Integrator™/CM10200E and CM10220E
HBI-0098

User Guide
Integrator/CM10200E and CM10220E
User Guide

Copyright © 2003 ARM Limited. All rights reserved.

Release Information

<table>
<thead>
<tr>
<th>Description</th>
<th>Issue</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 2003</td>
<td>A</td>
<td>First release</td>
</tr>
</tbody>
</table>

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks owned by ARM Limited, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the 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 in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of 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.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

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

Web Address

http://www.arm.com
Conformance Notices

This section contains conformance notices.

Federal Communications Commission Notice

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

CE Declaration of Conformity

The system should be powered down when not in use.

The Integrator generates, uses, and can radiate radio frequency energy and may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment causes harmful interference to radio or television reception, which can be determined by turning the equipment off or on, you are encouraged to try to correct the interference by one or more of the following measures:

• ensure attached cables do not lie across the card
• reorient the receiving antenna
• increase the distance between the equipment and the receiver
• connect the equipment into an outlet on a circuit different from that to which the receiver is connected
• consult the dealer or an experienced radio/TV technician for help

Note

It is recommended that wherever possible Shielded interface cables be used.
Chapter 4  Programmer’s Reference
4.1 Memory organization ................................................................. 4-2
4.2 Exception vector mapping ......................................................... 4-7
4.3 Registers .................................................................................. 4-8
4.4 SDRAM SPD memory ................................................................. 4-32

Chapter 5  Using Core Modules with an Integrator/AP
5.1 About the system architecture .................................................. 5-2
5.2 Module ID selection ................................................................. 5-4
5.3 Top-level memory map .............................................................. 5-5
5.4 System bus bridge ................................................................. 5-7
5.5 Interrupt control ................................................................. 5-14

Appendix A  Signal Descriptions
A.1 HDRA ................................................................................ A-2
A.2 HDRB ................................................................................ A-4
A.3 Trace connectors pinout ......................................................... A-10
A.4 Logic analyzer connectors .................................................. A-13

Appendix B  Specifications
B.1 Electrical specification ......................................................... B-2
B.2 Timing specification ............................................................ B-3
B.3 Mechanical details ............................................................... B-7
List of Tables

Integrator/CM10200E and CM10220E User Guide

Change history ........................................................................................................... ii
Table 1-1 LED functional summary ................................................................. 1-10
Table 1-2 Test point functions ................................................................. 1-11
Table 3-1 Controllable processor configuration signals ........................................... 3-4
Table 3-1 Fixed value processor configuration signals ........................................... 3-6
Table 3-2 CFGSEL[1:0] encoding ................................................................. 3-8
Table 3-3 Core and cache power enable ....................................................... 3-13
Table 3-4 Wakeup events ................................................................. 3-14
Table 3-5 Reset and power-management signal descriptions ...................... 3-15
Table 3-1 Internal clock selection ................................................................. 3-27
Table 3-2 Test clock output ................................................................. 3-27
Table 3-1 Clock control signal assignment ................................................................. 3-29
Table 3-1 Values for output divider ................................................................. 3-30
Table 3-2 JTAG signal description ................................................................. 3-36
Table 3-3 Link LK1 positions ................................................................. 3-40
Table 4-1 Core module memory map ................................................................. 4-2
Table 4-1 Private SDRAM memory map on power-up ........................................... 4-5
Table 4-2 Configured private SDRAM ................................................................. 4-6
Table 4-3 Core module registers ................................................................. 4-8
Table 4-1 ID Register, CM_ID bit assignment ................................................... 4-11
Table 4-2 Oscillator Register, CM_OSC bit assignment ........................................... 4-12
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 4-1</td>
<td>Control Register, CM_CTRL register bit assignment</td>
<td>4-13</td>
</tr>
<tr>
<td>Table 4-2</td>
<td>Status Register, CM_STAT register bit assignment</td>
<td>4-14</td>
</tr>
<tr>
<td>Table 4-3</td>
<td>Lock Register, CM_LOCK bit assignment</td>
<td>4-15</td>
</tr>
<tr>
<td>Table 4-4</td>
<td>Auxiliary Oscillator Register, CM_AUXOSC bit assignment</td>
<td>4-17</td>
</tr>
<tr>
<td>Table 4-5</td>
<td>SDRAM Status and Control Register, CM_SDRAM bit assignment</td>
<td>4-19</td>
</tr>
<tr>
<td>Table 4-6</td>
<td>Core Operating Voltage Register, CM_VOLTAGE_CTL0</td>
<td>4-21</td>
</tr>
<tr>
<td>Table 4-7</td>
<td>Cache Operating Voltage Register, CM_VOLTAGE_CTL1</td>
<td>4-21</td>
</tr>
<tr>
<td>Table 4-8</td>
<td>Core Power-Down Voltage Register, CM_VOLTAGE_CTL2</td>
<td>4-22</td>
</tr>
<tr>
<td>Table 4-9</td>
<td>Cache Power-Down Voltage Register, CM_VOLTAGE_CTL3</td>
<td>4-22</td>
</tr>
<tr>
<td>Table 4-10</td>
<td>Initialization Register, CM_INIT bit assignment</td>
<td>4-25</td>
</tr>
<tr>
<td>Table 4-11</td>
<td>Flag registers</td>
<td>4-26</td>
</tr>
<tr>
<td>Table 4-12</td>
<td>Interrupt control registers</td>
<td>4-28</td>
</tr>
<tr>
<td>Table 4-13</td>
<td>IRQ and FIQ register bit assignment</td>
<td>4-30</td>
</tr>
<tr>
<td>Table 4-14</td>
<td>IRQ register bit assignment</td>
<td>4-31</td>
</tr>
<tr>
<td>Table 5-1</td>
<td>SPD memory contents</td>
<td>4-32</td>
</tr>
<tr>
<td>Table 5-2</td>
<td>Core module address decode</td>
<td>5-4</td>
</tr>
<tr>
<td>Table 5-3</td>
<td>Core module interrupts</td>
<td>5-14</td>
</tr>
<tr>
<td>Table A-1</td>
<td>Bus bit assignment</td>
<td>A-3</td>
</tr>
<tr>
<td>Table A-2</td>
<td>Signal cross-connections (example)</td>
<td>A-7</td>
</tr>
<tr>
<td>Table A-3</td>
<td>HDRB signal description (AHB)</td>
<td>A-8</td>
</tr>
<tr>
<td>Table A-4</td>
<td>Trace connector one pinout</td>
<td>A-10</td>
</tr>
<tr>
<td>Table A-5</td>
<td>Trace connector two pinout</td>
<td>A-11</td>
</tr>
<tr>
<td>Table A-6</td>
<td>Connector HADDR pinout</td>
<td>A-14</td>
</tr>
<tr>
<td>Table A-7</td>
<td>Connector CONTROL pinout</td>
<td>A-15</td>
</tr>
<tr>
<td>Table A-8</td>
<td>Connector HDATA[31:0] pinout</td>
<td>A-16</td>
</tr>
<tr>
<td>Table A-9</td>
<td>HDATA[63:32] pinout</td>
<td>A-17</td>
</tr>
<tr>
<td>Table B-1</td>
<td>Core module electrical characteristics</td>
<td>B-2</td>
</tr>
<tr>
<td>Table B-2</td>
<td>Current requirements</td>
<td>B-2</td>
</tr>
<tr>
<td>Table B-3</td>
<td>Clock and reset parameters</td>
<td>B-3</td>
</tr>
<tr>
<td>Table B-4</td>
<td>AHB slave input parameters</td>
<td>B-4</td>
</tr>
<tr>
<td>Table B-5</td>
<td>AHB slave output parameters</td>
<td>B-4</td>
</tr>
<tr>
<td>Table B-6</td>
<td>Bus master input timing parameters</td>
<td>B-5</td>
</tr>
<tr>
<td>Table B-7</td>
<td>Bus master output timing parameters</td>
<td>B-5</td>
</tr>
<tr>
<td>Table B-8</td>
<td>AHB arbiter input parameters</td>
<td>B-5</td>
</tr>
<tr>
<td>Table B-9</td>
<td>AHB arbiter output parameters</td>
<td>B-6</td>
</tr>
</tbody>
</table>
List of Figures
Integrator/CM10200E and CM10220E User Guide

| Figure 1-1 | Integrator/CM10200E layout | 1-3 |
| Figure 1-2 | ARM Integrator/CM10200E block diagram | 1-5 |
| Figure 1-3 | Links and indicators | 1-9 |
| Figure 1-4 | Test points | 1-11 |
| Figure 2-1 | Power connector | 2-3 |
| Figure 2-2 | Multi-ICE connection to a core module | 2-4 |
| Figure 2-3 | Assembled Integrator system | 2-5 |
| Figure 3-1 | ARM10200E reference device block diagram | 3-3 |
| Figure 3-2 | Core module FPGA block diagram | 3-7 |
| Figure 3-3 | FPGA configuration | 3-8 |
| Figure 3-4 | Core module Reset and Power-Management Controller | 3-11 |
| Figure 3-5 | Voltage control registers | 3-16 |
| Figure 3-6 | Core voltage circuit | 3-19 |
| Figure 3-7 | Location of the voltage control resistors | 3-20 |
| Figure 3-8 | Core module clock generator | 3-25 |
| Figure 3-9 | Internal test chip clock control | 3-26 |
| Figure 3-10 | ARM PLLCLK divider control | 3-28 |
| Figure 3-11 | AUXCLK divider control | 3-29 |
| Figure 3-12 | JTAG connector, CFGEN link, and LED | 3-31 |
| Figure 3-13 | JTAG data path | 3-32 |
| Figure 3-14 | JTAG clock path | 3-33 |
List of Figures

Figure 3-15 Multi-ICE connector pinout ................................................................. 3-35
Figure 3-16 Trace connection .................................................................................. 3-38
Figure 4-1 Effect of remap ..................................................................................... 4-3
Figure 4-2 SDRAM repeat mapping for a 64MB DIMM ........................................... 4-5
Figure 4-3 ID Register, CM_ID .............................................................................. 4-11
Figure 4-4 Oscillator Register, CM_OSC ................................................................. 4-12
Figure 4-5 Control Register, CM_CTRL ................................................................. 4-13
Figure 4-6 Status Register, CM_STAT ................................................................. 4-14
Figure 4-7 Lock Register, CM_LOCK ................................................................. 4-15
Figure 4-8 Auxiliary Oscillator Register, CM_AUXOSC ....................................... 4-16
Figure 4-9 SDRAM Status and Control Register, CM_SDRAM ......................... 4-18
Figure 4-10 Voltage control registers .................................................................. 4-20
Figure 4-11 Initialization Register, CM_INIT ..................................................... 4-24
Figure 4-12 Interrupt control ............................................................................... 4-29
Figure 4-13 IRQ and FIQ bit assignments .............................................................. 4-30
Figure 5-1 FPGA functional diagram ................................................................... 5-2
Figure 5-2 Top-level memory map ....................................................................... 5-5
Figure 5-3 Core module local and alias addresses for SDRAM DIMM .............. 5-6
Figure 5-4 Processor writes to the system bus ...................................................... 5-8
Figure 5-5 Processor reads from the system bus .................................................. 5-9
Figure 5-6 System bus writes to SDRAM ............................................................ 5-10
Figure 5-7 System bus reads from SDRAM ......................................................... 5-11
Figure 5-8 Signal rotation on HDRB .................................................................... 5-12
Figure 5-9 Interrupt architecture (AP image) ...................................................... 5-15
Figure A-1 HDRA plug pin numbering ................................................................  A-2
Figure A-2 HDRB socket pin numbering .............................................................  A-5
Figure A-3 HDRB plug pin numbering .................................................................  A-6
Figure A-4 AMP Mictor connector ......................................................................  A-13
Figure B-1 Board outline .....................................................................................  B-7
Preface

This preface introduces the Integrator/CM10200E and CM10220E User Guide. It contains the following sections:

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

This document describes how to set up and use the ARM Integrator/CM10200E and CM10220E core modules.

Intended audience

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

Organization

This document is organized into the following chapters:

Chapter 1 **Introduction**
Read this chapter for an introduction to the core module. This chapter shows the physical layout of the core module and identifies the main components.

Chapter 2 **Getting Started**
Read this chapter for a description of how to set up and start using the core module. This chapter describes how to connect the core module and how to apply power.

Chapter 3 **Hardware Description**
Read this chapter for a description of the hardware architecture of the core module. This chapter describes the clocks, resets, and debug hardware provided by the core module.

Chapter 4 **Programmer’s Reference**
Read this chapter for a description of the core module memory map and registers.

Chapter 5 **Using Core Modules with an Integrator/AP**
Read this chapter for a description of how to use the core module with an Integrator/AP motherboard.

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:

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

*bold*  Highlights interface elements, such as menu names. Denotes ARM processor signal names. Also used for terms in descriptive lists, where appropriate.

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

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

*monospace italic*  Denotes arguments to commands and functions where the argument is to be replaced by a specific value.

*monospace bold*  Denotes language keywords when used outside example code.

Further reading

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

ARM publications

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

- *ARM10200E Test Chip Implementation Guide* (ARM DXI 0120)
- *ARM Integrator/AP User Guide* (ARM DUI 0098)
Other publications

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

- *MicroClock OSCaR ICS525 User Configurable Clock Data Sheet* (MDS525), MicroClock Division of ICS, San Jose, CA.
Feedback

ARM Limited welcomes feedback both on the ARM Integrator/CM10200E and CM10220E core modules and on the documentation.

Feedback on the ARM Integrator/CM10200E and CM10220E

If you have any comments or suggestions about this product, contact your supplier giving:
• the product name
• an explanation of your comments.

Feedback on this document

If you have any comments about this document, send email to errata@arm.com giving:
• the document title
• the document number
• the page number(s) to which your comments refer
• an explanation of your comments.

General suggestions for additions and improvements are also welcome.
Chapter 1
Introduction

This chapter introduces the ARM Integrator/CM10200E and CM10220E core modules. It contains the following sections:

- *About the core module* on page 1-2
- *Core module architecture* on page 1-4
- *Links and indicators* on page 1-9
- *Test points* on page 1-11
- *Precautions* on page 1-12.
1.1 About the core module

The Integrator/CM10200E core module provides a development system that you can use to develop products around the ARM10200E™ reference device.

You can use the core module in different ways:

- with a power supply and a connection to a Multi-ICE unit, the core module provides a basic development system
- by mounting the core module onto an Integrator motherboard or other Integrator modules, you can build a realistic emulation of the system being developed.

The core module can be used as:
- a standalone development system
- mounted onto an ARM Integrator/AP motherboard
- mounted onto an ARM Integrator logic module without a motherboard, with the logic module providing the system controller functions of a motherboard
- integrated into a third-party development or ASIC prototyping system.

Figure 1-1 on page 1-3 shows the layout of the ARM Integrator/CM10200E.

—— Note ————

The CM10200E and CM10220E core modules are identical except that:

- the CM10220E module uses the ARM1022™ core that has 16KB of instruction cache and 16KB of data cache
- the CM10200E module uses the ARM1020™ core that has 32KB of instruction cache and 32KB of data cache.

Unless otherwise stated, all references to the CM10200E apply also to the CM10220E core module.
Figure 1-1 Integrator/CM10200E layout
1.2 Core module architecture

The major components on the core module are:

- ARM1020E reference device equipped with:
  - ARM1020E processor
  - Embedded Trace Module (ETM)
  - Vector Floating Point (VFP) coprocessor
  - Synchronous Dynamic Random Access Memory (SDRAM) controller
  - power manager.
- Core module Field Programmable Gate-Array (FPGA) that implements:
  - SDRAM controller
  - system bus bridge
  - Synchronous Static Random Access Memory (SSRAM) controller
  - reset and power-management controller
  - interrupt controller
  - status, configuration, interrupt, and power registers.
- 64MB of private SDRAM controlled by the test chip SDRAM controller
- 2MB SDRAM
- up to 256MB of SDRAM (optional) plugged into the Dual In-line Memory Module (DIMM) socket referred to as SDRAM DIMM in this document
- clock generators
- system bus connectors
- logic analyzer connectors for the processor Advanced High-performance Bus (AHB) and Trace port
- programmable power supplies.

The architecture is further described in:

- System architecture on page 1-5
- ARM processor test chip on page 1-5
- Core module FPGA on page 1-5
- Volatile memory on page 1-6
- Clock generator on page 1-7
- Multi-ICE® connector on page 1-8
- Test chip power supplies on page 1-8
- Logic analyzer and trace connectors on page 1-8.
1.2.1 System architecture

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

![ARM Integrator/CM10200E block diagram](image)

1.2.2 ARM processor test chip

The Integrator/CM10200E is fitted with an ARM10200E test chip. For a brief description of this chip, see ARM microprocessor test chip on page 3-2.

--- Note ---

The Integrator/CM10220E is fitted with an ARM10220E test chip.

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 DIMM controller is implemented in the FPGA. This provides support for DIMMs with a capacity of between 16 and 256MB. See SDRAM DIMM controller on page 3-9.

**Reset and Power-Management Controller**

The Reset and Power-Management Controller initializes the core and FPGA as a result of a reset and manages the core and cache voltages when the core enters or exits power-down mode. The core module can be reset from five sources:

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

For information about the Reset and Power-Management Controller, see Reset and power-management controller on page 3-10.

System bus bridge

The system bus bridge provides an AMBA interface between the memory bus on the core module and the system bus. It enables the processor to access resources on the motherboard and on other modules. It also enables other masters to access the core module SDRAM DIMM, see System bus bridge on page 5-7.

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:

• processor status and configuration
• the position of the core module in a multi-module stack
• SDRAM DIMM size, address configuration, and CAS latency setup
• core module clocks setup
• interrupt control for the processor debug communications channel
• the setting and reading of power-down and operating voltages for:
  — core
  — cache.

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, private SDRAM for the processor, 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 DIMM controller is implemented in the core module controller FPGA.

The private SDRAM controller is directly implemented on the ARM chip.

The SDRAM DIMM can be accessed:
- by the local processor
- by processors on other core modules
- by other system bus masters.

The SSRAM and the private SDRAM can only be accessed by the local processor.

### 1.2.5 Clock generator

The core module generates three clock signals:

- **CLKREF24MHZ**  A fixed frequency 24MHz signal that can be used by the FPGA to generate real-time delays.
- **ARM_PLLCLKIN**  A programmable frequency clock for the ARM test chip. This clock is used by the test chip to generate GCLK and HCLK.
- **AUXCLK**  A programmable frequency clock reserved for future use.

The programmable clocks are supplied by three clock generator chips. Their frequencies are set using the oscillator control registers in the FPGA. A reference clock is supplied to the clock generators and to the FPGA, see *Clock generators* on page 3-25.

Note
---

If the core module is used with a motherboard, communication between the core module and the motherboard is over the system bus.

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

The system bus clock is provided by the Integrator/AP motherboard.
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 3-31.

1.2.7 Test chip power supplies

Five power supplies are dedicated to the ARM test chip. Three of them have their voltage values fixed by two on-board resistors for each supply:

- $V_{DDPLL}$ is the power supply for the Phase Locked Loop (PLL)
- $V_{DD2}$ is the power supply for partner supplied logic
- $V_{DDPM}$ is the power supply for the Power Management Unit (PMU)

The remaining two power supplies have their value set by a register in the FPGA. You can read the voltages of these power supplies:

- $V_{DDCACHE}$ is the power supply for the cache
- $V_{DDCORE}$ is the power supply for the core.

Reading of the core and cache voltages is achieved by the use of an on-board Analog-to-Digital Converter (ADC).

--- Caution ---

Before you modify the core or cache voltage, you must ensure that the test chip you are using enables voltages to be modified. This facility is not present on some test chips.

1.2.8 Logic analyzer and trace connectors

Four logic analyzer connectors enable you to gain access to the ARM data, address, and control signals.

Similarly, two logic analyzer trace connectors connect to the test chip Embedded Trace Macrocell (ETM) signals. This enables you to connect a Trace Port Analyzer (TPA) to the board in ETM demultiplexed mode.

