Integrator®/CM926EJ-S, CM946E-S, CM966E-S, CM1026EJ-S, and CM1136JF-S

HBI-0087: CM926EJ-S, CM1026EJ-S, and CM1136JF-S HBI-0066: CM946E-S and CM966E-S

User Guide

ARM®

ARM DUI 0138E

Integrator/CM926EJ-S, CM946E-S, CM966E-S, CM1026EJ-S, and CM1136JF-S
User Guide

Release Information

Description	Issue	Change
26 September 2000	A	New document
9 November 2002	B	Second release
19 November 2003	C	Third release. Updated clock register information
2 March 2004	D	Added CM1026 and CM1136
19 April 2005	E	Fourth release. Corrected contents for CM1136JF-S cache, HBUSREQ status, HCLK reset, and SRAM FIFO

Proprietary Notice

Confidentiality Status

Product Status

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

Web Address

http://www.arm.com

Conformance Notices

This section contains conformance notices.

Federal Communications Commission Notice

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

CE Declaration of Conformity

The system should be powered down when not in use.

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

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

Note

It is recommended that wherever possible shielded interface cables be used.

Contents
Integrator/CM926EJ-S, CM946E-S, CM966E-S, CM1026EJ-S, and CM1136JF-S User Guide

Preface

This preface introduces the ARM Integrator/CM926EJ-S, Integrator/CM946E-S, Integrator/CM966E-S, Integrator/CM1026EJ-S, and Integrator/CM1136JF-S core modules and their reference documentation. It contains the following sections:

About this document

This document describes how to set up and use the ARM Integrator/CM926EJ-S, Integrator/CM946E-S, Integrator/CM966E-S, Integrator/CM1026EJ-S, and Integrator/CM1136JF-S 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 ARM Integrator/CM926EJ-S, Integrator/CM946E-S, Integrator/CM966E-S, Integrator/CM1026EJ-S, and Integrator/CM1136JF-S core modules as part of a development system.

Organization

This document is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an introduction to the core module.

Chapter 2 ***Getting Started***

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

Chapter 3 ***Hardware Description***

Read this chapter for a description of the hardware architecture of the core module. This includes clocks and debug.

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***

Refer to this chapter for information on using the core module as part of an Integrator/AP system.

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

Refer to this chapter for information on using the core module as part of an Integrator/CP system.

Appendix A ***Signal Descriptions***

Refer to this appendix for connector pinouts.

Appendix B *Specifications*

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

Appendix C *Features specific to the CM1136JF-S*

Refer to this appendix for details of features that are specific to the CM1136JF-S such as ETB and additional clock control registers.

Typographical conventions

The following typographical conventions are used in this document:

bold	Highlights ARM signal names within text, and interface elements such as menu names. This style is also used for emphasis in descriptive lists where appropriate.
italic	Highlights special terminology, cross-references and citations.
`monospace`	Denotes text that can be entered at the keyboard, such as commands, file names and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
`monospace italic`	Denotes arguments to commands or functions where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.

Further reading

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

ARM publications

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

- *ARM926EJ-S Technical Reference Manual* (ARM DDI 0198)
- *ARM946E-S Technical Reference Manual* (ARM DDI 0155)
- *ARM966E-S Technical Reference Manual* (ARM DDI 0164)
- *ARM1026EJ-S Technical Reference Manual* (ARM DDI 0224)
- *ARM1136JF-S Technical Reference Manual* (ARM DDI 0211)
- *ETM9 Technical Reference Manual* (ARM DDI 0157)
- *ETM10 Technical Reference Manual* (ARM DDI 0206)
- *ETB Technical Reference Manual* (ARM DDI 0242)
- *ARM Integrator/AP User Guide* (ARM DUI 0098)
- *ARM Integrator/CP User Guide* (ARM DUI 0159)

- *ARM Multi-ICE User Guide* (ARM DUI 0048)
- *AMBA Specification* (ARM IHI 0011)
- *ARM Architecture Reference Manual* (ARM DDI 0100)
- *ARM Firmware Suite Reference Guide* (ARM DUI 0102)
- *ADS Tools Guide* (ARM DUI 0067)
- *ADS Debuggers Guide* (ARM DUI 0066)
- *ADS Debug Target Guide* (ARM DUI 0058)
- *ADS Developer Guide* (ARM DUI 0056)
- *ADS CodeWarrior IDE Guid*e (ARM DUI 0065)
- *RealView Debugger User Guid*e (ARM DUI 0153)
- *RealView Compilers and Libraries Guid*e (ARM DUI 0205)
- *RealView Linker and Utilities Guid*e (ARM DUI 0206).
- *RealView ICE User Guide* (ARM DUI 0155).

Other publications

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

CM926EJ-S, CM1026EJ-S, and CM1136JF-S

MicroClock ICS307 Data Sheet (MDS307), MicroClock Division of ICS, San Jose, CA.

CM946E-S and CM966E-S

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

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

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

Feedback

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

Feedback on this document

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

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

General suggestions for additions and improvements are also welcome.

Feedback on the ARM Integrator core modules

If you have any comments or suggestions about these products, contact your supplier giving:

- the product name
- an explanation of your comments.

Chapter 1
Introduction

This chapter introduces the ARM Integrator/CM926EJ-S, Integrator/CM946E-S, Integrator/CM966E-S, Integrator/CM1026EJ-S, and Integrator/CM1136JF-S core modules. It contains the following sections:

1.1 About the core modules

The core modules are a compact development platform that enable you to develop products based on the ARM926EJ-S™, ARM946E-S™, ARM966E-S™, ARM1026EJ-S™, or ARM1136JF-S™processors.

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

The core module can be used in the following ways:

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

Note

The CM946E-S and CM966E-S core modules both use the same printed circuit board (HBI-0066) and differ only in the test chip fitted.

The CM926EJ-S, CM1026EJ-S, and CM1136JF-S all use the same printed circuit board (HBI-0087) and only differ in the test chip fitted.

1.2 Overview of the Integrator/CMx6

The major components on the core module are as follows:

- ARM926EJ-S, ARM946E-S, ARM966E-S, ARM1026EJ-S, or ARM1136JF-S microprocessor core
- volatile memory comprising:
  - up to 256MB of SDRAM (optional) plugged into the DIMM socket
  - 1MB SSRAM.
- core module FPGA that implements:
  - SDRAM controller
  - system bus bridge
  - reset controller
  - interrupt controller
  - status, configuration, and interrupt registers
  - power supply control (only CM926EJ-S, CM1026EJ-S, or CM1136JF-S).
- SSRAM controller PLD (CM946E-S and CM966E-S only)
- clock generator
- Integrator system bus connectors
- Multi-ICE debug connector
- logic analyzer connectors for local memory bus
- controllable power supply (only CM926EJ-S, CM1026EJ-S, or CM1136JF-S)
- Trace port.

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

Figure 1-1 on page 1-4 shows the layout of the ARM Integrator/CM946E-S and CM966E-S.

Figure 1-2 on page 1-5 shows the layout of the ARM Integrator/CM926EJ-S, CM1026EJ-S, and CM1136JF-S.

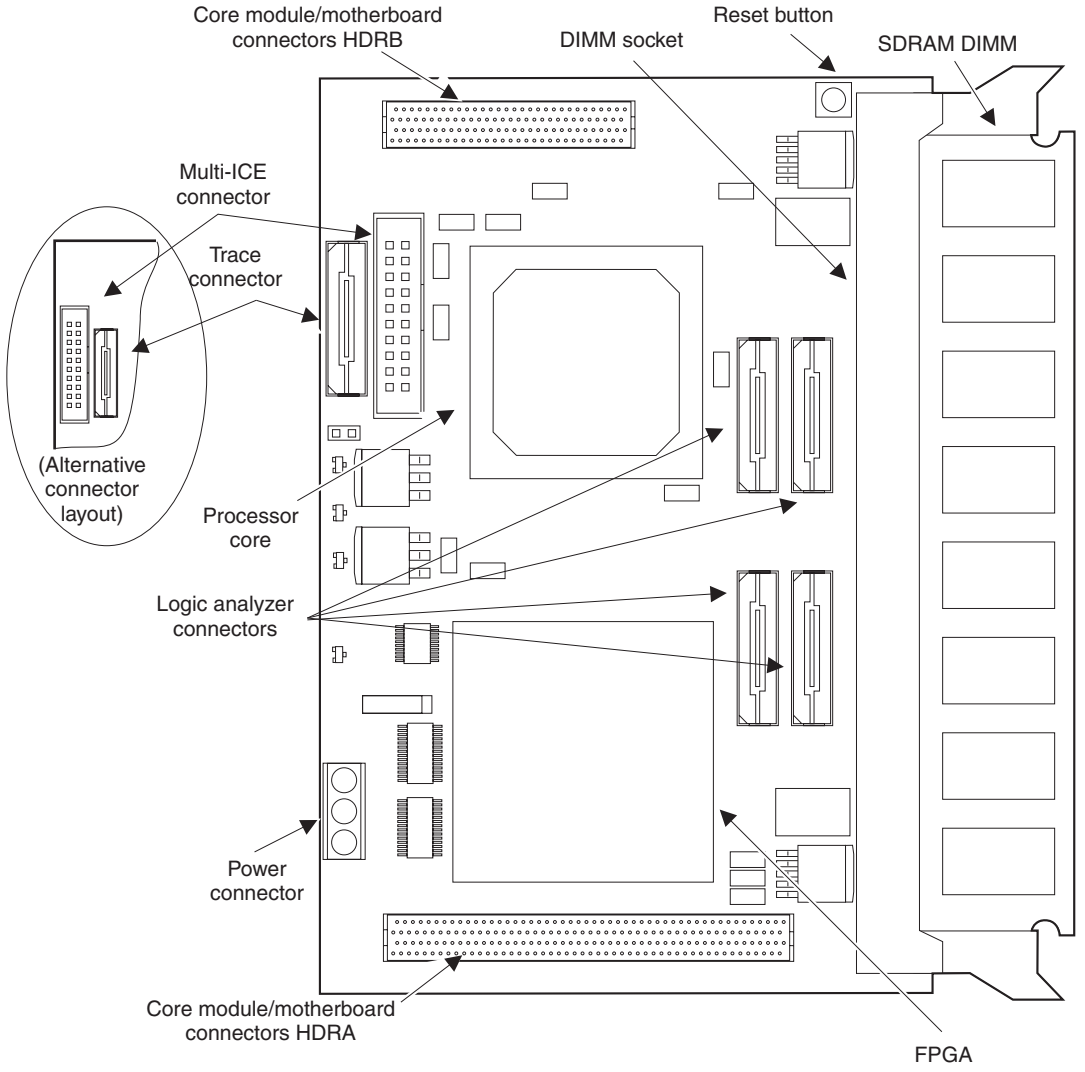


Figure 1-1 Integrator/CM946E-S and CM966E-S layout

Note

The positions of the Multi-ICE and Trace port analyzer are reversed on some board versions.

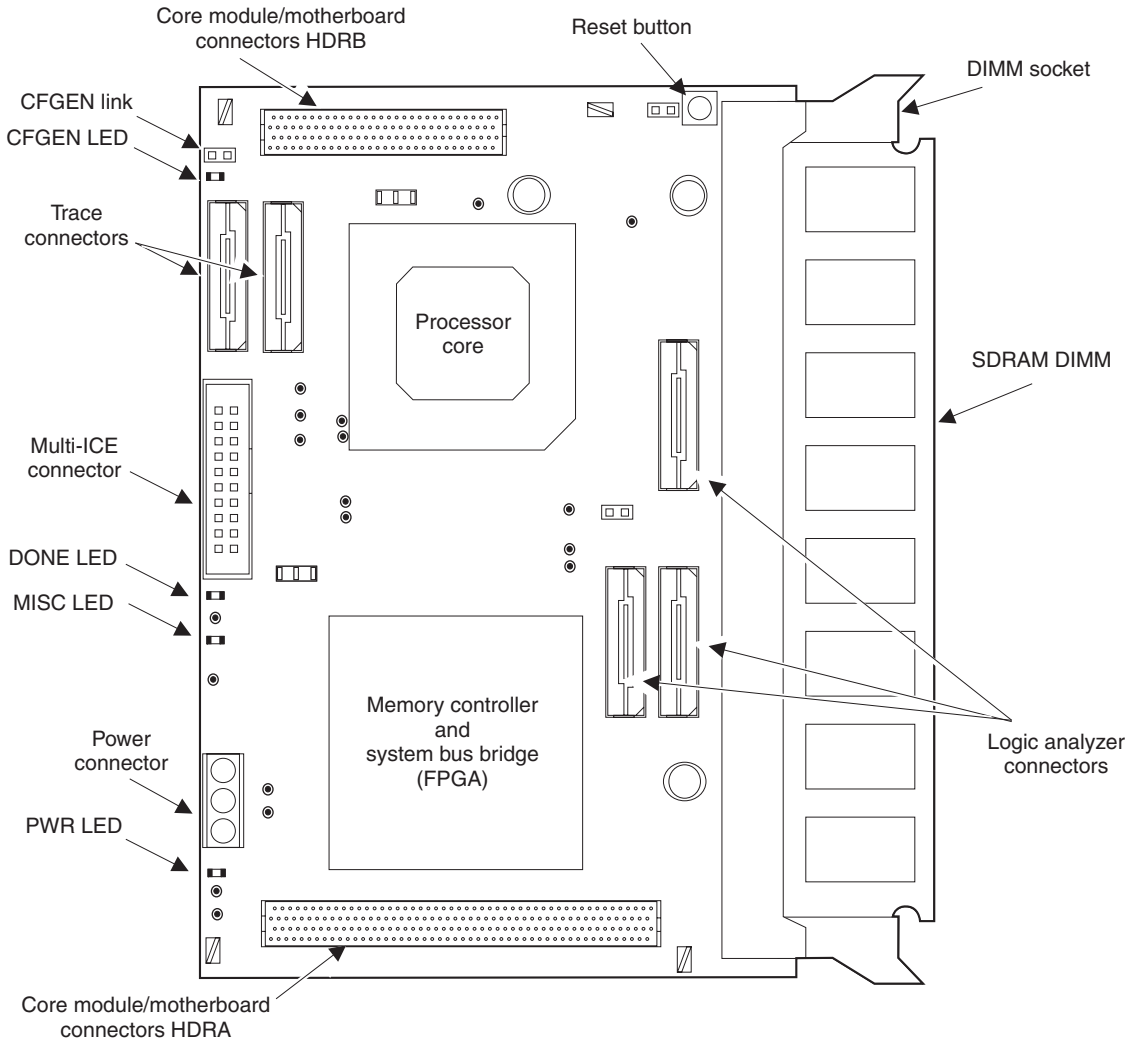


Figure 1-2 Integrator/CM926EJ-S, CM1026EJ-S, and CM1136JF-S layout

1.2.1 System architecture

Figure 1-3 shows the architecture of the core module.

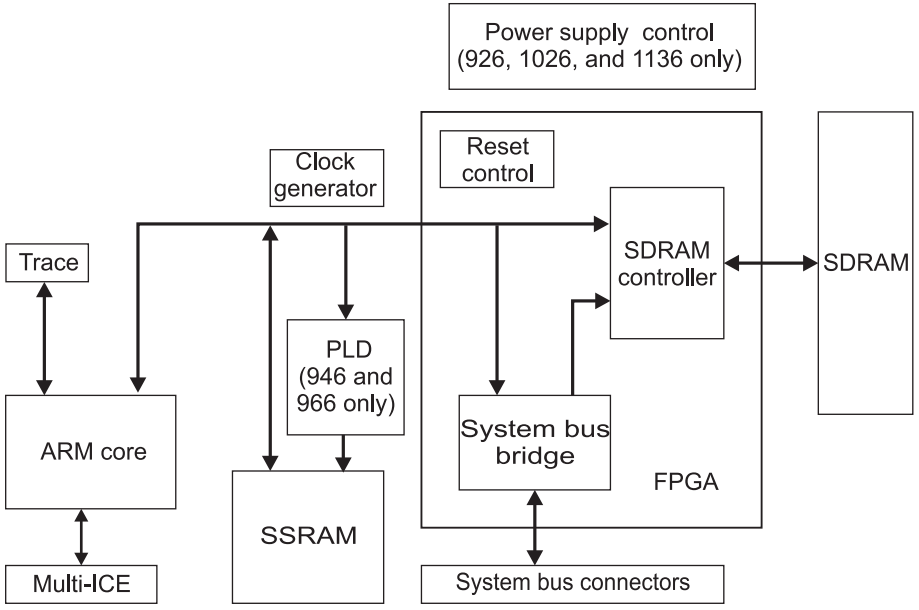


Figure 1-3 Core module block diagram

Note

The FPGA image for the Integrator/CP is different from the image for the Integrator/AP. For more information on using the core module with an Integrator motherboard, see Chapter 5 *Using Core Modules with an Integrator/AP* and Chapter 6 *Using Core Modules with an Integrator/CP*.

1.2.2 ARM processor test chip

The test chip fitted into the core module is:

- Integrator/CM926EJ-S uses the ARM926EJ-S
- Integrator/CM946E-S uses the ARM946E-S
- Integrator/CM966E-S uses the ARM966E-S
- Integrator/CM1026EJ-S uses the ARM1026EJ-S
- Integrator/CM1136JF-S uses the ARM1136JF-S.

For a brief description of these processors, see *ARM microprocessor test chip* on page 3-2.

You can configure all of these processor types using several input signals. In a final product, these signals are permanently tied HIGH or LOW. However, the Integrator allows you to reprogram these inputs in order to experiment with different processor configurations (see *Test chip configuration control* on page 3-4).

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*.

Note

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

SDRAM controller

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

Reset controller

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

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

For information about the reset controller (see *Reset controller* on page 5-20 for the Integrator/AP or *Reset controller* on page 6-27 for the Integrator/CP).

System bus bridge

The system bus bridge provides an AMBA AHB interface between the memory bus on the core module and the system bus on a motherboard. It allows the processor to access resources on the motherboard and on other modules. It also allows other masters to access the core module SDRAM (see *System bus bridge* on page 5-13).

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 size, address configuration, and CAS latency setup
- core module clock speed and configuration
- power supply control registers (CM926EJ-S, CM1026EJ-S, and CM1136JF-S only)
- interrupt control for the processor debug communications channel.

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

1.2.4 Volatile memory

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

The SDRAM controller is implemented within the core module controller FPGA. For the CM926EJ-S, CM1026EJ-S, and CM1136JF-S, the SSRAM controller is also implemented within the core module FPGA. For the CM946E-S and CM966E-S however, a separate SSRAM controller is implemented with a *Programmable Logic Device* (PLD). (The Integrator/CP does not use a SSRAM PLD controller.) The SSRAM can only be accessed by the local processor.

Note

The CM926EJ-S, CM946E-S, CM966E-S, CM1026EJ-S, and CM1136JF-S all feature *Tightly Coupled SRAM Memory* (TCM) inside the test chip (see *About the memory map* on page 4-2). The size of the TCM depends on the design of the test chip installed in the core module.

1.2.5 Clock generation

The naming of the clocks generated by the core module differs between the CM926EJ-S, CM1026EJ-S, and CM1136JF-S boards and the CM946E-S and CM966E-S boards.

For the CM926EJ-S, CM1026EJ-S, and CM1136JF-S, the clocks are:

REF24MHZ A fixed frequency 24MHz signal that can be used by a custom peripheral to generate real-time delays.

ARM_PLLCLKIN A programmable frequency reference clock input to the PLL in the ARM test chip. This signal is called CORECLK at the output of the clock generator. The signal passes without modification through the FPGA and is then fed to the test chip.

BUSCLK A programmable frequency clock on the Integrator/CP. This is reserved for future use on the Integrator/AP.

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

For the CM946E-S and CM966E-S, the clocks are:

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

ARM_PLLCLKIN A programmable frequency reference clock input to the PLL in the ARM test chip.

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

The programmable clocks are supplied by two clock generator chips (or by three clock generator chips for the CM926EJ-S, CM1026EJ-S, and CM1136JF-S). Their frequencies are set by programming the oscillator control registers within the FPGA. A fixed-frequency reference clock is supplied to the two clock generators and to the FPGA (see *Clock generators* on page 3-10).

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

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

1.2.6 Multi-ICE connector

The Multi-ICE connector enables JTAG hardware debugging equipment, such as Multi-ICE, to be connected to the core module. You can drive and sense the system-reset line (**nSRST**), and drive JTAG reset (**nTRST**) to the core from the Multi-ICE connector (see *Multi-ICE support* on page 3-17).

1.2.7 Power supply control

For the CM926EJ-S, CM1026EJ-S, and CM1136JF-S only, the power supply control enables you to control the core voltage from software. (See *Power supply control (CM926EJ-S, CM1026EJ-S, and CM1136JF-S only)* on page 3-31.) The voltage control range is set when the board is manufactured. The control circuitry also allows you to read different voltages on the board (including voltages indicating the current consumption on the core power rails).

1.2.8 CP peripherals

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

1.3 Links and indicators

The CM946E-S and CM966E-S core modules provide one user-configurable link and four surface-mounted LEDs (see Figure 1-4).

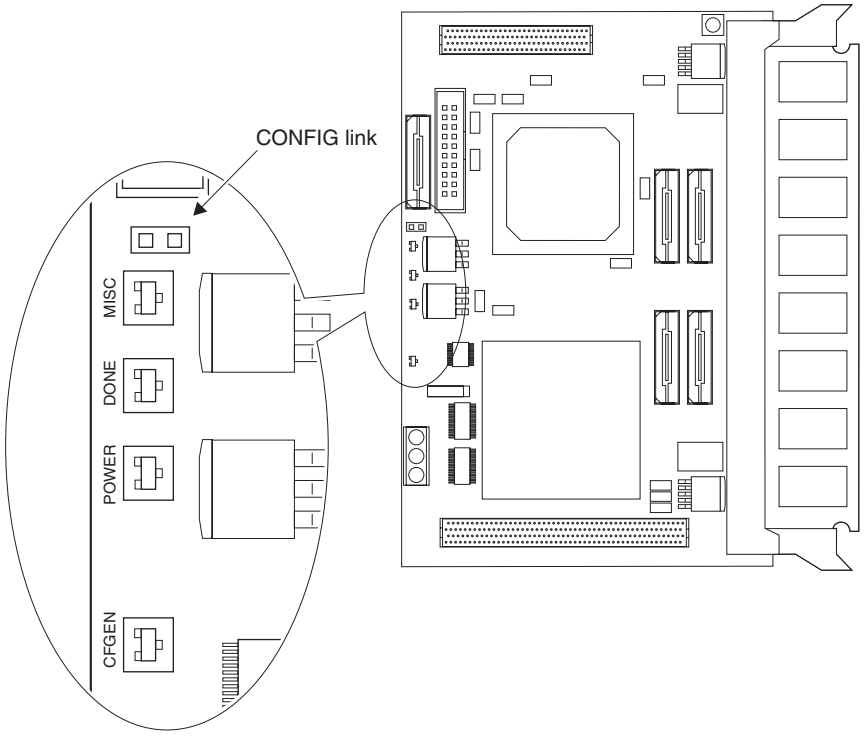


Figure 1-4 Links and indicators for the CM946E-S and CM966E-S

The CM926EJ-S, CM1026EJ-S, and CM1136JF-S core modules provide one user-configurable link and four surface-mounted LEDs (see Figure 1-5 on page 1-12).

Note

The SSRAM VIO and LK1 links are set during manufacture and are not typically changed by the user.

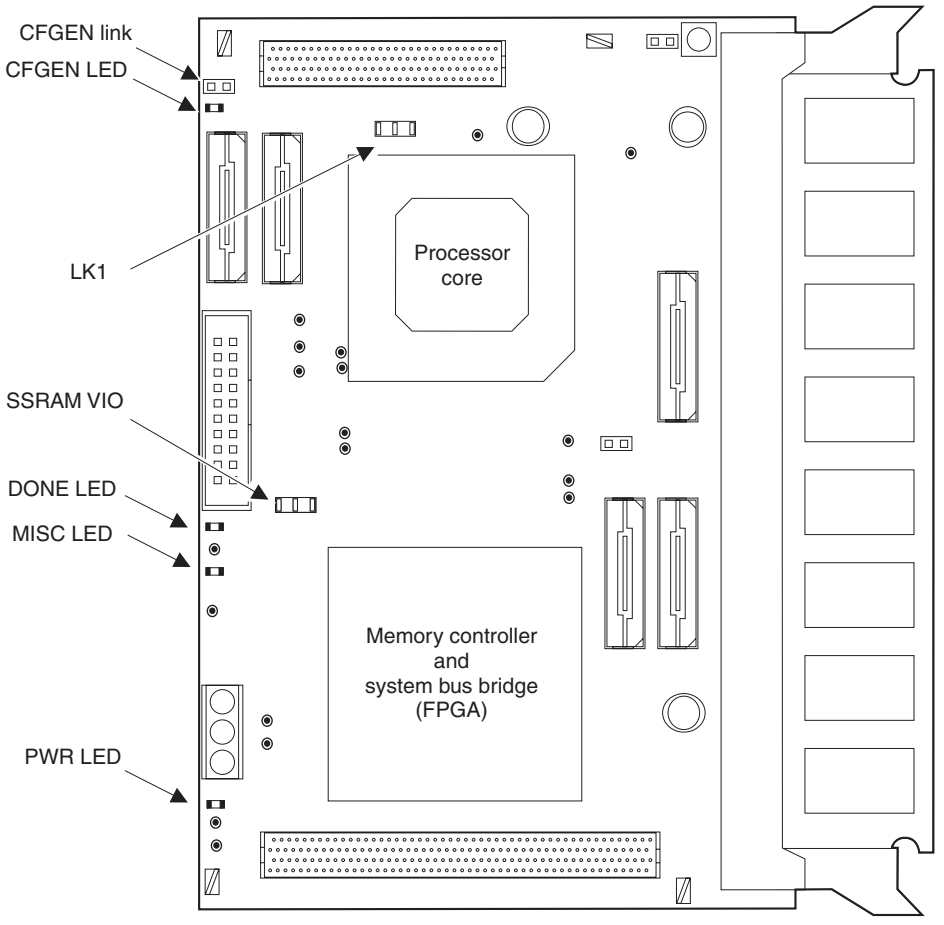


Figure 1-5 Links and indicators for the CM926EJ-S, CM1026EJ-S, and CM1136JF-S

1.3.1 CONFIG link

The core modules have only one user-selectable link, known as the CONFIG link (shown in Figure 1-4 on page 1-11). This is left open during normal operation. Only fit this link to download new FPGA and PLD configurations (see *Multi-ICE support* on page 3-17).

1.3.2 LED indicators

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

Table 1-1 LED functional summary

Name	Color	Function
MISC	Green	This LED is controlled by the LED bit in the CM_CTRL register (see *Core module control register* on page 4-17).
DONE	Green	This LED is lit when the FPGA has successfully loaded its configuration information following power-on.
POWER	Green	This LED is lit to indicate that a 3.3V supply is present.
CFGEN	Orange	This LED is lit to indicate that the CONFIG link is fitted. Fitting this link to the core module at the top of the stack places all modules in the stack in CONFIG mode (see Debugging modes on page 3-23).

1.4 Test points

Figure 1-6 shows the test points and ground points on the CM946E-S and CM966E-S core modules.

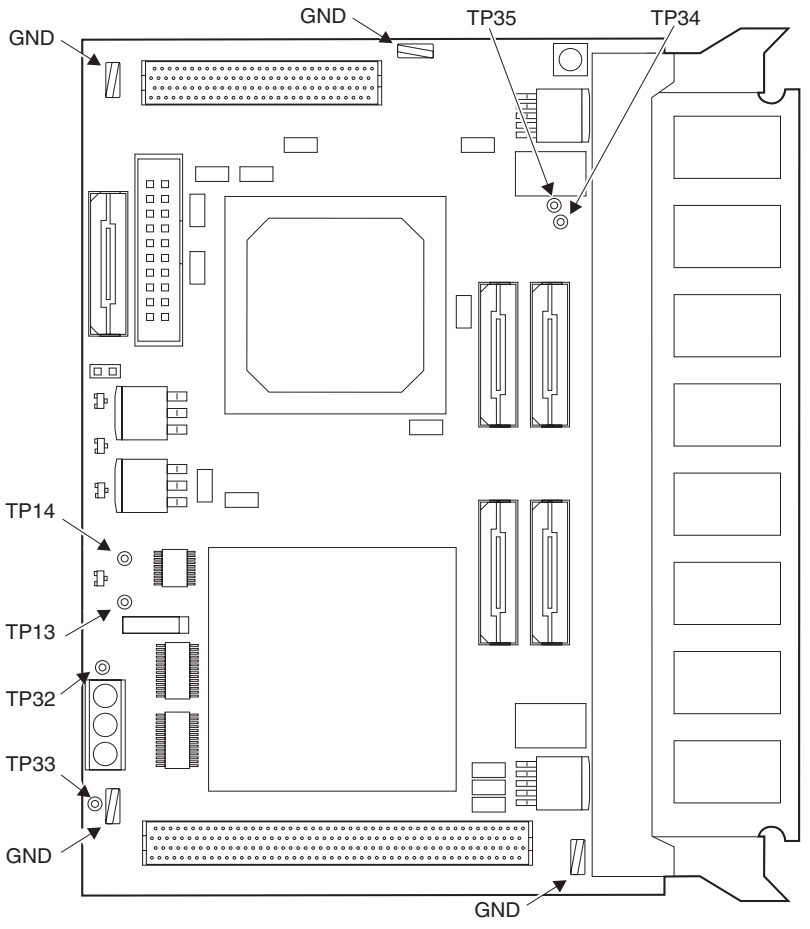


Figure 1-6 Test points (CM946E-S and CM966E-S)

Figure 1-7 on page 1-15 shows the test points and ground points on the CM926EJ-S, CM1026EJ-S, and CM1136JF-S core modules.

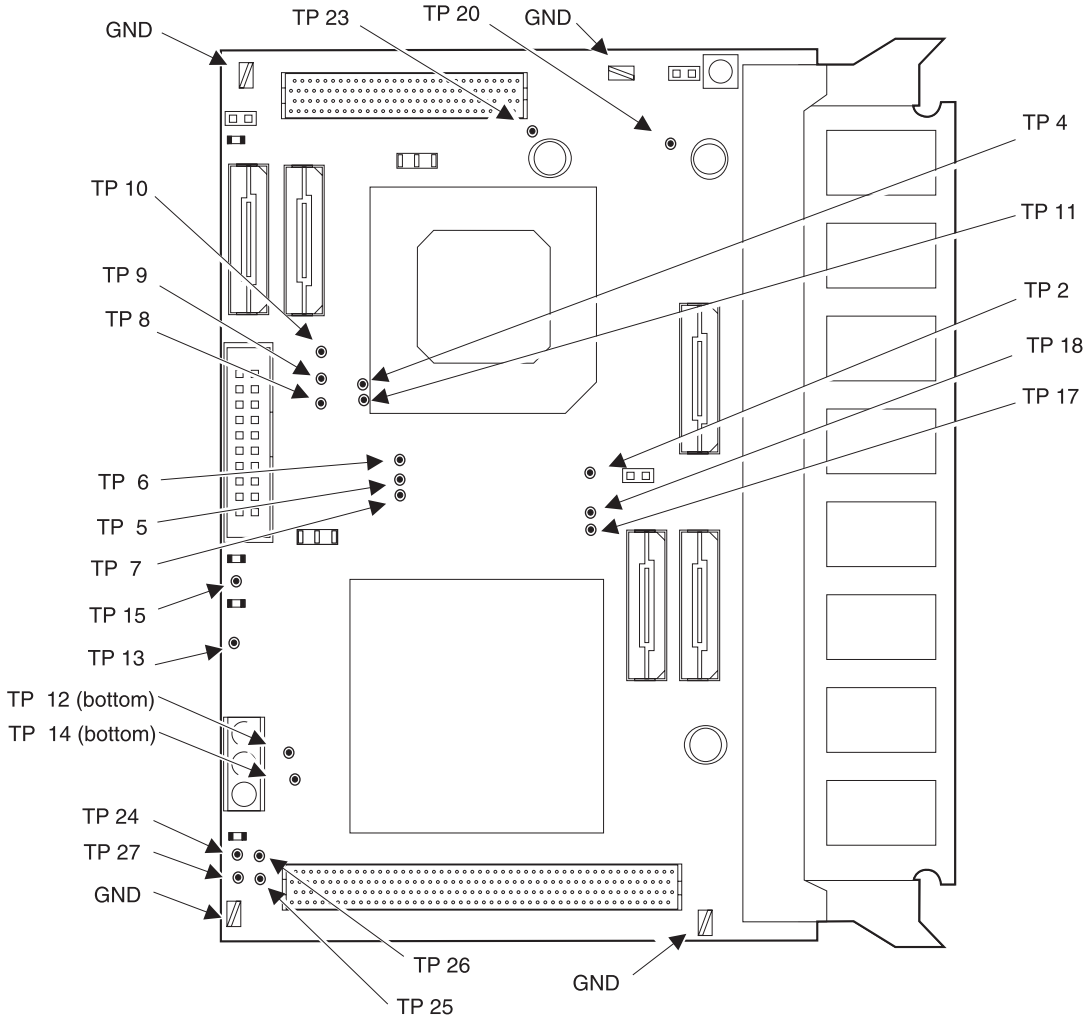


Figure 1-7 Test points (CM926EJ-S, CM1026EJ-S, and CM1136JF-S)

The functions of the test points are shown in Table 1-2 and Table 1-3. (Test points not mentioned in the tables are for manufacturing use only.)

Table 1-2 Test point functions (CM946E-S and CM966E-S)

Test point	Name	Function
TP13	**PLLCLKIN**	Input clock to core
TP14	**REFCLK**	Reference clock (24MHz)
TP32	**AUXCLK**	Auxiliary clock
TP33	**PLLLOCK**	Phase locked (if supported by the test chip)
TP34	**SDRAM HCLK**	Buffered **HCLK** to SDRAM DIMM
TP35	**HCLK**	**HCLK** output from test chip

Table 1-3 Test point functions (CM926EJ-S, CM1026EJ-S, and CM1136JF-S)

Test point	Name	Function
TP12	**CORECLK**	Input clock to core module
TP13	**REF24MHZ**	Reference clock (24MHz)
TP16	**HCLK**	HCLK output from test chip
TP15	**AUXCLK**	Auxiliary clock
TP11	**PLLLOCK**	Phase locked (if supported by the test chip)
TP14	**BUSCLK**	Input to test chip

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 *Safety Extra Low Voltage* (SELV) equipment must be connected to the module.

1.5.2 Preventing damage

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

Caution

To avoid damage to the board you must 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.
- If you are using the core module with a motherboard, do not connect power to the core module power connector.

Caution

Do not use the board near equipment that could be:

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

1.5.3 Ensuring correct operation

The core module FPGA must have an appropriate image for the Integrator/AP or Integrator/CP. Ensure that the correct image is loaded and selected.

Chapter 2
Getting Started

This chapter describes how to set up and prepare an ARM Integrator/CM926EJ-S, Integrator/CM946E-S, Integrator/CM966E-S, Integrator/CM1026EJ-S, or Integrator/CM1136JF-S core module for use. It contains the following sections:

2.1 Setting up a standalone core module

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

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

2.1.1 Fitting an SDRAM DIMM

You can fit the following type of SDRAM module:

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

To install an SDRAM DIMM:

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

Note

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

2.1.2 Using the core module without SDRAM

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

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

2.1.3 Connecting power

When using the core module as a standalone development system, you must connect a bench power supply with 3.3V and 5V outputs to the power connector, as illustrated in Figure 2-1. For information about power consumption by the core module, see *Electrical specification* on page B-2.

