Integrator/IM-AD1

User Guide

ARM

ARM DUI 0163B

Integrator/IM-AD1
User Guide

Release Information

Date	Issue	Change
Oct 2001	A	New document
Nov 2003	B	Second release with minor corrections

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks owned by ARM Limited, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

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

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

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

Confidentiality Status

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

Product Status

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

Web Address

`http://www.arm.com`

Conformance Notices

This section contains conformance notices.

Federal Communications Commission Notice

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

CE Declaration of Conformity

The system should be powered down when not in use.

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

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

Note

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

Contents
Integrator/IM-AD1 User Guide

Chapter 4 Reference Design Example

Appendix A Signal Descriptions

Appendix B Mechanical Specification

Glossary

Preface

This preface introduces the Integrator/IM-AD1 interface module and its user documentation. It contains the following sections:

- *About this book* on page viii
- *Feedback* on page xi.

About this book

This book provides user information for the ARM Integrator/IM-AD1 interface module. It describes the major features and how to use the interface module with an Integrator development platform.

Intended audience

This book is written for all developers who are using an Integrator/LM logic module to develop ARM-based devices. It assumes that you are an experienced developer, and that you are familiar with the ARM development tools.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the Integrator/IM-AD1 interface module. This chapter describes the main features of the interface module and identifies the main components.

Chapter 2 *Getting Started*

Read this chapter for information about preparing the interface module for use with a logic module and Integrator/AP motherboard.

Chapter 3 *Hardware Reference*

Read this chapter for a description of the interface module hardware.

Chapter 4 *Reference Design Example*

Read this chapter for a description of the example logic module configuration supplied that enables you to experiment with the interface module.

Appendix A *Signal Descriptions*

Read this appendix for connector pinout information.

Typographical conventions

The following typographical conventions are used in this book:

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

Further reading

This section lists publications from both ARM Limited and third parties that provide additional information on developing code for the ARM family of processors.

ARM periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets and addenda.

See also the ARM Frequently Asked Questions list on the ARM web site.

ARM publications

The following documents provide information about related Integrator products:

- *ARM Integrator/AP User Guide* (ARM DUI 0098)
- *ARM Integrator/CM9x6E-S User Guide* (ARM DUI 0138)
- *ARM Integrator/CM920T-ETM User Guide* (ARM DUI 0149)
- *ARM Integrator/CM9x0T and CM7x0T User Guide* (ARM DUI 0157)
- *ARM Integrator/CM7TDMI User Guide* (ARM DUI 0126)
- *ARM Integrator/LM-XCV600E+ and LM-EP20K600E+ User Guide* (ARM DUI 0146).

The following publications provide information about ARM PrimeCell devices that can be used to control some of the interfaces described in this manual:

- *ARM PrimeCell UART (PL011) Technical Reference Manual* (ARM DDI 0183)
- *ARM PrimeCell Synchronous Serial Port Master and Slave (PL022) Technical Reference Manual* (ARM DDI 0194)
- *ARM PrimeCell DC-DC Converter Interface (PL160) Technical Reference Manual* (ARM DDI 0147)
- *ARM PrimeCell Vectored Interrupt Controller (PL190) Technical Reference Manual* (ARM DDI 0181).

The following publications provide reference information about ARM architecture:

- *AMBA Specification* (ARM IHI 0011)
- *ARM Architectural Reference Manual* (ARM DDI 0100).

The following publications provide information about the ARM Developer Suite:

- *Getting Started* (ARM DUI 0064)
- *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).

The following publication provides information about Multi-ICE:

- *Multi-ICE User Guide* (ARM DUI 0048).

Third-party documents

The following documents provide information about third-party components used on the Integrator/IM-AD1:

- *CC770 Stand Alone CAN Controller Target Specification*
  Robert Bosch GmbH
- *L6506 L6506D Current Controller For Stepping Motors*
  SGS-Thomson Microelectronics
- *L298 Dual Full Bridge Driver*
  SGS-Thomson Microelectronics
- *AD7859 Datasheet*
  Analog Devices, Inc.
- *AD5342 Datasheet*
  Analog Devices, Inc.

Feedback

ARM Limited welcomes feedback on both the Integrator/IM-AD1 and its documentation.

Feedback on this document

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

- the document title
- the document number
- the page number(s) to which your comments apply
- a concise explanation of your comments.

General suggestions for additions and improvements are also welcome.

Feedback on the Integrator/IM-AD1

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

- the product name
- an explanation of your comments.

Chapter 1
Introduction

This chapter introduces the Integrator/IM-AD1. It contains the following sections:

1.1 About the Integrator/IM-AD1

The Integrator/IM-AD1 is an interface module that is designed to be used in conjunction with the Integrator/LM-XCV600E+ or LM-EP20K600E+ and future compatible logic modules. It provides several standard automotive and industrial interfaces, and can be used for:

- proof of concept and application development
- benchmarking
- PrimeCell development and testing
- SoC prototyping.

The interface module is designed to be mounted on top of the logic module and provides connectivity for peripherals in the logic module FPGA.

Figure 1-1 on page 1-3 shows the layout of the IM-AD1 and identifies the connectors.

The IM-AD1 can be used to implement additional peripherals to aid software development, for example additional timers or a vector interrupt controller. The IM-AD1 product consists of:

- a fully populated PCB
- example bit files for an LM
- example driver source code
- this documentation.

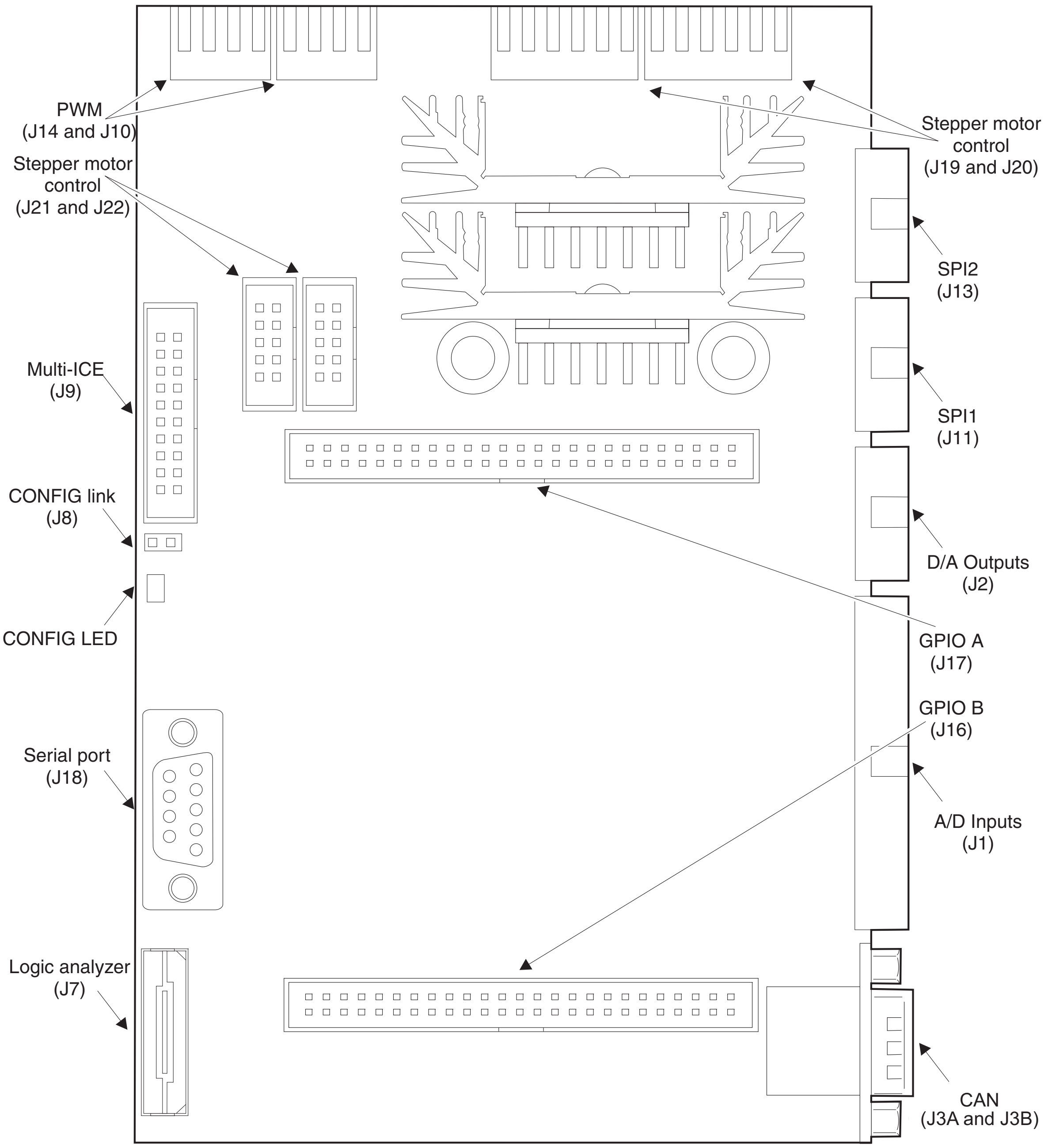


Figure 1-1 Integrator/IM-AD1 layout

1.2 Interface module features and architecture

This section describes the main features of the interface module and its architecture.

1.2.1 Features

The main features of the interface module are as follows:

- two Bosch CC770 *Controller Area Network* (CAN) controllers
- two 8-channel 12-bit *Analog to Digital Converters* (ADC) with 200ksamples/s sampling rate and 0-5V buffered inputs
- two 12-bit *Digital to Analog Converter* (DAC) channels with 0-5V outputs
- two L298 stepper motor drivers configured for bipolar motors
- two stepper motor control outputs for use with off-board drivers
- two 3A MOSFETs configured as switches for the *Pulse Width Modulated* (PWM) outputs
- RS232 serial transceiver
- two 32-bit *General Purpose Input Output* (GPIO) connectors
- JTAG (Multi-ICE) pass-through connector
- logic analyzer connector connected to the GPIO bus.

1.2.2 Architecture

Figure 1-2 shows the architecture of the interface module. For more detail on signal routing between the expansion connectors and the interface circuits, see Chapter 3 *Hardware Reference*.

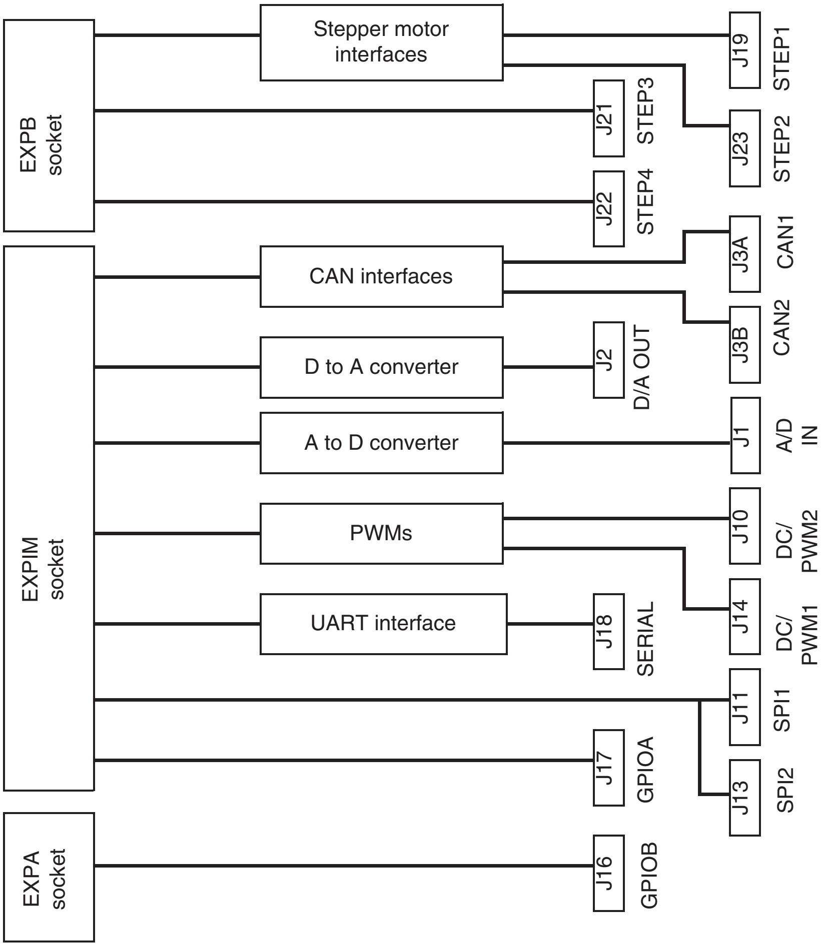


Figure 1-2 Integrator/IM-AD1 block diagram

1.3 Links and LEDs

The interface module provides one link and one LED. These are the CONFIG link and CONFIG LED.

Fitting the CONFIG link places all of the modules in the stack on which the interface module is mounted into CONFIG mode. This mode enables you to reprogram the FPGA image in the configuration flash on the logic module(s) using Multi-ICE (see the user guide for the logic module).

The CONFIG LED lights to indicate when the stack is in CONFIG mode.