See Trace connectors pinout on page A-10 for connection information.
1.3 **Links and indicators**

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

![Figure 1-3 Links and indicators](image)

1.3.1 **CFGEN link**

The core module has only one link, configuration enable, marked CFGEN. This is left open during normal operation. It is only fitted when downloading new FPGA or PLD configuration information, see *Configuration mode* on page 3-34.
1.3.2 LED indicators

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Color</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MISC</td>
<td>Green</td>
<td>This LED is controlled using the control register (see <em>Control Register, CM_CTRL</em> on page 4-13)</td>
</tr>
<tr>
<td>DONE</td>
<td>Green</td>
<td>This LED illuminates when the FPGA has successfully loaded its configuration information following power-on</td>
</tr>
<tr>
<td>PWR</td>
<td>Green</td>
<td>This LED illuminates to indicate that a 3.3V supply is present</td>
</tr>
<tr>
<td>CFGEN</td>
<td>Orange</td>
<td>This LED illuminates to indicate that the CONFIG link is fitted</td>
</tr>
</tbody>
</table>
1.4 Test points

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

![Figure 1-4 Test points](image)

The functions of these test points are summarized in Table 1-2. For information about setting the frequency of the core clock and auxiliary clock, see Clock generators on page 3-25.

<table>
<thead>
<tr>
<th>Test point</th>
<th>Signal</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP6</td>
<td>HCLK</td>
<td>Bus HCLK generated by the ARM core</td>
</tr>
<tr>
<td>TP7</td>
<td>AUXCLK</td>
<td>Auxiliary clock</td>
</tr>
<tr>
<td>TP8</td>
<td>FPGA_PLLCLK</td>
<td>Clock routed through the FPGA to the ARM test chip</td>
</tr>
<tr>
<td>TP9</td>
<td>CLKREF24MHZ</td>
<td>Reference clock (24MHz)</td>
</tr>
</tbody>
</table>
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 in a laboratory or engineering development environment. It is supplied without an enclosure that leaves the board sensitive to electrostatic discharges and enables electromagnetic emissions.

——— Caution ————

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

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

——— Caution ————

Do not use the board near equipment that is:

• sensitive to electromagnetic emissions (such as medical equipment)
• a transmitter of electromagnetic emissions.
Chapter 2
Getting Started

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

- Setting up a standalone core module on page 2-2
- Attaching the core module to a motherboard on page 2-5.
2.1 Setting up a standalone core module

To set up the core module as a standalone development system:
1. Optionally, fit an SDRAM DIMM.
2. Supply power.
3. Connect Multi-ICE.

2.1.1 Fitting an SDRAM DIMM

You must fit the following type of SDRAM module:
- PC66, PC100, or PC133- compliant 168-pin DIMM
- unbuffered
- 3.3V
- 16MB, 32MB, 64MB, 128MB or 256MB.

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

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

See SDRAM SPD memory on page 4-32 and SDRAM Status and Control Register, CM_SDRAM on page 4-18 for information about how to initialize and control the SDRAM DIMM.

2.1.2 Using the core module without an external SDRAM DIMM

You can operate the core module without an external SDRAM DIMM because it has 2MB of SSRAM and 64MB of private SDRAM permanently fitted. When using ADW or AXD, you can adjust the $top_of_memory internal variable from its default value to 0x200000 (the top of the SSRAM).

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

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

![Power connector diagram]

**Note**

Do not connect this 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 on page 2-4.
Caution

Because the core module does not provide nonvolatile 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 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 enable 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.

--- Note ---

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

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

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

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

2.2.2 Powering the assembled Integrator development system

Power the assembled Integrator development system by connecting an external power supply to the motherboard.

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:

- ARM microprocessor test chip on page 3-2
- Core module FPGA on page 3-7
- SDRAM DIMM controller on page 3-9
- Clock generators on page 3-25
- Multi-ICE support on page 3-31
- Embedded Trace support on page 3-38
- Stacking options on page 3-40.
3.1 ARM microprocessor test chip

The test chip is described in:
- ARM10200E reference device overview
- Test chip configuration control on page 3-4
- Fixed value test chip configuration control on page 3-6

Note

The CM10200E and CM10220E core modules are identical except that:
- the CM10220E module uses the ARM1022 core that has 16KB of instruction cache and 16KB of data cache
- the CM10200E module uses the ARM1020 core that has 32KB of instruction cache and 32KB of data cache.

All references to the CM10200E apply also to the CM10220E core module.

3.1.1 ARM10200E reference device overview

The ARM10200E incorporates the following features:

ARM1020E The ARM1020E is a member of the ARM10 Thumb® family. The ARM1020E macrocell is a 32-bit cached processor with ARMv5TE architecture that supports the ARM and Thumb instruction sets and includes EmbeddedICE-RT logic and JTAG software debug features.

ETM10 The Embedded Trace Module (ETM) provides signals for off-chip trace. These are high-speed signals that require careful handling.

VFP10 This high-performance, low-power Vector Floating-Point (VFP) coprocessor is the first implementation of the VFPv2 vector floating-point architecture.

Clock generator

The clock generator enables you to generate the ARM10200E reference device clocks from the external reference clock ARM_PLLCLKIN.

Power manager

All power manager programming and control is handled with a double-ended handshake protocol through the System Control Coprocessor (SCC), CP15 register 15.
AHB  The ARM1020E processor uses separate AMBA High-speed Bus (AHB) masters for instructions and data to maximize the ability to fetch and execute instructions in parallel with a data cache miss.

AHB bridge  The AHB bridge supports the implementation of an AHB system with on-chip masters and slaves and off-chip slaves only. The only function of the AHB bridge is to forward requests from on-chip masters and return responses and data from off-chip slaves.

SDRAM Controller

The SDRAM controller operates at a submultiple of the processor clock frequency, up to a maximum of 100MHz. The SDRAM interface is 64 bits wide and provides the fast memory access required for ARM1020E reference device applications. The SDRAM controller uses the private SDRAM on the Integrator/CM10200E board.

AHB arbiter  The AHB arbiter determines priority when there are clashes between transfer requests issued by the instruction Bus Interface Unit (BIU) and the data BIU.

Figure 3-1 shows the main blocks of the ARM1020E reference device.
3.1.2 Test chip configuration control

The ARM10200E reference device is configurable and has several input signals that control the configuration of the ARM test chip. In a typical product, core configuration is static and these signals are tied HIGH or LOW as appropriate. However, because the Integrator is a development platform, it offers you the facility to program the levels of these signals experimentally by using the CM_INIT Register. The configuration signals driven by CM_INIT are described in Table 3-1.

--- Note ---

The configuration inputs are controlled by the CM_INIT Register in the FPGA. CM_INIT bit-states are retained during manual resets. During Power-On Resets (POR), before the FPGA is configured, the bit-states are set to default values by pull-down and pull-up resistors to the values shown in the POR default column.

The values of VINITHI and BIGENDINIT are only sampled at startup. Changing the values has no effect until the next manual reset.

---

### Table 3-1 Controllable processor configuration signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Function</th>
<th>POR default</th>
</tr>
</thead>
<tbody>
<tr>
<td>VINITHI</td>
<td>Use this signal to initialize the vector base address:</td>
<td>LOW</td>
</tr>
<tr>
<td></td>
<td>LOW = 0x00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HIGH = 0xFFFF0000.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Integrator system memory map allocates the high vector address location as logic module expansion space. This means that there might not be any physical memory at this location.</td>
<td></td>
</tr>
<tr>
<td>BIGENDINIT</td>
<td>Use this signal to initialize the endianness:</td>
<td>LOW</td>
</tr>
<tr>
<td></td>
<td>LOW = little-endian</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HIGH = big-endian.</td>
<td></td>
</tr>
</tbody>
</table>

--- Note ---

For details on configuring the internal test chip clocks, see *Clock generation functional overview* on page 3-25.
Changing the processor configuration

To change the configuration of the processor:

1. Program the appropriate values in the CM_INIT Register, see Initialization Register, CM_INIT bit assignment on page 4-25.

2. Reset the core module, but do not power-cycle, by either:
   • pressing the reset button
   • performing a software reset by writing a 1 to bit 3 of the CM_CTRL Register, see Control Register, CM_CTRL register bit assignment on page 4-13.

Restoring the default configuration

To restore the default processor configuration, power-cycle the core module.
3.1.3 Fixed value test chip configuration control

You cannot change the configuration inputs shown in Table 3-1.

Table 3-1 Fixed value processor configuration signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TESTMODE</td>
<td>This signal controls the <em>Design For Test</em> (DFT) functionality:</td>
</tr>
<tr>
<td></td>
<td>LOW = disable</td>
</tr>
<tr>
<td></td>
<td>HIGH = enable.</td>
</tr>
<tr>
<td></td>
<td>This signal is tied LOW.</td>
</tr>
<tr>
<td>CLKTESTCTL[3:0]</td>
<td>These test control input signals select clock generator internal clocks</td>
</tr>
<tr>
<td></td>
<td>for viewing on <em>CLKTESTOUT</em> during testing and select the clock divider</td>
</tr>
<tr>
<td></td>
<td>source (PLL output or reference frequency). These signals are controlled</td>
</tr>
<tr>
<td></td>
<td>by the PLLCLKTST bits in CM_INIT. For normal operation, the PLL is not</td>
</tr>
<tr>
<td></td>
<td>bypassed and the <em>CLKTESTOUT</em> signal is LOW.</td>
</tr>
</tbody>
</table>


3.2 Core module FPGA

The core module FPGA contains six main functional blocks:

- *SDRAM DIMM controller* on page 3-9
- *Reset and power-management controller* on page 3-10
- *System bus bridge* on page 5-7
- *Debug communications interrupts* on page 3-24
- *Registers* on page 4-8
- SSRAM controller.

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 3-2 illustrates the function of the core module FPGA and shows how it connects to the other devices in the system.

![Figure 3-2 Core module FPGA block diagram](image-url)
At power-up the FPGA loads its configuration data from a flash memory device. Parallel data from the flash is streamed by the Programmable Logic Device (PLD) into the configuration ports of the FPGA. Figure 3-3 shows the FPGA configuration mechanism.

The config flash can contain multiple images that enable the FPGA to be configured to support different types of system bus. 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 3-2.

--- Note ---

For the current release, only one image is provided. This image supports the AHB system bus with an Integrator/AP.

--- Table 3-2 CFGSEL[1:0] encoding ---

<table>
<thead>
<tr>
<th>CFGSEL[1:0]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Little-endian ASB, Integrator/AP (not supported)</td>
</tr>
<tr>
<td>01</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>Little-endian AHB, Integrator/AP</td>
</tr>
<tr>
<td>11</td>
<td>AHB Lite, Integrator/CP (not supported)</td>
</tr>
</tbody>
</table>
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 3-31.

3.2.1 SDRAM DIMM controller

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

SDRAM operating mode

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

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

You can program the CAS latency and memory size using the SDRAM control register (CM_SDRAM) at address 0x10000020, see SDRAM Status and Control Register, CM_SDRAM on page 4-18.

--- Note ---

Before the SDRAM is used, it is necessary to read the SPD memory and program the CM_SDRAM register with the parameters indicated in Table 4-3 on page 4-19. If these values are not correctly set then SDRAM accesses might be slow or unreliable.

---

Access arbitration

The SDRAM controller provides two ports to support reads and writes by the local processor core and by masters on the system bus. The SDRAM controller uses an alternating priority scheme to ensure that the processor core and other system bus masters have equal access, see System bus bridge on page 5-7.

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) in 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 \texttt{0x10000100}–\texttt{0x100001FC}, see \textit{SDRAM SPD memory} on page 4-32.

Write accesses to the SPD EEPROM are not supported.

3.2.2 SSRAM Controller

The SSRAM controller is implemented in the FPGA to achieve single-cycle (zero-wait state) access to the SSRAM.

3.2.3 Reset and power-management controller

The Reset and Power-Management Controller is described in:

- \textit{Reset}
- \textit{Power management} on page 3-11
- \textit{Reset and power-management signals} on page 3-15.

Reset

The core module FPGA incorporates a reset and power-management 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 and logic modules
- Multi-ICE
- software.

Figure 3-4 on page 3-11 shows the architecture of the reset and power-management controller.
Power management

A Power Management Unit (PMU) is located beside the ARM processor in the test chip. The ARM PMU provides methods of saving power by generating two signals that are used to control the core and cache voltages:

- **COREPWREN**
- **CACHEPWREN**.

The FPGA Power Management Controller uses these two signals and values that are stored in the voltage configuration registers (see Core Module Voltage Configuration Registers, CM_VOLTAGE_CTL0-3 on page 4-20) to change the core and cache voltages. To change a voltage, the FPGA writes to a Digital to Analog Converter (DAC). There are separate DACs for the core and cache voltages.
There are four power states:

**Run**  The processor is executing instructions.

**Standby**  The processor clocks are stopped. Both core and cache voltage are unchanged, but power consumption is reduced.

**Dormant**  The core voltage is reduced but the cache voltage remains unchanged.

**Shutdown**  Both the core and cache voltages are reduced.

The power device states are entered through writes to the CP15 coprocessor, to the Power Manager Transmit Data Register (PMTDR). For details of the PMTDR, see the ARM10200E Test Chip Implementation Guide.

--- Caution ---

Before entering the dormant or shutdown state, software must save the current state in external memory. State saving and restoring steps are not shown in the examples. If you change the power mode, you must ensure that there are no pending accesses of the coprocessor registers.

---

To enter a power down state, write to the PMTDR Register:

0x00  Enter the shutdown state
0x40  Enter the dormant state
0x80  Enter the standby state

An example of entering dormant state from the operating state by writing 0x40 to the PMTDR Register is provided in Example 3-1.

**Example 3-1 Entering dormant state from the operating state**

```
MOV r1, #0x40
MCR p15, 0, r1, c15, c14, 1
```

After a power mode command is issued to CP15 to enter dormant or shutdown state:

- the **COREPWREN** and **CACHEPWREN** signals change state
- the FPGA modifies the voltages according to the values stored in the Voltage Configuration Registers
- the MISC LED flashes
- the system waits for a wake-up event.
**Note**

All of the values stored in the voltage configuration registers that correspond to the core and cache operating and power-down voltages default to the same value on power-on reset. This is the nominal core voltage. This means that writing to CP15 to enter a power-down state has no effect. However, the system then waits for a wake-up event before entering operating state again.

The core operating state, resultant signal combinations, values programmed to the core and cache voltages are listed in Table 3-3.

**Table 3-3 Core and cache power enable**

<table>
<thead>
<tr>
<th>Operating state</th>
<th>COREPWREN</th>
<th>CACHEPWREN</th>
<th>Core voltage</th>
<th>Cache voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal or standby</td>
<td>0</td>
<td>0</td>
<td>CM_VOLTAGE_CTL0[7:0]</td>
<td>CM_VOLTAGE_CTL1[7:0]</td>
</tr>
<tr>
<td>Dormant</td>
<td>1</td>
<td>0</td>
<td>CM_VOLTAGE_CTL2[7:0]</td>
<td>CM_VOLTAGE_CTL1[7:0]</td>
</tr>
<tr>
<td>Shutdown</td>
<td>1</td>
<td>1</td>
<td>CM_VOLTAGE_CTL2[7:0]</td>
<td>CM_VOLTAGE_CTL3[7:0]</td>
</tr>
<tr>
<td>Not permitted</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

As shown in Table 3-4 on page 3-14, the preferred way to exit standby is by an interrupt or debug request. The preferred way to exit dormant or shutdown state is by pressing the core module pushbutton. This wakes the processor without losing context information.
From all states, the processor enters run state after a full system reset, but previous context information in modules might be lost.

### Table 3-4 Wakeup events

<table>
<thead>
<tr>
<th>Wakeup source</th>
<th>Effect on standby state</th>
<th>Effect on dormant or shutdown state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt or debug request</td>
<td>Wakeup</td>
<td>No effect</td>
</tr>
<tr>
<td>Core module pushbutton</td>
<td>System reset</td>
<td>The FPGA restores the core and cache voltages and resets the core. Other modules in the Integrator system are not reset. The MISC LED stops flashing.</td>
</tr>
<tr>
<td>Reset from other core module, logic module or motherboard</td>
<td>System reset</td>
<td>System reset</td>
</tr>
</tbody>
</table>

After a wake-up event, the FPGA performs the following actions:
- the voltages are restored
- if the previous state was dormant or shutdown, the MISC LED stops flashing.

If a full reset has occurred, the previous power-mode state can be read from the coprocessor registers. Example 3-2 shows how to read the previous power mode.

#### Example 3-2 Detecting the power state at reset

```assembly
reset
; place other code here
MRC CP15, 0, r0, c15, c14, 0
TST R0, #R_flag               ; wait for incoming data from coprocessor
BNE reset
MRC CP15, 0, r0, c15, c14, 1  ; read in previous state
TST R0, 0xC0                 ; was previous state RUN
BEQ last_state_run
TST R0, 0x80                 ; was previous state STANDBY
BEQ last_state_standby
TST R0, 0x40                 ; was previous state DORMANT
BEQ last_state_dormant
; execute default power-on reset code here
```

If the previous context information was stored in memory, the context can be recovered after the reset by user-supplied code.
## Reset and power-management signals

Table 3-5 describes the external reset and power-management signals.

### Table 3-5 Reset and power-management signal descriptions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACHEPWREN</td>
<td>Input</td>
<td>Power-down core request to the FPGA to turn down the cache voltage.</td>
</tr>
<tr>
<td>COREPWREN</td>
<td>Input</td>
<td>Power-down core request to the FPGA to turn down the core voltage.</td>
</tr>
<tr>
<td>GLOBAL_DONE</td>
<td>Input</td>
<td>FPGA configured. The <code>GLOBAL_DONE</code> signal is an open-collector signal generated by all FPGAs when they have completed their configuration. The <code>FPGADONE</code> signal is routed round the system through the HDRB connectors to the inputs of all other FPGAs in the system. The signal <code>nSRST</code> is held asserted until <code>GLOBAL_DONE</code> is driven LOW.</td>
</tr>
<tr>
<td>nMBDET</td>
<td>Input</td>
<td>Motherboard detect. The <code>nMBDET</code> signal is pulled LOW when the core module is attached to a motherboard and HIGH when the core module is used standalone. When <code>nMBDET</code> is LOW, <code>nSYSRST</code> is used to generate the <code>FPGA_HRESETn</code> signal and the reset for the FPGA power management control. When <code>nMBDET</code> is HIGH, <code>nSRST</code> is used to generate the <code>FPGA_HRESETn</code> signal and the reset for the FPGA power management control.</td>
</tr>
<tr>
<td>nPORES</td>
<td>Output</td>
<td>Power-on reset. This signal provides a hard reset for the ARM10200E reference device on power-up.</td>
</tr>
<tr>
<td>nSFRES</td>
<td>Output</td>
<td>Soft reset. This signal provides a soft reset for the ARM10200E reference device.</td>
</tr>
<tr>
<td>nSRST</td>
<td>I/O</td>
<td>System reset. The <code>nSRST</code> output signal is driven LOW by the core module FPGA when one of the following is asserted: * PBRST * SWRST * nDONE. The <code>nSRST</code> input signal can be driven LOW by Multi-ICE. If there is no motherboard present, the <code>nSRST</code> signal is synchronized to the processor bus clock to generate the <code>FPGA_HRESETn</code> signal and the reset for the FPGA power management control.</td>
</tr>
</tbody>
</table>
3.2.4 Control of core and cache voltages

The core module provides facilities to control the core and cache voltages of the test chip. How to use these facilities is described in:

- Reading the voltages on page 3-17
- Modifying the voltages on page 3-18
- Voltage control and voltage sense resistors on page 3-19
- Calculating the core voltage on page 3-21
- Calculating the core voltage as an offset from the default voltage on page 3-22
- Calculating the cache voltage on page 3-23.

See Reset and power-management controller on page 3-10 for details of operating and power-down modes. Figure 3-5 shows the bit assignment of the registers.

Table 3-5 Reset and power-management signal descriptions (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSYSRST</td>
<td>Input</td>
<td>System reset. The nSYSRST signal is generated by the system controller FPGA on the motherboard. It is used to generate the FPGA_HRESETn signal and the reset for the FPGA power management control when the core module is attached to a motherboard. It is selected by the motherboard detect signal, nMBDET.</td>
</tr>
<tr>
<td>PBRST</td>
<td>Input</td>
<td>Push-button reset. The PBRST signal is generated by pressing the reset button.</td>
</tr>
<tr>
<td>SWRST</td>
<td>Output</td>
<td>Software reset. The core module FPGA provides a software reset that can be triggered by writing to the reset bit in the CM_CTRL register. This generates the internal reset signal SWRST that generates nSRST and resets the whole system, see Control Register, CM_CTRL on page 4-13</td>
</tr>
</tbody>
</table>

3.2.4 Control of core and cache voltages

The core module provides facilities to control the core and cache voltages of the test chip. How to use these facilities is described in:

- Reading the voltages on page 3-17
- Modifying the voltages on page 3-18
- Voltage control and voltage sense resistors on page 3-19
- Calculating the core voltage on page 3-21
- Calculating the core voltage as an offset from the default voltage on page 3-22
- Calculating the cache voltage on page 3-23.

