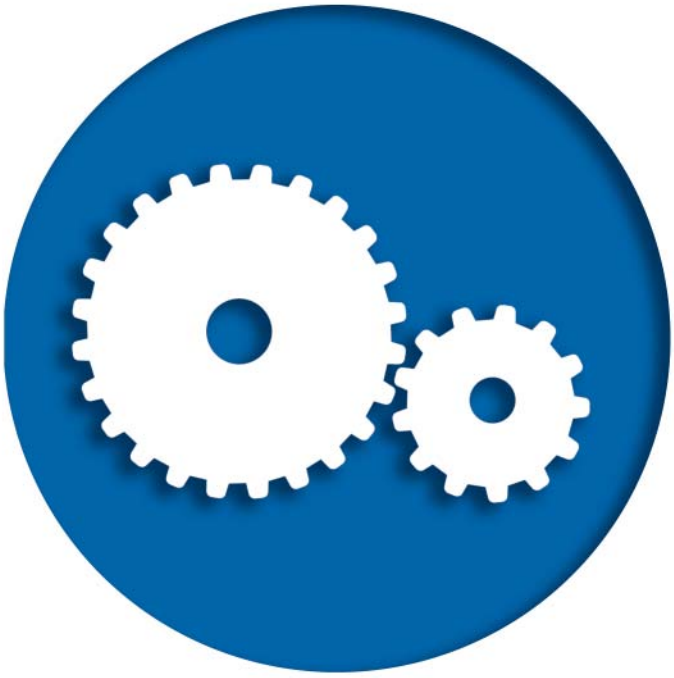
XMP Hardware Installation Guide



XMP Product Family

Helping you build a better machine, faster.™

Motion Engineering™

XMP Motion Controller
Hardware Installation Guide

March 2002
Doc # M001-0066 Rev. A
ECO 1465

Motion Engineering, Inc.
33 South La Patera Lane
Santa Barbara, CA 93117-3214
ph 805-681-3300
fax 805-681-3311
e-mail: technical@motioneng.com
website: http://www.motioneng.com
ftp site: ftp.motioneng.com
support website: http://support.motioneng.com

Motion Engineering, Inc.

XMP Motion Control Hardware Installation Manual Table of Contents

Chapter 1 XMP Hardware Overview

Chapter 2 XMP Bus Interface

Chapter 3 Configuring an XMP-analog Motion Control System

CHAPTER 1
XMP HARDWARE OVERVIEW

XMP Hardware

Because computer-based motion control systems vary widely, the XMP has been designed to accommodate most common industry hardware. This means providing a variety of platform, porting, form factor, and voltage options.

Platform Options

XMP-series motion controllers are ported to accommodate common industry software platforms, including:

- Windows95/98
- Windows2000
- WindowsNT
- LynxOS
- VxWorks
- VenturCom RTSS
- QNX
- PharLap

Form Factor Options

“Form factor” refers to the physical shape and size of an XMP controller and the component(s) it is designed to function with. XMP controllers are available in the following form factors:

- PCI
- CPCI-3U
- CPCI-6U
- PMC

If your application requires a different form factor, please contact MEI.

Various form factors are shown in the figures below. Four types of I/O connectors are utilized: VHDCI; SERCOS; and, CPCI (rear I/O).

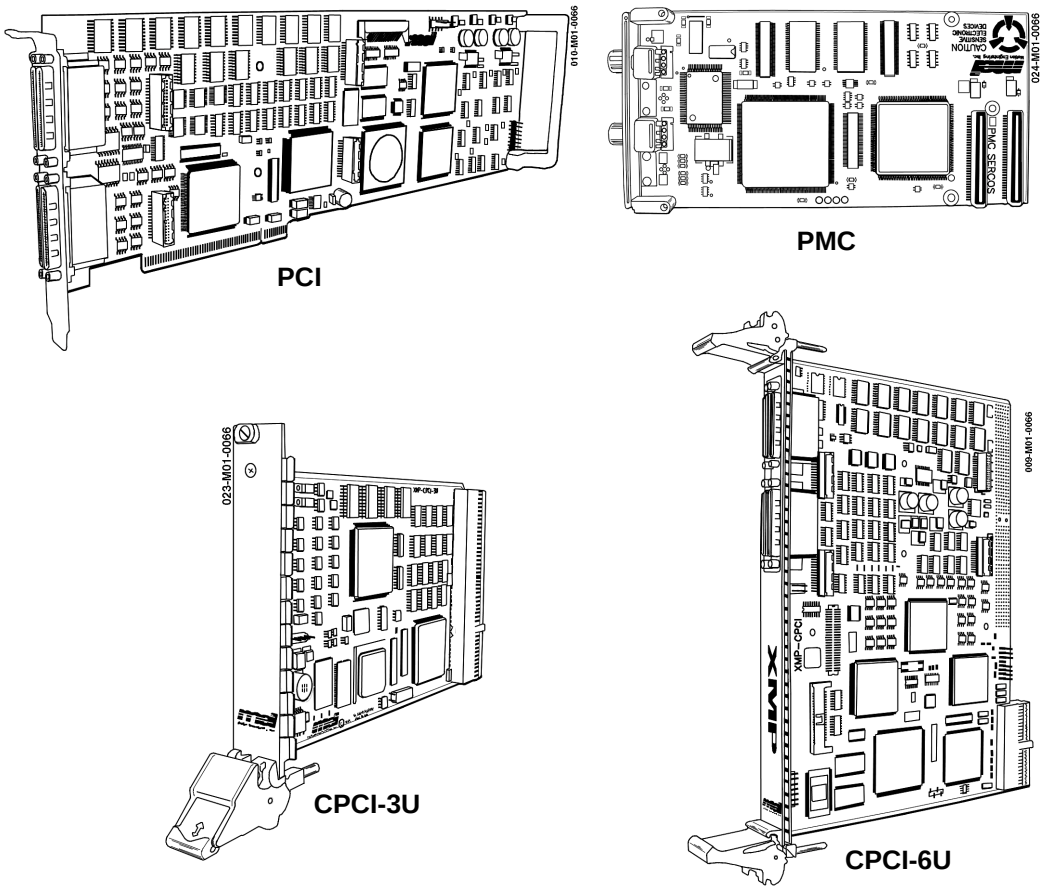


Figure 1-1 Controller form factors: PCI, PMC, and CPCI (-3U and -6U).

XMP Interface Options: Networked and XMP-analog

XMP control systems accommodate two interface categories: networked and XMP-Analog. The types of hardware associated with each category are show here:

Network

- SERCOS

XMP-analog

- Analog ± 10 VDC
- Pulse (Step / Dir)
- Quadrature encoder
- Absolute encoder

XMP-SERCOS Systems

SERCOS (SErial Real-time COmmunication System) is the most common type of networked motion control. All data is transmitted over a fiber-optic ring, with each SERCOS device having both a dedicated input and output. At regular intervals, the XMP controller receives a coded data packet from each device in the ring, reporting its current status. The controller then processes the information, and transmits an updated command data packet to each SERCOS device. Typical SERCOS cycle rates vary between 500-1000 samples per second.

XMP-analog Systems

XMP-analog controllers represent a traditional form of motion control, with command and feedback data transmitted via numerous discrete electrical connections.

XMP Controller Configurations

As demonstrated in the “XMP Hardware Overview” above, XMP controllers come from MEI with a variety of configurable options; however, not all options apply. (For example, SERCOS controllers have no voltage options because they function optically.)

MEI organizes its XMP controllers by form factor, bus, axes and voltage. Several common controllers are listed below in Table 1-1.

Table 1-1 Common XMP controllers.

Data Type	*Form Factor*	*Opto Input Voltage*	*No. of Axes*[a]	*No. of Rings*[b]	*Description*	*Part No.*
XMP-analog	PCI	5	4		XMP-PCI-4-5V	T003-0006
			8		XMP-PCI-8-5V	T003-0001
			16		XMP-PCI-16-5V	T003-0003
		24	4		XMP-PCI-4-24V	T003-0005
			8		XMP-PCI-8-24V	T003-0002
			16		XMP-PCI-16-24V	T003-0004
	CPCI-3U	5	4		XMP-CPCI-3U-4-5V	T001-0010
	CPCI-6U	5	4		XMP-CPCI-4-5V	T001-0016
			8		XMP-CPCI-8-5V	T001-0003
		24	4		XMP-CPCI-4-24V	T001-0017
			8		XMP-CPCI-8-24V	T001-0004
			12		XMP-CPCI-12-24V	T001-0015
			16		XMP-CPCI-16-24V	T001-0006
Network (XMP-SERCOS)	PCI	N/A	8	1	XMP-SERCOS-PCI-S1	T004-0005
		N/A	16	2	XMP-SERCOS-PCI-S2	T004-0004
	CPCI-6U	N/A	24	3	XMP-SERCOS-CPCI-S3	T004-0001
	PMC	N/A	8	1	XMP-SERCOS-PMC-S1	T004-0002

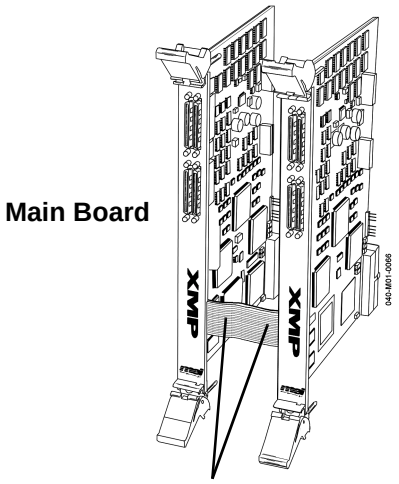
a. Most basic (“main”) XMP analog controllers come configured for 8 axes. Additional axes are configured by adding an expansion board to the controller, boosting the total number to either 12 or 16 axes per controller. (Expansion is not available in XMP-CPCI-3U or PMC form factors.)
b. SERCOS controllers are configured for either 1 or 3 rings, with each ring handling a maximum of 8 SERCOS devices.

IMPORTANT NOTE: When ordering parts or service from MEI, please refer to part number (e.g., “T003-0005”).

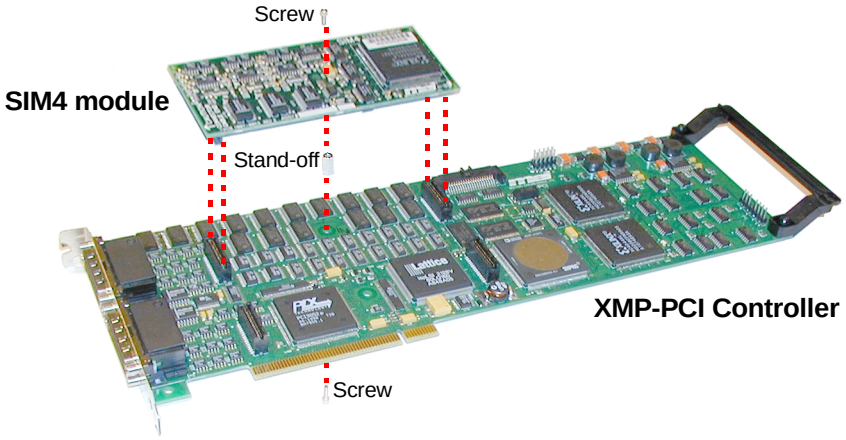
Expansions and Add-ons

Some XMP controllers may be enhanced with additional hardware. These include:

• **Expansion boards**— These expand the number of axes run by a controller by adding up to two motion blocks for a total of 16 axes. Unlike main boards, expansion boards do not contain SHARC DSP processors. They are available for XMP-PCI and XMP-CPCI-6U form factors only (not available for XMP-CPCI-3U and XMP-SERCOS-PMC controllers). An expanded XMP-CPCI-6U controller is shown below.



