ARM Integrator/CM940T

User Guide

ARM DUI 0125A

ARM Integrator/CM940T
User Guide

Release information

Change history

Description	Issue	Change
8 September1999	A	New document

Proprietary notice

ARM, the ARM Powered logo, Thumb and StrongARM are registered trademarks of ARM Limited.

The ARM logo, AMBA, Angel, ARMulator, EmbeddedICE, ModelGen, Multi-ICE, ARM7TDMI, ARM7TDMI-S, ARM9TDMI, PrimeCell, and STRONG are trademarks of ARM Limited.

All other products or services mentioned herein may be trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM Limited in good faith. However, all warranties implied or expressed, including but not limited to implied warranties or merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Document 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).

ARM web address

`http://www.arm.com`

Electromagnetic conformity

This section contains *electromagnetic conformity* (EMC) notices.

Federal Communications Commission Notice

NOTE: This equipment has been tested and found to comply with the limits for a class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

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.

Contents
ARM Integrator/CM940T User Guide

Preface

Chapter 1 Introduction

Chapter 2 Getting Started

Chapter 3 Hardware Description

Preface

This preface introduces the ARM Integrator/CM940T core module and its reference documentation. It contains the following sections:

- *About this document* on page viii
- *Further reading* on page x
- *Feedback* on page xi.

About this document

This document describes how to set up and use the ARM Integrator/CM940T 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 ARM Integrator/CM940T 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 header's hardware architecture, including clocks, resets, and debug.

Chapter 4 *Programmer's Reference*

Read this chapter for a description of the header 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 document:

bold Highlights ARM processor signal names within text, and interface elements such as menu names. May also be used for emphasis in descriptive lists where appropriate.

italic Highlights special terminology, cross-references and citations.

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

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

`typewriter italic`
Denotes arguments to commands or functions where the argument is to be replaced by a specific value.

`typewriter bold`
Denotes language keywords when used outside example code.

Further reading

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

ARM publications

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

- *ARM740T Technical Reference Manual* (DDI 0008)
- *ARM Integrator/AP User Guide* (ARM DUI 0098)
- *ARM Integrator/SP User Guide* (ARM DUI 0099)
- *ARM Multi-ICE User Guide* (ARM DUI 0048)
- *AMBA Specification* (ARM IHI 0011)
- *ARM Architectural 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 both on the ARM Integrator/CM940T 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/CM940T

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/CM940T core module. It contains the following sections:

- *About the ARM Integrator/CM940T core module* on page 1-2
- *ARM Integrator/CM940T overview* on page 1-4
- *Links and indicators* on page 1-8
- *Test points* on page 1-10
- *Precautions* on page 1-11.

1.1 About the ARM Integrator/CM940T core module

The Integrator/CM940T core module provides you with the basis of a flexible development system which can be used in a number of different ways. With power and a simple connection to a Multi-ICE debugger, the core module 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/SP development motherboard
- mounted onto an ARM Integrator/AP development motherboard
- integrated into a third-party development or ASIC prototyping system.

Figure 1-1 on page 1-3 shows the layout of the ARM Integrator/CM940T.

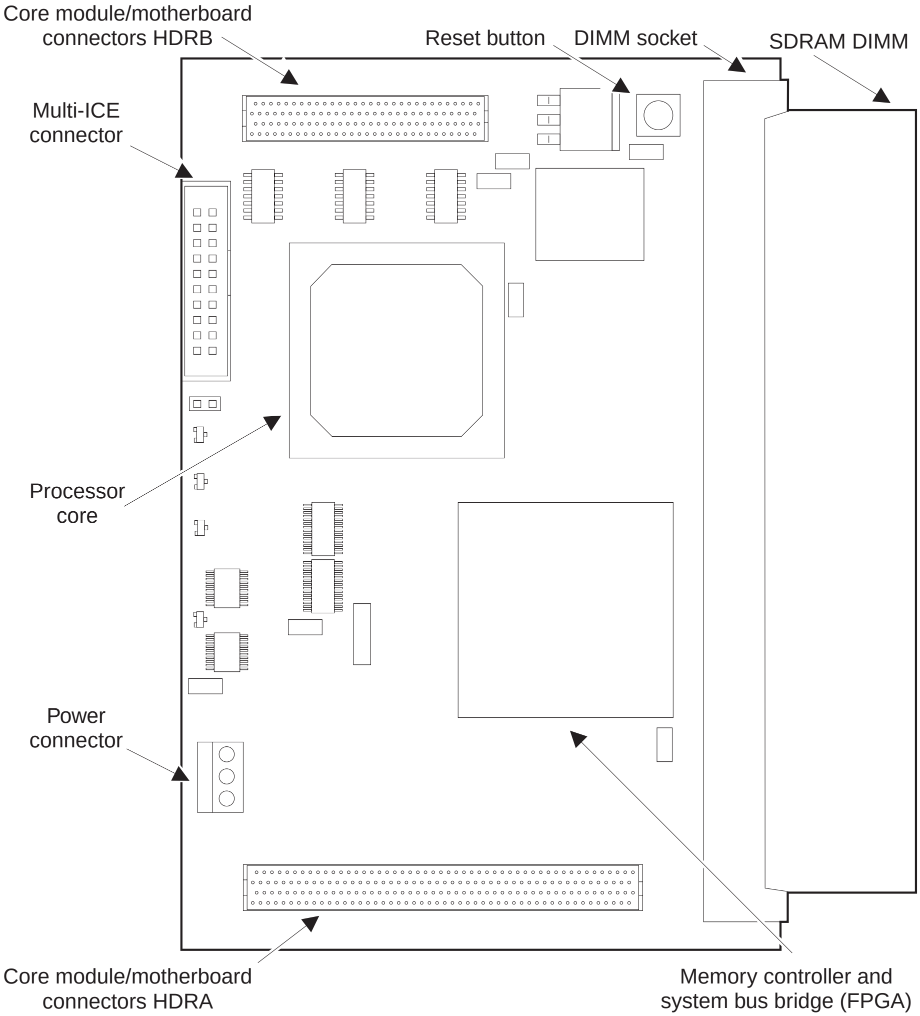

Core module/motherboard connectors HDRB
Reset button
DIMM socket
SDRAM DIMM
Multi-ICE connector
Processor core
Power connector
Core module/motherboard connectors HDRA
Memory controller and system bus bridge (FPGA)

Figure 1-1 Integrator/CM940T layout

1.2 ARM Integrator/CM940T overview

The major components on the core module are as follows:

- ARM940T microprocessor core
- core module FPGA which implements:
  - — SDRAM controller
  - — system bus bridge
  - — reset controller
  - — interrupt controller
  - — status, configuration, and interrupt registers.
- volatile memory comprising:
  - — up to 256MB of SDRAM (optional) via DIMM socket
  - — 256KB SSRAM.
- SSRAM controller
- clock generator
- system bus connectors
- Multi-ICE debug connector.

1.2.1 System architecture

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

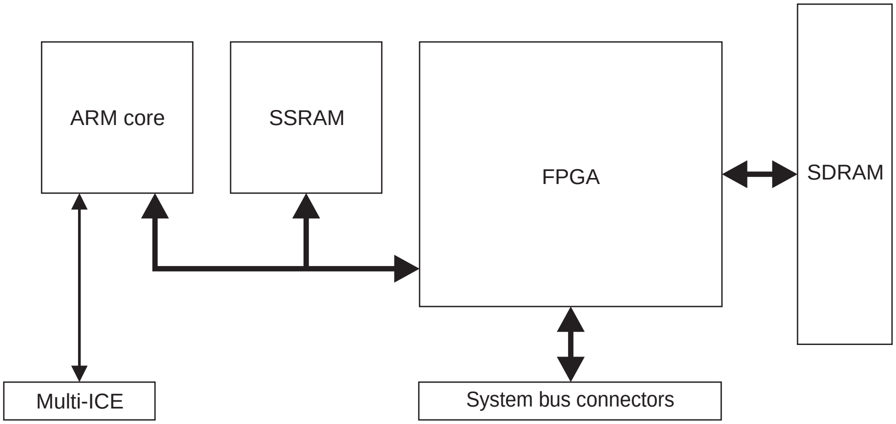


Figure 1-2 ARM Integrator/CM940T block diagram

1.2.2 Core module FPGA

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

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 3-6.

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 3-8.

System bus bridge

The system bus bridge provides an interface between the memory bus on the core module and the system bus on a motherboard. It allows the local processor access to interface resources on the motherboard and to the SDRAM on other core modules. It also allows access to the local SDRAM from the PCI bridge on the motherboard and processors on other core modules (see *System bus bridge* on page 3-11).

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*.

1.2.3 Volatile memory

The volatile memory system includes an SSRAM device, and a plug-in SDRAM memory module (referred to as *local* SDRAM when it is on the same core module as the processor). These areas of memory are closely coupled to the processor core to 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, by processors on other core modules, and by other system bus masters.

The SSRAM can only be accessed by the local processor.

1.2.4 Clock generator

The core module uses two on-board clocks:

- CPU clock up to 160MHz
- memory bus clock up to 66MHz.

These clocks are supplied by two clock generator chips. Their frequencies are selected via the oscillator control register (CM_OSC) within the FPGA. A reference clock is supplied to the two clock generators and to the FPGA (see *Clock generators* on page 3-17). The memory bus and system bus are asynchronous. This allows each bus to be run at the speed of its slowest device without compromising the performance of other buses in the system.

1.2.5 Multi-ICE connector

The Multi-ICE connector enables JTAG hardware debugging equipment, such as Multi-ICE, to be connected to the core module. It is possible to both drive and sense the system-reset line (**nSRST**), and to drive JTAG reset (**nTRST**) to the core from the Multi-ICE connector. See *Multi-ICE support* on page 3-21.