See Reset and power-management controller on page 3-10 for details of operating and power-down modes. Figure 3-5 shows the bit assignment of the registers.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSYSRST</td>
<td>Input</td>
<td>System reset. The nSYSRST signal is generated by the system controller FPGA on the motherboard. It is used to generate the FPGA_HRESETn signal and the reset for the FPGA power management control when the core module is attached to a motherboard. It is selected by the motherboard detect signal, nMBDET.</td>
</tr>
<tr>
<td>PBRST</td>
<td>Input</td>
<td>Push-button reset. The PBRST signal is generated by pressing the reset button.</td>
</tr>
<tr>
<td>SWRST</td>
<td>Output</td>
<td>Software reset. The core module FPGA provides a software reset that can be triggered by writing to the reset bit in the CM_CTRL register. This generates the internal reset signal SWRST that generates nSRST and resets the whole system, see Control Register, CM_CTRL on page 4-13</td>
</tr>
</tbody>
</table>

Table 3-5 Reset and power-management signal descriptions (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>nSYSRST</td>
<td>Input</td>
<td>System reset. The nSYSRST signal is generated by the system controller FPGA on the motherboard. It is used to generate the FPGA_HRESETn signal and the reset for the FPGA power management control when the core module is attached to a motherboard. It is selected by the motherboard detect signal, nMBDET.</td>
</tr>
<tr>
<td>PBRST</td>
<td>Input</td>
<td>Push-button reset. The PBRST signal is generated by pressing the reset button.</td>
</tr>
<tr>
<td>SWRST</td>
<td>Output</td>
<td>Software reset. The core module FPGA provides a software reset that can be triggered by writing to the reset bit in the CM_CTRL register. This generates the internal reset signal SWRST that generates nSRST and resets the whole system, see Control Register, CM_CTRL on page 4-13</td>
</tr>
</tbody>
</table>
Reading the voltages

The core module has an onboard 8-bit multichannel *Analogue to Digital Converter (ADC)* that enables you to read the core, cache, 1.8V, and 2.5V supply voltages.

The converted values are constantly refreshed at a rate of 20KHz. They are stored in the relevant Voltage Configuration Register, CM_VOLTAGE_CTLx. See *Core Module Voltage Configuration Registers, CM_VOLTAGE_CTL0-3* on page 4-20 for more information.

The voltage reference for the ADC is 2.05V, therefore the ADC step voltage is $2.05V/2^8$ or 8mV. However, all input voltages except for 1V8 are divided by two before the ADC and therefore have a step value of 16mV.

To calculate the voltage value, multiply the 8-bit register value by the voltage step:

**Core**  
The core operating voltage is:  
$$V_{\text{CORE, OPERATING}} = \text{CM\_VOLTAGE\_CTL0}[15:8] \times 0.016V$$

**Cache**  
The cache operating voltage is:  
$$V_{\text{CORE, OPERATING}} = \text{CM\_VOLTAGE\_CTL1}[15:8] \times 0.016V$$

**1V8**  
The 1V8 supply voltage is not divided by two before it is input to the ADC, therefore the voltage is:  
$$V_{\text{CORE, OPERATING}} = \text{CM\_VOLTAGE\_CTL2}[15:8] \times 0.008V$$

**2V5**  
The 2V5 supply voltage is:  
$$V_{\text{CORE, OPERATING}} = \text{CM\_VOLTAGE\_CTL3}[15:8] \times 0.016V$$
Modifying the voltages

Depending on the **COREPWREN** and **CACHEPWREN** signals from the test chip, and the external wake-up events, the FPGA Power-Management Controller uses the values stored in the voltage control registers to sequence the cache and core voltages from:

- operating to power-down levels
- power-down to operating levels.

Note

The cache voltage must always be greater than or equal to the core voltage. Set the register values so that the:

- core operating voltage is less than or equal to the cache operating voltage
- core power-down voltage is less than or equal to the cache power-down voltage.

A protective mechanism is implemented on the FPGA to prevent damage occurring from this, see *Voltage polarity protection* on page 4-23.

The FPGA controls the core and cache voltages by writing to an 8-bit DAC. There is one DAC allocated for each voltage.

In operating state:

- the core DAC gets written with the contents of CM_VOLTAGE_CTL0[7:0]
- the cache DAC gets written with the contents of CM_VOLTAGE_CTL1[7:0]

In power-down state:

- the core DAC gets written with the contents of CM_VOLTAGE_CTL2[7:0]
- the cache DAC gets written with the contents of CM_VOLTAGE_CTL3[7:0]

The voltage that corresponds to the CM_VOLTAGE_CTLx registers depends on the voltage control resistors (see *Voltage control and voltage sense resistors* on page 3-19).
Voltage control and voltage sense resistors

The following paragraphs describe how the actual voltage values stored in the voltage configuration registers relate to the resistor values using the core voltage in operating state as an example.

The set voltages that you get by writing to these registers is dictated by the value of four resistors on the board:

- resistors R64 (24KΩ default value) and R68 (15KΩ default value) control the core operating voltage
- resistors R78 (24KΩ default value) and R81 (15KΩ default value) control the cache operating voltage.

__Note__

The default values above provide V_DDCore_Reset and V_DDCache_Reset voltage of 0.97V with ±0.6V adjustment.

Future core modules might have different values of resistors fitted from those described above.

The core voltage DAC modifies the feedback voltage in the regulator circuit as shown in Figure 3-6.

__Figure 3-6 Core voltage circuit__
—— Caution ———

The voltage control resistors are surface-mounted to the board. You might cause damage to the board if you do not use the appropriate tools and procedures to remove or refit these resistors.

If you change the resistor values, ensure that the ratio does not result in excessive voltage being applied to the core or cache.

Figure 3-7 shows the location of the resistors.
Calculating the core voltage

The 8-bit DAC injects a current between 0 and 50µA into the feedback node. If no current is injected, the core voltage is determined by resistors R64 and R68 only and the maximum core voltage is supplied.

Example 3-3 shows how to calculate the maximum voltage.

Caution

The maximum voltage that can be set depends on the feedback resistors (see Voltage control and voltage sense resistors on page 3-19). You must not exceed the specified operating conditions for the test chip.

Example 3-3 Maximum core voltage value

If the 8-bit DAC control register CM_VOLTAGE_CTL0[7:0] is loaded with 0x00, no current will be injected into the feedback node and the output voltage is given by:

\[ V_{DDCORE_{\text{MAX}}} = V_{DAC1_{\text{OUT}}} \times \left( 1 + \frac{R64}{R68} \right) \]

Where \( V_{DAC1_{\text{OUT}}} \) is the feedback voltage at DAC1_OUT, or 0.605V. If R64 is 24KΩ and R68 is 15KΩ, the maximum value for the core voltage is 1.573V.

Example 3-4 Voltage step value

If current is injected into the feedback node, the core voltage drops. Example 3-4 uses the value for \( I_{DAC} \), the full-scale current from the DAC, and the value for R64 to calculate \( V_{STEP} \). The step current is the full-scale DAC output current of 50µA divided by \( 2^{8} - 1 \). The step voltage is the step current times the 24KΩ resistor R64.

\[ V_{STEP_{DAC}} = \frac{50 \mu A}{255} \times R64 \]

\[ V_{STEP_{DAC}} = 4.71 \text{mV} \]

In this example, R64 is the default value of 24KΩ. If R64 is not 24KΩ, the step voltage will be different.
The 8-bit DAC resets to half of its scale corresponding to the register value CM_VOLTAGE_CTL0[7:0] of 0x80. The default value for the core voltage after reset is 0.971V as shown in Example 3-5:

Example 3-5 Core reset voltage value

\[
V_{DDCORE} = V_{DDCORE\_MAX} - (CM\_VOLTAGE\_CTL0[7:0] \times V_{STEP})
\]

\[
V_{DDCORE\_RESET} = 1.573V - (0x80 \times 4.7mV)
\]

\[
V_{DDCORE\_RESET} = 0.971V
\]

The default value of CM_VOLTAGE_CTL2[7:0] is also 0x80. This gives a default power-down core operating voltage of 0.971V.

--- Note ---

The reset voltage and the step value are dependent on the values of resistors R64 and R68. If the resistors on your board are not 24KΩ and 15KΩ, the reset and step voltage are different. See Voltage control and voltage sense resistors on page 3-19.

---

Calculating the core voltage as an offset from the default voltage

Because the default value for the voltage control registers is 0x80, you can calculate the register values based on the difference between the desired voltage and the reset voltage. The resultant voltage, after writing a new value in CM_VOLTAGE_CTL0[7:0], is determined by:

\[
V_{DDCORE\_OPERATING} = V_{DDCORE\_RESET} + ((0x80 - CM\_VOLTAGE\_CTL0[7:0]) \times V_{STEP})
\]

The power-down voltage, based on the value in CM_VOLTAGE_CTL2[7:0], is:

\[
V_{DDCORE\_POWER\_DOWN} = V_{DDCORE\_RESET} + ((0x80 - CM\_VOLTAGE\_CTL2[7:0]) \times V_{STEP})
\]

Writing a value larger than 0x80 decreases the voltage and writing a value lower than 0x80 increases the voltage.

Example 3-6 on page 3-23 and Example 3-7 on page 3-23 use the $V_{DDCORE\_RESET}$ value calculated from Example 3-5 and the $V_{STEP}$ value calculated from Example 3-4 on page 3-21.
Example 3-6 Core operating voltage for CM_VOLTAGE_CTL0 = 0x70

A value of 0x70 written to CM_VOLTAGE_CTL0[7:0] increases the voltage by 16 steps:

\[ V_{DDCORE,OPERATING} = 0.972 + ((0x80 - 0x70) \times 0.00471) \]
\[ V_{DDCORE,OPERATING} = 0.972 + (16 \times 0.00471) \]
\[ V_{DDCORE,OPERATING} = 0.972 + 0.075 \]
\[ V_{DDCORE,OPERATING} = 1.047 \]

Example 3-7 Core power-down voltage for CM_VOLTAGE_CTL2 = 0x90

A value of 0x90 written to CM_VOLTAGE_CTL0[7:0] decreases the voltage by 16 steps:

\[ V_{DDCORE,POWER_DOWN} = 0.972 + ((0x80 - 0x90) \times 0.00471) \]
\[ V_{DDCORE,POWER_DOWN} = 0.972 - (16 \times 0.00471) \]
\[ V_{DDCORE,POWER_DOWN} = 0.972 - 0.075 \]
\[ V_{DDCORE,POWER_DOWN} = 0.897 \]

Calculating the cache voltage

To set or calculate the cache voltage, use the same method as described in Example 3-3 on page 3-21 to Example 3-7, but modify the equations to:

- use R78 instead of R64 (24KΩ is the current default value for both resistors)
- use R81 instead of R68 (15KΩ is the current default value for both resistors)
- use the Cache Operating Voltage Control Register CM_VOLTAGE_CTL1[7:0] instead of the Core Operating Voltage Control Register CM_VOLTAGE_CTL0[7:0]
- use the Cache Power-Down Voltage Control Register CM_VOLTAGE_CTL3[7:0] instead of the Core Power-Down Voltage Control Register CM_VOLTAGE_CTL2[7:0].
The default value for CM_VOLTAGE_CTL1[7:0] and CM_VOLTAGE_CTL3[7:0] is 0x80. The default value for both the cache operating voltage and cache power-down voltage is also 0.971V.

--- Note ---

The cache voltage must always be greater or equal to the core voltage.

A mechanism has been implemented to prevent this. If you try to set the cache voltage to less than the core voltage, the value in the core control register is ignored and the core and cache voltages are both controlled from the cache registers CM_VOLTAGE_CTL1 and CM_VOLTAGE_CTL3, see Voltage polarity protection on page 4-23.

### 3.2.5 Debug communications interrupts

The ARM10200E reference device incorporates EmbeddedICE logic and provides debug communications data registers that are used to pass data between the processor and JTAG equipment. See the ARM1020E Technical Reference Manual for a description of the debug communications channel.

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 in the core module FPGA and can be enabled and cleared by accessing the interrupt registers (see Interrupt control registers on page 4-27).
3.3 Clock generators

The core module provides its own clock generators and operates asynchronously with the motherboard. The clock generators provide two programmable clock sources. In addition, a fixed-frequency reference clock CLKREF24MHZ is supplied to the FPGA. These are discussed in the following subsections:

- Clock generation functional overview
- Programming the processor clock (PLLCLK) on page 3-28
- Programming the auxiliary clock (AUXCLK) on page 3-29
- FPGA reference clock (CLKREF24MHZ) on page 3-30.

3.3.1 Clock generation functional overview

The architecture of the clock generators is shown in Figure 3-8.

![Figure 3-8 Core module clock generator](image)

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

The ratio between the core clock, GCLK, and the internal HCLK is determined by coprocessor CP15 in the ARM1020E core.

The internal HCLK is divided by 2, 3, or 4 and output as the external HCLK. The ratio between the internal HCLK and the external HCLK is programmable by the HBUSSEL[1:0] bits in CM_INIT (see Core Module Initialization Register, CM_INIT on page 4-24).

Figure 3-9 Internal test chip clock control
The clock selection circuitry inside the test chip selects normal or test modes. Table 3-1 shows the relationship between the selection signals and the internal clocks.

Table 3-1 Internal clock selection

<table>
<thead>
<tr>
<th>PLLCLKTST[3:0]</th>
<th>BYPASS[1:0]</th>
<th>GCLK</th>
<th>HCLK (internal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b0000</td>
<td>b00</td>
<td>ARM_PLLCLKIN × MDIV[7:0]</td>
<td>ARM_PLLCLKIN × MDIV[7:0] / HDIV[1:0]</td>
</tr>
<tr>
<td>b0000</td>
<td>b01</td>
<td>ARM_PLLCLKIN</td>
<td>ARM_PLLCLKIN / HDIV[1:0]</td>
</tr>
<tr>
<td>b0000</td>
<td>b1x</td>
<td>ARM_PLLCLKIN</td>
<td>ARM_PLLCLKIN</td>
</tr>
<tr>
<td>b0011</td>
<td>bxx</td>
<td>ARM_PLLCLKIN</td>
<td>ARM_PLLCLKIN</td>
</tr>
<tr>
<td>b01xx</td>
<td>bxx</td>
<td>PLLCLKTST[1]</td>
<td>PLLCLKTST[0]</td>
</tr>
</tbody>
</table>

Note

Other values for PLLCLKTST[3:0] are reserved for testing and select the clock signal that is output on PLLCLKTSTOUT. See Table 3-2.

Table 3-2 Test clock output

<table>
<thead>
<tr>
<th>PLLCLKTST[3:0]</th>
<th>PLLCLKTSTOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>b0000</td>
<td>0, normal mode. GCLK and HCLK are controlled by BYPASS[1:0]</td>
</tr>
<tr>
<td>b0001</td>
<td>ARM_PLLCLKIN / MDIV[7:0]</td>
</tr>
<tr>
<td>b0010</td>
<td>0, clock circuit in current test mode</td>
</tr>
<tr>
<td>b0011</td>
<td>ARM_PLLCLKIN</td>
</tr>
<tr>
<td>b1000</td>
<td>GCLK</td>
</tr>
<tr>
<td>b1001</td>
<td>HCLK</td>
</tr>
<tr>
<td>b1010</td>
<td>PLL output</td>
</tr>
<tr>
<td>b1011</td>
<td>ARM_PLLCLKIN</td>
</tr>
<tr>
<td>b110x</td>
<td>Implementation-specific test modes</td>
</tr>
<tr>
<td>b111x</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
3.3.2 Programming the processor clock (PLLCLK)

The frequency of PLLCLK is controllable in 1MHz steps in the range 12MHz to 160MHz. This is achieved by setting the Voltage Controlled Oscillator (VCO) divider for the PLLCLK 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 and output divider values are fixed. The default value for PLLCLK is 20MHz.

Figure 3-10 shows the values placed on the divider input pins and how the clock speeds are obtained.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 1 1 0</td>
<td>0 C C C C C C</td>
<td>0 0 1</td>
</tr>
<tr>
<td>22 (fixed value)</td>
<td>4 to 152</td>
<td>2 (fixed value)</td>
</tr>
</tbody>
</table>

**Figure 3-10 ARM PLLCLK divider control**

The bits marked:
- C are programmable in the CM_OSC register
- 1 are tied HIGH
- 0 are tied LOW.

The frequency in MHz of PLLCLK can be calculated using the formula:

\[ \text{freq} = \text{C}_\text{VDW} + 8 \]

where:

C_VDW is the VCO divider word for the core clock.

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

The FPGA_PLLCLK signal is converted to test chip pads supply level by the FPGA. The clock, now called ARM_PLLCLK, is series terminated with a 33Ω resistor and then drives a single load on the microprocessor core.

--- Note ---

The ARM10200E test chip uses PLLCKTST to control whether the test chip PLL is bypassed. For details on configuring the PLL in the test chip, see Core Module Initialization Register, CM_INIT on page 4-24.
3.3.3 Programming the auxiliary clock (AUXCLK)

The frequency of the AUXCLK is controlled by the register CM_AUXOSC, see Auxiliary Oscillator Register, CM_AUXOSC on page 4-16. The values for RDW, VDW, and OD are all programmable, as shown in Figure 3-11, to give frequencies in the range 1 to 160MHz with a better than 0.1% accuracy. The default frequency following a power-on reset is 32.369MHz.

![Figure 3-11 AUXCLK divider control](image)

The clock frequency is controlled by the divider signals AUXCLKCONTROL[18:0] that are assigned to the control parameters as shown in Table 3-1.

<table>
<thead>
<tr>
<th>Signals</th>
<th>Control parameter</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXCLKCONTROL[18:16]</td>
<td>Output divider</td>
<td>A_OD</td>
</tr>
<tr>
<td>AUXCLKCONTROL[15:9]</td>
<td>Reference divider</td>
<td>A_RDW</td>
</tr>
<tr>
<td>AUXCLKCONTROL[8:0]</td>
<td>VCO divider</td>
<td>A_VDW</td>
</tr>
</tbody>
</table>

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

\[
\text{freq} = \frac{48(A\_VDW + 8)}{(A\_RDW + 2)A\_OD}
\]

**Note**

Values for VDW, RDW, and OD can be calculated using the ICS525 calculator on the Microclock website.
3.3.4 Programming the output dividers

The output divider, OD, is not a straightforward binary representations of a decimal values but have assigned values. The value assigned to each bit pattern is shown in Table 3-1.

<table>
<thead>
<tr>
<th>OD[2:0]</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>4</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>111</td>
<td>6</td>
</tr>
<tr>
<td>101</td>
<td>7</td>
</tr>
<tr>
<td>010</td>
<td>8</td>
</tr>
<tr>
<td>110</td>
<td>9</td>
</tr>
<tr>
<td>000</td>
<td>10</td>
</tr>
</tbody>
</table>

3.3.5 FPGA reference clock (CLKREF24MHZ)

The CLKREF24MHZ signal is used by the FPGA to generate:

- the SDRAM refresh clock
- SPD EEPROM clock
- real time counter.
3.4 Multi-ICE support

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

- Multi-ICE connection
- JTAG scan paths on page 3-32
- JTAG connection modes on page 3-34
- JTAG signals on page 3-35.

3.4.1 Multi-ICE connection

Figure 3-12 shows the Multi-ICE connector, CFGEN link, and LED.

The CFGEN link is used to enable in-circuit programming of the FPGA and PLDs using Multi-ICE (see JTAG connection modes on page 3-34).

The Multi-ICE connector provides a set of JTAG signals that enable JTAG debugging equipment to be used (see JTAG signals on page 3-35). If you are debugging a development system with multiple core modules, connect the JTAG debugging equipment to the top core module.
3.4.2 JTAG scan paths

This section describes JTAG scan chain data path and clock path.

Data path

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

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

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

The PLDs and FPGAs are included in the scan chain if the core module is in configuration mode, as described in JTAG connection modes on page 3-34.
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.

Synthesizable cores require that TCK is synchronous with the core clock. The cores sample TCK and generate the delayed clock signal RTCK. RTCK is used by the core and passed 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 pull-ups on nMBDET are omitted for clarity).