• **SIM4 modules**— Scale interpolation modules (SIMs) are mounted mezzanine-style to motion controllers that use encoders with sinusoidal outputs. These enhance positional resolution significantly and are available for XMP-PCI and XMP-CPCI-6U controllers only. A maximum of two SIM4 modules may be added to each main or expansion board. A SIM4 module is shown below with an XMP-PCI controller.

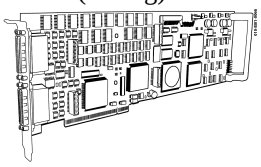
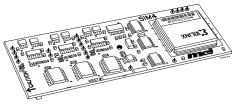
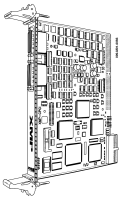
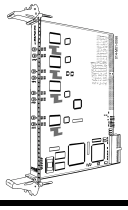
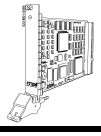
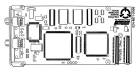
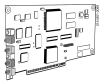


For detailed information regarding SIM4 modules, please refer to *Application Note 206* from Motion Engineering.

Host Computer Electrical Requirements

Below is shown a table of basic host computer electrical requirements for XMP controllers.

Table 1-2 XMP voltage and current specifications.

Controller	*Voltage*[a]	*Current*[b] *(Typ. amps)*	*Current (Max. amps)*
XMP-PCI (analog) 	+5 VDC ± 5%	1.70 (main) 1.20 (expansion)	3.20 (main) 2.20 (expansion)
	+12 VDC ± 10%	0.03 (main) 0.01 (expansion)	0.05 (main) 0.01 (expansion)
	-12 VDC ± 10%	0.01 (main) 0.01 (expansion)	0.01 (main) 0.01 (expansion)
SIM4 Module on XMP-PCI 	+5 VDC ± 5%	0.18	0.40
	+12 VDC ± 10%	0.01	0.01
	-12 VDC ± 10%	0.03	0.05
XMP-CPCI-6U (analog) 	+3.3 VDC ± 10%	0.90 (main) 0.70 (expansion)	1.40 (main) 1.00 (expansion)
	+5 VDC ± 5%	0.90 (main) 0.90 (expansion)	1.90 (main) 1.90 (expansion)
	+12 VDC ± 10%	0.03 (main) 0.01 (expansion)	0.05 (main) 0.01 (expansion)
	-12 VDC ± 10%	0.01 (main) 0.01 (expansion)	0.01 (main) 0.01 (expansion)
SIM4 Module on XMP-CPCI 	+3.3 VDC ± 10%	0.08	0.20
	+5 VDC ± 5%	0.12	0.20
	+12 VDC ± 10%	0.01	0.01
	-12 VDC ± 10%	0.03	0.05
XMP-SERCOS-CPCI-6U 	+3.3 VDC ± 10%	0.50	0.75
	+5 VDC ± 5%	0.50	0.75
XMP-CPCI-3U (analog) 	+3.3 VDC ± 10%	0.50	0.90
	+5 VDC ± 5%	0.45	1.15
XMP-SERCOS-PMC 	+3.3 VDC ± 10%	0.375	0.50
	+5 VDC ± 5%	0.375	0.50
XMP-SERCOS-PCI-S2 	+5 VDC ± 5%	0.60	1.00

a. Controller voltage required at computer or card cage.
b. Installed SIM4 modules and encoders also contribute to current demands.

Using the XMP controller to power encoders?

Then make sure that voltage drops in your cabling aren't a problem. If voltage drops are a problem, power the encoders using an external supply. For every 100 mA of current drawn, the 5V_OUT voltage may drop 0.2 volts or more (due to cable resistance). For example, if two encoders each draw 150 mA, then 5V_OUT may drop to 4.4 volt at the encoders. This could cause your encoder to fail!

SERCOS Drives

SERCOS is a standardized industrial format utilizing fiber-optic technology. The optical power demands of drives vary widely, depending upon the drive itself, length of cables, etc. For more information regarding SERCOS motion drive power demands, please see your drive's manufacturer.

XMP-analog Controllers

XMP-analog systems rely upon traditional copper wiring to pass analog signals between the drive and controller. Two voltages are available for opto-inputs on XMP controllers: 5VDC and 24VDC. To determine whether your controller supports 5V or 24V, look for a sticker on the XMP circuit board. The sticker will say either "5V" or "24V."

Detailed information regarding configuration of analog I/O is provided in Chapter 4 of this manual. Specifications for transceiver components used in the XMP-Analog controller's I/O are provided in tables below.

Table 1-3 RS422 receiver.

Description	***Specification***
Input High Threshold	V_{DIFF} > 0.2V
Input Low Threshold	V_{DIFF} < -0.2V
Input Current	I_{IN} < 1.0mA @ V_{IN} = 12V I_{IN} < -0.8mA @ V_{IN} = -7V NOTE: In addition, encoder inputs have 100Ω differential termination.
Propagation Delay	T_p < 200 ns
Absolute Maximums[a]	ESD Protection ± 15kV -8V < V_{IN} < +12.8V (input to GND)

a. Exceeding absolute maximums may damage components.

Table 1-4 RS422 transmitter.

Description	***Specification***
Differential Output Voltage	V_{DIFF} < 5V @ I = 0mA V_{DIFF} > 2V @ I = 20mA
Common Mode Output Voltage	V_{OUT} (common mode) < 3V (100Ω load)
Propagation Delay	T_p < 60ns

Table 1-4 RS422 transmitter.

Description	Specification
Absolute Maximums[a]	$I_{OUT} < 50mA$

a. Exceeding absolute maximums may damage components.

Table 1-5 Dedicated opto-inputs.

Description	Specification
Active Input Guaranteed	$I_{IN} > 2mA$ 5V input (820Ω current limit resistor): $V_{IN} < 3V$ @ $I_{IN} = 2mA$ 24V input (6.8kΩ current limit resistor): $V_{IN} < 15V$ @ $I_{IN} = 2mA$
Inactive Input Guaranteed	$V_{IN} <$ 0.1mA
Propagation Delay	$T_p < 20\mu s$
Absolute Maximums[a]	5V input (820Ω current limit resistor): $V_{IN} < 10V$ $V_{IN} < 10mA$ 24V input (6.8kΩ current limit resistor): $V_{IN} < 30V$ $V_{IN} < 5mA$ $V_{ISOLATION} < 40V$

a. Exceeding absolute maximums may damage components.

Table 1-6 Dedicated opto-outputs.

Description	Specification
Active Output Guaranteed	V_{OUT} < 0.3V @ I_{OUT} < 2mA V_{OUT} < 1.1V @ I_{OUT} < 10mA
Inactive Output Guaranteed	I_{OUT} < 0.01mA
Propagation Delay	T_p < 100µs (load < 10 kΩ)
Absolute Maximums[a]	I_{OUT} < 50mA V_{OUT} < 30V $I_{REVERSE}$ < 100mA @ $V_{REVERSE}$ 0.4V (protection diode) $V_{ISOLATION}$ < 40V

a. Exceeding absolute maximums may damage components.

Table 1-7 Bi-directional opto-inputs.

Description	Specification
Active Input Guaranteed	I_{IN} > 2mA V_{IN} < 1.7V @ I_{IN} = 2mA
Inactive Input Guaranteed	I_{IN} < 0.1mA
Propagation Delay	T_p < 20µs
Absolute Maximums[a]	I_{IN} < 50mA $I_{REVERSE}$ < 30V $V_{ISOLATION}$ < 40V

a. Exceeding absolute maximums may damage components.

Table 1-8 Bi-directional opto-outputs.

Description	Specification
Active Output Guaranteed	V_{OUT} < 0.3V @ I_{OUT} < 2mA V_{OUT} < 1.1V @ I_{OUT} < 10mA
Inactive Input Guaranteed	I_{OUT} < 0.01mA
Propagation Delay	T_p < 100µs (load < 10 kΩ)
Absolute Maximums[a]	I_{OUT} < 50mA V_{OUT} < 30V $I_{REVERSE}$ < 50mA @ $V_{REVERSE}$ 1.7V (input circuit) $V_{ISOLATION}$ < 40V

a. Exceeding absolute maximums may damage components.

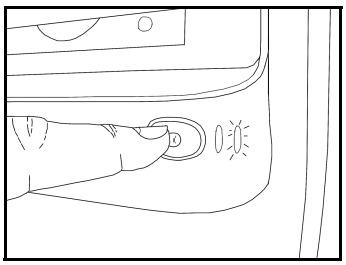
XMP Controller Installation

Depending upon form factor, installation of your XMP controller will vary. However, in all cases, good electrical contact must be ensured at connectors; otherwise, noise and power problems will develop. (Connections should be verified through inspection and testing.)

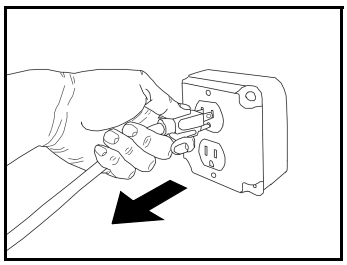
Standard safety rules prevail during installation of any hardware. Some are summarized below for the XMP. For more information, refer to local occupational safety regulations and the manufacturer of your motion drive.

Turn Off All Power Before Installing Equipment

Before installing any motion control equipment, including XMP controllers, power should be switched OFF. Unplug all power plugs from their sources of power.



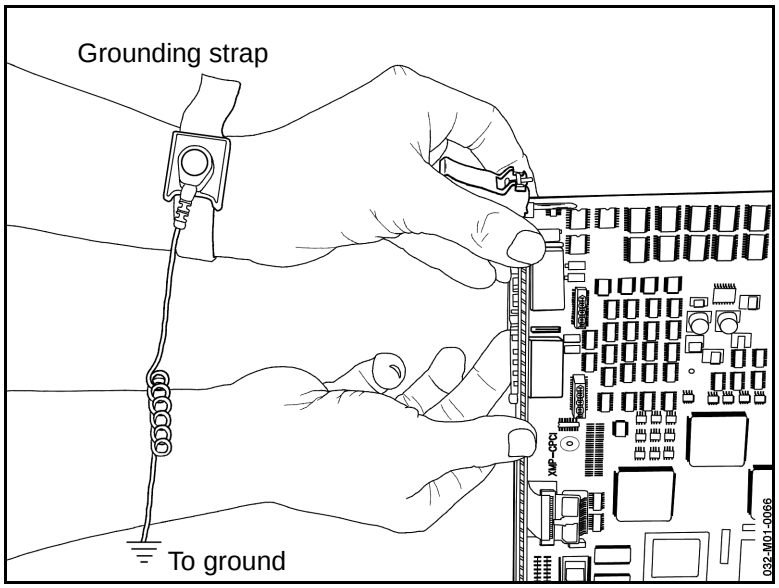
Switch OFF equipment.