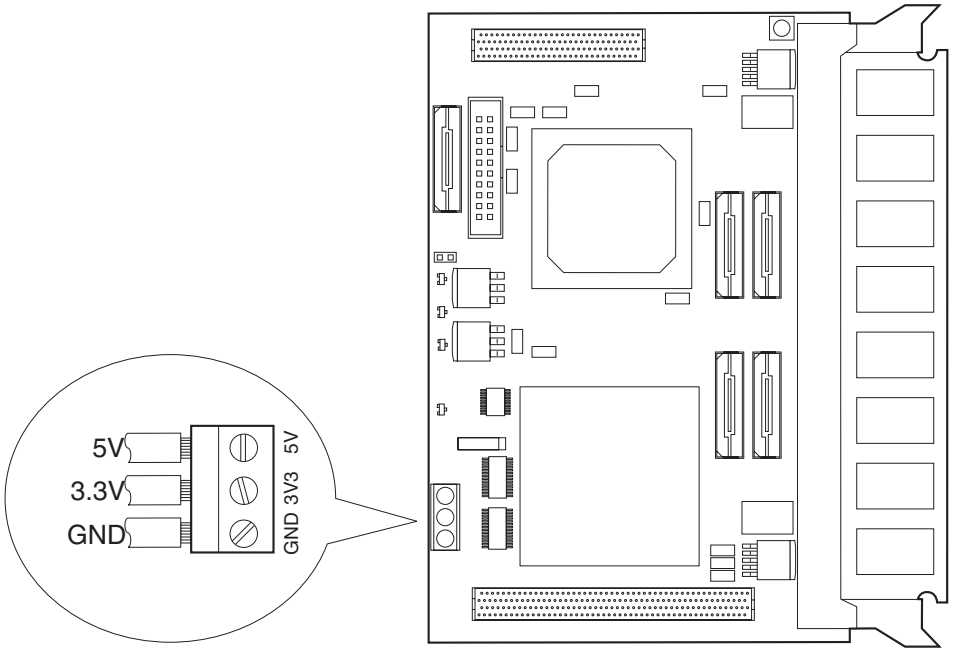


Figure 2-1 Power connector

Caution

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

2.1.4 Connecting Multi-ICE or RealView ICE and Multi-Trace

When you are using the core module as a standalone system, the RealView ICE or Multi-ICE debugging equipment can be used to download programs. The Multi-ICE setup for a standalone CM946E-S or CM966E-S core module is shown in Figure 2-2. (The setup for a CM926EJ-S, CM1026EJ-S, and CM1136JF-S is similar, but the connector location is slightly different.)

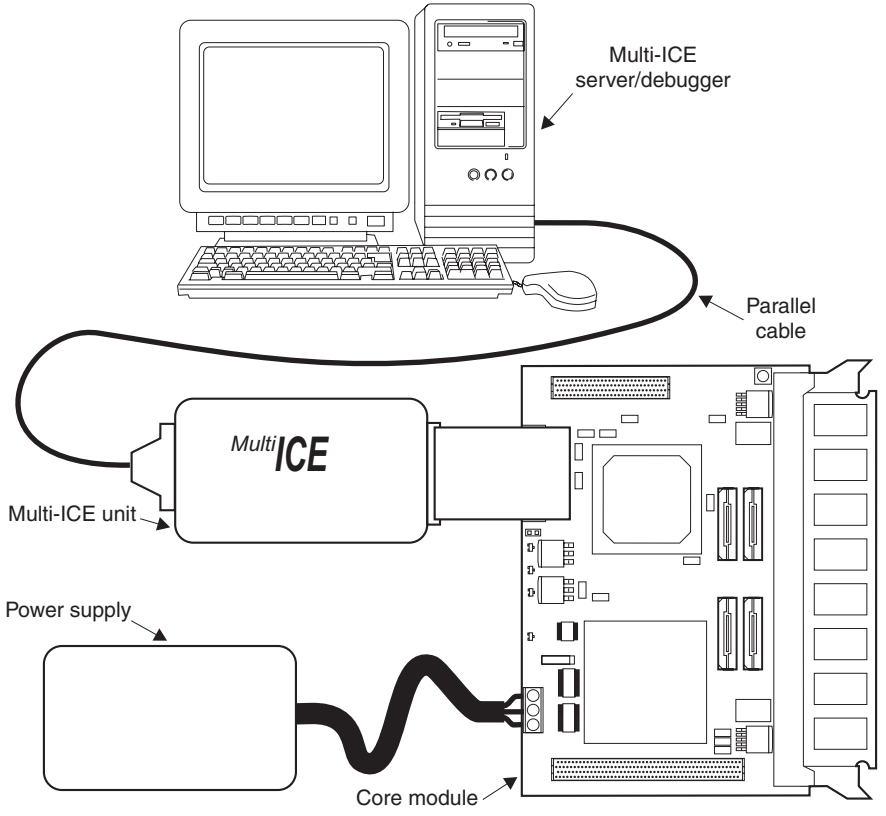


Figure 2-2 Multi-ICE connection to a core module

Caution

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

Some core modules can operate with a CPU clock faster than the trace unit can capture. If tracing becomes unreliable, try reducing the CPU clock.

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

Note

Some board versions have the Multi-ICE (RealView ICE) and Trace connectors reversed as shown in Figure 2-3. The CM926EJ-S, CM1026EJ-S, and CM1136JF-S connectors are placed as shown in Figure 2-3.

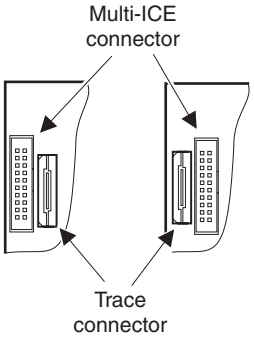


Figure 2-3 Alternative connector layout (CM946E-S and CM966E-S)

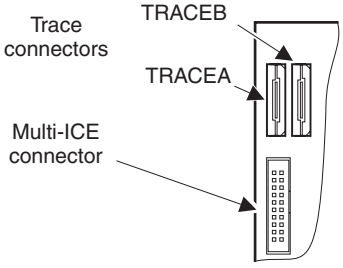


Figure 2-4 Connector layout for CM926EJ-S, CM1026EJ-S, and CM1136JF-S

If you are using Trace for the CM946E-S or CM966E-S, connect the trace port adapter board to the target board as shown in Figure 2-5 on page 2-6.

If you are using Trace for the CM926EJ-S, CM1026EJ-S, or CM1136JF-S and are using a single trace cable, connect the trace port adapter board to the TRACEA port shown in Figure 2-4. See *Trace connector pinout* on page A-9 for details of the trace connector signals.

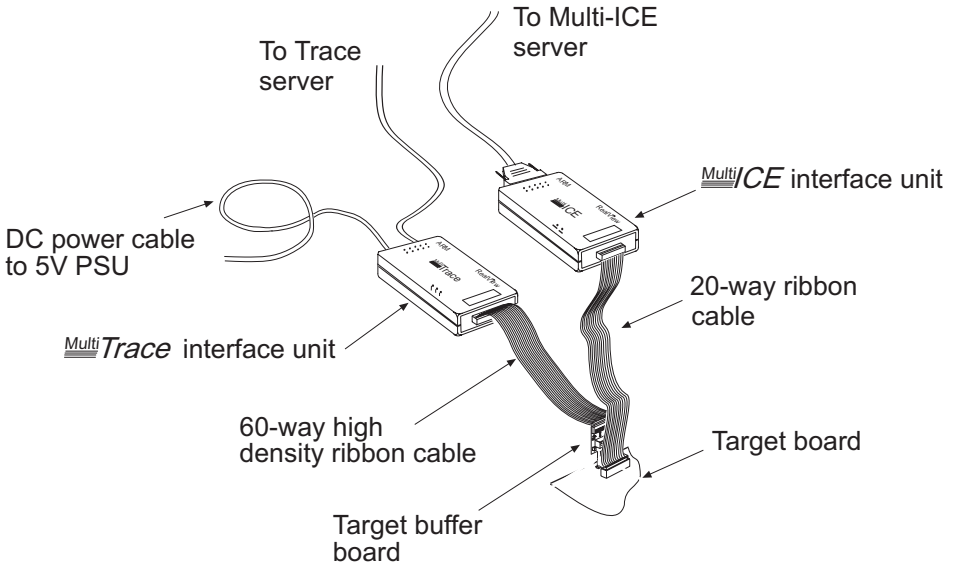


Figure 2-5 Connecting Trace

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

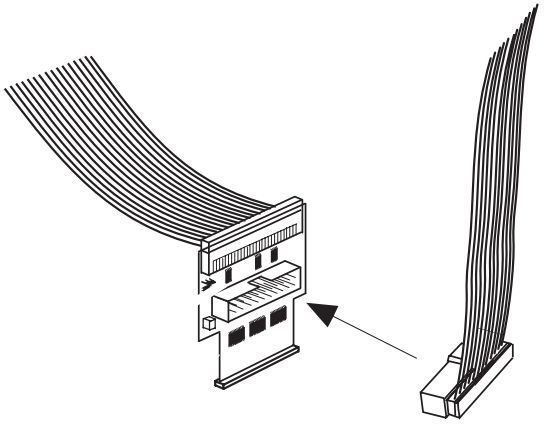


Figure 2-6 Connecting Multi-ICE to the trace port adapter board

2.2 Attaching the core module to an Integrator/AP motherboard

Attach the core module onto an Integrator/AP motherboard by engaging the connectors HDRA and HDRB on the bottom of the core module with the corresponding connectors on the top of the motherboard. The lower side of the core module has sockets and the upper side of the core module has plugs to allow core modules to be mounted on top of one another. A maximum of four core modules can be stacked on an Integrator/AP motherboard.

Figure 2-7 illustrates an example development system with four core modules attached to a motherboard.

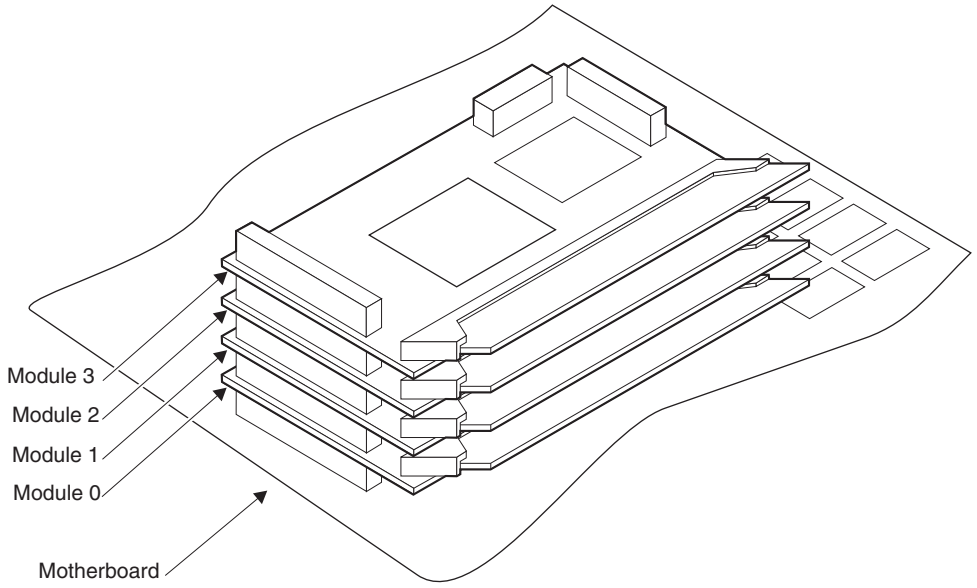


Figure 2-7 Assembled Integrator system

Note

To ensure reliable operation of the core module:

- Do not use the core module in the EXPA/EXPB stack position on the Integrator/AP.
- Configure the motherboard for AHB operation. The core modules only support an AHB system bus interface.

2.2.1 Core module ID

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

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

The ID of the core module also defines the address of SDRAM in the alias memory region of the system memory map. 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 *Core module status register* on page 4-15).

2.2.2 SDRAM, Trace, and Multi-ICE

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

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

2.2.3 Powering the assembled Integrator/AP development system

Power the assembled Integrator development system by:

- connecting a bench power supply to the motherboard (not to the core module)
- installing the motherboard in a card cage or an ATX-type PC case, depending on type.

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

2.3 Attaching the core module to an Integrator/CP baseboard

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

Figure 2-8 illustrates an example Integrator/CP development system.

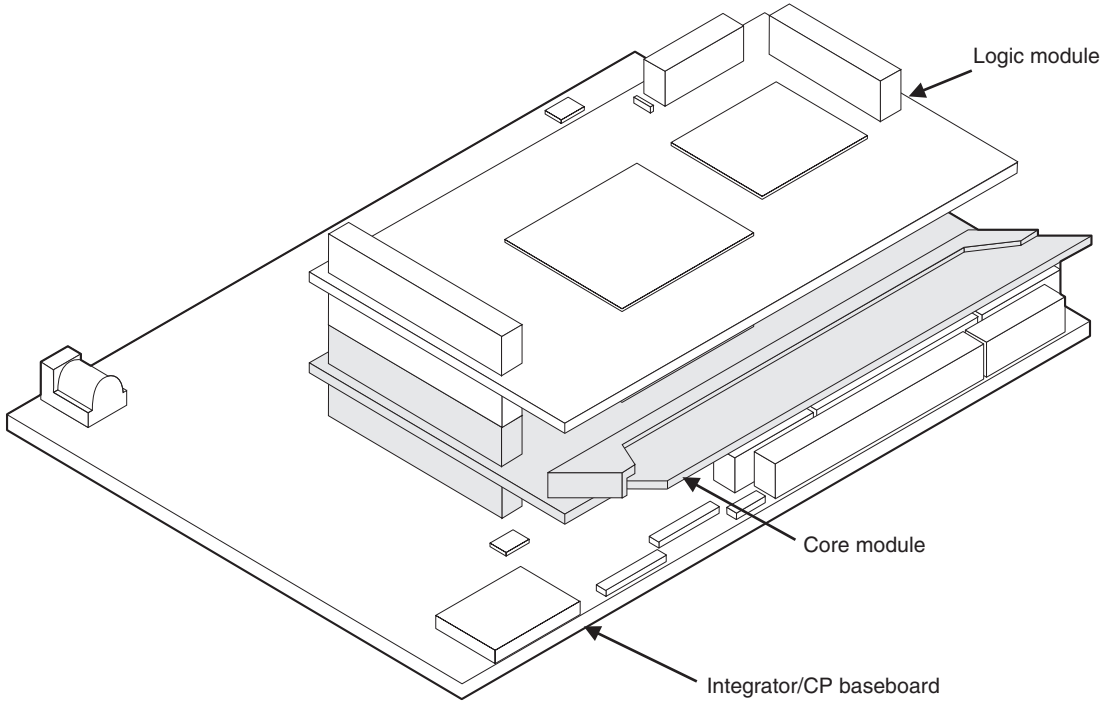


Figure 2-8 Assembled Integrator system

Note

To ensure reliable operation of the core module:

- Ensure that the core module loads a CP-compatible image to the FPGA.
- Do not connect power to the core module. Apply power to the baseboard.

2.3.1 FPGA image

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

Note

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

2.3.2 Core module ID

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

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

2.3.3 SDRAM, Trace, and Multi-ICE

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

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

2.3.4 Powering the assembled Integrator/CP development system

Power the assembled Integrator/CP development system by:

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

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

Chapter 3
Hardware Description

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

Note

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

3.1 ARM microprocessor test chip

This section provides a brief overview of the ARM926EJ-S, ARM946E-S, ARM926EJ-S, ARM1026EJ-S, and ARM1136JF-S processor macrocells. It outlines their main features and differences, and describes the operational configuration options that they provide.

3.1.1 Test chip overview

ARM926EJ-S, ARM946E-S, ARM966E-S, ARM1026EJ-S, and ARM1136JF-S processor macrocells are based on the ARM9EJ-S™, ARM9E-S™, ARM10EJ-S™, and ARM11JF-S™ RISC processor cores.

These macrocells are user-code compatible with code written for earlier versions of the ARM architecture (for example, ARM7TDMI™ and ARM9TDMI). They also support the instructions specific to the ARMv5TE architecture. The additional instructions improve DSP performance.

ARM9EJ-S cores

The ARM9EJ-S provides the Jazelle implementation of Java code acceleration.

The ARM926EJ-S combines the ARM9EJ-S core with:

- instruction and data caches
- tightly coupled instruction and data SRAM
- write buffer
- memory management unit
- ETM9 Embedded Trace Macrocell
- Jazelle extensions for Java acceleration.

ARM9E-S cores

The ARM9E-S is a 32-bit RISC processor based on the ARM9TDMI™ core. It includes signal processing extensions to the ARM instruction set and a single-cycle 16 x 32 *Multiply-ACcumulate* (MAC) unit.

The ARM946E-S combines the ARM9E-S core with:

- instruction and data caches
- tightly coupled instruction and data SRAM
- ETM9 Embedded Trace Macrocell
- write buffer
- memory protection unit.

The ARM966E-S combines the ARM9E-S core with:
- tightly coupled instruction and data SRAM
- ETM9 Embedded Trace Macrocell
- write buffer.

ARM10EJ-S core

The ARM10EJ-S provides the Jazelle implementation of Java code acceleration.

The ARM1026EJ-S combines the ARM10EJ-S core with:
- instruction and data caches
- tightly coupled instruction and data SRAM
- write buffer
- memory management and protection unit
- 64-bit internal buses
- vectored interrupt controller
- ETM10 Embedded Trace Macrocell
- VFP10 vector floating point coprocessor
- Jazelle extensions for Java acceleration.

ARM11JF-S core

The ARM11JF-S provides the Jazelle implementation of Java code acceleration.

The ARM1136JF-S combines the ARM11JF-S core with:
- instruction and data caches
- tightly coupled instruction and data SRAM
- DMA controller
- memory management unit
- 64-bit internal buses
- vectored interrupt controller
- ETM11 Embedded Trace Macrocell
- Vector floating point coprocessor and DSP extensions
- clock generator control block
- AHB matrix
- Jazelle extensions for Java acceleration.

3.1.2 Test chip configuration control

The ARM926EJ-S, ARM946E-S, ARM926EJ-S, ARM1026EJ-S, and ARM1136JF-S processors are configurable and have a number of input signals that control the configuration of the ARM test chip. In a final product, core configuration is static and these signals are tied HIGH or LOW as appropriate. However, because the Integrator is a development platform, it allows you to program the levels on these signals in the CM_INIT register for experimentation. The configuration signals driven by CM_INIT are described in Table 3-1.

Table 3-1 Controllable processor configuration signals

Signal/Bit name	Function	POR default
INITRAM	Use this signal to enable (HIGH) or disable (LOW) the internal (tightly-coupled) SRAM.	LOW
VINITHI	Use this signal to initialize the vector base address to `0xFFFF0000` (HIGH) or `0x00000000` (LOW). The Integrator system memory map allocates the high vector address location as logic module expansion space. This means that there may not be any physical memory at this location.	LOW
PLLBYPASS	Use this signal to control clock usage by processors that contain an internal PLL. When **PLLBYPASS** is HIGH, the processor uses the clock signal supplied on its **PLLCLKIN** signal pin. When **PLLBYPASS** is LOW, the processor uses the clock signal supplied by its internal PLL (derived from **PLLCLKIN**).	LOW
BIGENDINIT	Use this signal to control whether the big-endian or little-endian memory organization is used. This signal is typically modified by setting a bit in CP15. When **BIGENDINIT** is HIGH, the processor uses big-endian memory organization.	LOW

Note

The configuration inputs are controlled by the CM_INIT register within the FPGA. CM_INIT bit-states are retained during normal resets. During *Power-On Reset* (POR), before the FPGA is configured, the bit-states are defaulted by pull-down resistors to the values shown in the POR default column.

Changing the processor configuration

To change the configuration of the processor:

1. Program the appropriate values in the CM_INIT register (see *Core module initialization register* on page 4-23).

2. Reset the core module (but do not power-cycle) by pressing the reset button, or with a software reset, by writing a 1 to bit 3 of the CM_CTRL register (see *Core module control register* on page 4-17).

Restoring the default configuration

To restore the default processor configuration, power-cycle the core module.

Using CP15

Some processor configuration settings (cache enable/disable, MMU enable/disable, TCM enable/disable, and translation registers) are controlled by setting bits in CP15 registers.

For example, set bit 7 of CP15 register r1 to select big-endian operation. For more information on system control via the CP15 registers, see the Technical Reference Manual for the processor.

3.1.3 Fixed value test chip configuration controls

The ARM946E-S and ARM966E-S configuration inputs shown in Table 3-2 cannot be changed.

Table 3-2 Fixed processor configuration signals

Signal	Function
EFIBYPASS	This signal enables (HIGH) or disables (LOW) a bypass of synchronizers for **nFIQ** and **nIRQ**. This signal is tied LOW.
RESBYPASS	This signal enables (HIGH) or disables (LOW) a bypass of the synchronizer for the **nRESET** signal. This signal is tied LOW.
MICEBYPASS	This signal enables (HIGH) or disables (LOW) enables a bypass of the synchronizer for the JTAG signals. This signal is tied LOW.

3.2 Core module FPGA

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

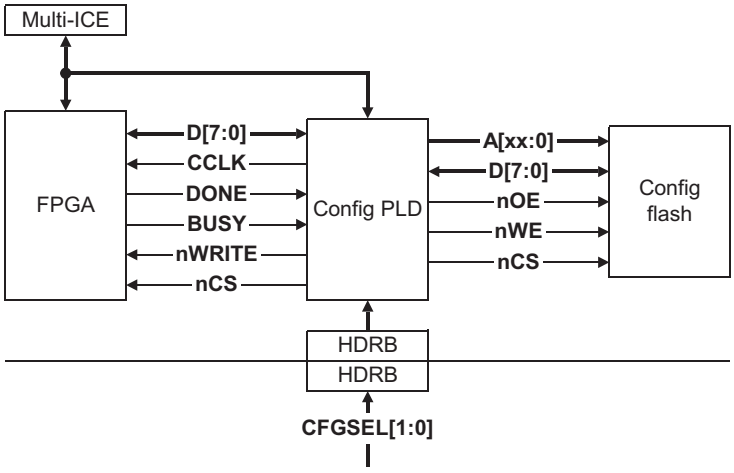


Figure 3-1 FPGA configuration

The following sections describe functional blocks implemented in the FPGA:

- *Core module control registers* on page 4-13
- Debug interrupt controller, see *Debug communications interrupts* on page 3-25.

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

You can use Multi-ICE to reprogram the PLD, FPGA, and flash when the core module is placed in configuration mode (see *Multi-ICE support* on page 3-17).

3.2.1 Integrator/AP FPGA image

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

For the Integrator/AP, the FPGA image also contains a system bus bridge. See *System bus bridge* on page 5-13.

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

3.2.2 Integrator/CP FPGA image

For the Integrator/CP, the FPGA image contains additional peripherals, the LCD controller for example. See *Programmable logic* on page 6-16 for more details.

A secondary interrupt controller is provided by the system controller FPGA on the motherboard. See the *Integrator/CP User Guide* for further information.

3.2.3 FPGA image selection

The configuration flash contains multiple images that enable the FPGA to support different types of motherboard. Image selection is controlled by the static configuration select signals, **CFGSEL[1:0]**, from the motherboard as shown in Table 3-3.

Table 3-3 Image selection signals

CFGSEL[1:0]	Image loaded
00	Little-endian ASB, Integrator/AP (not supported by 9x6 processors).
01	Reserved.
10	Little-endian AHB, Integrator/AP and standalone. If there is not a motherboard connected to the core module, pull-up and pull-down resistors on the core module select this as the default image selection code.
11	AHB Lite, Bi-endian, Integrator/CP.

3.3 Memory

This section describes the core module SSRAM and SDRAM.

3.3.1 SSRAM controller

All new CM946E-S and CM966E-S FPGA configurations implement the SSRAM controller in the FPGA image. Some early versions of the CM946E-S and CM966E-S have a PLD SSRAM controller fitted, but these PLDs are programmed with a blank design.

The CM926EJ-S, CM1026EJ-S, and CM1136JF-S versions do not have a PLD and the controller is always implemented in the FPGA.

3.3.2 SDRAM operating mode

The core module contains a DIMM socket for fitting SDRAM. The operating mode of the SDRAM devices is controlled with the mode set register within each SDRAM. These registers are set immediately after power-up to specify:

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

Note

Before the SDRAM can be used, you must program the CAS latency and memory size parameters in the SDRAM control register (CM_SDRAM) at address `0x10000020`. The required parameters are read from the SPD memory and are listed in Table 4-18 on page 4-29. If you do not correctly set the SDRAM parameters, accesses could be slow or unreliable. See *SDRAM status and control register* on page 4-21.

3.3.3 Serial presence detect

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

- module type
- memory organization
- timing parameters.

The EEPROM clock (SCL) operates at 93.75kHz (24MHz divided by 256). The transfer rate for read accesses to the EEPROM is 100kbit/s maximum. The data is read out serially 8 bits at a time, preceded by a start bit and followed by a stop bit. This makes reading the EEPROM a very slow process because it takes approximately 27ms to read

all 256 bytes. However, during power-up the contents of the EEPROM are copied into a 64 x 32-bit area of memory (CM_SPD) within the SDRAM controller. The SPD flag is set in the SDRAM control register (CM_SDRAM) when the SPD data is available. This copy can be randomly accessed at `0x10000100–0x100001FC` (see *SDRAM SPD memory* on page 4-34).

Write accesses to the SPD EEPROM are not supported.

3.4 Clock generators

The CM946E-S and CM966E-S core modules provides two programmable clock sources:

- the processor clock **ARM_PLLCLKIN**
- the auxiliary clock **AUXCLK**.

These two clocks are supplied by two MicroClock ICS525 clock generators, as shown in Figure 3-2. In addition, the core module produces the fixed-frequency clock **REFCLK** and a buffered version of the processor clock **FPGA_PLLCKIN**.

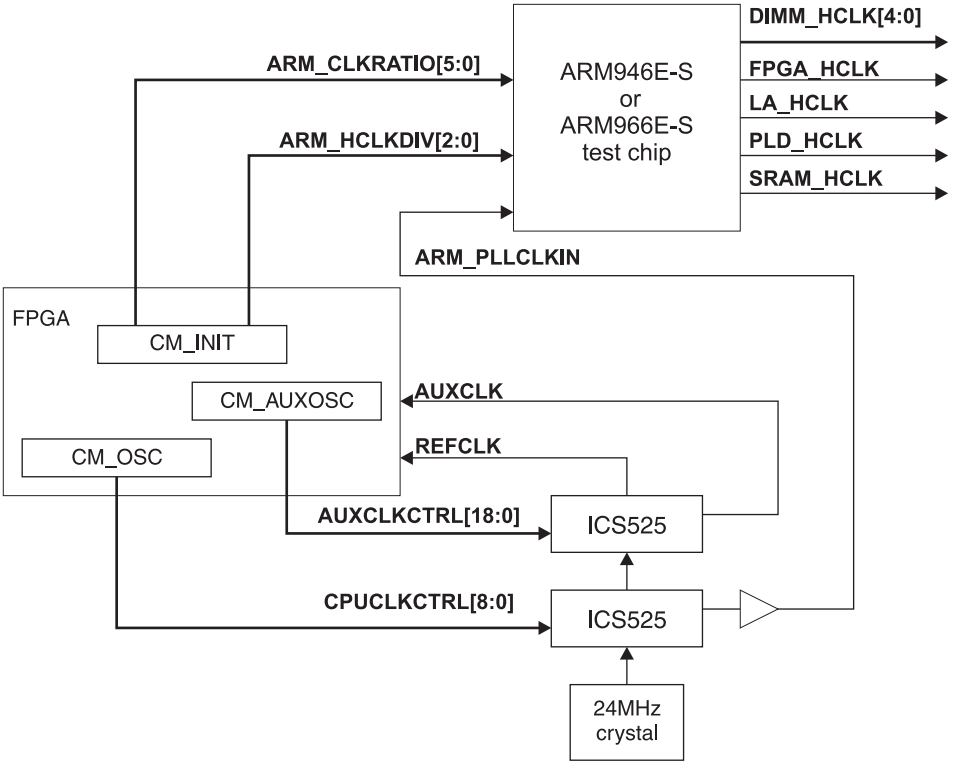


Figure 3-2 CM946E-S and CM966E-S clock generator

The CM926EJ-S, CM1026EJ-S, and CM1136JF-S core modules provide three programmable clock sources:

REFCLK Processor clock. This is the ICS307 **CORECLK** output converted to the correct voltage level for the test chip.

BUSCLK Auxiliary clock. This is reserved for theCM926EJ-S and CM1026EJ-S. It is used for asynchronous mode for the CM1136JF-S, see Appendix C *Features specific to the CM1136JF-S*.

AUXCLK Auxiliary clock. This is reserved for future use.

These clocks are supplied by three MicroClock ICS307 clock generators, as shown in Figure 3-3. In addition, the core module produces the fixed-frequency clock **REF24MHZ**.

The clock generators are loaded with the programmed value over a serial bus. Interface logic in the FPGA manages the conversion from parallel register data to the serial generator control signals.

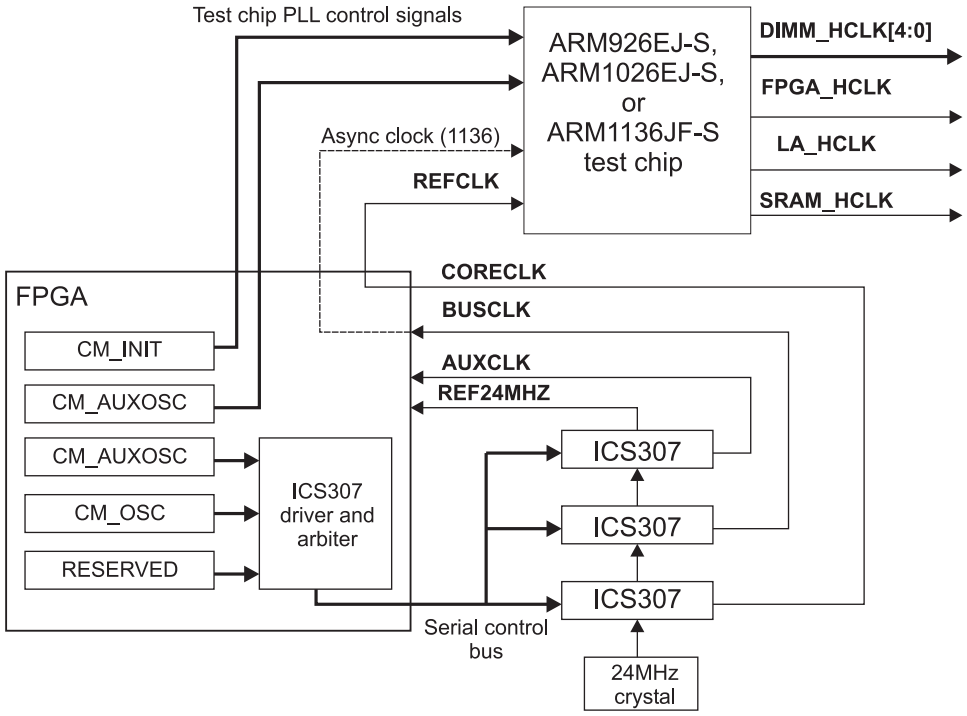


Figure 3-3 CM926EJ-S clock generator

3.4.1 Clock generation functional overview

Note

The CM946E-S and CM966E-S core modules use ICS525 clock generators, but the CM926EJ-S, CM1026EJ-S, and CM1136JF-S use ICS307 clock generators. The ICS307 is a serially controlled version of the ICS525. The term ICS generator is used to refer to both the ICS525 and the ICS307 clock generator.

The ICS generators are supplied with a reference clock by a 24MHz crystal oscillator. The frequency outputs are controlled using their *divider* input value. This enables them to produce a wide range of frequencies.

Note

The maximum output for the ICS525 is 160MHz and the maximum output for the ICS307 is 200MHz.

CM946E-S and CM966E-S

The output **ARM_PLLCLKIN** from the first ICS525 provides the clock source for a PLL incorporated into the ARM test chip. The PLL generates the core clock **CLK** that is internally divided to produce the AMBA AHB clock **HCLK**. The **HCLK** signal is distributed to devices around the core module using a number of dedicated output pins that are provided by the ARM test chip.

The signal **ARM_FPGACKIN** signal is supplied to the FPGA for future use. It always operates at the same frequency as **ARM_PLLCKIN**.

The second ICS generator outputs the clock signal **AUXCLK** that is supplied to the FPGA for future use.

The fixed frequency 24MHz clock signal **REFCLK** is supplied to the FPGA for use by the internal counter CM_REFCNT. This can be used, for example, to provide real-time delays.

CM926EJ-S, CM1026EJ-S, and CM1136JF-S

The output **CORECLK** from the first ICS307 provides the clock source for a PLL incorporated into the ARM test chip. The PLL generates the core clock **CLK** that is internally divided to produce the AMBA AHB clock **HCLK**. The **HCLK** signal is distributed to devices around the core module using a number of dedicated output pins that are provided by the ARM test chip.

The second and third ICS307 generators output the clock signals **AUXCLK** and **BUSCLK**. **AUXCLK** is used as a pixel clock on the Integrator/CP and **BUSCLK** is reserved for future use. For the Integrator/AP, these are supplied to the FPGA for future use.

Note

For the CM1136JF-S, **BUSCLK** is used in the standalone/AP configuration for asynchronous mode.

The fixed frequency 24MHz clock signal **REF24MHZ** is supplied to the FPGA for use by the internal counter CM_REFCNT. This can be used, for example, to provide real-time delays.

3.4.2 Programming the clocks

The frequency of clocks from an ICS generator is controlled by the reference divider value, *Voltage Controlled Oscillator* (VCO) divider value, and output divider value. For the ICS525, these values are set by placing HIGH and LOW logic levels on its input pins. For the ICS307, the values are set by serially loading the generator control registers.

The parameters determine the output of the ICS generator using the formula:

$$\text{Freq} = 48\,\text{MHz} \times \frac{(\text{VDW} + 8)}{(\text{RDW} + 2) \times \text{OD}}$$

where:

VDW Is the VCO divider word.

RDW Is the reference divider word.

OD Is the output divider.

This formula is used to calculate the frequency of **AUXCLK** but can be simplified for **ARM_PLLCKIN** (**CORECLK** for the CM926EJ-S, CM1026EJ-S, and CM1136JF-S), as described in *Setting the frequency of test chip core clock* on page 3-14.

Note

You can also calculate values for RDW, VDW, and OD using the ICS calculator on the Microclock website.

Setting the frequency of test chip core clock

In the case of the clock generator for **ARM_PLLCLKIN** (**CORECLK** for the CM926EJ-S, CM1026EJ-S, and CM1136JF-S), the values for RDW and OD are fixed. Pull-up and pull-down resistors are used for the ICS525 and a fixed serial value is loaded for the ICS307. VDW is programmable in the CM_OSC register (see *Core module oscillator register* on page 4-16). You can program the clock to frequencies in the range 12 to 160MHz in 1MHz steps (18 to 200MHz for the CM926EJ-S, CM1026EJ-S, and CM1136JF-S).

The value for RDW is fixed at 22 and the value for OD is fixed at 2.

This means that you can calculate the frequency in MHz of **ARM_PLLCKIN** (or **CORECLK** for the CM926EJ-S, CM1026EJ-S, and CM1136JF-S) from the following formula:

Freq = VDW+ 8

where VDW can be assigned any value between a minimum of 4 and a maximum of 152 (maximum VDW for the CM926EJ-S, CM1026EJ-S, and CM1136JF-S is 192).

Setting the frequency of AUXCLK

In the case of the clock generator for **AUXCLK**, the values for RDW, VDW, and OD are all programmable in the CM_AUXOSC register (see *Core module auxiliary oscillator register* on page 4-19). This gives frequencies in the range 1 to 160MHz with a better than 0.1% accuracy (6 to 200MHz for the CM926EJ-S). The default frequency following a power-on reset is 32.369MHz.

3.4.3 Test chip clocks

The test chip on the core module produces two clocks:

- the core clock **CLK**
- the local memory bus clock **HCLK**.

The frequency of **CLK** is controlled by a PLL within the test chip.

The internal PLL can be bypassed by setting PLLBYPASS bit in the CM_INIT register. This sets **CLK** to the same frequency as the reference clock and allows you to model implementations where the core does not have a PLL.

The test chip produces the **HCLK** signal by dividing **CLK** by a value of between 2 and 8. The divider can be effectively bypassed by setting a divide by value of 1.

CM946E-S and CM966E-S core clock

The PLL within the test chip uses the **ARM_CLKRATIO[5:0]** signals to determine the frequency of **CLK**. (In some cases, the **USERIN[7:0]** signals are used as multipliers or dividers to modify the clock. Usage of these signals is partner specific.) These signals are programmed in the register CM_INIT (see *Core module initialization register* on page 4-23).

CM926EJ-S and CM1026EJ-S core clock

The PLL within the test chip uses the **PLLFBDIV[7:0]** signals from the CM_INIT register to determine the frequency of **CLK**. (One or more of the **PLLCTRL[1:0]**, **PLLREFDIV[3:0]**, or **PLLOUTDIV[3:0]** signals from the CM_AUXOSC register are used as multipliers or dividers to modify the clock. Usage of these signals is partner specific.) See *Core module initialization register* on page 4-23.