The ARM10200E reference device does not sample TCK but routes the TCK signal straight through to the next board down the stack. If one or more modules in a stack drives RTCK (and so asserts nRTCKEN), you must ensure that the board at the bottom of the stack provides the necessary return path. All Integrator motherboards do so.
3.4.3 JTAG connection modes

The core module is capable of operating in:

- *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 3-31). 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 3-32.

**Configuration mode**

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

To select configuration mode, fit a jumper to the CONFIG link on the core module at the top of the stack (see Figure 3-12 on page 3-31). This has the effect of pulling the reset signal LOW. nCFGEN illuminates the CFG LED 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 might be unreliable if more than one core module is attached. The larger loads on the TCK and TMS lines can cause unreliable operation.

After configuration or code updates you must:

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

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

- The FPGAs are volatile, but load their configuration from flash memory. Flash memory does not have a JTAG port, but it can be programmed by loading designs into the FPGAs and PLDs using JTAG. These devices transfer the data from the JTAG programming utility to the flash.

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

Figure 3-15 shows the pinout of the Multi-ICE connector,

--- Note ---

In the description in Table 3-2 on page 3-36, the term JTAG equipment refers to any hardware that can drive the JTAG signals to devices in the scan chain. Typically this will be Multi-ICE, although you can also use hardware from other suppliers to debug ARM processors.

---

Table 3-2 on page 3-36 provides a description of the JTAG and related signals.
Table 3-2 JTAG signal description

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBGRQ</td>
<td>Debug request (from JTAG equipment)</td>
<td>DBGRQ is a request for the processor core to enter the debug state. It is provided for compatibility with third-party JTAG equipment.</td>
</tr>
<tr>
<td>DBGACK</td>
<td>Debug acknowledge (to JTAG equipment)</td>
<td>DBGACK indicates to the debugger that the processor core has entered debug mode. It is provided for compatibility with third-party JTAG equipment.</td>
</tr>
<tr>
<td>GLOBAL_DONE</td>
<td>FPGA configured</td>
<td>GLOBAL_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 GLOBAL_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.</td>
</tr>
<tr>
<td>nCFGEN</td>
<td>Configuration enable (from jumper on module at the top of the stack)</td>
<td>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.</td>
</tr>
<tr>
<td>nRTCKEN</td>
<td>Return TCK enable (from core module to motherboard)</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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, except for the TAP controller, 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 ARM10200E reference device. 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, and this has the effect of delaying the
time at which a component actually captures data. Using a
mechanism called *adaptive clocking*, the RTCK signal is returned
by the core to the JTAG equipment, and the clock is not advanced
until the core has captured the data. In *adaptive clocking mode*,
Multi-ICE waits for an edge on RTCK before changing TCK. In
a multiple device JTAG chain, the RTCK output from a
component connects to the TCK input of the next device in the
chain. The RTCK signal on the module connectors HDRB returns
TCK to the JTAG equipment. If there are no synchronizing
components in the scan chain then it is unnecessary to use the
RTCK signal and it is connected to ground on the motherboard.

TCK  Test clock (from JTAG equipment)
TCK synchronizes all JTAG transactions. TCK connects to all
JTAG components in the scan chain. Series termination resistors
are used to reduce reflections and maintain good signal integrity.
TCK flows down the stack of modules and connects to each JTAG
component. However, if there is a device in the scan chain that
synchronizes TCK to some other clock, then all down-stream
devices are connected to the RTCK signal on that component (see
RTCK).

TDI  Test data in (from JTAG equipment)
TDI goes down the stack of modules to the motherboard and then
back up the stack, labelled TDO, connecting to each component in
the scan chain.

TDO  Test data out (to JTAG equipment)
TDO is the return path of the data input signal TDI. The module
connectors HDRB have two pins labelled TDI and TDO. TDI
refers to data flowing down the stack and TDO to data flowing up
the stack. The JTAG components are connected in the return path
so that the length of track driven by the last component in the chain
is kept as short as possible.

TMS  Test mode select (from JTAG equipment)
TMS controls transitions in the tap controller state machine. TMS
connects to all JTAG components in the scan chain as the signal
flows down the module stack.

### Table 3-2 JTAG signal description (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
</table>
| 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 ARM10200E reference device. 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, and this has the effect of delaying the
                                                                 | time at which a component actually captures data. Using a
                                                                 | mechanism called *adaptive clocking*, the RTCK signal is returned
                                                                 | by the core to the JTAG equipment, and the clock is not advanced
                                                                 | until the core has captured the data. In *adaptive clocking mode*,
                                                                 | Multi-ICE waits for an edge on RTCK before changing TCK. In
                                                                 | a multiple device JTAG chain, the RTCK output from a
                                                                 | component connects to the TCK input of the next device in the
                                                                 | chain. The RTCK signal on the module connectors HDRB returns
                                                                 | TCK to the JTAG equipment. If there are no synchronizing
                                                                 | components in the scan chain then it is unnecessary to use the
                                                                 | RTCK signal and it is connected to ground on the motherboard.     |
| TCK      | Test clock (from JTAG equipment) | TCK synchronizes all JTAG transactions. TCK connects to all
                                                                 | JTAG components in the scan chain. Series termination resistors
                                                                 | are used to reduce reflections and maintain good signal integrity.
                                                                 | TCK flows down the stack of modules and connects to each JTAG
                                                                 | component. However, if there is a device in the scan chain that
                                                                 | synchronizes TCK to some other clock, then all down-stream
                                                                 | devices are connected to the RTCK signal on that component (see
                                                                 | RTCK).                                                             |
| TDI      | Test data in (from JTAG equipment)| TDI goes down the stack of modules to the motherboard and then
                                                                 | back up the stack, labelled TDO, connecting to each component in
                                                                 | the scan chain.                                                   |
| TDO      | Test data out (to JTAG equipment) | TDO is the return path of the data input signal TDI. The module
                                                                 | connectors HDRB have two pins labelled TDI and TDO. TDI
                                                                 | refers to data flowing down the stack and TDO to data flowing up
                                                                 | the stack. The JTAG components are connected in the return path
                                                                 | so that the length of track driven by the last component in the chain
                                                                 | is kept as short as possible.                                     |
| TMS      | Test mode select (from JTAG equipment) | TMS controls transitions in the tap controller state machine. TMS
                                                                 | connects to all JTAG components in the scan chain as the signal
                                                                 | flows down the module stack.                                    |
3.5 Embedded Trace support

The ARM10200E reference device incorporates an *ARM10 Embedded Trace Macrocell* (ETM10). This enables you to carry out real-time debugging by connecting external trace equipment to the core module. To trace program flow, the ETM broadcasts branch addresses, data accesses, and status information through the trace port. Later in the debug process, the complete instruction flow can be reconstructed by the ARM *Trace Debug Tools* (TDT).

---

**Note**

It is not possible to reconstruct self-modifying code.

3.5.1 About using trace

Figure 3-16 illustrates a trace debugging setup with the core module.

---

**Figure 3-16 Trace connection**

---

**Note**

The routing of the JTAG scan chain on the Integrator system is described in *Multi-ICE support* on page 3-31.
The components in the trace debug setup shown in Figure 3-16 on page 3-38 are:

**Embedded trace macrocell**

The ETM monitors the ARM core buses and outputs compressed information through the trace port to a trace port analyzer. The on-chip ETM contains trigger and filter logic to control what is traced.

**Trace port analyzer**

The Trace Port Analyzer (TPA) is an external device that stores information from the trace port.

**JTAG unit**

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

**Trace debug tools**

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

### 3.5.2 Core trace configuration

The ETM configuration provided by the test chip is defined by its manufacturer, see the release note for your test chip.

### 3.5.3 Trace interface description

The two trace connectors enable you to connect an external embedded trace interface module. These connectors are high-density AMP Mictor connector. The pinout for these connectors is provided in *Trace connectors pinout* on page A-10.
3.6 Stacking options

The core module provides two stacking options that can be selected by the position of a surface mount link, LK1.

The stacking options are:

**Normal** The normal option enables the core module to be used with a motherboard or standalone.

**Core module at the bottom of the stack with no motherboard**

This option uses a core module at the bottom of a stack of one or more other modules. One logic module must be included to provide the system control functions such as system bus arbiter, that are usually provided by the motherboard. To use this option:

- Place the Integrator/CM10200E board at the bottom of the stack, set link LK1 to the B:C position.
- Stack at least one logic module on top of the stack. Refer to the appropriate logic module documentation for information about how to use it in this configuration.

Table 3-3 lists the link LK1 positions used to select the different stacking options.

<table>
<thead>
<tr>
<th>Position</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A:C</td>
<td>Normal</td>
</tr>
<tr>
<td>B:C</td>
<td>Core module at the bottom of the stack with no motherboard</td>
</tr>
</tbody>
</table>
Chapter 4
Programmer’s Reference

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

- Memory organization on page 4-2
- Exception vector mapping on page 4-7
- Registers on page 4-8
- SDRAM SPD memory on page 4-32.
4.1 Memory organization

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

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

4.1.1 Core module memory map

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

Table 4-1 Core module memory map

<table>
<thead>
<tr>
<th>nMBDET</th>
<th>REMAP</th>
<th>Address range</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0x00000000–0x001FFFFF</td>
<td>2MB</td>
<td>Boot ROM or flash on motherboard</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>0x00000000–0x001FFFFF</td>
<td>2MB</td>
<td>SSRAM</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0x00200000–0x007FFFFF</td>
<td>254MB</td>
<td>Local SDRAM (DIMM)</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>0x10000000–0x107FFFFF</td>
<td>8MB</td>
<td>Core module registers</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>0x10800000–0x10FFFFF</td>
<td>8MB</td>
<td>SSRAM alias</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
<td>0x11000000–0x1FFFFFFF</td>
<td>3824MB</td>
<td>System bus address space</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>0x11000000–0x1FFFFFFF</td>
<td>3824MB</td>
<td>Bus error response</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>0x30000000–0x3FFFFFFF</td>
<td>256MB</td>
<td>Private SDRAM address space</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
<td>0x40000000–0xFFFFFFFFF</td>
<td>3568MB</td>
<td>System bus address space</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>0x40000000–0xFFFFFFFFF</td>
<td>3568MB</td>
<td>Bus error response</td>
</tr>
</tbody>
</table>

Note

If the size of the physical memory is less than the address range, the physical memory is aliased and repeated to fill the address space.
**Note**

You must configure and enable the private SDRAM prior to writing to the corresponding address space. Failing to do this produces unpredictable results when trying to access the address space. Refer to *Private SDRAM configuration* on page 4-5 for information about how to configure the private SDRAM.

The memory map shows the private SDRAM located at the default location. You can relocate the private SDRAM although you must not mask important memory regions such as the register locations.

### 4.1.2 Using REMAP

The SSRAM on the core module and the alias of the boot ROM or flash memory on an Integrator motherboard share the same locations in the Integrator memory map. Accesses to these devices are controlled by the REMAP bit (see *Control Register, CM_CTRL* on page 4-13) and \texttt{nMBDET}, the motherboard detect signal, that is permanently grounded by the motherboard. The effect on the memory map is shown in Figure 4-1.

<table>
<thead>
<tr>
<th>Standalone</th>
<th>Attached to a motherboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3FFF FFFF</td>
<td>0x3FFF FFFF</td>
</tr>
<tr>
<td>0x3000 0000</td>
<td>0x3000 0000</td>
</tr>
<tr>
<td>Private SDRAM</td>
<td>Private SDRAM</td>
</tr>
<tr>
<td>Abort</td>
<td>Motherboard</td>
</tr>
<tr>
<td></td>
<td>Motherboard</td>
</tr>
<tr>
<td>0x1000 0000</td>
<td>0x1000 0000</td>
</tr>
<tr>
<td>Core module registers</td>
<td>Core module registers</td>
</tr>
<tr>
<td></td>
<td>Core module registers</td>
</tr>
<tr>
<td>SDRAM DIMM</td>
<td>SDRAM DIMM</td>
</tr>
<tr>
<td>0x020 0000</td>
<td>0x020 0000</td>
</tr>
<tr>
<td>Private SDRAM</td>
<td>Boot ROM/flash</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>nMBDET=1</td>
<td>nMBDET=0</td>
</tr>
<tr>
<td>REMAP=x</td>
<td>REMAP=0</td>
</tr>
<tr>
<td></td>
<td>nMBDET=0</td>
</tr>
<tr>
<td></td>
<td>REMAP=1</td>
</tr>
</tbody>
</table>

![Figure 4-1 Effect of remap](image-url)
The REMAP bit only has any effect if the core module is attached to a motherboard (nMBDET=0). It is controlled by bit 2 of the CM_CTRL register at 0x100000C and functions as follows:

**REMAP=0**  Accesses to locations in the address range 0x00000000–0x001FFFFF are to the boot ROM or flash on the motherboard.

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

--- Note ---

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

---

**REMAP=1**  Accesses to locations in the address range 0x00000000–0x001FFFFF are to the SSRAM on the core module.

--- Note ---

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

### 4.1.3 SDRAM DIMM mapping

The Integrator memory map provides two regions in which the SDRAM can be accessed. One region enables access only by the local processor, and the second enables access by any master in the Integrator system.

**Local SDRAM DIMM access**

The local processor can access the SDRAM DIMM at the local address range of 0x00000000–0x001FFFFF. This range is local to the core module only. However, the lowest 2MB, 0x00000000–0x00200000, is hidden by the SSRAM on boot-ROM, depending on:

- whether the core module is attached to the motherboard
- the state of the remap bit.

Access the lowest part of the SDRAM DIMM at:

- one of its repeat images if the DIMM is less than 256MB.
- the alias location, see *Global SDRAM access* on page 5-5.

Figure 4-2 on page 4-5 shows an example of a 64MB DIMM mapped four times.
4.1.4 Global SDRAM access

If multiple core modules are attached to a motherboard, each core module can access its own and the memory of other modules. The memory map depends on the type of motherboard used. See Global SDRAM access on page 5-5.

4.1.5 Private SDRAM configuration

The board is fitted with 64MB of private SDRAM that is directly connected to the ARM test chip. The SDRAM controller is implemented in the test chip.

The memory space taken by the private SDRAM is 256MB. On power-up, this space is disabled and the memory looks as shown in Table 4-1.

Table 4-1 Private SDRAM memory map on power-up

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x30000000-0x3FFxxxx</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>0x3FCxxxx</td>
<td>Configuration Register</td>
</tr>
</tbody>
</table>
To configure the private SDRAM, you must write the value 0x0s60000A to the Configuration Register at address 0x3FC00000. Where s denotes the base address on a 256MB boundary.

You must not mask any important areas such as the registers.

The resulting memory map looks as shown in Table 4-2.

### Table 4-2 Configured private SDRAM

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3FDxxxxx</td>
<td>Refresh Timer Register</td>
</tr>
<tr>
<td>0x3FExxxxx</td>
<td>Mode 0 Register</td>
</tr>
<tr>
<td>0x3FFxxxxx</td>
<td>Mode 1 Register</td>
</tr>
</tbody>
</table>

For more information on the private SDRAM Configuration Registers, refer to the [ARM10200E Test Chip Implementation Guide](#).

### Private SDRAM performance

The maximum clock frequency that the private SDRAM can be operated at is 100MHz.

——— **Note** ———

The private SDRAM is clocked from the internal bus **HCLK**.
4.2 Exception vector mapping

The convention for ARM cores is to map the exception vectors to begin at address 0. However, the ARM10200E reference device enables the vectors to be moved to 0xFFFF0000 by setting the V bit in coprocessor 15 register 1. To maintain compatibility across all cores, the default reset value maps the vector to begin at address 0 (see the ARM10200E Technical Reference Manual).

You can use the CM_INIT register to control the value of the V bit at reset, see Core Module Initialization Register, CM_INIT on page 4-24.

If you use high vectors, you must ensure that the vector table is mapped to a physical address that contains real memory, for example, core module private SDRAM.
4.3 Registers

The core module status and control registers enable the processor to determine its environment and to control some core module operations. The registers, listed in Table 4-3, are located at 0x10000000 and can only be accessed by the local processor.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Address</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM_ID</td>
<td>0x10000000</td>
<td>Read</td>
<td>Static</td>
<td>See ID Register, CM_ID on page 4-11</td>
</tr>
<tr>
<td>CM_PROC</td>
<td>0x10000004</td>
<td>Read</td>
<td>Static</td>
<td>See Processor Register, CM_PROC on page 4-11</td>
</tr>
<tr>
<td>CM_OSC</td>
<td>0x10000008</td>
<td>Read/write</td>
<td>POR</td>
<td>See Oscillator Register, CM_OSC on page 4-12</td>
</tr>
<tr>
<td>CM_CTRL</td>
<td>0x1000000C</td>
<td>Read/write</td>
<td>Reset</td>
<td>See Control Register, CM_CTRL on page 4-13</td>
</tr>
<tr>
<td>CM_STAT</td>
<td>0x10000010</td>
<td>Read</td>
<td>Reset</td>
<td>See Status Register, CM_STAT on page 4-14</td>
</tr>
<tr>
<td>CM_LOCK</td>
<td>0x10000014</td>
<td>Read/write</td>
<td>Reset</td>
<td>See Lock Register, CM_LOCK on page 4-15</td>
</tr>
<tr>
<td>CM_LMBUSCNT</td>
<td>0x10000018</td>
<td>Read</td>
<td>Reset</td>
<td>See Local Memory Bus Cycle Counter, CM_LMBUSCNT on page 4-16</td>
</tr>
<tr>
<td>CM_AUXOSC</td>
<td>0x1000001C</td>
<td>Read/write</td>
<td>POR</td>
<td>See Auxiliary Oscillator Register, CM_AUXOSC on page 4-16</td>
</tr>
<tr>
<td>CM_SDRAM</td>
<td>0x10000020</td>
<td>Read/write</td>
<td>POR</td>
<td>See SDRAM Status and Control Register, CM_SDRAM on page 4-18</td>
</tr>
<tr>
<td>CM_INIT</td>
<td>0x10000024</td>
<td>Read/write</td>
<td>POR</td>
<td>See Core Module Initialization Register, CM_INIT on page 4-24</td>
</tr>
<tr>
<td>CM_REFCNT</td>
<td>0x10000028</td>
<td>Read</td>
<td>Reset</td>
<td>See Reference Clock Cycle Counter Register, CM_REFCNT on page 4-26</td>
</tr>
<tr>
<td>CM_UNUSED1</td>
<td>0x1000002C</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>Register name</td>
<td>Address</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>--------</td>
<td>-------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>CM_FLAGS</td>
<td>0x10000030</td>
<td>Read</td>
<td>Reset</td>
<td>See Flag and Nonvolatile Flag Register on page 4-27</td>
</tr>
<tr>
<td>CM_FLAGSET</td>
<td>0x10000030</td>
<td>Write</td>
<td>Reset</td>
<td>See Flag and Nonvolatile Flag Set Register on page 4-27</td>
</tr>
<tr>
<td>CM_FLAGSCLR</td>
<td>0x10000034</td>
<td>Write</td>
<td>Reset</td>
<td>See Flag and Nonvolatile Flag Clear Register on page 4-27</td>
</tr>
<tr>
<td>CM_NVFLAGS</td>
<td>0x10000038</td>
<td>Read</td>
<td>POR</td>
<td>See Flag and Nonvolatile Flag Register on page 4-27</td>
</tr>
<tr>
<td>CM_NVFLAGSSET</td>
<td>0x10000038</td>
<td>Write</td>
<td>POR</td>
<td>See Flag and Nonvolatile Flag Set Register on page 4-27</td>
</tr>
<tr>
<td>CM_NVFLAGSCLR</td>
<td>0x1000003C</td>
<td>Write</td>
<td>POR</td>
<td>See Flag and Nonvolatile Flag Clear Register on page 4-27</td>
</tr>
<tr>
<td>CM_IRQ_STATUS</td>
<td>0x10000040</td>
<td>Read</td>
<td>Reset</td>
<td>See IRQ Status and FIQ Status Registers on page 4-29</td>
</tr>
<tr>
<td>CM_IRQ_RSTAT</td>
<td>0x10000044</td>
<td>Read</td>
<td>Reset</td>
<td>See IRQ Raw Status and FIQ Raw Status Registers on page 4-29</td>
</tr>
<tr>
<td>CM_IRQ_ENSET</td>
<td>0x10000048</td>
<td>Read</td>
<td>Reset</td>
<td>See IRQ Enable Set and FIQ Enable Set Registers on page 4-29</td>
</tr>
<tr>
<td>CM_IRQ_ENCLR</td>
<td>0x1000004C</td>
<td>Write</td>
<td>Reset</td>
<td>See IRQ Enable Clear and FIQ Enable Clear Registers on page 4-29</td>
</tr>
<tr>
<td>CM_SOFT_INTSET</td>
<td>0x10000050</td>
<td>Read/write</td>
<td>Reset</td>
<td>See Soft Interrupt Set and Soft Interrupt Clear Registers on page 4-31</td>
</tr>
<tr>
<td>CM_SOFT_INTCLR</td>
<td>0x10000054</td>
<td>Write</td>
<td>Reset</td>
<td>See Soft Interrupt Set and Soft Interrupt Clear Registers on page 4-31</td>
</tr>
<tr>
<td>CM_FIQ_STATUS</td>
<td>0x10000060</td>
<td>Read</td>
<td>Reset</td>
<td>See IRQ Status and FIQ Status Registers on page 4-29</td>
</tr>
</tbody>
</table>
Table 4-3 Core module registers (continued)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Address</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM_FIQ_RSTAT</td>
<td>0x10000064</td>
<td>Read</td>
<td>Reset</td>
<td>See IRQ Raw Status and FIQ Raw Status Registers on page 4-29</td>
</tr>
<tr>
<td>CM_FIQ_ENSET</td>
<td>0x10000068</td>
<td>Read/write</td>
<td>Reset</td>
<td>See IRQ Enable Set and FIQ Enable Set Registers on page 4-29</td>
</tr>
<tr>
<td>CM_FIQ_ENCLR</td>
<td>0x1000006C</td>
<td>Write</td>
<td>Reset</td>
<td>See IRQ Enable Set and FIQ Enable Set Registers on page 4-29</td>
</tr>
<tr>
<td>CM_VOLTAGE_CTL0</td>
<td>0x10000080</td>
<td>Read/write</td>
<td>Reset</td>
<td>See Core Module Voltage Configuration Registers, CM_VOLTAGE_CTL0-3 on page 4-20</td>
</tr>
<tr>
<td>CM_VOLTAGE_CTL1</td>
<td>0x10000084</td>
<td>Read/write</td>
<td>Reset</td>
<td>See Core Module Voltage Configuration Registers, CM_VOLTAGE_CTL0-3 on page 4-20</td>
</tr>
<tr>
<td>CM_VOLTAGE_CTL2</td>
<td>0x10000088</td>
<td>Read/write</td>
<td>Reset</td>
<td>See Core Module Voltage Configuration Registers, CM_VOLTAGE_CTL0-3 on page 4-20</td>
</tr>
<tr>
<td>CM_VOLTAGE_CTL3</td>
<td>0x1000008C</td>
<td>Read/write</td>
<td>Reset</td>
<td>See Core Module Voltage Configuration Registers, CM_VOLTAGE_CTL0-3 on page 4-20</td>
</tr>
<tr>
<td>CM_SPD</td>
<td>0x10000100-0x100001FC</td>
<td>Read</td>
<td>POR</td>
<td>See SDRAM SPD memory on page 4-32</td>
</tr>
</tbody>
</table>