Unplug from source of power.

Observe ESD Precautions

To prevent damage to controller and drive electronics due to electrostatic discharge (ESD), service personnel are cautioned to observe proper grounding during handling of components.



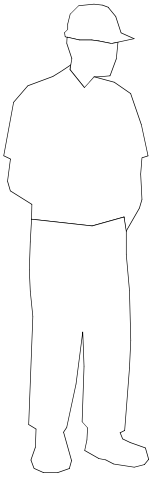
Grounding straps should be worn at all times when handling XMP controller boards and connection hardware.

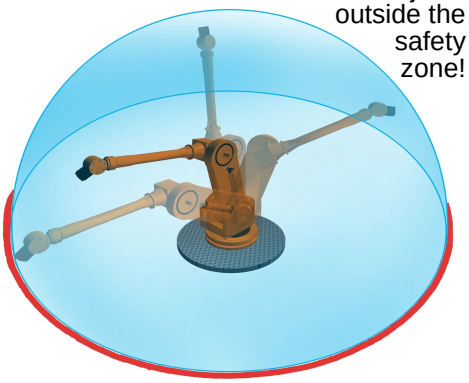
Define and Clear a Safety Zone!

During installation and testing of motion control hardware-software, a safety zone should be defined around moving components and kept clear of personnel, hands, fingers, and loose hardware. During repowering of the system, motion control components may behave erratically due to misconnected lines, or wrongly configured software settings.

Sudden and unexpected moves by components can cause injury, property damage or even death! Under NO circumstances, should a motion system be tested or operated while personnel are within the safety zone.

Additionally, beware of flying debris from unsecured hardware operating at high speeds. The use of safety shielding is highly recommended.



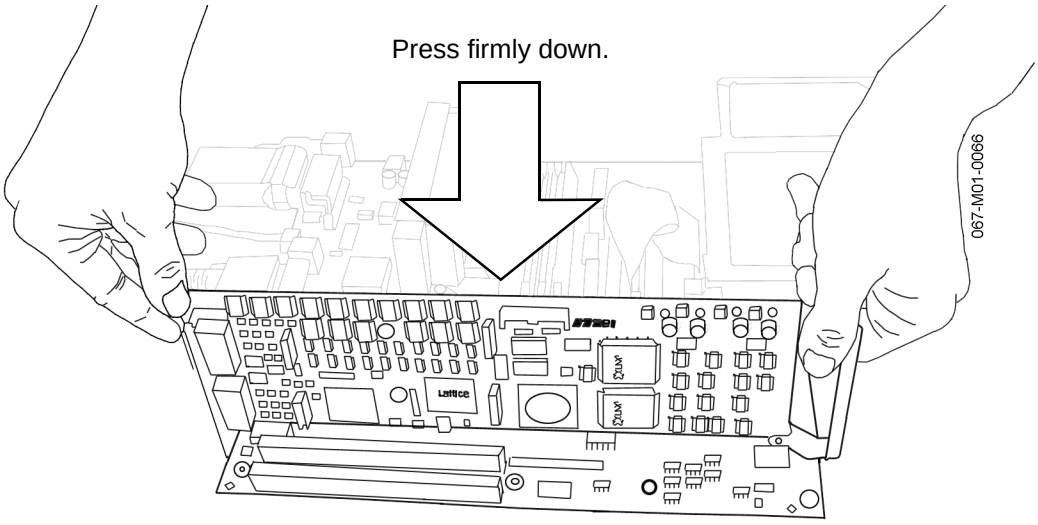


Install Controller

XMP controllers are designed to fit standard industry buses and card cages (e.g., PCI, CPCI). When installing and removing controllers, verify that good electrical contact is made between the controller and receptacle.

PCI

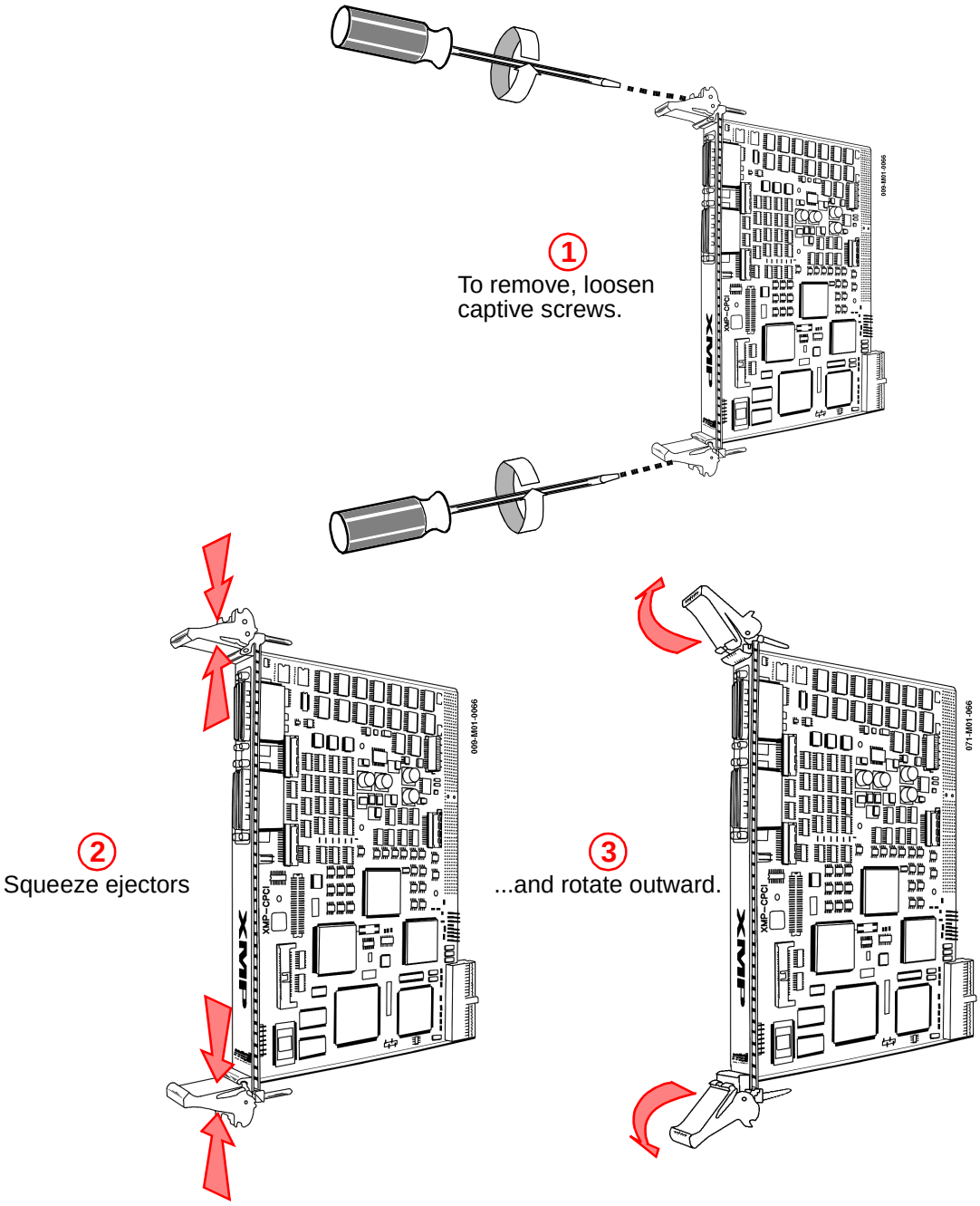
XMP-PCI controllers are installed into the PCI buses of their host computer's mother boards. During installation, verify the controller is seated firmly in its assigned PCI slot by firmly pressing downward. The I/O bracket should be screwed securely to the expansion bay chassis.



XMP-CPCI-6U and XMP-SERCOS-CPCI-6U

XMP-CPCI-6U controllers are installed in industry standard CPCI slots. They are secured with ejectors. To install, first verify that both ejectors are in the open (rotated outward) position. Align the PC board with the card slot in the CPCI chassis, then press home. To secure, rotate each ejector until claws engage holes in the top-bottom chassis rails and lock the controller into place.

To remove an XMP-CPCI-6U controller from its slot, first loosen the captive screws located at the top and bottom chassis rails. Next, simultaneously pinch each ejector at its flex point to unlock, then rotate outward. The controller should eject.

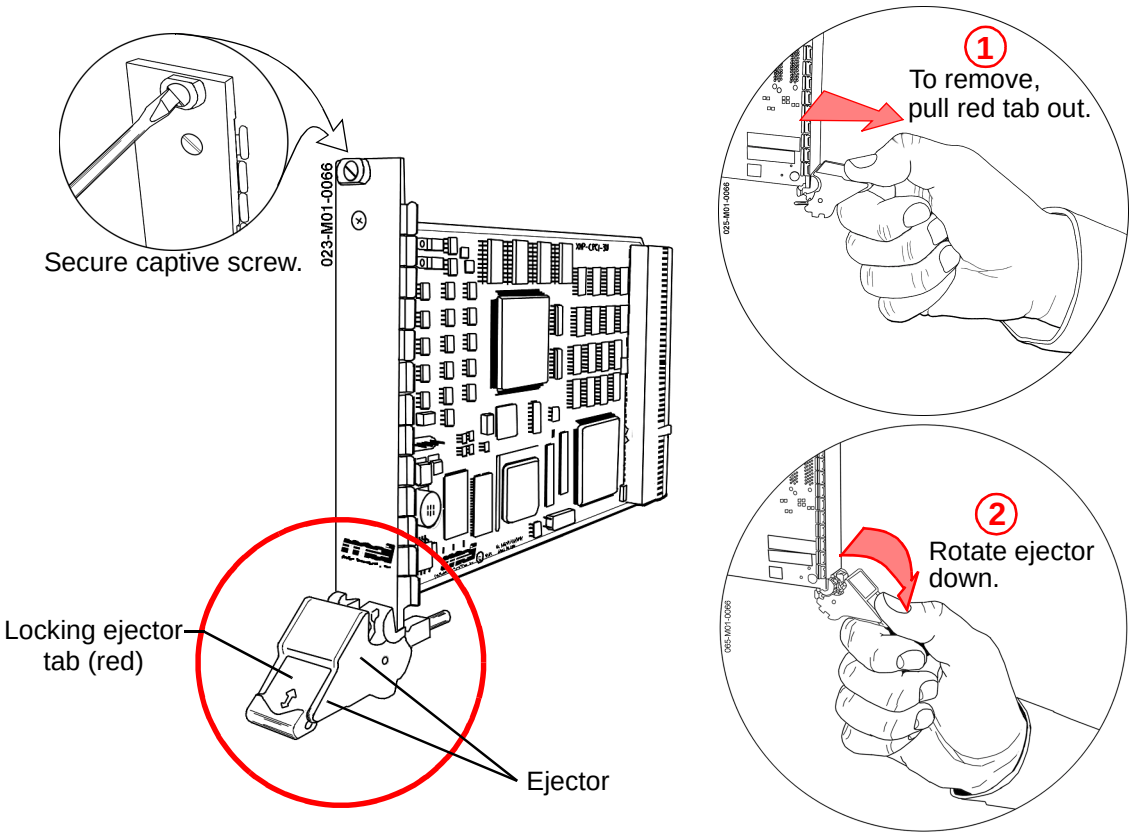


XMP-CPCI-3U

XMP-CPCI-3U controllers feature one locking ejector at the base of the front panel and a captive screw at the top. The XMP-CPCI-3U chassis has an ejector rail along its bottom and a threaded hole on its top rail to match the captive screw.