CM946E-S and CM946E-S local memory bus clock

The **HCLK** signal is generated by dividing **CLK** by a value of between 1 and 8 as determined by the value of signals **ARM_HCLKDIV[3:0]** programmed in the CM_INIT register (see *Core module initialization register* on page 4-23). The **HCLK** signal is distributed by the test chip on four identical signals:

- **ARM_HCLK[0]** drives the FPGA
- **ARM_HCLK[1]** drives the SSRAM PLD
- **ARM_HCLK[2]** drives the SSRAM
- **ARM_HCLK[3]** drives a low-skew clock buffer that produces six outputs. These are supplied to the logic analyzer module, a test point, and to the SDRAM DIMM.

CM926EJ-S and CM1026EJ-S local memory bus clock

The **HCLK** signal is generated by dividing **CLK** by a value of between 1 and 8 as determined by the value of signals **ARM_HCLKDIV[3:0]** programmed in the CM_INIT register (see *Core module initialization register* on page 4-23). The **HCLK** signal is distributed by the test chip on five identical signals:

- **HCLKEXT0** clocks the FPGA
- **HCLKEXT1** can be used if the core module is connected to an Integrator/CP
- **HCLKEXT2** clocks the SSRAM
- **HCLKEXT3** drives a low-skew clock buffer that produces six outputs. These are supplied to the logic analyzer module, a test point, and to the SDRAM DIMM
- **HCLKEXT4** connected to test point 16 (HCLK).

CM1136JF-S clocks

The ARM1136JF-S can be operated with synchronous or asynchronous core and internal bus clocks.

A programmable clock and reset generator controls clock selection and frequency.

Note

See Appendix C *Features specific to the CM1136JF-S* for details on programming the CM1136JF-S clocks.

3.5 Multi-ICE support

The core module provides support for debug using JTAG. It provides a Multi-ICE connector and JTAG scan paths around the system. Figure 3-4 shows the Multi-ICE connector, the CONFIG link, and LED for a CM966E-S module.

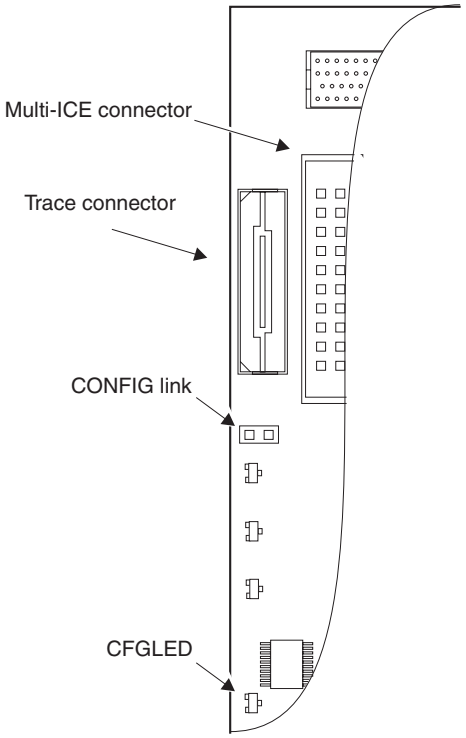


Figure 3-4 JTAG connector, CONFIG link, and LED

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

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

3.5.1 JTAG scan paths

This section describes the routing of the JTAG scan chain on the core module. It discusses the data path and the clock path.

Data path

Figure 3-5 shows a simplified diagram of the data path for an Integrator/AP and two core modules.

Note

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

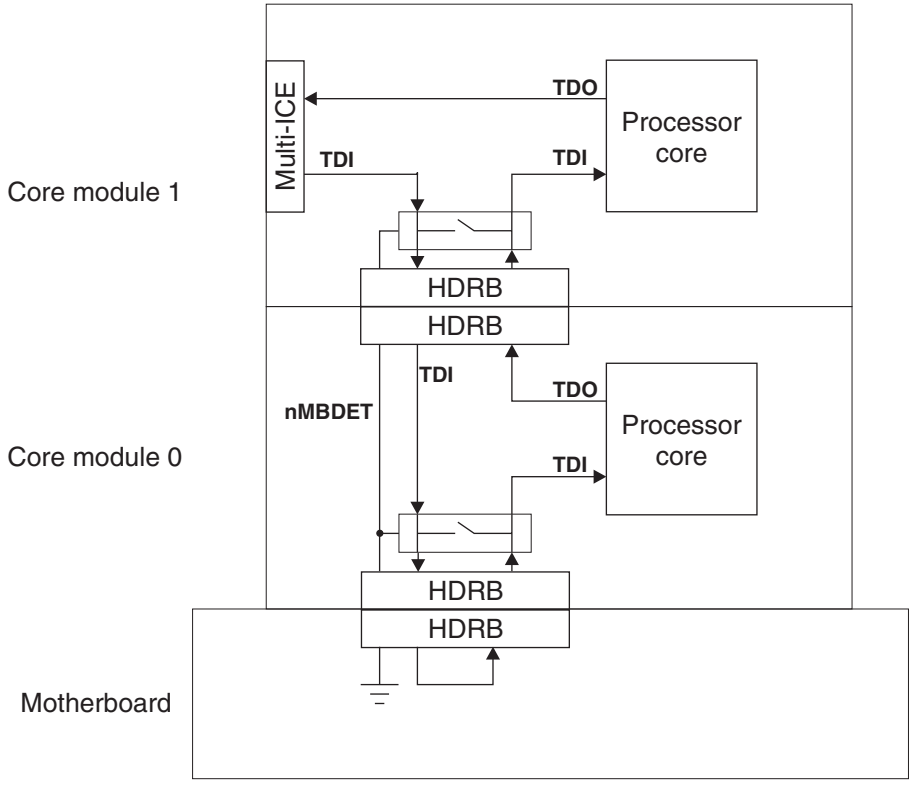


Figure 3-5 JTAG data path

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 3-20.

Clock path

The clock path is routed in a similar way to the data path, although in the opposite direction. Figure 3-6 on page 3-20 shows a simplified diagram of the clock path.

Note

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

The processors on the core module use a synthesized core that samples **TCK**. This introduces a delay into the clock path. For this reason, it passes on the clock signal as **RTCK**, that is fed to the **TCK** input of the next device in the chain. The **RTCK** signal at the Multi-ICE connector is used by Multi-ICE (or RealView 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, although in this case the loopback is controlled by the signal **nRTCKEN** and an AND gate on the motherboard (the pullups on **nMBDET** are omitted for clarity).

Not all cores sample **TCK** (for example, the ARM7TDMI), in this case the **TCK** signal is routed straight through to the next board down the stack. If one or more modules in a stack drives **RTCK** (and so asserts **nRTCKEN**), you must ensure that the board at the bottom of the stack provides the necessary return path. All Integrator motherboards do so.

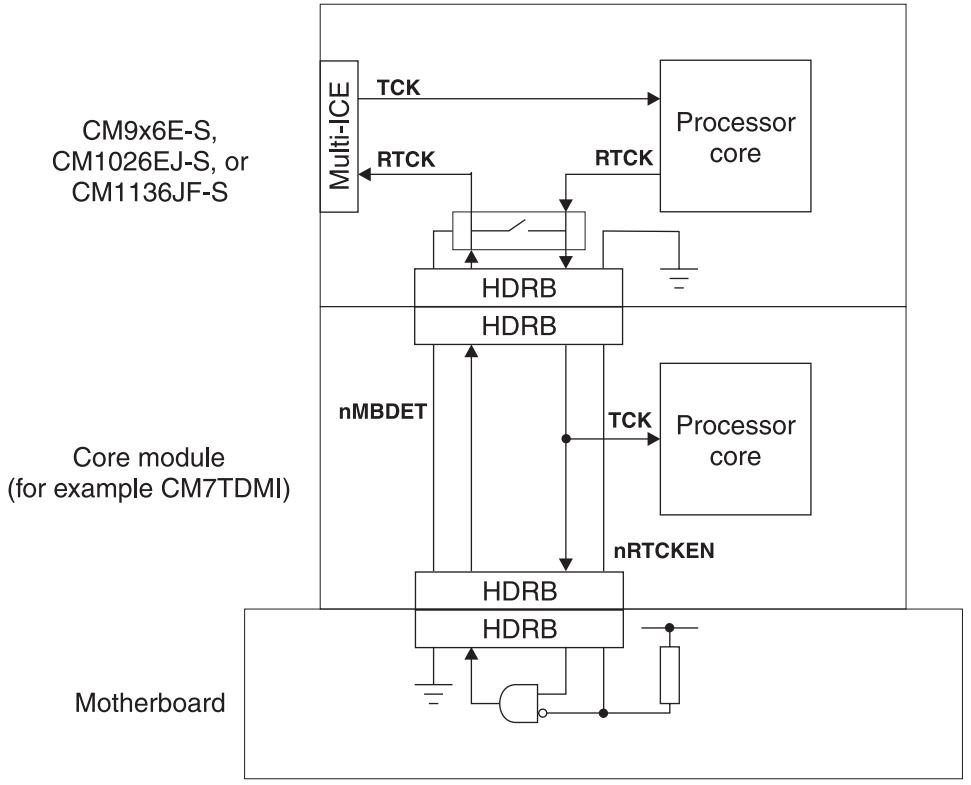


Figure 3-6 JTAG clock path

3.5.2 JTAG connection modes

The core module operates in one of two JTAG modes:

- *Debug mode*
- *Configuration mode* on page 3-21.

Debug mode

Debug mode is selected by default (when a jumper is *not* fitted at the CONFIG link, see Figure 3-4 on page 3-17). It is the mode used for general system development and debug, including using trace (see *Embedded Trace support* on page 3-26). In this mode, only the processor core and debuggable devices on other modules are accessible on the scan chain, as shown in Figure 3-5 on page 3-18.

Configuration mode

In configuration mode, all FPGAs and PLDs in the system are added into the scan chain. This mode allows the programmable devices in the system to be reprogrammed in the field using Multi-ICE.

To select configuration mode, fit a jumper to the CONFIG link on the core module *at the top of the stack (*see Figure 3-4 on page 3-17*)*. This has the effect of pulling the **nCFGEN** signal LOW and illuminating the CFG LED (yellow) on each module in the stack and rerouting the JTAG scan path on all modules in the stack. The LED provides a warning that the development system is in the configuration mode.

Note

Configuration mode is guaranteed for a single core module attached to a motherboard but may be unreliable if more than one core module is attached.

After configuration or code updates you must:

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

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

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

3.5.3 JTAG signals

Figure 3-7 on page 3-22 shows the pinout of the Multi-ICE connector and Table 3-4 on page 3-22 provides a description of the JTAG signals.

Note

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

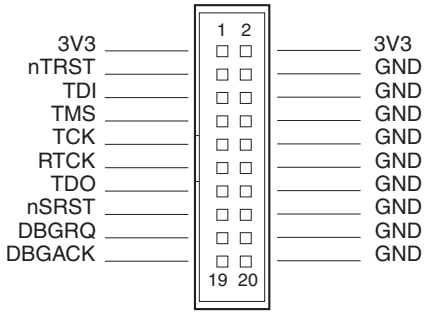


Figure 3-7 Multi-ICE connector pinout

Table 3-4 JTAG signal description

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

Table 3-4 JTAG signal description (continued)

Name	Description	Function
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. The open collector **nSRST** reset signal can be driven LOW by the reset controller on the core module to cause the motherboard to reset the whole system by driving **nSYSRST** LOW. This is also used in configuration mode to control the initialization pin (**nINIT**) on the FPGAs. Though not a JTAG signal, **nSRST** is described because it can be controlled by JTAG equipment.
nTRST	Test reset (from JTAG equipment)	This active LOW open-collector signal is used to reset the JTAG port and the associated debug circuitry on the processor. It is asserted at power-up by each module, and can be driven by the JTAG equipment. This signal is also used in configuration mode to control the programming pin (**nPROG**) on FPGAs.
RTCK	Return TCK (to JTAG equipment)	Some devices sample **TCK** (for example a synthesizable core with only one clock), and this has the effect of delaying the time at which a component actually captures data. **RTCK** is a mechanism for returning the sampled clock to the JTAG equipment, so that the clock is not advanced until the synchronizing device captured the data. In *adaptive clocking mode*, Multi-ICE is required to detect an edge on **RTCK** before changing **TCK**. In a multiple device JTAG chain, the **RTCK** output from a component connects to the **TCK** input of the down-stream device. The **RTCK** signal on the module connectors HDRB returns **TCK** to the JTAG equipment. If there are no synchronizing components in the scan chain then it is unnecessary to use the **RTCK** signal and it is connected to ground on the motherboard.
TCK	Test clock (from JTAG equipment)	**TCK** synchronizes all JTAG transactions. **TCK** connects to all JTAG components in the scan chain. Series termination resistors are used to reduce reflections and maintain good signal integrity. **TCK** flows down the stack of modules and connects to each JTAG component. However, if there is a device in the scan chain that synchronizes **TCK** to some other clock, then all down-stream devices are connected to the **RTCK** signal on that component (see **RTCK**).

Table 3-4 JTAG signal description (continued)

Name	Description	Function
TDI	Test data in (from JTAG equipment)	**TDI** goes down the stack of modules to the motherboard and then back up the stack, labeled **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 labeled **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.

3.5.4 Debug communications interrupts

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

The processor accesses the debug communications channel registers using `MCR` and `MRC` instructions. The JTAG equipment reads and writes the register using the scan chain. For a description of the debug communications channel, see the *ARM926EJ-S Technical Reference Manual*, the *ARM966E-S Technical Reference Manual,* the *ARM946E-S Technical Reference Manual*, *ARM1026EJ-S Technical Reference Manual,* or the *ARM1136JF-S Technical Reference Manual*.

Interrupts can be used to signal when data has been written into one side of the register and is available for reading from the other side. These interrupts are supported by the interrupt controller within the core module FPGA, and can be enabled and cleared by accessing the interrupt registers (see *Core module interrupt registers* on page 4-28).

3.6 Embedded Trace support

The ARM926EJ-S, ARM946E-S, and ARM966E-S processors incorporate an *ARM9 Embedded Trace Macrocell* (ETM9).

The ARM1026EJ-S processor incorporates an *ARM10 Embedded Trace Macrocell* (ETM10).

The ARM1136JF-S processor incorporates an *ARM11 Embedded Trace Macrocell* (ETM11) and an *Embedded Trace Buffer* (ETB).

The trace macrocell 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.

3.6.1 About using trace

Figure 3-8 on page 3-27 illustrates a trace debugging setup with the core module. (A CM9x6E-S is shown, but the setup is the same for the CM1026EJ-S and CM1136JF-S)

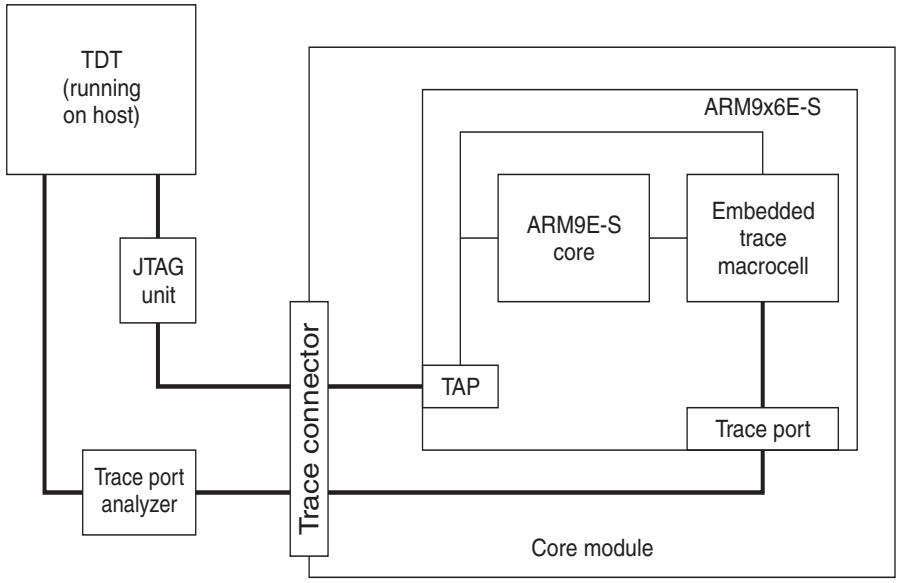


Figure 3-8 Trace connection

Note

The routing of the JTAG scan chain on the Integrator system is described in *Multi-ICE support* on page 3-17.

The components in the trace debug setup shown in Figure 3-8 are as follows:

Embedded trace macrocell

The ETM monitors the ARM core buses and outputs compressed information through the trace port to a trace port analyzer. 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 may be a separate device or may be incorporated within the TPA.

Trace debug tools

The *Trace Debug Tools* (TDT) is an optional component of the ARM Developer Suite 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*.

3.6.2 Core module trace configuration

The ARM926EJ-S, ARM946E-S, ARM966E-S, ARM1026EJ-S, and ARM1136JF-S test chips typically provide a *large* ETM configuration. The resources provided by this configuration are summarized in Table 3-5.

Table 3-5 ETM resources

Resource type	Number
Address comparator pairs	8
Data comparator	8
Memory map decoders	16
Counters	4
Sequencer	Yes
External inputs	4
External outputs	4
FIFOFULL	Yes
FIFO depth	45
Trace packet width	4/8/16

The TRACE connector enables you to connect an external embedded trace interface module. (The CM946E-S and CM966E-S have one connector, but the CM926EJ-S, CM1026EJ-S, and CM1136JF-S have two connectors.)

Note

The size of ETM depends on the test chip fitted to the board. ETM functionality is not guaranteed for all test chip versions.

3.6.3 Trace interface description

The logic analyzer connection is a high-density AMP Mictor connector. The pinout for this connector is provided in Trace interface pinout on page A-9.

3.7 Stacking options

The core module provides two stacking options selected by the position of a surface mount link, LK1, that is set at manufacture.

Table 3-6 shows the link LK1 positions and the corresponding stacking option.

Table 3-6 Link LK1 positions

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

3.8 Power supply control (CM926EJ-S, CM1026EJ-S, and CM1136JF-S only)

Caution

Changing the resistors that control the core power-supply voltage might result in damage to the test chip (see *Setting the VDDCORE voltage* on page 3-33). The fitted resistors enable you to modify the core voltage within a safe region.

Modifying the core voltage is not required for normal operation.

The CM926EJ-S, CM1026EJ-S, and CM1136JF-S all have four 32-bit registers (located in the status and control register bank) that enable you to:

- Change the voltage supplied to the core. An 8-bit value is written to a register and transferred to a serially-programmed *digital to analog converter* (DAC).
- Read onboard voltages from an 8-channel 12-bit *analog to digital converter* (ADC):
  - ARM_VDDIO, ARM_VDDLEVEL, and ARM_VDDPLL. To prevent the ADC input range being exceeded, these voltages are divided by two before being fed to the ADC.
  - VDDCORE. This is the voltage supplied to the core.
  - VDDCORE_DIFF[4:1]. This is the voltage generated through current-sensing resistors in the supply to the power rails ARM_VDDCOREx. The value of the voltage is proportional to the current consumed by the core.

  Note

  The FPGA ADC controller regularly reads the values from the ADC and stores them in the registers. The refresh rate is HCLK/4000 (10kHz for HCLK equal to 40MHz.)

For details on reading and setting core voltages, see *Voltage control registers* on page 4-32.

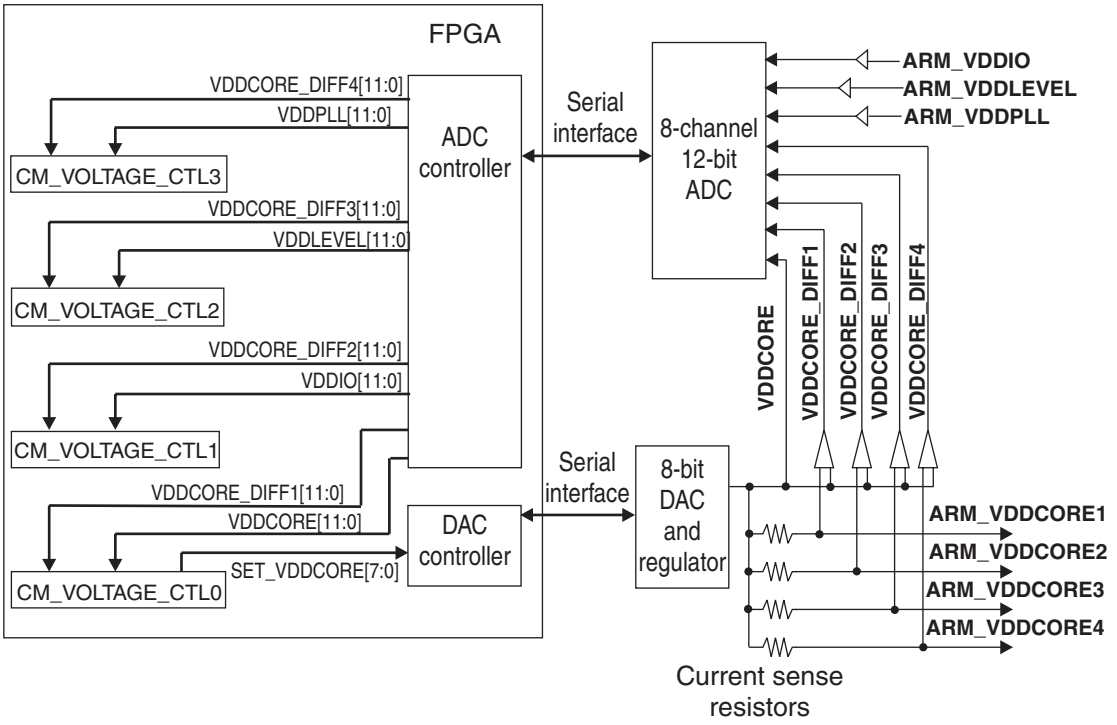


Figure 3-9 Voltage control and monitoring

3.8.1 Reading the voltages

The LSB of the register value corresponds to 0.6105 mV. (The 2.5V V_{REF} of the ADC divided by 4095.) To calculate the **VDDCORE** voltage, for example:

$V_{in} = V_{ref} * CM\_VOLTAGE\_CTL0[19:8] / (2^{12}-1)$

Note

The voltages **ARM_VDDIO**, **ARM_VDDLEVEL**, and **ARM_VDDPLL** are divided by two before being fed to the ADC. Therefore, the actual **ARM_VDDx** voltage is twice the measured value.

A voltage proportional to the **ARM_VDDCOREx** current is developed across the sense resistors. Each of the sense resistors has a voltage amplifier as the voltage is too low to measure directly with the ADC. To calculate the current to a **ARM_VDDCOREx** line:

$I_{ARM\_VDDCOREx} = V_{ref} * CM\_VOLTAGE\_CTLx[31:20] / (R_{SENSE} * GAIN * (2^{12}-1))$

Note

By default, R_{SENSE} is 0Ω. It is not therefore possible to measure the current flow.

You can replace the existing sense resistors with new ones of, for example, 1Ω.

For the CM926EJ-S, CM1026EJ-S, and CM1136JF-S, the default value for `GAIN` is 22.

For the CM946E-S and CM966E-S, the default value for `GAIN` is 10.

If you change the sense resistors, you must adjust the core voltage to compensate for the voltage drop (a 1Ω sense resistor drops 0.2V at 200mA).

3.8.2 Setting the VDDCORE voltage

The core voltage depends on:

- Resistors R136 and R143. These define the core voltage at power-on and the range of adjustment possible. These resistors are fitted at manufacture to give the correct core voltage for the test chip fitted to the core module.
- The value in CM_VOLTAGE_CTRL0[7:0]. This value provides a positive or negative offset to the default power-on voltage. The register content at power-on reset is `0x80 (this corresponds to` I_{DAC} of 25µA). (This is also the POR default value for the DAC converter.) Writing a higher value to the register decreases the core voltage and writing a lower value increases the voltage.

The output voltage is given by:

$V_{CORE} = V_{DAC}*(1+R136/R143) - CM\_VOLTAGE\_CTL0[7:0]*R136*(I_{DAC}/255)$

where:

I_{DAC} The full-range DAC output current (50µA)

V_{DAC} The DAC output voltage (0.605V).

The register contents for a particular output voltage is given by:

$CM\_VOLTAGE\_CTL0[7:0] = (255/(R136*I_{DAC})) * ((V_{DAC}*(R136 +R143)/R143)-V_{CORE})$

For example, if R136 is 10kΩ and R143 is 5.1kΩ, then the default output voltage is approximately 1.5V and the adjustment range is plus or minus 0.25V.

To change the core voltage:

1. Write `0xA05F` to the lock register to unlock the control registers.
2. Write the new value for the output voltage to CM_VOLTAGE_CTL0[7:0].

3. The core voltage changes on the next DAC refresh cycle (wait approximately 200µs).
4. Read the value in CM_VOLTAGE_CTL0[19:8] and verify that the voltage is correct (see *Reading the voltages* on page 3-32).
5. Write any value other than `0xa05f` to the lock register to relock the control registers.

Chapter 4
Programmer's Reference

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

Note

The core module functionality is different when used with either the Integrator/AP or Integrator/CP. See Chapter 5 *Using Core Modules with an Integrator/AP* and Chapter 6 *Using Core Modules with an Integrator/CP* for more details.

4.1 About the memory map

To understand the memory map options provided by the CM926EJ-S, CM946E-S, CM966E-S, CM1026EJ-S, and CM1136JF-S, it is helpful to be familiar with the main features of the ARM926EJ-S, ARM946E-S, ARM966E-S, ARM1026EJ-S, and ARM1136JF-S that influence their memory maps. This section provides a brief overview of these features. For detailed information, see the technical reference manual for the processor.

All of the processors incorporate a write buffer and typically include internal *Tightly-Coupled RAM* (TCM). However, the processors differ in the following ways:

- Memory management
  - The ARM926EJ-S and ARM1136JF-S incorporate a memory management unit that allows you to protect or remap memory regions.
  - The ARM946E-S incorporates a protection unit that allows you to define protection attributes for up to eight regions. Protection attributes include cacheable, bufferable, user access, and supervisor access.
  - The ARM1026EJ-S incorporates both memory management and memory protection units that allows you to protect or remap memory regions and define region attributes.
  - By contrast, the ARM966E-S has 16 fixed attribute regions within its memory map that are predefined alternately as buffered and unbuffered.
- Tightly coupled memory
  - The ARM946E-S has separate areas of internal TCM for data and instructions. These can be individually enabled or disabled, and can be varied in size. The Instruction TCM (I-RAM), if enabled, is always located at address `0x00000000`, but the Data TCM (D-RAM), if enabled, can be assigned to any size-aligned boundary.
  - The ARM966E-S also has areas for internal I-RAM and D-RAM that can be separately enabled or disabled but differs in that these are of fixed size. The I-RAM, if enabled, is always located at base address `0x00000000` and the D-RAM, if enabled, is always located at base address `0x04000000`.
  - The ARM926EJ-S, ARM1026EJ-S, and ARM1136JF-S also have areas for internal I-RAM and D-RAM that can be separately enabled or disabled but differs in that the areas can be independently located in the 4GB address space. The boundary must equal the size of the TCM, for example, a 32KB TCM must be aligned on 32KB boundary. The I-RAM and D-RAM memory range must not overlap.

Note

You must be careful when choosing a location for the D-RAM and I-RAM on the ARM926EJ-S, ARM946E-S, ARM1026EJ-S, and ARM1136JF-S that you do not mask registers or other important locations.

Figure 4-1 shows example memory maps for the core modules with I-RAM and D-RAM enabled. Accesses within these areas stay on-chip. This means, for example, that any Integrator memory at address `0x00000000`–`0x04000000` in the ARM966E-S memory map is masked if the I-RAM is enabled.

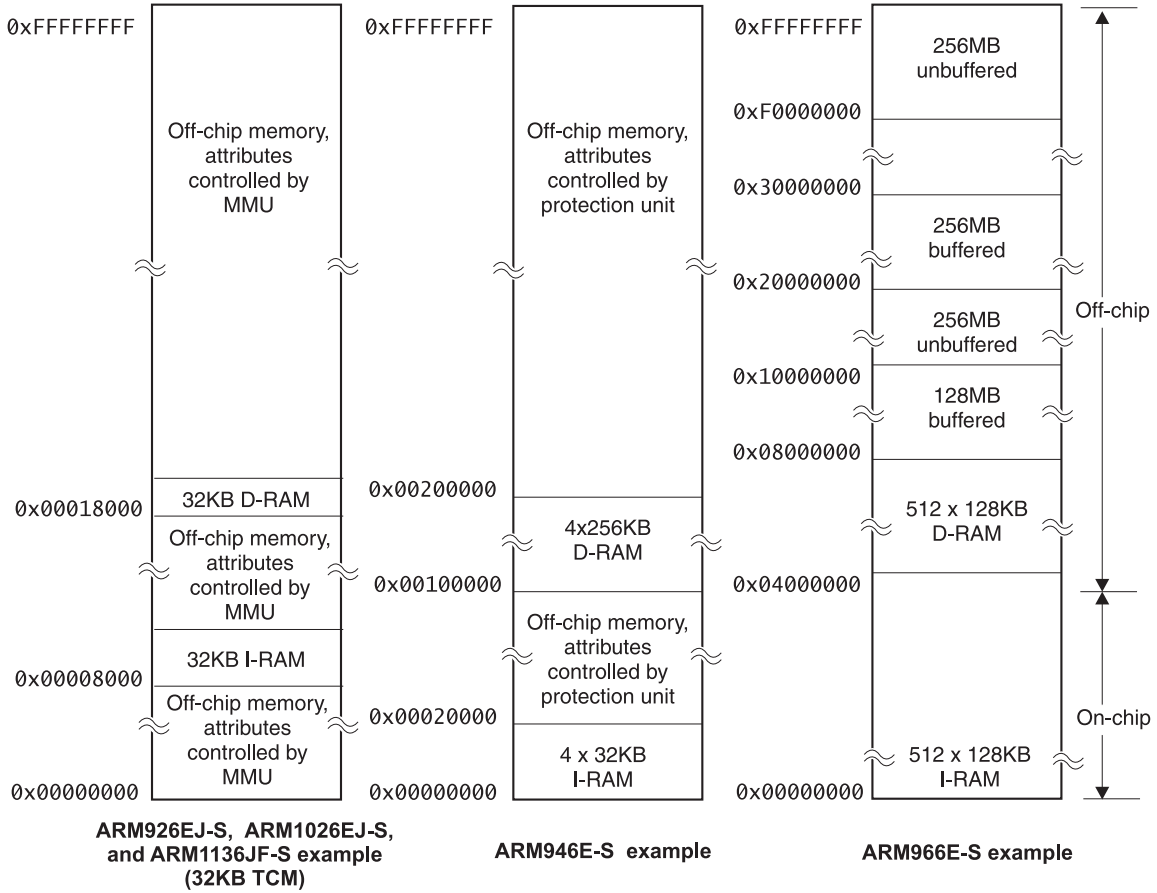


Figure 4-1 Example memory maps

4.1.1 TCM state after reset

On all ARM926EJ-S, ARM946E-S, ARM966E-S, ARM1026EJ-S and ARM1136JF-S processors, the TCMs are enabled using the system coprocessor register (CP15r1). In some of these processors this bit is initialized at reset from the INITRAM bit (see *Core module initialization register* on page 4-23):

ARM966E-S Both I-RAM and D-RAM are enabled or disabled immediately after reset according to the state of INITRAM.

ARM946E-S rev1

The I-RAM is enabled or disabled immediately after reset according to the state of INITRAM.

ARM946E-S rev0

Both I-RAM and D-RAM are disabled after reset.

ARM926EJ-S, ARM1026EJ-S, and ARM1136JF-S

D-RAM is always disabled at reset and I-RAM is enabled immediately after reset if INITRAM is equal to 1.

4.1.2 TCM size and aliasing (CM946E-S and CM966E-S)

Typically, the amount of I-RAM and D-RAM physically present within the test chip is less than the address space allocated to it. However, each is aliased throughout its assigned range.

Note

A feature of the core modules is that it allows you to use the on-board SSRAM to mimic the TCRAM (see *SSRAM mode selection (CM946E-S and CM966E-S only)* on page 4-8). This enables you to model different TCRAM implementations, even if your test chip does not have any TCRAM, and provides logic analyzer visibility of accesses to the RAM. However, if you use the SSRAM to mimic the TCRAM, performance is slower.

4.2 Core module memory map configuration

The core module provides a configurable memory map that enables you to experiment with different system configurations. The mapping of the control and status registers, and region for access to motherboard resources are fixed. However, the mapping for the SDRAM, SSRAM and TCRAM (built into the test chip) change when the configuration of the core module is changed. These configuration options are described in the following sections:

- *SSRAM mode selection (CM946E-S and CM966E-S only)* on page 4-8
- *Using REMAP* on page 4-9.

Table 4-1 shows the configurable area of the memory map and the fixed areas.

Table 4-1 Overview of core module memory map

Address range	Size	Description
0x00000000–0x07FFFFFF	128MB	Configurable mapping of: • TCRAM • On-board SSRAM • boot ROM (on motherboard) • flash (on motherboard) • SDRAM
0x08000000–0x0FFFFFFF	128MB	SDRAM (alias)
0x10000000–0x107FFFFF	8MB	Core module registers
0x10800000–0x10FFFFFF	8MB	On-board SSRAM (alias)
0x11000000–0xFFFFFFFF	3824MB	System bus (Abort if core module is not fitted to a motherboard)

Configuration of the memory map in the address space 0x00000000–0x07FFFFFF is controlled by the SSRAM mode (CM946E-S and CM966E-S only) and by REMAP. Figure 4-2 on page 4-6 shows the affect on the CM926EJ-S memory map of configuring REMAP. Figure 4-3 on page 4-7 shows the affect on the CM946E-S and CM966E-S memory map of configuring the SSRAM mode and REMAP.

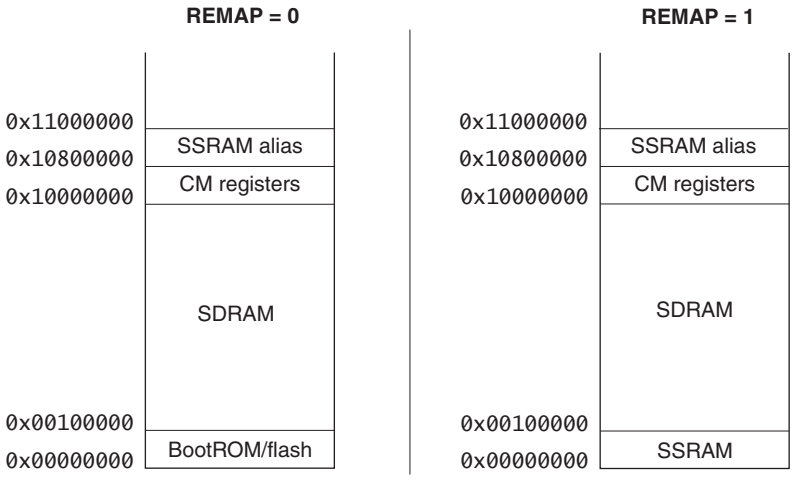


Figure 4-2 Configurable memory map for CM926EJ-S, CM1026EJ-S, and CM1136JF-S

Note

Figure 4-3 on page 4-7 shows the memory map for the four SSRAM-mode/REMAP configurations with the TCRAM *disabled*. The TCRAM, when enabled, masks accesses to the areas of the memory map that it occupies (see *About the memory map* on page 4-2.)

Setting the REMAP bit to 0 only takes effect if there is a motherboard connected. If there is no motherboard connected, the behavior is the same as that shown for REMAP=1.