--- Note ---

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

The ID Register, CM_ID, at 0x10000000 is a read-only register that identifies the board manufacturer, board type, and revision. Figure 4-3 shows the bit assignment of the register.

![Figure 4-3 ID Register, CM_ID](image)

Table 4-1 describes the Core Module ID Register assignment.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:24]</td>
<td>MAN</td>
<td>Read</td>
<td>Manufacturer: 0x41 = ARM</td>
</tr>
<tr>
<td>[23:16]</td>
<td>ARCH</td>
<td>Read</td>
<td>Architecture: 0x00 = ASB system bus, ASB processor bus, 0x08 = AHB system bus, ASB processor bus</td>
</tr>
<tr>
<td>[15:12]</td>
<td>FPGA</td>
<td>Read</td>
<td>FPGA type: 0x03 = XVC600 or XVC600E</td>
</tr>
<tr>
<td>[11:4]</td>
<td>BUILD</td>
<td>Read</td>
<td>Build value (ARM internal use)</td>
</tr>
<tr>
<td>[3:0]</td>
<td>REV</td>
<td>Read</td>
<td>0x1 = Rev B (AHB)</td>
</tr>
</tbody>
</table>

4.3.2 Processor Register, CM_PROC

The Core Module Processor Register, CM_PROC, at 0x10000004 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 ARM10200E reference device, information about the processor can be obtained by reading coprocessor 15 register 0.
4.3.3 Oscillator Register, CM_OSC

The Core Module Oscillator Register, CM_OSC, at 0x10000008 is a read/write register that controls the frequency of the clocks generated by the two clock generators (see Clock generators on page 3-25). In addition, it provides information about processor bus mode setting. Figure 4-4 shows the bit assignment of the register.

![Figure 4-4 Oscillator Register, CM_OSC](image)

**Note**

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 4-2 describes the Core Module Oscillator Register bit assignments.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:25] Reserved</td>
<td>Use read-modify-write to preserve value.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| [24:23] BMODE | Read | This field contains 11 and indicates that:  
• the ratio between the test chip GCLK and HCLK is selected by writing to coprocessor 15 register 1  
• the ratio between the test chip HCLK and the external HCLK is selected by the HBUSSEL bits in register CM_INIT. |
| [22:8] Reserved | Use read-modify-write to preserve value. |
4.3.4 Control Register, CM_CTRL

The Core Module Control Register, CM_CTRL, at 0x1000000C is a read/write register that provides control of a number of user-configurable features of the core module. Figure 4-5 shows the bit assignment of the register.

![Figure 4-5 Control Register, CM_CTRL](image)

Figure 4-5 Control Register, CM_CTRL

Table 4-1 describes the Core Module Control Register bit assignment.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3]</td>
<td>RESET</td>
<td>Write</td>
<td>This is used to reset the Integrator system. A reset is triggered by writing a 1. Reading this bit always returns a 0 enabling you to use read-modify-write operations without masking the RESET bit.</td>
</tr>
<tr>
<td>[2]</td>
<td>REMAP</td>
<td>Read/write</td>
<td>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 2MB (0x00000000–0x001FFFFF) of memory are redirected into the motherboard (see Using REMAP on page 4-3).</td>
</tr>
<tr>
<td>[1]</td>
<td>nMBDET</td>
<td>Read</td>
<td>This bit indicates whether or not the core module is mounted on a motherboard: 0 = mounted on motherboard 1 = standalone.</td>
</tr>
<tr>
<td>[0]</td>
<td>LED</td>
<td>Read/write</td>
<td>This bit controls the green MISC LED on the core module: 0 = LED OFF 1 = LED ON.</td>
</tr>
</tbody>
</table>
4.3.5 Status Register, CM_STAT

The Core Module Status Register, CM_STAT, at \(0x10000010\) is a read-only register that can be read to determine the SSRAM size, core type, and where in a multi-core module stack this core module is positioned. Figure 4-6 shows the bit assignment of the register.

![Figure 4-6 Status Register, CM_STAT](image)

Table 4-2 describes the Core Module Status Register bit assignment.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:24]</td>
<td>Reserved</td>
<td>Read-modify-write to preserve value.</td>
<td></td>
</tr>
<tr>
<td>[23:16]</td>
<td>SSRAMSIZE</td>
<td>Read</td>
<td>SSRAM size. This contains (0x20) to indicate that 2MB is fitted.</td>
</tr>
<tr>
<td>[15:8]</td>
<td>SI_ID</td>
<td>Read</td>
<td>Silicon manufacturer identification. Identifies the manufacturer and type of core fitted to the module: (0x00) = unknown or socket fitted (0x01) = Cirrus (0.25µ, 2V5 core and pads) (0x02) = Panasonic (0.18µ, 1V8 core and 3V3 pads) (0x03) = Samsung (0.25µ, 2V5 core and 3V3 pads) (0x04)–(0xFF) = reserved.</td>
</tr>
<tr>
<td>[7:0]</td>
<td>ID</td>
<td>Read</td>
<td>Card number in stack: (0x00) = core module 0 (0x01) = core module 1 (0x02) = core module 2 (0x03) = core module 3 (0xFF) = invalid or no motherboard attached.</td>
</tr>
</tbody>
</table>
4.3.6 Lock Register, CM_LOCK

The Core Module Lock Register, CM_LOCK, at 0x10000014 controls access to the following registers to enable them to be locked and unlocked:

- Oscillator Register, CM_OSC
- Auxiliary Oscillator Register, CM_AUXOSC
- Voltage Configuration Registers, CM_VOLTAGE_CTL0-3
- Initialization Register, CM_INIT.

This mechanism prevents the registers from being overwritten accidently. Figure 4-7 shows the bit assignment of the register.

![Figure 4-7 Lock Register, CM_LOCK](image)

Table 4-1 describes the Core Module Lock Register bit assignment.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:17]</td>
<td>Reserved</td>
<td>Read-modify-write</td>
<td>Use read-modify-write to preserve value.</td>
</tr>
<tr>
<td>[16]</td>
<td>LOCKED</td>
<td>Read</td>
<td>This bit indicates if the CM_OSC, CM_AUXOSC, CM_INIT, and CM_VOLTAGE_CTL0-3 Registers are locked or unlocked: 0 = unlocked 1 = locked.</td>
</tr>
<tr>
<td>[15:0]</td>
<td>LOCKVAL</td>
<td>Read/write</td>
<td>Write the value 0xA05F to this register to enable write accesses to the CM_OSC, CM_AUXOSC, CM_INIT, and CM_VOLTAGE_CTL0-3 Registers. Write any other value to this register to lock the CM_OSC register.</td>
</tr>
</tbody>
</table>
4.3.7 Local Memory Bus Cycle Counter, CM_LMBUSCNT

The Core Module Local Memory Bus Cycle Counter Register, CM_LMBUSCNT, at 0x10000018 provides a 32-bit count value. The count increments at the memory bus frequency and can be used as a cycle counter for performance measurement. The register is set to zero by a reset.

4.3.8 Auxiliary Oscillator Register, CM_AUXOSC

The Core Module Auxiliary Oscillator Register, CM_AUXOSC, at 0x1000001C is a read/write register that controls the frequency of the clock generated by the clock generator for AUXCLK (see Programming the auxiliary clock (AUXCLK) on page 3-29). This register enables you to set the all three of the control inputs to the clock generator. The default setting of this register gives a 10MHz output. Figure 4-8 shows the bit assignment of the Auxiliary Oscillator Register.

<table>
<thead>
<tr>
<th>31</th>
<th>19</th>
<th>18</th>
<th>16</th>
<th>15</th>
<th>9</th>
<th>8</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>AUX_RDW</td>
<td>AUX_VDW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AUX_OD

Figure 4-8 Auxiliary Oscillator Register, CM_AUXOSC

——— Note ————

Before writing to the CM_AUXOSC Register, you must unlock it by writing the value 0xA05F to the CM_LOCK Register. After writing the CM_AUXOSC Register, relock it by writing any value other than 0xA05F to the CM_LOCK Register.
Table 4-2 describes the Core Module Auxiliary Oscillator Register bit assignment.

### Table 4-2 Auxiliary Oscillator Register, CM_AUXOSC bit assignment

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:19]</td>
<td>Reserved</td>
<td>Use read-modify-write to preserve value.</td>
<td></td>
</tr>
</tbody>
</table>
| [18:16]| AUX_OD    | Read-write   | Auxiliary output divider. Sets the binary code on the S[2:0] pins of the clock generator. The divider is encoded as follows:  
  b000 = divide by 0  
  b001 = divide by 2 (default)  
  b010 = divide by 8  
  b011 = divide by 4  
  b100 = divide by 5  
  b101 = divide by 7  
  b110 = divide by 9  
  b111 = divide by 6. |
  b0101110 = 46 (default). |
| [8:0]  | AUX_VD    | Read/write   | Auxiliary clock VCO divider word. Defines the binary value on the V[8:0] pins of the clock generator.  
  b000001100 = 12 (default). |
4.3.9 SDRAM Status and Control Register, CM_SDRAM

The Core Module SDRAM Status and Control Register, CM_SDRAM, at 0x10000020 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. Figure 4-9 shows the bit assignment of the register.

---

Note

Before the SDRAM DIMM is used, it is necessary to read the SPD memory and program the CM_SDRAM register with the parameters indicated in Table 4-3 on page 4-19. If these values are not correctly set then SDRAM accesses might be slow or unreliable. See SDRAM SPD memory on page 4-32.
Table 4-3 describes the Core Module SDRAM Status and Control Register bit assignment.

### Table 4-3 SDRAM Status and Control Register, CM_SDRAM bit assignment

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:20]</td>
<td>Reserved</td>
<td>Use read-modify-write to preserve value.</td>
<td></td>
</tr>
<tr>
<td>[19:16]</td>
<td>NBANKS</td>
<td>Read/write</td>
<td>Number of SDRAM banks. Set to the same value as byte 5 of SPD EEPROM.</td>
</tr>
<tr>
<td>[15:12]</td>
<td>NCOLS</td>
<td>Read/write</td>
<td>Number of SDRAM columns. Set to the same value as byte 4 of SPD EEPROM.</td>
</tr>
<tr>
<td>[11:8]</td>
<td>NROWS</td>
<td>Read/write</td>
<td>Number of SDRAM rows. Set to the same value as byte 3 of SPD EEPROM.</td>
</tr>
<tr>
<td>[5]</td>
<td>SPDOK</td>
<td>Read</td>
<td>This bit indicates that the automatic copying of the SPD data from the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SDRAM module into CM_SPDMEM is complete:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = SPD data ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 = SPD data not available.</td>
</tr>
<tr>
<td>[4:2]</td>
<td>MEMSIZE</td>
<td>Read/write</td>
<td>These bits specify the size of the SDRAM module fitted to the core</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>module. The bits are encoded as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b000 = 16MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b001 = 32MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b010 = 64MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b011 = 128MB (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b100 = 256MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b101 = Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b110 = Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b111 = Reserved</td>
</tr>
<tr>
<td>[1:0]</td>
<td>CASLAT</td>
<td>Read/write</td>
<td>These bits specify the CAS latency set for the core module. The bits are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>encoded as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b00 = Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b01 = Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b10 = 2 cycles (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b11 = 3 cycles.</td>
</tr>
</tbody>
</table>
4.3.10 Core Module Voltage Configuration Registers, CM_VOLTAGE_CTL0-3

The four Voltage Configuration Registers enable you to read and write different voltages on the board.

See Reset and power-management controller on page 3-10 for details of operating and power-down modes. Figure 4-10 shows the bit assignment of the register.

![Figure 4-10 Voltage control registers](image)

The bit assignments for these registers and configuration process are described in:
- Core Operating Voltage Register, CM_VOLTAGE_CTL0 on page 4-21
- Cache Operating Voltage Register, CM_VOLTAGE_CTL1 on page 4-21
- Core Power-Down Voltage Register, CM_VOLTAGE_CTL2 on page 4-22
- Cache Power-Down Voltage Register, CM_VOLTAGE_CTL3 on page 4-22
- Voltage polarity protection on page 4-23
- Unlocking before writing to the registers on page 4-23
- Confirming the voltage register value on page 4-23.

--- Caution ---

Voltage control is dependent on the values of the voltage feedback resistors. The equations assume feedback resistor values of 24KΩ and 15KΩ.

See Control of core and cache voltages on page 3-16 for details on calculating the target voltages from the values of the control voltage register and feedback resistors.
Core Operating Voltage Register, CM_VOLTAGE_CTL0

The bit assignment for the Core Operating Voltage Register, CM_VOLTAGE_CTL0, at 0x10000080 is listed in Table 4-1.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:16]</td>
<td>Reserved</td>
<td>Use read-modify-write to preserve value</td>
</tr>
<tr>
<td>[15:8]</td>
<td>Read</td>
<td>Core operating voltage value is read from the ADC &lt;br&gt;( V_{\text{CORE OPERATING}} = \text{CM_VOLTAGE_CTL0}[15:8] \times 0.016V )</td>
</tr>
<tr>
<td>[7:0]</td>
<td>Read/write</td>
<td>Sets the core operating voltage DAC value &lt;br&gt;The voltage equation, with feedback resistors of 24KΩ and 15KΩ, is: &lt;br&gt;( V_{\text{CORE}} = 0.972V + (0x80-\text{CM_VOLTAGE_CTL0}[7:0]) \times 0.00471V )</td>
</tr>
</tbody>
</table>

Cache Operating Voltage Register, CM_VOLTAGE_CTL1

The bit assignment for the Cache Operating Voltage Register, CM_VOLTAGE_CTL1, at 0x10000084 is listed in Table 4-2.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:16]</td>
<td>Reserved</td>
<td>Use read-modify-write to preserve value</td>
</tr>
<tr>
<td>[15:8]</td>
<td>Read</td>
<td>Cache operating voltage value is read from the ADC &lt;br&gt;( V_{\text{CACHE OPERATING}} = \text{CM_VOLTAGE_CTL1}[15:8] \times 0.016V )</td>
</tr>
<tr>
<td>[7:0]</td>
<td>Read/write</td>
<td>Sets the cache operating voltage DAC value &lt;br&gt;The voltage equation, with feedback resistors of 24KΩ and 15KΩ, is: &lt;br&gt;( V_{\text{CACHE}} = 0.972V + (0x80-\text{CM_VOLTAGE_CTL1}[7:0]) \times 0.00471V )</td>
</tr>
</tbody>
</table>
Core Power-Down Voltage Register, CM_VOLTAGE_CTL2

The bit assignment for the Core Power-Down Voltage Register, CM_VOLTAGE_CTL2, at 0x10000088 is listed in Table 4-3.

Table 4-3 Core Power-Down Voltage Register, CM_VOLTAGE_CTL2

<table>
<thead>
<tr>
<th>Bits</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:16]</td>
<td>Reserved</td>
<td>Use read-modify-write to preserve value</td>
</tr>
<tr>
<td>[15:8]</td>
<td>Read</td>
<td>Value for 1V8 is read from the ADC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[V_{1V8} = \text{CM_VOLTAGE_CTL2}[15:8] \times 0.008V]</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The 1V8 level is not divided by two before being passed to the ADC.</td>
</tr>
<tr>
<td>[7:0]</td>
<td>Read/write</td>
<td>Sets the core power-down voltage DAC value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The voltage equation, with feedback resistors of 24KΩ and 15KΩ, is:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[V_{\text{CORE_PD}} = 0.972V + (0x80-\text{CM_VOLTAGE_CTL2}[7:0]) \times 0.00471V]</td>
</tr>
</tbody>
</table>

Cache Power-Down Voltage Register, CM_VOLTAGE_CTL3

The bit assignment for the Cache Power-Down Voltage Register, CM_VOLTAGE_CTL3, at 0x1000008C is listed in Table 4-4.

Table 4-4 Cache Power-Down Voltage Register, CM_VOLTAGE_CTL3

<table>
<thead>
<tr>
<th>Bits</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:16]</td>
<td>Reserved</td>
<td>Use read-modify-write to preserve value</td>
</tr>
<tr>
<td>[15:8]</td>
<td>Read</td>
<td>Value for 2V5 is read from the ADC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[V_{2V5} = \text{CM_VOLTAGE_CTL3}[15:8] \times 0.016V]</td>
</tr>
<tr>
<td>[7:0]</td>
<td>Read/write</td>
<td>Sets the cache power-down voltage DAC value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The voltage equation, with feedback resistors of 24KΩ and 15KΩ, is:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[V_{\text{CACHE_PD}} = 0.972V + (0x80-\text{CM_VOLTAGE_CTL3}[7:0]) \times 0.00471V]</td>
</tr>
</tbody>
</table>
**Voltage polarity protection**

The cache voltage must always be greater than, or equal to, the core voltage to prevent damage to the test chip. A mechanism has been implemented to prevent this. If you try to set the cache voltage to less than the core voltage, the value in the core control register is ignored and the core and cache voltages are both controlled from the cache registers CM_VOLTAGE_CTL1 and CM_VOLTAGE_CTL3, see *Confirming the voltage register value*.

The protection mechanism does not change the value of the control registers, see Example 4-1. The protection mechanism applies to operating and power-down modes.

