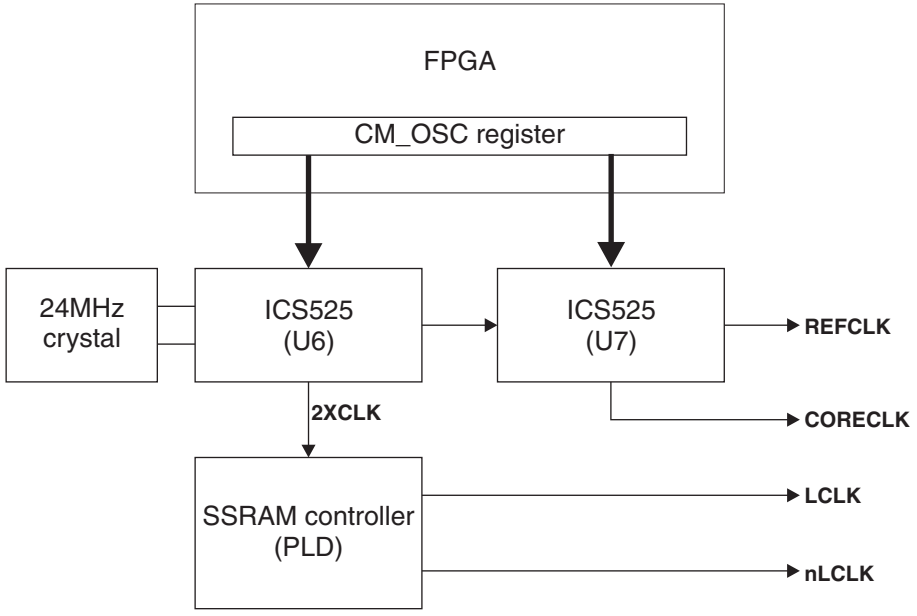


Figure 3-9 Core module clock generator

The ICS525s are supplied with a reference clock signal from a 24MHz crystal oscillator. The **2XCLK** output from the first ICS525 (U6) is supplied to the PLD and divided by two to produce the signals **LCLK** and **nLCLK**. The output from U7 provides the **CORECLK** signal. The reference output from U6 supplies the reference input to U7 and the reference output from U7 supplies the FPGA reference clock.

The output frequencies from the ICS525s are configured using *divider* input pins to produce a wide range of frequencies. However, this allows the clocks to be set to a higher speed than the maximum operating speed of the core module.

3.8.1 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 3-10 shows the values placed on the divider input pins and how the clock speeds are derived. The bits marked:

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

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 3-10 CORECLK divider values

The frequency of **CORECLK** can be derived from the formula:

freq = 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.

For details about programming C_VDW and C_OD, see *Core module oscillator register* on page 4-10.

Note

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

The **CORECLK** is buffered by a PI49FCT3805 to convert the *Phase-Locked Loop* (PLL) output to 3.3V signal level. The clock is series terminated with a 33Ω resistor and then drives a single load on the microprocessor core.

3.8.2 Processor bus clocks (LCLK and nLCLK)

The frequency of the processor bus clocks **LCLK** and **nLCLK** is determined by the frequency of **2XCLK**. The clock signal **2XCLK** is divided by 2 by the SSRAM controller PLD to produce **LCLK** and **nLCLK**.

The frequency of **LCLK** is controllable in 0.5MHz steps in the range 6MHz to 66MHz. This is achieved 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 maximum speed of **2XCLK** is limited by the speed of the SSRAM PLD.

Figure 3-11 shows the values placed on the divider input pins and how the clock speeds are obtained. The bits marked:

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

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 3-11 2XCLK divider values

The frequency of **LCLK** can be derived from the formula:

freq = (L_VDW+ 8)/L_OD

where:

L_VDW is the VCO divider word for the processor bus clock

L_OD is the output divider for the processor bus clock.

Note

Values for L_VDW and L_OD can be calculated using the ICS525 calculator on the Microclock website.

For details about programming L_VDW and L_OD, see *Core module oscillator register* on page 4-10.

The **LCLK** clock signal is buffered by a 5-output low-skew buffer PI49FCT3805 to drive five loads. These are:

- **SDRAM_CLK[3:0]**
- **SSRAM_CLK**.

The **nLCLK** clock signal is a phase-aligned inversion of the **LCLK** signal. It is buffered by a 5-output low-skew buffer PI49FCT3805 to four loads. These are:

- **ARM_BCLK_M**
- **PLD_BCLK_M**
- **FPGA_BCLK_M**
- **LA_BCLK_M**.

All clocks are series-terminated with 33Ω resistors placed as close to the source as possible.

3.8.3 FPGA reference clock (REFCLK)

The **REFCLK** signal is used by the FPGA to generate the SDRAM refresh clock and SPD EEPROM clock. This is a fixed-frequency clock of 24MHz output from the reference pin of the second ICS525 chip U7.

3.9 Multi-ICE support

The core module provides support for debug using JTAG. It provides a Multi-ICE connector and JTAG scan paths around the development system. Figure 3-12 shows the Multi-ICE connector and the CONFIG link.

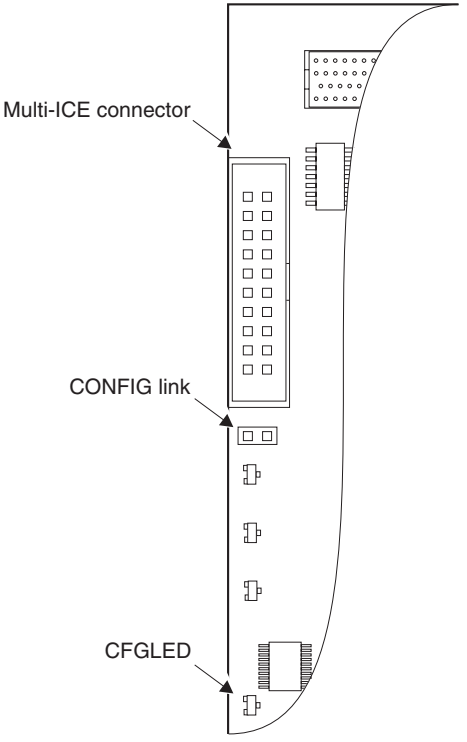


Figure 3-12 JTAG connector, CONFIG link, and LED

The CONFIG link is used to enable in-circuit programming of the FPGA and PLDs using Multi-ICE (see *JTAG connection modes* on page 3-27).