1.4 Care of modules

This section contains advice about how to prevent damage to your Integrator modules.

Caution

To prevent damage to your Integrator system, observe the following precautions:

- When removing a core or logic module from a motherboard, or when separating modules, take care not to damage the connectors. Do not apply a twisting force to the ends of the connectors. Loosen each connector first before pulling on both ends of the module at the same time.
- Use the system in a clean environment and avoid debris fouling the connectors on the underside of the PCB. Blocked holes can cause damage to connectors on the motherboard or module below. Visually inspect the module to ensure that connector holes are clear before mounting it onto another board.
- Observe *ElectroStatic Discharge* (ESD) precautions when handling any Integrator board.

Integrator/IM Interface Modules are designed to connect with ARM and third party development platforms that meet the interface specification. Interface Modules connect to the top of a stack because there are no EXPA/B connectors on the top side.

Chapter 2
Getting Started

This chapter describes how to set up and start using the logic module. It contains the following sections:

- *Fitting the interface module* on page 2-2
- *Setting up the logic module* on page 2-3
- *Running the test software* on page 2-4

2.1 Fitting the interface module

The interface module is installed at the top of a stack of up to four logic modules. However, it only provides interface connections for the logic module immediately beneath it.

Figure 2-1 shows an example system comprising a core module and logic module attached to an Integrator/AP (see the *Integrator/AP User Guide* for more details). The interface module is installed at the top of the logic module stack.

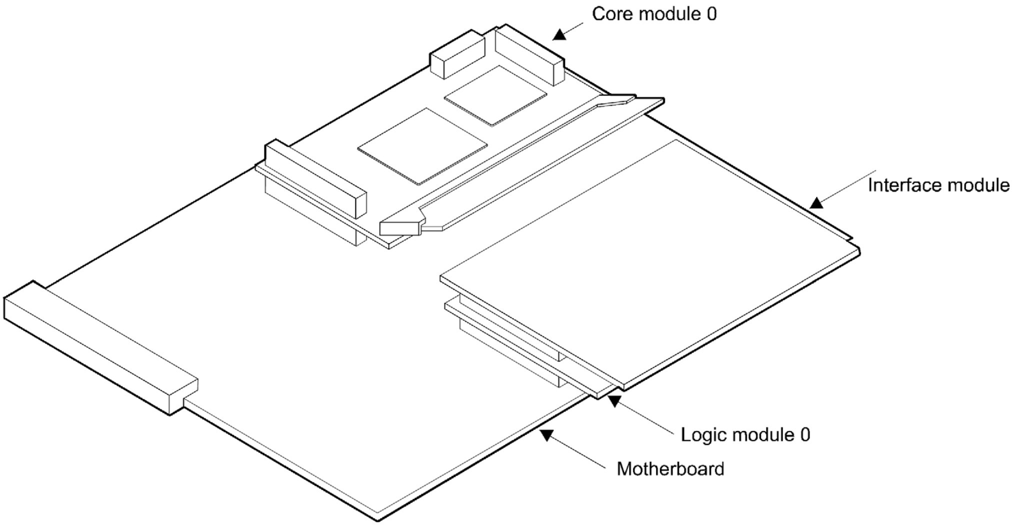


Figure 2-1 Assembled Integrator development system

2.2 Setting up the logic module

You must load the required peripheral controllers into the logic module FPGA to drive the interfaces. The interface module is supplied with example configurations that provide PrimeCell peripherals for supported logic modules.

The logic module can be programmed using Multi-ICE with or without the IM-AD1 fitted. If the IM-AD1 is fitted however, the manufacturer-specific download connector on the logic module is inaccessible. See the logic module user guide for detailed instructions on downloading new FPGA configurations

To download the supplied example logic module FPGA configuration using Multi-ICE:

1. Insert CONFIG link on the logic module (or IM-AD1 if fitted to logic module).
2. Connect Multi-ICE unit to J10 on the logic module (or J9 on the IM-AD1).
3. Power up the Integrator system.
4. Start the Multi-ICE server on your PC and click the **`Autoconfigure`** button.
5. If you are using an Altera logic module, LM-EP20K1000E, switch 4 of switchpack S1 must be set to the CLOSED position.
6. Browse to: *`Install_directory`*`\IM-AD1\configure`.
7. Double-click the `progcards.exe` program file.
8. The `progcards` program automatically detects whether the logic module is an Altera or a Xilinx module and uses the appropriate `.brd` file to download the configuration file.
9. After the programming has completed:
    - power down the system
    - remove the CONFIG link
    - move the Multi-ICE connection to the core module.
10. Set the S1 switches on the logic module as follows:

    Switch 1 Open
    Switch 2 Closed
    Switch 3 Open
    Switch 4 Open.

The logic module will now be configured with the example design.

If the IM-AD1 is not already fitted, install it on top of the logic module and the system is ready to use.

2.3 Running the test software

The supplied test program tests each of the interfaces on the IM-AD1. The example logic module configuration must be programmed into the logic module before the test program can be run.

Note

The test software requires various cables to be connected to the IM-AD1, details of these are given in the `readme.txt` file on the IM-AD1 CD.

To run the test program:

1. Connect a Multi-ICE unit to the core module.
2. Power up the Integrator system.
3. Start the Multi-ICE server and autoconfigure it.
4. Browse to:

`Install_directory\IM-AD1\example\software\selftest\build\ads1.1\selftest_Data\Release`

5. Double-click the `selftest.axf` file. This starts the ARM debugger and loads the test program.
6. Check the debugger is configured to use Multi-ICE by selecting **Options** → **Configure Target**. If Multi-ICE is not highlighted, select it and click **`OK`**.
7. Click **`Yes`** at the **`Reload last image`** prompt.
8. Press the **`Go`** button on the debugger to run the test software.

Use the menu in the debugger console window to test individual interfaces or to run all the tests on all interfaces.

Chapter 3

Hardware Reference

This chapter describes the hardware interfaces and controllers on the interface module. This chapter contains the following sections:

3.1 Differences in signal routing between supported logic modules

The Integrator/LM-XCV600E+ and LM-EP20K600E+ logic module types route the signals from the interface module differently as follows:

The

- LM-XCV600E+ is fitted with a Xilinx FPGA and routes the interface module **ABANK[59:0]** signals to bank 0 on the FPGA and the **BBANK[53:0]** signals to bank 1 on the FPGA.
- The LM-EP20K600E+ is fitted with an Altera FPGA and routes the interface module **ABANK[59:0]** signals to bank 5 on the FPGA and the **BBANK[53:0]** signals to bank 6 on the FGPA.

Note

These pin assignments are contained in the example pin constraints file on the CD supplied with the interface module.

3.2 UART interface

The interface module provides one serial transceiver suitable for use with the PrimeCell UART (PL011) or other similar peripheral. Figure 3-1 shows the architecture of the UART interface.

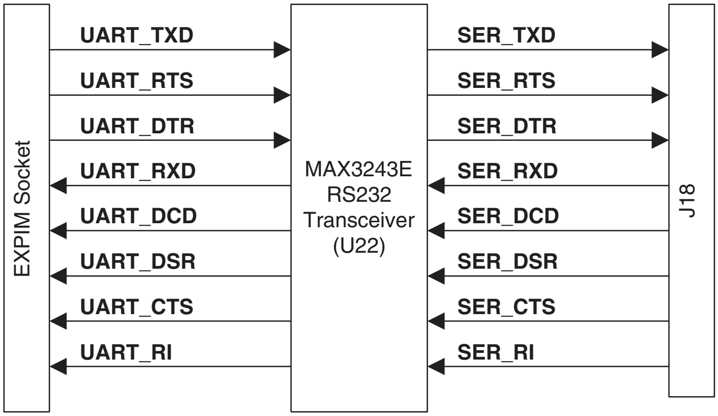


Figure 3-1 Serial interface

The signals associated with the UART interface are assigned to the EXPIM socket pins as shown in Table 3-1.

Table 3-1 Serial interface signal assignment

Signal name	EXPIM connector	Description
UART_TXD	IM_BBANK41	Transmit data
UART_RTS	IM_BBANK42	Ready to send
UART_DTR	IM_BBANK43	Data terminal ready
UART_RXD	IM_BBANK44	Receive data
UART_DCD	IM_BBANK45	Data carrier detect
UART_DSR	IM_BBANK46	Data set ready
UART_CTS	IM_BBANK47	Clear to send
UART_RI	IM_BBANK48	Ring indicator

The serial interface uses a 9-pin D-type male connector for which the pin numbering is shown in Figure 3-2.

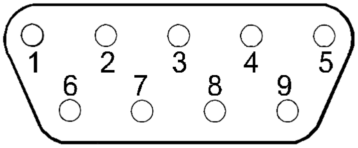


Figure 3-2 Serial connector pinout

Table 3-2 shows the signal assignment for the connector.

Table 3-2 Serial connector signal assignment

Pin	J18	Type	Description
1	**SER_DCD**	Input	Data carrier detect
2	**SER_RXD**	Input	Receive data
3	**SER_TXD**	Output	Transmit data
4	**SER_DTR**	Output	Data terminal ready
5	**SER_GND**	Input	ground
6	**SER_DSR**	Input	Data set ready
7	**SER_RTS**	Output	Ready to send
8	**SER_CTS**	Input	Clear to send
9	**SER_RI**	Input	Ring indicator

Note

The serial interfaces signals operate at RS232 signal levels.

Serial port functionality corresponds to the DTE configuration.

3.3 SPI

This interface module provides two connectors for SPI ports. They are connected directly to the logic module FPGA and are used by the SSP PrimeCell (PL022) in the example configuration.

Table 3-3 shows the assignment of the SPI signals to the logic module signals on the EXPIM connector.

Table 3-3 SPI signals

Signal	EXPIM connector	Description
SPI_CLK	IM_BBANK31	SPI Clock
SPI_TXD	IM_BBANK32	SPI transmit data
SPI_RXD	IM_BBANK33	SPI receive data
SPI_nCS0	IM_BBANK34	SPI chip select 0
SPI_nCS1	IM_BBANK35	SPI chip select 1
SPI_nCS2	IM_BBANK36	SPI chip select 2

Three chip select signals are provided to allow connection of three separate SPI devices. The SPI signals are routed to two connectors, J11 and J13, for ease of connection to different SPI devices, although both are connected to the same set of signals.

Figure 3-3 shows the pinout of the SPI connectors.

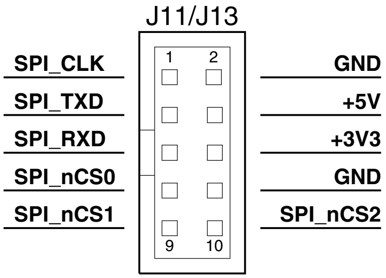


Figure 3-3 SPI interface connector pinout

3.4 PWM interface

The interface module is fitted with a dual MOSFET switch. This provides two outputs that can be configured as *Pulse Width Modulated* (PWM) outputs or used as DC switches to switch external loads.

The MOSFET can switch loads at up to 30V. Although the MOSFET is rated for 3A, because of the power dissipation of the package the maximum load current is 2.5A if only one PWM output is used or 1.75A if both outputs are used.

Warning

The device U21, fitted to the underside of the PCB, and the surrounding area of the board becomes very hot when high load currents are used.

As a PWM output, the interfaces can be driven by the DC-DC PrimeCell (PL160). The DC-DC PrimeCell has feedback inputs that negate the drive outputs when LOW. These inputs can be used to implement a current limit with external circuitry.

Table 3-4 shows the assignment of the PWM interface signals to the logic module signals on the EXPIM connector.

Table 3-4 PWM interface signals

Signal	EXPIM connector	Description
PWM1_DRIVE	IM_BBANK37	PWM1 switch control signal
PWM2_DRIVE	IM_BBANK38	PWM2 switch control signal
PWM1_FB	IM_BBANK39	PWM1 feedback
PWM2_FB	IM_BBANK40	PWM2 feedback

Figure 3-4 shows the pin numbering of the PWM connectors.

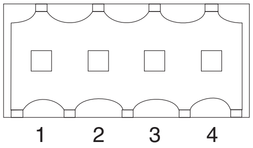


Figure 3-4 PWM interface connector (J10/J14)

Table 3-5 shows the signal assignment.

Table 3-5 PWM connector signals

Pin	J14	J10	Description
1	**PWM1_+V**	**PWM2_+V**	PWM supply voltage
2	**PWM1_SWITCH**	**PWM2_SWITCH**	PWM switched load connection
3	**PWM1_FB**	**PWM2_FB**	PWM feedback signal
4	**PWM_GND**	**PWM_GND**	PWM ground

3.5 Stepper motor interface

The IM-AD1 provides four stepper motor interfaces. Two of these, Step 1 and Step 2, are provided with on-board motor drivers for bipolar motors. The remaining two, Step 3 and Step 4, provide logic-level signals that are connected to two 10-pin headers. This enables you to connect to off-board motor drivers.

3.5.1 Functional description

The on-board stepper motor drivers comprise a L6506 current controller and L298 bridge drivers. These are controlled directly by outputs from the logic module and are configured to drive bipolar motors. Figure 3-5 shows the architecture of one stepper motor driver.