**Example 4-1 Protection mechanism**

1. Current core and cache operating voltage is 1.2 V.
2. The core voltage is changed to 1.4 V, but because of the protection mechanism, stays at 1.2V.
3. The cache operating voltage is changed to 1.5 V. Because this is greater than the intended core voltage of 1.4V, the protection mechanism is no longer in effect and the core voltage changes to 1.4V.

**Unlocking before writing to the registers**

Before writing to any of the Voltage Configuration Registers, you must unlock it by writing the value 0xA05F to the CM_LOCK register. After writing to the Voltage Configuration Registers, relock it by writing any value other than 0xA05F to the CM_LOCK register.

**Confirming the voltage register value**

You can confirm the written values in the Voltage Configuration Registers by reading bits[7:0]. See *Control of core and cache voltages* on page 3-16 for details on calculating the target voltages from the values of the control voltage register and feedback resistors.

Use bits[15:8] of the registers to determine the actual operating voltages.
4.3.11 Core Module Initialization Register, CM_INIT

Figure 4-11 shows the bit assignment of the Initialization Register, CM_INIT, at 0x10000024. The CM_INIT register controls some of the signals to the test chip, see Test chip configuration control on page 3-4.

--- Note ---
Before writing to the CM_INIT register, you must unlock it by writing the value 0x0000A05F to the CM_LOCK register. After writing the CM_INIT register, relock it by writing any value other than 0xA05F to the CM_LOCK register.

--- Note ---
The ARM10200E test chip clocks are affected by the value of PLLCLKTST:

b0011 The internal PLL is bypassed.
  • GCLK is equal to PLLREFCLK
  • HCLK is equal to GCLK.

b0000 The internal PLL is used.
  • GCLK is equal to PLLREFCLK * (CP15_MDIV+1)
  • HCLK is equal to GCLK / (CP15_HDIV+1).
Table 4-5 describes the Core Module Initialization Register bit assignment.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:16]</td>
<td>Reserved</td>
<td>Use read-modify-write to preserve value.</td>
<td></td>
</tr>
</tbody>
</table>
| [15:14]  | PLLClkTst Mux Mode | Read/write| Bits[15:14] controls when the values in bits[11:8] are passed to the test chip:  
|          |                    |           | b00 = synchronized to HCLK (the raw APB register is accessed, this is the default)  
|          |                    |           | b01 = delayed until the next reset event  
|          |                    |           | b10 = synchronized to ARM_PLLCLKIN  
|          |                    |           | b11 = synchronized to ARM_PLLCLKIN falling edge.  
|          |                    |           | This selects the clock source for the internal GCLK and HCLK:  
|          |                    |           | b0000 = the internal PLL is engaged. See Clock generation functional overview on page 3-25  
|          |                    |           | b0001 = reserved  
|          |                    |           | b0010 = reserved  
|          |                    |           | b0011 = the internal PLL is bypassed (this is the default).  
| [5:4]    | HBUSSEL            | Read/write| Bits[5:4] are passed to the test chip as the HBUSSEL[1:0] signal. This controls the divider between the internal bus HCLK and the external bus HCLK. The ratios that can be achieved are:  
|          |                    |           | b00 = ratio is two (internal bus HCLK/external bus HCLK)  
|          |                    |           | b01 = ratio is three (internal bus HCLK/external bus HCLK)  
|          |                    |           | b10 = ratio is four (internal bus HCLK/external bus HCLK)  
|          |                    |           | b11 reserved  
|
4.3.12 Reference Clock Cycle Counter Register, CM_REFCNT

This register, CM_REFCNT, at 0x10000028 provides a 32-bit count value. The count increments at the fixed reference clock frequency of 24MHz and can be used as a real-time counter. The register is reset to zero by a reset.

4.3.13 Flag registers

The Core Module Flag Registers shown in Table 4-6 provide two 32-bit register locations containing general-purpose flags. You can assign any meaning to the flags.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Address</th>
<th>Access</th>
<th>Reset by</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM_FLAGS</td>
<td>0x10000030</td>
<td>Read</td>
<td>Reset</td>
<td>core module Flag register</td>
</tr>
<tr>
<td>CM_FLAGSSET</td>
<td>0x10000030</td>
<td>Write</td>
<td>Reset</td>
<td>core module Flag Set register</td>
</tr>
<tr>
<td>CM_FLAGSCCLR</td>
<td>0x10000034</td>
<td>Write</td>
<td>Reset</td>
<td>core module Flag Clear register</td>
</tr>
<tr>
<td>CM_NVFLAGS</td>
<td>0x10000038</td>
<td>Read</td>
<td>POR</td>
<td>core module Nonvolatile Flag register</td>
</tr>
<tr>
<td>CM_NVFLAGSSET</td>
<td>0x10000038</td>
<td>Write</td>
<td>POR</td>
<td>core module Nonvolatile Flag Set register</td>
</tr>
<tr>
<td>CM_NVFLAGSCCLR</td>
<td>0x1000003C</td>
<td>Write</td>
<td>POR</td>
<td>core module Nonvolatile Flag Clear register</td>
</tr>
</tbody>
</table>
The core module provides two distinct types of flag registers:

- the CM_FLAGS Register is cleared by a normal reset, such as a reset caused by pressing the reset button
- the CM_NVFLAGS Register retain its contains after a normal reset and is only cleared by a *Power-On Reset* (POR).

### Flag and Nonvolatile Flag Register

The CM_FLAGS and CM_NVFLAGS registers contain the current state of the flags.

### Flag and Nonvolatile Flag Set Register

The CM_FLAGSSET and CM_NVFLAGSSET registers are used to set bits in the CM_FLAGS and CM_NVFFLAGS registers:
- write 1 to SET the associated flag
- write 0 to leave the associated flag unchanged.

### Flag and Nonvolatile Flag Clear Register

Use the CM_FLAGSCLR and CM_NVFLAGSCLR registers to clear bits in CM_FLAGS and CM_NVFFLAGS:
- write 1 to CLEAR the associated flag
- write 0 to leave the associated flag unchanged.

### 4.3.14 Interrupt control 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. See *Interrupt control* on page 5-14 for more information on using the core module with an Integrator/AP motherboard.
The interrupt control registers are listed in Table 4-7.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Address</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM_IRQHandler_STAT</td>
<td>0x10000040</td>
<td>Read</td>
<td>core module IRQ Status register</td>
</tr>
<tr>
<td>CM_IRQHandler_RSTAT</td>
<td>0x10000044</td>
<td>Read</td>
<td>core module IRQ Raw Status register</td>
</tr>
<tr>
<td>CM_IRQHandler_ENSET</td>
<td>0x10000048</td>
<td>Read/write</td>
<td>core module IRQ Enable Set register</td>
</tr>
<tr>
<td>CM_IRQHandler_ENCLR</td>
<td>0x1000004C</td>
<td>Write</td>
<td>core module IRQ Enable Clear register</td>
</tr>
<tr>
<td>CM_Software_IRQHandler SET</td>
<td>0x10000050</td>
<td>Read/write</td>
<td>core module software interrupt set</td>
</tr>
<tr>
<td>CM_Software_IRQHandler CLR</td>
<td>0x10000054</td>
<td>Write</td>
<td>core module software interrupt clear</td>
</tr>
<tr>
<td>CM_FIQ_STAT</td>
<td>0x10000060</td>
<td>Read</td>
<td>core module FIQ Status register</td>
</tr>
<tr>
<td>CM_FIQ_RSTAT</td>
<td>0x10000064</td>
<td>Read</td>
<td>core module FIQ Raw Status register</td>
</tr>
<tr>
<td>CM_FIQ_ENSET</td>
<td>0x10000068</td>
<td>Read/write</td>
<td>core module FIQ Enable Set register</td>
</tr>
<tr>
<td>CM_FIQ_ENCLR</td>
<td>0x1000006C</td>
<td>Write</td>
<td>core module FIR Enable Clear register</td>
</tr>
</tbody>
</table>

**Note**

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

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

- status register
- raw status register
- enable register (read the current value from CM_IRQHandler_ENSET and CM_FIQ_ENSET).

The way that the interrupt enable, clear, and status bits function for each interrupt is shown in Figure 4-12 on page 4-29 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.
**IRQ Status and FIQ Status Registers**

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

**IRQ Raw Status and FIQ Raw Status Registers**

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.

**IRQ Enable Set and FIQ Enable Set Registers**

Use the Enable Set Registers to enable interrupts:
- write 1 to SET the associated bit and enable the interrupt
- write 0 to leave the associated bit unchanged.

Read the current state of the enable bits from the Enable Set Registers.

**IRQ Enable Clear and FIQ Enable Clear Registers**

Use the Enable Clear Registers to disable interrupts as follows:
- write 1 to CLEAR the associated bit and disable the interrupt
- write 0 to leave the associated bit unchanged.
Interrupt register bit assignments

The bit assignments for the IRQ and FIQ Status, Raw Status, Enable registers are shown in Figure 4-13 and Table 4-8.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:3]</td>
<td>Reserved</td>
<td>Write 0 to these bits.</td>
</tr>
<tr>
<td>[2]</td>
<td>COMMTx</td>
<td>Debug communications transmit interrupt. This interrupt indicates that the communications channel is available for the processor to pass messages to the debugger.</td>
</tr>
<tr>
<td>[1]</td>
<td>COMMRx</td>
<td>Debug communications receive interrupt. This interrupt indicates to the processor that messages are available for the processor to read.</td>
</tr>
<tr>
<td>[0]</td>
<td>SOFT</td>
<td>Software interrupt. Enabling and disabling the software interrupt is done with the Enable Set and Enable Clear Registers. Triggering the interrupt however, is done from the Soft Interrupt Set register.</td>
</tr>
</tbody>
</table>
Soft Interrupt Set and Soft Interrupt Clear Registers

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

- Set the software interrupt by writing to the CM_SOFT_INTSET register at 0x10000050:
  - 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 at 0x10000050. A 1 indicates that the interrupt request is active.

- Clear the software interrupt by writing to the CM_SOFT_INTCLR register at 0x10000054:
  - write a 1 to CLEAR the software interrupt
  - write a 0 to leave the software interrupt unchanged.

Note

Enabling and disabling the software interrupt is done through the IRQ and FIQ registers (see IRQ Enable Set and FIQ Enable Set Registers on page 4-29).

The bit assignment for the software interrupt register is shown in Table 4-9.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:1]</td>
<td>Reserved</td>
<td>Write as 0. Reads undefined.</td>
</tr>
<tr>
<td>[0]</td>
<td>SOFT</td>
<td>Software interrupt.</td>
</tr>
</tbody>
</table>

Note

The software interrupt described in this section is used by software to generate IRQs or FIQs. Do not confuse it with the ARM SWI software interrupt instruction. See the ARM Architecture Reference Manual.
4.4 SDRAM SPD memory

The area of memory from 0x10000100-0x100001FC contains a copy of the SPD data from the SPD EEPROM on the DIMM. Because accesses to the EEPROM are very slow, the data is copied to this memory during board initialization to enable faster random access to the SPD data (see Serial presence detect on page 3-9). The SPD memory contains 256 bytes of data, the most important are shown in Table 4-10.

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2]</td>
<td>Memory type</td>
</tr>
<tr>
<td>[3]</td>
<td>Number of row addresses</td>
</tr>
<tr>
<td>[4]</td>
<td>Number of column addresses</td>
</tr>
<tr>
<td>[5]</td>
<td>Number of chip select banks</td>
</tr>
<tr>
<td>[31]</td>
<td>Module bank density (MB divided by 4)</td>
</tr>
<tr>
<td>[18]</td>
<td>CAS latencies supported</td>
</tr>
<tr>
<td>[63]</td>
<td>Checksum</td>
</tr>
<tr>
<td>[64:71]</td>
<td>Manufacturer</td>
</tr>
<tr>
<td>[73:90]</td>
<td>Module part number</td>
</tr>
</tbody>
</table>

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

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

The code segment shown in Example 4-2 on page 4-33 can be used to correctly setup and remap the SDRAM DIMM.
Example 4-2 SDRAM DIMM setup and remap

```
CM_BASE EQU 0x10000000 ; base address of core module registers
SPD_BASE EQU 0x10000100 ; base address of SPD information

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

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

setupsdram
; work out the SDRAM size
LDR r0, =SPD_BASE ; point at SPD memory
LDRB r1,[r0,#3] ; number of row address lines
LDRB r2,[r0,#4] ; number of column address lines
LDRB r3,[r0,#5] ; number of banks
LDRB r4,[r0,#31] ; module bank density
MUL r5,r4,r3 ; calculate size of SDRAM (MB divided by 4)
MOV r5,r5,ASL#2 ; size in MB
CMP r5,#0x10 ; is it 16MB?
BNE not16 ; if no, move on
MOV r6,#0x2 ; store size and CAS latency of 2
B writesize
not16
CMP r5,#0x20 ; is it 32MB?
BNE not32 ; if no, move on
MOV r6,#0x6 ; store size and CAS latency of 2
B writesize
not32
CMP r5,#0x40 ; is it 64MB?
BNE not64 ; if no, move on
MOV r6,#0xa ; store size and CAS latency of 2
B writesize
```
not64
CMP      r5,#0x80  ; is it 128MB?
BNE      not128   ; if no, move on
MOV      r6,#0xe  ; store size and CAS latency of 2
B        writesize

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

writesize
MOV      r1,r1,ASL#8  ; get row address lines for SDRAM register
ORR      r2,r1,r2,ASL#12  ; OR in column address lines
ORR      r3,r2,r3,ASL#16  ; OR in number of banks
ORR      r6,r6,r3  ; OR in size and CAS latency
LDR      r0, =CM_BASE  ; point at module registers
STR      r6,[r0,#0x20]  ; store SDRAM parameters
Chapter 5
Using Core Modules with an Integrator/AP

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

- *About the system architecture* on page 5-2
- *Module ID selection* on page 5-4
- *Top-level memory map* on page 5-5
- *System bus bridge* on page 5-7
- *Interrupt control* on page 5-14.
5.1 About the system architecture

Figure 5-1 illustrates the function of the core module FPGA and shows how it connects to the other devices in the system.
5.1.1 Configuring little or big-endian operation

The core module can be configured to operate as a little-endian or big-endian system. To change to big-endian operation, write to the appropriate register in CP15 on the ARM core. The initial endianess value is programmed by writing to the BIGENDINIT of the CM_INIT register (see Core Module Initialization Register, CM_INIT on page 4-24).

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

Caution

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

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

--- Note ---

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

5.2.1 Module address decoding

The Integrator system implements a distributed address decoding system. This means that each core or logic module must decode its own area of the memory map. The central decoder in the system controller FPGA (on the motherboard) responds with an error 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 flag 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. These signals rotate as they pass up the stack, as described in System bus signal routing on page 5-11. Only one signal in each group is pulled LOW for each module.

The alias SDRAM address of a core module is determined in hardware, although a module can determine its own position by reading a decoded version of \( ID[3:0] \) from the CM_STAT register (see Status Register, CM_STAT on page 4-14). Table 5-1 shows alias addresses for a core module fitted to a the motherboard on the HDRA/HDRB stack.

<table>
<thead>
<tr>
<th>ID[3:0]</th>
<th>Module ID</th>
<th>Address range</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1101</td>
<td>3 (top)</td>
<td>0x80000000-0xBFFFFFFF</td>
<td>256MB</td>
</tr>
<tr>
<td>b1011</td>
<td>2</td>
<td>0xA0000000-0xAFFFFFFF</td>
<td>256MB</td>
</tr>
<tr>
<td>b0111</td>
<td>1</td>
<td>0x90000000-0x9FFFFFFF</td>
<td>256MB</td>
</tr>
<tr>
<td>b1110</td>
<td>0 (bottom)</td>
<td>0x80000000-0xBFFFFFFF</td>
<td>256MB</td>
</tr>
</tbody>
</table>
5.3 Top-level memory map

Figure 5-2 shows the top-level memory map of an Integrator/AP system. The memory map maintains compatibility with other platforms in the Integrator product family. This ensures code portability and enables you to expand the system with additional Integrator logic modules.

![Top-level memory map](image)

See the Integrator/AP User Guide and Chapter 4 Programmer’s Reference for more details on memory mapping and register function.

--- Note ---

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

5.3.1 Global SDRAM access

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

--- Note ---

A processor can read from and write to its own SDRAM DIMM at the system bus address. However, these accesses are slower than local accesses because they pass through a bridge to the system bus.
You can determine the core module that a processor is on by reading the CM_STAT register. From this you can determine the alias location of the SDRAM on the same core module (see Status Register, CM_STAT on page 4-14).

The alias address for a core module SDRAM DIMM is automatically controlled by its position in the stack (see Module ID selection on page 5-4). Figure 5-3 shows the alias address of the SDRAM on four core modules.

### Figure 5-3 Core module local and alias addresses for SDRAM DIMM

#### 5.3.2 Access arbitration

The SDRAM DIMM controller provides two ports to support reads and writes by the local processor core and by masters on the motherboard. The SDRAM DIMM controller uses an alternating priority scheme to ensure that the processor core and motherboard have equal access (see System bus bridge on page 5-7).
5.4 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. Accesses to the SDRAM DIMM controller are supported by a 16-entry 74-bit FIFO. Each of the 16 entries in the FIFO contain:

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

5.4.1 Processor accesses to the system bus

Processor accesses to the system bus are described in:

- Processor writes
- Processor reads on page 5-8.

Processor writes

The data routing for processor writes to the system bus is illustrated in Figure 5-4 on page 5-8.
The data, address, and control information associated with the transfer are posted into a buffer in the FPGA. Some time later when the system bus is available, two 32-bit accesses corresponding to the 64-bit access from the processor are performed. System bus error responses to write transfers are reported back to the processor as Data Aborts. If the buffer is full, the processor receives a wait response until space becomes available.

**Processor reads**

The data routing for processor reads from the system bus is illustrated in Figure 5-5 on page 5-9.
The order of processor transactions is preserved on the system bus. The processor receives a wait response until the two read transfers have completed on the system bus corresponding to a 64-bit access by the ARM, after this it receives the data and any associated bus error response from the system bus.

### 5.4.2 Motherboard accesses to the SDRAM DIMM

An overview of SDRAM access is provided in:
- *System bus writes*
- *System bus reads* on page 5-11.

See also *SDRAM DIMM mapping* on page 4-4.

#### System bus writes

The data routing for system bus writes to SDRAM DIMM is illustrated in Figure 5-6 on page 5-10.
Write transactions from the system bus to the SDRAM DIMM 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 DIMM completes when the SDRAM DIMM is available. If the FIFO is full, then the system bus master receives a retract 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 DIMM is illustrated in Figure 5-7.

![System bus reads from SDRAM](image)

**Figure 5-7 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 DIMM 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 DIMM are completed) before the read transfer is performed.

5.4.3 System bus signal routing

The core module is mounted onto an Integrator/AP motherboard using the HDRA and HDRB connectors. The signal routing is described in:

- *HDRA* on page 5-12
- *HDRB* on page 5-12.
**HDRA**

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

**HDRB**

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

This signal rotation scheme is illustrated in Figure 5-8.

The example in Figure 5-8 shows how a group of four signals (labelled A, B, C, and D) are routed through the stack. Signal C is rotated as it passes up through the stack and is only utilized on module 2.
All four signals are rotated and used 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.
5.5 Interrupt control

The system controller FPGA on the Integrator/AP motherboard incorporates an interrupt controller that can generate FIQ or IRQ interrupts from various sources in the system to up to four processors. The interrupts that a core module receives are determined by the position of the core module in the stack, as shown in Table 5-2.

<table>
<thead>
<tr>
<th>Module ID</th>
<th>Interrupt</th>
<th>Fast interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 (top)</td>
<td>nIRQ3</td>
<td>nFIQ3</td>
</tr>
<tr>
<td>2</td>
<td>nIRQ2</td>
<td>nFIQ2</td>
</tr>
<tr>
<td>1</td>
<td>nIRQ1</td>
<td>nFIQ1</td>
</tr>
<tr>
<td>0 (bottom)</td>
<td>nIRQ0</td>
<td>nFIQ0</td>
</tr>
</tbody>
</table>

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).

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

The debug comms interrupts registers are described in Debug communications interrupts on page 3-24.