The Multi-ICE connector provides a set of JTAG signals that allow third-party JTAG debugging equipment to be used (see Table 3-5 on page 3-29). If you are debugging a development system with multiple core modules, connect the Multi-ICE hardware to the top core module.

3.9.1 JTAG scan path

This section describes the routing of JTAG data and clock signals around the core module and other Integrator modules to which it is attached.

Data path

Figure 3-13 shows a simplified diagram of the scan path.

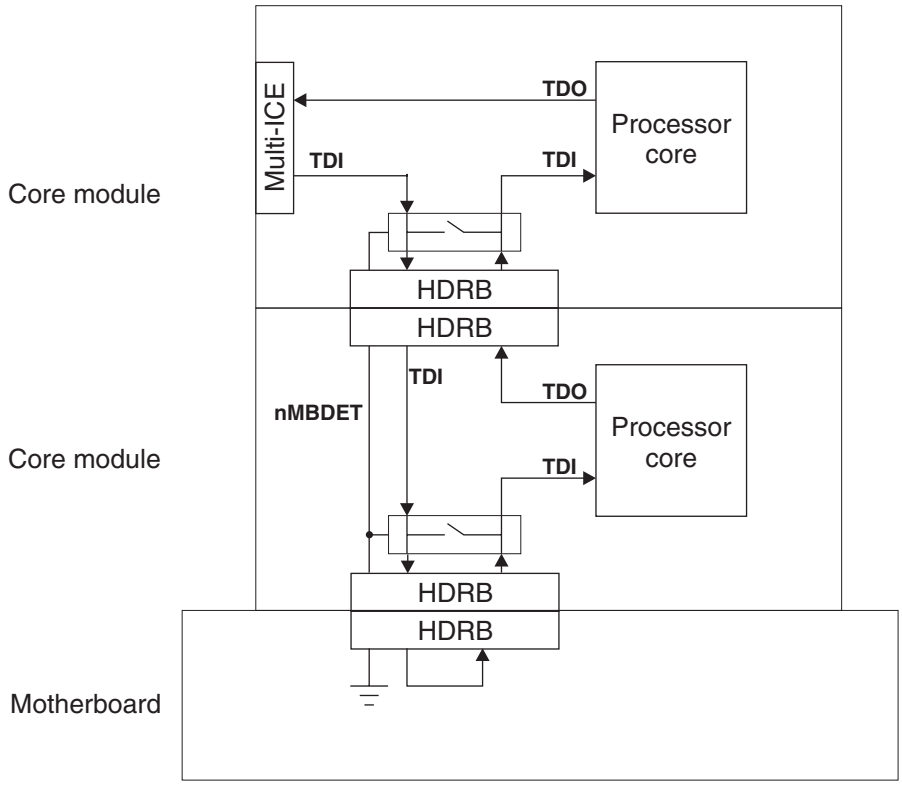


Figure 3-13 JTAG scan path (simplified)

When the core module is used as a standalone development system, the JTAG scan path is routed through 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 3-27.

Clock path

The clock path is routed in a similar way to the data path, although in the opposite direction. Figure 3-14 shows a simplified diagram of the clock path.

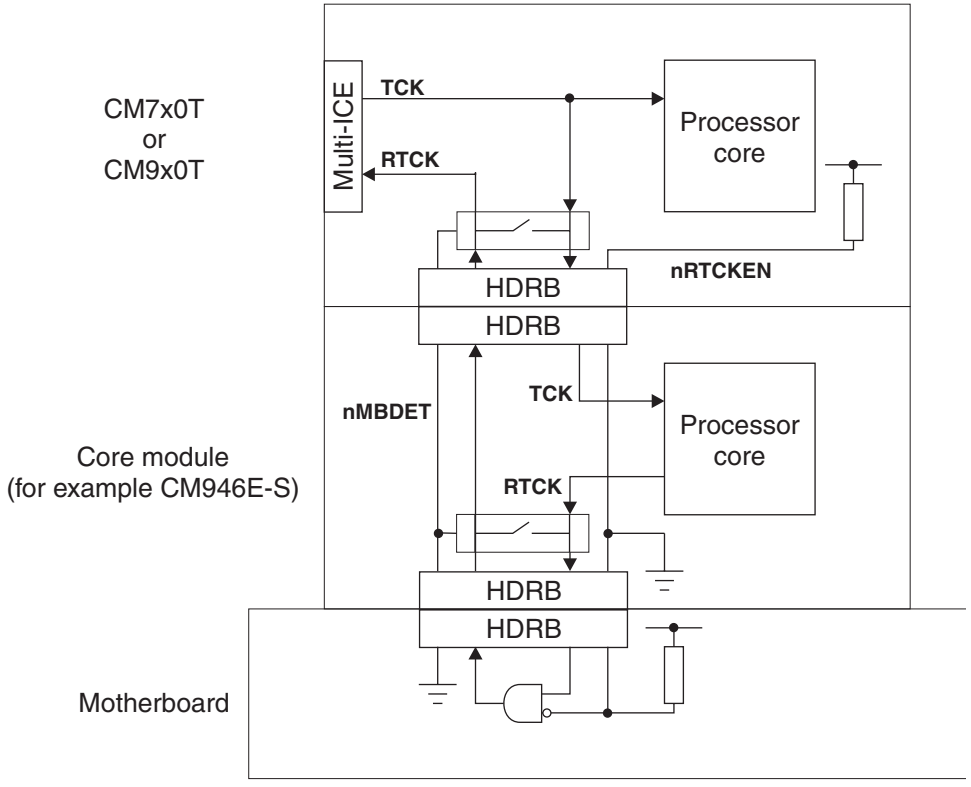


Figure 3-14 JTAG clock path

A number of synthesized cores (for example, the ARM946E-S) sample **TCK**. This introduces a delay into the clock path. Cores of this type pass on the clock signal as **RTCK**, and this 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 ARM940T, ARM920T, ARM740T, and ARM720T 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 requires **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.