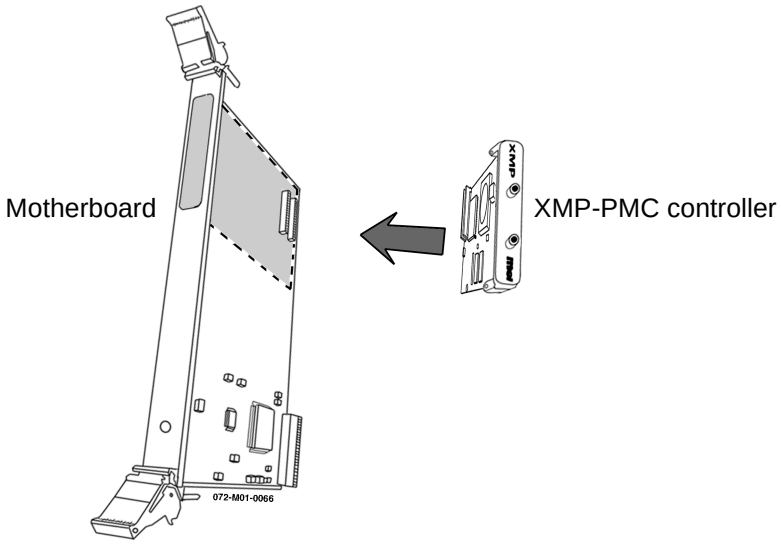
To install, first verify that the bottom ejector is in the open (rotated outward) position. Align the PC board with the card slot in the CPCI chassis, then press home. To secure, apply gentle pressure to the edge of the card while rotating the bottom ejector inward. The ejector's claws will engage holes in the bottom chassis rail and lock the controller into place. Secure the captive screw in the chassis at the top of the controller by rotating clockwise.

To remove an XMP-CPCI-3U controller from its slot, first loosen the captive screw by rotating counterclockwise. Pull the red ejector tab outward, then rotate the ejector downward. The controller should eject.



XMP-SERCOS-PMC

XMP-PMC controllers are connected directly to motherboards via mezzanine connectors. The boards are secured to the motherboard using two (2) screws and stand-offs. To prevent damage to the motherboard, do not overtighten the screws.



This is a blank page.

CHAPTER 2
XMP BUS INTERFACE

Introduction

This chapter describes XMP controller-computer bussing, and is organized by form factor. The reader should note that both XMP-analog and XMP-SERCOS controllers utilize identical bus connections for each respective form factor. (For example, a CPCI controller may have the same bus connections, whether the I/O is configured for XMP-analog or -SERCOS.)

"Main" versus "Expansion" Boards (XMP-analog only)

XMP-analog controllers may be configured for varying numbers of axes. Up to 8 axes are controlled by a single "main" controller board. Where more than 8 axes are required, it is necessary to add an "expansion" board.

Expansion boards consist of separate electronics, very similar in appearance and function to main boards. Typically, an expansion board is located on the bus or backplane immediately adjacent to a main board. Both boards (main and expansion) are connected via a short ribbon cable. When connected together, one main and one expansion board constitute a single, unified "controller."

Expansion boards are *not* available for XMP-SERCOS controllers.

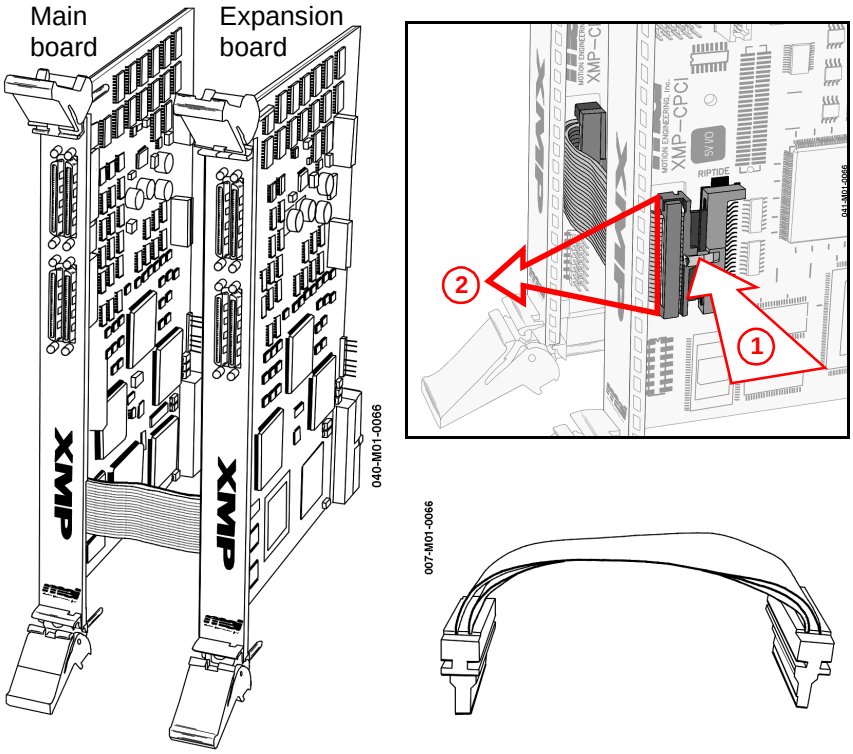
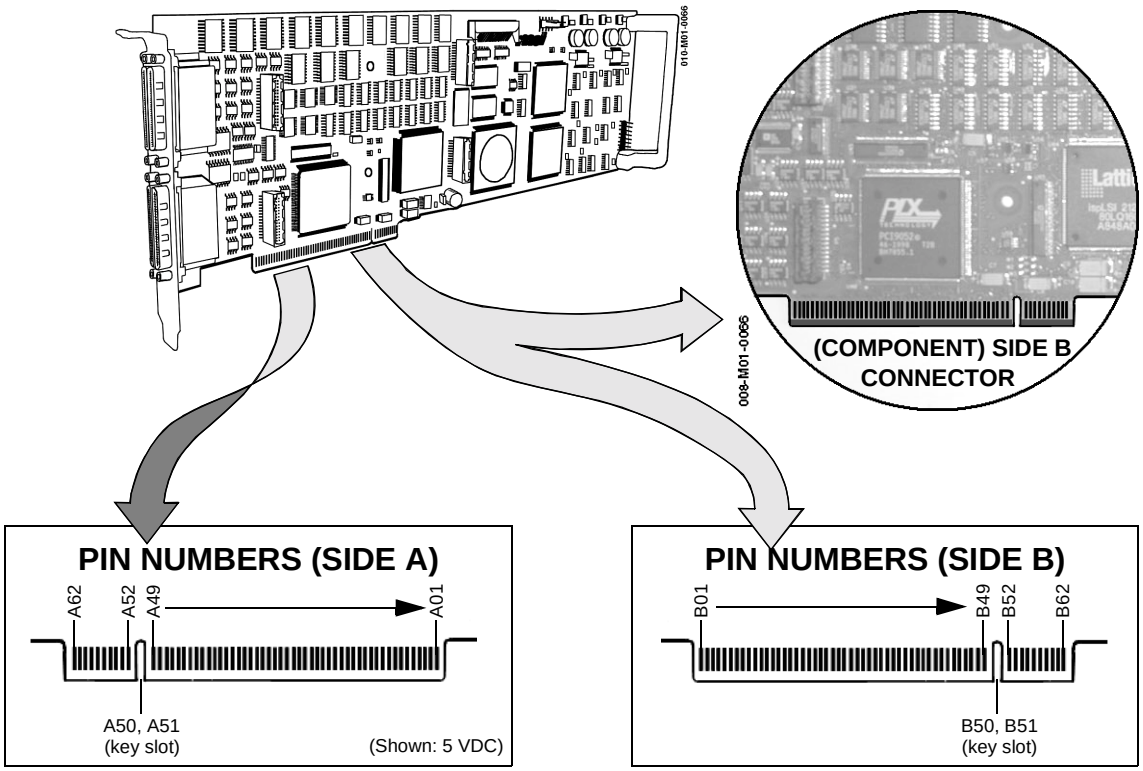


Figure 2-1 XMP-CPCI-6U main and expansion boards. Connection is via ribbon cable. To release ribbon cable, press clip (1) while disconnecting connector (2). XMP-PCI connection is similar.

PCI Bus



PCI buses are found in many common desktop and tower-type computer systems. The standard XMP-PCI controller features a 120-pin edge connector which plugs directly into standard 5VDC PCI bus receptacles. (Refer to figure above.) A “universal” PCI that supports 3.3V or 5V signalling is pending from MEI. Please contact MEI for availability.

All motion control I/O for the XMP-PCI controller is performed via the VHDCI connectors on the rear panel of the controller. For more information on I/O, please refer to Chapter 4.