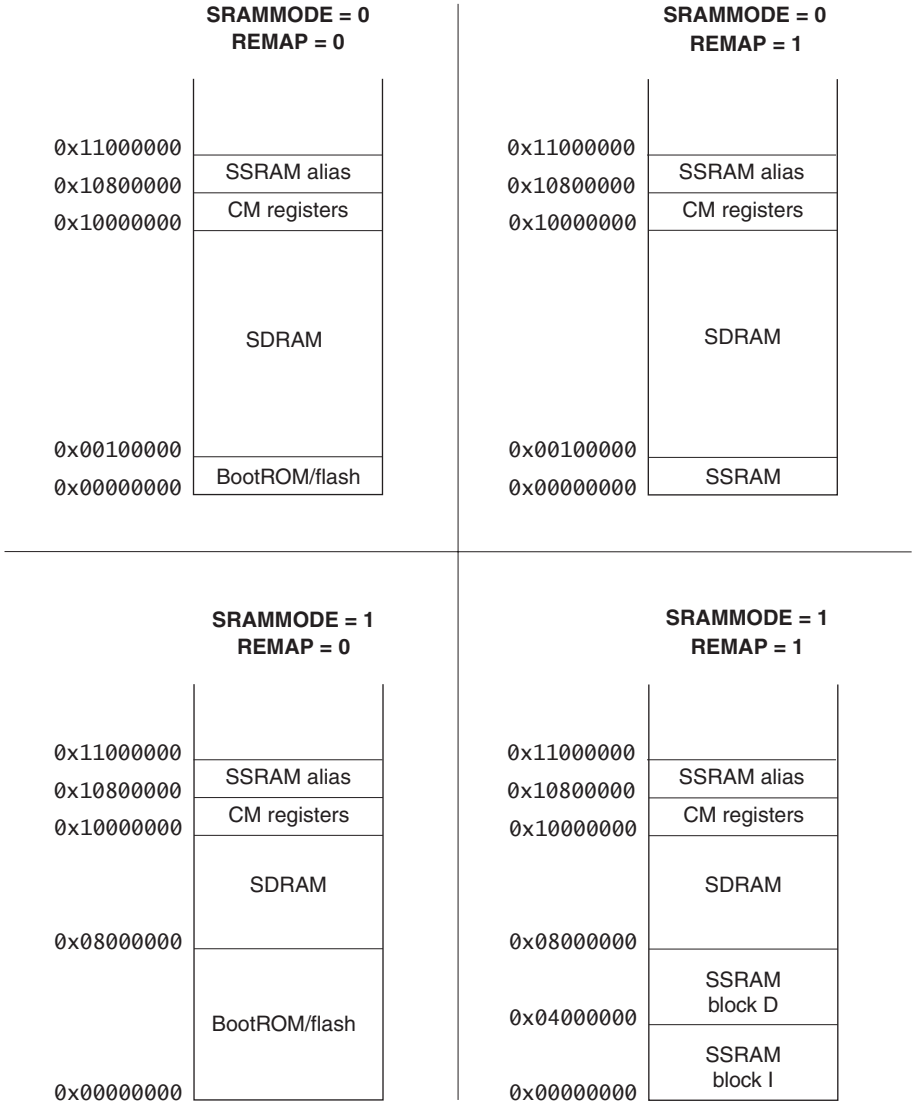


Figure 4-3 Configurable memory map for CM946E-S and CM966E-S

4.2.1 SSRAM mode selection (CM946E-S and CM966E-S only)

The on-board SSRAM appears within the configurable region and at an alias location (see *SSRAM alias* on page 4-10). Within the configurable region, you can map the SSRAM either in place of the TCRAM or in a similar way to previous Integrator core modules. This mapping is defined by the operating mode of the SSRAM as follows:

SSRAM Mode 0

This mode is similar to other Integrator core modules. The SSRAM appears as a 1MB block at `0x00000000–0x000FFFFF`. If the core module is fitted to a motherboard and the REMAP bit is 0, then accesses within `0x00000000–0x000FFFFF` are routed to the boot ROM or flash on the motherboard (see *Using REMAP* on page 4-9).

Note

This the only mode available for the CM926EJ-S.

SSRAM Mode 1

This mode uses the on-board SSRAM to emulate the TCRAM. The SSRAM is organized as two 512KB blocks. The first of these (block I) is accessed within the address space `0x00000000-x03FFFFFF` and the second (block D) within the address space `0x04000000–0x07FFFFFF`. Each of these 64MB regions is filled with 128 aliases of the associated SSRAM block.

The two modes of operations are selected using the SRAMMODE bit in the CM_INIT register (see *Core module initialization register* on page 4-23).

Changing the SSRAM mode also affects the amount of SDRAM that can be accessed. The SDRAM always appears contiguously at the top of the SSRAM. Table 4-2 shows how the configurable area of the memory map appears for the two SSRAM modes.

Table 4-2 Effect of SSRAM mode 0 and mode 1

SRAMMODE	Address range	Size	Description
0	`0x00000000–0x000FFFFF`	1MB	SSRAM
0	`0x00100000–0x0FFFFFFF`	255MB	SDRAM
1	`0x00000000–0x03FFFFFF`	64MB	SSRAM block I (128 copies of block I)
1	`0x04000000–0x07FFFFFF`	64MB	SSRAM block D (128 copies of block D)
1	`0x08000000–0x0FFFFFFF`	128MB	SDRAM

4.2.2 Using REMAP

When the core module is mounted on an Integrator motherboard, the mapping of the boot ROM on the motherboard or the SSRAM can be changed using the REMAP bit in the CM_CTRL register (see *Core module control register* on page 4-17).

Note

Program execution normally begins at `0x00000000`. A switch on the motherboard determines whether the boot ROM or flash is mapped to this location, enabling you to boot from the boot ROM or from flash memory. See the user guide for your motherboard for more information.

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

For the CM946E-S and CM966E-S, there is an interaction between the REMAP bit and the SSRAM mode as shown in Figure 4-3 on page 4-7. The REMAP bit functions as follows:

REMAP=0 In SSRAM mode 0, the boot ROM or flash on the motherboard is mapped into the address range `0x00000000`–`0x000FFFFF`. The SDRAM is accessible from `0x00100000`.

In SSRAM mode 1, the boot ROM or flash on the motherboard is multiply mapped into the address range `0x00000000`–`0x07FFFFFF`. The SDRAM is accessible from `0x08000000`–`0x0FFFFFFF`.

REMAP=1 In SSRAM mode 0, the SSRAM is mapped to the address range `0x00000000`–`0x000FFFFF` (as one 1MB block).

In SSRAM mode 1, the first block (block I) of SSRAM is multiply mapped into the address range `0x00000000`–`0x03FFFFFF` and the second block (block D) of SSRAM is multiply mapped into `0x04000000`–`0x07FFFFFF`.

For the CM926EJ-S, CM1026EJ-S, and the CM1136JF-S, the REMAP bit functions as follows:

REMAP=0 The boot ROM or flash on the motherboard is mapped into the address range `0x00000000`–`0x000FFFFF`. The SDRAM is accessible from `0x00100000`.

REMAP=1 The SSRAM is mapped to the address range `0x00000000`–`0x000FFFFF` (as one 1MB block).

4.3 SSRAM alias

The SSRAM is always accessible at `0x10800000`–`0x10FFFFFF`, independently of whether the TCRAM is enabled or disabled, and of the SRAMMODE, and REMAP controls. The SSRAM is mapped repeatedly to fill the SSRAM alias region. However, the mapping of each SSRAM repeat depends on the SSRAM mode in operation:

- In SSRAM mode 0, each repeat contains the whole 1MB of SSRAM as one block. This is the only mode available for the CM926EJ-S, CM1026EJ-S, and CM1136JF-S.
- In SSRAM mode 1 (CM946E-S and CM966E-S only), each repeat contains a 512KB alias of block I and a 512KB alias of block D, (see *SSRAM mode selection (CM946E-S and CM966E-S only)* on page 4-8).

The mapping of the aliases for the two SSRAM modes is shown in Figure 4-4.

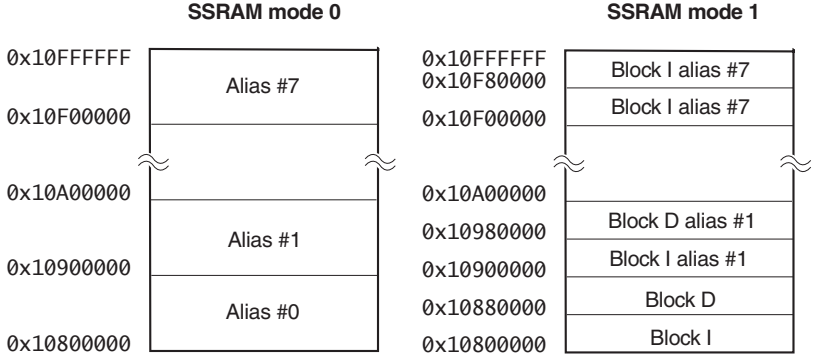


Figure 4-4 SSRAM aliases

4.4 SDRAM mapping

The Integrator memory map provides two regions in which SDRAM can be accessed. One region provides access by the local processor, and the second allows access by any master within the Integrator system. The global access region is not available on a standalone core module. For information on global access, see *Global SDRAM access* on page 5-7 and *Top level memory map* on page 6-7.

4.4.1 Local access

The amount of SDRAM that can be accessed within this region depends upon the SSRAM mode. Accesses to the SDRAM in this region are restricted to the core on the same core module. In SSRAM mode 0, the SDRAM is mapped to a 255MB space at `0x00100000`–`0x0FFFFFFF`. In SSRAM mode 1, the SDRAM is mapped to a 128MB space at `0x08000000`–`0x0FFFFFFF`.

You can fit SDRAM DIMMs of up to 256MB to the core module but you cannot access the lower 1MB (SSRAM mode 0) or 128MB (SSRAM mode 1) of the DIMM within the local access region. The SDRAM DIMM is mapped repeatedly within the accessible space but the SSRAM masks part of the SDRAM local access region. This means, for example, that in SSRAM mode 1, the first eight aliases of a 16MB DIMM are masked, whereas in SSRAM mode 0, only 1MB of the first alias is masked.

Figure 4-5 shows the repeat mapping of a 64MB SDRAM with SSRAM mode 0 in operation. The lowest image is partly masked by the SSRAM, but the repeat images are fully accessible.

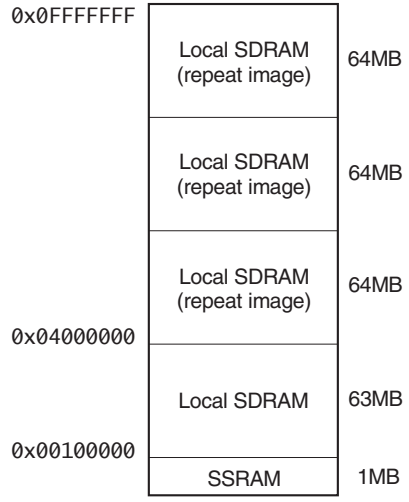


Figure 4-5 Repeat DRAM mapping

4.5 Processor configuration

The ARM926EJ-S, ARM946E-S, ARM966E-S, ARM1026EJ-S, and ARM1136JF-S processors provide several configuration options. Configuration options include:

- TCRAM
- vector locations
- byte order (big or little-endian).

These options are described in *Test chip configuration control* on page 3-4.

4.5.1 Limitations

The core module and Integrator motherboards have no physical memory at the high vector location `0xFFFF0000`. When the core module is being used with a motherboard, it is possible to implement an area of physical memory at this address on a logic module.

The core modules support both big-endian and little-endian byte ordering, however the Integrator/AP motherboard only supports little-endian byte ordering. Core modules, therefore, always default to little-endian.

4.5.2 Identifying the test chip type on your core module

To identify the manufacturer and type of ARM core fitted to your core module, read the SI_ID bits in the CM_STAT registers (see *Core module status register* on page 4-15). These bits are set by resistors that are fitted to the board at build time. It is important to be able to identify the type of core fitted because different manufacturers implement different features in the test chip. These differences typically relate to the control signals for the PLL clock dividers, to the size of TCRAM, and to the cache sizes.

4.6 Core module control registers

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

Table 4-3 Core module status, control, and interrupt registers

Register Name	Address	Access	Reset	Description
CM_ID	`0x10000000`	R	Static	Core module identification register
CM_PROC	`0x10000004`	R	Static	Core module processor register
CM_OSC	`0x10000008`	R/W	POR	Core module oscillator register
CM_CTRL	`0x1000000C`	R/W	Reset	Core module control
CM_STAT	`0x10000010`	R	Reset	Core module status
CM_LOCK	`0x10000014`	R/W	Reset	Core module lock
CM_LMBUSCNT	`0x10000018`	R	Reset	Core module local memory bus cycle counter
CM_AUXOSC	`0x1000001C`	R/W	POR	Core module auxiliary clock oscillator register
CM_SDRAM	`0x10000020`	R/W	POR	SDRAM status and control register
CM_INIT	`0x10000024`	R/W	POR	Core module initialization register
CM_REFCT	`0x10000028`	R	Reset	Reference clock cycle counter
CM_UNUSED1	`0x1000002C`	-	-	Reserved
CM_FLAGS	`0x10000030`	R	Reset	Core module flag register
CM_FLAGSS	`0x10000030`	W	Reset	Core module flag set register
CM_FLAGSC	`0x10000034`	W	Reset	Core module flag clear register
CM_NVFLAGS	`0x10000038`	R	POR	Core module non-volatile flag register
CM_NVFLAGSS	`0x10000038`	W	POR	Core module non-volatile flag set register
CM_NVFLAGSC	`0x1000003C`	W	POR	Core module non-volatile flag clear register
CM_IRQ_STAT	`0x10000040`	R	Reset	Core module IRQ status register
CM_IRQ_RSTAT	`0x10000044`	R	Reset	Core module IRQ raw status register

Table 4-3 Core module status, control, and interrupt registers (continued)

Register Name	Address	Access	Reset	Description
CM_IRQ_ENSET	0x10000048	R	Reset	Core module IRQ enable set register
CM_IRQ_ENCLR	0x1000004C	W	Reset	Core module IRQ enable clear register
CM_SOFT_INTSET	0x10000050	R/W	Reset	Core module software interrupt set
CM_SOFT_INTCLR	0x10000054	W	Reset	Core module software interrupt clear
CM_FIQ_STAT	0x10000060	R	Reset	Core module FIQ status register
CM_FIQ_RSTAT	0x10000064	R	Reset	Core module FIQ raw status register
CM_FIQ_ENSET	0x10000068	R/W	Reset	Core module FIQ enable set register
CM_FIQ_ENCLR	0x1000006C	W	Reset	Core module FIR enable clear register
CM_VOLTAGE_CTL0	0x10000080	R	Reset	Core module voltage-control register 0
CM_VOLTAGE_CTL1	0x10000084	R	Reset	Core module voltage-control register 1
CM_VOLTAGE_CTL2	0x10000088	R	Reset	Core module voltage-control register 2
CM_VOLTAGE_CTL3	0x1000008C	R/W	Reset	Core module voltage-control register 3
CM_SPD	0x10000100– 0x100001FC	R	POR	SSRAM SPD memory

Note

All registers are 32-bits wide and only support word-wide writes. Bits marked as *reserved* in the following sections must be preserved using read-modify-write operations.

4.6.1 Core module ID register

The core module ID register, CM_ID, at 0x10000000 is a read-only register that identifies the board manufacturer, board type, and revision.

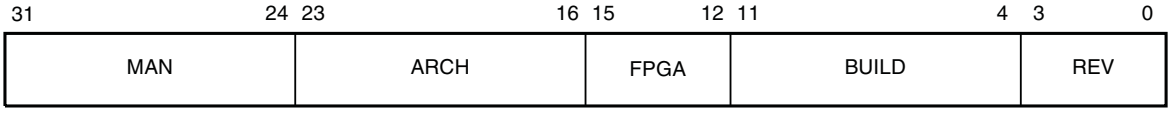


Figure 4-6 Core module ID

Table 4-4 and Figure 4-6 on page 4-14 describe the core module ID register bits.

Table 4-4 CM_ID register bit descriptions

Bits	Name	Access	Function
[31:24]	MAN	Read	Manufacturer: 0x41 = ARM
[23:16]	ARCH	Read	Architecture: 0x1A = AHB interface, 64-bit SDRAM
[15:12]	FPGA	Read	FPGA type: 0x3 = XCV600 or XCV600E
[11:4]	BUILD	Read	Build value (FPGA build)
[3:0]	REV	Read	Revision: 0x0 = Rev A 0x1 = Rev B

4.6.2 Core module processor register

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). Information about the processor can be obtained by reading coprocessor 15 register 0 (CP15 c0).

4.6.3 Core module status register

The core module status register, CM_STAT, at 0x10000010 is a read-only register that can be read to determine the size of the SSRAM present, the test chip type (see *Processor configuration* on page 4-12), and where in a stack this core module is positioned.

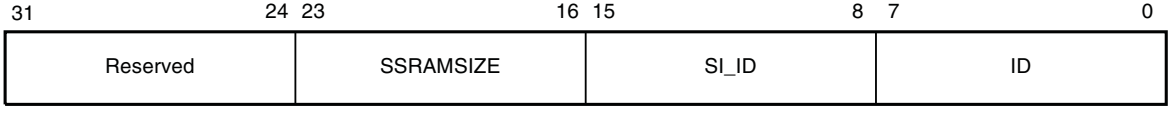


Figure 4-7 Core module status register

Table 4-5 and Figure 4-7 on page 4-15 describe the core module status register bits.

Table 4-5 CM_STAT register

Bit	Name	Access	Function
[31:24]	Reserved	Use read-modify-write to preserve value.	
[23:16]	SSRAMSIZE	Read	SSRAM size. This contains `0x10` and indicates that 1MB is fitted.

[15:8]	SI_ID	Read	Silicon manufacturer identification. Identifies the manufacturer and type of test chip fitted to the module.
			For the CM966E-S and CM946E-S:
			`0x00 = unknown or socket fitted`
			`0x01 = Lucent M-1291 ARM966E-S r0 (3V3 core)`
			`0x02 = LSI L9A0225 ARM966E-S r0 (2V5 core)`
			`0x03 = LSI L9A231 ARM966E-S r0 (1V8 core)`
			`0x04 = TI F741823/AX ARM966E-S r0 (1V8 core)`
			`0x05 = LSI L9A267 ARM946E-S r0a (2V5 core)`
			`0x06 = NEC 14212901 ARM946E-S r1 (1V5 core)`
			`0x07-0xFF = reserved.`
			For the CM926EJ-S, CM1026EJ-S, or CM1136JF-S:
			`0x00 = unknown or socket fitted`
			`0x01 = TI 1533C035.1 ARM926EJ-S r0p3 (1V5 core)`
			`0x02 = TSMC ARM926EJ-S r0p2 (1V3 core)`
			`0x03 = ST ARM926EJ-S r0p3 (1V3 core)`
			`0x04-0x05 = reserved`
			`0x06 = TSMC ARM1136JF-S r0p1 (1V2 core)`
			`0x07 = TSMC ARM1026EJ-S r0p1 (1V2 core)`
			`0x08-0xFF = reserved.`

[7:0]	ID	Read	Card number in stack: `0x00` = core module 0 `0x01` = core module 1`0x02` = core module 2 `0x03` = core module 3 `0xFF` = invalid or no motherboard attached.

4.6.4 Core module oscillator register

The core module oscillator register, CM_OSC, at `0x10000008` is a read/write register that controls the frequency of the clock generated by the clock generator for **ARM_PLLCLKIN** (see *Programming the clocks* on page 3-13).

Note

The CM1136JF-S clock control has additional clock registers and has a different format for this register. See Appendix C *Features specific to the CM1136JF-S* for details.

Figure 4-8 Core module oscillator register

Note

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

Table 4-6 and Figure 4-8 describe the core module oscillator register bits.

Table 4-6 CM_OSC register

Bits	Name	Access	Function
[31:25]	Reserved	Use read-modify-write to preserve value.	
[24:23]	BMODE	Read	Bus mode. Always contains `b10`. Write to CM_INIT to select bus operation.
[22:8]	Reserved	Use read-modify-write to preserve value.	
[7:0]	PLL_VDW	Read/write	Core clock VCO divider word. Defines the binary value on the V[7:0] pins of the clock generator for **ARM_PLLCKIN** (V[8] is tied LOW). `b01001000` = 80MHz (default). **Note** Some earlier versions of the ARM9x6 test chips used different clock and divisor combinations. For example, builds 19 to 22 used 120MHz (`b01110000`) for PLLCLKIN and divide by 3 (`b00000010`) for HCLKDIV.

4.6.5 Core module control register

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.

The FPGA image for the Integrator/AP provides different register function for the CM control register than the function provided by the Integrator/CP image. See *CM control register for Integrator/CP* on page 6-20 and *CM control register for Integrator/AP* on page 5-11 for details.

4.6.6 Core module lock register

The core module lock register, CM_LOCK, at `0x10000014` is a read/write register that is used to control access to the CM_OSC, CM_INIT, and CM_AUXOSC registers, allowing them to be locked and unlocked. This mechanism prevents these registers from being overwritten accidently.

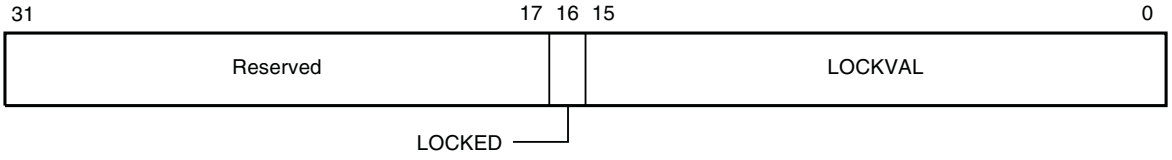


Figure 4-9 Core module lock register

Table 4-7 and Figure 4-9 describe the core module lock register bits.

Table 4-7 CM_LOCK register

Bits	Name	Access	Function
[31:17]	Reserved	Use read-modify-write to preserve value.	
[16]	LOCKED	Read	This bit indicates if the CM_OSC register is locked or unlocked: 0 = unlocked 1 = locked.
[15:0]	LOCKVAL	Read/write	Write the value `0x0000A05F` to this register to enable write accesses to the CM_OSC register. Write any other value to this register to lock the CM_OSC register.

4.6.7 Core module local memory bus cycle counter

This 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 reset to zero by a reset.

4.6.8 Core module auxiliary oscillator register

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 clocks* on page 3-13). 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 32.369MHz output.

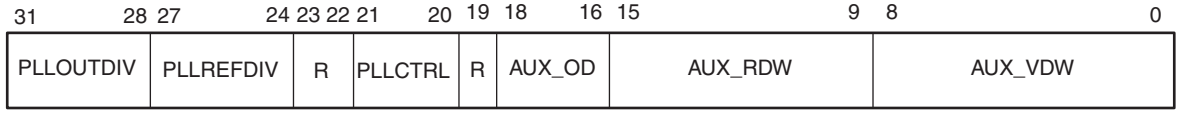


Figure 4-10 Core module auxiliary oscillator register

Note

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

Table 4-8 and Figure 4-10 describe the core module oscillator register bits.

Table 4-8 CM_AUXOSC register

Bits	Name	Access	Function
[31:28]	PLLOUTDIV	Read/write	(CM926EJ-S, CM1026EJ-S, and CM1136JF-S only) test chip PLL output divider
[27:24]	PLLREFDIV	Read/write	(CM926EJ-S, CM1026EJ-S, and CM1136JF-S only) test chip PLL reference divider
[23:22]	Reserved	Use read-modify-write to preserve value.	
[21:20]	PLLCTRL	Read/write	(CM926EJ-S, CM1026EJ-S, and CM1136JF-S only) test chip PLL control
[19]	Reserved	Use read-modify-write to preserve value.	

Table 4-8 CM_AUXOSC register (continued)

Bits	Name	Access	Function
[18:16]	AUX_OD	Read-write	Auxiliary output divider. Sets the binary code on the S[2:0] pins of the clock generator. The divider is encoded as follows: 000 = divide by 10 001 = divide by 2 010 = divide by 8 011 = divide by 4 100 = divide by 5 101 = divide by 7 110 = divide by 3 111 = divide by 6 (default).
[15:9]	AUX_RDW	Read/write	Auxiliary reference divider word. Defines the binary value on the R[6:0] pins of the clock generator. `b0111111` = 63 (default).
[8:0]	AUX_VDW	Read/write	Auxiliary clock VCO divider word. Defines the binary value on the V[8:0] pins of the clock generator. `b011111111` = 255 (default).

4.6.9 SDRAM status and control register

The 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.

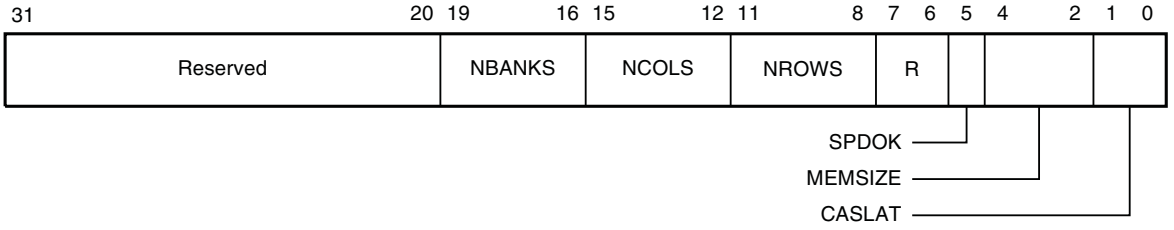


Figure 4-11 SDRAM status and control register

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 4-9. If these values are not correctly set then SDRAM accesses may be slow or unreliable. See *SDRAM SPD memory* on page 4-34.

Table 4-9 and Figure 4-11 describe the SDRAM status and control register bits.

Table 4-9 CM_SDRAM register

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

Table 4-9 CM_SDRAM register (continued)

Bits	Name	Access	Function
[5]	SPDOK	Read	This bit indicates that the automatic copying of the SPD data from the SDRAM module into CM_SPDMEM is complete: 1 = SPD data ready 0 = SPD data not available. See *SDRAM SPD memory* on page 4-34.
[4:2]	MEMSIZE	Read/write	These bits specify the size of the SDRAM module fitted to the core module. The bits are encoded as follows: 000 = 16MB 001 = 32MB 010 = 64MB (default) 011 = 128MB 100 = 256MB 101 = Reserved 110 = Reserved 111 = Reserved.
[1:0]	CASLAT	Read/write	These bits specify the CAS latency for the core module SDRAM. The bits are encoded as follows: 00 = Reserved 01 = Reserved 10 = 2 cycles (default) 11 = 3 cycles.

4.6.10 Core module initialization register

The core module initialization register, CM_INIT, at `0x10000024` is used to control the configuration pins of the testchip (see *Test chip configuration control* on page 3-4) and the mapping of the on-board SRAM (see *Using REMAP* on page 4-9).

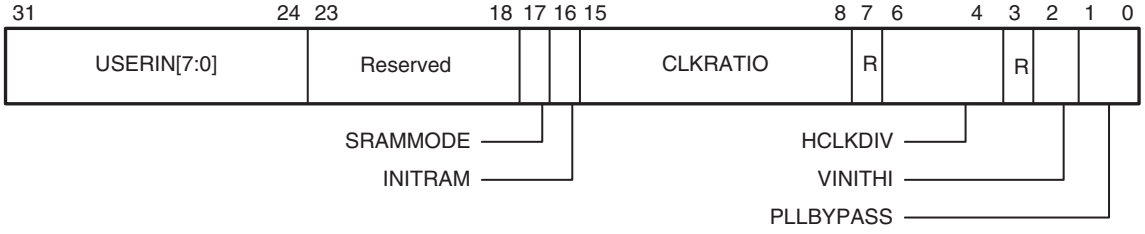


Figure 4-12 Core module initialization register

Note

Bit 3 of the CM_INIT register is reserved for CM946E-S and CM966E-S. For the CM926EJ-S, CM1026EJ-S, and CM1136JF-S however, bit 3 can be used to force the test chip into big-endian mode from reset.

For all core modules, CP15 can be used to control endianness.

Note

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

Table 4-10 and Figure 4-12 on page 4-23 describe the core module initialization register bits. The test chip samples the configuration values during reset. To reconfigure the test chip, set the required bit values shown in Table 4-10, and then press the reset button. The default settings are always in effect following a power-on reset.

Table 4-10 CM_INIT register

Bits	Name	Access	Function
[31:24]	USERIN[7:0]	Read	This bits connect with the USERIN[7:0] pins of the test chip. **Note** For the CM926EJ-S, CM1026EJ-S, and CM1136JF-S, there are only 6 input bits available in USERIN[5:0]. The remaining bits at USERIN[7:6] (register bits [31:30]) are reserved.
[23:18]	Reserved	Use read-modify-write to preserve value.	
[17]	SRAMMODE	Read/write	This bit controls the mapping of the on-board SSRAM for the CM946E-S and CM966E-S: 0 = SSRAM mode 0 (default after POR). 1 = SSRAM mode 1. Emulates the TCRAM. See *SSRAM mode selection (CM946E-S and CM966E-S only)* on page 4-8. **Note** This bit is reserved on the CM926EJ-S, CM1026EJ-S, and CM1136JF-S.
[16]	INITRAM (ARM966E-S and ARM946E-S only)	Read/write	This bit is used to enable or disable the TCRAM: 0 = TCRAM disabled 1 = TCRAM enabled. See *About the memory map* on page 4-2. **Note** For Rev 0 of the ARM946E-S test chips, the TCRAM is always disabled after POR rather than using the value of INITRAM.

Table 4-10 CM_INIT register (continued)

Bits	Name	Access	Function
[15:8]	CLKRATIO (PLLFBDIV)	Read/write	For the CM946E-S and CM966E-S, these are the CLKRATIO bits and are used to set the internal clock multiplier. The default value after POR is `b000001` (multiply by 2). See *Test chip clocks* on page 3-14. The CLKRATIO multiplier only has any effect if PLLBYPASS bit is 0. (For the CM926EJ-S, CM1026EJ-S, and CM1136JF-S, this is the PLL feedback divisor. The default value is `b00000001` at POR.)
[7]	Reserved	Use read-modify-write to preserve value.	
[6:4]	HCLKDIV	Read/write	These bits are used to set the internal **HCLK** divider, to divide **CLK** by n+1. The default value after POR is `b000001` (divide by 2) giving a default local memory bus clock speed of 40MHz. **Note** Some earlier versions of the ARM9x6 or ARM1026EJ-S test chips used different clock and divisor combinations. For example, builds 19 to 22 used 120MHz (`b01110000`) for PLLCLKIN and divide by 3 (`b00000010`) for HCLKDIV. See *Test chip clocks* on page 3-14. **Note** Normally the **CLK** to **HCLK** divider is changed as soon as the HCLKDIV value in the CM_INIT register is changed. This is not the case, however, if HCLKDIV is set to 0. The ratio remains the default value (**HCLK** equal to **CLK**/2) until a soft reset is generated.
[3]	Reserved	Use read-modify-write to preserve value.	

Table 4-10 CM_INIT register (continued)

Bits	Name	Access	Function
[2]	VINITHI	Read	This bit controls the mapping of the vectors 0 = vectors at 0 (default after POR) 1 = vectors at `0xFFFF0000`. See *Test chip configuration control* on page 3-4.
[1]	PLLBYPASS	Read	Current state on the PLLBYPASS pin of the testchip after a reset
[0]	PLLBYPASS	Write	Enables the PLL bypass feature of the testchip. The setting only takes effect after a reset. 0 = bypass OFF 1 = bypass ON (default after POR). See *Test chip configuration control* on page 3-4.

4.6.11 Core module reference clock cycle counter

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

4.7 Core module flag registers

The core module flag registers provide you with two 32-bit register locations containing general purpose flags. You can assign any meaning to the flags. The flag registers are shown in Table 4-12 on page 4-28.

Table 4-11 Core module flag registers

Register Name	Address	Access	Reset by	Description
CM_FLAGS	0x10000030	R	Reset	Core module flag register
CM_FLAGSS	0x10000030	W	Reset	Core module flag set register
CM_FLAGSC	0x10000034	W	Reset	Core module flag clear register
CM_NVFLAGS	0x10000038	R	POR	Core module nonvolatile flag register
CM_NVFLAGSS	0x10000038	W	POR	Core module nonvolatile flag set register
CM_NVFLAGSC	0x1000003C	W	POR	Core module nonvolatile flag clear register

The core module provides two distinct types of flag registers:

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

4.7.1 Flag/nonvolatile flag register

The status register contains the current state of the flags.

4.7.2 Flag/nonvolatile flag set register

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

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

4.7.3 Flag/nonvolatile flag clear register

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

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

4.8 Core module interrupt registers

The core module provides a 3-bit IRQ controller and 3-bit FIQ controller to support the debug communications channel used for passing information between applications software and the debugger. The interrupt control registers are shown in Table 4-12.

Table 4-12 Interrupt controller registers

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

Note

All registers are 32-bits wide and only support word-wide writes. Bits marked as *reserved* in the following sections must be preserved using read-modify-write operations.

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

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

The way that the interrupt enable, clear, and status bits function for each interrupt is illustrated in Figure 4-13 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.

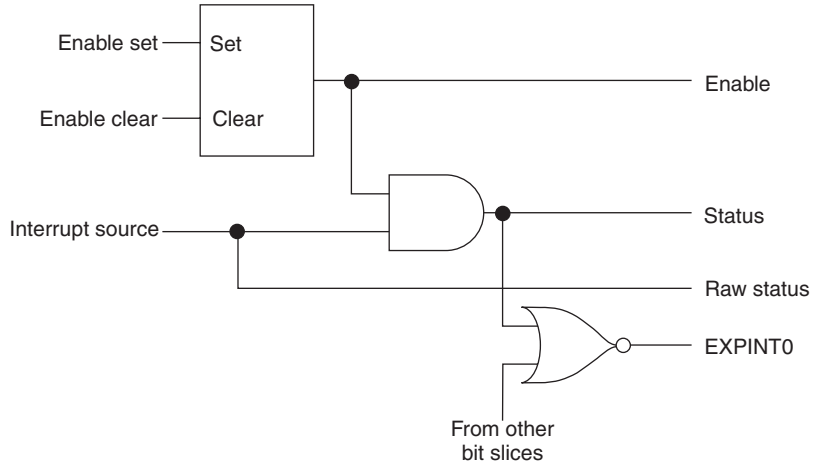


Figure 4-13 Interrupt control

4.8.1 IRQ/FIQ status register

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

4.8.2 IRQ/FIQ raw status register

The raw status register indicates the signal levels on the interrupt request inputs. A bit set to 1 indicates that the corresponding interrupt request is active.

4.8.3 IRQ/FIQ enable set register

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

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

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

4.8.4 IRQ/FIQ enable clear register

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

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

4.8.5 Interrupt register bit assignment

The bit assignments for the IRQ and FIQ status, raw status and enable register are shown in Table 4-13.

Table 4-13 IRQ and FIQ register bit assignment

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

4.8.6 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 the software interrupt set and software interrupt clear locations. The set and clear locations are used as follows:

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

The bit assignment for the software interrupt register is shown in Table 4-14.

Table 4-14 IRQ register bit assignment

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

Note

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

4.9 Voltage control registers

The bit assignments for the registers used for reading and setting core voltages are shown in Table 4-15 to Table 4-18 on page 4-33. See *Power supply control (CM926EJ-S, CM1026EJ-S, and CM1136JF-S only)* on page 3-31 for details on reading and writing voltages levels. VDDCORE_DIFF*x* values are only valid if a sense-resistor is installed instead of the 0Ω resistor.

Table 4-15 CM_VOLTAGE_CTL0 register (0x80)

Bit	Name	Function
[31:20]	VDDCORE_DIFF1	Value proportional to the current to VDDCORE1
[19:8]	VDDCORE	Value proportional to the core voltage (voltage is measured before the sense resistors)
[7:0]	Core DAC value	Positive or negative offset to the default power-on voltage (`0x80` is the default value)

Table 4-16 CM_VOLTAGE_CTL1 register (0x84)

Bit	Name	Function
[31:20]	VDDCORE_DIFF2	Value proportional to the current to VDDCORE2
[19:8]	VDDIO	Value proportional to the I/O voltage
[7:0]	Reserved	Write as 0. Reads undefined.

Table 4-17 CM_VOLTAGE_CTL2 register (0x88)

Bit	Name	Function
[31:20]	VDDCORE_DIFF3	Value proportional to the current to VDDCORE3
[19:8]	VDDLEVEL	Value proportional to the logic-level voltage
[7:0]	Reserved	Write as 0. Reads undefined.

Table 4-18 CM_VOLTAGE_CTL3 register (0x8C)

Bit	Name	Function
[31:20]	VDDCORE_DIFF4	Value proportional to the current to VDDCORE4
[19:8]	VDDPLL	Value proportional to the PLL voltage
[7:0]	Reserved	Write as 0. Reads undefined.

4.10 SDRAM SPD memory

This area of memory contains a copy of the SPD data from the SPD EEPROM on the DIMM. Because accesses to the EEPROM are very slow, the data is copied to this memory during board initialization to allow faster random access to the SPD data (see *Serial presence detect* on page 3-8). The SPD memory contains 256 bytes of data, the most important of these are as shown in Table 4-19.