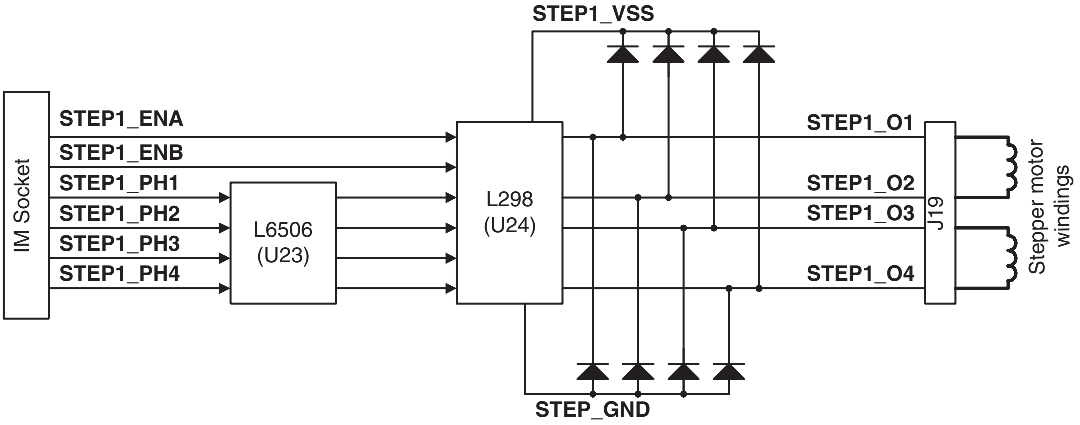


Figure 3-5 Stepper motor drivers (Step 1)

The L298 contains driver circuitry for two bridges, and each bridge has separate enable signals. The enable signals and four phase drive signals are supplied by a stepper motor controller instantiated in the logic module FPGA (see Chapter 4 *Reference Design Example*). The controller logic uses the 4MHz **IM_CLK** signal divided to provide a step clock.

The L6506 uses a chopper circuit, operating at a frequency of 21kHz, to control the current on the phase drive signals to the stepper motor. A 0.1Ω sense resistor is provided to generate a voltage drop that is proportional to the motor current. The sense voltage is compared against a reference voltage of 0.15V that is supplied to the L6506 by a resistive divider. When the reference voltage is reached, the phase drive signals are turned off until the start of the next chopper period.

The current limit is set by the reference voltage and sense resistor according to the equation:

$$I_{peak} = \frac{V_{ref}}{R_{sense}}$$

Therefore, with a 0.1Ω sense resistor fitted:

$$I_{peak} = 0.15 \text{ x } 10 = 1.5\text{A}$$

The reference voltage, and therefore the current limit, can be adjusted by altering the values of the divider resistors. Although the DC current switching limit of L298 is 2A, due to the power dissipation limit of the device and the heatsink, the maximum load current for each stepper winding is 1.5A. The L298 can switch drive voltages up to 46V.

Warning

The L298 devices, U24 and U26, and their heatsinks are very hot when high load currents are used.

3.5.2 Stepper motor interface signal summary

Table 3-6 shows the assignment of the stepper motor interface signals to the logic module signals on the EXPB connector.

Table 3-6 Stepper motor interface signals

Signal	EXPB connector	Description
STEP1_ENA	F0	Enable signal for **STEP1_O1** and **STEP1_O2**
STEP1_ENB	F1	Enable signal for **STEP1_O3** and **STEP1_O4**
STEP1_PH1	F2	Step1 phase 1 drive signal
STEP1_PH2	F3	Step1 phase 2 drive signal
STEP1_PH3	F4	Step1 phase 3 drive signal
STEP1_PH4	F5	Step1 phase 4 drive signal
STEP2_ENA	F6	Enable signal for **STEP2_O1** and **STEP2_O2**
STEP2_ENB	F7	Enable signal for **STEP2_O3** and **STEP2_O4**

Table 3-6 Stepper motor interface signals (continued)

Signal	EXPB connector	Description
STEP2_PH1	F8	Step2 phase 1 drive signal
STEP2_PH2	F9	Step2 phase 2 drive signal
STEP2_PH3	F10	Step2 phase 3 drive signal
STEP2_PH4	F11	Step2 phase 4 drive signal
STEP3_ENA	F12	Enable signal for **STEP3_O1** and **STEP3_O2**
STEP3_ENB	F13	Enable signal for **STEP3_O3** and **STEP3_O4**
STEP3_PH1	F14	Step3 phase 1 drive signal
STEP3_PH2	F15	Step3 phase 2 drive signal
STEP3_PH3	F16	Step3 phase 3 drive signal
STEP3_PH4	F17	Step3 phase 4 drive signal
STEP4_ENA	F18	Enable signal for **STEP4_O1** and **STEP4_O2**
STEP4_ENB	F19	Enable signal for **STEP4_O3** and **STEP4_O4**
STEP4_PH1	F20	Step4 phase 1 drive signal
STEP4_PH2	F21	Step4 phase 2 drive signal
STEP4_PH3	F22	Step4 phase 3 drive signal
STEP4_PH4	F23	Step4 phase 4 drive signal

3.5.3 Stepper motor connectors

Figure 3-6 shows the pin numbering of the stepper motor connectors.

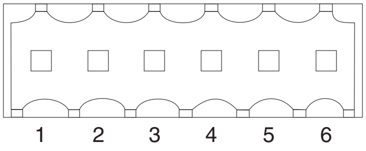


Figure 3-6 Stepper motor connector (J19/J23)

Table 3-7 shows the signal assignment.

Table 3-7 Stepper motor connector signals

Pin	J19	J23	Description
1	**STEP1_VSS**	**STEP2_VSS**	Stepper motor supply
2	**STEP1_O1**	**STEP2_O1**	Stepper motor drive output 1
3	**STEP1_O2**	**STEP2_O2**	Stepper motor drive output 2
4	**STEP1_O3**	**STEP2_O3**	Stepper motor drive output 3
5	**STEP1_O4**	**STEP2_O4**	Stepper motor drive output 4
6	**STEP_GND**	**STEP_GND**	Stepper motor ground

3.6 GPIO

The interface module provides two connectors for GPIO interfaces. Each connector provides 32 GPIO lines connected directly to the logic module FPGA. The connectors are shown in Figure 3-7.

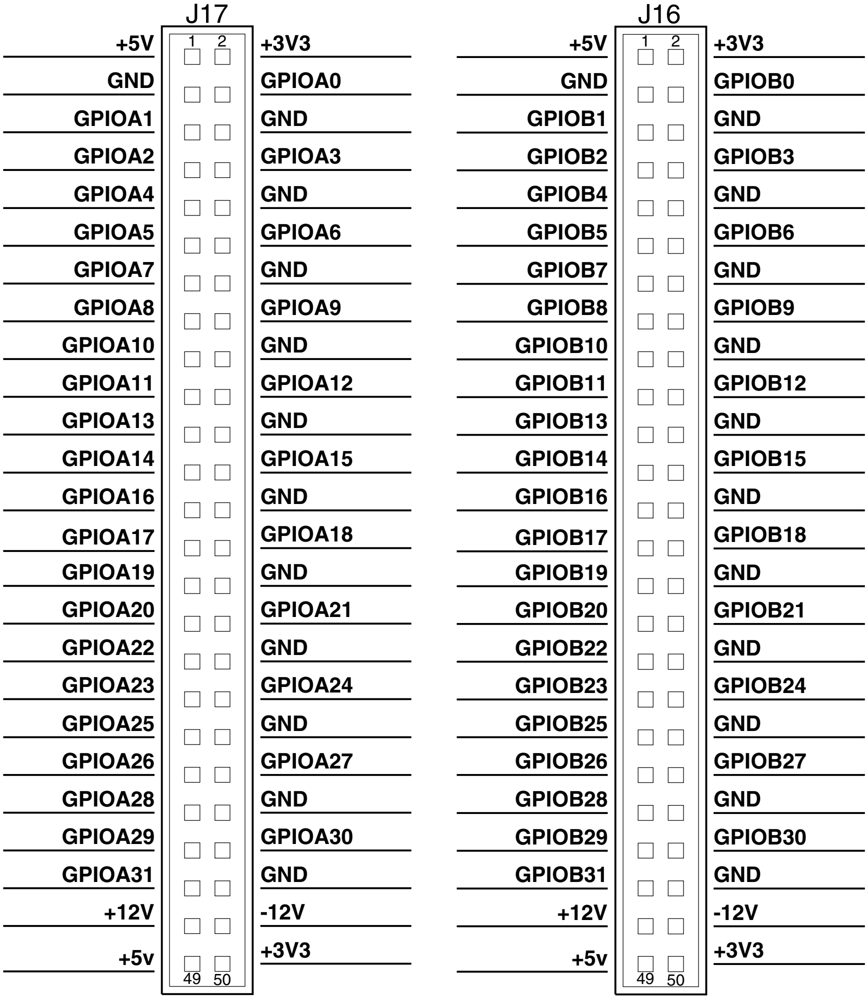


Figure 3-7 GPIO connectors J16 and J17

The example configuration includes two simple 32-bit GPIO controllers. **GPIOA[31:0]** connect to the EXPIM signals **IM_ABANK[31:0]** and **GPIOB[31:0]** connects to the EXPA signals **B[31:0]**. The **B[31:0]** signals can be monitored on the logic analyzer connector J7.

3.7 CAN interface

The IM-AD1 has two CAN interfaces provided by Bosch CC770 serial communications controllers. The network interfaces are provided by Philips TJA1050 transceivers, each capable of 1Mb/s data transfer.

Figure 3-8 shows the architecture of the CAN interface. The CAN controllers are 5V devices and are supported by buffers at their interface with the 3.3V system buses provided by the logic module. The CAN controllers are configured to operate with an 8-bit non-multiplexed asynchronous host interface. Each of the CAN controllers has a 16MHz crystal that it uses for its internal clocks.

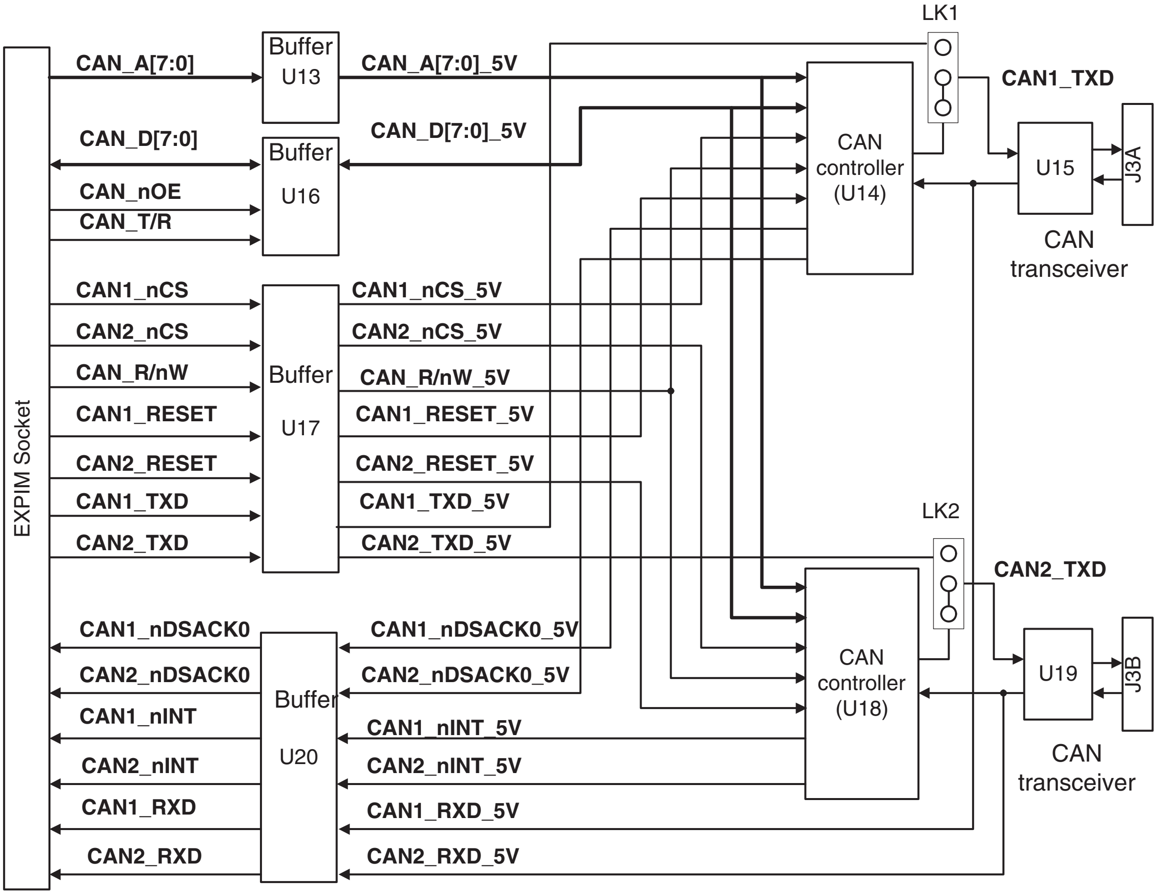


Figure 3-8 CAN interface architecture

All interface signals are routed to the logic module. The CAN controllers are supported by an AHB interface instantiated into the logic module code example supplied with the IM-AD1.