PCI Connector (Bus)

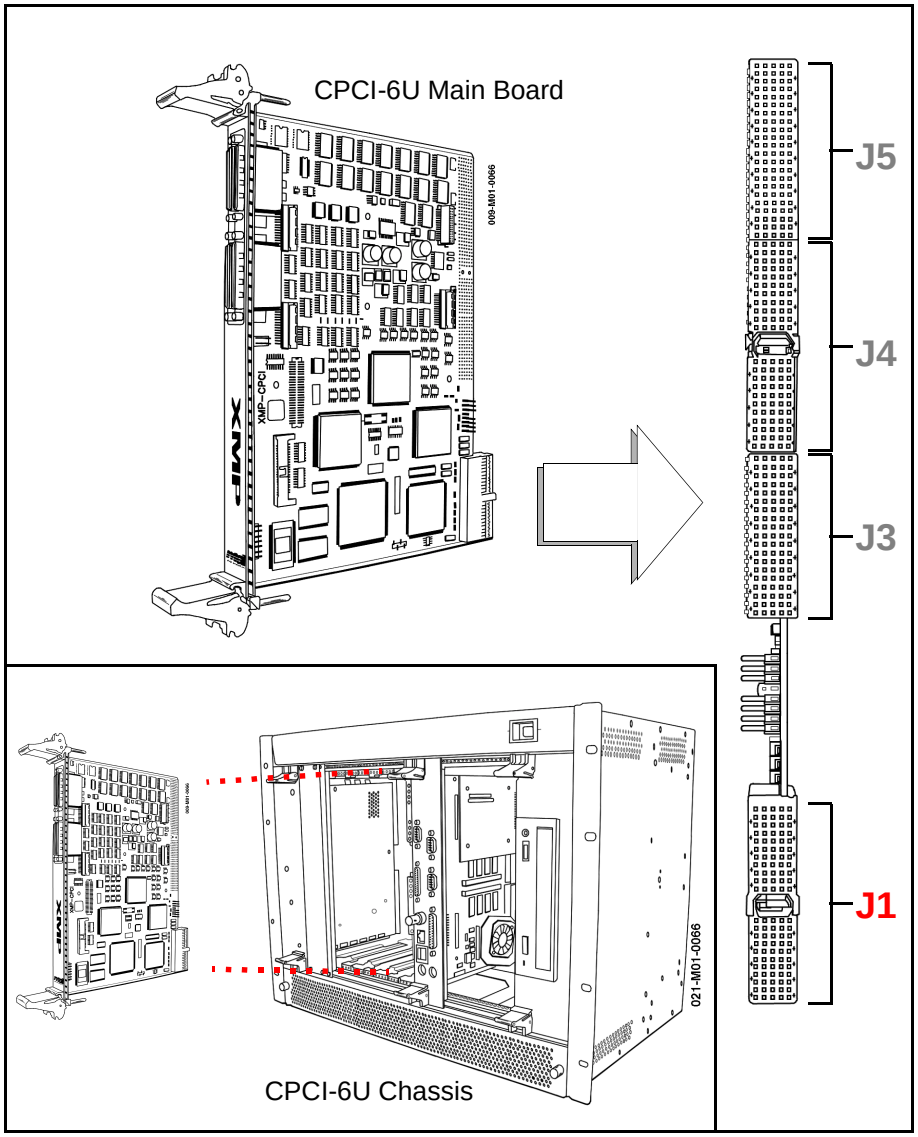
Table 4-1. PCI bus connector (main & expansion boards).

5 V System				Universal System (5 or 3.3 V)				
Side B		Side A		Side B		Side A		
Pin	Function	Pin	Function	Pin	Function	Pin	Function	Comments
01B	-12V	01A	TRST#	01B	-12V	01A	TRST#	32-bit Connector Start
02B	TCK	02A	+12V	02B	TCK	02A	+12V	
03B	GND	03A	TMS	03B	GND	03A	TMS	
04B	TDO	04A	TDI	04B	TDO	04A	TDI	
05B	+5V	05A	+5V	05B	+5V	05A	+5V	
06B	+5V	06A	INTA#	06B	+5V	06A	INTA#	
07B	INTB#	07A	INTC#	07B	INTB#	07A	INTC#	
08B	INTD#	08A	+5V	08B	INTD#	08A	+5V	
09B	PRSNT1#	09A	Reserved	09B	PRSNT1#	09A	Reserved	
10B	Reserved	10A	+5V (I/O)	10B	Reserved	10A	+5V (I/O)	
11B	PRSNT2#	11A	Reserved	11B	PRSNT2#	11A	Reserved	
12 B	Ground	12A	Ground	12B	KEY SLOT	12A	KEY SLOT	
13B	Ground	13A	Ground	13B		13A		
14B	Reserved	14A	Reserved	14B	Reserved	14A	Reserved	
15B	Ground	15A	RST#	15B	Ground	15A	RST#	
16B	CLK	16A	+5V (I/O)	16B	CLK	16A	+5V (I/O)	
17B	Ground	17A	GNT#	17B	Ground	17A	GNT#	
18B	REQ#	18A	Ground	18B	REQ#	18A	Ground	
19B	+5V (I/O)	19A	Reserved	19B	+V (I/O)	19A	Reserved	
20B	AD[31]	20A	AD[30]	20B	AD[31]	20A	AD[30]	
21B	AD[29]	21A	+3.3V	21B	AD[29]	21A	+3.3V	
22B	Ground	22A	AD[28]	22B	Ground	22A	AD[28]	
23B	AD[27]	23A	AD[26]	23B	AD[27]	23A	AD[26]	
24B	AD[25]	24A	Ground	24B	AD[25]	24A	Ground	
25B	+3.3V	25A	AD[24]	25B	+3.3V	25A	AD[24]	
26B	C/BE[3]#	26A	IDSEL	26B	C/BE[3]#	26A	IDSEL	
27B	AD[23]	27A	+3.3V	27B	AD[23]	27A	+3.3V	
28B	Ground	28A	AD[22]	28B	Ground	28A	AD[22]	
29B	AD[21]	29A	AD[20]	29B	AD[21]	29A	AD[20]	
30B	AD[19]	30A	Ground	30B	AD[19]	30A	Ground	
31B	+3.3V	31A	AD[18]	31B	+3.3V	31A	AD[18]	
32B	AD[17]	32A	AD[16]	32B	AD[17]	32A	AD[16]	
33B	C/BE[2]#	33A	+3.3V	33B	C/BE[2]#	33A	+3.3V	
34B	Ground	34A	FRAME#	34B	Ground	34A	FRAME#	
35B	IRDY#	35A	Ground	35B	IRDY#	35A	Ground	
36B	+3.3V	36A	TRDY#	36B	+3.3V	36A	TRDY#	
37B	DEVSEL#	37A	Ground	37B	DEVSEL#	37A	Ground	
38B	Ground	38A	STOP#	38B	Ground	38A	STOP#	

Table 4-1. PCI bus connector (main & expansion boards).

5 V System				Universal System (5 or 3.3 V)				
Side B		Side A		Side B		Side A		
Pin	**Function**	**Pin**	**Function**	**Pin**	**Function**	**Pin**	**Function**	**Comments**
39B	LOCK#	39A	+3.3V	39B	LOCK#	39A	+3.3V	
40B	PERR#	40A	SDONE	40B	PERR#	40A	SDONE	
41B	+3.3V	41A	SBO#	41B	+3.3V	41A	SBO#	
42B	SERR#	42A	Ground	42B	SERR#	42A	Ground	
43B	+3.3V	43A	PAR	43B	+3.3V	43A	PAR	
44B	C/BE[1]#	44A	AD[15]	44B	C/BE[1]#	44A	AD[15]	
45B	AD[14]	45A	+3.3V	45B	AD[14]	45A	+3.3V	
46B	Ground	46A	AD[13]	46B	Ground	46A	AD[13]	
47B	AD[12]	47A	AD[11]	47B	AD[12]	47A	AD[11]	
48B	AD[10]	48A	Ground	48B	AD[10]	48A	Ground	
49B	Ground	49A	AD[09]	49B	M66EN	49A	AD[09]	66 MHz / Gnd
50B	KEY SLOT	50A	KEY SLOT	50B	KEY SLOT	50A	KEY SLOT	5 Volt Key
51B		51A		51B		51A		5 Volt Key
52B	AD[08]	52A	C/BE[0]#	52B	AD[08]	52A	C/BE[0]#	
53B	AD[07]	53A	+3.3V	53B	AD[07]	53A	+3.3V	
54B	+3.3V	54A	AD[06]	54B	+3.3V	54A	AD[06]	
55B	AD[05]	55A	AD[04]	55B	AD[05]	55A	AD[04]	
56B	AD[03]	56A	Ground	56B	AD[03]	56A	Ground	
57B	Ground	57A	AD[02]	57B	Ground	57A	AD[02]	
58B	AD[01]	58A	AD[00]	58B	AD[01]	58A	AD[00]	
59B	+5V (I/O)	59A	+5V (I/O)	59B	+V (I/O)	59A	+5V (I/O)	
60B	ACK64#	60A	REQ64#	60B	ACK64#	60A	REQ64#	
61B	+5V	61A	+5V	61B	+5V	61A	+5V	
62B	+5V	62A	+5V	62B	+5V	62A	+5V	32-bit Connector End

CPCI-6U Bus

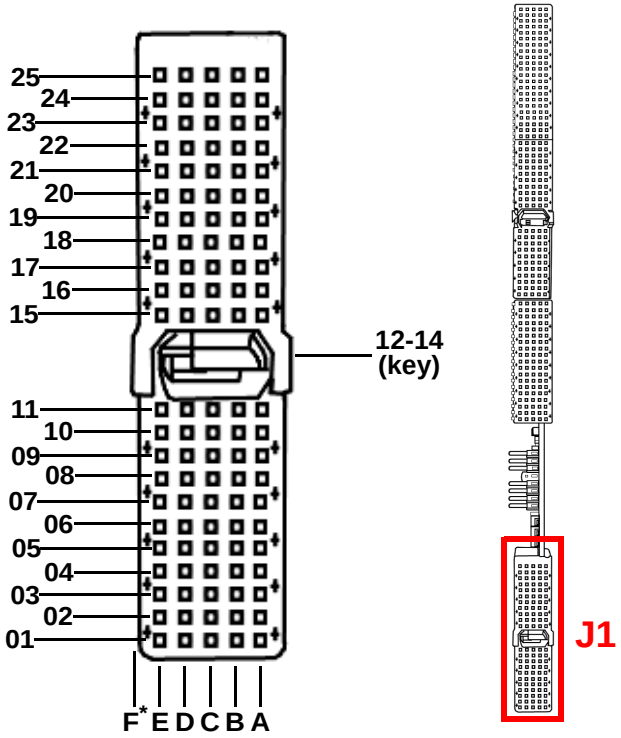


Compact PCI (CPCI) utilizes a card-type computer chassis, having a multi-pin backplane. The main advantage is smaller sizing and simplified access to electronics. There are two sizes of CPCI: -6U and -3U. The CPCI-6U is the larger of the two form factors and connects to the backplane bus via jumper connector J1. The CPCI-6U is described in this section.

The CPCI-3U is described in the next section of this chapter.

CPCI-6U Main Board, Rear Panel Connector J1

CPCI-6U: J1



NOTE: the CPCI XMP does not use the 64-bit extension.

Table 4-2. CPCI backplane connector (main and expansion boards).