Table 4-19 SPD memory contents

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

Check for valid SPD data as follows:

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

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

Note

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

The code segment shown in Example 4-1 on page 4-35 can be used to correctly setup and remap the SDRAM.

Example 4-1

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

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

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

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

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

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

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

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

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


Chapter 5
Using Core Modules with an Integrator/AP

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

5.1 About the 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.

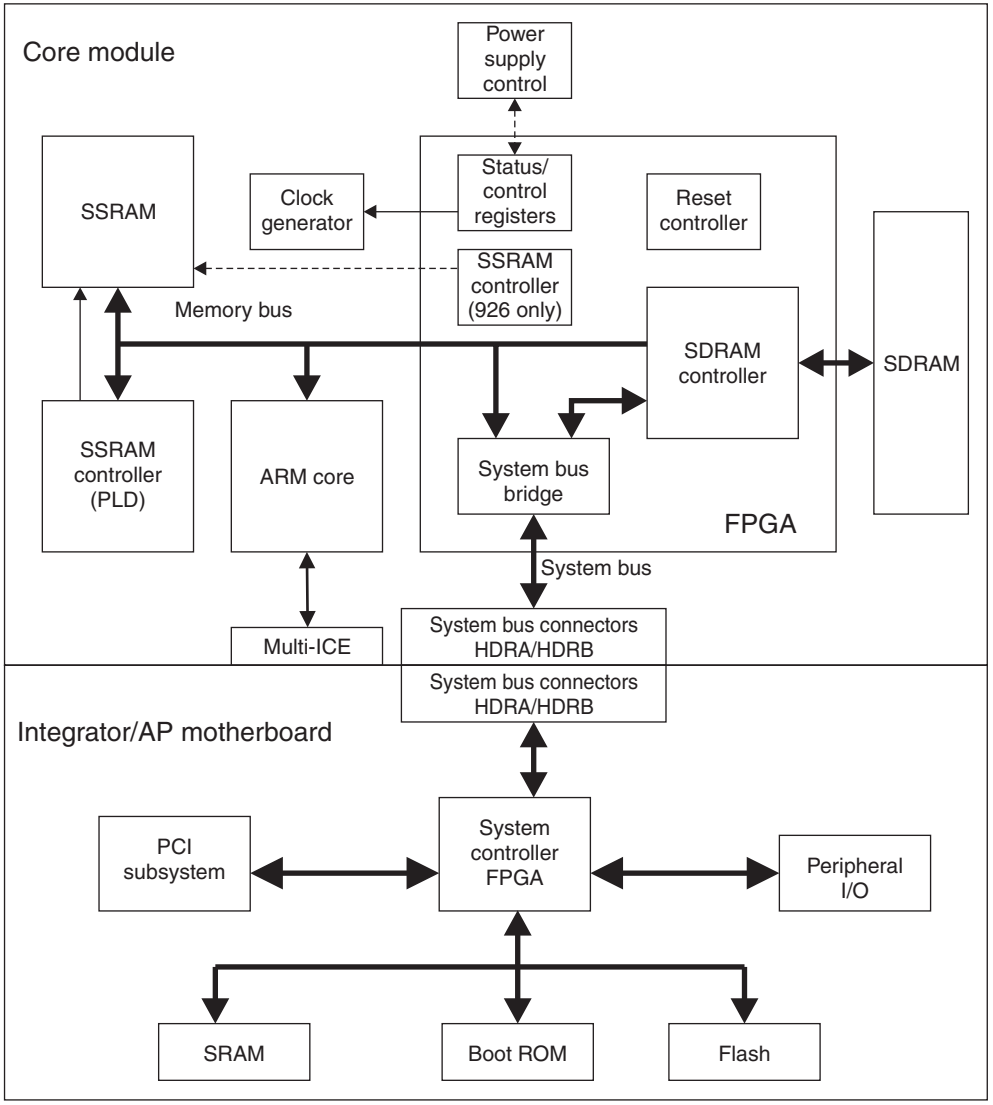


Figure 5-1 FPGA functional diagram

5.1.1 Configuring little or big-endian operation

The core module can be configured to operate as a big-endian or little-endian system. To change to big-endian operation, write to the appropriate register in CP15 on the ARM core.

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

Caution

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

5.2 Module ID selection

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

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

Note

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

5.2.1 Module address decoding

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

The signals **nPPRES[3:0]** (core module present) and **nEPRES[3:0]** (logic module present) are used to signal the presence of modules to the central decoder. 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 5-18. 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 *Core module status register* on page 4-15). Table 5-1 shows alias addresses for a core module fitted to the motherboard HDRA/HDRB stack.

Table 5-1 Core module address decode

ID[3:0]	Module ID	Address range	Size
1101	3 (top)	0xB0000000–0xBFFFFFFF	256MB

Table 5-1 Core module address decode (continued)

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

5.3 Top level memory map

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

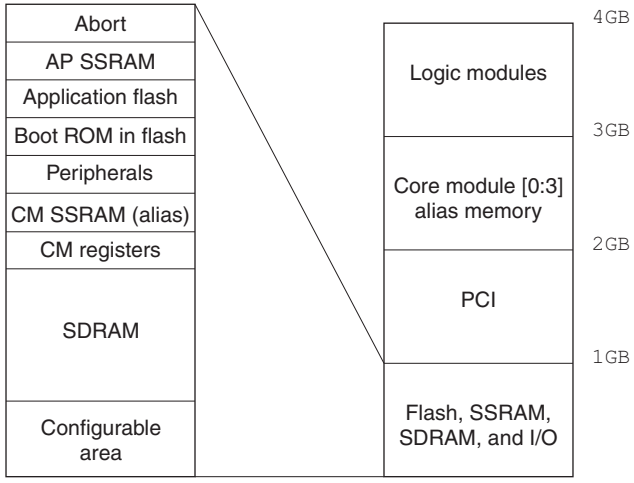


Figure 5-2 Top-level memory map

Note

The processor cores on different boards might have different amounts of cache and TCM. The size of the cache and TCM on a board can be identified by reading registers in CP15.

See the *Integrator/AP User Guide* and Chapter 4 *Programmer's Reference* for more details on memory mapping and register function. The FPGA image for the Integrator/AP provides different register function for the CM control register than the function provided by the FPGA image for the Integrator/CP. See *CM control register for Integrator/AP* on page 5-11 for details.

Note

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

5.3.1 Global SDRAM access

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

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

A processor can determine which core module it is on and, therefore, the alias location of its own SDRAM by reading the CM_STAT register (see *Core module status register* on page 4-15).

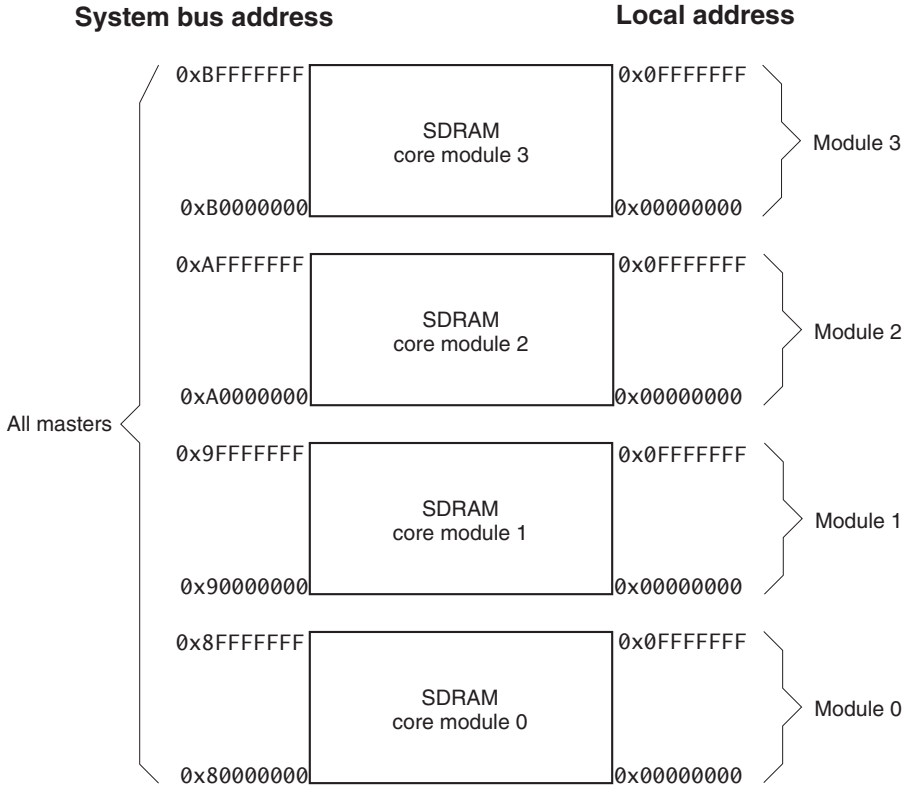


Figure 5-3 Core module local and alias addresses

5.3.2 Access arbitration

The SDRAM controller provides two ports to support reads and writes by the local processor core and by masters on the motherboard. The SDRAM controller uses an alternating priority scheme to ensure that the processor core and motherboard have equal access (see *System bus bridge* on page 5-13).

5.4 Register and memory overview

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

Note

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

Table 5-2 System control register map

Peripheral or device	Address range	Size
Boot flash. Mapped at this address only at power ON, and then can be disabled under software control to allow access to SSRAM. **Note** The memory map below `0x10000000` depends on the settings for SSRAM mode, TCM enabled, and REMAP.	`0x00000000–0x000FFFFF`	1MB
SDRAM.	`0x00010000–0x0FFFFFFF`	255MB
Core Module control registers.	`0x10000000-0x1000003F`	64 bytes
Core Module interrupt controller.	`0x10000040–0x1000007F`	64 bytes
Reserved **Note** The CM926, CM1026, and CM1136 use `0x10000080–0x10000008C` as voltage control registers. See *Voltage control registers* on page 4-32.	`0x10000080–0x100000FF`	128 bytes
Serial Presence Detect memory.	`0x10000100–0x100001FF`	256 bytes
AP system controller registers.	`0x11000000–0x11FFFFFF`	16MB
AP EBI configuration registers.	`0x12000000–0x12FFFFFF`	16MB
AP Counter/timers.	`0x13000000–0x13FFFFFF`	16MB
AP interrupt controller registers.	`0x14000000–0x14FFFFFF`	16MB

Table 5-2 System control register map (continued)

Peripheral or device	Address range	Size
AP Real time clock.	0x15000000–0x15FFFFFF	16MB
AP UART0.	0x16000000–0x16FFFFFF	16MB
AP UART1.	0x17000000–0x17FFFFFF	16MB
AP Keyboard.	0x18000000–0x18FFFFFF	16MB
AP Mouse.	0x19000000–0x19FFFFFF	16MB
AP Debug LEDs and DIP switch.	0x1A000000–0x1AFFFFFF	16MB
AP GPIO.	0x1B000000–0x1BFFFFFF	16MB
Reserved (abort if accessed).	0x1C000000–0x1FFFFFFF	64MB
AP External bus interface.	0x20000000–0x2FFFFFFF	256MB
Reserved (abort if accessed).	0x30000000–0x3FFFFFFF	256MB
AP PCI memory space.	0x40000000–0x7FFFFFFF	1GB
Core module 0 alias memory.	0x80000000-0x8FFFFFFF	256MB
Core module 1 alias memory.	0x90000000–0x9FFFFFFF	256MB
Core module 2 alias memory.	0xE0000000–0xEFFFFFFF	256MB
Core module 3 alias memory.	0xF0000000–0xFFFFFFFF	256MB
Logic module 0 memory.	0xC0000000–0xCFFFFFFF	256MB
Logic module 1 memory.	0xD0000000–0xDFFFFFFF	256MB
Logic module 2 memory.	0xE0000000–0xEFFFFFFF	256MB
Logic module 3 memory.	0xF0000000–0xFFFFFFFF	256MB

Note

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

5.4.1 CM control register for Integrator/AP

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. The functionality in this section is for the FPGA image for use with a standalone core module or with the core module connected to an Integrator/AP motherboard. See the *Integrator/AP User Guide* and Chapter 4 *Programmer's Reference* for more details on memory mapping and register function.

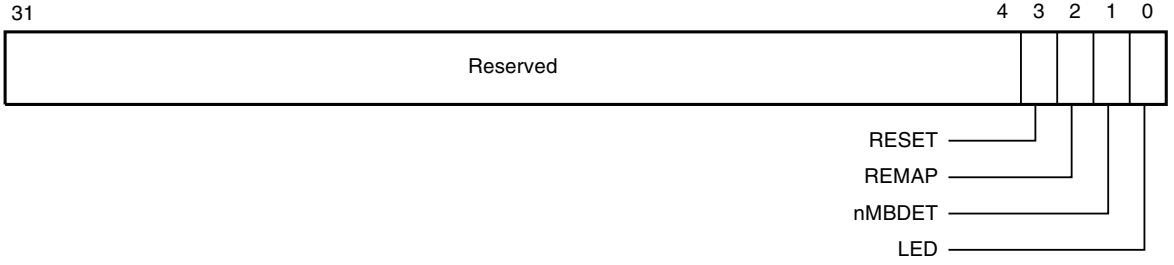


Figure 5-4 Core module control register

Table 5-3 and Figure 5-4 describe the core module control register bits

Table 5-3 CM_CTRL register

Bits	Name	Access	Function
[31:4]	Reserved	Use read-modify-write to preserve value.	
[3]	RESET	Write	This is used to reset the core module, the motherboard on which it is mounted, and any modules in a stack. A reset is triggered by writing a 1. Reading this bit always returns a 0 allowing you to use read-modify-write operations without masking the RESET bit.

Table 5-3 CM_CTRL register (continued)

Bits	Name	Access	Function
[2]	REMAP	Read/write	This only has affect when the core module is mounted on a motherboard. The function of remap is described in *Using REMAP* on page 4-9.
[1]	nMBDET	Read	This bit indicates whether or not the core module is mounted on a motherboard: 0 = mounted on motherboard 1 = standalone.
[0]	LED	Read/write	This bit controls the green MISC LED on the core module: 0 = LED OFF 1 = LED ON.

5.5 System bus bridge

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 controller are supported by a single-entry 74-bit buffer that contains:

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

Note

The FIFO depth for system bus bridges in synchronous mode is typically 16 entries.

However, the system bridges in the core modules do not have a FIFO. A buffer is used instead which gives an equivalent FIFO depth of 1.

5.5.1 Processor accesses to the system bus

This section gives an overview of system bus accesses. System bus accesses do not use the FIFO (actually a single stage buffer).

Processor writes

The data routing for processor writes to the system bus is illustrated in Figure 5-5 on page 5-14.

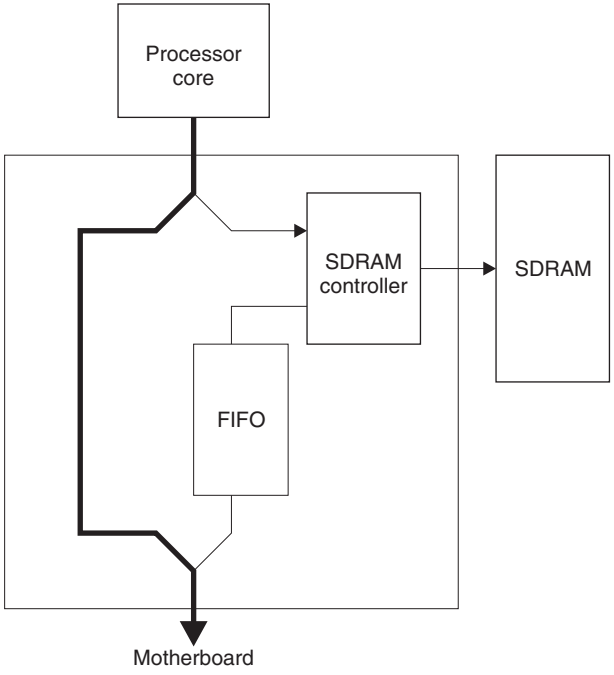


Figure 5-5 Processor writes to the system bus

The data, address, and control information associated with the transfer are posted into a buffer, and the transfer on the system bus occurs some time later when that bus is available. System bus error responses to write transfers are reported back to the processor as Data Aborts.

For some other core modules, a full buffer results in the processor receives a wait response until space becomes available. The FIFO is only a buffer, so only the first access completes in one cycle.

Processor reads

The data routing for processor reads from the system bus is illustrated in Figure 5-6 on page 5-15.

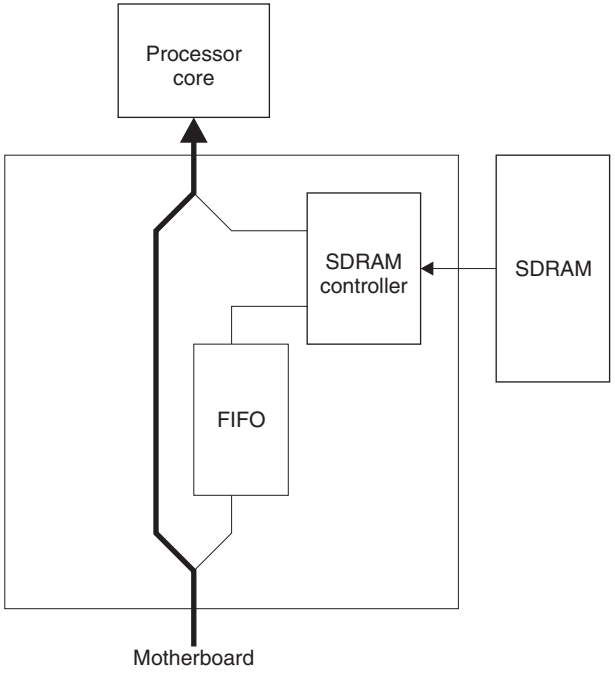


Figure 5-6 Processor reads from the system bus

The order of processor transactions is preserved on the system bus. The processor receives a wait response until the read transfer has completed on the system bus, after that it receives the data and any associated bus error response from the system bus.

5.5.2 Motherboard accesses to SDRAM

This section gives an overview of SDRAM access (see also *SDRAM mapping* on page 4-11.)

System bus writes

The data routing for system bus writes to SDRAM is illustrated in Figure 5-7 on page 5-16.

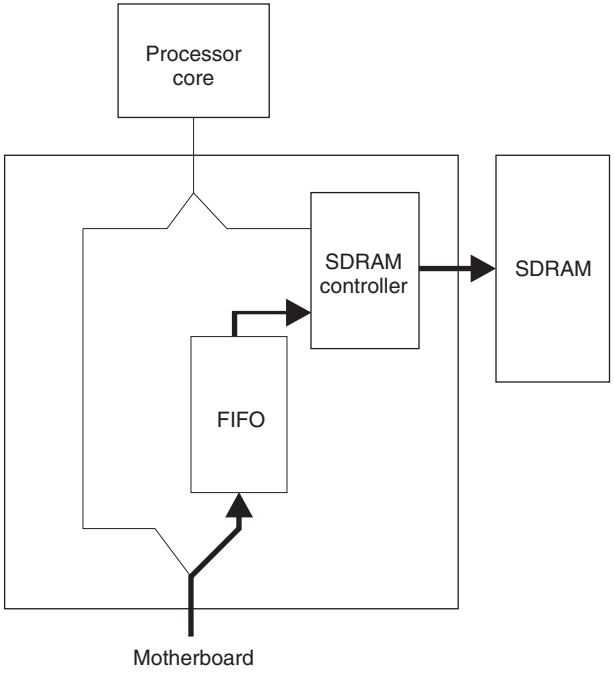


Figure 5-7 System bus writes to SDRAM

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

System bus reads

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

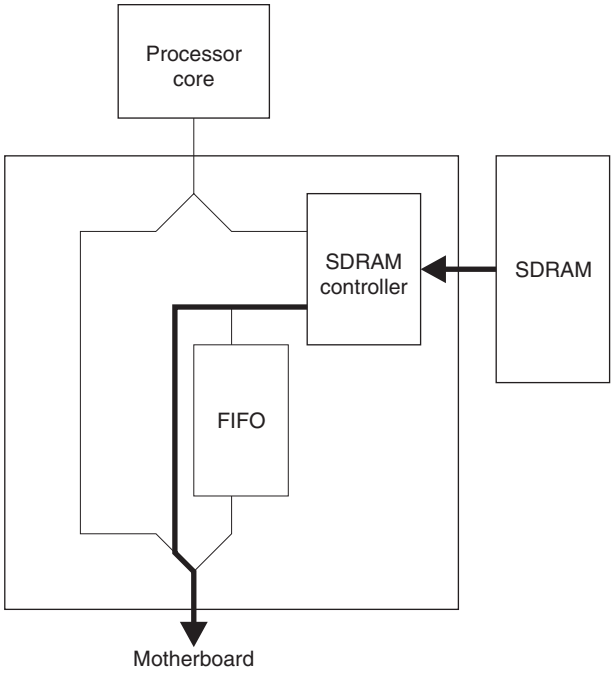


Figure 5-8 System bus reads from SDRAM

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

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

5.5.3 Multiprocessor support

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

5.5.4 System bus signal routing

The core module is mounted onto an Integrator/AP or Integrator/CP using the HDRA and HDRB connectors.

HDRA

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

HDRB

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

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

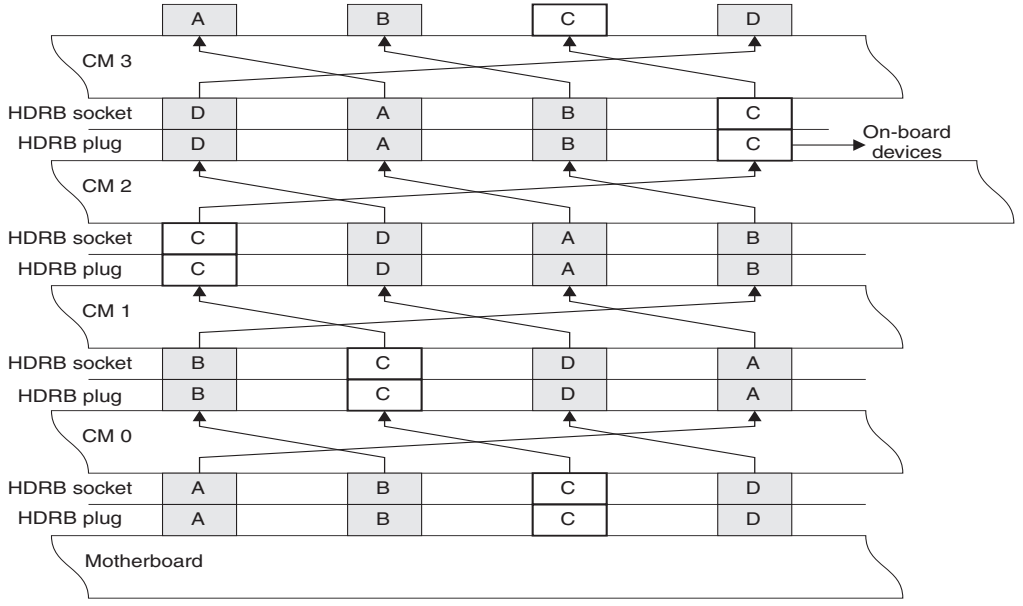


Figure 5-9 Signal rotation on HDRB (Integrator/AP)

The example in Figure 5-9 on page 5-18 shows how a group of four signals (labeled A, B, C, and D) are routed through the stack. Signal C is rotated as it passes up through the stack and only utilized on module 2.

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

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

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

Note

Only one core module can be used with the Integrator/CP. Additional logic modules can be used however.

5.6 Reset controller

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

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

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

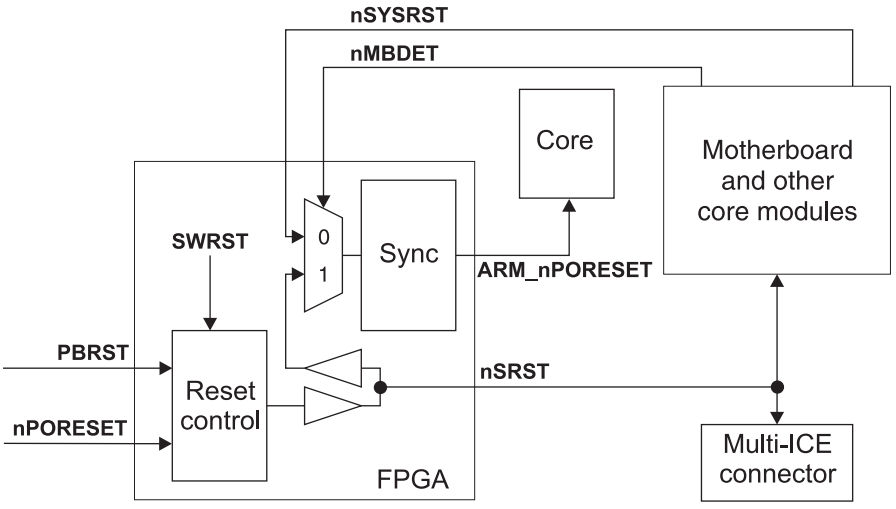


Figure 5-10 Core module reset controller

5.6.1 Reset control signals

Table 5-4 describes the external reset signals.

Table 5-4 Reset signal descriptions

Name	Description	Type	Function
ARM_nPORESET	Processor reset	Output	The **ARM_nPORESET** signal is used to reset the processor core. It is generated from **nSRST** LOW when the core module is used standalone, or **nSYSRST** LOW when the core module is attached to a motherboard. It is asserted as soon as the appropriate input becomes active. It is deasserted synchronously from the falling edge of the processor bus clock.
nMBDET	Motherboard detect	Input	The **nMBDET** signal is pulled LOW when the core module is attached to a motherboard and HIGH when the core module is used standalone. When **MBDET** is LOW, **nSYSRST** is used to generate the **ARM_nPORESET** signal. When **nMBDET** is HIGH, **nSRST** is used to generate the **ARM_nPORESET** signal.
PBRST	Push-button reset	Input	The **PBRST** signal is generated by pressing the reset button.
nSRST	System reset	Bidirectional	The **nSRST** open collector output signal is driven LOW by the core module FPGA when the signal **PBRST** or software reset (**SWRST**) is asserted. As an input, **nSRST** can be driven LOW by Multi-ICE. If there is no motherboard present, the **nSRST** signal is synchronized to the processor bus clock to generate the **ARM_nPORESET** signal.
nSYSRST	System reset	Input	The **nSYSRST** signal is generated by the system controller FPGA on the motherboard. It is used to generate the **ARM_nPORESET** signal when the core module is attached to a motherboard. It is selected by the motherboard detect signal (**nMBDET**).

5.6.2 Software resets

The core module FPGA provides a software reset that can be triggered by writing to the reset bit in the CM_CTRL register. This generates the internal reset signal **SWRST** that generates **nSRST** and resets the whole system (see *Core module control register* on page 4-17).

5.7 Interrupt control

Figure 5-11 on page 5-23 shows the interrupt control architecture for the Integrator/AP system.

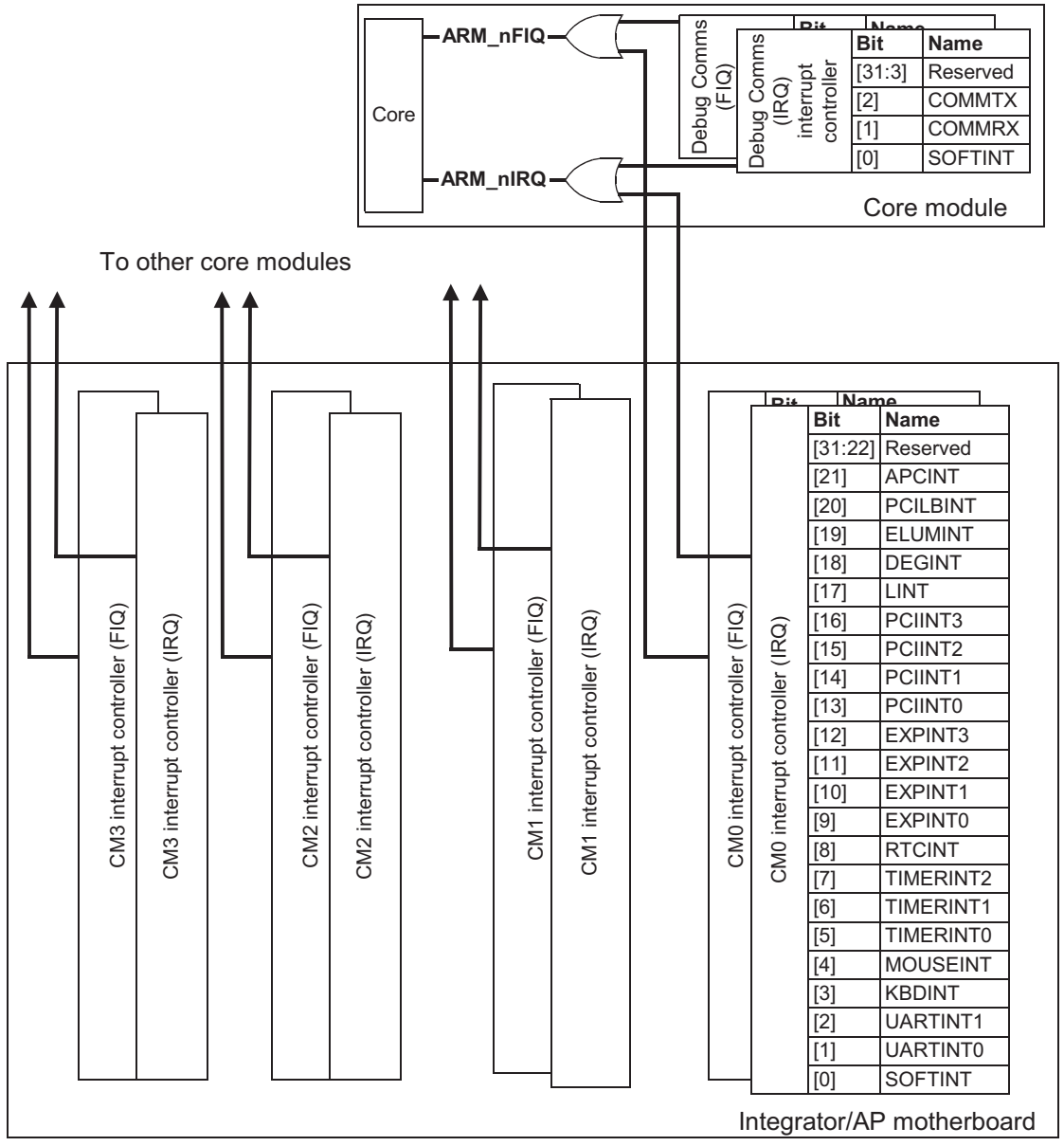


Figure 5-11 Interrupt architecture (AP image)

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

Table 5-5 Core module interrupts

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

The interrupt signals are routed to the core module using pins on the HDRB connectors (see *HDRB* on page A-4).

The interrupts and fast interrupts are enabled and handled using interrupt control registers on the motherboard (see the user guide for your motherboard). The I and F bits in the ARM core CPSR must also be cleared to enable interrupts.

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

The debug comms interrupts registers are described in *Core module interrupt registers* on page 4-28.

Chapter 6
Using Core Modules with an Integrator/CP

This chapter contains the instructions for using a CM926EJ-S, CM1026EJ-S, or CM1136JF-S core module with an Integrator/CP baseboard. It contains the following sections:

6.1 About the system architecture

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

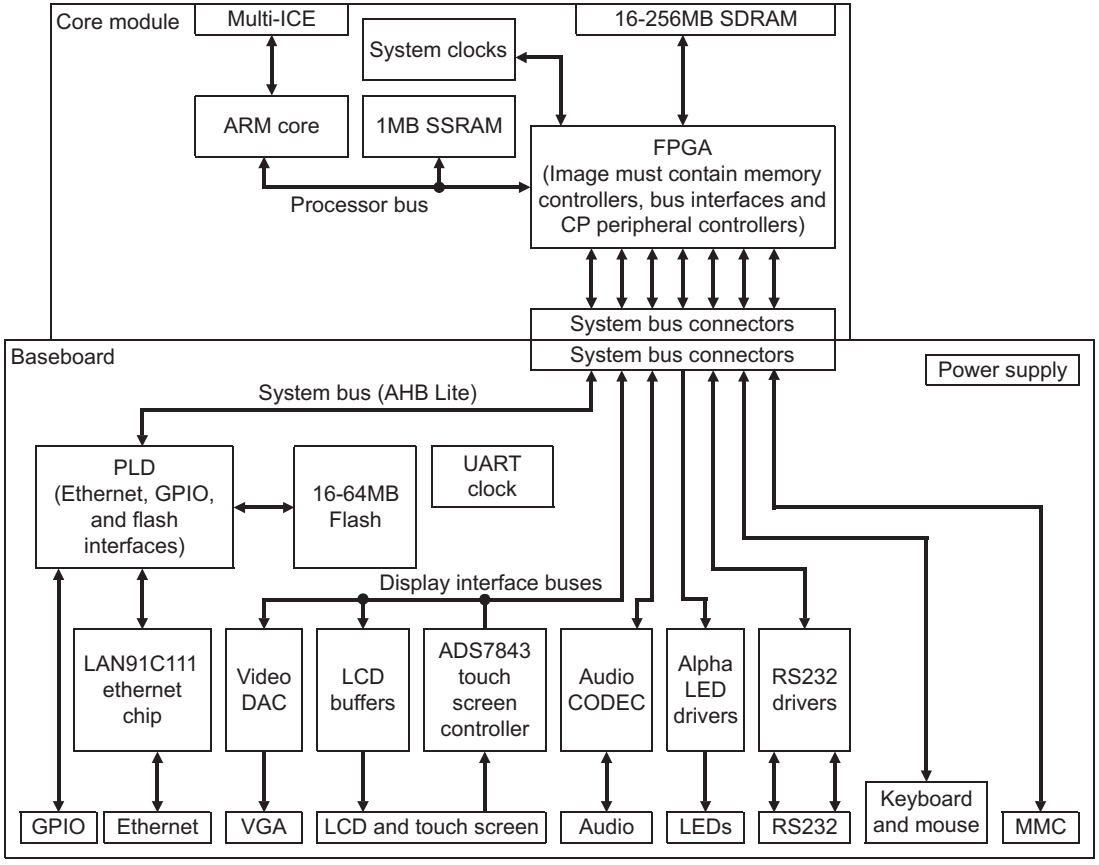


Figure 6-1 Integrator/CP system architecture

6.1.1 Integrator/CP system buses

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

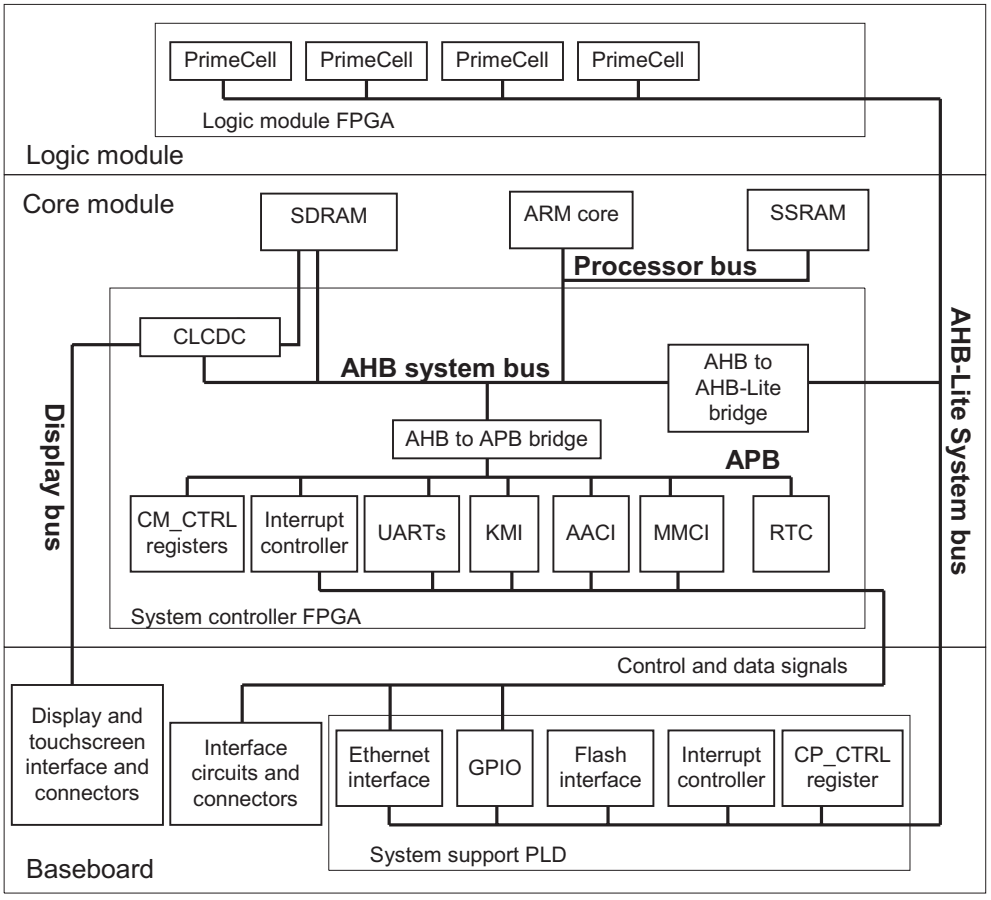


Figure 6-2 Bus routing between baseboard, core module, and logic module

System bus routing and bus interfaces

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

AHB system bus

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

AHB-Lite system bus

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

Peripheral bus

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

APB peripheral bus

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

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

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

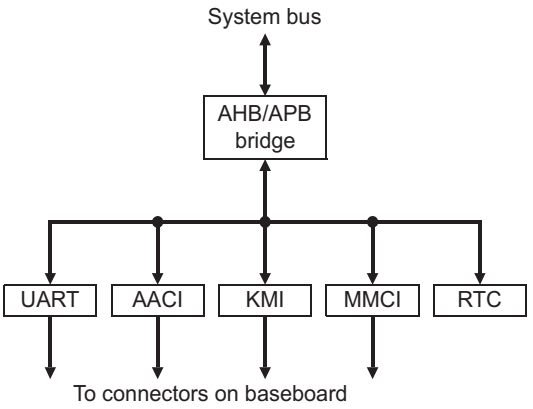


Figure 6-3 APB peripherals

System bus control

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

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

6.1.2 Configuring little or big-endian operation

The Integrator/CP can be configured to operate as a big-endian or little-endian system. To change to big-endian operation, write to the appropriate register in CP15 on the ARM core.