The transmit and receive data signals, **CANx_TXD** and **CANx_RXD**, at the EXPIM connectors are not used for the normal operation of the interfaces. They are provided to allow you to implement your own CAN controller logic in the logic module FPGA.

The surface mount links, LK1 and LK2, are provided so that the transmit data signals to the TJA1050 transceivers can be driven either from the CAN controllers or directly from the logic module FPGA.

Table 3-8 shows the assignment of the CAN controller interface signals to the logic module signals on the EXPIM connector.

Table 3-8 CAN interface signal assignment

Signal	EXPIM connector	Description
CAN_A[7:0]	IM_BBANK[7:0]	CAN address bus
CAN_D[7:0]	IM_BBANK[8:15]	CAN data bus
CAN_T/R	IM_BBANK16	CAN buffer direction control
CAN_nOE	IM_BBANK17	CAN buffer output enable
CAN1_nRESET	IM_BBANK18	CAN1 reset signal
CAN2_nRESET	IM_BBANK19	CAN2 reset signal
CAN_R/nW	IM_BBANK20	CAN read / write
CAN1_nCS	IM_BBANK21	CAN1 chip select
CAN2_nCS	IM_BBANK22	CAN2 chip select
CAN1_TXD	IM_BBANK23	CAN1 transmit data
CAN2_TXD	IM_BBANK24	CAN2 transmit data
CAN1_nDSACK0	IM_BBANK25	CAN1 data acknowledge
CAN2_nDSACK0	IM_BBANK26	CAN2 data acknowledge
CAN1_nINT	IM_BBANK27	CAN1 interrupt

Table 3-8 CAN interface signal assignment (continued)

Signal	EXPIM connector	Description
CAN2_nINT	IM_BBANK28	CAN2 interrupt
CAN1_RXD	IM_BBANK29	CAN1 receive data
CAN2_RXD	IM_BBANK30	CAN2 receive data

You connect the CAN interfaces through the 9-pin D-type plugs J3A (top) and J3B (bottom), with CAN1 connecting to J3A.

Figure 3-9 shows the pin locations for this type of connector.

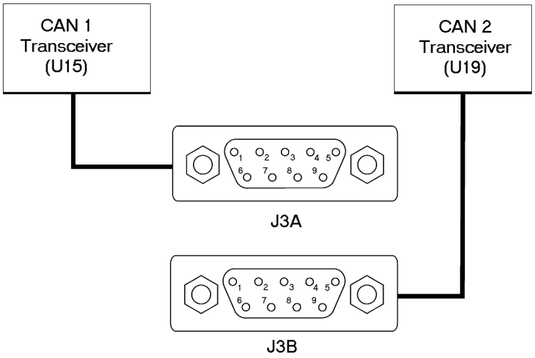


Figure 3-9 CAN connector pin locations

Table 3-9 shows the signal assignment.

Table 3-9 CAN connector signal assignments

Pin	J3A	J3B
1	Not connected	Not connected
2	**CAN1_L**	**CAN2_L**
3	GND	GND
4	Not connected	Not connected
5	GND	GND
6	GND	GND
7	**CAN1_H**	**CAN2_H**
8	Not connected	Not connected
9	Not connected	Not connected

3.8 ADC and DAC interfaces

The interface module provides two *A to D Converters* (ADC) and a *D to A Converter* (DAC). The two ADCs each provide eight analog inputs with buffered 0-5V inputs, an internal multiplexer, and a 12-bit converter. The ADCs provide a 16-bit host interface with conversion data appearing on **D[11:0]** (and zeros on **D[15:12]**). The ADCs are clocked by a 4MHz crystal and are able to perform 200ksamples/s.

The DAC provides two 0-5V outputs with a 12-bit resolution.

The ADCs and DAC are powered from a 5V supply and share buffers to interface them to the 3.3V system bus provided by the logic module.

Figure 3-10 shows the architecture of the ADCs and DACs.

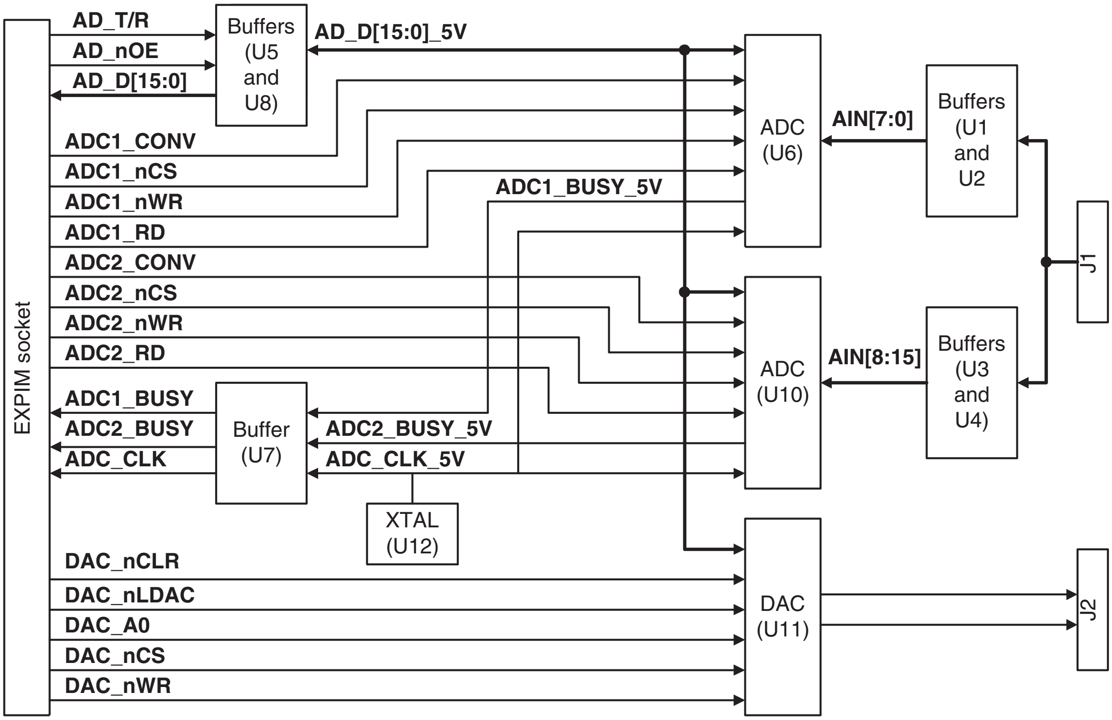


Figure 3-10 ADC and DAC interface architecture

All of the interface signals are routed to the FPGA on the logic module. The ADCs and DAC are supported by an AHB interface that is instantiated in the logic module code example supplied with the IM-AD1.

Table 3-10 shows the assignment of the ADC and DAC interface signals to the logic module signals on the EXPIM connector.

Table 3-10 ADC and DAC interface signals

Signal	EXPIM connector	Description
AD_D[15:0]	IM_ABANK[47:32]	ADC and DAC data bus
AD_T/R	IM_ABANK48	Buffer direction control
AD_nOE	IM_ABANK49	Buffer output enable
ADC1_nCONV	IM_ABANK50	ADC1 conversion start signal
ADC1_nCS	IM_ABANK51	ADC1 chip select
ADC1_nWR	IM_ABANK52	ADC1 write strobe
ADC1_nRD	IM_ABANK53	ADC1 read strobe
ADC2_nCONV	IM_ABANK54	ADC2 conversion start signal
ADC2_nCS	IM_ABANK55	ADC2 chip select
ADC2_nWR	IM_ABANK56	ADC2 write strobe
ADC2_nRD	IM_ABANK57	ADC2 read strobe
ADC1_BUSY	IM_ABANK58	ADC1 busy
ADC2_BUSY	IM_ABANK59	ADC2 busy
DAC_nCLR	IM_BBANK49	DAC clear
DAC_nLDAC	IM_BBANK50	DAC load signal
DAC_A0	IM_BBANK51	DAC address bit
DAC_nCS	IM_BBANK52	DAC chip select
DAC_nWR	IM_BBANK53	DAC write strobe
ADC_CLK	IM_CLK	ADC clock

The ADCs are clocked from a 4MHz oscillator. This also supplies the **IM_CLK** signal routed to the logic module FPGA. This is used in the example logic to clock the DC-DC converter PrimeCell peripheral and the stepper motor interfaces.

The analog inputs to the ADCs are buffered by LMV324 operational amplifiers (op-amps). The op-amps are configured to give unity gain but the inputs have a resistive divider that divides the input voltage by 2. A 0-5V input signal range at the buffer inputs provides a 0-2.5V full range at the ADC input. If different input ranges are required the divider resistor values can be changed.

The op-amp buffers cannot drive their outputs lower than 65mV. This means input signals less than 130mV will have incorrect ADC values.

The reference voltage from one of the ADCs is buffered and fed to the reference inputs of the other ADC and the DAC so that all devices share a common reference. The GAIN input to the DAC is tied HIGH to configure the output range of the DAC to be 0 to 2xVref.

Figure 3-11 shows the pinout of the ADC interface connector (J1).

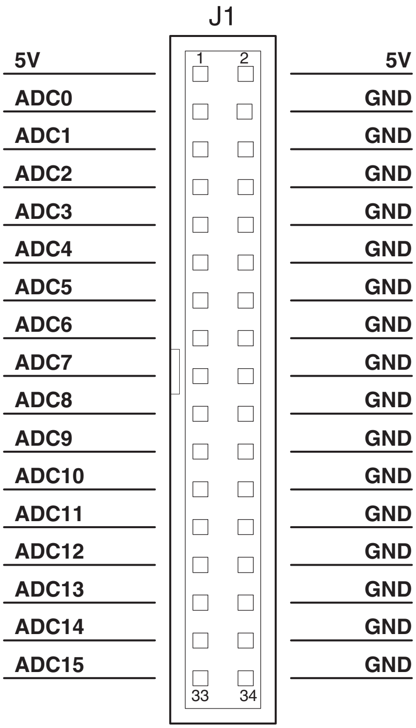


Figure 3-11 ADC connector pinout

Figure 3-12 shows the pinout of the DAC interface connector (J2).

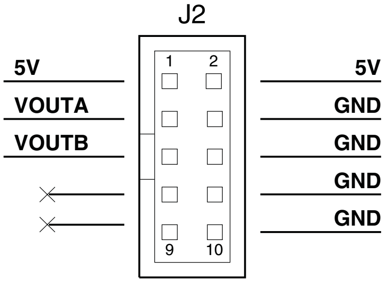


Figure 3-12 DAC connector pinout

Chapter 4
Reference Design Example

This chapter describes how to set up and start using the supplied example design. It contains the following sections:

4.1 About the design example

This chapter describes the reference design example supplied with the interface module. The interface module is not fitted with any programmable devices because it is intended to provide interfaces for peripherals instantiated into a logic module FPGA.

The interface module design example for the logic module is supplied in VHDL. Although the PrimeCell peripherals can be seen instantiated in the top level VHDL file `IMAD1fpga.vhd`, the HDL source code for the PrimeCell peripherals themselves are not supplied. All other non-PrimeCell HDL source code is provided on the CD.

The design example supports AHB-based designs for Integrator/LM-XCV600E+ and LM-EP20K600E+ logic modules.

4.1.1 About PrimeCells

The ARM PrimeCell peripherals are a range of synthesizable peripherals that are ideally suited for use in ARM-based designs. The interface module is supplied with PrimeCell peripherals for some of the interfaces on the board and the accompanying CD contains documentation for them.

4.1.2 Example architecture

The architecture of the example is shown in Figure 4-1 on page 4-3. The design example contains the following peripherals:

- PrimeCell:
    - UART
    - *Synchronous Serial Port* (SSP)
    - DC-DC converter
    - *Vectored Interrupt Controller* (VIC).
- Non-PrimeCell:
    - APB control registers
    - GPIO
    - Stepper motor interface
    - CAN controller interface
    - ADC and DAC interface
    - *Peripheral Information Block* (PIB).