Figure 5-9 on page 5-15 shows the interrupt control architecture for the Integrator/AP system.
### CM0 interrupt controller (FIQ)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:22]</td>
<td>Reserved</td>
</tr>
<tr>
<td>[21]</td>
<td>APCINT</td>
</tr>
<tr>
<td>[20]</td>
<td>PCILBINT</td>
</tr>
<tr>
<td>[19]</td>
<td>ELUMINT</td>
</tr>
<tr>
<td>[18]</td>
<td>DEGINT</td>
</tr>
<tr>
<td>[17]</td>
<td>LINT</td>
</tr>
<tr>
<td>[16]</td>
<td>PCIINT3</td>
</tr>
<tr>
<td>[15]</td>
<td>PCIINT2</td>
</tr>
<tr>
<td>[14]</td>
<td>PCIINT1</td>
</tr>
<tr>
<td>[13]</td>
<td>PCIINT0</td>
</tr>
<tr>
<td>[12]</td>
<td>EXPINT3</td>
</tr>
<tr>
<td>[10]</td>
<td>EXPINT1</td>
</tr>
<tr>
<td>[9]</td>
<td>EXPINT0</td>
</tr>
<tr>
<td>[8]</td>
<td>RTCINT</td>
</tr>
<tr>
<td>[7]</td>
<td>TIMERINT2</td>
</tr>
<tr>
<td>[6]</td>
<td>TIMERINT1</td>
</tr>
<tr>
<td>[5]</td>
<td>TIMERINT0</td>
</tr>
<tr>
<td>[4]</td>
<td>MOUSEINT</td>
</tr>
<tr>
<td>[3]</td>
<td>KBDINT</td>
</tr>
<tr>
<td>[2]</td>
<td>UARTINT1</td>
</tr>
<tr>
<td>[1]</td>
<td>UARTINT0</td>
</tr>
<tr>
<td>[0]</td>
<td>SOFTINT</td>
</tr>
</tbody>
</table>

### CM0 interrupt controller (IRQ)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:3]</td>
<td>Reserved</td>
</tr>
<tr>
<td>[2]</td>
<td>COMMTX</td>
</tr>
<tr>
<td>[1]</td>
<td>COMMRX</td>
</tr>
<tr>
<td>[0]</td>
<td>SOFTINT</td>
</tr>
</tbody>
</table>

---

**Figure 5-9 Interrupt architecture (AP image)**
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
- **Trace connectors pinout** on page A-10
- **Logic analyzer connectors** on page A-13.

--- **Note** ---

For the Multi-ICE connector pinout and signal descriptions see *JTAG signals* on page 3-35.
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.
The signals present on the pins labeled A[31:0], B[31:0], and C[31:0] are listed in Table A-1.

<table>
<thead>
<tr>
<th>Pin label</th>
<th>AHB signal name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[31:0]</td>
<td>HADDR[31:0]</td>
<td>System address bus</td>
</tr>
<tr>
<td>B[31:0]</td>
<td>Not used</td>
<td>-</td>
</tr>
<tr>
<td>C[31:16]</td>
<td>Not used</td>
<td>-</td>
</tr>
<tr>
<td>C15</td>
<td>HMASTLOCK</td>
<td>Locked transaction</td>
</tr>
<tr>
<td>C12</td>
<td>HREADY</td>
<td>Slave ready</td>
</tr>
<tr>
<td>C11</td>
<td>HWRITE</td>
<td>Write transaction</td>
</tr>
<tr>
<td>C[1:0]</td>
<td>HTRAN[1:0]</td>
<td>Transaction type</td>
</tr>
<tr>
<td>D[31:0]</td>
<td>HDATA[31:0]</td>
<td>System data bus</td>
</tr>
</tbody>
</table>
A.2 HDRB

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

- HDRB socket pinout
- HDRB plug pinout on page A-6
- Through-board signal connections on page A-7
- HDRB signal descriptions on page A-8.

A.2.1 HDRB socket pinout

Figure A-2 on page A-5 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>
<thead>
<tr>
<th>Table A-2 Signal cross-connections (example)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plug</td>
</tr>
<tr>
<td>E0</td>
</tr>
<tr>
<td>E1</td>
</tr>
<tr>
<td>E2</td>
</tr>
<tr>
<td>E3</td>
</tr>
</tbody>
</table>

For details about the signal rotation scheme, see Module ID selection on page 5-4.

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 3-35).
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>
<thead>
<tr>
<th>Pin label</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E[23:20]</td>
<td>nIRQ[3:0]</td>
<td>Interrupt request to processors 3, 2, 1, and 0 respectively.</td>
</tr>
<tr>
<td>E[19:16]</td>
<td>nFIQ[3:0]</td>
<td>Fast interrupt requests to processors 3, 2, 1, and 0 respectively.</td>
</tr>
<tr>
<td>E[11:8]</td>
<td>HLOCK[3:0]</td>
<td>System bus lock from processor 3, 2, 1, and 0 respectively.</td>
</tr>
<tr>
<td>E[7:4]</td>
<td>HGRANT[3:0]</td>
<td>System bus grant to processor 3, 2, 1, and 0 respectively.</td>
</tr>
<tr>
<td>E[3:0]</td>
<td>HBUSREQ[3:0]</td>
<td>System bus request from processors 3, 2, 1, and 0 respectively.</td>
</tr>
<tr>
<td>F[31:0]</td>
<td>-</td>
<td>Not connected.</td>
</tr>
<tr>
<td>G16</td>
<td>nRTCKEN</td>
<td>RTCK AND gate enable.</td>
</tr>
<tr>
<td>G13</td>
<td>nCFGGEN</td>
<td>Sets motherboard into configuration mode.</td>
</tr>
<tr>
<td>G12</td>
<td>nSRST</td>
<td>Multi-ICE reset (open collector).</td>
</tr>
<tr>
<td>G11</td>
<td>FPGADONE</td>
<td>Indicates when FPGA configuration is complete (open collector).</td>
</tr>
<tr>
<td>G10</td>
<td>RTCK</td>
<td>Returned JTAG test clock.</td>
</tr>
<tr>
<td>G9</td>
<td>nSYSRST</td>
<td>Buffered system reset.</td>
</tr>
<tr>
<td>G8</td>
<td>nTRST</td>
<td>JTAG reset.</td>
</tr>
<tr>
<td>G7</td>
<td>TDO</td>
<td>JTAG test data out.</td>
</tr>
<tr>
<td>G6</td>
<td>TDI</td>
<td>JTAG test data in.</td>
</tr>
<tr>
<td>G5</td>
<td>TMS</td>
<td>JTAG test mode select.</td>
</tr>
<tr>
<td>Pin label</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>G4</td>
<td>TCK</td>
<td>JTAG test clock.</td>
</tr>
<tr>
<td>G[3:1]</td>
<td>MASTER[2:0]</td>
<td>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.</td>
</tr>
<tr>
<td>G0</td>
<td>nMBDET</td>
<td>Motherboard detect pin.</td>
</tr>
</tbody>
</table>
A.3  Trace connectors pinout

Table A-4 and Table A-5 on page A-11 show the pinout of the trace connectors. The ARM_EXTOUT pins are routed to the FPGA.

<table>
<thead>
<tr>
<th>Channel</th>
<th>Pin</th>
<th>Pin</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not connected</td>
<td>1</td>
<td>2</td>
<td>Not connected</td>
</tr>
<tr>
<td>Not connected</td>
<td>3</td>
<td>4</td>
<td>Not connected</td>
</tr>
<tr>
<td>GND</td>
<td>5</td>
<td>6</td>
<td>ARM_TRACECLK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ARM_DBGACK</td>
</tr>
<tr>
<td>ARM_DBGREQ</td>
<td>7</td>
<td>8</td>
<td>ARM_EXTTRIG</td>
</tr>
<tr>
<td>ARM_TDO</td>
<td>11</td>
<td>12</td>
<td>ARM_VDDIO</td>
</tr>
<tr>
<td>RTCK</td>
<td>13</td>
<td>14</td>
<td>3V3</td>
</tr>
<tr>
<td>TCK</td>
<td>15</td>
<td>16</td>
<td>TRACEPKTA7</td>
</tr>
<tr>
<td>TMS</td>
<td>17</td>
<td>18</td>
<td>TRACEPKTA6</td>
</tr>
<tr>
<td>TDI</td>
<td>19</td>
<td>20</td>
<td>TRACEPKTA5</td>
</tr>
<tr>
<td>nTRST</td>
<td>21</td>
<td>22</td>
<td>TRACEPKTA4</td>
</tr>
<tr>
<td>TRACEPKTA1</td>
<td>23</td>
<td>24</td>
<td>TRACEPKTA3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRACEPKTA1</td>
<td>25</td>
<td>26</td>
<td>TRACEPKTA2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRACEPKTA1</td>
<td>27</td>
<td>28</td>
<td>TRACEPKTA1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRACEPKTA1</td>
<td>29</td>
<td>30</td>
<td>TRACEPKTA0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRACEPKTA1</td>
<td>31</td>
<td>32</td>
<td>ARM_PIPESTATA3</td>
</tr>
</tbody>
</table>
### Table A-4 Trace connector one pinout (continued)

<table>
<thead>
<tr>
<th>Channel</th>
<th>Pin</th>
<th>Pin</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACEPKTA1</td>
<td>33</td>
<td>34</td>
<td>ARM PIPESTATA2</td>
</tr>
<tr>
<td>TRACEPKTA9</td>
<td>35</td>
<td>36</td>
<td>ARM PIPESTATA1</td>
</tr>
<tr>
<td>TRACEPKTA8</td>
<td>37</td>
<td>38</td>
<td>ARM PIPESTATA0</td>
</tr>
</tbody>
</table>

### Table A-5 Trace connector two pinout

<table>
<thead>
<tr>
<th>Channel</th>
<th>Pin</th>
<th>Pin</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not connected</td>
<td>1</td>
<td>2</td>
<td>Not connected</td>
</tr>
<tr>
<td>Not connected</td>
<td>3</td>
<td>4</td>
<td>Not connected</td>
</tr>
<tr>
<td>GND</td>
<td>5</td>
<td>6</td>
<td>ARM TRACECLKB</td>
</tr>
<tr>
<td>Not connected</td>
<td>7</td>
<td>8</td>
<td>Not connected</td>
</tr>
<tr>
<td>Not connected</td>
<td>9</td>
<td>10</td>
<td>Not connected</td>
</tr>
<tr>
<td>Not connected</td>
<td>11</td>
<td>12</td>
<td>ARM VDDIO</td>
</tr>
<tr>
<td>Not connected</td>
<td>13</td>
<td>14</td>
<td>3V3</td>
</tr>
<tr>
<td>Not connected</td>
<td>15</td>
<td>16</td>
<td>TRACEPKTB7</td>
</tr>
<tr>
<td>Not connected</td>
<td>17</td>
<td>18</td>
<td>TRACEPKTB6</td>
</tr>
<tr>
<td>Not connected</td>
<td>19</td>
<td>20</td>
<td>TRACEPKTB5</td>
</tr>
<tr>
<td>Not connected</td>
<td>21</td>
<td>22</td>
<td>TRACEPKTB4</td>
</tr>
<tr>
<td>TRACEPKTB15</td>
<td>23</td>
<td>24</td>
<td>TRACEPKTB3</td>
</tr>
<tr>
<td>TRACEPKTB14</td>
<td>25</td>
<td>26</td>
<td>TRACEPKTB2</td>
</tr>
<tr>
<td>TRACEPKTB13</td>
<td>27</td>
<td>28</td>
<td>TRACEPKTB1</td>
</tr>
<tr>
<td>TRACEPKTB12</td>
<td>29</td>
<td>30</td>
<td>TRACEPKTB0</td>
</tr>
<tr>
<td>TRACEPKTB11</td>
<td>31</td>
<td>32</td>
<td>ARM PIPESTATB3</td>
</tr>
<tr>
<td>TRACEPKTB10</td>
<td>33</td>
<td>34</td>
<td>ARM PIPESTATB2</td>
</tr>
<tr>
<td>TRACEPKTB9</td>
<td>35</td>
<td>36</td>
<td>ARM PIPESTATB1</td>
</tr>
<tr>
<td>TRACEPKTB8</td>
<td>37</td>
<td>38</td>
<td>ARM PIPESTATB0</td>
</tr>
</tbody>
</table>
The ARM_EXTIN pins from the test chip are connected to the FPGA so that they can be driven by a special ETM validation FPGA configuration. In normal configurations the ARM_EXTTRIG signal from trace port connector one is routed through the FPGA to the ARM_EXTIN[1] pin.
A.4 Logic analyzer connectors

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

- Logic analyzer connector HADDR on page A-14
- Logic analyzer connector CONTROL on page A-15
- Logic analyzer connector HDATA[31:0] on page A-16

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

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

--- Note ---

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

---
### A.4.1 Logic analyzer connector HADDR

Table A-6 shows the pinout of the HADDR connector (J12).

<table>
<thead>
<tr>
<th>Channel</th>
<th>Pin</th>
<th>Pin</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not connected</td>
<td>1</td>
<td>2</td>
<td>Not connected</td>
</tr>
<tr>
<td>GND</td>
<td>3</td>
<td>4</td>
<td>Not connected</td>
</tr>
<tr>
<td>ARM_HTRANS1</td>
<td>5</td>
<td>6</td>
<td>ARM_HCLK</td>
</tr>
<tr>
<td>ARM_HADDR31</td>
<td>7</td>
<td>8</td>
<td>ARM_HADDR15</td>
</tr>
<tr>
<td>ARM_HADDR30</td>
<td>9</td>
<td>10</td>
<td>ARM_HADDR14</td>
</tr>
<tr>
<td>ARM_HADDR29</td>
<td>11</td>
<td>12</td>
<td>ARM_HADDR13</td>
</tr>
<tr>
<td>ARM_HADDR28</td>
<td>13</td>
<td>14</td>
<td>ARM_HADDR12</td>
</tr>
<tr>
<td>ARM_HADDR27</td>
<td>15</td>
<td>16</td>
<td>ARM_HADDR11</td>
</tr>
<tr>
<td>ARM_HADDR26</td>
<td>17</td>
<td>18</td>
<td>ARM_HADDR10</td>
</tr>
<tr>
<td>ARM_HADDR25</td>
<td>19</td>
<td>20</td>
<td>ARM_HADDR9</td>
</tr>
<tr>
<td>ARM_HADDR24</td>
<td>21</td>
<td>22</td>
<td>ARM_HADDR8</td>
</tr>
<tr>
<td>ARM_HADDR23</td>
<td>23</td>
<td>24</td>
<td>ARM_HADDR7</td>
</tr>
<tr>
<td>ARM_HADDR22</td>
<td>25</td>
<td>26</td>
<td>ARM_HADDR6</td>
</tr>
<tr>
<td>ARM_HADDR21</td>
<td>27</td>
<td>28</td>
<td>ARM_HADDR5</td>
</tr>
<tr>
<td>ARM_HADDR20</td>
<td>29</td>
<td>30</td>
<td>ARM_HADDR4</td>
</tr>
<tr>
<td>ARM_HADDR19</td>
<td>31</td>
<td>32</td>
<td>ARM_HADDR3</td>
</tr>
<tr>
<td>ARM_HADDR18</td>
<td>33</td>
<td>34</td>
<td>ARM_HADDR2</td>
</tr>
<tr>
<td>ARM_HADDR17</td>
<td>35</td>
<td>36</td>
<td>ARM_HADDR1</td>
</tr>
<tr>
<td>ARM_HADDR16</td>
<td>37</td>
<td>38</td>
<td>ARM_HADDR0</td>
</tr>
</tbody>
</table>
### Logic analyzer connector CONTROL

Table A-7 shows the pinout of the CONTROL connector (J15).

<table>
<thead>
<tr>
<th>Channel</th>
<th>Pin</th>
<th>Pin</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not connected</td>
<td>1</td>
<td>2</td>
<td>Not connected</td>
</tr>
<tr>
<td>GND</td>
<td>3</td>
<td>4</td>
<td>Not connected</td>
</tr>
<tr>
<td>HRESETn</td>
<td>5</td>
<td>6</td>
<td>Spare FPGA pin 2</td>
</tr>
<tr>
<td>ARM_nIRQ</td>
<td>7</td>
<td>8</td>
<td>Spare FPGA pin 3</td>
</tr>
<tr>
<td>ARM_nFIQ</td>
<td>9</td>
<td>10</td>
<td>Spare FPGA pin 4</td>
</tr>
<tr>
<td>Not connected</td>
<td>11(^{a})</td>
<td>12</td>
<td>SRAM_CKE(_{n})</td>
</tr>
<tr>
<td>Not connected</td>
<td>13(^{a})</td>
<td>14</td>
<td>SRAM_AVLD(_{n})</td>
</tr>
<tr>
<td>Not connected</td>
<td>15(^{a})</td>
<td>16</td>
<td>SRAM_RnW</td>
</tr>
<tr>
<td>ARM_HPROT3</td>
<td>17</td>
<td>18</td>
<td>SRAM_RAM(<em>{mat})(</em>{0})</td>
</tr>
<tr>
<td>ARM_HPROT2</td>
<td>19</td>
<td>20(^{a})</td>
<td>Not connected</td>
</tr>
<tr>
<td>ARM_HRESP1</td>
<td>21</td>
<td>22</td>
<td>SRAM_CE(_{n})</td>
</tr>
<tr>
<td>ARM_HRESP0</td>
<td>23</td>
<td>24</td>
<td>SRAM_O(_{E})</td>
</tr>
<tr>
<td>ARM_HREADY</td>
<td>25</td>
<td>26</td>
<td>FPGA_O(_{E})</td>
</tr>
<tr>
<td>Spare FPGA pin 1</td>
<td>27</td>
<td>28(^{a})</td>
<td>Not connected</td>
</tr>
<tr>
<td>ARM_HSIZE1</td>
<td>29</td>
<td>30</td>
<td>ARM_HBURST(_{2})</td>
</tr>
<tr>
<td>ARM_HSIZE0</td>
<td>31</td>
<td>32</td>
<td>ARM_HBURST(_{1})</td>
</tr>
<tr>
<td>ARM_HPROT1</td>
<td>33</td>
<td>34</td>
<td>ARM_HBURST(_{0})</td>
</tr>
<tr>
<td>ARM_HPROT0</td>
<td>35</td>
<td>36</td>
<td>ARM_HTRANS(_{1})</td>
</tr>
<tr>
<td>ARM_HWRITE</td>
<td>37</td>
<td>38</td>
<td>ARM_HTRANS(_{0})</td>
</tr>
</tbody>
</table>

\(^{a}\) This pin is connected to a test point on the board.
### A.4.3 Logic analyzer connector HDATA[31:0]

Table A-8 shows the pinout of the HDATA[31:0] connector (J14).

<table>
<thead>
<tr>
<th>Channel</th>
<th>Pin</th>
<th>Pin</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not connected</td>
<td>1</td>
<td>2</td>
<td>Not connected</td>
</tr>
<tr>
<td>GND</td>
<td>3</td>
<td>4</td>
<td>Not connected</td>
</tr>
<tr>
<td>HREADY</td>
<td>5</td>
<td>6</td>
<td>Not connected</td>
</tr>
<tr>
<td>ARM_HDATA31</td>
<td>7</td>
<td>8</td>
<td>ARM_HDATA15</td>
</tr>
<tr>
<td>ARM_HDATA30</td>
<td>9</td>
<td>10</td>
<td>ARM_HDATA14</td>
</tr>
<tr>
<td>ARM_HDATA29</td>
<td>11</td>
<td>12</td>
<td>ARM_HDATA13</td>
</tr>
<tr>
<td>ARM_HDATA28</td>
<td>13</td>
<td>14</td>
<td>ARM_HDATA12</td>
</tr>
<tr>
<td>ARM_HDATA27</td>
<td>15</td>
<td>16</td>
<td>ARM_HDATA11</td>
</tr>
<tr>
<td>ARM_HDATA26</td>
<td>17</td>
<td>18</td>
<td>ARM_HDATA10</td>
</tr>
<tr>
<td>ARM_HDATA25</td>
<td>19</td>
<td>20</td>
<td>ARM_HDATA9</td>
</tr>
<tr>
<td>ARM_HDATA24</td>
<td>21</td>
<td>22</td>
<td>ARM_HDATA8</td>
</tr>
<tr>
<td>ARM_HDATA23</td>
<td>23</td>
<td>24</td>
<td>ARM_HDATA7</td>
</tr>
<tr>
<td>ARM_HDATA22</td>
<td>25</td>
<td>26</td>
<td>ARM_HDATA6</td>
</tr>
<tr>
<td>ARM_HDATA21</td>
<td>27</td>
<td>28</td>
<td>ARM_HDATA5</td>
</tr>
<tr>
<td>ARM_HDATA20</td>
<td>29</td>
<td>30</td>
<td>ARM_HDATA4</td>
</tr>
<tr>
<td>ARM_HDATA19</td>
<td>31</td>
<td>32</td>
<td>ARM_HDATA3</td>
</tr>
<tr>
<td>ARM_HDATA18</td>
<td>33</td>
<td>34</td>
<td>ARM_HDATA2</td>
</tr>
<tr>
<td>ARM_HDATA17</td>
<td>35</td>
<td>36</td>
<td>ARM_HDATA1</td>
</tr>
<tr>
<td>ARM_HDATA16</td>
<td>37</td>
<td>38</td>
<td>ARM_HDATA0</td>
</tr>
</tbody>
</table>
A.4.4 Logic analyzer connector HDATA[63:32]

Table A-9 shows the pinout of the HDATA[63:32] connector (J13).