Note

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

1.3 Links and indicators

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

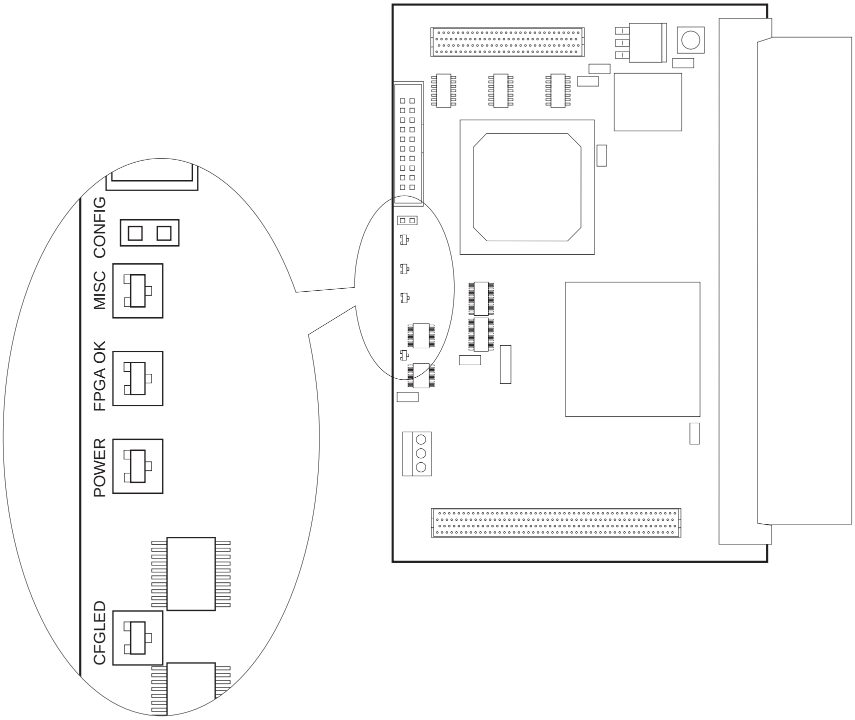


Figure 1-3 Links and indicators

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 via the control register (see *CM_CTRL (0x1000000C)* on page 4-11).
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 and five signal test points as an aid to debug. These are illustrated in Figure 1-4.

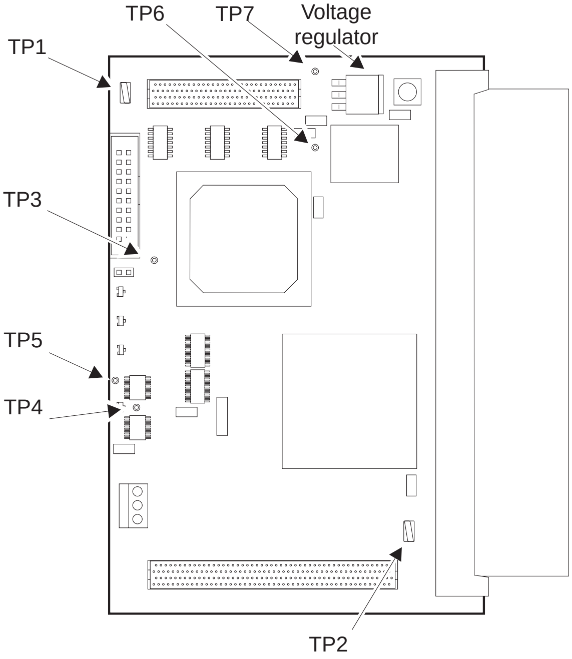


Figure 1-4 Test points

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 from the ARM940T microprocessor core
TP4	LBCLK	Local memory bus clock
TP5	Core clock	Clock supplied to the ARM940T microprocessor core
TP6	VDD940T	Output from voltage regulator
TP7	ADJ	ADJ pin of voltage regulator

1.5 Precautions

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

1.5.1 Ensuring safety

Warning

To avoid a safety hazard, only *Safety Extra Low Voltage* (SELV) equipment should be connected 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 you should 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 which could be:
  - 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 ARM Integrator/CM940T core module for use. It contains the following sections:

- *Setting up a standalone ARM Integrator/CM940T* on page 2-2
- *Attaching the ARM Integrator/CM940T to a motherboard* on page 2-5.

2.1 Setting up a standalone ARM Integrator/CM940T

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

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

2.1.1 Fitting an SDRAM DIMM

You should fit the following type of SDRAM module:

- PC66- or PC100-compliant 168pin DIMM
- unbuffered
- 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 ARM debuggers, 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, you should connect a bench power supply with 3.3V and 5V outputs to the power connector, as illustrated in Figure 2-1.

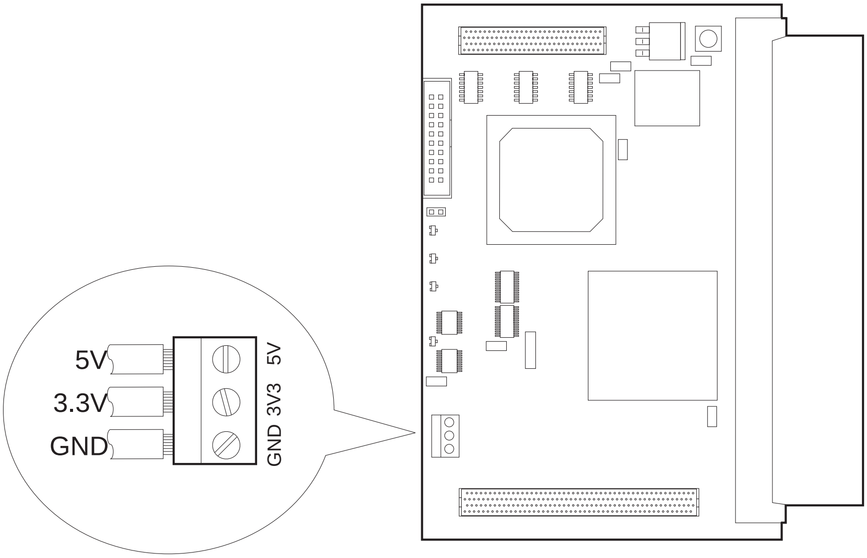


Figure 2-1 Power connector

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.

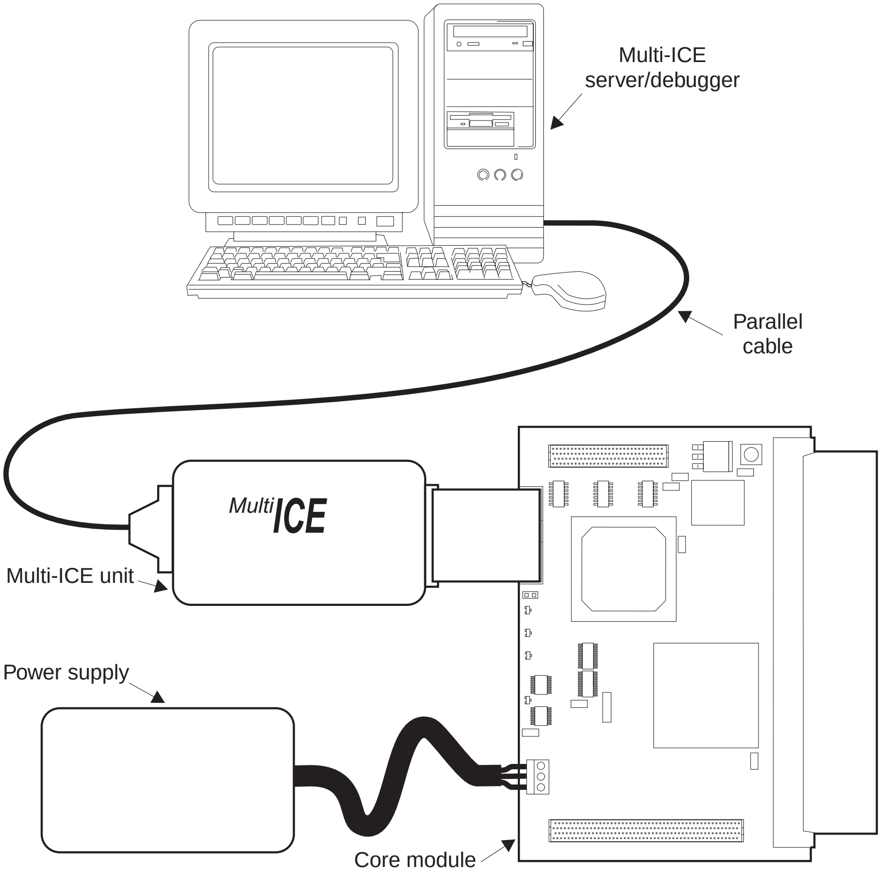


Figure 2-2 Multi-ICE connection to a core module

——— **Caution** ———

Because the core module does not provide non-volatile memory, 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 the Multi-ICE unit must be connected to the module at the top of the stack. The Multi-ICE server and the debugger can be on one computer or on two networked computers.

2.2 Attaching the ARM Integrator/CM940T to a motherboard

Attach the core module onto a motherboard (for example, the ARM Integrator/SP) 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/SP motherboard.

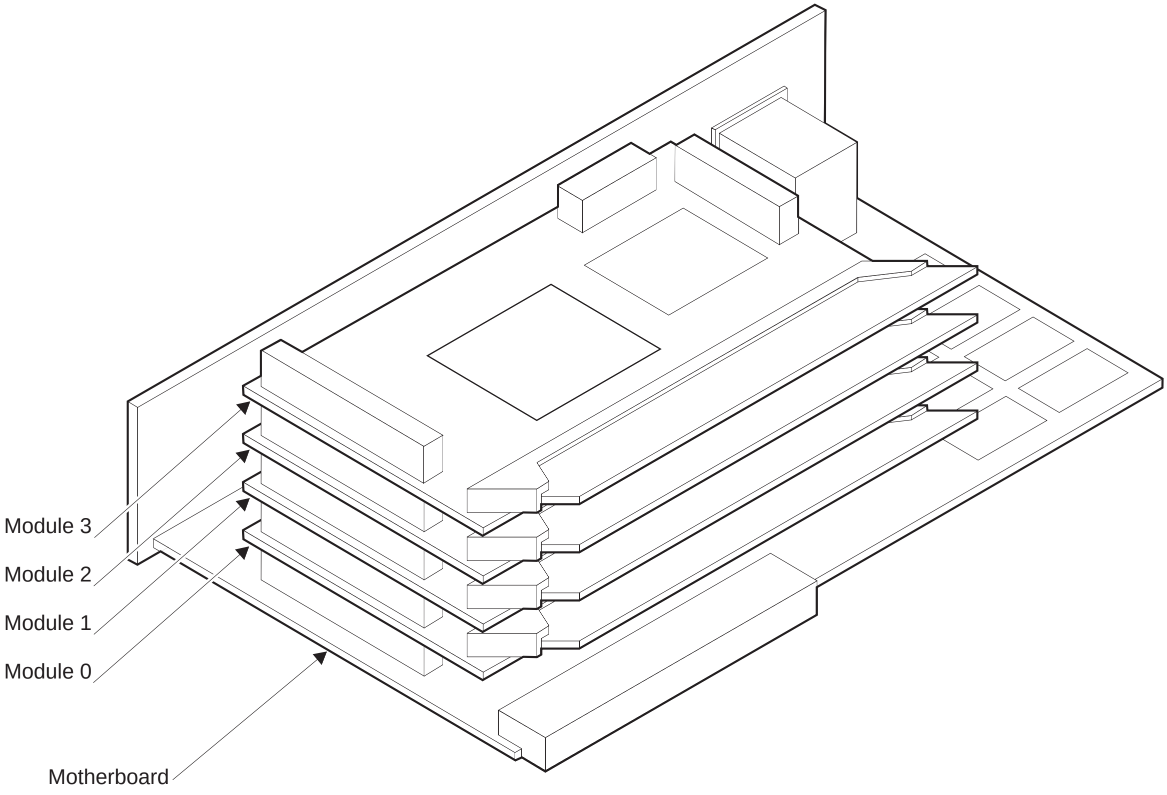


Figure 2-3 Assembled Integrator system

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 ID of the microprocessor it carries and 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 *CM_STAT (0x10000010)* on page 4-12.