3.9.2 JTAG connection modes

The core module is capable of operating in two modes:

- normal debug mode
- 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 3-12 on page 3-24). In this mode, the processor core and debuggable devices on other modules are accessible on the scan chain, as shown in Figure 3-13 on page 3-25.

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 allows 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 3-12 on page 3-24). This has the effect of pulling the **nCFGEN** signal LOW, 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 might cause unreliable operation.

After configuration or code updates you must:

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

The configuration mode allows 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. The PLD handles the transfer of data to the flash using JTAG.
- The PLDs are nonvolatile devices that can be programmed directly by JTAG.

3.9.3 JTAG signals

Figure 3-15 on page 3-29 shows the pinout of the Multi-ICE connector and Table 3-5 on page 3-29 provides a description of the JTAG signals.

Note

In the descriptions in Table 3-5 on page 3-29, the term JTAG equipment refers to any hardware that can drive the JTAG signals to devices in the scan chain. In most cases this will be Multi-ICE, although hardware from third-party suppliers can also be used to debug ARM processors.

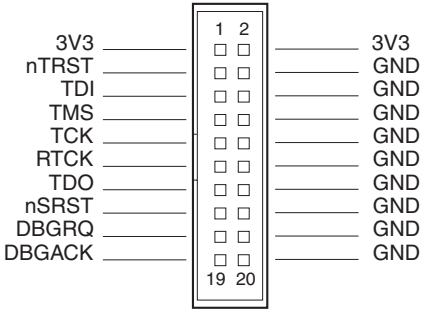


Figure 3-15 Multi-ICE connector pinout

Table 3-5 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 effect **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.

Table 3-5 JTAG signal description (continued)

Name	Description	Function
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.
nTRST	Test reset (from JTAG equipment)	This active LOW open-collector is used to reset the JTAG port and the associated debug circuitry on the core. 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 that 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.

Table 3-5 JTAG signal description (continued)

Name	Description	Function
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.

3.9.4 Debug communications interrupts

The processor core incorporates EmbeddedICE hardware 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 the scan chain. For a description of the debug communications channel, see the Technical Reference Manual for your core.

You can use interrupts to signal when 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 interrupt registers* on page 4-17).

Chapter 4
Programmer's Reference

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

4.1 Memory organization

This section describes the memory map. For a standalone core module, the memory map is limited to local SSRAM, SDRAM, and core module registers. For the full memory map of an Integrator development system, including 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 other ARM modules and Integrator systems. Table 4-1 shows the memory map.

Table 4-1 ARM Integrator/CM7x0T memory map

nMBDET	REMAP	Address range	Region size	Description
0	0	`0x00000000` to `0x0003FFFF`	256KB	Boot ROM (on motherboard)
0	1	`0x00000000` to `0x0003FFFF`	256KB	SSRAM
1	X	`0x00000000` to `0x0003FFFF`	256KB	SSRAM
X	X	`0x00040000` to `0x0FFFFFFF`	256MB	Local SDRAM
X	X	`0x10000000` to `0x107FFFFF`	8MB	Core module registers
X	X	`0x10800000` to `0x10FFFFFF`	8MB	SSRAM alias
0	X	`0x11000000` to `0xFFFFFFFF`	272MB to 4GB	System bus address space
1	X	`0x11000000` to `0xFFFFFFFF`	272MB to 4GB	Abort

4.1.2 Boot ROM and SSRAM accesses

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 to these devices are controlled by the REMAP bit and the motherboard detect signal, **nMBDET**. The **nMBDET** signal is permanently grounded by the motherboard so that it is pulled LOW on the core module when it is fitted. The effect on the memory map is shown in Figure 4-1 on page 4-3. The SSRAM is also mapped into the address space `0x10800000` to `0x10FFFFFF`. This is filled with repeated images of the 256KB of SSRAM that is fitted to the core module.

Note

Earlier versions of the FPGA do not have the SSRAM alias at `0x10800000`. If you want to use this alias you must upgrade the FPGA configuration, See `www.arm.com` for details.

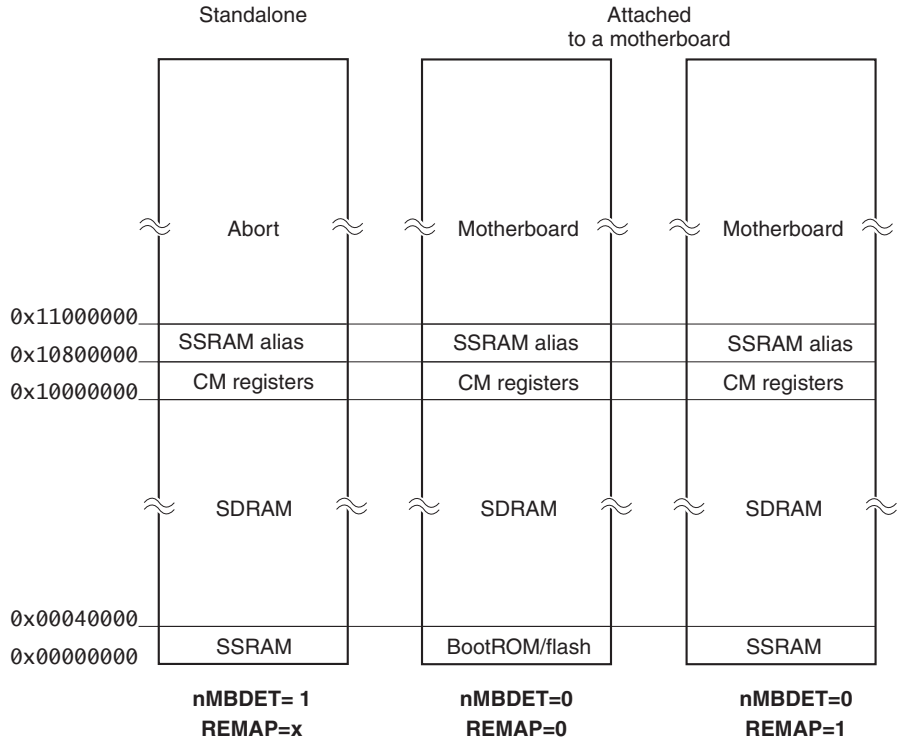


Figure 4-1 Effect of remap and motherboard detect

4.1.3 Using remap

The REMAP bit only has 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 As it is after a reset. The boot ROM or first 256KB of flash on the motherboard appears in the `0x00000000` to `0x0003FFFF` address range.