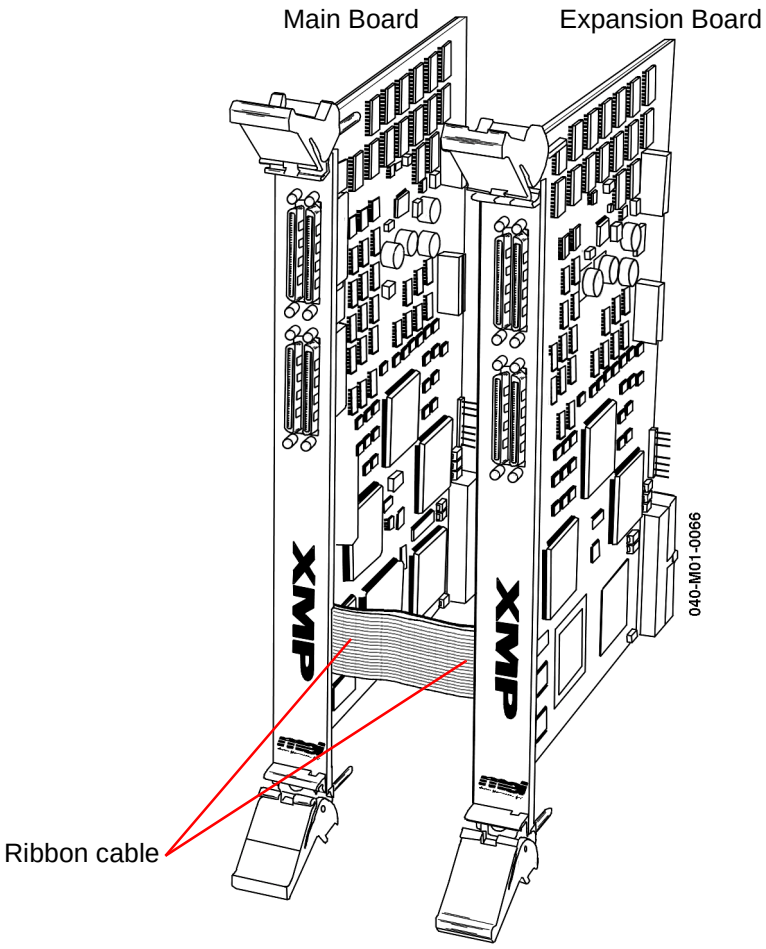
Pin	*A*	*B*	*C*	*D*	*E*
25	5V	REQ64#	ENUM#	3.3V	5V
24	AD[01]	5V	V(I/O)	AD[00]	ACK64#
23	3.3V	AD[04]	AD[03]	5V	AD[02]
22	AD[07]	GND	3.3V	AD[06]	AD[05]
21	3.3V	AD[09]	AD[08]	M66EN	C/BE[0]#
20	AD[12]	GND	V(I/O)	AD[11]	AD[10]
19	3.3V	AD[15]	AD[14]	GND	AD[13]
18	SERR#	GND	3.3V	PAR	C/BE[1]#
17	3.3V	SDONE	SBO#	GND	PERR#
16	DEVSEL#	GND	V(I/O)	STOP#	LOCK#
15	3.3V	FRAME#	IRDY#	GND	TRDY#
12 - 14	Key Area				
11	AD[18]	AD[17]	AD[16]	GND	C/BE[2]#
10	AD[21]	GND	3.3V	AD[20]	AD[19]
9	C/BE[3]#	IDSEL	AD[23]	GND	AD[22]
8	AD[26]	GND	V(I/O)	AD[25]	AD[24]

Table 4-2. CPCI backplane connector (main and expansion boards).

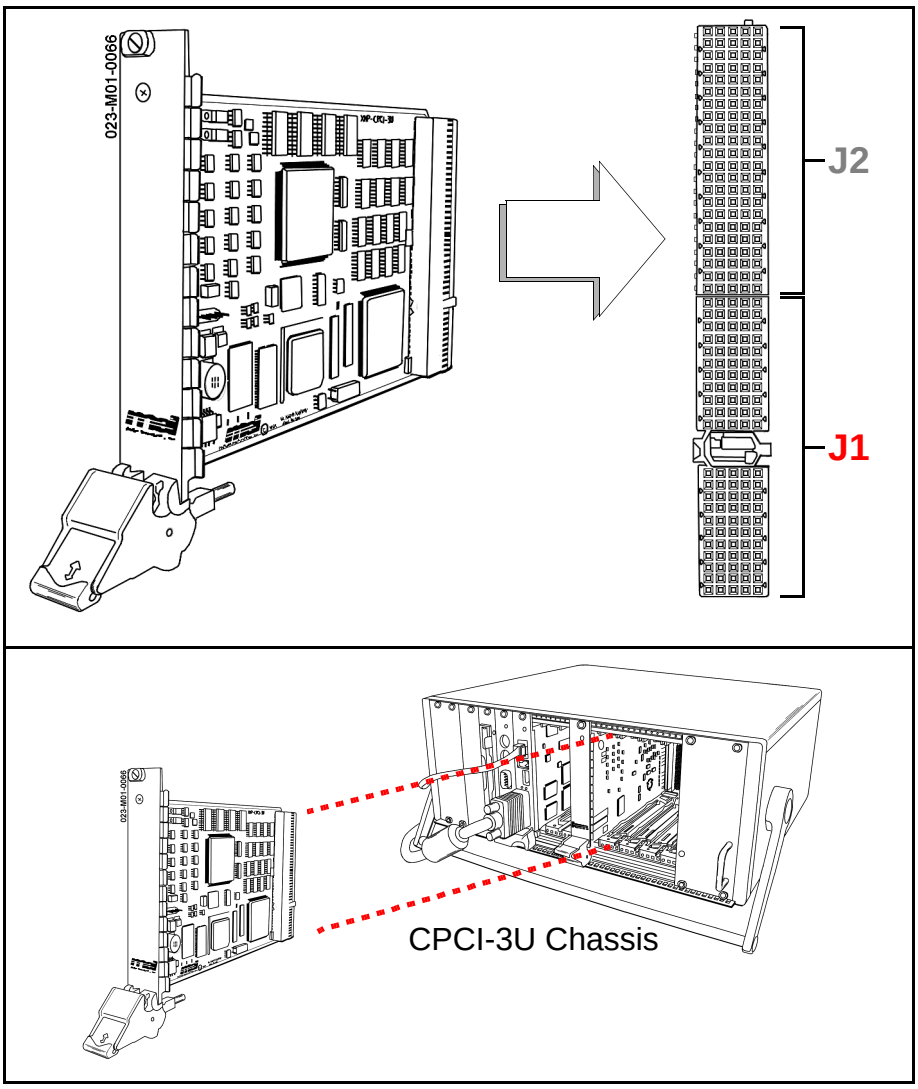
Pin	*A*	*B*	*C*	*D*	*E*
7	AD[30]	AD[29]	AD[28]	GND	AD[27]
6	REQ#	GND	3.3V	CLK	AD[31]
5	BRSVP1A5	BRSVP1B5	RST#	GND	GNT#
4	BRSVP1A4	GND	V(I/O)	INTP	INTS
3	INTA#	INTB#	INTC#	5V	INTD#
2	TCK	5V	TMS	TDO	TDI
1	5V	-12V	TRST#	+12V	5V

CPCI-6U Expansion Board

Special Note: On CPCI-6U motion control systems having an expansion board, all computer bus data is conveyed via the main board's J1 connector, then relayed via the ribbon cable connector. Expansion boards also utilize a J1 backplane connector; however, it provides only power and ground to the expansion board.

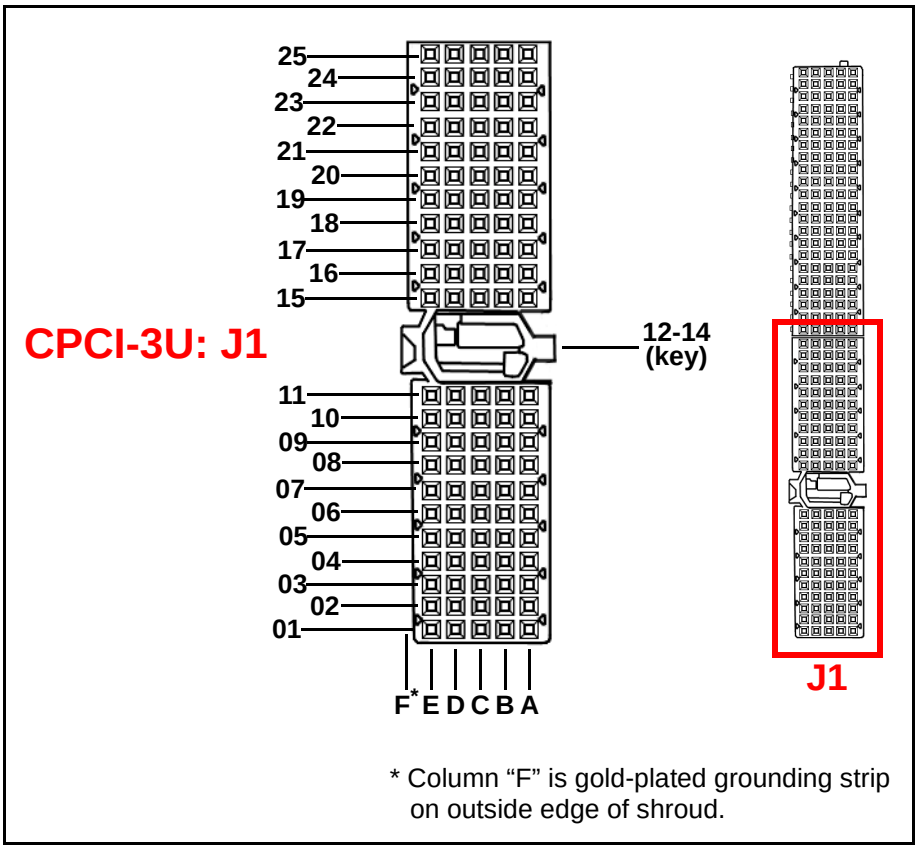


CPCI-3U Bus



The four-axis CPCI-3U is a smaller version of the -6U board. It connects to the backplane via two connectors: J1 and J2 (see figure above). Motion drive I/O is conveyed via J2, which is described in Chapter 4. Connector J1 is reserved as a computer bus, and is described in this section.

Backplane connector pins are referred to by numbered rows and lettered columns as shown below:



CPCI-3U Main Board, Rear Panel Connector J1: User I/O

NOTE: the CPCI XMP does not use the 64-bit extension.

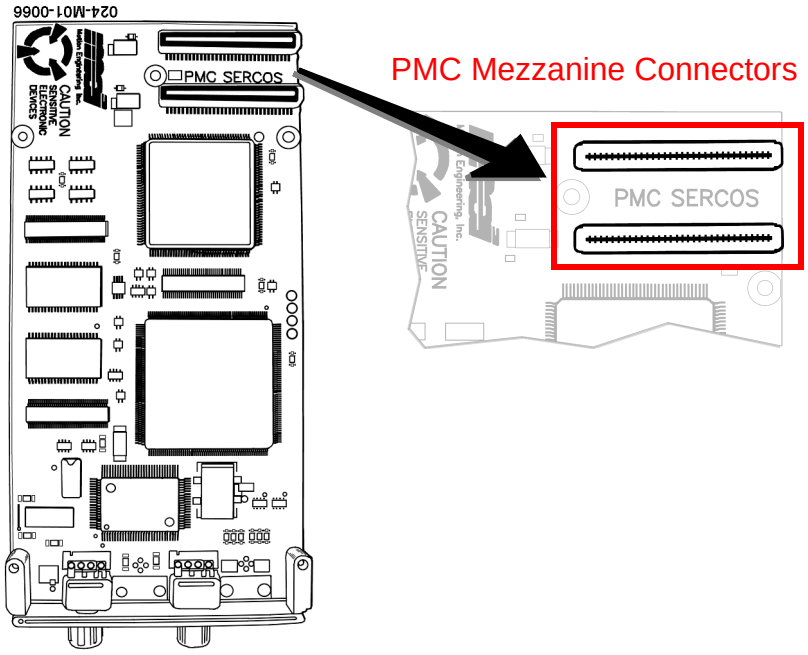
Table 4-3. CPCI backplane connector (main and expansion boards).