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:

- *ARM940T microprocessor core* on page 3-2
- *SSRAM controller* on page 3-3
- *Core module FPGA* on page 3-4
- *SDRAM controller* on page 3-6
- *Reset controller* on page 3-8
- *System bus bridge* on page 3-11
- *Clock generators* on page 3-17
- *Multi-ICE support* on page 3-21.

3.1 ARM940T microprocessor core

The ARM940T cached processor macrocell is a member of the ARM9 Thumb family of high-performance 32-bit system-on-a-chip processors. It provides the following:

- ARM9TDMI RISC integer CPU
- 4KB instruction and data caches
- write buffer
- protection unit
- AMBA ASB bus interface.

The ARM940T processor employs a Harvard cache architecture, and so has separate 4KB instruction and 4KB data caches. Each cache has a 4-word line length.

The protection unit allows eight regions of memory to be defined, each with individual cache and write buffer configurations and access permissions.

The cache system is software-configurable to provide highest average performance or to meet the needs of real-time systems. Software configurable options include:

- random or round-robin cache line replacement algorithm
- write-through or writeback cache operation
- cache locking with granularity $^{1}/_{64}$ th of cache size.

The caches and write buffers improve CPU performance and minimize accesses to off-chip memory, thus reducing overall system power consumption. The ARM940T includes support for coprocessors, allowing a floating point unit or other application-specific hardware acceleration to be added.

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

The ARM940T also features a TrackingICE mode which allows an approach similar to a conventional ICE mode of operation.

For more information about the ARM940T, refer to the *ARM940T Technical Reference Manual*.

3.2 SSRAM controller

The SSRAM controller is implemented in a Xilinx 9572 PLD which enables the SSRAM to achieve single-cycle operation. In addition to controlling accesses to the SSRAM, the controller generates the processor response signals (**BWAIT**, **BERROR**, **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 3-6
- *Reset controller* on page 3-8
- *System bus bridge* on page 3-11
- *Core module registers* on page 4-7
- Debug interrupt controller, see *Debug communications interrupts* on page 3-27.

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.

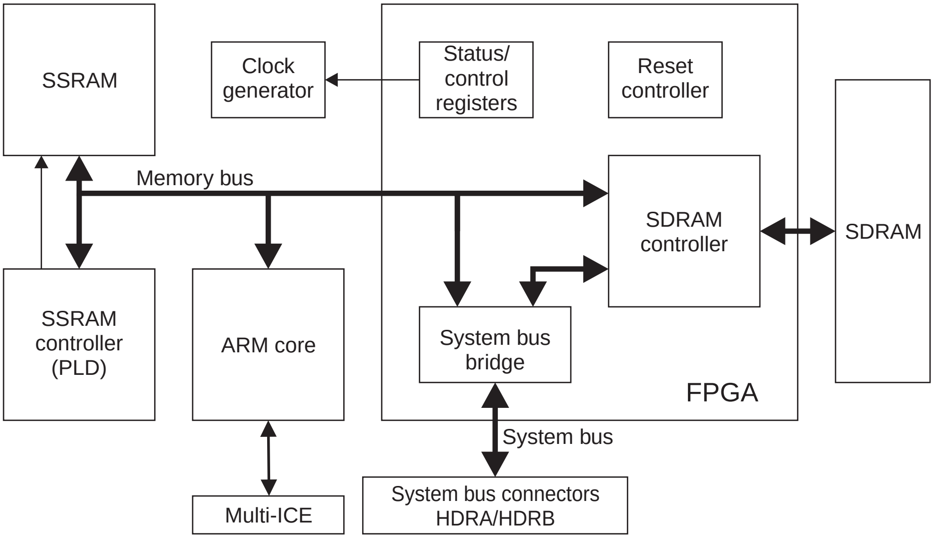


Figure 3-1 FPGA functional diagram

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 shows the FPGA configuration mechanism.

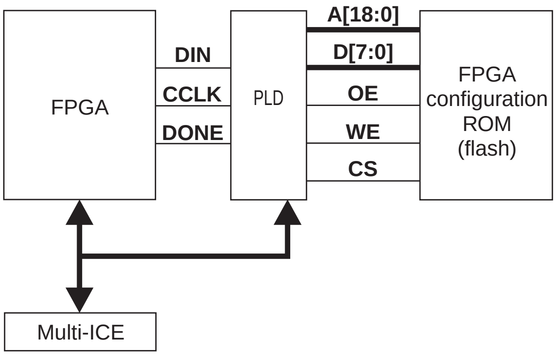


Figure 3-2 FPGA configuration

Multi-ICE can be used to reprogram the PLD, FPGA, and flash when the core module is placed in configuration mode. See *Multi-ICE support* on page 3-21.

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 via the SDRAM control register (CM_SDRAM) at address 0x10000020. See *CM_SDRAM (0x10000020)* on page 4-14.

Note

Before the SDRAM is used it is necessary to read the SPD memory and program the CM_SDRAM register with the parameters indicated in Table 4-8 on page 4-14. 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 3-11).

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 *CM_SPD (0x10000100 to 0x100001FC)* on page 4-16).

Write accesses to the SPD EEPROM are not supported.

3.5 Reset controller

The core module FPGA incorporates a reset controller which 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.

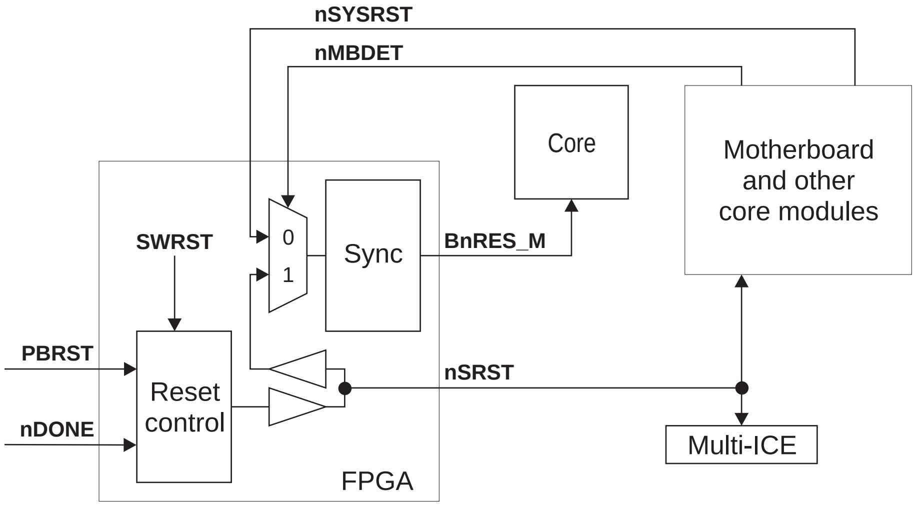


Figure 3-3 Core module reset controller

3.5.1 Reset signals

Table 3-1 describes the external reset signals.

Table 3-1 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 deasserted synchronously from the falling edge of the processor bus clock.
nDONE	FPGA configured	Input	The **nDONE** signal is an inversion of the open collector signal **FPGADONE** which 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 which can be triggered by writing to the reset bit in the CM_CTRL register. This generates the internal reset signal **SWRST** which generates **nSRST** and resets the whole system (see *CM_CTRL (0x1000000C)* on page 4-11).

3.6 System bus bridge

The system bus bridge provides an asynchronous bus interface between the local system 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, which extends onto the motherboard and other modules.

Processor writes

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

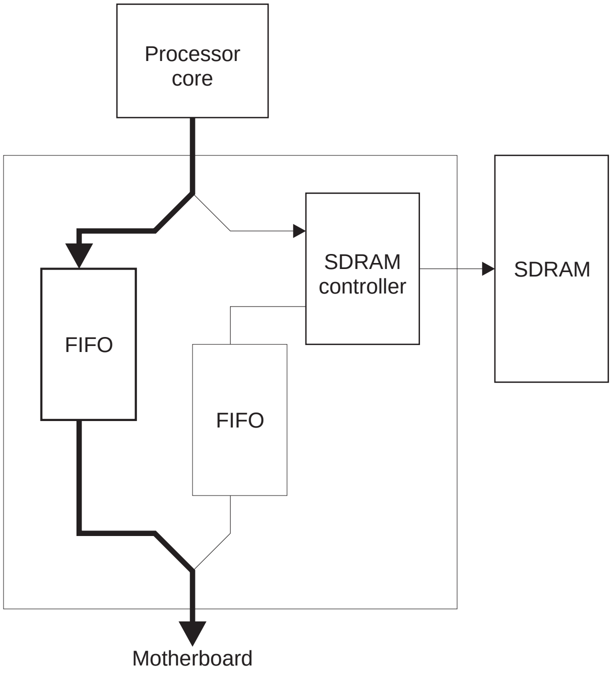


Figure 3-4 Processor writes to the system bus

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.