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

Caution

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

6.2 Module ID selection and interrupt routing

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

The position of a logic module in the HDRA/HDRB stack is used to determine its ID. The address in the alias memory region and the interrupts that it responds to are based on this ID.

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

The signals **nPPRES[3:0]** (core module present) and **nEPRES[3:0]** (logic module present) are used to signal the presence of modules to the central decoder. The signals **ID[3:0]** indicate to the module its position in the stack and the address range that its own decoder must respond to. These signals rotate as they pass up the stack. Unless the module claims additional address ranges, only one signal in each group is pulled LOW for each module.

6.3 Top level memory map

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

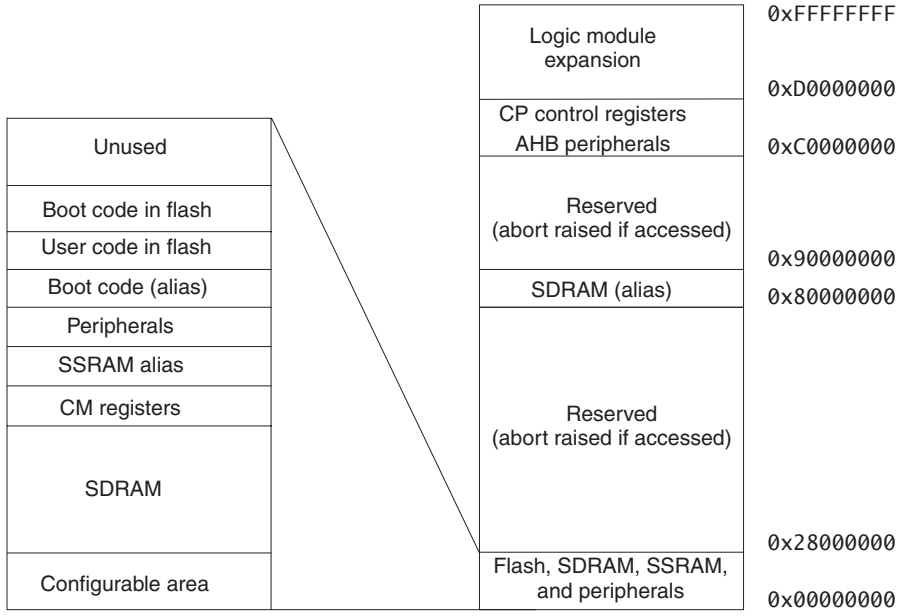


Figure 6-4 Top-level memory map

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

Note

The processor cores on different boards might have different amounts of cache and TCM. The size of the cache and TCM on a board can be identified by reading registers in CP15.

6.3.1 Physical location of memory chips

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

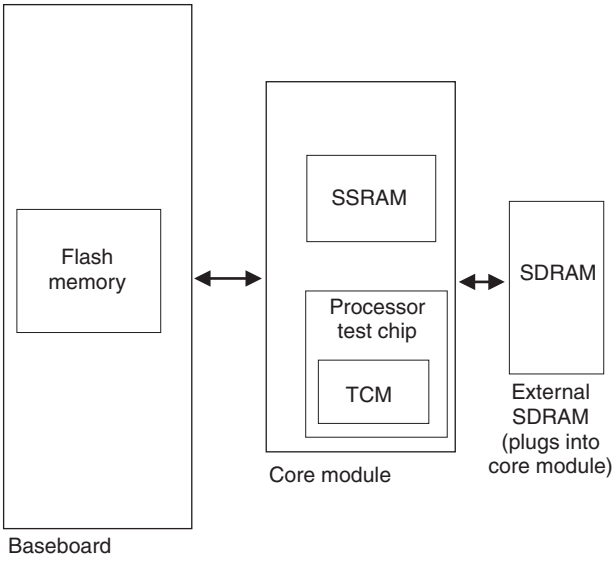


Figure 6-5 Top-level memory map

6.3.2 Configurable area of memory map

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

- The flash memory is located at `0x24000000`. The memory area at `0x0` can be configured to be either RAM or flash (REMAP =0). Placing flash at `0x0` enables you to use boot code in the flash at startup, and placing RAM at `0x0` provides a mechanism for loading custom interrupt vectors and vector routines.
- The ARM9x6 family, the ARM1026EJ-S, and the ARM1136JF-S can have *Tightly-Coupled Memory* (TCM) located inside the chip. This provides very fast access for frequently used code or data, but the accesses are not visible on the address and data buses. For the ARM946E-S and ARM966E-S, the TCM can be switched off and external SSRAM used to simulate TCM (though with slower access times).
- Some applications require large amounts of memory, or require large amounts during the development phase. The core modules can be extended with SDRAM when the core module RAM is not sufficient. A SDRAM DIMM is supplied with the core module.

Example memory map

The ARM926EJ-S, ARM1026EJ-S, and ARM1136JF-S cores have the following characteristics:

- The TCM location is configurable. Do not map the TCM to an area of memory that is used by control registers or memory-mapped IO. Also, the TCM I-RAM and TCM D-RAM regions must not overlap.
- The MMU can map physical memory to a different logical address.
- Only SSRAM mode 0 is available.

Table 6-1 shows a typical memory map with TCM enabled.

Table 6-1 Example of 32KB TCM and SSRAM mode 0

Address range	Size	Description
0x00000000–0x00007FFF	32KB	SSRAM (REMAP=0) or flash (REMAP=1).
0x00008000–0x0000FFFF	32KB	TCM I-RAM (address can be assigned to any 32KB boundary) If TCM is disabled, this space is filled by SSRAM or flash.
0x00010000–0x00017FFF	32KB	SSRAM (REMAP=0) or flash (REMAP=1).
0x00018000–0x0001FFFF	32KB	TCM D-RAM (address can be assigned to any 32KB boundary) If TCM is disabled, this space is filled by SSRAM or flash.
0x00020000–0x000FFFFF	896KB	SSRAM (REMAP=0) or flash (REMAP=1).
0x00100000–0x0FFFFFFF	256MB	SDRAM (repeats physical memory to fill space) Part of the first SDRAM image is masked by the SSRAM image.
0x10000000–0x107FFFFF	8MB	Core module registers.
0x10800000–0x10FFFFFF	8MB	On-board SSRAM alias (repeats physical memory to fill space).
0x11000000–0x1FFFFFFF	256MB	Peripherals
0x20000000–0x23FFFFFF	64MB	Boot code alias
0x24000000–0x24FFFFFF	16MB	Baseboard flash memory (contains user code and boot code).
0x25000000–0x27FFFFFF	48MB	Unused (larger sized baseboard flash devices occupy this area).
0x28000000–0x7FFFFFFF	2.3GB	Reserved (abort if accessed).
0x80000000–0x8FFFFFFF	256MB	SDRAM (alias).

Table 6-1 Example of 32KB TCM and SSRAM mode 0 (continued)

Address range	Size	Description
0x90000000–0xBFFFFFFF	2.3GB	Reserved (abort if accessed).
0xC0000000–0xCFFFFFFF	256MB	CP control registers and APB peripherals.
0xD0000000–0xFFFFFFFF	768MB	Logic module address space.

Examples of ARM946E-S memory maps

The ARM946E-S core has the following characteristics:

- If enabled, the TCM I-RAM location is always located at 0x00000000.
- The TCM D-RAM location is configurable. Do not map the TCM D-RAM to an area of memory that is used by control registers or memory-mapped IO. Also, the TCM I-RAM and TCM D-RAM regions must not overlap.
- The MPU can protect up to eight regions of physical memory.
- SSRAM modes 0 and 1 are available. Mode 1 allows SSRAM to emulate TCM.

Table 6-2 to Table 6-4 on page 6-12 show typical memory maps.

Table 6-2 Example of TCM disabled and SSRAM mode 0

Address range	Size	Description
0x00000000–0x000FFFFF	1MB	(REMAP=1) SSRAM or (REMAP=0) flash.
0x00100000–0x0FFFFFFF	256MB	SDRAM (repeats physical memory to fill space) Part of first SDRAM image is masked by SSRAM or flash.
0x10000000–0x107FFFFF	8MB	Core module registers.
0x10800000–0x10FFFFFF	8MB	On-board SSRAM alias (repeats physical memory to fill space).
0x11000000–0x1FFFFFFF	256MB	Peripherals
0x20000000–0x23FFFFFF	64MB	Boot code alias
0x24000000–0x24FFFFFF	16MB	Baseboard flash memory (contains user code and boot code).
0x25000000–0x27FFFFFF	48MB	Unused (larger sized baseboard flash devices occupy this area).
0x28000000–0x7FFFFFFF	2.3GB	Reserved (abort if accessed).
0x80000000–0x8FFFFFFF	256MB	SDRAM (alias).

Table 6-2 Example of TCM disabled and SSRAM mode 0 (continued)

Address range	Size	Description
0x90000000–0xBFFFFFFF	2.3GB	Reserved (abort if accessed).
0xC0000000–0xCFFFFFFF	256MB	CP control registers and APB peripherals.
0xD0000000–0xFFFFFFFF	768MB	Logic module address space.

Table 6-3 Example of TCM disabled and SSRAM mode 1

Address range	Size	Description
0x00000000–0x03FFFFFF	512KB	(REMAP=1) SSRAM Block I (SSRAM emulating I-RAM) or (REMAP=0) flash.
0x04000000–0x07FFFFFF	512KB	(REMAP=1) SSRAM Block D (SSRAM emulating D-RAM) or (REMAP=0) flash.
0x08000000–0x0FFFFFFF	256MB	SDRAM (repeats physical memory to fill space) Part of first SDRAM image is masked by SSRAM or flash.
0x10000000–0x107FFFFF	8MB	Core module registers.
0x10800000–0x10FFFFFF	8MB	On-board SSRAM alias (repeats physical memory to fill space).
0x11000000–0x1FFFFFFF	256MB	Peripherals
0x20000000–0x23FFFFFF	64MB	Boot code alias
0x24000000–0x24FFFFFF	16MB	Baseboard flash memory (contains user code and boot code).
0x25000000–0x27FFFFFF	48MB	Unused (larger sized baseboard flash devices occupy this area).
0x28000000–0x7FFFFFFF	2.3GB	Reserved (abort if accessed).
0x80000000–0x8FFFFFFF	256MB	SDRAM (alias).
0x90000000–0xBFFFFFFF	2.3GB	Reserved (abort if accessed).
0xC0000000–0xCFFFFFFF	256MB	CP control registers and APB peripherals.
0xD0000000–0xFFFFFFFF	768MB	Logic module address space.

Table 6-4 Example of 32KB TCM enabled and SSRAM mode 0

Address range	Size	Description
0x00000000–0x0001FFFF	2 x 32KB	TCM I-RAM (physical memory repeatedly aliased to fill space).
0x00020000–0x000FFFFF	840KB	SSRAM (REMAP=0) or flash (REMAP=1).
0x00100000–0x0011FFFF	2x 32KB	TCM D-RAM (address can be assigned to any TCM-sized boundary, physical memory repeatedly aliased to fill space) If TCM is disabled, this space is filled by SDRAM.
0x00100000–0x0FFFFFFF	256MB	SDRAM (repeats physical memory to fill space) Part of first SDRAM image is masked by TCM.
0x10000000–0x107FFFFF	8MB	Core module registers.
0x10800000–0x10FFFFFF	8MB	On-board SSRAM alias (repeats physical memory to fill space).
0x11000000–0x1FFFFFFF	256MB	Peripherals
0x20000000–0x23FFFFFF	64MB	Boot code alias
0x24000000–0x24FFFFFF	16MB	Baseboard flash memory (contains user code and boot code).
0x25000000–0x27FFFFFF	48MB	Unused (larger sized baseboard flash devices occupy this area).
0x28000000–0x7FFFFFFF	2.3GB	Reserved (abort if accessed).
0x80000000–0x8FFFFFFF	256MB	SDRAM (alias).
0x90000000–0xBFFFFFFF	2.3GB	Reserved (abort if accessed).
0xC0000000–0xCFFFFFFF	256MB	CP control registers and APB peripherals.
0xD0000000–0xFFFFFFFF	768MB	Logic module address space.

Examples of ARM966E-S memory maps

The ARM966E-S core has the following characteristics:

- The TCM I-RAM and TCM D-RAM are of fixed size and can be separately enabled or disabled.
- If enabled, the TCM I-RAM location is always located at `0x00000000`.
- If enabled, the TCM D-RAM location is always located at `0x04000000`.

- The ARM966E-S has 16 fixed attribute regions within its memory map that are predefined alternately as buffered and unbuffered.
- SSRAM modes 0 and 1 are available. Mode 1 allows SSRAM to emulate TCM.

Table 6-5 to Table 6-7 on page 6-14 show typical memory maps.

Table 6-5 Example of TCM disabled and SSRAM mode 0

Address range	Size	Description
0x00000000–0x000FFFFF	1MB	(REMAP=1) SSRAM or (REMAP=0) flash.
0x00100000–0x0FFFFFFF	256MB	SDRAM (repeats physical memory to fill space) Part of first SDRAM image is masked by SSRAM or flash.
0x10000000–0x107FFFFF	8MB	Core module registers.
0x10800000–0x10FFFFFF	8MB	On-board SSRAM alias (repeats physical memory to fill space).
0x11000000–0x1FFFFFFF	256MB	Peripherals
0x20000000–0x23FFFFFF	64MB	Boot code alias
0x24000000–0x24FFFFFF	16MB	Baseboard flash memory (contains user code and boot code).
0x25000000–0x27FFFFFF	48MB	Unused (larger sized baseboard flash devices occupy this area).
0x28000000–0x7FFFFFFF	2.3GB	Reserved (abort if accessed).
0x80000000–0x8FFFFFFF	256MB	SDRAM (alias).
0x90000000–0xBFFFFFFF	2.3GB	Reserved (abort if accessed).
0xC0000000–0xCFFFFFFF	256MB	CP control registers and APB peripherals.
0xD0000000–0xFFFFFFFF	768MB	Logic module address space.

Table 6-6 Example of TCM disabled and SSRAM mode 1

Address range	Size	Description
0x00000000–0x03FFFFFF	128 x 512KB	(REMAP=1) SSRAM Block I (SSRAM emulating I-RAM) or (REMAP=0) flash (repeatedly aliased to fill space).
0x04000000–0x07FFFFFF	128 x 512KB	(REMAP=1) SSRAM Block D (SSRAM emulating D-RAM) or (REMAP=0) flash (repeatedly aliased to fill space).
0x08000000–0x0FFFFFFF	256MB	SDRAM (repeatedly aliased to fill space).
0x10000000–0x107FFFFF	8MB	Core module registers.

Table 6-6 Example of TCM disabled and SSRAM mode 1 (continued)

Address range	Size	Description
0x10800000–0x10FFFFFF	8MB	On-board SSRAM alias (repeats physical memory to fill space).
0x11000000–0x1FFFFFFF	256MB	Peripherals
0x20000000–0x23FFFFFF	64MB	Boot code alias
0x24000000–0x24FFFFFF	16MB	Baseboard flash memory (contains user code and boot code).
0x25000000–0x27FFFFFF	48MB	Unused (larger sized baseboard flash devices occupy this area).
0x28000000–0x7FFFFFFF	2.3GB	Reserved (abort if accessed).
0x80000000–0x8FFFFFFF	256MB	SDRAM (alias).
0x90000000–0xBFFFFFFF	2.3GB	Reserved (abort if accessed).
0xC0000000–0xCFFFFFFF	256MB	CP control registers and APB peripherals.
0xD0000000–0xFFFFFFFF	768MB	Logic module address space.

Table 6-7 Example of 128KB TCM enabled and SSRAM mode 0

Address range	Size	Description
0x00000000–0x03FFFFFF	512 x 128KB	TCM I-RAM (physical memory repeatedly aliased to fill space).
0x04000000–0x07FFFFFF	512 x 128KB	TCM D-RAM (physical memory repeatedly aliased to fill space).
0x08000000–0x0FFFFFFF	256MB	SDRAM (alias).
0x10000000–0x107FFFFF	8MB	Core module registers.
0x10800000–0x10FFFFFF	8MB	On-board SSRAM alias (repeats physical memory to fill space).
0x11000000–0x1FFFFFFF	256MB	Peripherals
0x20000000–0x23FFFFFF	64MB	Boot code alias
0x24000000–0x24FFFFFF	16MB	Baseboard flash memory (contains user code and boot code).
0x25000000–0x27FFFFFF	48MB	Unused (larger sized baseboard flash devices occupy this area).
0x28000000–0x7FFFFFFF	2.3GB	Reserved (abort if accessed).

Table 6-7 Example of 128KB TCM enabled and SSRAM mode 0 (continued)

Address range	Size	Description
0x80000000–0x8FFFFFFF	256MB	SDRAM (alias).
0x90000000–0xBFFFFFFF	2.3GB	Reserved (abort if accessed).
0xC0000000–0xCFFFFFFF	256MB	CP control registers and APB peripherals.
0xD0000000–0xFFFFFFFF	768MB	Logic module address space.

6.3.3 Baseboard flash memory

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

6.3.4 Memory timing

Accesses to memory require a different number of cycles depending on the type of memory as shown in Table 6-8.

Table 6-8 Wait states for memory access

Memory Type	CP9x6E(J)-S
SSRAM read or write	0
SDRAM read typical (no bus conflict)	0–5
SDRAM read maximum (write in progress before read)	31
Flash read	6

6.4 Programmable logic

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

- *Baseboard system support PLD*
- *System controller FPGA*.

6.4.1 Baseboard system support PLD

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

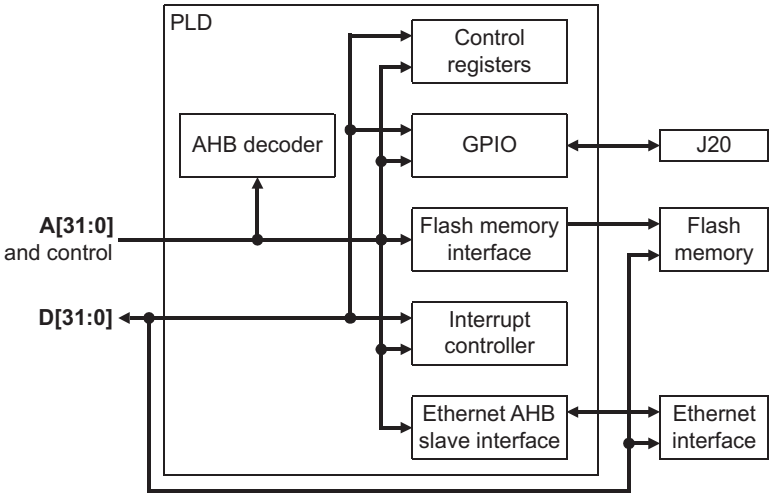


Figure 6-6 Baseboard PLD

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

6.4.2 System controller FPGA

The system controller FPGA for an Integrator/CP system is located on the core module. It contains the main bus bridges, memory controllers, and peripheral controllers. Figure 6-7 on page 6-17 shows a simplified block diagram of a system controller FPGA.

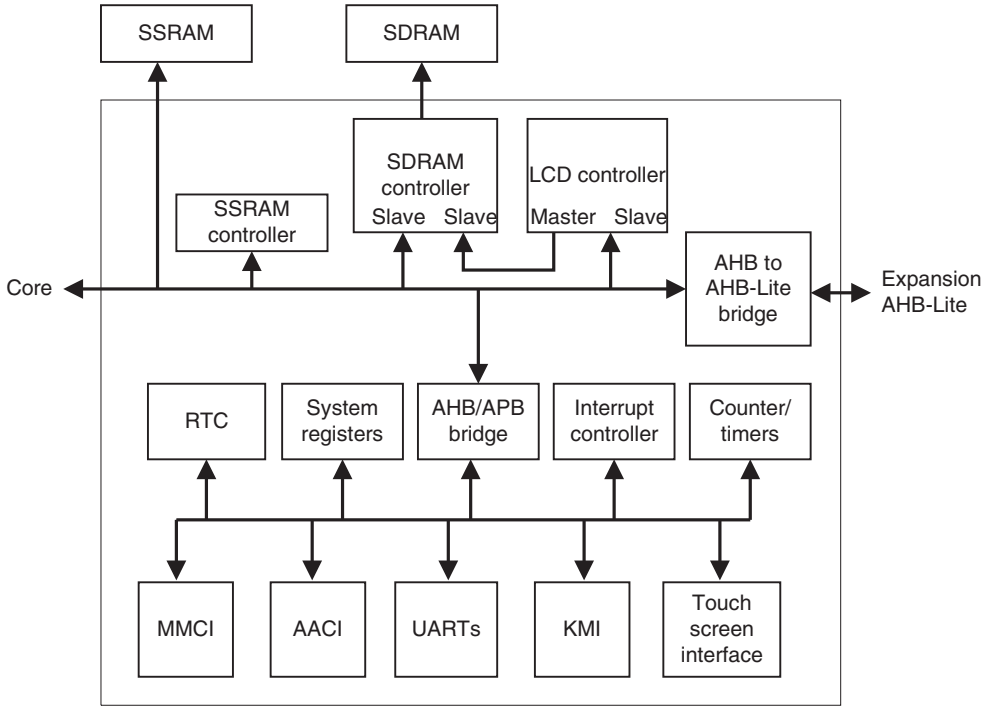


Figure 6-7 System controller FPGA block diagram

Note

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

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

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

The configuration flash can store up to four images. The images are selected by the **CFGSEL[1:0]** signals from the baseboard as described in *Core module FPGA* on page 3-6.

6.4.3 HDL files

The core module image provides the CP functional blocks as a set of HDL files. These are described in Table 6-9.

Table 6-9 CM image functional block HDL file descriptions

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

6.5 Register and memory overview

Table 6-10 shows a map for a typical Integrator/CP system. See the *Integrator/CP User Guide* for more details on register function.

Note

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

Table 6-10 System control register map

Peripheral or device	Address range	Size
Boot flash. Mapped at this address only at power ON, and then disabled to allow access to SSRAM. **Note** The memory map below `0x10000000` depends on the settings for SSRAM mode, TCM enabled, and REMAP.	`0x00000000–0x000FFFFF`	1MB
SDRAM.	`0x00100000–0x0FFFFFFF`	255MB
Core Module control registers.	`0x10000000-0x1000003F`	64 bytes
Core Module interrupt controller.	`0x10000040–0x1000007F`	64 bytes
Reserved **Note** The CM926, CM1026, and CM1136 use `0x10000080–0x10000008C` as voltage control registers. See *Voltage control registers* on page 4-32.	`0x10000080–0x100000FF`	128 bytes
Serial Presence Detect memory.	`0x10000100–0x100001FF`	256 bytes
Counter/timers.	`0x13000000–0x13FFFFFF`	16MB
Primary interrupt controller registers.	`0x14000000–0x14FFFFFF`	16MB
Real time clock.	`0x15000000–0x15FFFFFF`	16MB
UART0.	`0x16000000–0x16FFFFFF`	16MB

Table 6-10 System control register map (continued)

Peripheral or device	Address range	Size
UART1.	0x17000000–0x17FFFFFF	16MB
Keyboard.	0x18000000–0x18FFFFFF	16MB
Mouse.	0x19000000–0x19FFFFFF	16MB
Debug LEDs and DIP switch.	0x1A000000–0x1AFFFFFF	16MB
Reserved.	0x1B000000–0x1BFFFFFF	16MB
Multimedia Card Interface.	0x1C000000–0x1CFFFFFF	16MB
Advanced Audio CODEC Interface.	0x1D000000–0x1DFFFFFF	16MB
Touch Screen Controller Interface.	0x1E000000–0x1EFFFFFF	16MB
Reserved.	0x28000000–0xBFFFFFFF	2416MB
CLCD regs/palette.	0xC0000000–0xC0FFFFFF	16MB
Reserved.	0xC1000000–0xC7FFFFFF	112MB
Ethernet.	0xC8000000–0xC8FFFFFF	16MB
GPIO.	0xC9000000–0xC9FFFFFF	16MB
Secondary interrupt controller.	0xCA000000–0xCAFFFFFF	16MB
CP control registers.	0xCB000000–0xCBFFFFFF	16MB
Reserved.	0xCC000000–0xCFFFFFFF	64MB
Logic modules.	0xD0000000–0xFFFFFFFF	768MB

Note

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

6.5.1 CM control register for Integrator/CP

This section describes the CM_CTRL register that is modified in the FPGA image used for the Integrator/CP. The other CM registers are unchanged (see *Core module control registers* on page 4-13).

Core module control, CM_CTRL

The core module and LCD control register (CM_CTRL) at `0x1000000C` is a read/write register that controls a number of user-configurable features of the core module and the display interface on the baseboard.

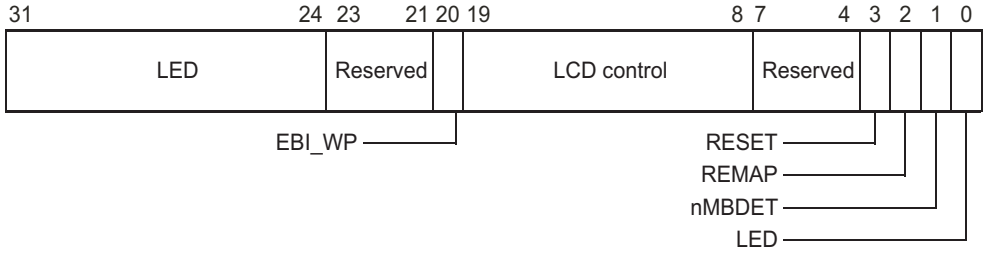


Figure 6-8 CM_CTRL

Table 6-11 and Figure 6-8 describe the core module control register bits

Table 6-11 CM_CTRL register

Bits	Name	Access	Function
[31:24]	USR_LEDs	Write	Writing a 1 to a bit in this register illuminates the associated LED.
[23:21]	Reserved	Use read-modify-write to preserve value.	
[20]	EBI_WP	Write	This bit controls the write power signal to the core module user flash in EBI[1] and EBI[2]. 0 = flash write protected 1= flash can be written to.
[19]	n24BITEN	Write	Select VGA depth: 0 = 24bit VGA 1 = 18bit VGA Must be 1 to enable BIAS control.
[18]	STATIC	Write	No connection on Sharp panel.
[17]	STATIC2	Write	Up/down axis flip on Sharp panel.
[16]	STATIC1	Write	Right/left axis flip on Sharp panel.
[15]	Enable LCD1	Write	Enable, active high.
[14]	Enable LCD0	Write	Enable, active high

Table 6-11 CM_CTRL register (continued)

Bits	Name	Access	Function
[13:11]	LCDMUXSEL	Write	001 = Generic LCD connector, 24-bit mode 011 = Sharp LCD panel 100 = Sharp LCD panel 111 = 24-bit VGA
[10]	LCDBIASDN	Write	Low to high transition decreases LCD bias voltage (dimmer)
[9]	LCDBIASUP	Write	Low to high transition increases LCD bias voltage (brighter)
[8]	LCDBIASEN	Read/write	Enable LCD bias supply
[7:4]	Reserved	Use read-modify-write to preserve value.	
[3]	RESET	Write	This is used to reset the core module, the baseboard on which it is mounted, and any modules in a stack. A reset is triggered by writing a 1. Reading this bit always returns a 0 allowing you to use read-modify-write operations without masking the RESET bit.
[2]	REMAP	Read/write	This bit is used to control REMAP: 0 = flash at address 0 1= SRAM at address 0.
[1]	nMBDET	Read	This bit indicates whether or not the core module is mounted on a baseboard: 0 = mounted on baseboard 1 = reserved.
[0]	LED	Read/write	This bit controls the green MISC LED on the core module: 0 = LED OFF 1 = LED ON.

6.6 Peripherals and interfaces

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

- *Clock control*
- *Counter/timers*
- *Real-time clock*
- *UARTs* on page 6-24
- *Keyboard and mouse interface* on page 6-25
- *MMC interface* on page 6-25
- *Audio interface* on page 6-25
- *Touchscreen controller interface* on page 6-25
- *Display interface* on page 6-26
- *Ethernet interface* on page 6-26.

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

6.6.1 Clock control

The baseboard Micrologic IC525 clock generator provides:

- the 14.7456MHz UART clock.
- the 25MHz Ethernet clock.

A baseboard crystal oscillator provides a 12.288MHz AACI bit clock.

6.6.2 Counter/timers

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

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

Periodic
: The counter counts down to zero and then reloads the period value.

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

6.6.3 Real-time clock

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

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

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

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

6.6.4 UARTs

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

The UARTs are functionally similar to standard 16C550 devices.

For detailed information about the UART, see the *UART (PL011) Technical Reference Manual*.

6.6.5 Keyboard and mouse interface

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

6.6.6 MMC interface

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

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

6.6.7 Audio interface

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

Note

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

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

6.6.8 Touchscreen controller interface

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

6.6.9 Display interface

The touchscreen controller that accompanies the LCD is described in *Touchscreen controller interface* on page 6-25. See also the *ARM PrimeCell Color LCD Controller (PL110) Technical Reference Manual*.

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

CM_CTRL and LCD control bits

In addition to the PrimeCell LCD registers, there are some LCD control bits in the CM_CTRL register at `0x1000000C`. For more information, see *CM control register for Integrator/CP* on page 6-20.

6.6.10 Ethernet interface

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

6.7 Reset controller

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

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

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

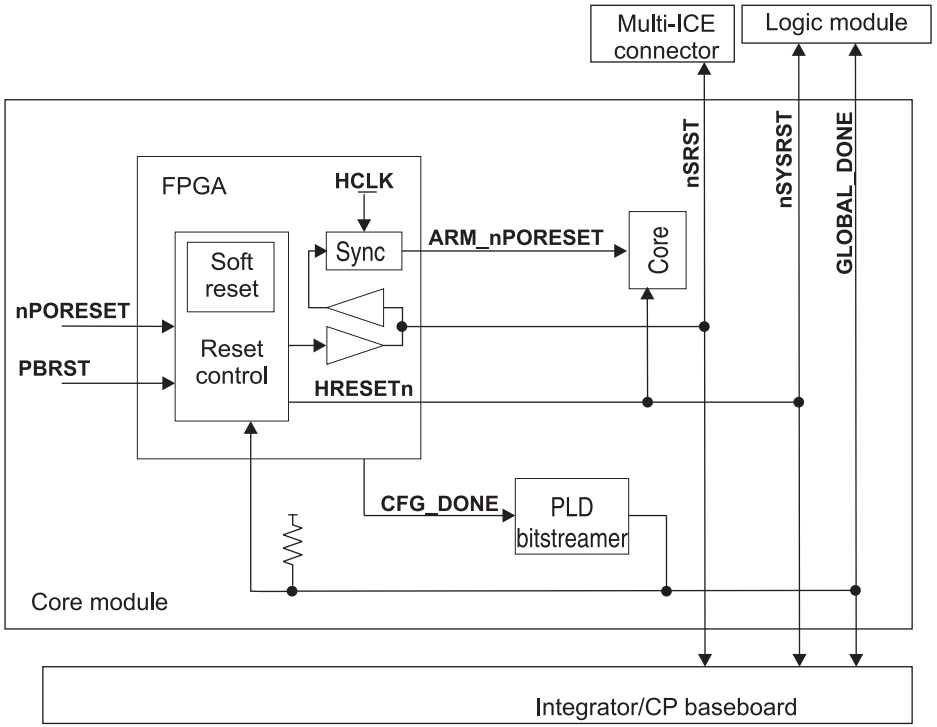


Figure 6-9 Core module reset controller

6.7.1 Reset control signals

Table 6-12 describes the external reset signals.

Table 6-12 Reset signal descriptions

Name	Description	Type	Function
ARM_nPORESET	Processor reset	Output	The **ARM_nPORESET** signal is used to reset the processor core. It is generated from **nSRST** LOW when the core module is used standalone, or **nSYSRST** LOW when the core module is attached to a motherboard.
PBRST	Push-button reset	Input	The **PBRST** signal is generated by pressing the reset button.
nSRST	System reset	Bidirectional	The **nSRST** open collector output signal is driven LOW by the core module FPGA when the signal **PBRST** or software reset (**SWRST**) is asserted. As an input, **nSRST** can be driven LOW by Multi-ICE. If there is no motherboard present, the **nSRST** signal is synchronized to the processor bus clock to generate the **ARM_nPORESET** signal.
nSYSRST	System reset	Output	The **nSYSRST** signal is generated by the system controller FPGA on the core module. It is used to generate **HRESETn** to the test chip and the system.

6.7.2 Software resets

The core module FPGA provides a software reset that can be triggered by writing to the reset bit in the CM_CTRL register. This generates the internal reset signal **SWRST** that generates **nSRST** and resets the whole system (see *Core module control register* on page 4-17).

6.8 Interrupt control

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

- *Interrupt controllers* on page 6-30
- *Interrupt routing between Integrator modules* on page 6-30
- *CP image interrupt control registers* on page 6-33
- *Handling interrupts* on page 6-35.