REMAP=1 The SSRAM appears in the `0x00000000` to `0x0003FFFF` address range.

4.1.4 Motherboard detect

The **nMBDET** signal functions as follows:

nMBDET=LOW

The core module is attached to a motherboard, and accesses in the `0x00000000` to `0x0003FFFF` address range (to the boot ROM/flash or SSRAM) are controlled by the REMAP bit.

nMBDET=HIGH

The core module is not attached, and accesses in the `0x00000000` to `0x0003FFFF` address range are routed to the SSRAM.

4.1.5 SDRAM accesses

The Integrator memory map provides a 256MB address space for SDRAM. When a smaller sized SDRAM DIMM is fitted, it is mapped repeatedly to fill the 256MB space. For example, a 64MB DIMM appears four times, as shown in Figure 4-2.

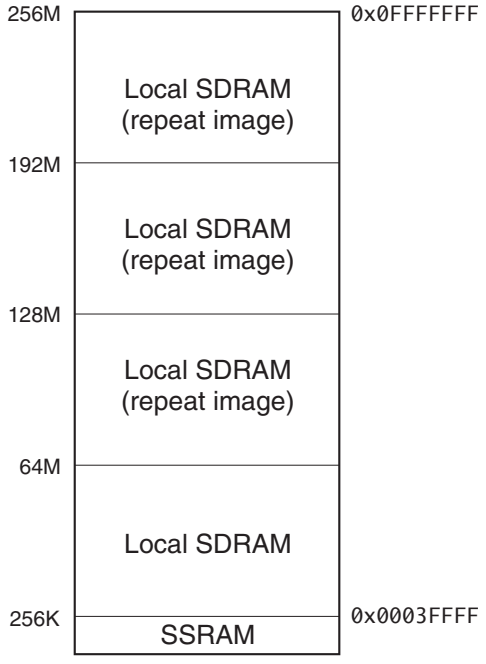


Figure 4-2 SDRAM repeat mapping for a 64MB DIMM

Local SDRAM access

The local processor can access the local SDRAM (that is, the SDRAM on the same core module) at `0x00000000` to `0x0FFFFFFF` in the core module address space. However, the lowest 256KB (`0x00000000` to `0x0003FFFF`) is hidden by the SSRAM or boot ROM, depending on whether the core module is attached to a motherboard and on the state of the REMAP bit.

The SDRAM cannot be accessed within this address space, although it can be accessed at one of its repeat images or at its alias location. In the case of a 256MB DIMM that fills the local SDRAM space, the first 256KB can only be accessed at the alias location. (see *Global SDRAM access* on page 4-5).

Global SDRAM access

If the core module is mounted on a motherboard, the SDRAM is mapped to appear at the *aliased module memory* region of the combined Integrator system bus memory map. The SDRAM can be accessed by all bus masters at its alias location, and accessed by the local processor at both its local and alias locations.

The system bus address for a core module is automatically controlled by its position in the stack (see *Module ID selection* on page 3-18). Figure 4-3 on page 4-6 shows the local and alias addresses of the SDRAM on four core modules.

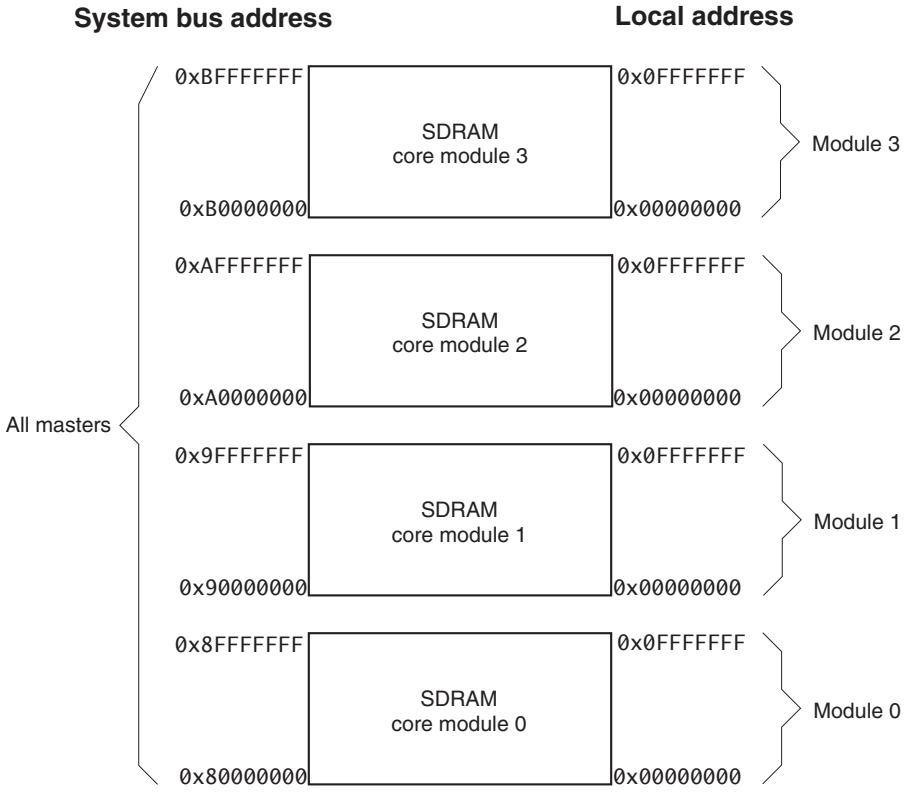


Figure 4-3 Core module local and alias addresses

By reading the CM_STAT register, a processor can determine which core module it is on and, therefore, the alias location of its own SDRAM (see *Core module status register* on page 4-14).

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.

4.2 Exception vector mapping

By default ARM cores map the exception vectors to begin at address 0. The ARM940T, ARM920T, and ARM720T cores allow the vectors to be moved to `0xFFFF0000` by setting the V bit in coprocessor 15 register 1 (see the technical reference manual for your core). However, Integrator core modules and motherboards have no physical memory at the high vector location (`0xFFFF0000` to `0xFFFF001C`). The measures you can take to use high vectors depends on the core module type as follows:

ARM720T and ARM920T

These cores contain an MMU which enables you to map the logical address `0xFFFF0000` to `0xFFFF001C` to a physical address that contains real memory, for example, to the core module SDRAM.