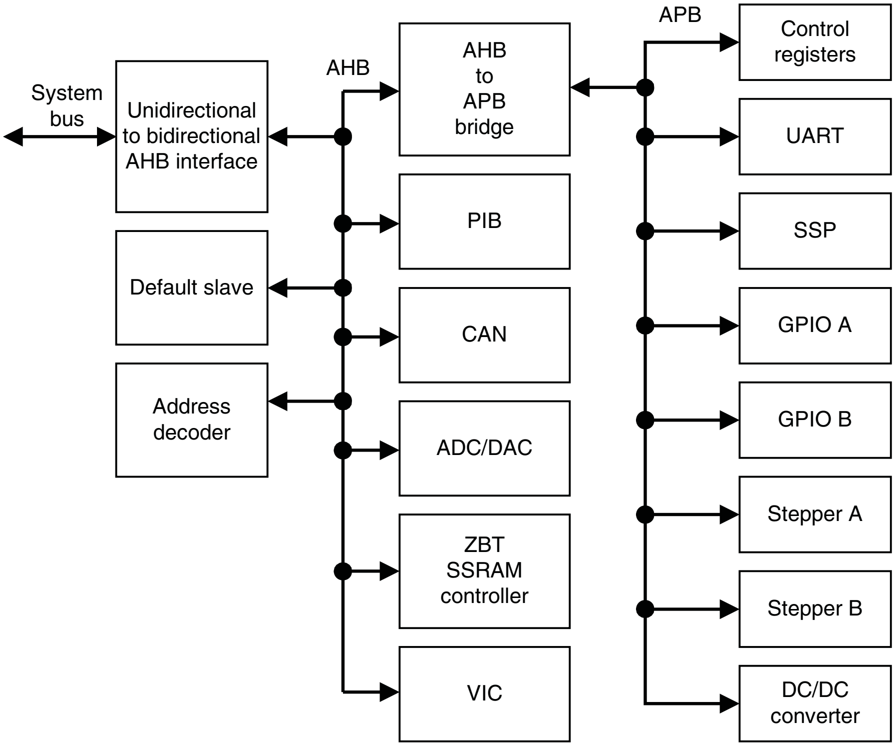


Figure 4-1 Design example architecture

Table 4-1 provides a summary description of the supplied VHDL files. A more detailed description of each VHDL block is included within the files in the form of comments.

Table 4-1 VHDL file descriptions

File	Description
`IMAD1fpga`	This file is the top-level VHDL that instantiates all of the interface for the example. The VHDL for the PrimeCell interfaces are not supplied but are available from ARM as separate products.
`AHBDecoder`	The decoder provides the AHB peripherals with select line generated from the address lines and the module ID (position in stack) signals from the motherboard. The Integrator family of boards uses a distributed address decoding system (see *Address assignment of logic modules* on page 4-5).
`AHBDefaultSlave`	This block provides a default slave response when the logic module address space is addressed but the address does not correspond to any of the instantiated peripherals.

Table 4-1 VHDL file descriptions (continued)

File	Description
`AHBMux7S1M`	This is the AHB multiplexor that connects the read data buses and the **HRESP** and **HREADY** signals from all of the slaves to the AHB master.
`AHBZBTRAM`	An SSRAM controller block to support word, halfword, and byte operations to the SSRAM on the logic module.
`AHB2APB`	This is the bridge block required to connect APB peripherals to the high-speed AMBA AHB bus. It produces the peripheral select signals for each of the APB peripherals.
`APBRegs`	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 • produce an interrupt for the LM push button.
`AHBPIB`	This provides the ROM block that gives the following information about each peripheral: • base address • PrimeCell number • peripheral revision number • FPGA revision number.
`BuildOptions`	This file defines generation of the PrimeCells in the example and allows control over the synthesis so that PrimeCells can be included or excluded. It specifies the base address of all the peripherals.
`GPIO32`	This is a simple 32-bit GPIO block.
`SPICSReg`	This is a simple register that is used to generate the chip selects for the SPI bus.
`AHB2CAN`	This is the interface between the AHB and Bosch CC770 CAN devices.
`AHB2AD7859`	This is the interface between the AHB and the ADC and DAC converters.
`Stepper`	This is the stepper motor controller logic, which has an APB interface.

4.1.3 Example memory map

The supplied examples set up the memory map for the logic module as shown in Figure 4-2 on page 4-5. This shows the locations to which logic modules are assigned by the main address decoder on an Integrator/AP motherboard when the logic modules are fitted in the EXPA/EXPB connector positions. The diagram also shows how the example decodes the address space for the logic module when it is LM0 (bottom of the stack).

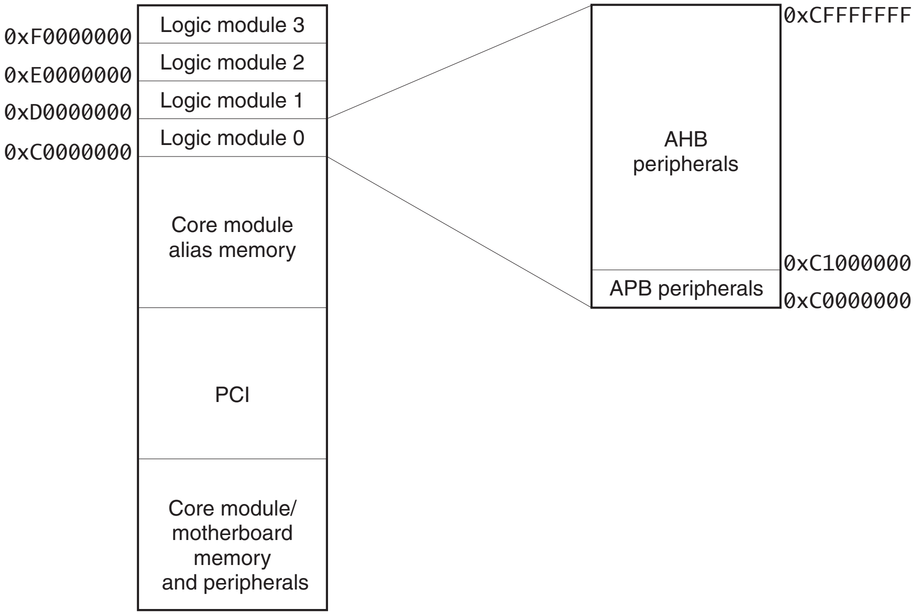


Figure 4-2 Integrator system memory map

Note

The Integrator system implements a distributed address decoding scheme in which each core or logic module is responsible for decoding its own address space. It is important when implementing a logic module design, to ensure that the module responds to all memory accesses in the appropriate memory region (see the user guide for your motherboard).

4.1.4 Address assignment of logic modules

The Integrator motherboards can have more than one logic module mounted on them. The base address of each logic module depends on its position in the stack and defines the value of bits [31:28] of the address for all devices on the logic module. Table 4-2 on page 4-6 shows the values of address bits [31:28] on logic modules fitted to an Integrator/AP in the EXPA/EXPB connector position (see the *Integrator/AP User Guide* for more information).

Table 4-2 Logic module addresses

Position in stack	Bits 31:28
0 (bottom)	`0xC`
1	`0xD`
2	`0xE`
3 (top)	`0xF`

4.1.5 Integrator/IM-AD1 memory map

The memory model for the design is shown in Table 4-2 and assumes that the logic module is mounted in position 0.

Table 4-3 Integrator/IM-AD1 memory map

Device	Address
logic module APB registers	`0xC0000000`
UART0	`0xC0100000`
SPICS	`0xC0200000`
SSP	`0xC0300000`
Reserved	`0xC0400000`
Reserved	`0xC0500000`
Reserved	`0xC0600000`
Reserved	`0xC0700000`
Reserved	`0xC0800000`
Reserved	`0xC0900000`
DCDC	`0xC0A00000`
STEPPERA	`0xC0B00000`

Table 4-3 Integrator/IM-AD1 memory map (continued)

Device	Address
STEPPERB	`0xC0C00000`
GPIOA	`0xC0D00000`
GPIOB	`0xC0E00000`
Reserved	`0xC1000000`
SSRAM	`0xC2000000`
VIC	`0xC3000000`
CAN	`0xC4000000`
ADC/DAC	`0xC5000000`
PIB	`0xCFFFFF00`

4.2 Example APB register peripheral

Table 4-4 shows the mapping of the logic module registers. The addresses shown are offsets from the base addresses shown in Figure 4-2 on page 4-5.

Table 4-4 Logic module registers

Offset address	Name	Type	Function
`0x0000000`	LM_OSC1	Read/write	Oscillator 1 divisor register
`0x0000004`	LM_OSC2	Read/write	Oscillator 2 divisor register
`0x0000008`	LM_LOCK	Read/write	Oscillator lock register
`0x000000C`	LM_LEDS	Read/write	User LEDs control register
`0x0000010`	LM_INT	Read/write	Push button interrupt register
`0x0000014`	LM_SW	Read	Switches register

4.2.1 Oscillator divisor registers

The oscillator registers control the frequency of the clocks generated by the two clock generators on the logic module.

Before writing to the oscillator registers, you must unlock them by writing the value `0x0000A05F` to the LM_LOCK register. After writing the oscillator register, relock them by writing any value other than `0x0000A05F` to the LM_LOCK register.

The reference divider (R[6:0]) and VCO divider (V[8:0]) are used to calculate the output frequency as follows:

$$\text{Frequency} = 48\text{MHz} \cdot \frac{(V[8:0] + 8)}{(R[6:0] + 2) \cdot OD}$$

Table 4-5 on page 4-10 describes the oscillator register bits.

Note

You can calculate values for the clock control signals using the ICS525 calculator on the Integrated Circuit Systems web site at:

`http://www.icst.com/`

You must also observe the operating range limits:

$$10\text{MHz} < 48\text{MHz} \cdot \frac{(V[8:0] + 8)}{(R[6:0] + 2)}$$

$$R[6:0] < 118$$

Table 4-5 LM_OSCx registers

Bits	Name	Access	Function
18:16	OD	Read/write	Output divider: 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 9 111 = divide by 6.
15:9	RDW	Read/write	Reference divider word. Defines the binary value of the R[6:0] pins of the clock generator.
8:0	VDW	Read/write	VCO divider word. Defines the binary value of the V[8:0] pins of the clock generator.

Note

The default values for these registers set **CLK1** to 25MHz and **CLK2** to 12MHz.

4.2.2 Oscillator lock register

The lock register is used to control access to the oscillator registers, allowing them to be locked and unlocked. This mechanism prevents the oscillator registers from being overwritten accidently. Table 4-6 describes the lock register bits.

Table 4-6 LM_LOCK register

Bits	Name	Access	Function
16	LOCKED	Read	This bit indicates if the oscillator registers are 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 oscillator registers. Write any other value to this register to lock the oscillator registers.

4.2.3 User LEDs control register

The LEDs register is a 9-bit register used to control the nine user LEDs on the logic module. Writing a 0 to a bit lights the associated LED.

4.2.4 Push button interrupt register

The push button interrupt register contains 1 bit. It is a latched indication that the push button on the logic module has been pressed. The output from this register is used to drive an input to the interrupt controller. Table 4-7 describes the operation of this register.

Table 4-7 LM_INT register

Bits	Name	Access	Function
0	LM_INT	Read	This bit when SET is a latched indication that the push button has been pressed.
		Write	Write 0 to this register to CLEAR the latched indication. Writing 1 to this register has the same effect as pressing the push button.

4.2.5 Switches register

This register is used to read the setting of the 8-way DIP switch on the logic module. A 0 indicates that the associated switch element is closed (ON).

4.3 UART

The UART used in the design example is the PrimeCell PL011. Refer to the *ARM PrimeCell UART (PL011) Technical Reference Manual* for more information.

The UART is clocked by the signal **CLK2** from the logic module. **CLK2** is set to 12MHz by default.

4.4 SPI chip select register

This is a 3-bit read/write register that controls the three chip select signals on the connectors J11 and J13. Writing a 1 causes the associated SPI chip select signal to go LOW.

Table 4-8 SPI chip select register bit assignment

Bit	Name	Access	Function
2	SPICS2	Read/write	0 = **SPI_nCS2** is HIGH 1 = **SPI_nCS2** is LOW
1	SPICS1	Read/write	0 = **SPI_nCS1** is HIGH 1 = **SPI_nCS1** is LOW
0	SPICS0	Read/write	0 = **SPI_nCS0** is HIGH 1 = **SPI_nCS0** is LOW

4.5 Synchronous serial port

The synchronous serial port PrimeCell is used to implement the SPI interface. Refer to the *ARM PrimeCell Synchronous Serial Port Master and Slave (PL022) Technical Reference Manual* for information about this device.

The SSP is clocked by the **CLK1** signal from the logic module. This clock is set to 25MHz by default.

4.6 PWM controller

The PWM control function is implemented by the DC-DC converter PrimeCell (PL160). Refer to the ARM *PrimeCell DC-DC Converter Interface (PL160) Technical Reference Manual* for information about this device.

The DC-DC PrimeCell uses the 4MHz **IM_CLK** signal to supply the **DCDCCLK** reference clock. It can divide this by 16, 32, 128, or 304 to provide four possible switching frequencies of 250kHz, 125kHz, 31.25kHz, and 13.158kHz.

The switching circuitry has turn on and turn off delays that limit the minimum pulse width and can also affect the accuracy of the PWM, particularly at the higher switching frequencies between 125kHz to 250kHz. The turn on and turn off delays for different voltages are shown in Table 4-9.

Table 4-9 PWM turn on and turn off delays

Load voltage	Turn on time	Turn off time
5V	1µs	0.6µs
30V	2µs	1.2µs

4.7 Stepper motor peripheral

The example design instantiates two stepper controller blocks, each of which has two stepper motor controllers. Stepper A controls the Step 1 and 2 interfaces which are connected to the L298 stepper motor drivers. Stepper B controls the Step 3 and 4 interfaces which are connected at logic level to the connectors J21 and J22.