<table>
<thead>
<tr>
<th>Channel</th>
<th>Pin</th>
<th>Pin</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not connected</td>
<td>1</td>
<td>2</td>
<td>Not connected</td>
</tr>
<tr>
<td>GND</td>
<td>3</td>
<td>4</td>
<td>Not connected</td>
</tr>
<tr>
<td>Not connected</td>
<td>5</td>
<td>6</td>
<td>Not connected</td>
</tr>
<tr>
<td>ARM_HDATA63</td>
<td>7</td>
<td>8</td>
<td>ARM_HDATA47</td>
</tr>
<tr>
<td>ARM_HDATA62</td>
<td>9</td>
<td>10</td>
<td>ARM_HDATA46</td>
</tr>
<tr>
<td>ARM_HDATA61</td>
<td>11</td>
<td>12</td>
<td>ARM_HDATA45</td>
</tr>
<tr>
<td>ARM_HDATA60</td>
<td>13</td>
<td>14</td>
<td>ARM_HDATA44</td>
</tr>
<tr>
<td>ARM_HDATA59</td>
<td>15</td>
<td>16</td>
<td>ARM_HDATA43</td>
</tr>
<tr>
<td>ARM_HDATA58</td>
<td>17</td>
<td>18</td>
<td>ARM_HDATA42</td>
</tr>
<tr>
<td>ARM_HDATA57</td>
<td>19</td>
<td>20</td>
<td>ARM_HDATA41</td>
</tr>
<tr>
<td>ARM_HDATA56</td>
<td>21</td>
<td>22</td>
<td>ARM_HDATA40</td>
</tr>
<tr>
<td>ARM_HDATA55</td>
<td>23</td>
<td>24</td>
<td>ARM_HDATA39</td>
</tr>
<tr>
<td>ARM_HDATA54</td>
<td>25</td>
<td>26</td>
<td>ARM_HDATA38</td>
</tr>
<tr>
<td>ARM_HDATA53</td>
<td>27</td>
<td>28</td>
<td>ARM_HDATA37</td>
</tr>
<tr>
<td>ARM_HDATA52</td>
<td>29</td>
<td>30</td>
<td>ARM_HDATA36</td>
</tr>
<tr>
<td>ARM_HDATA51</td>
<td>31</td>
<td>32</td>
<td>ARM_HDATA35</td>
</tr>
<tr>
<td>ARM_HDATA50</td>
<td>33</td>
<td>34</td>
<td>ARM_HDATA34</td>
</tr>
<tr>
<td>ARM_HDATA49</td>
<td>35</td>
<td>36</td>
<td>ARM_HDATA33</td>
</tr>
<tr>
<td>ARM_HDATA48</td>
<td>37</td>
<td>38</td>
<td>ARM_HDATA32</td>
</tr>
</tbody>
</table>
Appendix B
Specifications

This appendix contains the specifications for the ARM Integrator/CM10200E and Integrator/CM10220E core modules. It contains the following sections:

- *Electrical specification* on page B-2
- *Timing specification* on page B-3
- *Mechanical details* on page B-7.
B.1 Electrical specification

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

B.1.1 Bus interface characteristics

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>3V3</td>
<td>Supply voltage (interface signals)</td>
<td>3.1</td>
<td>3.5</td>
<td>V</td>
</tr>
<tr>
<td>5V</td>
<td>Supply voltage</td>
<td>4.75</td>
<td>5.25</td>
<td>V</td>
</tr>
<tr>
<td>V_H</td>
<td>High-level input voltage</td>
<td>2.0</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>V_L</td>
<td>Low-level input voltage</td>
<td>0</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>V_OH</td>
<td>High-level output voltage</td>
<td>2.4</td>
<td>-</td>
<td>V</td>
</tr>
<tr>
<td>V_OL</td>
<td>Low-level output voltage</td>
<td>-</td>
<td>0.4</td>
<td>V</td>
</tr>
<tr>
<td>C_IN</td>
<td>Input capacitance</td>
<td>-</td>
<td>20</td>
<td>pF</td>
</tr>
</tbody>
</table>

B.1.2 Current requirements

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

<table>
<thead>
<tr>
<th>System</th>
<th>At 3.3V</th>
<th>At 5V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standalone core module</td>
<td>1A</td>
<td>1A</td>
</tr>
<tr>
<td>Motherboard and one core module</td>
<td>1.5A</td>
<td>1.5A</td>
</tr>
</tbody>
</table>

An Integrator/AP with additional core or logic modules draws more current, and future core modules might require more current. For these reasons, provision is made to power the system with an ATX-type power supply.
B.2 Timing specification

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

The following sections describe the timing parameters for typical AHB modules and motherboards:

- Integrator timing parameters and the AMBA Specification
- AHB system bus timing parameters
- Notes on FPGA timing analysis on page B-6.

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 can 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>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Time (ns)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{clk}$</td>
<td>HCLK minimum clock period</td>
<td>30</td>
<td>Representative of worst case maximum frequency</td>
</tr>
<tr>
<td>$T_{isrst}$</td>
<td>HRESETn deasserted setup time before HCLK</td>
<td>15</td>
<td>Applies to modules only</td>
</tr>
<tr>
<td>$T_{ovrst}$</td>
<td>HRESETn deasserted valid time before HCLK</td>
<td>15</td>
<td>Applies only to the module or motherboard implementing the reset source</td>
</tr>
</tbody>
</table>
Table B-4 shows the AHB slave input parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Time (ns)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{isel}$</td>
<td>HSELx setup time before HCLK</td>
<td>n/a</td>
<td>HSELx are internally generated, not visible at the pins</td>
</tr>
<tr>
<td>$T_{istr}$</td>
<td>Transfer type setup time before HCLK</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>$T_{isa}$</td>
<td>HADDR[31:0] setup time before HCLK</td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>$T_{isctl}$</td>
<td>HWRITE, HSIZE[2:0] and HBURST[2:0] control signal setup time before HCLK</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>$T_{iswd}$</td>
<td>Write data setup time before HCLK</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>$T_{isrdy}$</td>
<td>Ready setup time before HCLK</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>$T_{ismst}$</td>
<td>Master number setup time before HCLK (SPLIT-capable only)</td>
<td>n/a</td>
<td>Applies to modules with split capable slaves only</td>
</tr>
<tr>
<td>$T_{ismlck}$</td>
<td>Master locked setup time before HCLK (SPLIT-capable only)</td>
<td>n/a</td>
<td>Applies to modules with split capable slaves only</td>
</tr>
</tbody>
</table>

Table B-5 shows the AHB slave output parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Time (ns)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{ovrsp}$</td>
<td>Response valid time after HCLK</td>
<td>15</td>
<td>-</td>
</tr>
<tr>
<td>$T_{ovrd}$</td>
<td>Data valid time after HCLK</td>
<td>15</td>
<td>-</td>
</tr>
<tr>
<td>$T_{ovrdy}$</td>
<td>Ready valid time after HCLK</td>
<td>15</td>
<td>-</td>
</tr>
<tr>
<td>$T_{ovsplit}$</td>
<td>Split valid time after HCLK (SPLIT-capable only)</td>
<td>n/a</td>
<td>Applies to modules with split capable slaves only</td>
</tr>
</tbody>
</table>
Table B-6 shows the bus master input timing parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Time (ns)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{isgn}$</td>
<td>HGRANTx setup time before HCLK</td>
<td>5</td>
<td>Modules implementing masters only</td>
</tr>
<tr>
<td>$T_{isrd}$</td>
<td>Ready setup time before HCLK</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$T_{isrsp}$</td>
<td>Response setup time before HCLK</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$T_{isrd}$</td>
<td>Read data setup time before HCLK</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Table B-7 shows bus master output timing parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Time (ns)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{ovtr}$</td>
<td>Transfer type valid time after HCLK</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>$T_{ova}$</td>
<td>Address valid time after HCLK</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>$T_{ovctl}$</td>
<td>Control signal valid time after HCLK</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>$T_{ovwd}$</td>
<td>Write data valid time after HCLK</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>$T_{ovreq}$</td>
<td>Request valid time after HCLK</td>
<td>15</td>
<td>Modules implementing masters only</td>
</tr>
<tr>
<td>$T_{ovlck}$</td>
<td>Lock valid time after HCLK</td>
<td>15</td>
<td>Modules implementing masters only</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{isrd}$</td>
<td>Ready setup time before HCLK</td>
<td>5</td>
</tr>
<tr>
<td>$T_{isrsp}$</td>
<td>Response setup time before HCLK</td>
<td>5</td>
</tr>
<tr>
<td>$T_{isreq}$</td>
<td>Request setup time before HCLK</td>
<td>10</td>
</tr>
<tr>
<td>$T_{islck}$</td>
<td>Lock setup time before HCLK</td>
<td>10</td>
</tr>
</tbody>
</table>
Table B-9 shows the AHB arbiter output parameters. These only apply to the module or motherboard implementing the arbiter.

### Table B-9 AHB arbiter output parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tovgnt</td>
<td>Grant valid time after HCLK</td>
<td>15</td>
</tr>
<tr>
<td>Tovmst</td>
<td>Master number valid time after HCLK</td>
<td>15</td>
</tr>
<tr>
<td>Tovmlck</td>
<td>Master locked valid time after HCLK</td>
<td>15</td>
</tr>
</tbody>
</table>

#### B.2.3 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. The exact performance of a system depends on the timing parameters of the motherboard and all modules in the system. Some allowance also must be made for clock skew, routing delays and number of modules (that is, loading).

Not all FPGAs will meet the ideal timing parameters, due to the complexity of the design or routing congestion in 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 enables it to be mounted onto a Compact PCI motherboard while enabling the motherboard to be installed in a card cage.

Figure B-1 shows the mechanical outline of the core module.
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
Access arbitration, SDRAM 3-9, 5-6
Accesses
boot ROM 4-3
SDRAM DIMM 4-4
SSRAM 4-3
Address decoding, module 5-4
Alias SDRAM addresses 5-5, 5-6
ARM processor, overview 1-5
Assembled Integrator system 2-5
Auxiliary Oscillator Register 4-16
AUX_OD bits 4-17
AUX_RDW bits 4-17
AUX_VCW bits 4-17

B
Block diagram 1-5
Boot ROM, accesses 4-3
Bus bridge, system 5-7

C
Cache Operating Voltage Register 4-21
Cache Power-Down Voltage Register 4-22
Calculating the CORECLK speed 3-28
CAS latency, setting 4-19
Checking for valid SPD data 4-32
Chip selects, EBI 3-8
Clock generator 1-7, 3-25
Clock, reference 3-30
CM_AUXOSC Register 4-16
CM_CTL Register 4-4
CM_FIQ_ENCLR Register 4-13
CM_FIQ_ENCLR Register 4-28
CM_FIQ_ENSET register 4-28
CM_FIQ_RSTAT Register 4-28
CM_FIQ_STAT Register 4-28
CM_ID Register 4-11
CM_INIT Register 4-24
CM_IRQ_ENCLR Register 4-28
CM_IRQ_ENSET Register 4-28
CM_IRQ_RSTAT Register 4-28
CM_IRQ_STAT Register 4-28
CM_LMBUSCNT Register 4-16
CM_LOCK Register 4-15, 4-24
CM_OSC Register 3-28, 4-12
CM_PROC Register 4-11
CM_REFCNT Register 4-26
CM_SDRAM Register 3-10, 4-18
CM_SOFT_INTCLR Register 4-28
CM_SOFT_INTSET Register 4-28
CM_VOLTAGE_CTL0 Register 4-21
CM_VOLTAGE_CTL1 Register 4-21
CM_VOLTAGE_CTL2 Register 4-22
CM_VOLTAGE_CTL3 Register 4-22
CONFIG LED 3-31
CONFIG link 3-31
Configuration mode 3-34
Connecting Multi-ICE 2-3
Connecting power 2-3
Connectors
HDRA and HDRB 1-3
Index

logic analyzer A-13
Multi-ICE 2-4
power 2-3
Control Register 4-13
Controllers
clock 1-7, 3-25
FIQ 4-27
IRQ 4-27
reset 1-5, 3-10
SDRAM 1-5, 3-9
Core module
FPGA 1-5
ID 2-6
Core module ID
ID selection 5-4
ID signals 5-4
Core Operating Voltage Register 4-21
Core Power-Down Voltage Register 4-22
CORECLK 3-28

D
Debug comms channel 4-27
Debugging modes 3-34
DIMM socket 1-3

E
EBI chip selects 3-8
Electrical characteristics B-2
Ensuring safety 1-12
ETM 1-8
ETM10 3-38
Exception vector mapping 4-7

F
FIFOs 5-7
FIQ controller 4-27
FIQ Enable Clear Register 4-28, 4-29
FIQ Enable Set Register 4-28, 4-29
FIQ Raw Status Register 4-28, 4-29
FIQ Status Register 4-28, 4-29
Fitting SDRAM 2-2
Flag Clear Register 4-27
Flag Register 4-27
Flag registers 4-26
Flag Set Register 4-27
FPGA 1-5

G
Global SDRAM 5-5

H
HBUSSEL bits 4-25
HDRA and HDRB connectors 1-3
HDRA pinout A-2
HDRB plug pinout A-6
HDRB signals A-8
HDRB socket pinout A-4

I
ID bits, Core module ID, reading 4-14
ID Register 4-11
ID, core module 2-6
Initialization Register 4-24
Interrupt control 4-29
registers 4-27, 5-14
Interrupt Register
bit assignments 4-30
Interrupt, signal routing 5-14
IRQ and FIQ register bit assignment 4-30
IRQ controller 4-27
IRQ Enable Clear Register 4-28, 4-29
IRQ Enable Set Register 4-28, 4-29
IRQ Raw Status Register 4-28, 4-29
IRQ Status Register 4-28, 4-29

J
JTAG 3-31
connecting 2-3
debug 1-8
scan path 3-32
signals 3-35

L
Local Memory Bus Cycle Counter
Register 4-16
Local SDRAM DIMM 4-4
Location of connectors 1-3
Lock Register 4-15, 4-24
Logic analyzer connector 1-8, A-13

M
MBDET bit 4-13
Memory
map 4-2
volatile 1-6
MEMSIZE 4-19
Microprocessor core 3-2
MISC LED control 4-13
Module ID selection 5-4
Motherboard, attaching the core module
2-5
Multi-ICE 1-8, 3-31
connecting 2-3
coster 1-3

N
nEPRES signals 5-4
nMBDET signal 3-32
Nonvolatile Flag Clear Register 4-27
Nonvolatile Flag Registers 4-27
Nonvolatile Flag Set Register 4-27
Normal debug mode 3-34
nPRES signals 5-4

O
Operating mode, SDRAM 3-9
Oscillator Register 3-28, 4-12

P
Pinout
HDRA A-2
HDRB plug A-6
HDRB socket A-4
Power
  cache voltage polarity 4-23
  connector 1-3, 2-3
  management controller 1-5
  supplies 1-8
Powering an attached core module 2-6
Precautions 1-12
Preventing damage 1-12, 3-24
Processor
  core clock 3-28
  reads 5-8
  register 4-11
  writes 5-7
Protection
  cache voltage polarity 3-24, 4-23

R
Reads by the processor from the system bus 5-8
REFCLK 3-25, 3-30
Reference clock 3-30
Reference Clock Cycle Counter Register 4-26
Register addresses 4-8
Registers 1-6, 4-8
  Auxiliary Oscillator 4-16
  Cache Operating Voltage 4-21
  Cache Power-Down Voltage 4-22
  CM_AUXOSC 4-16
  CM_CTRL 4-4
  CM_CTRL 4-13
  CM_FIQ_ENCLR 4-28
  CM_FIQ_ENSET 4-28
  CM_FIQ_RSTAT 4-28
  CM_FIQ_STAT 4-28
  CM_ID 4-11, 5-6
  CM_INIT 4-24
  CM_IRQ_ENCLR
    Registers
      IRQ Enable Clear 4-28
    CM_IRQ_ENSET 4-28
    CM_IRQ_RSTAT 4-28
    CM_LMBUSCNT 4-16
    CM_LOCK 4-15, 4-24
    CM_OSC 3-28, 4-12
    CM_PROC 4-11
CM_REFCNT 4-26
CM_SDRAM 3-10, 4-18
CM_SOFT_INTCLR 4-28
CM_SOFT_INTSET 4-28
CM_STAT 4-14
CM_VOLTAGE_CTL0 Register 4-21
CM_VOLTAGE_CTL1 Register 4-21
CM_VOLTAGE_CTL2 Register 4-22
CM_VOLTAGE_CTL3 Register 4-22
Control 4-4, 4-13
Core Operating Voltage 4-21
Core Power-Down Voltage 4-22
FIQ Enable Clear 4-28, 4-29
FIQ Enable Set 4-28, 4-29
FIQ Raw Status 4-28, 4-29
FIQ Status 4-28, 4-29
Flag 4-26, 4-27
Flag Clear 4-27
Flag Set 4-27
ID 4-11
Identification 5-6
Initialization 4-24
Interrupt Control 4-27, 4-30
IRQ Enable Clear 4-29
IRQ Enable Set 4-28, 4-29
IRQ Raw Status 4-28, 4-29
IRQ Status 4-28, 4-29
Local Memory Bus Cycle Counter 4-16
Lock 4-15, 4-24
Nonvolatile Flag 4-27
Nonvolatile Flag Clear 4-27
Nonvolatile Flag Set 4-27
Oscillator 3-28, 4-12
Processor 4-11
Reference Clock Cycle Counter 4-26
SDRAM Control and Status 3-10
SDRAM Status and Control 4-18
Soft Interrupt Clear 4-31
Soft Interrupt Set 4-31
Software Interrupt Clear 4-28
Software Interrupt Set 4-28
Status 4-14
REMAP bit 4-13
Remap, effect of 4-4
Reset control bit 4-13
Reset controller 1-5, 3-10

S
SDRAM 4-32, 5-9
  access arbitration 3-9
  accesses to DIMM 4-4
  Control and Status Register 3-10
  controller 1-5, 3-9
  fitting 2-2
  global access 5-5
  operating mode 3-9
  operating without 2-2
  private 4-5
  repeat mapping of DIMM 4-5
  SPD memory 4-32
  Status and Control Register 4-18
Serial Presence Detect 3-10
Setting CAS latency 4-19
Setting SDRAM size 4-19
Setup
  power connections 2-3
  standalone 2-2
Signal routing, interrupts 5-14
Signals
  ID 5-4
  nEPRES 5-4
  nPRES 5-4
SI_ID bits 4-14
Soft Interrupt Clear Register 4-31
Soft Interrupt Set Register 4-31
Software Interrupt Clear Register 4-28
Software Interrupt Set Register 4-28
Software reset 4-13
SPD 3-10, 4-32
SPDOK bit 4-19
SSRAM accesses 4-3
SSRAMSIZE bits 4-14
Stacking 3-40
Status and configuration registers 1-6
Status Register 4-14
Supplying power 2-3
System
  architecture 1-5
System bus
Index

bridge 1-6, 5-7
reads of the SDRAM 5-11
signal routing 5-11
writes to SDRAM 5-9

T
TDI signal 3-32
Test chip, overview 1-5
Through-board signals A-7
Trace
  connection 3-38
  connectors 1-8
  core configuration 3-39
  interface description 3-39
  support 3-38
  using 3-38

U
Using the core module with a
  motherboard 2-5

V
VCO divider 3-28
Vector mapping 4-7
VINITHI bit 4-26
Volatile memory 1-6
Voltage
  cache polarity protection 3-24, 4-23
  configuration registers 4-20

W
Writes by the processor to the system
  bus 5-7