ARM720T, ARM920T, and ARM940T

A second approach that is suitable for all cores that support high vectors, including the ARM940T (that does not have an MMU) is to implement an area of physical memory at `0xFFFF0000` to `0xFFFF001C`. For example, you could mount the core module onto a motherboard and use a logic module to provide physical memory at this location.

Note

The ARM740T core does not support high vectors and maps the exception vectors to begin at 0.

4.3 Core module registers

The core module status and control registers allow 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	Description
CM_ID	`0x10000000`	Read	Core module identification register
CM_PROC	`0x10000004`	Read	Core module processor register
CM_OSC	`0x10000008`	Read/write	Core module oscillator values
CM_CTRL	`0x1000000C`	Read/write	Core module control
CM_STAT	`0x10000010`	Read	Core module status
CM_LOCK	`0x10000014`	Read/write	Core module lock
CM_SDRAM	`0x10000020`	Read/write	SDRAM status and control
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
CM_SPD	`0x10000100` to `0x100001FC`	Read	SDRAM SPD memory

Note

All registers are 32-bits wide and do not support byte writes. Write operations must be wordwide. Preserve the bits marked as *reserved* 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.

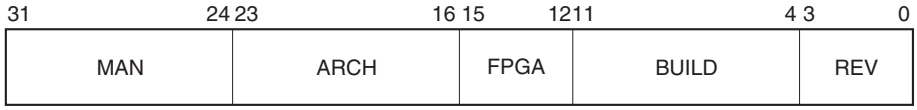


Table 4-3 describes the core module ID register bits.

Table 4-3 CM_ID register bit descriptions

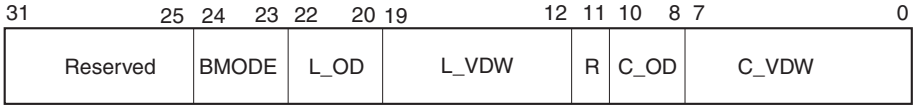
Bits	Name	Access	Function
31:24	MAN	Read	Manufacturer: 0x41 = ARM
23:16	ARCH	Read	Architecture: 0x00 = generic ARM7x0T or ARM9x0T, 4 word SDRAM bursts
15:12	FPGA	Read	FPGA type: 0x00 = XC4036
11:4	BUILD	Read	Build value
3:0	REV	Read	Revision: 0x0 = Rev A 0x1 = Rev B

4.3.2 Core module processor ID register

The core module processor register (CM_PROC) is a read-only register that contains the value 0x00000000. Information about the processor can be obtained by reading coprocessor 15 register 0.

4.3.3 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 3-20). In addition, it provides information about processor bus mode setting.



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

Table 4-4 describes the core module oscillator register bits.

Table 4-4 CM_OSC register

Bits	Name	Access	Function
31:25	Reserved	Use read-modify-write to preserve value.	
24:23	BMODE	Read	For ARM740T and ARM720T, this field contains 00 which indicates that the processor bus mode is selected by writing to CM_CTRL register (see *Core module control register* on page 4-12). For ARM940T and ARM920T, 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.

Table 4-4 CM_OSC register (continued)

Bits	Name	Access	Function
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). 00100000 = 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). 00101010 = 50MHz (default with OD = 2).

4.3.4 Core module control register

The core module control register (CM_CTRL) is a read/write register that provides control of a number of user-configurable features of the core module.

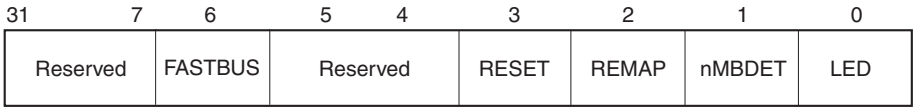


Table 4-5 describes the core module control register bits.

Table 4-5 CM_CTL register

Bits	Name	Access	Function
31:7	Reserved	Use read-modify-write to preserve value.	
6	FASTBUS	Read/write	ForARM740T and ARM720T, this is used to select the bus clocking mode: 0 = asynchronous 1 = FastBus This bit is reserved for ARM940T and ARM920T cores. The bus clocking mode is controlled by a bit in coprocessor 15 register 1.
5:4	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 core 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.

Table 4-5 CM_CTL register (continued)

Bits	Name	Access	Function
2	REMAP	Read/write	This only has affect when the core module is mounted onto a motherboard. When this is the case, and this bit is a 0, accesses to the first 256KB (`0x00000000` to `0x0003FFFF`) of memory are redirected into the motherboard.
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.

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 where in a stack this core module is positioned.

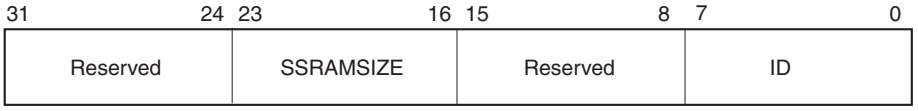


Table 4-6 describes the core module status register bits.

Table 4-6 CM_STAT register

Bit	Name	Access	Function
31:24	Reserved	Use read-modify-write to preserve value.	
23:16	SSRAMSIZE	Read	SSRAM size. This contians `0x04` to indicate that 256MB is fitted to the core module.
15:8	Reserved	Use read-modify-write to preserve value.	
7:0	ID	Read	Card number: `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, allowing it to be locked and unlocked. This mechanism prevents the CM_OSC register from being overwritten accidently.

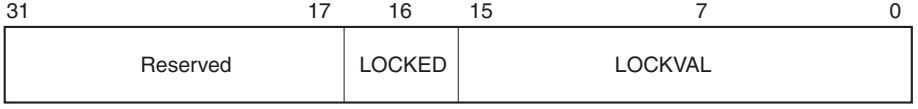


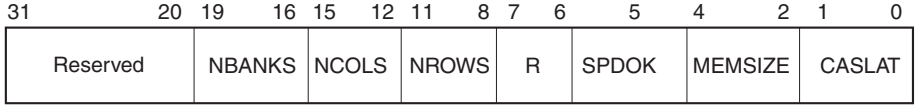
Table 4-7 describes the core module lock register bits.