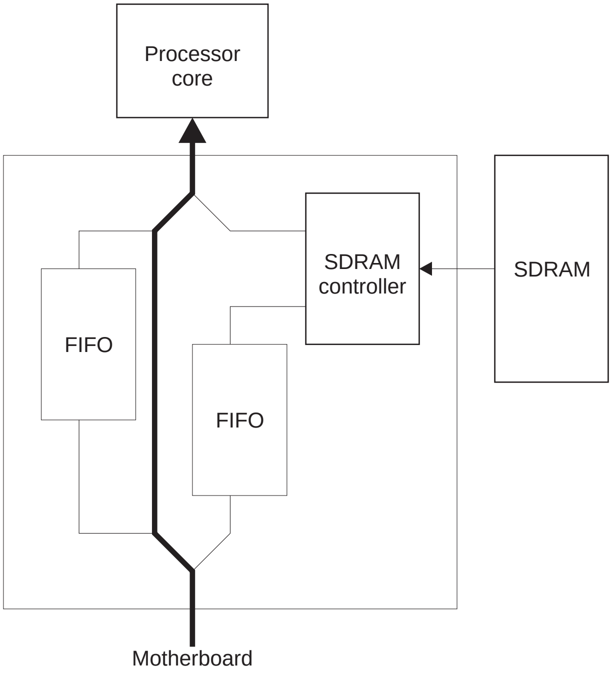


Figure 3-5 Processor reads from the system bus

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, completed on the system bus) before the read transfer is performed. The processor receives a wait response until the read transfer has completed on the system bus, when it receives the data and any associated bus error response from the system bus. For information about SDRAM addresses, see *SDRAM 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.

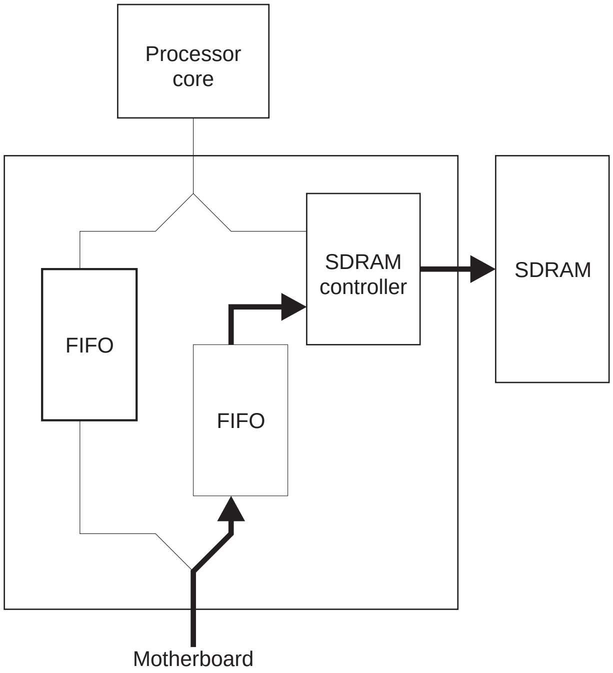


Figure 3-6 System bus writes to SDRAM

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 the FIFO, and the transfer into the SDRAM completes when the SDRAM is available. If the FIFO is full, then the system bus master receives a retract response indicating that the arbiter may grant the bus to another master and that this transaction must be retried later.

System bus reads

The data routing for system bus reads from SDRAM is illustrated in Figure 3-7.

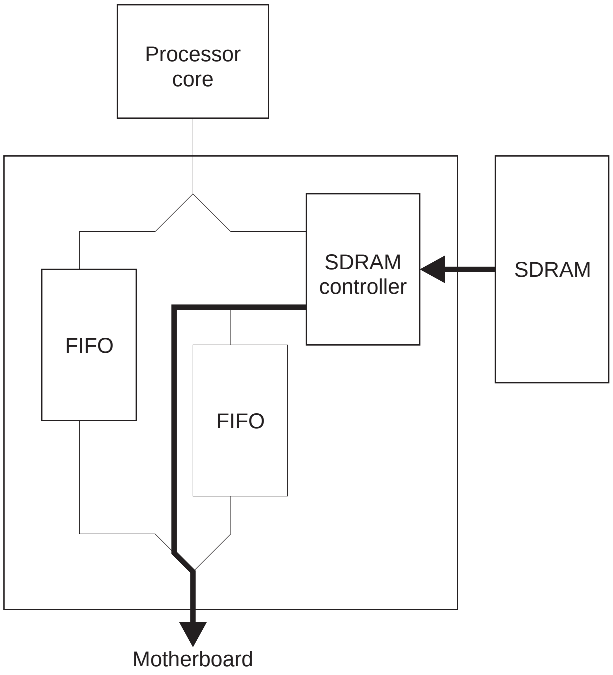


Figure 3-7 System bus reads from SDRAM

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

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 via 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 via the connectors HDRA and HDRB. As well as carrying all signal connections between the boards, these provide mechanical mounting (see *Attaching the ARM Integrator/CM920T 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

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

This signal rotation scheme is illustrated in Figure 3-8.

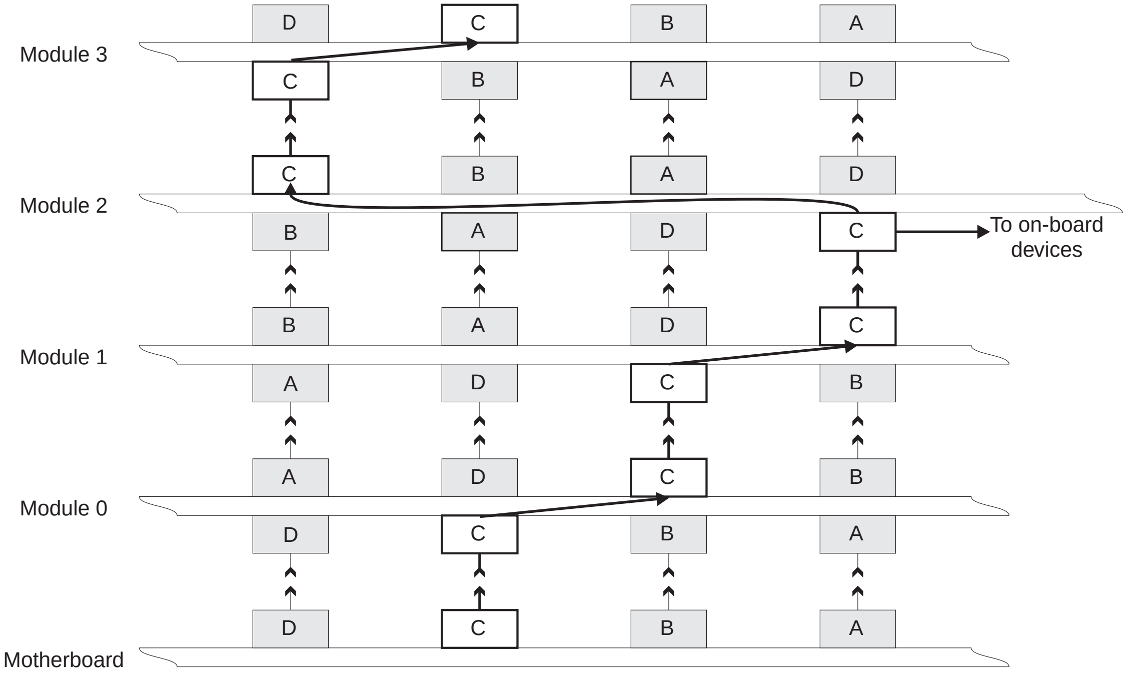


Figure 3-8 Signal rotation on HDRB

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 only utilized on module 2.

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

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

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

Note

The JTAG signals are discussed in *Multi-ICE support* on page 3-21.

3.6.5 Bus operating modes

The bus operating modes are programmed by writing to coprocessor 15 register 1 within the ARM940T microprocessor core.

The Integrator system supports:

- asynchronous and FastBus clocking
- little-endian addressing.

The Integrator system does not support:

- synchronous clocking
- big-endian addressing.

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

3.7 Clock generators

The core module provides its own clock generators and operates asynchronously with the motherboard. The clock generator provides two 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.

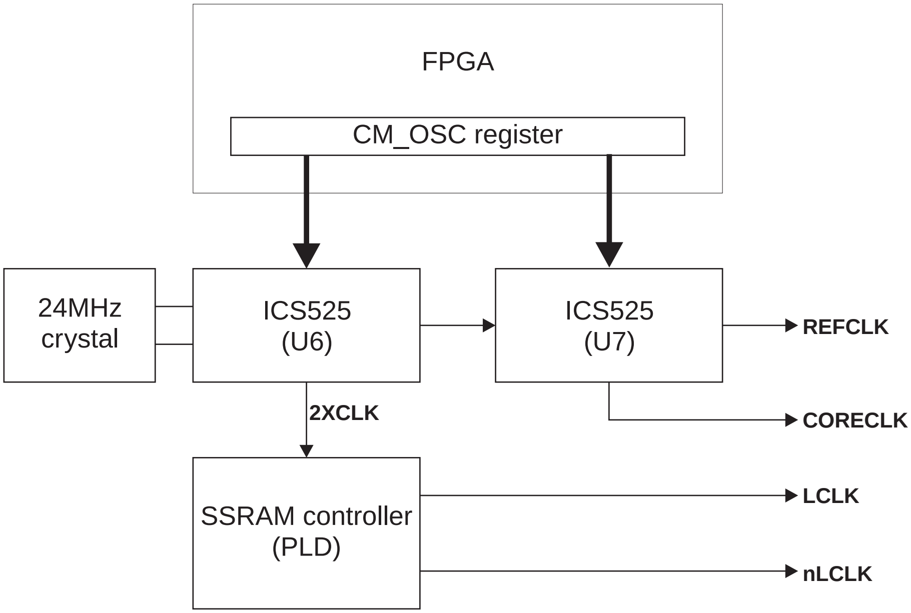


Figure 3-9 Core module clock generator

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.

3.7.1 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 via 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.

Table 3-2 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.

Table 3-2 CORECLK divider values

C_RDW R[6:0]							C_VDW V[8:0]									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 – 152 (<4 and >152 not allowed)									2-10		
12 – 160MHz in 1MHz steps																		

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 *CM_OSC (0x10000008)* on page 4-9.