Each controller contains three registers, as shown in Table 4-10. These define phase sequence generation, speed, and number of steps to rotate. The stepper motor controller is clocked from **PCLK** to keep it synchronous with the APB. The step speed register is fed with a 10kHz clock (which is divided down from the 4MHz **IM_CLK** signal) to control the speed of the motor.

Table 4-10 Stepper motor registers

Offset address	Name	Access	Function
`0x0B00000`	STEP1CONT	Read/write	Stepper 1 control register
`0x0B00004`	STEP1COUNT	Read/write	Stepper 1 step count register
`0x0B00008`	STEP1SPEED	Read/write	Stepper 1 Clock divider register
`0x0B00010`	STEP2CONT	Read/write	Stepper 2 controller register
`0x0B00014`	STEP2COUNT	Read/write	Stepper 2 step count register
`0x0B00018`	STEP2SPEED	Read/write	Stepper 2 Clock divider register
`0x0C00000`	STEP3CONT	Read/write	Stepper 3 control register
`0x0C00004`	STEP3COUNT	Read/write	Stepper 3 step count register
`0x0C00008`	STEP3SPEED	Read/write	Stepper 3 Clock divider register
`0x0C00010`	STEP4CONT	Read/write	Stepper 4 controller register
`0x0C00014`	STEP4COUNT	Read/write	Stepper 4 step count register
`0x0C00018`	STEP4SPEED	Read/write	Stepper 4 Clock divider register

Each of the registers is double buffered, allowing a new value to be written to a holding register while a previous count continues. Write to the STEPxCOUNT and STEPxSPEED register locations first and then follow this with a write to the STEPxCONT register. The controller loads the new values into the target registers when the current count completes.

4.7.1 Stepper x control register

The stepper controller control register defines the operating mode of the stepper.

Note

You must consider the maximum speed of the stepper motor when programming the step speed register or issuing consecutive single step commands in the stepper control register.

Ensure there is a delay, typically a few milliseconds, between a DOCOUNT and a SINGLESTEP. Failing to leave a delay between the end of the count and writing a SINGLESTEP command results in unpredictable behavior. The minimum duration of the delay depends on the stepper motor.

The bits in this register are described in Table 4-11.

Table 4-11 Stepper control register

Bits	Name	Access	Function
7	BUSY	Read	This bit contains 1 when a count is in progress and 0 when the count is complete.
6	BUFFERFULL	Read	This bit contains 1 when the buffer is full and 0 when the buffer is available for a new value to be written. An inverted version of this bit is used as an interrupt source.
5	DRIVE ENABLE	Read/write	This bit enables and disables the phase outputs to the motor. When this bit is 0, **ENA**, **ENB**, **PH1**, **PH2**, **PH3**, and **PH4** are held at 0. When this bit is 1, these signals output the relevant drive waveform.
4	HALFSTEP	Read/write	These bits are used in combination to select the drive mode: 00 = Full step, two phase on (non-wave drive) drive sequence (see Figure 4-3 on page 4-19) 01 = Full step, one phase on (wave drive) drive sequence (see Figure 4-4 on page 4-19) 10 = Half step drive sequence (see Figure 4-5 on page 4-20) 11 = Reserved
3	WAVEDRIVE	Read/write	

Table 4-11 Stepper control register (continued)

Bits	Name	Access	Function
2	DOCOUNT	Read/write	Write a 1 to this bit to transfer the contents of the buffer register to the count and speed registers. This causes the corresponding number of steps to be performed.
1	SINGLESTEP	Read/write	Write a 1 to this bit to advance the stepper motor by one step. The step speed register, step count register, and bit 2 are ignored.
0	DIR	Read/write	This bit controls the direction of rotation. The actual direction of rotation (clockwise or anticlockwise) depends on how the motor is wired to the interface module.

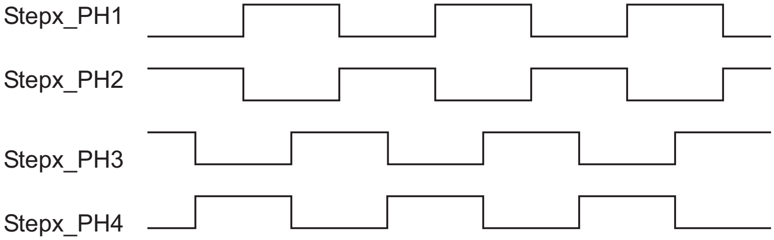


Figure 4-3 Full-step two-phase output waveforms

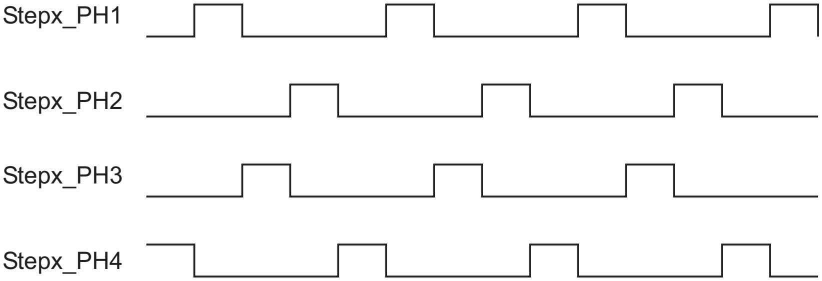


Figure 4-4 Full-step single-phase output waveforms

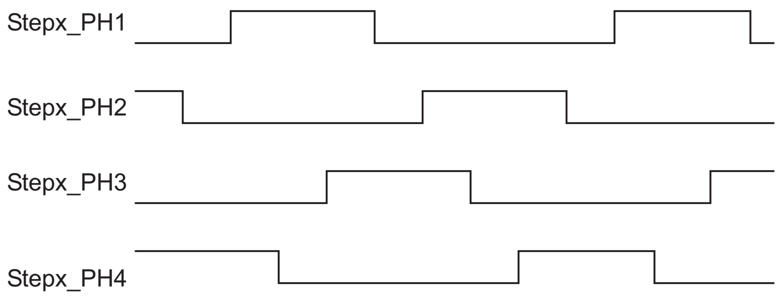


Figure 4-5 Half-step output waveforms

4.7.2 Stepx count register

This is a 9-bit register that is used to specify the number of steps to advance. When the required number of steps are complete, the count stops and the register is loaded with the next value.

4.7.3 Stepx speed register

This register contains a 14-bit value that is used to divide a 10kHz clock signal to regulate the speed of the stepper motor. That is, the step speed register defines the number of 0.1ms periods between steps.

4.8 GPIO

There are two 32-bit GPIO blocks instantiated in the example design. Each GPIO provides 32 general-purpose input and output signals that are connected to the connectors J16 and J17. GPIOB is also connected to the 38-way Mictor connector J7 for easy connection to a logic analyzer. The GPIO registers are shown in Table 4-12.

Table 4-12 GPIO registers

Address offset	Name	Access	Size	Function
`0x000000`	GPIO_DATASET	Write	32	Data output set
	GPIO_DATAIN	Read	32	Read data input pins
`0x000004`	GPIO_DATACLR	Write	32	Data register output clear
	GPIO_DATAOUT	Read	32	Read data output pins
`0x000008`	GPIO_DIRN	Read/write	32	Data direction

4.8.1 Data output set register

The GPIO_DATASET location is used to set individual GPIO output bits as follows:

1 = SET the associated GPIO output bit

0 = leave the associated GPIO bit unchanged.

4.8.2 Read data input register

Read the current state of the GPIO input bits from this location.

4.8.3 Data register output clear

The GPIO_DATACLR location is used to clear individual GPIO output bits as follows:

1 = CLEAR the associated GPIO output bit

0 = leave the associated GPIO bit unchanged.

4.8.4 Read data output pins

Read the current state of the GPIO output bits from this location.

4.8.5 Data direction

The GPIO_DIRN location is used to set the direction of each GPIO pin as follows:

1 = pin is an output

0 = pin is an input (default).

Figure 4-6 shows the data direction control for one GPIO bit.

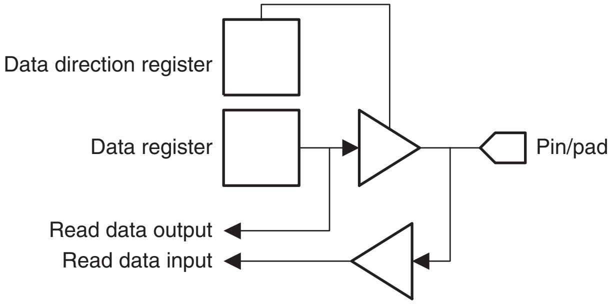


Figure 4-6 GPIO direction control (1 bit)

4.9 SSRAM interface

The SSRAM interface provides read and write access to the 1MB ZBT SSRAM on the logic module. Accesses take two system clock cycles for reads and writes. The interface supports word, halfword, and byte accesses to the SSRAM.

4.10 Vectored interrupt controller

The interrupt controller used in the example design is the V*ectored Interrupt Controller* (VIC) PrimeCell (PL190). Refer to the *ARM PrimeCell Vectored Interrupt Controller (PL190) Technical Reference Manual* for information about this device.

The assignment of interrupt sources to the VIC are shown in Table 4-13.

Table 4-13 Interrupt assignment

Interrupt source number	Interrupt source
0	LM_INT APB register
1	UART0
2	Reserved
3	SSP
4	STEP1 buffer empty
5	STEP2 buffer empty
6	STEP3 buffer empty
7	STEP4 buffer empty
8	CAN1
9	CAN2
10	ADC1 conversion complete
11	ADC2 conversion complete

4.10.1 Interrupt sources

The LM_INT interrupt comes from the push button interrupt register in the APB register block. The interrupt is latched if the push button on the logic module is pressed or if a 1 is written to the push button interrupt register. The interrupt is cleared by writing a 0 to the push button interrupt register.

The UART interrupt is the combined interrupt from the UART PrimeCell. Refer to *ARM PrimeCell UART (PL011) Technical Reference Manual* for details of the interrupt sources.

The SSP interrupt is the combined interrupt from the SSP PrimeCell. Refer to *ARM PrimeCell Synchronous Serial Port (PL022) Technical Reference Manual* for details of the interrupt sources.

The STEP1, STEP2, STEP3, and STEP4 interrupts are set active when the buffer registers of the corresponding stepper motor controller are empty. This indicates that a new step instruction can be written. The interrupts are cleared if the stepper controller buffer registers are holding a step instruction that is waiting to be carried out.

The CAN1 and CAN2 interrupts are interrupt signals from the CAN controller chips. The interrupt signals are a combination of interrupts from different sources within the CAN controller. Refer to the data sheet for the Bosch CC770 for details of the interrupt sources.

The ADC1 and ADC2 interrupts are generated from the **BUSY** signal of the corresponding AD7859 A/D converter chip. The ADC1 and ADC2 interrupts signal that the ADC has finished its conversion and the value can be read. The interrupt is set active when the BUSY signal falls at the end of a conversion. The interrupt is cleared by any read access to the ADC.

Note

The **BUSY** signal goes active during the power-on calibration of the ADC chips. That is, the ADC1 and ADC2 interrupts are set after power-on and the interrupts must be cleared by doing a dummy read access to the ADCs.

4.11 CAN controller interface

The CAN controller interface gives you access to the internal registers and reset signals of the Bosch CC770 CAN controllers. The offset addresses of CAN controller interfaces are shown in Table 4-14.

Table 4-14 CAN controller interface registers

Offset address	Name	Function
`0x000000`	CAN1Base	Interface to CAN1 controller registers
`0x100000`	CAN2Base	Interface to CAN2 controller registers
`0x200000`	CANRESET	CAN reset control register

4.11.1 CANxBase

You use the register interface locations to read and write the CAN registers. Register accesses take at least six system bus clock cycles and can be stretched by the CAN controller to a maximum of 550ns plus three system clock cycles.

The address pins **CAN_A[7:0]** of the CAN controllers are connected to **HADDR[9:2]**. This means that individual CAN registers are located on word boundaries starting from the base address of the device.

4.11.2 CAN reset control register

The CAN reset register controls the **nRESET** signals to the CAN controllers. The assignment of the bits in the register is shown in Table 4-15.

Table 4-15 CAN reset register bit assignment

Bit	Name	Access	Function
1	CAN2nRESET	Read/write	Controls the **nRESET** signal to CAN2.
0	CAN1nRESET	Read/write	Controls the **nRESET** signal to CAN1.

The CAN controllers are reset by writing a 0 to the associated bit so the **nRESET** signal goes LOW. The default setting of this register after power up is 0, so you must write a 1 before you can read and write the internal registers of the CAN controllers. However, after power up the CAN resets must be held LOW for at least 1ms.

4.12 ADC and DAC interface

This interface gives you access to the ADCs and DAC. The interface also contains a status and control register. The offset addresses of the ADC and DAC interface are shown in Table 4-16.

Table 4-16 ADC and DAC interface registers

Offset address	Name	Function
`0x000000`	ADCSTATUS	This register enables you to monitor the status of the ADC busy signals
`0x000004`	DACnCLR	This register controls the **nCLR** signal to the DAC.
`0x100000`	ADC1Base	Interface to ADC1
`0x200000`	ADC2Base	Interface to ADC2
`0x300000`	DACBase	Interface to the DAC

The ADCs each appear as one 16-bit location at the corresponding base address. The DAC appears as two locations at DACBase and DACBase+4 that correspond to the DAC A and B channels respectively. Refer to the AD7859 and AD5342 data sheets for details of ADC and DAC operations.