Table 4-7 CM_LOCK register

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 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 using the SDRAM 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 4-21.

Table 4-8 describes the SDRAM status and control register bits.

Table 4-8 CM_SDRAM register

Bits	Name	Access	Function
31:20	Reserved	Use read-modify-write to preserve value.	
19:16	NBANKS	Read/write	Number of SDRAM banks. Set to the same value as byte 5 of SPD EEPROM.

Table 4-8 CM_SDRAM register (continued)

Bits	Name	Access	Function
15:12	NCOLS	Read/write	Number of SDRAM columns. Set to the same value as byte 4 of SPD EEPROM.
11:8	NROWS	Read/write	Number of SDRAM rows. Set to the same value as byte 3 of SPD EEPROM.
7:6	Reserved	Use read-modify-write to preserve value.	
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.
1:0	CASLAT	Read/write	These bits specify the CAS latency set for the core module. The bits are encoded as follows: 00 = Reserved 01 = Reserved 10 = 2 cycles (default) 11 = 3 cycles.

4.4 Core module 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 listed in Table 4-9.

Table 4-9 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 wordwide and bits marked as reserved in the following section should be written with zeros.

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

- status register
- raw status register
- enable register, which is accessed at 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 on page 4-18 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.

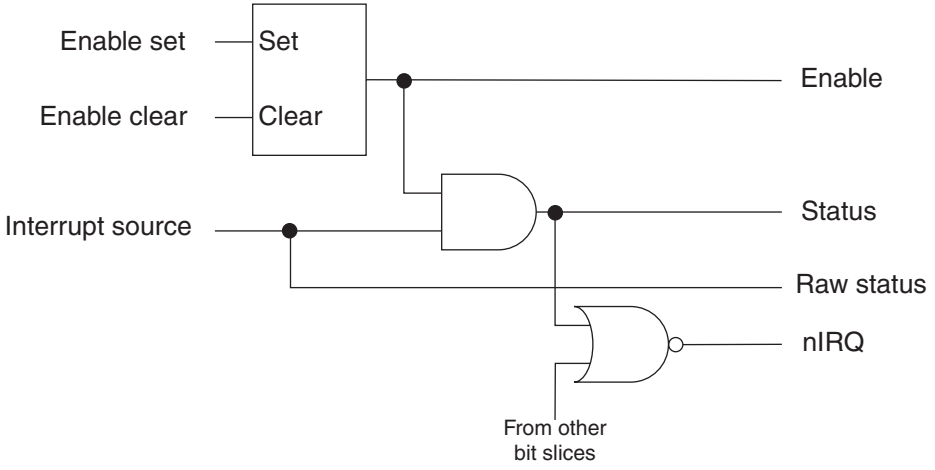


Figure 4-4 Interrupt control

4.4.1 IRQ and FIQ status register

The status register contains the logical AND of the bits in the raw status register and the enable register.

4.4.2 IRQ and FIQ raw status register

The 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.

4.4.3 IRQ and FIQ enable set register

The enable set locations are used to set bits in the enable register as follows:

- 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.

4.4.4 IRQ and FIQ enable clear register

The clear set locations are used to set bits in the enable register as follows:

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

4.4.5 Interrupt register bit assignment

The bit assignments for the IRQ and FIQ status, raw status, and enable register are shown in Table 4-10.

Table 4-10 IRQ and FIQ register bit assignment

Bit	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

4.4.6 Core module software interrupt registers

The core module interrupt controller provides a register for controlling and clearing software interrupts. This register is accessed at the software interrupt set and software interrupt clear locations. 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 4-11.

Table 4-11 IRQ register bit assignment

Bit	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 should not be confused with the ARM SWI software interrupt instruction. See the *ARM Architecture Reference Manual*.

4.5 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 allow faster random access to the SPD data (see *Serial presence detect* on page 3-7). The SPD memory contains 256 bytes of data, the most important of which are as shown in Table 4-12.

Table 4-12 SPD memory contents

Byte	Contents
2	Memory type
3	Number of row addresses
4	Number of column addresses
5	Number of 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 to 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

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

You can use the code segment shown in Example 4-1 on page 4-22 to correctly setup and remap the SDRAM.

Example 4-1 Setting up and remapping SDRAM

```
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
```


Appendix A
Signal Descriptions

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

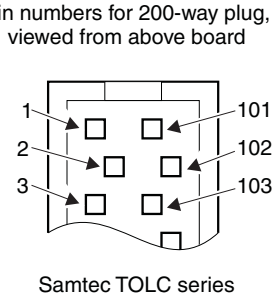
- *HDRA* on page A-2
- *HDRB* on page A-4.

Note

For the Multi-ICE connector pinout and signal descriptions see *JTAG signals* on page 3-28.

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.



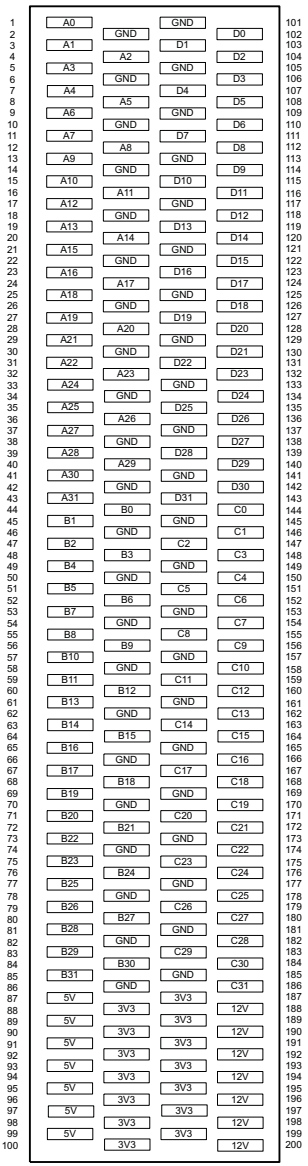


Figure A-1 HDRA plug pin numbering

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

Table A-1 Bus bit assignment

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

A.2 HDRB

The HDRB plug and socket have slightly different pinouts, as described below.