Note

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

The **CORECLK** is buffered with 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.

3.7.2 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.

Table 3-3 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.

Table 3-3 2XCLK divider values

L_RDW R[6:0]							L_VDW V[8:0]									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 – 124 (<4 and >124 not allowed)									2-10		
12 – 132MHz in 1MHz steps (for **2XCLK**)																		

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 *CM_OSC (0x10000008)* on page 4-9.

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.

3.7.3 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 which is output from the reference pin of the second ICS525 chip U7.

3.8 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 3-10 shows the Multi-ICE connector and the CONFIG link.

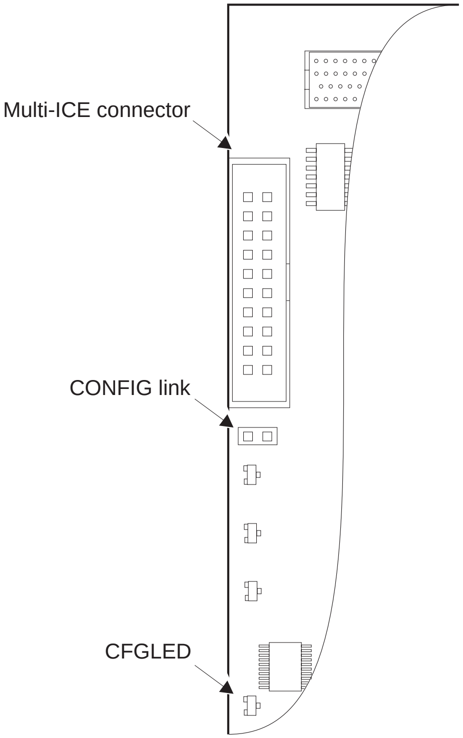


Figure 3-10 JTAG connector, CONFIG link, and LED

The CONFIG link is used to enable in-circuit programming of the FPGA and PLDs using Multi-ICE (see *Debugging modes* on page 3-23).

The Multi-ICE connector provides a set of JTAG signals allowing third-party JTAG debugging equipment to be used (see *JTAG signals* on page 3-24). If you are debugging a development system with multiple core modules, connect the Multi-ICE hardware to the top core module.

3.8.1 JTAG scan path

Figure 3-11 shows a simplified diagram of the scan path.

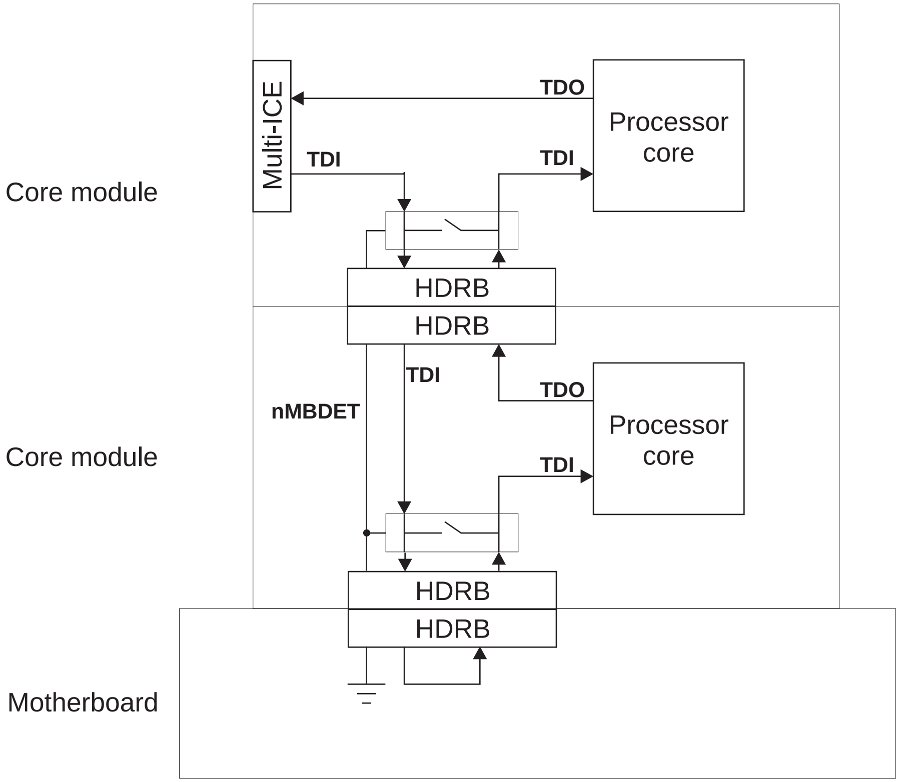


Figure 3-11 JTAG scan path (simplified)

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.

When the core module is attached to a motherboard, either directly or with other core modules, the TDI signal from the top core module is routed down through the HDRB connectors to the motherboard. The path is routed back up the stack through the processor core on each core module, before being returned to the Multi-ICE connector.

The motherboard detect signal **nMBDET** signal is used to control the routing for a standalone or an attached core module.

The PLDs and FPGAs are included in the scan chain when the core module is in configuration mode.

3.8.2 Debugging 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 3-10 on page 3-21). In this mode, the processor core and debuggable devices on other modules are accessible on the scan chain, as shown in Figure 3-11 on page 3-22.

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 3-10 on page 3-21*)*. This has the effect of pulling the **nCFGEN** signal LOW which illuminates the CFG LED (yellow) on each module in the stack and reroutes 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 may be unreliable if more than one core module is attached. The larger loads on the **TCK** and **TMS** lines may 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 which handle the transfer of data to the flash using JTAG.
- The PLDs are non-volatile devices which can be programmed directly by JTAG.

3.8.3 JTAG signals

Figure 3-12 shows the pinout of the Multi-ICE connector and Table 3-4 on page 3-25 provides a description of the JTAG signals.

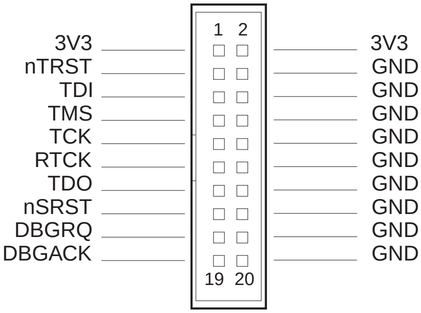


Figure 3-12 Multi-ICE connector pinout

Note

In the description in Table 3-4 on page 3-25, 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.

Table 3-4 JTAG signal description

Name	Description	Function
DBGRQ	Debug request (from JTAG equipment)	**DBGRQ** is a request for the processor core to enter the debug state. It is provided for compatibility with third-party JTAG equipment.
DBGACK	Debug acknowledge (to JTAG equipment)	**DBGACK** indicates to the debugger that the processor core has entered debug mode. It is provided for compatibility with third-party JTAG equipment.
DONE	FPGA configured	**DONE** is an open-collector signal which 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. In configuration mode all FPGAs and PLDs are connected to the scan chain so that they can be configured by the JTAG equipment.
nRTCKEN	Return TCK enable (from core module to motherboard)	**nRTCKEN** is an active LOW signal driven by any core module that requires **RTCK** to be routed back to the JTAG equipment. If **nRTCKEN** is HIGH, the motherboard drives **RTCK** LOW. If **nRTCKEN** is LOW, the motherboard drives the **TCK** signal back up the stack to the JTAG equipment.
nSRST	System reset (bidirectional)	**nSRST** is an active LOW open-collector signal which 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. The open collector **nRST** reset signal may be driven LOW by the reset controller on the core module to cause the motherboard to reset 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 ARM940T processor. It is asserted at power-up by each module, and can be driven by the JTAG equipment. This signal is also used in configuration mode to control the programming pin (**nPROG**) on FPGAs.

Table 3-4 JTAG signal description (continued)

Name	Description	Function
RTCK	Return TCK (to JTAG equipment)	Some devices sample **TCK** (for example a synthesizable core with only one clock), and this has the effect of delaying the time at which a component actually captures data. **RTCK** is a mechanism for returning the sampled clock to the JTAG equipment, so that the clock is not advanced until the synchronizing device captured the data. In *adaptive clocking mode*, Multi-ICE is required to detect 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 down-stream device. The **RTCK** signal on the module connectors HDRB returns **TCK** to the JTAG equipment. If there are no synchronizing components in the scan chain then it is unnecessary to use the **RTCK** signal and it is connected to ground on the motherboard.
TCK	Test clock (from JTAG equipment)	**TCK** synchronizes all JTAG transactions. **TCK** connects to all JTAG components in the scan chain. Series termination resistors are used to reduce reflections and maintain good signal integrity. **TCK** flows down the stack of modules and connects to each JTAG component. However, if there is a device in the scan chain that synchronizes **TCK** to some other clock, then all down-stream devices are connected to the **RTCK** signal on that component (see **RTCK**).
TDI	Test data in (from JTAG equipment)	**TDI** goes down the stack of modules to the motherboard and then back up the stack, labelled **TDO**, connecting to each component in the scan chain.
TDO	Test data out (to JTAG equipment)	**TDO** is the return path of the data input signal **TDI**. The module connectors HDRB have two pins labelled **TDI** and **TDO**. **TDI** refers to data flowing down the stack and **TDO** to data flowing up the stack. The JTAG components are connected in the return path so that the length of track driven by the last component in the chain is kept as short as possible.
TMS	Test mode select (from JTAG equipment)	**TMS** controls transitions in the tap controller state machine. **TMS** connects to all JTAG components in the scan chain as the signal flows down the module stack.

3.8.4 Debug communications interrupts

The ARM940T processor core incorporates EmbeddedICE hardware and provides a debug communications data register which 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 *ARM940T Technical Reference Manual*.

Interrupts can be used 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 *Interrupt registers* on page 4-19).

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, which includes a motherboard, you should refer to the user guide for the motherboard.

4.1.1 Core module memory map

The core module has a fixed memory map which maintains compatibility with other ARM modules and Integrator systems. Table 4-1 shows the memory map.

Table 4-1 ARM Integrator/CM940T 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 0x10FFFFFF	16MB	Core module registers
0	X	0x11000000 to 0xFFFFFFFF	4GB to 272MB	System bus address space
1	X	0x11000000 to 0xFFFFFFFF	4GB to 272MB	Abort

4.1.2 Boot ROM and SSRAM accesses

The boot ROM on the motherboard and the SSRAM on the core module share the same location within the Integrator memory map. Accesses to either the boot ROM or SSRAM are controlled by the **REMAP** bit and the motherboard detect signal (**nMBDET**) from the motherboard, as shown in Table 4-1 on page 4-2.