Accesses to these devices take four system bus clock cycles, although consecutive accesses incur an additional three wait states for the second and subsequent access. The DAC has the signal LDAC tied LOW. This means that a value is passed to the DAC as soon as it is written.

The ADC status register provides you with read-only access to the ADC busy signals. The bit assignment is shown in Table 4-17.

Table 4-17 ADC status register bit assignment

Bit	Name	Access	Function
1	ADC2BUSY	Read	Gives value of ADC2 busy signal
0	ADC1BUSY	Read	Gives value of ADC1 busy signal

The DACnCLR register provides you with read/write access to control the signal **nCLR** routed to the DAC. Write 0 to this register to reset the DAC value to 0. You must write a 1 to enable normal operation of the DAC.

4.13 Peripheral information block

The *Peripheral Information Block* (PIB) is a block of 32 words in ROM that provides you with information about the peripherals used in the design. The PIB is located at the top of the address space for the logic module.

Each word in the PIB provides information about one peripheral. A value of `0x00000000` indicates that there is no entry and that the next address must be checked. Each valid entry contains the information shown in Table 4-18.

Table 4-18 PIB entry format

Bits	Name	Function
31:24	Peripheral Base	Bits [27:20] of the peripheral base address. Bits [31:28] of the address are defined by the location of the logic module in the stack see *Address assignment of logic modules* on page 4-5.
23:8	Peripheral ID	For a PrimeCell, this is the PrimeCell number in BCD. For example, the VIC PrimeCell PL190 would be represented by `0x0190`. For other peripherals the value `0xFFnn` is assigned, where `nn` is a unique look-up value. (See the AHBPIB HDL source file for details.) The value `0xFFFF` is a special case that is used to indicate the FPGA build number.
7:0	Peripheral Rev	This gives the revision number of the peripheral in BCD. For example, revision v1.2 is represented by `0x12`.

The last address of the PIB is used to store the FPGA build number. Bits [31:8] are all 1 and bits [7:0] store the revision number in BCD.

Note

Use the ARM executable utility `read_pib.axf`, supplied on the IM-AD1 CD, to display the PIB information.

Appendix A
Signal Descriptions

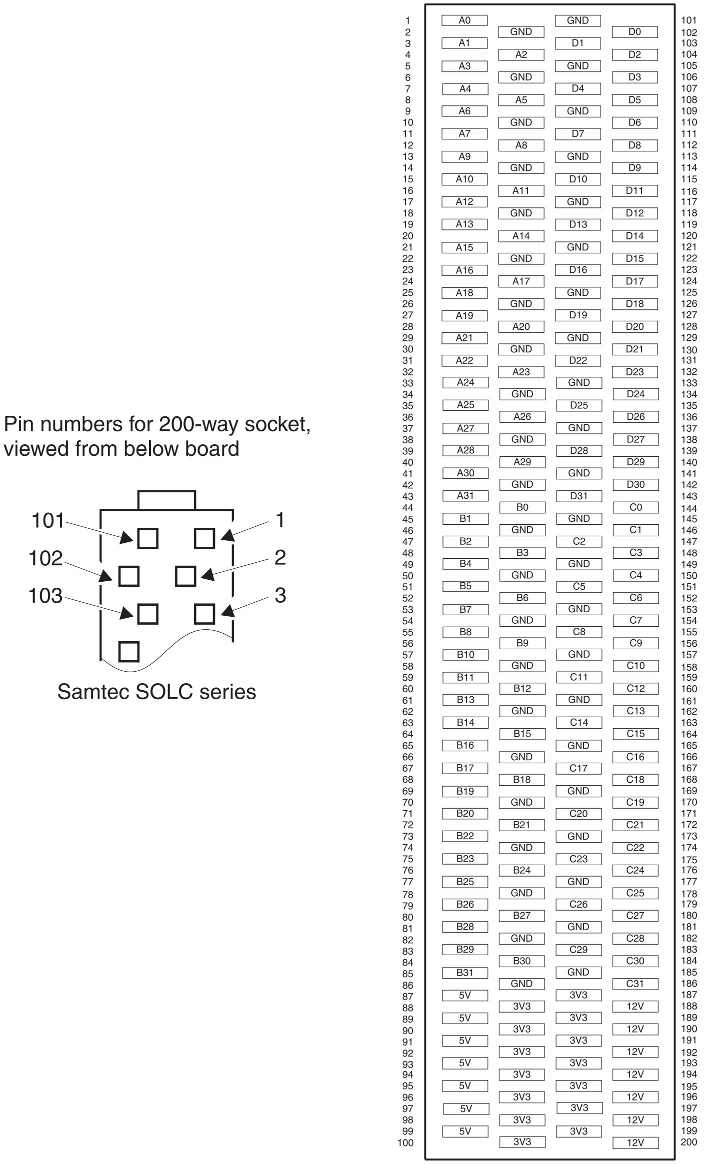
This appendix describes the Integrator/IM-AD1 interface connectors and signal connections. It contains the following sections:

Note

For details of the I/O connectors, see Chapter 3 *Hardware Reference*.

A.1 EXPA

Figure A-1 shows the pin numbers of the EXPA socket. The socket is viewed as if looking down through the stack.



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

Figure A-1 EXPA socket pin numbering

The signals present on the EXPA connector are described in Table A-1.

Table A-1 AHB signal assignment

Pin label	Signal	Description
A[31:0]	Not used	-
B[31:0]	**B[31:0]**	These signals connect to the FPGA on the logic module. They are used to carry the **GIPOB[31:0]** signals.
C[31:0]	Not used	-
D[31:0]	Not used	-

A.2 EXPB

Figure A-2 shows the pin numbers of the socket EXPB on the underside of the interface module.

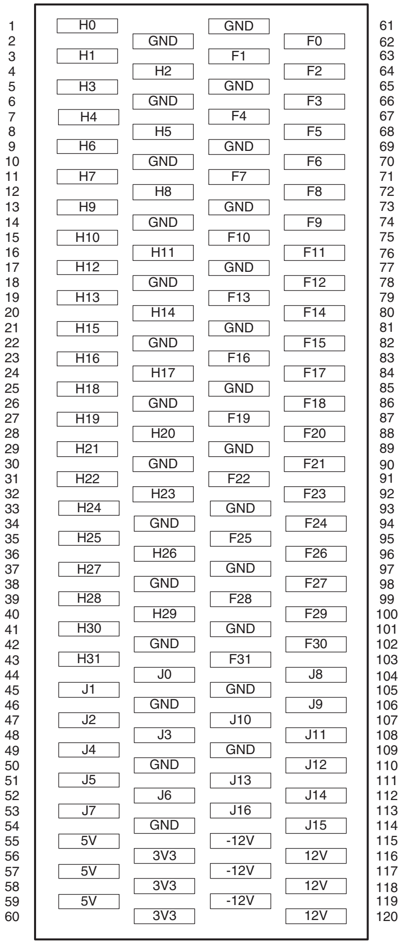


Figure A-2 EXPB socket pin numbering

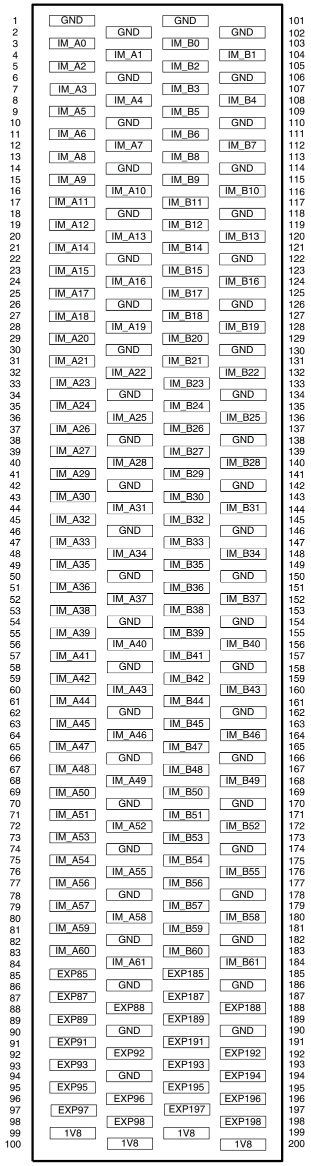
Table A-2 describes the signals on the pins labeled F[31:0], H[31:0], and J[16:0].

Table A-2 EXPB signal description

Pin label	Name	Description
F[31:24]	Not used	-
F[23:0]	**F[23:0]**	Stepper motor controller signals.
H[31:29]	Not used	-
H28	**SYSCLK**	System clock from the logic module
H[27:0]	Not used	-
J[15:14]	Not used	-
J13	**nCFGEN**	Sets motherboard into configuration mode
J12	**nSRST**	Multi-ICE reset (open collector)
J11	Not used	-
J10	**RTCK**	Returned JTAG test clock
J9	Not used	-
J8	**nTRST**	JTAG reset
J7	**TDO**	JTAG test data out
J6	**TDI**	JTAG test data in
J5	**TMS**	JTAG test mode select
J4	**TCK**	JTAG test clock
J[3:0]	Not used	-

A.3 EXPIM

This connector is the same type of as that used for EXPA. Figure A-3 shows the pin numbers for EXPIM.



1	GND	101	GND
2	GND	102	GND
3	IM_A0	103	IM_B0
4	IM_A1	104	IM_B1
5	IM_A2	105	IM_B2
6	GND	106	GND
7	IM_A3	107	IM_B3
8	IM_A4	108	IM_B4
9	IM_A5	109	IM_B5
10	GND	110	GND
11	IM_A6	111	IM_B6
12	IM_A7	112	IM_B7
13	IM_A8	113	IM_B8
14	GND	114	GND
15	IM_A9	115	IM_B9
16	IM_A10	116	IM_B10
17	IM_A11	117	IM_B11
18	GND	118	GND
19	IM_A12	119	IM_B12
20	IM_A13	120	IM_B13
21	IM_A14	121	IM_B14
22	GND	122	GND
23	IM_A15	123	IM_B15
24	IM_A16	124	IM_B16
25	IM_A17	125	IM_B17
26	GND	126	GND
27	IM_A18	127	IM_B18
28	IM_A19	128	IM_B19
29	IM_A20	129	IM_B20
30	GND	130	GND
31	IM_A21	131	IM_B21
32	IM_A22	132	IM_B22
33	IM_A23	133	IM_B23
34	GND	134	GND
35	IM_A24	135	IM_B24
36	IM_A25	136	IM_B25
37	IM_A26	137	IM_B26
38	GND	138	GND
39	IM_A27	139	IM_B27
40	IM_A28	140	IM_B28
41	IM_A29	141	IM_B29
42	GND	142	GND
43	IM_A30	143	IM_B30
44	IM_A31	144	IM_B31
45	IM_A32	145	IM_B32
46	GND	146	GND
47	IM_A33	147	IM_B33
48	IM_A34	148	IM_B34
49	IM_A35	149	IM_B35
50	GND	150	GND
51	IM_A36	151	IM_B36
52	IM_A37	152	IM_B37
53	IM_A38	153	IM_B38
54	GND	154	GND
55	IM_A39	155	IM_B39
56	IM_A40	156	IM_B40
57	IM_A41	157	IM_B41
58	GND	158	GND
59	IM_A42	159	IM_B42
60	IM_A43	160	IM_B43
61	IM_A44	161	IM_B44
62	GND	162	GND
63	IM_A45	163	IM_B45
64	IM_A46	164	IM_B46
65	IM_A47	165	IM_B47
66	GND	166	GND
67	IM_A48	167	IM_B48
68	IM_A49	168	IM_B49
69	IM_A50	169	IM_B50
70	GND	170	GND
71	IM_A51	171	IM_B51
72	IM_A52	172	IM_B52
73	IM_A53	173	IM_B53
74	GND	174	GND
75	IM_A54	175	IM_B54
76	IM_A55	176	IM_B55
77	IM_A56	177	IM_B56
78	GND	178	GND
79	IM_A57	179	IM_B57
80	IM_A58	180	IM_B58
81	IM_A59	181	IM_B59
82	GND	182	GND
83	IM_A60	183	IM_B60
84	IM_A61	184	IM_B61
85	EXP85	185	EXP185
86	GND	186	GND
87	EXP87	187	EXP187
88	EXP88	188	EXP188
89	EXP89	189	EXP189
90	GND	190	GND
91	EXP91	191	EXP191
92	EXP92	192	EXP192
93	EXP93	193	EXP193
94	GND	194	EXP194
95	EXP95	195	EXP195
96	EXP96	196	EXP196
97	EXP97	197	EXP197
98	EXP98	198	EXP198
99	1V8	199	1V8
100	1V8	200	1V8

Figure A-3 EXPIM connectors pin numbering

Table A-3 shows the signals for the interface module for Integrator/LM-XCV2000E or LM-EP20K1000E logic module types.