A.2.1 HDRB socket pinout

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

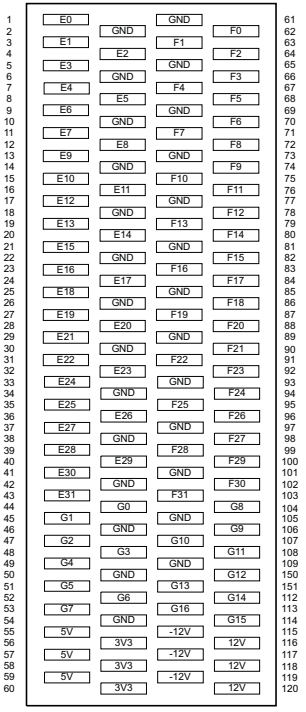


Figure A-2 HDRB socket pin numbering

A.2.2 HDRB plug pinout

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

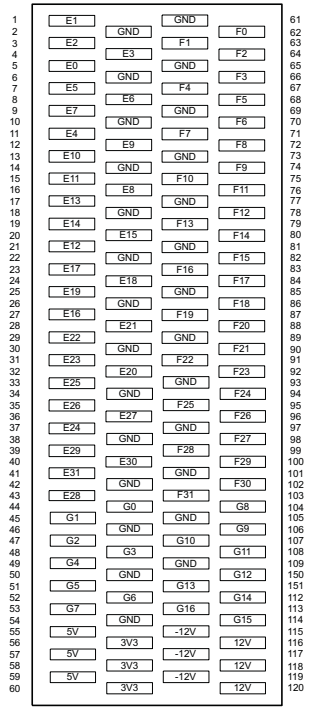


Figure A-3 HDRB plug pin numbering

A.2.3 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-2.

Table A-2 Signal cross-connections (example)

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 *System bus signal routing* on page 3-16.

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 3-28).

A.2.4 HDRB signal descriptions

Table A-3 describes the signals on the pins labeled E[31:0], F[31:0], and G[16:0] for AMBA AHB system bus.

Table A-3 HDRB signal description (AHB)

Pin label	Name	Description
E[31:28]	**HCLK[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.

Table A-3 HDRB signal description (AHB) (continued)

Pin label	Name	Description
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 **HBUSREQ** and **HGRANT** line numbers.
G0	**nMBDET**	Motherboard detect pin.

Table A-3 describes the signals on the pins labeled E[31:0], F[31:0], and G[16:0] for AMBA ASB system bus.

Table A-4 HDRB signal description (ASB)

Pin label	Name	Description
E[31:28]	**BCLK[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 processor.
E[19:16]	**nFIQ[3:0]**	Fast interrupt requests to processor.
E[15:12]	**ID[3:0]**	Core module stack position indicator.
E[11:8]	Reserved	-
E[7:4]	**AGNT[3:0]**	System bus grant to processor.
E[3:0]	**AREQ[3:0]**	System bus request from processor.
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.

Table A-4 HDRB signal description (ASB) (continued)

Pin label	Name	Description
G10	**RTCK**	Returned JTAG test clock.
G9	**nSYSRST**	Buffered system reset.
G8	**nTRST**	JTAG reset.
G7	**TDO**	JTAG test data out.
G6	**TDI**	JTAG test data in.
G5	**TMS**	JTAG test mode select.
G4	**TCK**	JTAG test clock.
G[3:1]	**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 **AREQ** and **AGNT** line numbers.
G0	**nMBDET**	Motherboard detect pin.

Appendix B
Specifications

This appendix contains the specifications for the core module. 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 supplies. The 12V inputs are supplied by the motherboard but not used by the core module.

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 current requirements at room temperature and nominal voltage. These include the current drawn by Multi-ICE, which is approximately 160mA at 3.3V.

Table B-2 Current requirements

System	At 3.3V	At 5V
Standalone core module	1A	100mA
Motherboard (AP or SP) and one core module	1.5A	500mA

An Integrator/AP or SP 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 Integrator timing parameters 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 figures are those you can expect under nominal conditions and should be used as a guideline when designing you own motherboards and modules. The 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
T_{clk}	**HCLK** minimum clock period	30	Representative of worst case maximum frequency
T_{isrst}	**HRESETn** deasserted setup time before HCLK	15	Applies to modules only
T_{ovrst}	**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
T_{issel}	**HSELx** setup time before **HCLK**	n/a	**HSELx** are internally generated, not visible at the pins
T_{istr}	Transfer type setup time before **HCLK**	5	-
T_{isa}	**HADDR[31:0]** setup time before **HCLK**	10	-
T_{isctl}	**HWRITE**, **HSIZE[2:0]** and **HBURST[2:0]** control signal setup time before **HCLK**	5	-
T_{iswd}	Write data setup time before **HCLK**	5	-
T_{isrdy}	Ready setup time before **HCLK**	5	-
T_{ismst}	Master number setup time before **HCLK**	n/a	Applies to modules with split capable slaves only
T_{ismlck}	Master locked setup time before **HCLK**	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
T_{ovrsp}	Response valid time after **HCLK**	15	-
T_{ovrd}	Data valid time after **HCLK**	15	-
T_{ovrdy}	Ready valid time after **HCLK**	15	-
T_{ovsplt}	Split valid time after **HCLK**	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
T_{isgnt}	**HGRANTx** setup time before **HCLK**	5	Modules implementing masters only
T_{isrdy}	Ready setup time before **HCLK**	5	-
T_{isrsp}	Response setup time before **HCLK**	5	-
T_{isrd}	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
T_{ovtr}	Transfer type valid time after **HCLK**	15	-
T_{ova}	Address valid time after **HCLK**	15	-
T_{ovctl}	Control signal valid time after **HCLK**	15	-
T_{ovwd}	Write data valid time after **HCLK**	15	-
T_{ovreq}	Request valid time after **HCLK**	15	Modules implementing masters only
T_{ovlck}	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)	Notes
T_{isrdy}	Ready setup time before **HCLK**	5	-
T_{isrsp}	Response setup time before **HCLK**	5	-
T_{isreq}	Request setup time before **HCLK**	10	-
T_{islck}	Lock setup time before **HCLK**	10	-

Table B-8 AHB arbiter input parameters (continued)