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 on the motherboard appears in the address range 0x0 to 0x3FFFF.

REMAP=1 The SSRAM appears in the 0x0 to 0x3FFFF address range.

Motherboard detect

The **nMBDET** signal operates as follows:

nMBDET=0 The core module is attached to a motherboard, and accesses in the address range 0x0 to 0x3FFFF to the boot ROM or SSRAM are controlled by the REMAP bit.

nMBDET=1 The core module is not attached, and accesses in the address range 0x0 to 0x3FFFF are routed to the SSRAM.

Note

The SSRAM is *local*, which means that it can only be read by the processor on the same core module. It provides fast local memory for the local processor and cannot be accessed by other system bus masters.

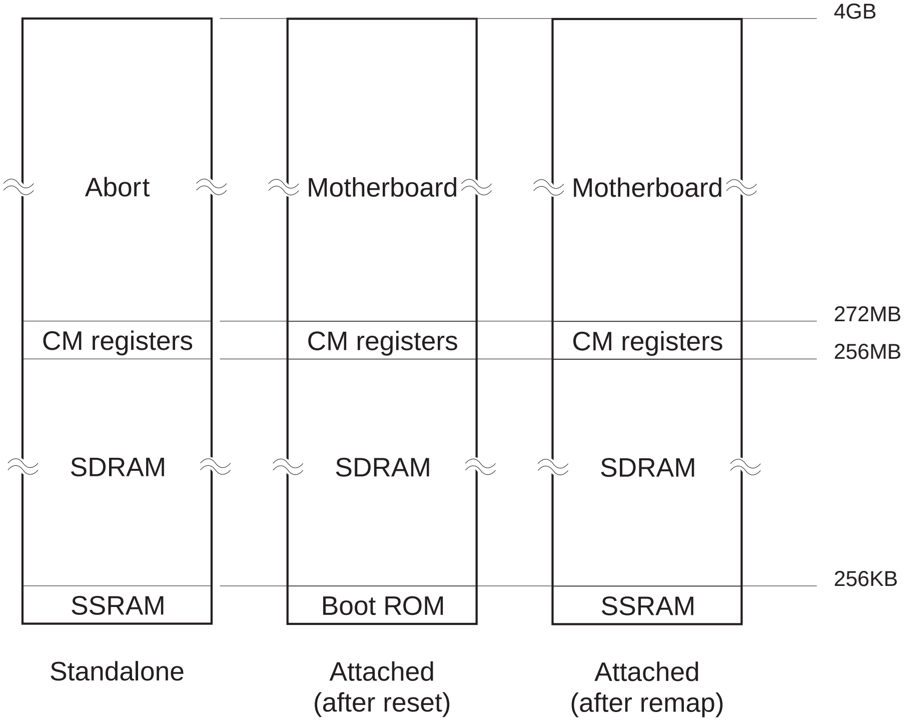


Figure 4-1 Effect of remap and motherboard detect

4.1.3 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.

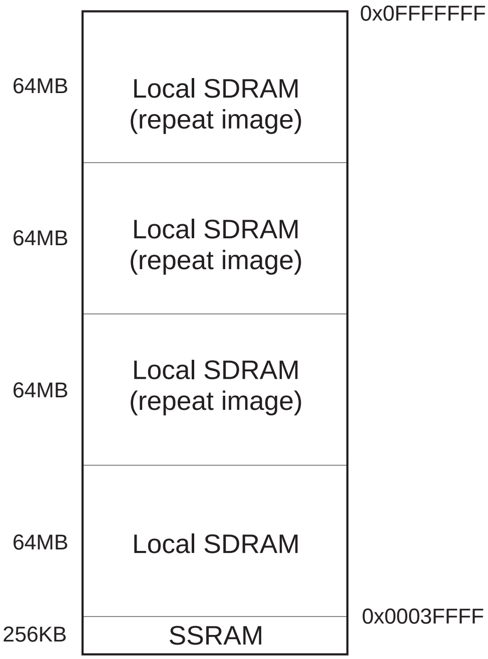

0x0FFFFFFF
64MB
Local SDRAM (repeat image)
64MB
Local SDRAM (repeat image)
64MB
Local SDRAM (repeat image)
64MB
Local SDRAM
0x0003FFFF
256KB
SSRAM

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

Local SDRAM

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 upon whether the core module is attached to a motherboard and upon 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 which fills the local SDRAM space, the first 256KB can only be accessed at the alias location (see *System bus accesses to SDRAM* below).

System bus accesses to SDRAM

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 *Core module ID* on page 2-6). Figure 4-3 shows the local and alias address of the SDRAM on four core modules.

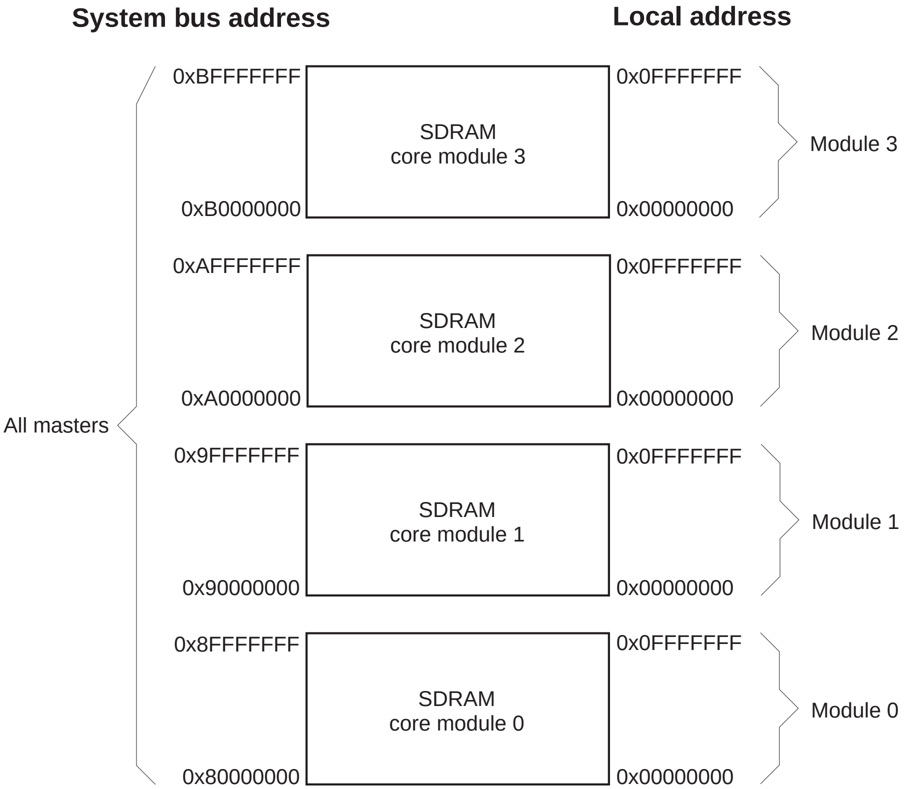


Figure 4-3 Core module local and alias addresses

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 *CM_STAT (0x10000010)* on page 4-12).

4.2 Exception vector mapping

The convention for ARM cores is to map the exception vectors to begin at address 0. However, the ARM940T core allows the vectors to be moved to 0xFFFF0000 by writing to the V bit in coprocessor 15 register 1 (CP15c1). The value of the V bit at reset is determined by the level on an external pin (**VINITHI**). To maintain compatibility across all cores, the default reset value maps the vector to begin at address 0 (see the *ARM940T Technical Reference Manual*).

When running applications with high vectors (at 0xFFFF0000 to 0xFFFFFFFF), software must write the correct value to the coprocessor register. However, Integrator motherboards have no physical memory at this location. This means that the MMU must be programmed to map an area of physical memory to this virtual address. When the core module is being used with a motherboard, an alternative is to implement an area of physical memory on an expansion card or logic module.

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. Bits marked as *reserved* in the following sections should be preserved using read-modify-write operations.

4.3.1 CM_ID (0x10000000)

The core module ID register (CM_ID) is a read-only register that identifies the board manufacturer, board type, and revision.

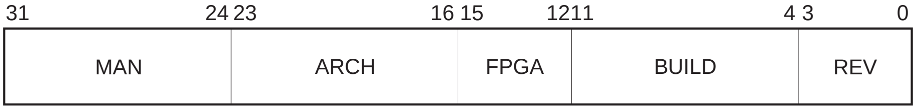


Table 4-3 describes the core module ID register bits.

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 (ARM internal use)
3:0	REV	Read	Revision: 0x0 = Rev A 0x1 = Rev B

4.3.2 CM_PROC (0x10000004)

The core module processor register (CM_PROC) is a read-only register that contains the value 0x00000000. This is provided for compatibility with processors that do not have a system control coprocessor (CP15). For the ARM940T, information about the processor can be obtained by reading coprocessor 15 register 0 (CP15 c0).

4.3.3 CM_OSC (0x10000008)

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 3-17). In addition, it provides information about processor bus mode setting.

31 25	24 23	22 20	19 12	11	10 8	7 0
Reserved	BMODE	L_OD	L_VDW	R	C_OD	C_VDW

Before writing to the CM_OSC register, you must 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	BMOD	Read	This field contains 00 which indicates that the processor bus mode is selected by writing to CM_CTRL register (see *CM_CTRL (0x1000000C)* on page 4-11).
22:20	L_OD	Read/write	Memory clock output divider: 000 = divide by 10 001 = divide by 2 (default) 010 = divide by 8 011 = divide by 4 100 = divide by 5 101 = divide by 7 110 = divide by 9 111 = divide by 6.
19:12	L_VDW	Read/write	Processor bus clock VCO divider word. Defines the binary value of the V[7:0] pins of the clock generator (V[8] is tied low). 00000100 = 6MHz (default with OD = 2).

Table 4-4 CM_OSC register (continued)

Bits	Name	Access	Function
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). 00000100 = 12MHz (default with OD = 2).