Pin	*A*	*B*	*C*	*D*	*E*
25	5V	REQ64#	ENUM#	3.3V	5V
24	AD[01]	5V	V(I/O)	AD[00]	ACK64#
23	3.3V	AD[04]	AD[03]	5V	AD[02]
22	AD[07]	GND	3.3V	AD[06]	AD[05]
21	3.3V	AD[09]	AD[08]	M66EN	C/BE[0]#
20	AD[12]	GND	V(I/O)	AD[11]	AD[10]
19	3.3V	AD[15]	AD[14]	GND	AD[13]
18	SERR#	GND	3.3V	PAR	C/BE[1]#
17	3.3V	SDONE	SBO#	GND	PERR#
16	DEVSEL#	GND	V(I/O)	STOP#	LOCK#
15	3.3V	FRAME#	IRDY#	GND	TRDY#
12 - 14	Key Area				
11	AD[18]	AD[17]	AD[16]	GND	C/BE[2]#
10	AD[21]	GND	3.3V	AD[20]	AD[19]

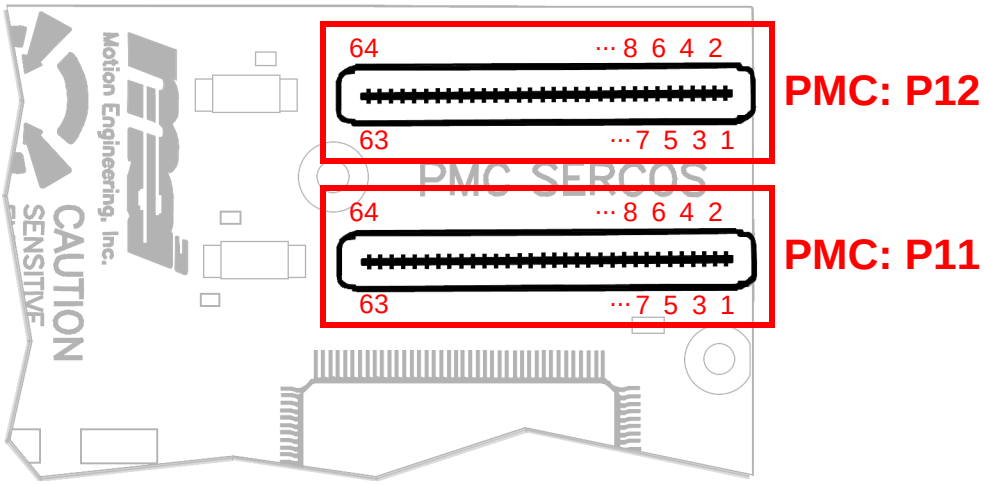
Table 4-3. CPCI backplane connector (main and expansion boards).

Pin	*A*	*B*	*C*	*D*	*E*
9	C/BE[3]#	IDSEL	AD[23]	GND	AD[22]
8	AD[26]	GND	V(I/O)	AD[25]	AD[24]
7	AD[30]	AD[29]	AD[28]	GND	AD[27]
6	REQ#	GND	3.3V	CLK	AD[31]
5	BRSVP1A5	BRSVP1B5	RST#	GND	GNT#
4	BRSVP1A4	GND	V(I/O)	INTP	INTS
3	INTA#	INTB#	INTC#	5V	INTD#
2	TCK	5V	TMS	TDO	TDI
1	5V	-12V	TRST#	+12V	5V

PMC Bus



PMC stands for **P**CI **M**ezzanine **C**ard. On XMP controllers, PMC bussing is via a “mezzanine” connector, mounted at one corner of the card.



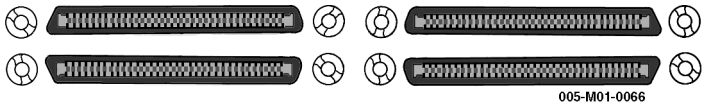
P11				P12			
Pin #	**Signal Name**	**Signal Name**	**Pin #**	**Pin #**	**Signal Name**	**Signal Name**	**Pin #**
1	TCK	-12V	2	1	+12V	TRST#	2
3	Ground	INTA#	4	3	TMS	TDO	4
5	INTB#	INTC#	6	5	TDI	Ground	6
7	BUSMODE1#	+5V	8	7	Ground	PCI-RSVD*	8
9	INTD#	PCI-RSVD*	10	9	PCI-RSVD*	PCI-RSVD*	10
11	Ground	PCI-RSVD#	12	11	BUSMODE2#	+3.3V	12
13	CLK	Ground	14	13	RST#	BUSMODE3#	14
15	Ground	GNT#	16	15	+3.3V	BUSMODE4#	16
17	REQ#	+5V	18	17	PCI-RSVD*	Ground	18
19	V (I/O)	AD[31]	20	19	AD[30]	AD[29]	20
21	AD[28]	AD[27]	22	21	Ground	AD[26]	22
23	AD[25]	Ground	24	23	AD[24]	+3.3V	24
25	Ground	C/BE[3]#	26	25	IDSEL	AD[23]	26
27	AD[22]	AD[21]	28	27	+3.3V	AD[20]	28
29	AD[19]	+5V	30	29	AD[18]	Ground	30
31	V (I/O)	AD[17]	32	31	AD[16]	C/BE[2]#	32
33	FRAME#	Ground	34	33	Ground	PMC-RSVD	34
35	Ground	IRDY#	36	35	TRDY#	+3.3V	36
37	DEVSEL#	+5V	38	37	Ground	STOP#	38
39	Ground	LOCK#	40	39	PERR#	Ground	40
41	SDONE#	SBO#	42	41	+3.3V	SERR#	42
43	PAR	Ground	44	43	C/BE[1]#	Ground	44
45	V (I/O)	AD[15]	46	45	AD[14]	AD[13]	46
47	AD[12]	AD[11]	48	47	Ground	AD[10]	48
49	AD[09]	+5V	50	49	AD[08]	+3.3V	50
51	Ground	C/BE[0]#	52	51	AD[07]	PMC-RSVD	52
53	AD[06]	AD[05]	54	53	+3.3V	PMC-RSVD	54
55	AD[04]	Ground	56	55	PMC-RSVD	Ground	56
57	V (I/O)	AD[03]	58	57	PMC-RSVD	PMC-RSVD	58
59	AD[02]	AD[01]	60	59	Ground	PMC-RSVD	60
61	AD[00]	+5V	62	61	ACK64#	+3.3V	62
63	Ground	RE064#	64	63	Ground	PMC-RSVD	64

CHAPTER 3
CONFIGURING AN XMP-ANALOG MOTION CONTROL SYSTEM

Introduction

This chapter describes basic encoder, amplifier, and limit signal connectivity via 68-pin VHDCI (Very High Density Cable Interconnect, also known as SCSI-5) receptacles for the XMP-PCI, XMP-CPCI-6U and XMP-CPCI-3U controllers. The XMP-CPCI-6U and XMP-CPCI-3U are also offer a rear I/O backplane connection option, which is described in Chapter 4.

VHDCI Connectors



I/O via VHDCI Receptacles

STC-136 Terminal Blocks

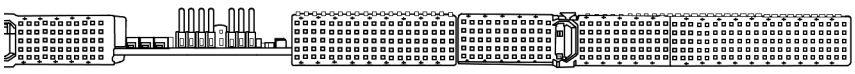
MEI offers STC-136 terminal blocks to assist in connecting controllers to drives via VHDCI. STC-136 terminal blocks utilize preconfigured cables from VHDCI receptacles. Each cable services one 68-pin VHDCI connector; each STC-136 can service two cables. The STC-136 terminal block provides a simple method for connecting hardware to your controllers using conventional screw terminals. Wires are connected to the terminal by inserting them into slots on the terminal block and tightening lug screws. For ease of use, numbering is identical between VHDCI receptacle pins and STC-136 terminals.

Some users prefer alternatives to the STC-136 for connecting to the XMP controller's VHDCI receptacles. If only a subset of lines are being used, more compact hardware may be preferred. Regardless how your connection scheme is configured, you may use the VHDCI pin-out information in this chapter to design your system.

I/O via Backplane Connection

Some CPCI-6U and -3U controllers perform input-output (I/O) via their backplane connectors. For more information regarding backplane analog I/O on CPCI form factors, please see Chapter 4 in this manual.

Backplane Connectors (see Chapter 4)



+5 and +24 Volt I/O

XMP controllers accomodate +5VDC or +24VDC optoisolated inputs. Resistors installed on each controller determine which I/O voltage it is configured for; otherwise, they are electronically identical. To determine which voltage your controller is configured for, look for the small, white label.