Parameter	Description	Time (ns)	Notes
T_{isplt}	Split setup time before **HCLK**	10	-
T_{istr}	Transfer type setup time before **HCLK**	5	-
T_{isctl}	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)	Notes
T_{ovgnt}	Grant valid time after **HCLK**	15	-
T_{ovmst}	Master number valid time after **HCLK**	15	-
T_{ovmlck}	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
T_{clk}	**BCLK** minimum clock period	40	Representative of worst case maximum frequency
T_{clkl}	**BCLK** LOW time	20	-
T_{clkh}	**BCLK** HIGH time	20	-
T_{isnres}	**BnRES** deasserted setup to rising **BCLK**	15	Applies to modules only
T_{ovnres}	**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
T_{isdsel}	**DSEL** setup to falling **BCLK**	n/a	**DSEL** is internally generated, not visible at the pins
T_{isa}	**BA[31:0]** setup to falling **BCLK**	10	Path through decoder is up to 30ns
T_{isctl}	**BWRITE** and **BSIZE[1:0]** setup to falling **BCLK**	10	-
T_{isdw}	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)	Notes
T_{ovresp}	**BWAIT**, **BERROR**, and **BLAST** valid after falling **BCLK**	10	-
T_{ovdr}	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
T_{isresp}	**BWAIT**, **BERROR** and **BLAST** setup to rising **BCLK**	15	-
T_{isdr}	For read transfers, **BD[31:0]** setup to falling **BCLK**	10	-
T_{isagnt}	**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)	Notes
T_{ovtr}	**BTRAN** valid after rising **BCLK**	10	-
T_{ova}	**BA[31:0]** valid after rising **BCLK**, all transfer types	10	-
T_{ovctl}	**BWRITE**, **BSIZE[1:0]**, and **BPROT[1:0]** valid after rising **BCLK**, all transfer types	10	-
T_{ovdw}	**BD[31:0]** valid after rising **BCLK**, all transfer types	30	-
T_{ovlok}	**BLOK** valid after rising **BCLK**	10	-
T_{ovareq}	**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)	Notes
T_{istr}	**BTRAN** setup to falling **BCLK**	10	-
T_{isresp}	**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
T_{ovresp}	**BWAIT**, **BERROR**, and **BLAST** valid after falling **BCLK**	10	-
T_{ovdsel}	**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)	Notes
T_{isareq}	**AREQ** setup to falling **BCLK**	10	-
T_{isresp}	**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)	Notes
T_{ovagnt}	**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
$T_{lokagnt}$	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 like those shown in the table 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 is also needed for clock skew, routing delays, and number of modules (that is, loading).

Not all FPGAs will 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 parameters. In general, a simpler approach is to increase the operating frequency 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 a number of different motherboards. Its size allows it to be mounted onto a CompactPCI motherboard while allowing the motherboard to be installed in a card cage.

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

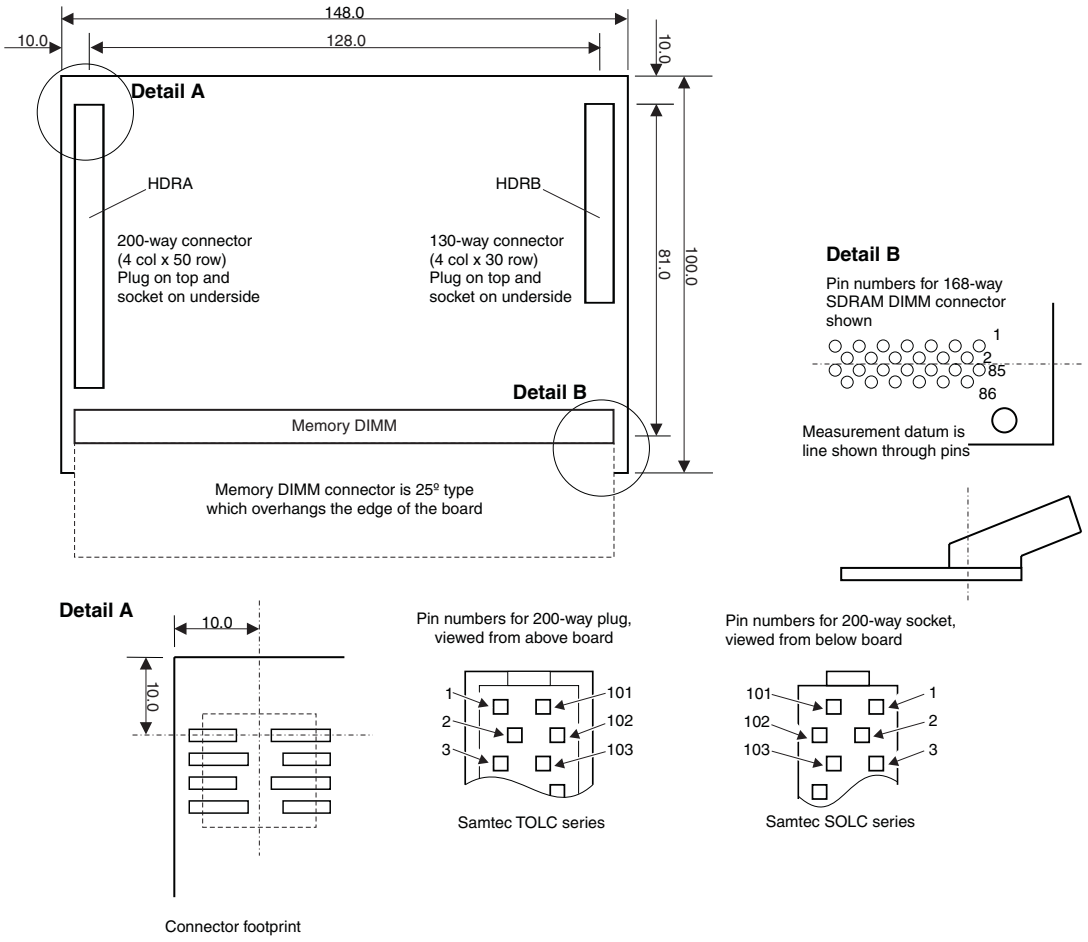


Figure B-1 Board outline

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.

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V