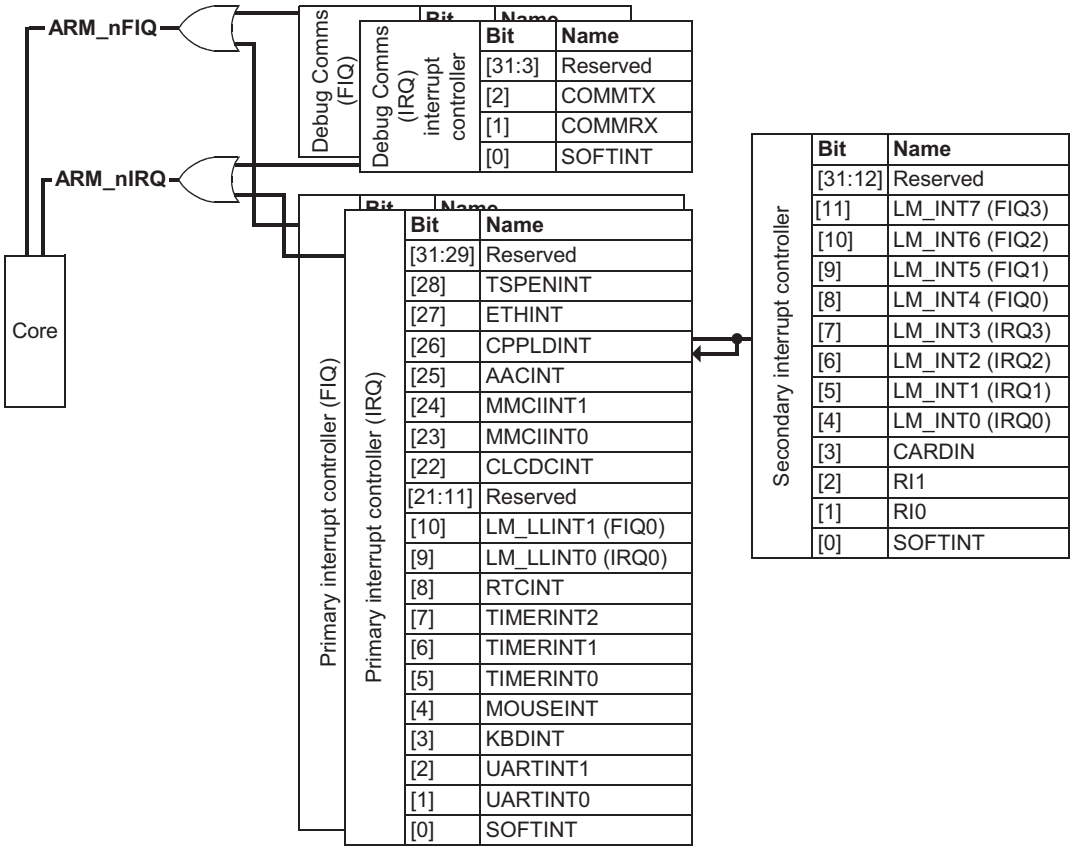


Figure 6-10 Interrupt architecture (CP image)

6.8.1 Interrupt controllers

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

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

Primary interrupt controller

The PIC is implemented within the core module FPGA and handles the majority of interrupts from the system. A substantial number of interrupts are reserved to maintain compatibility with other modules within the Integrator family. The PIC provides a set of registers to control and handle interrupts. These are described in *Primary interrupt controller registers* on page 6-33.

Secondary interrupt controller

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

The MMC interrupt on the SIC is generated by the card insertion detect switch and is different from the MMCI interrupts in the PIC generated by the MMCI PrimeCell.

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

Debug communications interrupts

The processor core incorporates EmbeddedICE hardware. This provides debug communications data read and write registers that are used to pass data between the processor and JTAG equipment. For a description of the debug communications channel, see the Technical Reference Manual for your core.

6.8.2 Interrupt routing between Integrator modules

Figure 6-11 on page 6-31 shows how the IRQ signals are routed from modules to the interrupt controllers.

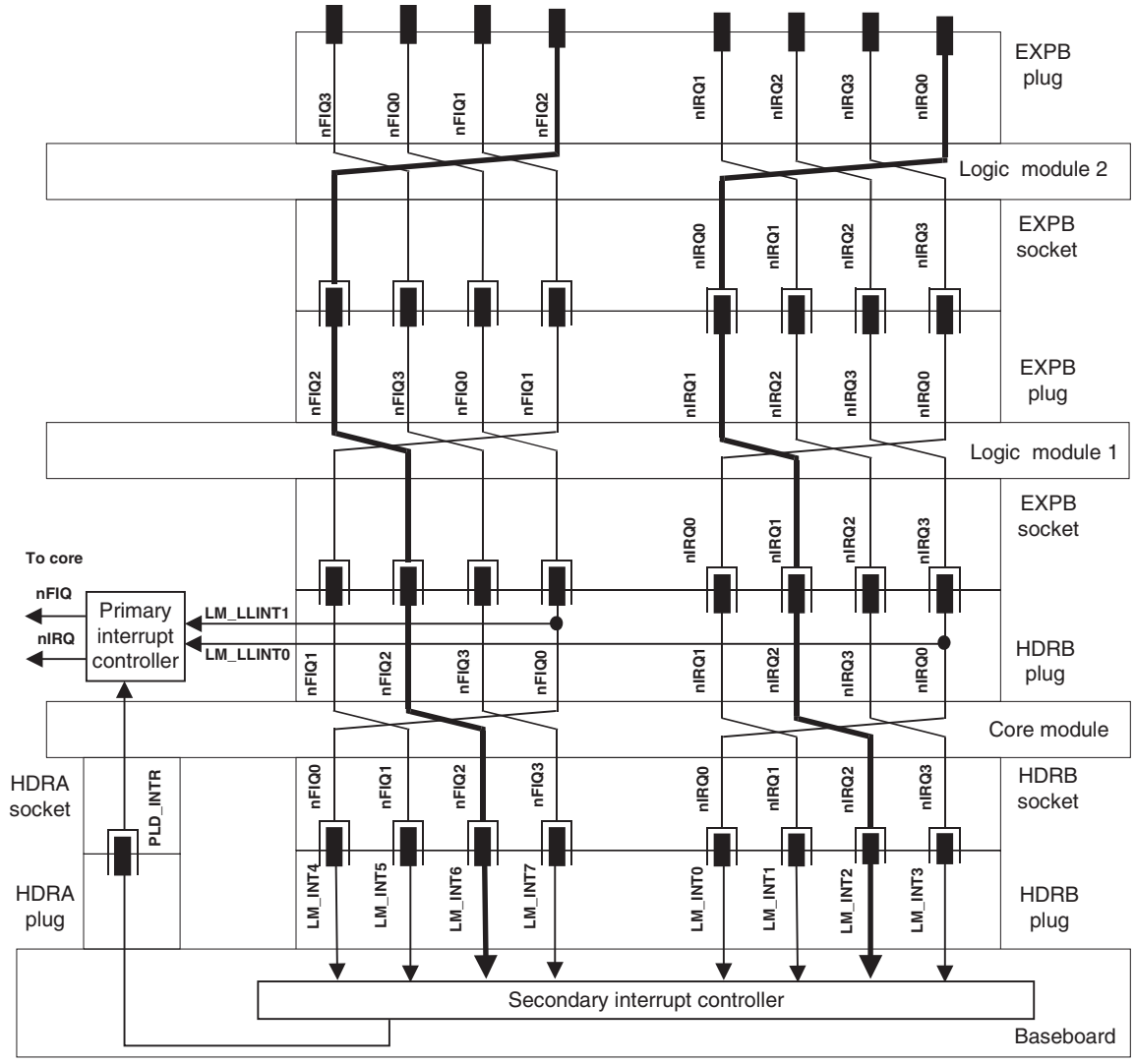


Figure 6-11 Interrupt signal routing

The diagram highlights how the signals are routed so that, for example, the interrupt request from logic module 2 connects to **LM_INT2** (**IRQ[2]**) on the baseboard. A similar routing applies for logic module 1 and 3. The operation of the interrupts relies on the core module being mounted on the baseboard first with any logic modules on top.

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

Note

Ensure that the correct interrupt line is driven. See the routing pattern in Figure 6-11 on page 6-31.

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

6.8.3 CP image interrupt control registers

The baseboard FPGA provides an interrupt controller to handle IRQs and FIQs from around the system. These are in addition to the CM interrupt controller (see *Core module interrupt registers* on page 4-28), and are described in:

- *Primary interrupt controller registers*
- *Debug communications interrupt registers* on page 6-34
- *Secondary interrupt controller registers* on page 6-35
- *Soft interrupt set and soft interrupt clear registers* on page 6-35.

Primary interrupt controller registers

The CP image for the FPGA provides interrupt controllers for both IRQs and FIQs that maintain compatibility with other Integrator modules to ensure code portability. The primary interrupt control registers are listed in Table 6-13

Table 6-13 Primary interrupt register addresses

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

The bit assignment for interrupts in the status, raw status, and enable registers for the IRQ and FIQ interrupt controllers is similar and is shown in Table 6-14.

Table 6-14 Primary interrupt register bit assignments

Bit	Name	Function
[31:29]	-	Reserved
[28]	TS_PENINT	Touchscreen pen-down event interrupt
[27]	ETH_INT	Ethernet interface interrupt
[26]	CPPLDINT	Interrupt from secondary interrupt controller, see *Secondary interrupt controller registers* on page 6-35.
[25]	AACIINT	Audio interface interrupt
[24]	MMCIINT1	MultiMedia card interface
[23]	MMCIINT0	MultiMedia card interface
[22]	CLCDCINT	Display controller interrupt
[21:11]	-	Reserved
[10]	LM_LLINT1	Logic module low-latency interrupt 1
[9]	LM_LLINT0	Logic module low-latency interrupt 0
[8]	RTCINT	Real time clock interrupt
[7]	TIMERINT2	Counter-timer 2 interrupt
[6]	TIMERINT1	Counter-timer 1 interrupt
[5]	TIMERINT0	Counter-timer 0 interrupt
[4]	MOUSEINT	Mouse interrupt
[3]	KBDINT	Keyboard interrupt
[2]	UARTINT1	UART 1 interrupt
[1]	UARTINT0	UART 0 interrupt
[0]	SOFTINT	Software interrupt

Debug communications interrupt registers

The debug comms interrupts registers are described in *Core module interrupt registers* on page 4-28.

Secondary interrupt controller registers

The secondary interrupt control registers are listed in Table 6-15

Table 6-15 Secondary interrupt register addresses

Address	Name	Type	Size	Function
0xCA000000	SIC_IRQ_STATUS	Read	22	SIC gated interrupt status
0xCA000004	SIC_IRQ_RAWSTAT	Read	22	SIC raw interrupt status
0xCA000008	SIC_IRQ_ENABLESET	Read/write	22	SIC enable set
0xCA00000C	SIC_IRQ_ENABLECLR	Write	22	SIC enable clear
0xCA000010	SIC_INT_SOFTSET	Read/write	16	Software interrupt set
0xCA000014	SIC_INT_SOFTCLR	Write	16	Software interrupt clear

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

Soft interrupt set and soft interrupt clear registers

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

Use the set and clear procedure described in *Soft interrupt set and soft interrupt clear registers* on page 4-30 for registers PIC_INT_SOFTSET, PIC_INT_SOFTCLR, SIC_INT_SOFTSET, and SIC_INT_SOFTCLR.

6.8.4 Handling interrupts

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

Enabling IRQ interrupts

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

1. Disable the primary interrupt by setting the appropriate bit in PIC_IRQ_ENCLR.

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

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

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

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

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

Note

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

Determining and clearing IRQ interrupts

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

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

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

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

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

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

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


Appendix A
Signal Descriptions

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

Note

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

A.1 HDRA

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

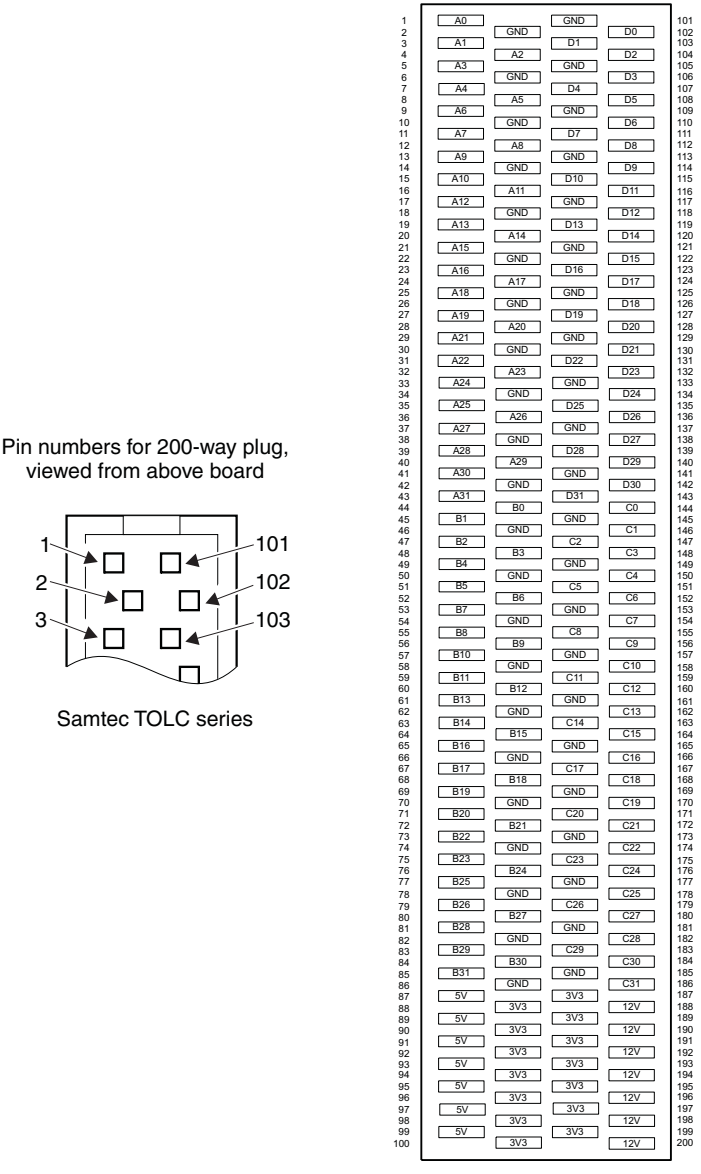


Figure A-1 HDRA plug pin numbering

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

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

Pin label	Signal	Description
A[31:0]	**HADDR[31:0]**	System address bus
B[31:0]	Not used	-
C[31:0]	**CONT[31:0]**	See below
C[31:16]	Not used	-
C15	**HMASTLOCK**	Locked transaction
C[14:13]	**HRESP[1:0]**	Slave response
C12	**HREADY**	Slave ready
C11	**HWRITE**	Write transaction
C[10:8]	**HPROT[2:0]**	Transaction protection type
C[7:5]	**HBURST[2:0]**	Transaction burst size
C4	**HPROT[3]**	Transaction protection type
C[3:2]	**HSIZE[1:0]**	Transaction width
C[1:0]	**HTRAN[1:0]**	Transaction type
D[31:0]	**HDATA[31:0]**	System data bus

A.2 HDRB

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

A.2.1 HDRB socket pinout

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

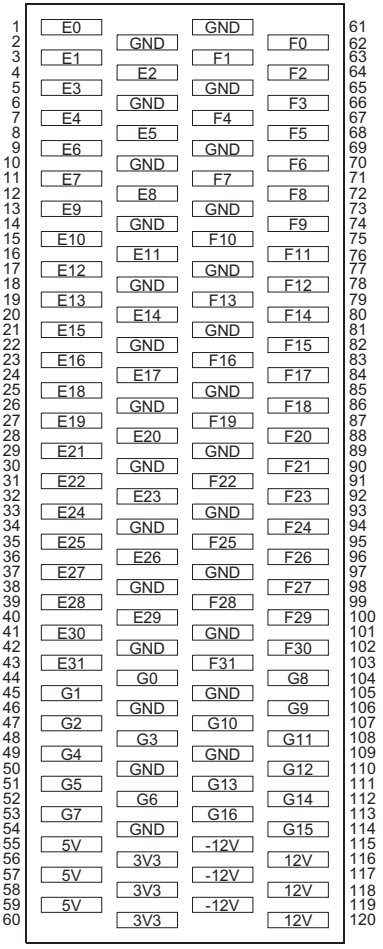
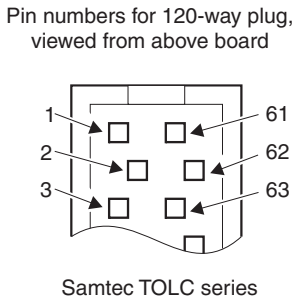


Figure A-2 HDRB socket pin numbering

A.2.2 HDRB plug pinout

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



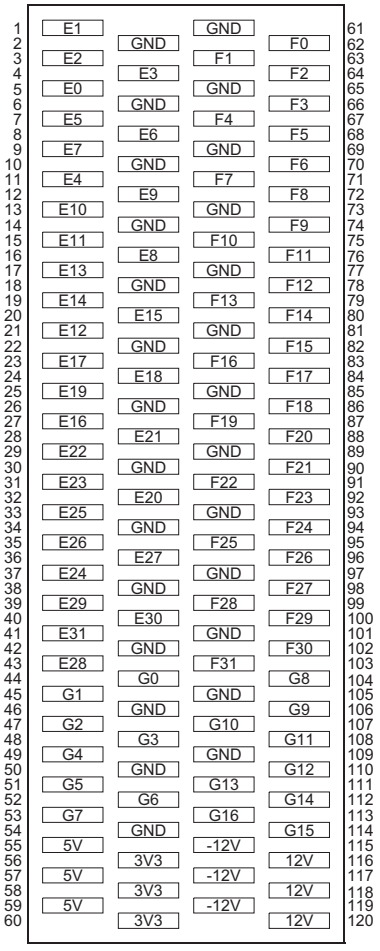


Figure A-3 HDRB plug pin numbering

A.2.3 Through-board signal connections

The signals on the pins labeled E[31:0] are cross-connected between the plug and socket so that the signals are rotated through the stack in groups of four. For example, the first block of four are connected as shown in Table A-2.

Table A-2 Signal cross-connections (example)

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

For details about the signal rotation scheme, see *Module ID selection* on page 5-4 (Integrator/AP) and *Module ID selection and interrupt routing* on page 6-6 (Integrator/CP).

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

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

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

A.2.4 HDRB signal descriptions

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

Table A-3 HDRB signal description (AHB)

Pin label	Name	Description
E[31:28]	**SYSCLK[3:0]**	System clock to each core module/expansion card.
E[27:24]	**nPPRES[3:0]**	Processor present.
E[23:20]	**nIRQ[3:0]**	Interrupt request to processors 3, 2, 1, and 0 respectively.
E[19:16]	**nFIQ[3:0]**	Fast interrupt requests to processors 3, 2, 1, and 0 respectively.
E[15:12]	**ID[3:0]**	Core module stack position indicator.
E[11:8]	**HLOCK[3:0]**	System bus lock from processor 3, 2, 1, and 0 respectively.
E[7:4]	**HGRANT[3:0]**	System bus grant to processor 3, 2, 1, and 0 respectively.
E[3:0]	**HBUSREQ[3:0]**	System bus request from processors 3, 2, 1, and 0 respectively.
F[31:0]	-	Not connected.
G16	**nRTCKEN**	RTCK AND gate enable.
G[15:14]	**CFGSEL[1:0]**	FPGA configuration select.
G13	**nCFGEN**	Sets motherboard into configuration mode.
G12	**nSRST**	Multi-ICE reset (open collector).
G11	**FPGADONE**	Indicates when FPGA configuration is complete (open collector).
G10	**RTCK**	Returned JTAG test clock.
G9	**nSYSRST**	Buffered system reset.
G8	**nTRST**	JTAG reset.
G7	**TDO**	JTAG test data out.

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

Pin label	Name	Description
G6	**TDI**	JTAG test data in.
G5	**TMS**	JTAG test mode select.
G4	**TCK**	JTAG test clock.
G[3:1]	**MASTER[2:0]**	Master ID. Binary encoding of the master currently performing a transfer on the bus. Corresponds to the module ID and to the **HBUSREQ** and **HGRANT** line numbers.
G0	**nMBDET**	Motherboard detect pin.

The **HBUSREQ** line goes active at the start of the transfer. In the AMBA AHB specification, the **BUSREQ** line remains active for the duration of the bus transaction.

Designs that respond with **HBUSGRANT** might expect the **HBUSREQ** line to clear one clock cycle later. If **HBUSREQ** followed AMBA spec, it would be possible to tie **HBUSGRANT** to the **HBUSREQ** from the core module. Because **HBUSREQ** and **HBUSGRANT** are only valid for one clock cycle, the **HTRANS1** bit is high for about 2 microseconds.

A work around is to extend **HBUSGRANT** until **HTRANS1** goes low. **HBUSREQ** is still only one clock cycle wide.

According to the AMBA specification (ARM IHI 0011A, page 3-29): "For undefined length bursts the master should continue to assert the request until it has started the last transfer. The arbiter cannot predict when to change the arbitration at the end of an undefined length burst."

The master can request the bus for a single cycle and do the transfer, but if the burst is of undefined length (**HBURST = INCR**), **HBUSREQ** should stay active until the last transfer.

The HDL for the core module FPGA has changed, see the ARM web site for new files.

A.3 Trace connector pinout

Table A-4 shows the pinout of the Trace connector for the CM946E-S and CM966E-S. This is also the first (A) of two Trace connectors for the CM926EJ-S, CM1026EJ-S, and CM1136JF-S. Where the signal names differ from the ones used on the CM946E-S and CM966E-S, the names for the CM926EJ-S, CM1026EJ-S, and CM1136JF-S are shown in parenthesis. See Figure A-4 on page A-11 for the Mictor connector pinout.

Table A-4 Trace connector A pinout

Channel	Pin	Pin	Channel
No connect	1	2	No connect
No connect	3	4	No connect
GND	5	6	**TRACECLK**
DBGRQ	7	8	**DBGACK**
nSRST	9	10	**EXTTRIG**
TDO	11	12	**VDD** (**ARM_VDDIO**) 3.3V
RTCK	13	14	**VDD** 3.3V
TCK	15	16	**TRACEPKT7** (**TRACEPKTA7**)
TMS	17	18	**TRACEPKT6** (**TRACEPKTA6**)
TDI	19	20	**TRACEPKT5** (**TRACEPKTA5**)
nTRST	21	22	**TRACEPKT4** (**TRACEPKTA4**)
TRACEPKT15 (**TRACEPKTA15**)	23	24	**TRACEPKT3** (**TRACEPKTA3**)
TRACEPKT14 (**TRACEPKTA14**)	25	26	**TRACEPKT2** (**TRACEPKTA2**)
TRACEPKT13 (**TRACEPKTA13**)	27	28	**TRACEPKT1** (**TRACEPKTA1**)
TRACEPKT12 (**TRACEPKTA12**)	29	30	**TRACEPKT0** (**TRACEPKTA0**)
TRACEPKT11 (**TRACEPKTA11**)	31	32	**TRACESYNC** (**PIPESTATA3**)
TRACEPKT10 (**TRACEPKTA10**)	33	34	**PIPESTAT2** (**PIPESTATA2**)
TRACEPKT9 (**TRACEPKTA9**)	35	36	**PIPESTAT1** (**PIPESTATA1**)
TRACEPKT8 (**TRACEPKTA8**)	37	38	**PIPESTAT0** (**PIPESTATA0**)

The CM926EJ-S, CM1026EJ-S, and CM1136JF-S use two Trace connectors. The second connector is used for ETM demultiplexed mode and is shown in Table A-5.

The ARM testchip **ETMEXTIN** pins are connected to the FPGA so that they can be driven by a special ETM validation FPGA configuration. In normal configurations the **ETMEXTTRIG** signal from the Trace Port connector is routed through the FPGA to the **ETMEXTIN[1]** pin.

Table A-5 Trace connector B (J15) pinout

Channel	Pin	Pin	Channel
No connect	1	2	No connect
No connect	3	4	No connect
GND	5	6	**ARM_TRACECLKB**
No connect	7	8	No connect
No connect	9	10	No connect
No connect	11	12	**ARM_VDDIO**
No connect	13	14	**3V3**
No connect	15	16	**TRACEPKTB7**
No connect	17	18	**TRACEPKTB6**
No connect	19	20	**TRACEPKTB5**
No connect	21	22	**TRACEPKTB4**
TRACEPKTB15	23	24	**TRACEPKTB3**
TRACEPKTB14	25	26	**TRACEPKTB2**
TRACEPKTB13	27	28	**TRACEPKTB1**
TRACEPKTB12	29	30	**TRACEPKTB0**
TRACEPKTB11	31	32	**ARM_PIPESTAT3**
TRACEPKTB10	33	34	**ARM_PIPESTAT2**
TRACEPKTB9	35	36	**ARM_PIPESTAT1**
TRACEPKTB8	37	38	**ARM_PIPESTAT0**

A.4 Logic analyzer connectors

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

- *HADDR connector* on page A-12
- *CONTROL connector* on page A-12
- *HRDATA connector* on page A-14
- *HWDATA connector* on page A-15.

Note

The CM926EJ-S, CM1026EJ-S, and CM1136JF-S do not have the **HWDATA** connector.

HRDATA follows **HWDATA** on write cycles. If logic analyzer connections are limited HRDATA must be used.

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

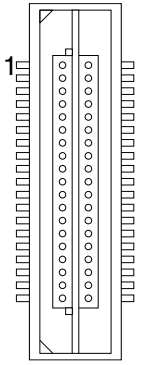


Figure A-4 AMP Mictor connector

Note

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

A.4.1 HADDR connector

Table A-6 shows the pinout of J9.

Table A-6 HADDR (J9)

Channel	Pin	Pin	Channel
No connect	1	2	No connect
GND	3	4	No connect
HTRANS1	5	6	**HCLK**
HADDR31	7	8	**HADDR15**
HADDR30	9	10	**HADDR14**
HADDR29	11	12	**HADDR13**
HADDR28	13	14	**HADDR12**
HADDR27	15	16	**HADDR11**
HADDR26	17	18	**HADDR10**
HADDR25	19	20	**HADDR9**
HADDR24	21	22	**HADDR8**
HADDR23	23	24	**HADDR7**
HADDR22	25	26	**HADDR6**
HADDR21	27	28	**HADDR5**
HADDR20	29	30	**HADDR4**
HADDR19	31	32	**HADDR3**
HADDR18	33	34	**HADDR2**
HADDR17	35	36	**HADDR1**
HADDR16	37	38	**HADDR0**

A.4.2 CONTROL connector

Table A-7 on page A-13 shows the pinout of J12.

Note

Pins 14 to 28 (even pin numbers) are unused on the CM946E-S and CM966E-S. These pins are used for SRAM on the CM926EJ-S, CM1026EJ-S, and CM1136JF-S.

Table A-7 Control (J12)

Channel	Pin	Pin	Channel
No connect	1	2	No connect
GND	3	4	No connect
nRESET	5	6	Unused
nIRQ	7	8	Unused
nFIQ	9	10	Unused
HBUSREQ	11	12	Unused
HGRANT	13	14	**SRAM_CKEN**
HLOCK	15	16	**SRAM_ADVLDn**
HPROT3	17	18	**SRAM_A20**
HPROT2	19	20	**SRAM_A19**
HRESP1	21	22	**SRAM_A18**
HRESP0	23	24	**SRAM_CEn**
HREADY	25	26	**SRAM_OEn**
Unused	27	28	**SRAM_A17**
HSIZE1	29	30	**HBURST2**
HSIZE0	31	32	**HBURST1**
HPROT1	33	34	**HBURST0**
HPROT0	35	36	**HTRANS1**
HWRITE	37	38	**HTRANS0**

A.4.3 HRDATA connector

Table A-8 shows the pinout of J11.

Table A-8 HRDATA (J11)

Channel	Pin	Pin	Channel
No connect	1	2	No connect
GND	3	4	No connect
HREADY	5	6	No connect
HRDATA31	7	8	**HRDATA15**
HRDATA30	9	10	**HRDATA14**
HRDATA29	11	12	**HRDATA13**
HRDATA28	13	14	**HRDATA12**
HRDATA27	15	16	**HRDATA11**
HRDATA26	17	18	**HRDATA10**
HRDATA25	19	20	**HRDATA9**
HRDATA24	21	22	**HRDATA8**
HRDATA23	23	24	**HRDATA7**
HRDATA22	25	26	**HRDATA6**
HRDATA21	27	28	**HRDATA5**
HRDATA20	29	30	**HRDATA4**
HRDATA19	31	32	**HRDATA3**
HRDATA18	33	34	**HRDATA2**
HRDATA17	35	36	**HRDATA1**
HRDATA16	37	38	**HRDATA0**

A.4.4 HWDATA connector

Table A-9 shows the pinout of J10 for the CM946E-S and CM966E-S.

Table A-9 HWDATA (J10)

Channel	Pin	Pin	Channel
No connect	1	2	No connect
GND	3	4	No connect
No connect	5	6	**HCLK**
HWDATA31	7	8	**HWDATA15**
HWDATA30	9	10	**HWDATA14**
HWDATA29	11	12	**HWDATA13**
HWDATA28	13	14	**HWDATA12**
HWDATA27	15	16	**HWDATA11**
HWDATA26	17	18	**HWDATA10**
HWDATA25	19	20	**HWDATA9**
HWDATA24	21	22	**HWDATA8**
HWDATA23	23	24	**HWDATA7**
HWDATA22	25	26	**HWDATA6**
HWDATA21	27	28	**HWDATA5**
HWDATA20	29	30	**HWDATA4**

Table A-9 HWDATA (J10) (continued)

Channel	Pin	Pin	Channel
HWDATA19	31	32	**HWDATA3**
HWDATA18	33	34	**HWDATA2**
HWDATA17	35	36	**HWDATA1**
HWDATA16	37	38	**HWDATA0**

Note

Only three logic analyzer connections are on the CM926EJ-S, CM1026EJ-S, and CM1136JF-S. For these modules, use HRDATA (J11) instead of the missing HWDATA as the FPGA also drives the write data onto the read data bus.

Appendix B
Specifications

This appendix contains the specifications for the ARM Integrator/CM926EJ-S, Integrator/CM946E-S, Integrator/CM966E-S, Integrator/CM1026EJ-S, and Integrator/CM1136JF-S core modules. It contains the following sections:

B.1 Electrical specification

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

B.1.1 Bus interface characteristics

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

Table B-1 Core module electrical characteristics

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

B.1.2 Current requirements

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

Table B-2 Current requirements (maximum)

System	At 3.3V	At 5V CM946E-S CM966E-S	At 5V CM926EJ-S CM1026EJ-S CM1136JF-S
Standalone core module	1A	100mA	1A
Motherboard (AP) and one core module	1.5A	500mA	1.5A

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

The core modules use an AHB system bus. The timing parameters for this bus are shown in:

- Table B-3 *Clock and reset parameters*
- Table B-4 on page B-5 *AHB slave input parameters* on page B-5
- Table B-5 on page B-5 *AHB slave output parameters* on page B-5
- Table B-6 on page B-5 *Bus master input timing parameters* on page B-5
- Table B-7 on page B-6 *Bus master output timing parameters* on page B-6.

In addition notes on how the parameters are derived are given in *Notes on FPGA timing analysis* on page B-6.

B.2.1 Core module timing and the AMBA Specification

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

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

Each version and revision of the FPGA has subtly different timing. The typical figures shown in Table B-3 are those you can expect under nominal conditions and must be used as a guideline when designing your own motherboards and modules. The typical figures have been rounded to simplify timing analysis and constraints.

B.2.2 Timing parameter tables

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

Table B-3 Clock and reset parameters

Parameter	Description	Typ
Tclk	**HCLK** clock period	30
Tisrst	**HRESETn** deasserted setup time before **HCLK**	15

Table B-4 shows the AHB slave input parameters.

Table B-4 AHB slave input parameters

Parameter	Description	Typ
Tistr	Transfer type setup time before **HCLK**	5
Tisa	**HADDR[31:0]** setup time before **HCLK**	10
Tisctl	**HWRITE**, **HSIZE[2:0]** and **HBURST[2:0]** setup time before HCLK	5
Tiswd	Write data setup time before **HCLK**	5
Tisrdy	Ready setup time before **HCLK**	5

Table B-5 shows the AHB slave output parameters.

Table B-5 AHB slave output parameters

Parameter	Description	Typ
Tovrsp	Response valid time after **HCLK**	15
Tovrd	Data valid time after **HCLK**	15
Tovrdy	Ready valid time after **HCLK**	15

Table B-6 shows the bus master input parameters.

Table B-6 Bus master input timing parameters

Parameter	Description	Typ
Tisgnt	**HGRANTx** setup time before **HCLK**	5
Tisrdy	Ready setup time before **HCLK**	5
Tisrsp	Response setup time before **HCLK**	5
Tisrd	Read data setup time before **HCLK**	5

Table B-7 shows the bus master output timing parameters.

Table B-7 Bus master output timing parameters

Parameter	Description	Typ
Tovtr	Transfer type valid time after **HCLK**	15
Tova	Address valid time after **HCLK**	15
Tovctl	**HWRITE**, **HSIZE[2:0]** and **HBURST[2:0]** valid time after HCLK	15
Tovwd	Write data valid time after **HCLK**	15
Tovreq	Request valid time after **HCLK**	15
Tovlck	Lock valid time after **HCLK**	15

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 similar to those shown the tables in *Timing parameter tables* on page B-4. The exact performance of a system depends on the timing parameters of the motherboard and all modules in the system. Some allowance also has to be made for clock skew, routing delays and number of modules (that is, loading).

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

A detailed timing analysis involves calculating the input/output delays between modules for all timing parameters. In general, the simplest approach to determine the maximum operating frequency is to increase the frequency of the clock generators until the system becomes unstable.

B.3 Mechanical details

The core module is designed to be stackable on a number of different motherboards. Its size allows it to be mounted onto a CompactPCI motherboard while allowing the motherboard to be installed in a card cage.

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

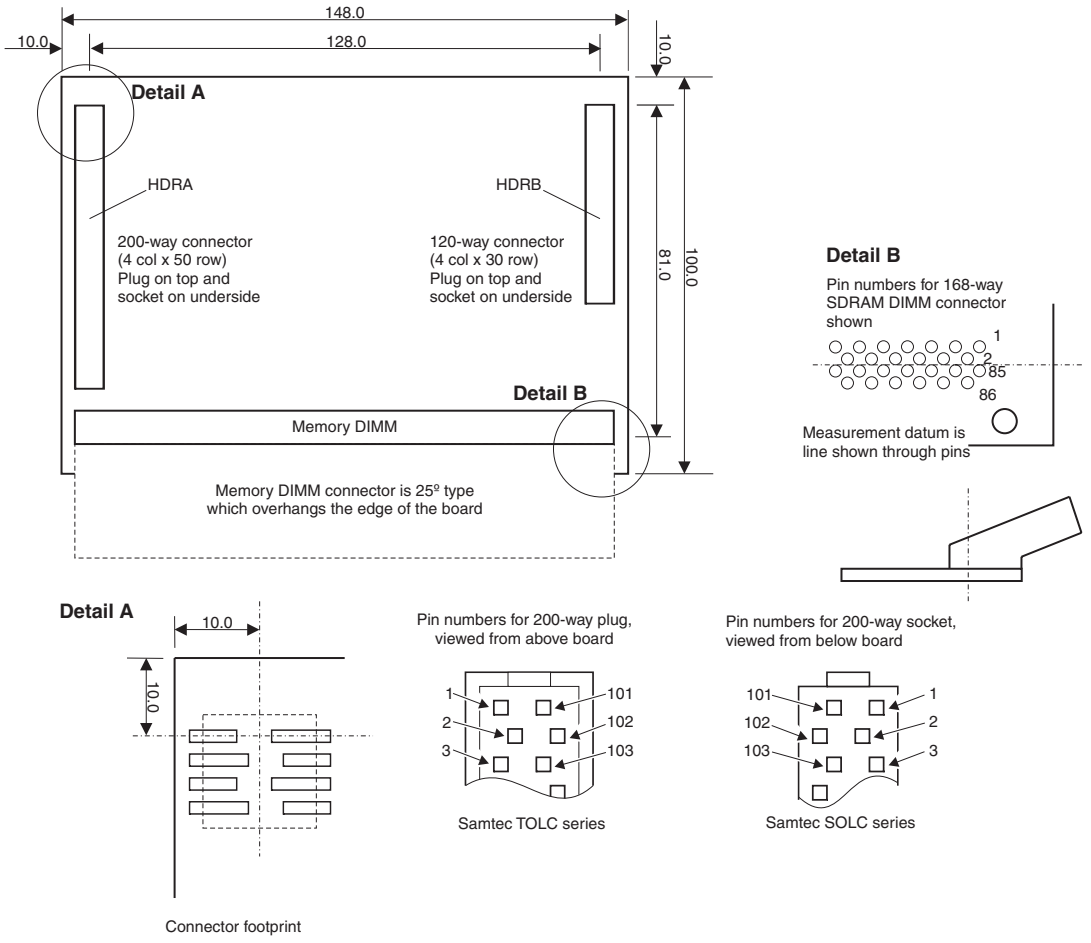


Figure B-1 Board outline

Appendix C
Features specific to the CM1136JF-S

This appendix describes features specific to the CM1136JF-S core module. It contains the following sections:

C.1 Introduction to the CM1136JF-S

The Integrator/CM1136JF-S Core Module contains an ARM1136JF-S r0p1 test chip and an FPGA which implements the core module functions.

The core module for the ARM1136JF-S is based on the core module for the ARM926EJ-S. This document highlights the differences.

C.1.1 Platform support

The FPGA on the core module contains two different configurations. When the module is plugged into a motherboard or baseboard the FPGA loads the appropriate configuration, depending upon the state of the **CFGSEL[1:0]** signals.