4.3.4 CM_CTRL (0x1000000C)

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>
<tr><th>31</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="2">Reserved</td><td>RESET</td><td>REMAP</td><td>nMBDET</td><td>LED</td></tr>
</table>

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

Table 4-5 CM_CTL register

Bits	Name	Access	Function
31:8	Reserved	Use read-modify-write to preserve value.	
7:4	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.
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 CM_STAT (0x10000010)

The core module status register (CM_STAT) is a read-only register that can be read to determine where in a multi-core module stack this core module is positioned.

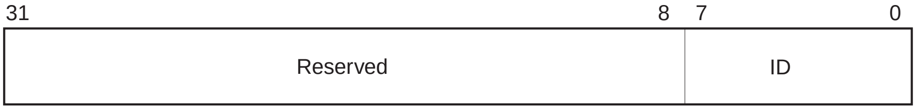


Table 4-6 describes the core module status register bits.

Table 4-6 CM_STAT register

Bit	Name	Access	Function
31:8	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 CM_LOCK (0x10000014)

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.

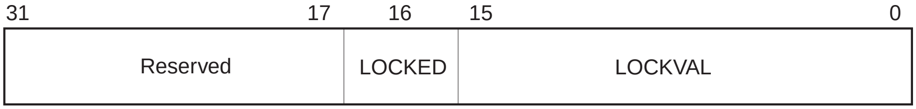


Table 4-7 describes the core module lock register bits.

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 CM_SDRAM (0x10000020)

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.

31 20	19 16	15 12	11 8	7 6	5	4 2	1 0
Reserved	NBANKS	NCOLS	NROWS	R	SPDOK	MEMSIZE	CASLAT

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. Should be set to the same value as byte 5 of SPD EEPROM.
15:12	NCOLS	Read/write	Number of SDRAM columns. Should be set to the same value as byte 4 of SPD EEPROM.
11:8	NROWS	Read/write	Number of SDRAM rows. Should be 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.

Table 4-8 CM_SDRAM register (continued)

Bits	Name	Access	Function
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.

Note

Before the SDRAM is used it is necessary to read the SPD memory and program the CM_SDRAM register with the parameters indicated in Table 4-8. If these values are not correctly set then SDRAM accesses may be slow or unreliable. See *CM_SPD (0x10000100 to 0x100001FC)* on page 4-16.

4.3.8 CM_SPD (0x10000100 to 0x100001FC)

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 3-6). The SPD memory contains 256 bytes of data, the most important of which are as shown in Table 4-9.

Table 4-9 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

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

The code segment shown in Example 4-1 on page 4-17 can be used to correctly setup and remap the SDRAM.

Example 4-1

```
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
```

4.4 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 4-10.

Table 4-10 Interrupt controller registers

Register Name	Address	Access	Size	Description
CM_IRQ_STAT	0x10000040	Read	3 bits	Core module IRQ status register
CM_IRQ_RSTAT	0x10000044	Read	3 bits	Core module IRQ raw status register
CM_IRQ_ENSET	0x10000048	Read/write	3 bits	Core module IRQ enable set register
CM_IRQ_ENCLR	0x1000004C	Write	3 bits	Core module IRQ enable clear register
CM_SOFT_INTSET	0x10000050	Read/write	1 bit	Core module software interrupt set
CM_SOFT_INTCLR	0x10000054	Write	1 bit	Core module software interrupt clear
CM_FIQ_STAT	0x10000060	Read	3 bits	Core module FIQ status register
CM_FIQ_RSTAT	0x10000064	Read	3 bits	Core module FIQ raw status register
CM_FIQ_ENSET	0x10000068	Read/write	3 bits	Core module FIQ enable set register
CM_FIQ_ENCLR	0x1000006C	Write	3 bits	Core module FIR enable clear register

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 using the enable set and enable clear locations.

The way that the interrupt enable, clear, and status bits function for each interrupt is illustrated in Figure 4-4 on page 4-20 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.

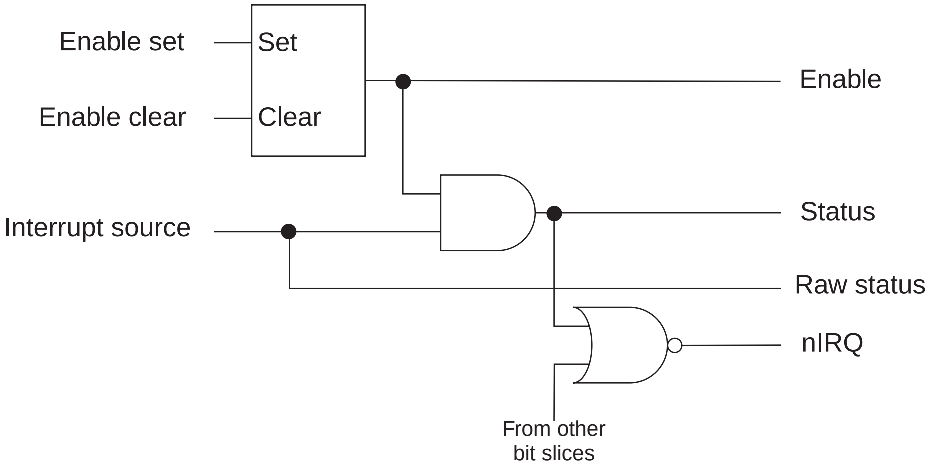


Figure 4-4 Interrupt control

4.4.1 CM_IRQ_STAT (0x10000040)/CM_FIQ_STAT (0x10000060)

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

4.4.2 CM_IRQ_RSTAT (0x10000044)/CM_FIQ_RSTAT (0x10000064)

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 CM_IRQ_ENSET (0x10000048)/CM_FIQ_ENSET (0x10000068)

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

- set bits in the enable register by writing to the ENSET location for the required IRQ or FIQ controller:

  1 = SET the bit.

  0 = leave the bit unchanged.
- read the current state of the enable bits from the ENSET location.

4.4.4 CM_IRQ_ENCLR(0x1000004C)/CM_FIQ_ENCLR (0x1000006C)

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

- clear bits in the enable register by writing to the ENCLR location for the required IRQ or FIQ controller:

  1 = CLEAR the bit.

  0 = leave the 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 4-11.

Table 4-11 IRQ and FIQ register bit assignment

Bit	Name	Function
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 CM_SOFT_INTSET (0x10000050)/CM_SOFT_INTCLT (0x10000054)

The core module interrupt controller provides a register for controlling and clearing software interrupts. This register is accessed using 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:

  1 = SET the software interrupt

  0 = 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:

  1 = CLEAR the software interrupt.

  0 = leave the software interrupt unchanged.

The bit assignment for the software interrupt register is shown in Table 4-12.

Table 4-12 IRQ register bit assignment

Bit	Name	Function
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*.

Appendix A
Signal Descriptions

This index 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.

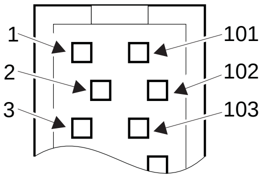
Note

For the Multi-ICE connector pinout and signal descriptions see *JTAG signals* on page 3-24.

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.

Pin numbers for 200-way plug, viewed from above board



Samtec TOLC series

1	A0		GND		101
2		GND		D0	102
3	A1		D1		103
4		A2		D2	104
5	A3		GND		105
6		GND		D3	106
7	A4		D4		107
8		A5		D5	108
9	A6		GND		109
10		GND		D6	110
11	A7		D7		111
12		A8		D8	112
13	A9		GND		113
14		GND		D9	114
15	A10		D10		115
16		A11		D11	116
17	A12		GND		117
18		GND		D12	118
19	A13		D13		119
20		A14		D14	120
21	A15		GND		121
22		GND		D15	122
23	A16		D16		123
24		A17		D17	124
25	A18		GND		125
26		GND		D18	126
27	A19		D19		127
28		A20		D20	128
29	A21		GND		129
30		GND		D21	130
31	A22		D22		131
32		A23		D23	132
33	A24		GND		133
34		GND		D24	134
35	A25		D25		135
36		A26		D26	136
37	A27		GND		137
38		GND		D27	138
39	A28		D28		139
40		A29		D29	140
41	A30		GND		141
42		GND		D30	142
43	A31		D31		143
44		B0		C0	144
45	B1		GND		145
46		GND		C1	146
47	B2		C2		147
48		B3		C3	148
49	B4		GND		149
50		GND		C4	150
51	B5		C5		151
52		B6		C6	152
53	B7		GND		153
54		GND		C7	154
55	B8		C8		155
56		B9		C9	156
57	B10		GND		157
58		GND		C10	158
59	B11		C11		159
60		B12		C12	160
61	B13		GND		161
62		GND		C13	162
63	B14		C14		163
64		B15		C15	164
65	B16		GND		165
66		GND		C16	166
67	B17		C17		167
68		B18		C18	168
69	B19		GND		169
70		GND		C19	170
71	B20		C20		171
72		B21		C21	172
73	B22		GND		173
74		GND		C22	174
75	B23		C23		175
76		B24		C24	176
77	B25		GND		177
78		GND		C25	178
79	B26		C26		179
80		B27		C27	180
81	B28		GND		181
82		GND		C28	182
83	B29		C29		183
84		B30		C30	184
85	B31		GND		185
86		GND		C31	186
87	5V		3V3		187
88		3V3		12V	188
89	5V		3V3		189
90		3V3		12V	190
91	5V		3V3		191
92		3V3		12V	192
93	5V		3V3		193
94		3V3		12V	194
95	5V		3V3		195
96		3V3		12V	196
97	5V		3V3		197
98		3V3		12V	198
99	5V		3V3		199
100		3V3		12V	200

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 Table A-1.