Table A-3 EXPIM signal descriptions

Label	LM-XCV2000E	LM-EP20K1000E	Description
IM_ABANK[59:0]	**IM_0BANK[59:0]**	**IM_5BANK[59:0]**	FPGA input/output pins.
IM_BBANK[53:0]	**IM_1BANK[53:0]**	**IM_6BANK[53:0]**	FPGA input/output pins.
EXP[92:85]	**Not used**	Not used	-
EXP93	**IM_CLK**	**IM_CLK**	Clock signal from IM-AD1 to the logic module FPGA.
EXP[96:94]	**Not used**	Not used	-
EXP97	**VCCO_0**	**VCCO_5**	Configurable voltage power supply rail. Not used (socket).
EXP98	**VCCO_0**	**VCCO_5**	Configurable voltage power supply rail. Not used (socket).
EXP185	Not used	Not used	-
EXP[189:187]	Not used	Not used	-
EXP191	**CLK1_1**	**CLK1_1**	Clock signal from the **CLK1** buffer on the logic module
EXP194	**GND**	**GND**	Ground
EXP[196:192]	Not used	**Not used**	-
EXP197	**VCCO_1**	**VCCO_6**	Configurable voltage power supply rail. Not used (socket).
EXP198	**VCCO_1**	**VCCO_6**	Configurable voltage power supply rail Not used (socket).

Caution

For correct operation of the interface module, VCCO_A and VCCO_B must be set to 3.3V. Ensure that the VCCO links are set correctly on the logic module.

A.4 Logic analyzer connector

A Mictor-type logic analyzer connector is provided. It connects to the B[31:0] signals used for GPIO B. If particular signals must be connected to a logic analyzer, the FPGA configuration can be changed to reassign the signal connections.

Caution

If the FPGA configuration is changed to reassign signal connections, the GPIO B connections on connector J16 also change.

Figure A-4 shows the pin numbers of this type of connector.

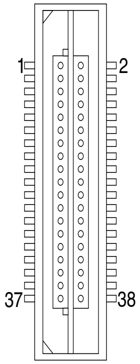


Figure A-4 J7 pin locations

Table A-4 shows the pinout of the logic analyzer connector.

Table A-4 J7 connector pinout

Signal	Pin	Pin	Signal
No connect	1	2	No connect
GND	3	4	No connect
SYSCLK	5	6	CLK_1
B31	7	8	**B15**
B30	9	10	**B14**
B29	11	12	**B13**
B28	13	14	**B12**
B27	15	16	**B11**
B26	17	18	**B10**
B25	19	20	**B9**
B24	21	22	**B8**
B23	23	24	**B7**
B22	25	26	**B6**
B21	27	28	**B5**
B20	29	30	**B4**
B19	31	32	**B3**
B18	33	34	**B2**
B17	35	36	**B1**
B16	37	38	**B0**

A.5 Multi-ICE (JTAG)

Figure A-5 shows the pinout of the Multi-ICE connector J21. For a description of the JTAG signals, see the user guide for your logic module.

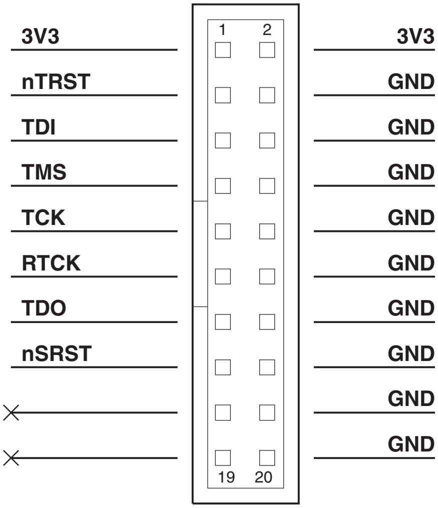


Figure A-5 Multi-ICE connector pinout

Appendix B
Mechanical Specification

This appendix contains the mechanical specification for Integrator/IM-AD1. It contains the following section:

- *Mechanical information* on page B-2
- *Connector reference* on page B-4.

B.1 Mechanical information

Figure B-1 shows the dimensions for the connectors on the top side of the board. See Table B-1 on page B-4 for details on connector type, part numbers, and manufacturers.

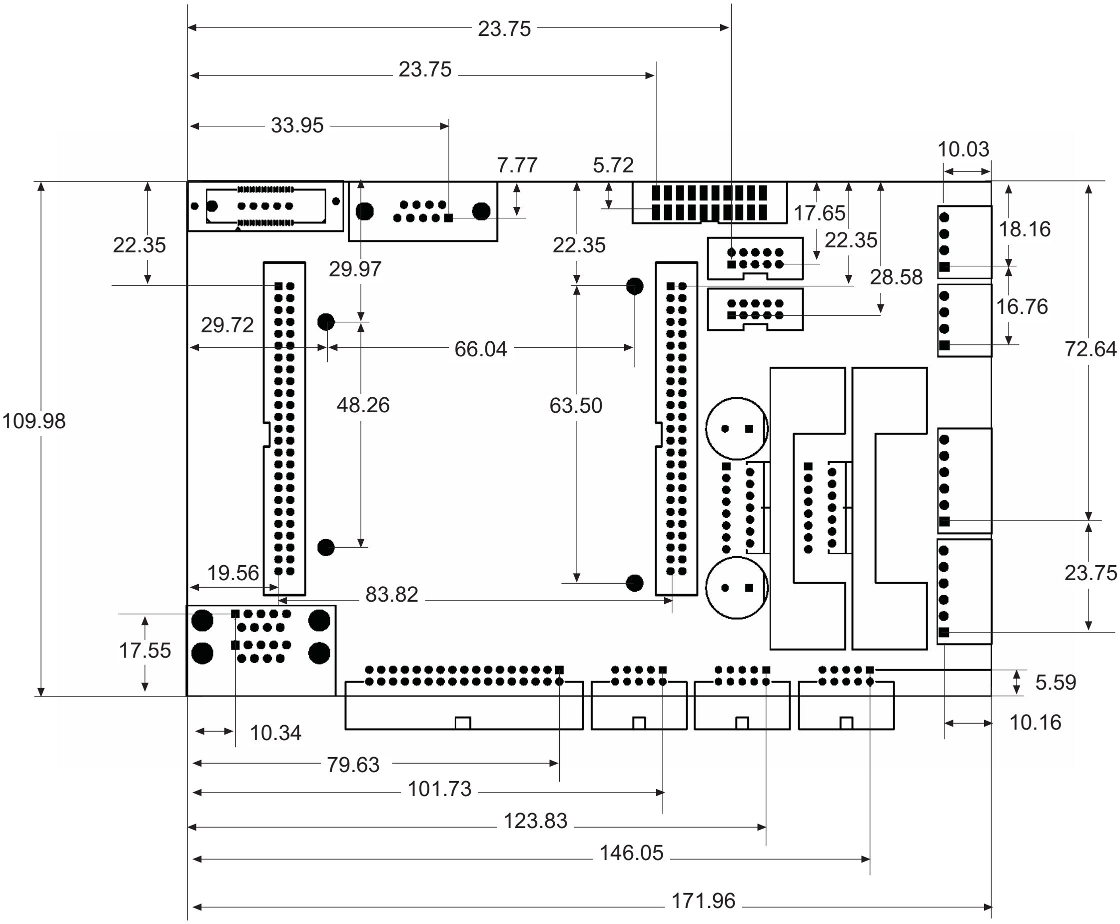


Figure B-1 Board dimensions (top view)

The Integrator/IM-AD1 is designed to be stackable (as the top card). Figure B-2 on page B-3 shows the dimensions for the connectors on the bottom side of the board as viewed from the top side of the board. These connectors carry the signals between the IM-AD1 and the logic module. (All dimensions are in mm.)

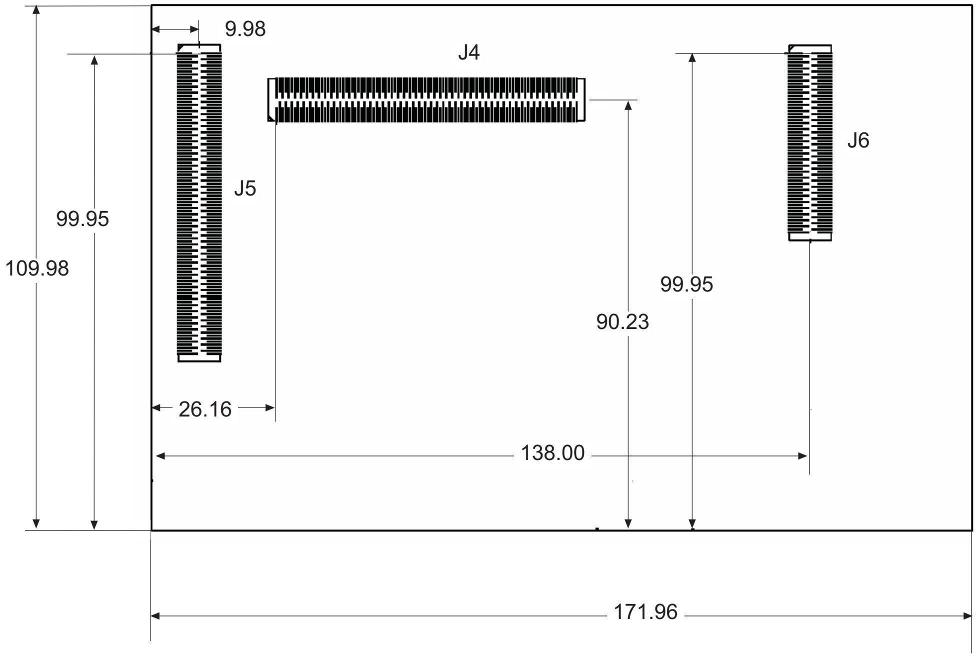


Figure B-2 Bottom board dimensions (viewed from top side)

B.2 Connector reference

Table B-1 lists the connectors on the IM-AD1. Two Weidmuller BL3.5/6 SN OR plugs and two BL3.5/4 SN OR plugs are supplied in a separate plastic bag. These mate with J10, J14, J19, and J23.

Table B-1 connector reference

Reference	Type	Manufacturer	Part number
J1	CON34_0.1"_34W_RA	Toby	302-R-34-D1-S1
J2	CON10_0.1"_10W_RA	Toby	302-R-10-D1-S1
J11	CON10_0.1"_10W_RA	Toby	302-R-10-D1-S1
J13	CON10_0.1"_10W_RA	Toby	302-R-10-D1-S1
J3	DB9_DUAL	Toby	DMR-DP-09P/09P-G-15.87
J5	SOLC-150-02-F-Q_0.64mm pitch	Samtec	SOLC-150-02-F-Q-P-A
J4	SOLC-150-02-F-Q_0.64mm pitch	Samtec	SOLC-150-02-F-Q-P-A
J6	SOLC-130-02-F-Q_0.64mm pitch	Samtec	SOLC-130-02-F-Q-P-A
J7	Mictor_38 way	Agilent	2-767004-2
J8	2x1_pin_header_0.1"_2W_LINK	Toby	THS-2-S
J9	CON20AP_0.1"_20W_VERT	Toby	302-S-20-D1-S1
J10	CON4_3.5mm_4W_RA	Weidmuller	SL 3.5/4/90G 3.2 SN OR
J14	CON4_3.5mm_4W_RA	Weidmuller	SL 3.5/4/90G 3.2 SN OR
J16	CON50_0.1"_50W_VERT	Toby	302-S-50-D1-S1
J17	CON50_0.1"_50W_VERT	Toby	302-S-50-D1-S1
J18	DB9_STRAIGHT	FCI	D09P24A4GV00
J19	CON6_3.5mm_6W_RA	Weidmuller	SL 3.5/6/90G 3.2 SN OR
J23	CON6_3.5mm_6W_RA	Weidmuller	SL 3.5/6/90G 3.2 SN OR
J21	CON10_0.1"_10W_VERT	Toby	302-S-10-D1-S1
J22	CON10_0.1"_10W_VERT	Toby	302-S-10-D1-S1

Glossary

This glossary lists all the abbreviations used in the Integrator/IM-AD1 User Guide.

ADC	Analog to Digital Converter. A device that converts an analog signal into digital data.
AHB	Advanced High Performance Bus. The ARM open standard for high-performance on-chip buses.
APB	Advanced Peripheral Bus. The ARM open standard for lower-speed peripherals.
CAN	Controller Area Network.
DAC	Digital to Analog Converter. A device that converts digital data into analog level signals.
FPGA	Field Programmable Gate Array.
GPIO	General Purpose Input/Output.
JTAG	Joint Test Action Group. The committee which defined the IEEE test access port and boundary-scan standard.
Multi-ICE	Multi-ICE is a system for debugging embedded processor cores using a JTAG interface.
PIB	Peripheral Information Block.
SPI	Serial Protocol Interface.

SSP	Synchronous Serial Port.
UART	Universal Asynchronous Receiver/Transmitter.
USB	Universal Serial Bus.
VCO	Voltage Controlled Oscillator.
VIC	Vectored Interrupt Controller.
ZBT SSRAM	Zero Bus Turnaround Synchronous Static Random Access Memory.

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.