Standalone and Integrator/AP operation

The Integrator/AP FPGA configuration defaults to synchronous operation of the test chip with clocks at 180:36:36MHz, (1:5:5 ratios) on power-up. The ratio and synchronous mode of operation can be changed on a reset (see *Test chip clock control* on page C-3).

Integrator/CP operation

The Integrator/CP FPGA configuration defaults to synchronous operation of the test chip with clocks at 150:30:30MHz (1:5:5 ratios) on power-up. The ratio and synchronous mode of operation can be changed on a reset (see *Test chip clock control* on page C-3).

C.1.2 Boot Monitor

If the core module is connected to the Integrator/AP or Integrator/CP baseboard, the Boot Monitor is executed from flash memory at power-up. Boot Monitor v1.4.2 does not support the new clock configuration implemented in the ARM1136JF-S Test Chip (See *Clocks* on page C-3). The core clock is recognized and can be changed correctly. However there is no **HCLKDIV** divider in the test chip and the **HCLKI** and **HCLKE** ratios cannot be altered. Therefore the user should be careful not to over clock the local memory bus.

C.2 ARM1136JF-S test chip characteristics

The CM1136JF-S Core Module uses test chips that conform to the generic test chip specification. The test chip in the CM1136JF-S contains:

- a Clock Generator Control Block that configures the clocking scheme for the core
- an AHB matrix
- a vectored interrupt controller (VIC)
- 16KB of Data and 16KB of Instruction tightly-coupled memory (TCM)
- 16KB of Data and 16KB of Instruction cache
- The test chip is contains an ETM11RV embedded trace macrocell. See the *ETM11RV Technical Reference Manual* for more details on the ETM.

See the ARM1136JF-S Technical Reference Manual for further details of the TCMs and cache memory.

Note

The standard cache and TCM size is 16KB, but some core modules versions might have larger sized memories. Refer to the release notes supplied with your core module for details.

C.2.1 Clocks

The ARM1136JF-S can be operated with synchronous or asynchronous core and internal bus clocks.

If the ARM1136JF-S is operating synchronously, the core frequency is divided to clock the internal bus and the core module local memory bus. The dividers are programmable and set the bus frequencies.

Test chip clock control

During a power-on reset the value on the **HRDATA[31:0]** pins is loaded into the test chip PLL configuration register. The value is hard-wired into the FPGA image and sets:

- the clock circuit to synchronous operation with a ratio of 1:5:5
- the PLL control is set to BYPASS by default
- the CPU core runs from the core clock, **CORECLK**, which directly feeds the **REFCLK** input on the test chip.

The core clock frequency can be set by accessing the core module registers starting at address `0x10000000`, see Chapter 4 *Programmer's Reference*. The ARM1136JF-S test chip does not use the HCLK DIV functions implemented on other core modules.

The clocks generated by the clock and reset generator are controlled by the Clock Generator Control Register at `0x3F200080`. Table C-1 lists the format of the Clock Generator Control Register.

Table C-1 Clock Generator Control Register

Bit	Description
[31]	Reserved, should be zero
[30]	Value of **HSYNCENIRW** and **SYNCENIRW** when the port is configured as asynchronous: 0 = **HSYNCENIRW** and **SYNCENIRW** signals are driven to 0 1 = **HSYNCENIRW** and **SYNCENIRW** signals are driven to 1.
[29]	Value of **HSYNCENPD** and **SYNCENPD** when the port is configured as synchronous: 0 = **HSYNCENPD** and **SYNCENPD** signals are driven to 0 1 = **HSYNCENPD** and **SYNCENPD** signals are driven to 1.
[28]	Selects the clock source for **HCLKI** and **HCLKE**: 0 = **HCLKI** and **HCLKE** derived from **PLLCLK** 1 = **HCLKI** and **HCLKE** derived from **HCLKIN**.
[27]	Configures Instruction, Data Read, and Data Write ports of the ARM1136JF-S test chip as synchronous or asynchronous: If Bit 27 is set to 0 the ports are configured as synchronous: **HCLKIRW** is driven from **CLK** **HCLKIRWEN** is driven from **HCLKIEN** **HSYNCENIRW** and **SYNCENIRW** are driven to 1. If Bit 27 is set to 1 the ports are configured as asynchronous: **HCLKIRW** is driven from **HCLKI** **HCLKIRWEN** is driven to 1 **HCLKIRW** is driven from **HCLKI** **HSYNCENIRW** and **SYNCENIRW** are driven from bit 30 of this register.

Table C-1 Clock Generator Control Register (continued)

Bit	Description
[26]	Configures whether Peripheral and DMA ports of the ARM1136JF-S processor are configured as synchronous or asynchronous: If Bit 26 is set to 0 the ports are configured as synchronous: **HCLKPD** is driven from CLK **HCLKPDEN** is driven from **HCLKIEN** **HSYNCENPD** and **SYNCENPD** are driven to 1 If Bit 26 is set to 1 the ports are configured as asynchronous: **HCLKPD** is driven from **HCLKI** **HCLKPDEN** is driven to 1 **HSYNCENPD** and **SYNCENPD** are driven from bit 30 of this register.
[25:20]	Sets the clock ratio for the **HCLKE** domain.
[19:14]	Sets the clock ratio for the **HCLKI** domain.
[13:8]	Sets the clock ratio for the **CLK** domain.
[7:0]	Reserved, should be zero

The programmed clock ratios determine the divisor between the source clock and the output clock. The source clock for **CLK** is always **PLLCLK**. The divisor selected is given by the following equation:

```
Divisor = Programmed ratio +1
```

The maximum divisor that you can program is 63.

Not all values of the Clock Generator Control Register are legal. The following restrictions apply:

1. **HCLKE** ratio must be an integer multiple of the **HCLKI**.
2. If you program the ports to be synchronous, that is if bit 27 is 0 and bit 26 is 0, you must program the **CLK** ratio to be 1:1.
3. If you select the clock source for **HCLKI** and **HCLKE** from **HCLKIN** (bit 28 is set to 1), then you must program the ports to be asynchronous (set bits 27 and 26 to 1). You must also set the **HCLKI** ratio must be 1:1.

Caution

Changes to bits [28:26] of this register only take effect during reset (**ARM_nRESET** is LOW). The changes to the clock ratios can take place at any point. You must exercise care to ensure that the programming of bits [28:26] have propagated to the clock

generator before selecting an asynchronous ratio. This means that the sequence for programming an asynchronous clock ratio involves setting bits [27:26], forcing a simple reset, and then programming the asynchronous clock ratio.

The clocks are disabled if **ARM_nPORESET** is LOW.

Changing any of the clock ratio values only takes effect during a reset. This is because clock ratio values are propagated from the register to the clock dividers during a reset only. This reset might be the same reset used to propagate changes in the asynchronous control bits, bits[28:26].

The minimum duration of an externally applied reset on the signal **ARM_nRESET** is one clock cycle of the slowest clock.

The outputs of this block are **CLKBUFDRV**, **HCLKIBUFDRV**, and **HCLKEBUFDRV**. These are exported to explicit clock buffering to help with the balancing of the clocks.

The Clock Divisor Register is accessed using the AHBPCAPT block as described in AHBPCAPT Slave Registers.

The reset values of the Clock Generator Control Register are defined from the Configuration Register, see *Configuration Register* on page C-23.

Synchronous/asynchronous clock control

Under normal operation, the clock ratios can be changed dynamically by accessing the Clock Generator Control Register at address `0x3F200080`. However, the synchronous/asynchronous mode of operation cannot be changed dynamically and requires the test chip to be reset.

During reset, predefined bits in the same format as the Test Chip Clock Generator are presented on **HRDATA**. This value is sampled on the rising edge of **CONFIGINIT**, when **nCONFIGRST** is HIGH and **ARM_nPORESET** is LOW, and written into the Test Chip Clock Generator Control Register. This programs the default clock ratios for the standalone/AP and CP images from a power on reset

In asynchronous mode, **HCLKIN** is driven directly from **BUSCLK**. This enables you to drive core clock **REFCLK** independently from **HCLKIN**.

Core module oscillator register

The Core Module Oscillator Register (CM_OSC at `0x10000008`) has been modified to enable control of the bus clock frequency.

Table C-2 Core Module Oscillator Register

Bits	Name	Access	Function
[31:16]	-	Read Modify Write to preserve value.	Reserved
[24:23]	BMODE	Read Modify Write to preserve value.	Bus Mode. Always contains b10. Select the bus operation by writing to CM_INIT.
[22:15]	BUS_VDW	Read/write	Bus clock VCO divider word. Defines the binary value on the V[7:0] pins of the clock generator for **ARM_HCLK** (**HCLK**). (Bit[8] is tied LOW)
[14:12]	BUS_ OD	Read/write	Bus output divider. Sets the binary code on the S[2:0] pins of the clock generator. The encoding is the same as for the AUX_OD bits in the CM_OSC register.
[11:8]	-	Read Modify Write to preserve value.	Reserved
[7:0]	PLL_VDW	Read/write	Core clock VCO divider word. Defines the binary value on the V[7:0] pins of the clock generator for **ARM_PLLCLKIN** (**REFCLK**). (Bit[8] is tied LOW)

Example of synchronous operation

The core module starts at a fixed frequency and ratio from power-up. The frequency is set to 180MHz for an Integrator/AP and 150MHz for an Integrator/CP configuration. The ratios are set to 1:5:5 for both configurations.

To change a ratio, write directly to the Clock Generator Control Register at address `0x3F200080`. The effect of changing the ratios is immediate.

```
ldr r0, =0x00410000   ; 1:5:5 ratio
ldr r1, =0x3F200080   ; Clock Generator Control Register address
str r0, [r1]          ; Change ratio to 1:5:5 immediately
```

Caution

Setting the core clock frequency takes affect immediately. Change the clock and ratio in the correct order. Increasing the core clock speed increases the system clock. Therefore, first increase the clock ratio then increase the core clock.

```
ldr r0, =0x00514000  ; 1:6:6 ratio
ldr r1, =0x3F200080  ; Clock Generator Control Register address
str r0, [r1]         ; Change ratio to 1:6:6 immediately
```

This allows the system to operate within a reasonable operating frequency. Before setting the core frequency, unlock the CM_OSC register.

```
ldr r0, =0x0000a05f    ; Unlock value
ldr r1, =0x10000014    ; Unlock Register address
str r0, [r1]           ; Write the unlock value to the unlock register
```

You can now modify the core clock.

```
ldr r0, =0x0000008C ; New core clock frequency
ldr r1, =0xFFFFFF00 ; Mask
ldr r2, =0x10000008 ; Core Module Oscillator Register (CM_OSC)
ldr r3, [r2]        ; Read Modify Write
and r1, r1, r3      ; Clear old core clock bits
orr r1, r1, r0      ; Add new clock value
str r1, [r2]        ; Write value back to CM_OSC register
```

Example of asynchronous operation

The core module starts at a fixed frequency and ratio from power-up. The frequency is set to 180MHz for an Integrator/AP and 150MHz for an Integrator/CP configuration. The ratios are set to 1:5:5 for both configurations.To change from synchronous mode to asynchronous mode and back, set and clear bits `[28:26]` in the clock control register. The only mechanism that exists to do this is the Clock Generator Control register at address `0x3F200080`. After changing the register values, perform a soft reset by pressing the reset button or by writing to the Core Module Control register.

```
ldr r0, =0x1C000000    ; 1:1:1 ratio (asynch)
ldr r1, =0x3F200080    ; Clock Generator Control Register address
str r0, [r1]           ; Change ratio to 1:1:1 (async mode)
ldr r0, =0x0000a05f    ; Unlock value
ldr r1, =0x10000014    ; Unlock Register address
str r0, [r1]           ; Write the unlock value to the unlock register
ldr r0, =0x1000000C    ; CM_CTRL Register
ldr r1, [r0]           ; Get the current value
orr r1, r1, #0x08      ; Set the reset bit
str r1, [r0]           ; Set the reset bit in the register
```

The core and bus clock must be set to appropriate values before the reset takes place to allow sensible values to be running from reset. The core module FPGA uses the clock output from the test chip to clock the internal circuit therefore only a 1:1 ratio for the **BUSCLK** is valid.

```
ldr r0, =0x0000a05f  ; Unlock value
ldr r1, =0x10000014  ; Unlock Register address
str r0, [r1]         ; Write the unlock value to the unlock register
ldr r0, =0x000B10AC  ; New core clock frequency (180MHz) bus clock (30MHz)
ldr r1, =0xFF800F00  ; Mask
ldr r2, =0x10000008  ; Core Module Oscillator Register (CM_OSC)
ldr r3, [r2]         ; Read Modify Write
and r1, r1, r3       ; Clear old core clock bits
orr r1, r1, r0       ; Add new clock value
str r1, [r2]         ; Write value back to CM_OSC register
```

On a CP image, a change in clock frequency does not occur until a simple reset is preformed. To increment the oscillator value immediately without performing a reset, set bit 26 to 1 in addition to setting the clock frequency.

```
ldr r0, =0x0000a05f    ; Unlock value
ldr r1, =0x10000014    ; Unlock Register address
str r0, [r1]           ; Write the unlock value to the unlock register
ldr r0, =0x0400008C    ; New core clock frequency & update bit
setldr r1, =0xFFFFFF00 ; Mask
ldr r2, =0x10000008    ; CM_OSC register
ldr r3, [r2]           ; Read Modify Write
and r1, r1, r3         ; Clear old core clock bits
orr r1, r1, r0         ; Add new clock value
str r1, [r2]           ; Write value back to CM_OSC register
```

C.2.2 AHB matrix and memories block

The AHB matrix and memories block provides a routing network for all memory accesses generated by the ARM1136JF-S ports. Memory accesses can be routed to:

- RAM implemented on the test chip
- memory mapped peripherals contained on the test chip
- the external AHB interface of the test chip.

The implementation of this block is constrained by the pinout of the test chip, and requires the external AHB interface to have a 32-bit data bus. The ARM1136JF-S processor has four 64-bit AHB-Lite master ports, and one 32-bit AHB-Lite master port. Table C-3 describes the five AHB-Lite interfaces.

Table C-3 AHB interfaces

Port	Type of port	Direction	Data bus width
ARM1136 Instruction Port	AHB-Lite	Input	64
ARM1136 Data Read Port	AHB-Lite	Input	64
ARM1136 Data Write Port	AHB-Lite	Output	64
ARM1136 DMA Port	AHB-Lite	I/O	64
ARM1136 Peripheral Port	AHB-Lite	I/O	32

The RAM implemented on the test chip is split into two parts:

- a large area of 64-bit wide RAM (the primary test chip RAM)
- a 8KB block of 32-bit wide RAM (the secondary test chip RAM).

These blocks, combined with the external AHB interface, enable accesses from any four ports to occur concurrently. Figure C-1 on page C-11 shows the AHB matrix and memories

Note

The **HADDRx** signal names in Figure C-1 on page C-11 indicate the different AHB buses.

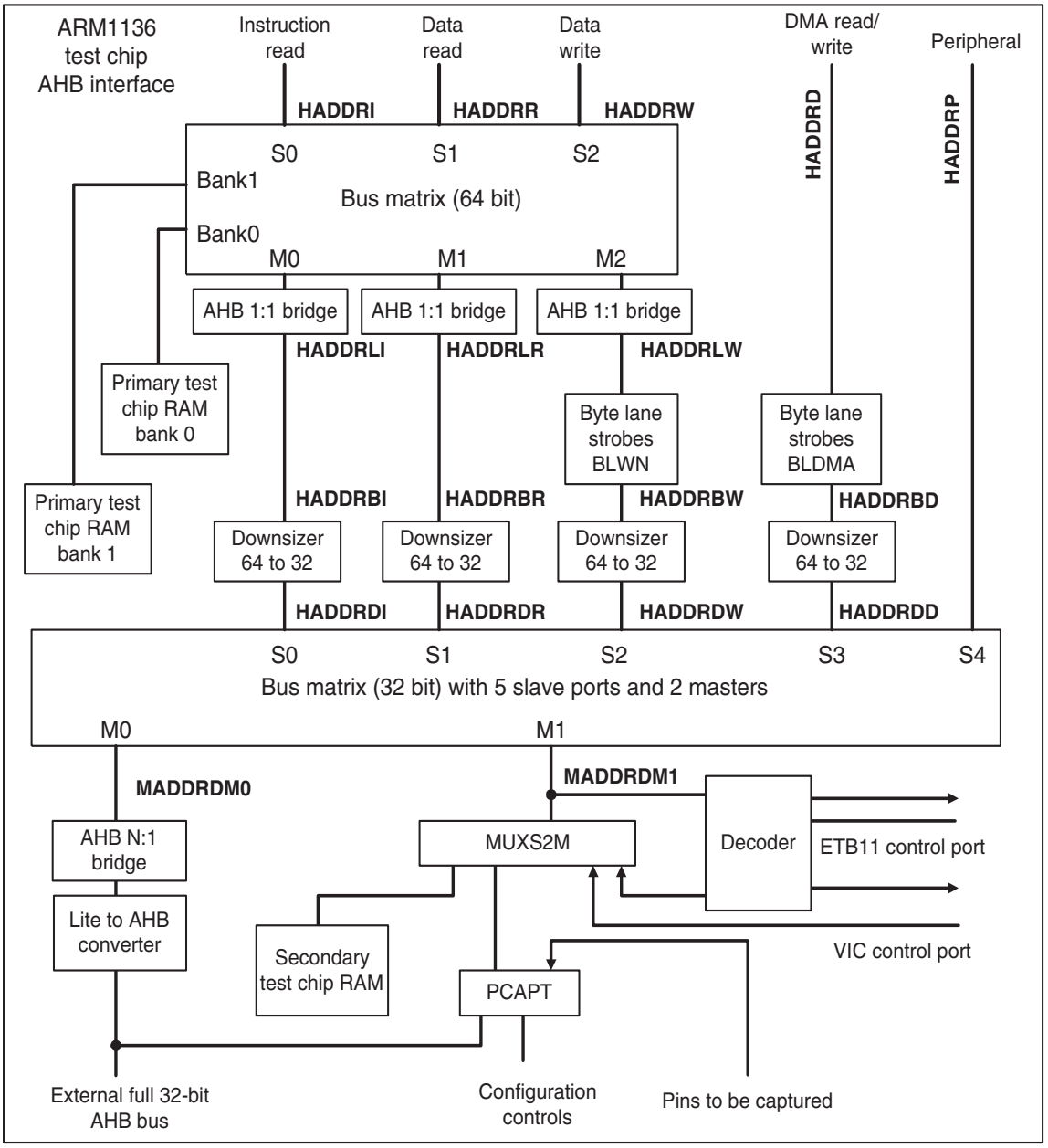


Figure C-1 AHB matrix and memories

For details of components, see the *AMBA Design Kit Technical Reference Manual* and *AMBA Extension Infrastructure Blocks Technical Reference Manual*.

C.2.3 On-chip memory-mapped peripherals

The on-chip memory-mapped peripherals are implemented in the test chip because:

- The AHBPCAPT and VIC blocks contain functionality to improve the validation coverage that can be achieved using the test chip
- The AHBPCAPT block contains registers to control the functionality of the test chip
- The ETB11 requires AHB accesses as a slave to validate its functionality.

C.2.4 External AHB interface

All accesses that do not access the primary test chip RAM, the secondary test chip RAM, or the on-chip memory-mapped peripherals are presented on the test chip external 32-bit wide AHB interface. This is a full AHB interface provides the interface to the majority of the memory map of the test chip. The rest of the test chip memory infrastructure is implemented using AHB-Lite and multi-layer AHB.

C.2.5 AHB address map

Table C-4 lists the base address of the various ARM1136JF-S test chip AHB peripherals. These peripherals are multiply mapped throughout their allocated address ranges. The table also indicates the ARM1136JF-S AHB-Lite ports that are able to access each peripheral.

Table C-4 Test chip address map

Module name	Address	Accessible from	Size	Description
TestChipRAM (BANK0)	0x3FFFFFFF-0x3F800000	Instruction read, Data Read, Data Write	4MB	Internal 64-bit primary test chip RAM bank. Interleaved on 1KB boundaries with BANK1. Actual size of this RAM depends on the space available within the test chip, but typically it is 256KB.
TestChipRAM (BANK1)	0x3FFFFFFF-0x3F800000	Instruction read, Data Read, Data Write	4MB	Internal 64-bit primary test chip RAM bank. Interleaved on 1KB boundaries with BANK0. Actual size of this RAM depends on the space available within the test chip, but typically it is 256KB.
SPARE	0x3F7FFFFF-0x3F400000	All ports	4MB	SPARE (access is presented on the test chip AHB interface).
Etb11TraceSRAM	0x3F3FFFFF-0x3F380000	All ports	512KB	ETB11 memory.
ETBREG	0x3F37FFFF-0x3F300000	All ports	512KB	ETB11 registers.
AHBPCAPT	0x3F2FFFFF-0x3F200000	All ports	1MB	Pin capture and ETM11RV validation control.
AHBVIC	0x3F1FFFFF-0x3F100000	All ports	1MB	Control port for VIC.
XRAM2Kx32	0x3F0FFFFF-0x3F000000	All ports	1MB	Secondary test chip RAM, 32-bit AHB RAM. 8KB RAM implemented within this space, otherwise wraps.

C.2.6 Block disables

Table C-5 lists block disables controlled from the Configuration Register and defined for the memory block. The programming of these registers is described in *Configuration Register* on page C-23.

Table C-5 External control inputs

Signal	Description
BLKDISABL[5]	Disables the AHBRAM0 and AHBRAM1 primary test chip RAM blocks. Accesses are presented on the test chip AHB interface if this bit is configured HIGH.
BLKDISABL[4]	Disables the AHBETB Mem address range. Accesses are presented on the test chip AHB interface if this bit is configured HIGH.
BLKDISABL[3]	Disables the AHBETB Reg address range. The accesses are presented on the test chip AHB interface if this bit is configured HIGH.
BLKDISABL[2]	Disables the AHBPCAPT block. Accesses are presented on the test chip AHB interface if this bit is configured HIGH.
BLKDISABL[1]	Disables the AHBVIC block. Accesses are presented on the test chip AHB interface if this bit is configured HIGH.
BLKDISABL[0]	Disables the AHBRAM2 secondary test chip RAM block. Accesses are presented on the test chip AHB interface if this bit is configured HIGH.

C.2.7 AHB submodules

This section describes the submodules in the test chip.

Bus matrix 64

The 64-bit bus matrix, provides a connection between the Instruction Read, Data Read, and Data Write ports of the ARM1136JF-S core to the primary test chip RAM blocks. There are other connections to pass the accesses in parallel to the rest of the memory system. The 64-bit bus matrix consists of five slave ports, and three master ports.

The bus matrix 64 includes extensions to AHB for the use of byte lane strobes, providing support for **HUNALIGN** and **HBSTRB[n:0]**.

Bus matrix arbitration is fixed IR-DR-DW (highest to lowest priority).

Primary test chip RAMs

The two 64-bit wide RAMs are designed to be fast RAM blocks that can be accessed efficiently close to the core, enabling a high-performance memory system.

The three primary ports of the ARM1136JF-S core (the Instruction Read, Data Read, and Data Write ports) are each 64 bits wide, and have accesses to the two 64-bit wide primary test chip RAMs, and to other parts of the memory map. You cannot use the remaining two ports, the DMA port and Peripheral port, to access the primary test chip RAMs. The primary test chip RAMs exist to provide a relatively high capacity, high bandwidth, and low latency area of RAM that can be accessed by the primary ports. The primary RAMs are interleaved on 1KB boundaries.

AHB 1:1 bridges

The AHB 1:1 bridges connect to three of the master ports in the bus matrix 64, providing a cycle boundary between the two bus matrices, so avoiding an unacceptable critical path. These bridges include extensions to AHB for the use of byte lane strobes providing support for **HUNALIGN** and **HBSTRB[n:0]**.

Byte lane strobe converter

The byte lane strobe converter block takes in ARM11 AMBA extension signals that include **HUNALIGN** and **HBSTRB[n:0]**, and provide a mirror slave AHB port on the other side without **HBSTRB[n:0]** or **HUNALIGN**, that is AMBA v2.0 compliant signals. These blocks are inserted so that any slaves after these blocks do not have to support **HUNALIGN** and **HBSTRB[n:0]**.

The byte lane strobe converters are implemented only for ports that can do unaligned writes. These are the Data Write port and the DMA port.

Downsizer 64-bit to 32-bit

The downsizer converts an AMBA AHB v2.0 with 64-bit data bus to the same type of bus, but with a 32-bit data bus. It arranges for two accesses to be performed on the narrow side for each wide (64-bit) access.

Secondary test chip RAM

One of the AHB peripherals connected to the bus matrix 32 is a 32-bit SRAM to give a maximum of four parallel accesses being handled in the design. The secondary test chip RAM exists to increase the number of regions of RAM that can be accessed concurrently. It consists of a simple 32-bit 8KB block of RAM which is implemented in parallel with the on-chip peripherals.

Bus matrix 32

Bus matrix arbitration must be fixed IR-DR-DW-DMA-PERIPH (highest to lowest priority).

You can configure bus matrix blocks to have round-robin or fixed priority. The ARM1136JF-S test chip bus matrix (32-bit) is a fixed priority block. Input port 0 has the highest priority, and the priority decreases as the input port number increases.

AHB N:1 bridge

The AHB N:1 bridge provides the boundary between the **HCLKI** and the **HCLKE** clock domains. The clock ratio between **HCLKI** and **HCLKE** can be any integer value, including 1.

Lite2AHB converter

To convert from an AHB-Lite to full AHB functionality, a Lite2AHB converter block is used. This provides a full AHB port at the external AHB interface.

C.2.8 AHB slaves and support blocks

There are various AHB slaves connected to the bus matrix port M1. Standard AHB Decoder and AHB MuxS2M blocks are required to support the connection of multiple slaves to a single port. Table C-6 lists the AHB slaves. See the *AMBA Specification (Rev 2.0)* for details on how to connect up an AMBA AHB system.

Table C-6 AHB slaves

Component	Description
AHBVIC	Control port for the Vectored Interrupt Controller
AHBPCAPT	Pin capture block.
AHBRAM2	32-bit AHB SRAM. This RAM is 8KB.
AHBETB	ARM11 Embedded Trace Buffer (ETB11) Control.

The base address of these AHB slave ports is defined in *AHB address map* on page C-13.

C.2.9 AHBPCAPT slave registers

Table C-7 lists the addresses of the ARM1136JF-S AHBPCAPT Slave Registers.

Table C-7 ARM1136JF-S AHBPCAPT Slave Registers

Address	Type	Register
0x3F200000	Read/write	ARM1136JF-S Pin Capture Register
0x3F200010	Read-only	Test Chip Memory Size Register
0x3F200020	Read-only	Instruction AHB Port Register 0
0x3F200024	Read-only	Instruction AHB Port Register 1
0x3F200028	Read-only	Instruction AHB Port Register 2
0x3F200030	Read-only	Data Read AHB Port Register 0
0x3F200034	Read-only	Data Read AHB Port Register 1
0x3F200038	Read-only	Data Read AHB Port Register 2
0x3F200040	Read-only	Data Write AHB Port Register 0
0x3F200044	Read-only	Data Write AHB Port Register 1
0x3F200048	Read-only	Data Write AHB Port Register 2
0x3F200050	Read-only	DMA AHB Port Register 0
0x3F200054	Read-only	DMA AHB Port Register 1
0x3F200058	Read-only	DMA AHB Port Register 2
0x3F200060	Read-only	Peripheral AHB Port0 Register
0x3F200064	Read-only	Peripheral AHB Port1 Register
0x3F200068	Read-only	Peripheral AHB Port2 Register
0x3F200080	Read/write	Clock Generator Control Register
0x3F2000C0	Read/write	Test Chip Control Register
0x3F200100	Read-only	ETM11RV Pin Capture1 Register
0x3F200104	Read-only	ETM11RV Pin Capture2 Register
0x3F200108	Read-only	ETM11RV Pin Capture3 Register
0x3F200110	Read/write	ETM11RV Pin Control1 Register

Table C-7 ARM1136JF-S AHBPCAPT Slave Registers (continued)

Address	Type	Register
0x3F200114	Read/write	ETM11RV Pin Control2 Register
0x3F200118	Read/write	ETM11RV Pin Control3 Register
0x3F20011C	Read/write	ETM11RV Pin Control4 Register

Table C-8 lists the Pin Capture Register bit allocation.

Table C-8 Pin Capture Register bit allocation

Bit	Pins	Reset Value
[31:24]	Reserved, should be zero	Unpredictable
[23:16]	**DMAASID**	0
[15:8]	**COREASID**	0
[7:5]	Reserved, should be zero	Unpredictable
[4]	**DBGNOPWRDWN**	0
[3]	**CFGBIGENDPD**	0
[2]	**CFGBIGENDIRW**	0
[1]	**STANDBYWFI**	0
[0]	**Sticky STANDBYWFI**	Unpredictable

The **Sticky STANDBYWFI** bit is set if **STANDBYWFI** has been asserted in any cycle since the value was written by the core.

Table C-9 lists the Test Chip Memory Size Register bit allocation.

Table C-9 Test Chip Memory Size Register

Bit	Function
[31:5]	Unpredictable
[4:0]	Test chip RAM size

Table C-10 lists the valid test chip RAM size values.

Table C-10 Valid test chip RAM sizes

Hex	Size
0x07	64KB
0x08	128KB
0x09	256KB
0x0A	512KB
0x0B	1MB
0x0C	2MB
0x0D	4MB
0x0E	8MB

The AHB port registers read the values of the **HBSTRB[7:0]**, **HUNALIGN**, and **HSIDEBAND** signals to the last three non-sequential AHB transfers that have occurred for each port. The values are held as follows:

- The most recent transfer is held in Register 0
- The second most recent is held in Register 1
- The third most recent is held in Register 2.

The registers therefore act as FIFOs. Table C-11 lists the contents of each AHB Port Register.

Table C-11 Contents of each AHB Port Register

BIT	Function
[31:14]	Unpredictable
[13]	**WRITEBACK** (Data Write Port only. Other ports are UNP)
[12:9]	**HSIDEBAND[3:0]**
[8]	**HUNALIGN**
[7:0]	**HBSTRB[7:0]** for the Peripheral Port bits [7:4] are UNP

Table C-12 lists the Test Chip Control Register bit allocations.

Table C-12 Test Chip Control Register

Bit	Function	Reset value
[31:1]	Reserved, should be zero	Unpredictable
[0]	VIC enable	0

The VIC is disabled by default. This means that no programming of the VIC is required to connect **nFIQX** and **nIRQX** directly to the ARM1136JF-S core.

Table C-13 lists the ETM11RV Pin Capture1 Register bit allocation.

Table C-13 ETM11RV Pin Capture1 Register

Bit	Function	Reset value
[31:10]	Reserved, should be zero	Unpredictable
[9]	**~nETMWFIREADY**	0
[8:6]	**ETMPORTMODE**	0
[5:2]	**ETMPORTSIZE**	0
[1]	**ETMEN**	0
[0]	**~ETMPWRUP**	0

Table C-14 lists the ETM11RV Pin Capture2 Register bit allocation.

Table C-14 ETM11RV Pin Capture2 Register

Bit	Function	Reset value
[31:2]	Reserved, should be zero	Unpredictable
[1:0]	**ETMEXTOUTX**	Unpredictable

Table C-15 lists the ETM11RV Pin Capture3 Register bit allocation.

Table C-15 ETM11RV Pin Capture3 Register

Bit	Function	Reset value
[31:8]	Reserved, should be zero	Unpredictable
[7:0]	**ETMASICCTL**	Unpredictable

Table C-16 lists the ETM11RV Pin Control1 Register bit allocation.

Table C-16 ETM11RV Pin Control1 Register

Bit	Function	Reset value
[31:4]	Reserved, should be zero	Unpredictable
[3:0]	**EtmExtInRega**	`0`

Table C-17 lists the ETM11RV Pin Control2 Register bit allocation.

Table C-17 ETM11RV Pin Control2 Register

Bit	Function	Reset value
[31:11]	Reserved, should be zero	Unpredictable
[10]	ORed into **ETMWFIPENDING**	`0`
[9:6]	**ETMMAXPORTSIZE**	`b0100`
[5:3]	**ETMMAXEXTOUT**	`b010`
[2:0]	**ETMMAXEXTIN**	`0`

Table C-18 lists the ETM11RV Pin Control3 Register bit allocation.

Table C-18 ETM11RV Pin Control3 Register

Bit	Function	Reset value
[31:20]	Reserved, should be zero	Unpredictable
[19:0]	**EtmEvntBusRega**	`0`

Table C-19 lists the ETM11RV Pin Control4 Register bit allocation.

Table C-19 ETM11RV Pin Control4 Register

Bit	Function	Reset value
[31:6]	Reserved, should be zero	Unpredictable
[5:0]	**EtmInputSel**	0

Table C-20 lists the EtmInputSel[1:0] encoding used to determine the source of **ETMEXTINX**.

Table C-20 EtmInputSel[1:0] encoding

EtmInputSel[1:0]	ETMEXTINX to ETM
b00	0
b01	EtmExtInReg
b10	(**DBGACKX**, PseudoRandom[2:0])
b11	((b000, **ETMEXTINX** (test chip))

Table C-21 lists the EtmInputSel[3:2] encoding used to determine the source of **ETMEVNTBUS**

Table C-21 EtmInputSel[3:2] encoding

EtmInputSel[3:2]	ETMEVNTBUS to ETM
b00	0
b01	EtmEvntBusReg
b10	PseudoRandom[19:0]
b11	**ETMEVNTBUS** from ARM1136JF-S

Table C-22 lists the EtmInputSel[5:4] encoding used to determine the source of **ETMTRACEREADY**

Table C-22 EtmInputSel[5:4]

EtmInputSel[5:4]	ETMEVNTBUS to ETM
b00	1
b01	PseudoRandom[0]
b10	0
b11	1 whenever an instruction is executed.

Configuration Register

The AHBPCAPT block contains the configuration register space. This space is programmed when the test chip is held in its lowest level of reset (that is, when **ARM_nPORESET** is asserted). The Configuration Register is a 32-bit register that is programmed to the value on **HRDATAX** signals at the rising edge of **CONFIGINIT**.

The Configuration Register is reset by asserting **nCONFIGRST**, and only when **ARM_nPORESETX** is asserted. The Configuration Register contents cannot be modified from software. Use the Clock Generator Control Register to change the clock frequency after power on.

Table C-23 lists the bit allocation of the Configuration Register.

Table C-23 Configuration Register

Bit	Function	Reset value (nCONFIGRST)
[31]	Reserved, should be zero	0
[30:8]	Reset value of bits [28:8] of the Clock Generator Control Register	0
[7:2]	Test chip memory block disables	0

C.2.10 Vectored Interrupt Controller (VIC) block

The VIC has the following interfaces:

- interrupt sources
- AHB slave interface to control the VIC
- daisy chain interface to connect two or more VICs together
- scan chain interface.

There are 32 interrupt source inputs. For details of the VIC interrupts sources connections, see the *ARM PrimeCell Vectored Interrupt Controller (PL192) Technical Reference Manual*.

Note

The daisy chain interface is tied off, because only a single VIC is present in the ARM1136JF-S test chip. By default, the VIC is disabled. Use the Test Chip Control Register to enable the VIC (see Table C-12 on page C-20).

Interrupt routing

Table C-24 lists the ARM1136JF-S test chip signal routing to the VIC interrupt sources.

Table C-24 Test chip interrupt routing

VIC input	Source	Comment
VICINTSOURCE[0]	NOT (**nFIQ**)	Pin on the test chip
VICINTSOURCE[1]	NOT (**nVALFIQ**)	ARM1136JF-S output
VICINTSOURCE[6]	**COMMRX**	ARM1136 JF-S output
VICINTSOURCE[7]	**COMMTX**	ARM1136 JF-S output
VICINTSOURCE[10]	NOT (**nIRQ**)	Pin on the test chip
VICINTSOURCE[11]	NOT (**nVALIRQ**)	ARM1136 JF-S output
VICINTSOURCE[12]	NOT (**nDMAIRQ**)	ARM1136 JF-S output
VICINTSOURCE[13]	NOT (**nPMUIRQ**)	ARM1136 JF-S output
VICINTSOURCE[14]	NOT (**nVALRESET**)	ARM1136 JF-S output

Note

All other **VICINTSOURCE** signals are tied to 0.

Index

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

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W