Table A-1 Bus bit assignment (for an AMBA ASB bus)

Pin label	Name (ASB)	Description
A[31:0]	System address bus	System address bus
B[31:0]	Not used	-
C[31:0]	System control bus	See remainder of table.
C[31:16]	Not used	-
C15	**BLOK**	Locked transaction
C14	**BLAST**	Last response
C13	**BERROR**	Error response
C12	**BWAIT**	Wait response
C11	**BWRITE**	Write transaction
C10	Not used	-
C[9:8]	**BPROT[1:0]**	Transaction protection type
C7	Not used	-
C[6:5]	**BURST[1:0]**	Transaction burst size
C4	Not used	-
C[3:2]	**BSIZE[1:0]**	Transaction width
C[1:0]	**BTRAN[1:0]**	Transaction type
D[31:0]	System data bus	System data bus

Note

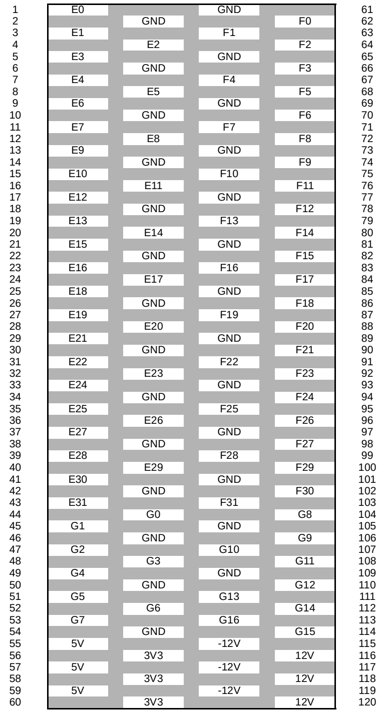
Table A-1 shows signal descriptions for an AMBA ASB bus implementation.

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.

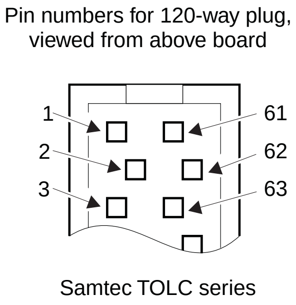


1	E0		GND		61
2		GND		F0	62
3	E1		F1		63
4		E2		F2	64
5	E3		GND		65
6		GND		F3	66
7	E4		F4		67
8		E5		F5	68
9	E6		GND		69
10		GND		F6	70
11	E7		F7		71
12		E8		F8	72
13	E9		GND		73
14		GND		F9	74
15	E10		F10		75
16		E11		F11	76
17	E12		GND		77
18		GND		F12	78
19	E13		F13		79
20		E14		F14	80
21	E15		GND		81
22		GND		F15	82
23	E16		F16		83
24		E17		F17	84
25	E18		GND		85
26		GND		F18	86
27	E19		F19		87
28		E20		F20	88
29	E21		GND		89
30		GND		F21	90
31	E22		F22		91
32		E23		F23	92
33	E24		GND		93
34		GND		F24	94
35	E25		F25		95
36		E26		F26	96
37	E27		GND		97
38		GND		F27	98
39	E28		F28		99
40		E29		F29	100
41	E30		GND		101
42		GND		F30	102
43	E31		F31		103
44		G0		G8	104
45	G1		GND		105
46		GND		G9	106
47	G2		G10		107
48		G3		G11	108
49	G4		GND		109
50		GND		G12	110
51	G5		G13		111
52		G6		G14	112
53	G7		G16		113
54		GND		G15	114
55	5V		-12V		115
56		3V3		12V	116
57	5V		-12V		117
58		3V3		12V	118
59	5V		-12V		119
60		3V3		12V	120

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.



1	E1		GND		61
2		GND		F0	62
3	E2		F1		63
4		E3		F2	64
5	E0		GND		65
6		GND		F3	66
7	E5		F4		67
8		E6		F5	68
9	E7		GND		69
10		GND		F6	70
11	E4		F7		71
12		E9		F8	72
13	E10		GND		73
14		GND		F9	74
15	E11		F10		75
16		E8		F11	76
17	E13		GND		77
18		GND		F12	78
19	E14		F13		79
20		E15		F14	80
21	E12		GND		81
22		GND		F15	82
23	E17		F16		83
24		E18		F17	84
25	E19		GND		85
26		GND		F18	86
27	E16		F19		87
28		E21		F20	88
29	E22		GND		89
30		GND		F21	90
31	E23		F22		91
32		E20		F23	92
33	E25		GND		93
34		GND		F24	94
35	E26		F25		95
36		E27		F26	96
37	E24		GND		97
38		GND		F27	98
39	E29		F28		99
40		E30		F29	100
41	E31		GND		101
42		GND		F30	102
43	E28		F31		103
44		G0		G8	104
45	G1		GND		105
46		GND		G9	106
47	G2		G10		107
48		G3		G11	108
49	G4		GND		109
50		GND		G12	110
51	G5		G13		111
52		G6		G14	112
53	G7		G16		113
54		GND		G15	114
55	5V		-12V		115
56		3V3		12V	116
57	5V		-12V		117
58		3V3		12V	118
59	5V		-12V		119
60		3V3		12V	120

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 3-15.

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[15:8] 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 baord as it passes up through the stack (see *JTAG signals* on page 3-24).

A.2.4 HDRB signal descriptions

Table A-3 describes the signals on the pins labeled E[31:0], F[31:0], and G[15:0].

Table A-3 HDRB signal description

Pin label	Name	Description
E[31:28]	**SYSCLK[3:0]**	System clock (ASB 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]	**Reserved**	-
E[7:4]	**AGNT[3:0]**	System bus grant
E[3:0]	**AREQ[3:0]**	System bus request
F[31:0]	**-**	Not connecetd
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 configurarion is complete (open collector)
G10	**RTCK**	Returned JTAG test clock
G9	**nSYSRST**	Buffered system reset
G8	**nTRST**	JTAG reset
G7	**TDO**	JTAG test data out
G6	**TDI**	JTAG test data in
G5	**TMS**	JTAG test mode select

Table A-3 HDRB signal description (continued)

Pin label	Name	Description
G4	**TCK**	JTAG test clock
G[3:1]	**MASTER[2:0]**	Master ID. Binary encoding of the master currently performing a transfer on the bus. Corresponds to the module ID and to the **AREQ** and **AGNT** line numbers.
G0	**nMBDET**	Motherboard detect pin

Note

Table A-3 shows signal descriptions for an AMBA ASB bus implementation.

Appendix B
Specifications

This appendix contains the specifications for the ARM Integrator/CM940T core module. It contains the following sections:

- *Electrical specification* on page B-2
- *Timing specification* on page B-3
- *Mechanical details* on page B-4.

B.1 Electrical specification

Table B-1 shows the core module electrical characteristics for the system bus interface.

The core module uses 3.3V and 5V source. 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.2 Timing specification

Table B-2 provides the operating timing characteristics for the system bus interface signals.

Table B-2 Core module timing (preliminary)

Symbol	Description	Min	Max	Units
F_{MAX}	Operating frequency	-	25	MHz
T_{CH}	Clock HIGH	19	-	ns
T_{CL}	Clock LOW	19	-	ns
T_{CO}	Clock to output – signals generated and sampled on same clock edge	-	16.0	ns
	Clock to output – signals generated and sampled on different clock edge	-	8.0	ns
T_{IC}	Input to clock – signals generated and sampled on same clock edge	-	8.0	ns
	Input to clock – signals generated and sampled on different clock edge	-	4.0	ns
T_{BPD}	Motherboard propagation delay (for guidance only)	-	1.0	ns
T_{SKEW}	Motherboard clock skew for guidance only)	-	1.0	ns

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.

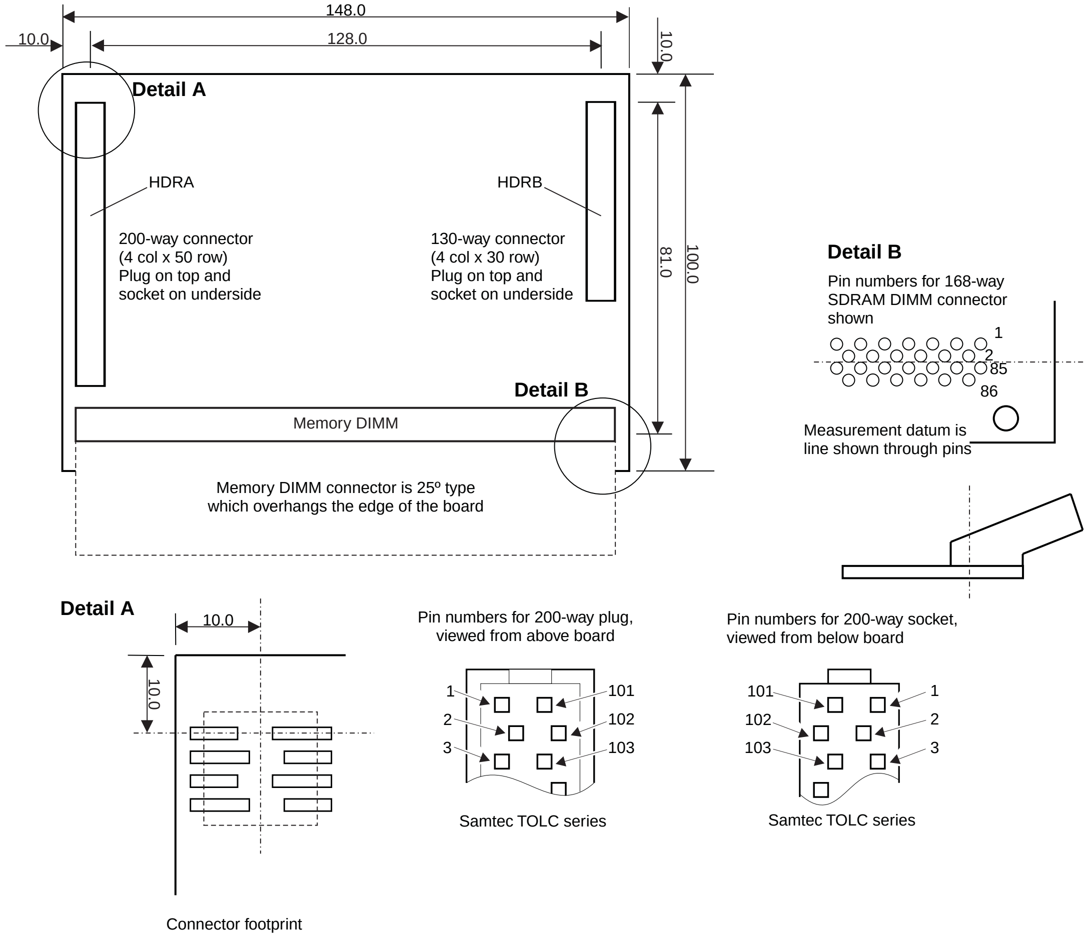


Figure B-1 Board outline

Index

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