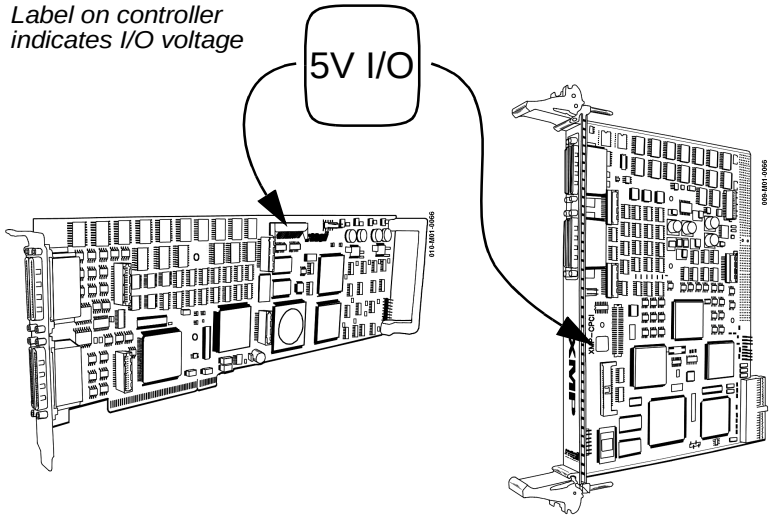
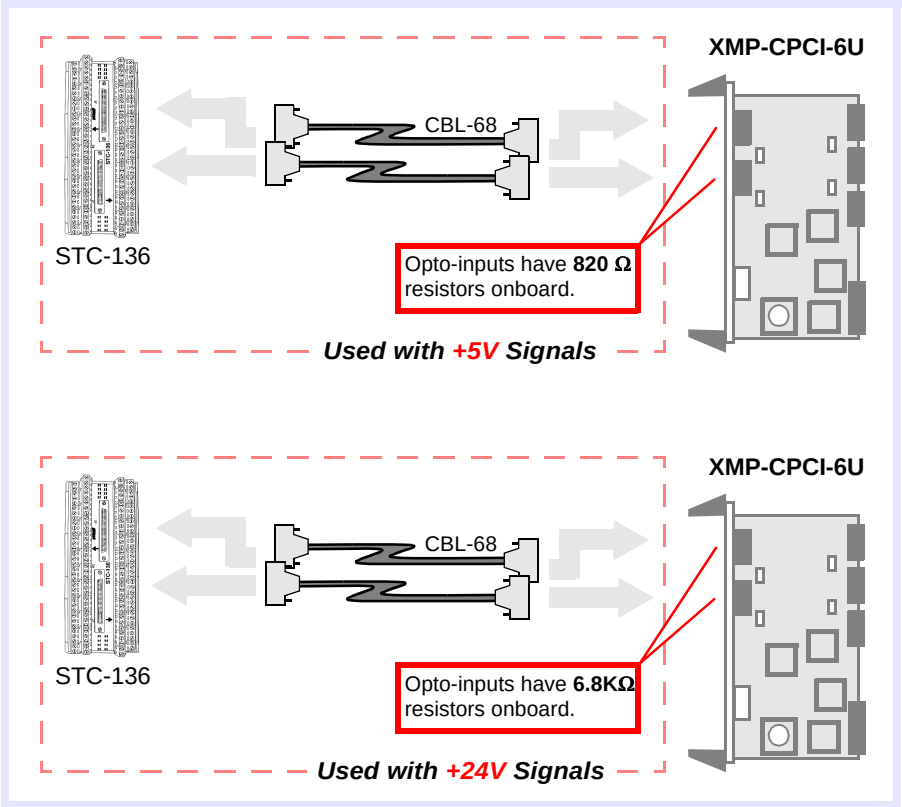


Figure 3-1. XMP configurations: +5V and +24V.



CAUTION! Under most conditions, motor axes are NOT freely interchangeable with each other! Always check the compatibility of wiring BEFORE plugging an axis into the connector of a different axis. Otherwise, damage to components and/or the controller may result.

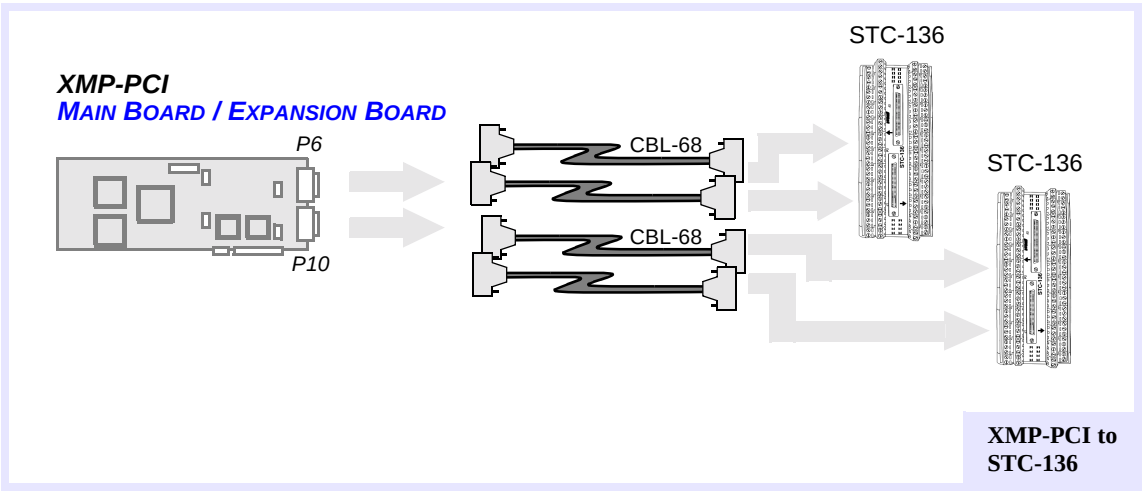
Pay attention to the use of power and ground returns. AGnd should be used for the analog inputs. Digital system grounds (Gnd) are provided for use as a system ground reference. These grounds (AGnd, Gnd) are all tied to a single point on the XMP board.

In-line, fused 5 VDC power is provided for external encoders. Be careful to estimate the additional 5 VDC current required, and the voltage drop across the cables.

Using STC-136 Terminal Blocks

Terminal blocks organize your system wiring, making them especially helpful for lab development. The STC-136 terminal blocks route each CBL-68 pin to a screw terminal on a DIN rail, (Phoenix Contact base). Each STC-136 provides connection to four (4) axes of I/O (a full motion block), and works with both 24 VDC and 5 VDC systems.

Figure 3-2. Each STC-136 provides connectivity for two (2) axes.



XMP-PCI to STC-136

Please note that terminal numbers for the STC-136 are matched one-to-one with VHDCI connector pins. (For example, Pin #1 on the VHDCI is the same as Terminal #1 on the STC-136, and so on.) Therefore, even if you choose not to use STC-136 terminal blocks for connection, you may still use the pin-out information in these chapters to configure your own connectors.

For full STC-136 wiring and pinout diagrams, see Chapter 4.

Connect to Analog Servo Amplifiers

NOTE: Opto-Isolation of Outputs

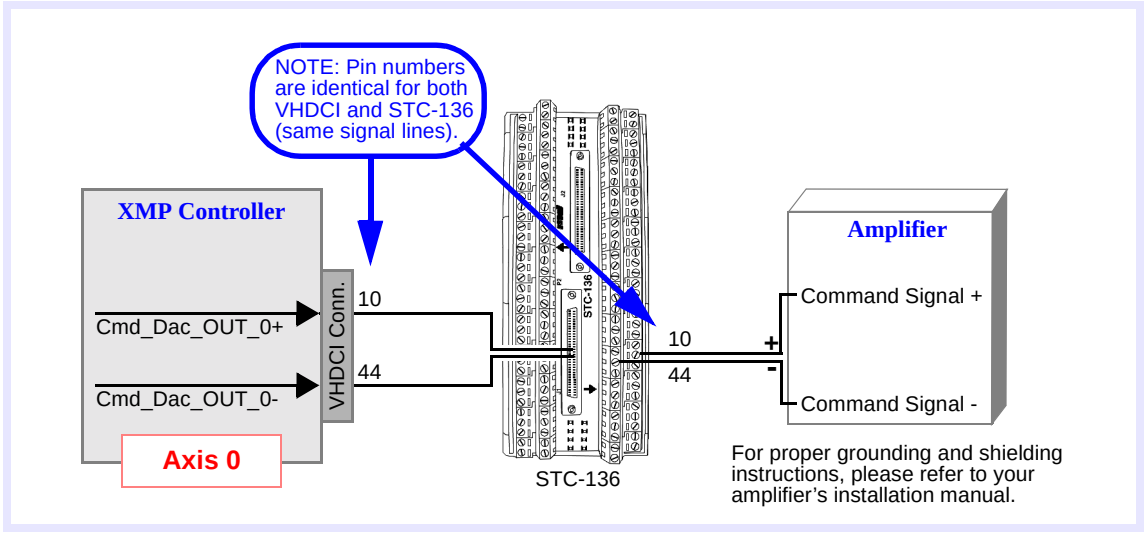
A number of controller outputs are optically isolated to protect both the controller and the drive amplifier, such as **HOME LIMIT** and **AMP FAULT**.

The XMP's internal logic only controls current to the opto-transistor, thus presenting a “conducting” state between the collector and emitter (of the opto-transistor). Your external wiring determines whether an active low or an active high is presented at the input to your drive.

Note	To maintain electrical isolation between the XMP and external I/O, the power and ground connections should be from an external power source, and should not be tied to the XMP’s power or ground connections.

Command Outputs (DAC) Wiring

Figure 3-3. Connections for command outputs.



Drive Enable Wiring

Definition of Fail Safe Wiring

One of the first questions motion system designers must address is how to wire the AmpEnable to a drive for fail safe operation. A fail safe drive enable circuit is one that is disabled when either the controller loses power or control lines are cut.

MEI strongly recommends avoiding any configuration where loss of power leads to a runaway axis. In many cases, the use of automatic, dynamic braking is recommended for high-mass, high-speed loads.

Wiring and Drive Behavior

The actual behavior of a drive enable circuit depends upon two factors:

- The drive's internal configuration.
- How the Amp Enable line is wired between the XMP controller and the drive.

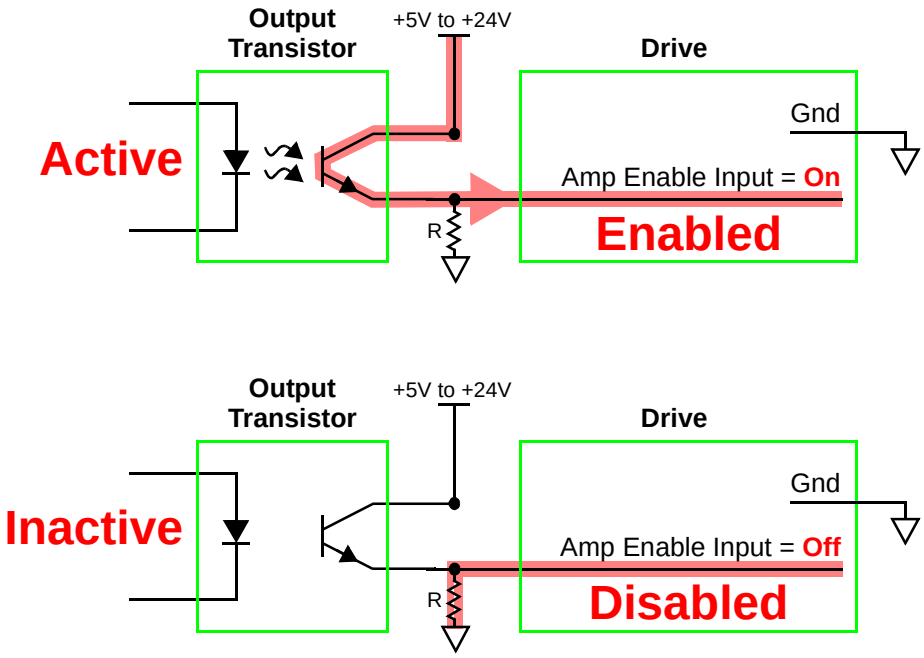
Two examples of fail safe wiring are shown below. They do not represent the only correct way to configure drives, but they can be used as guides.

Example 1: Active High

In the case of an active high drive enable input, the drive is “enabled” when the Amp Enable signal is high.

The diagram below shows how to connect a fail-safe circuit. Normally, the controller’s output transistor is inactive. When the transistor is inactive (OFF), the resistor pulls Amp Enable Input to ground, disabling the drive. When the transistor is active (ON), the Amp Enable Input is pulled up to +5V or +24V, enabling the drive.

Figure 3-4. Active high wiring.

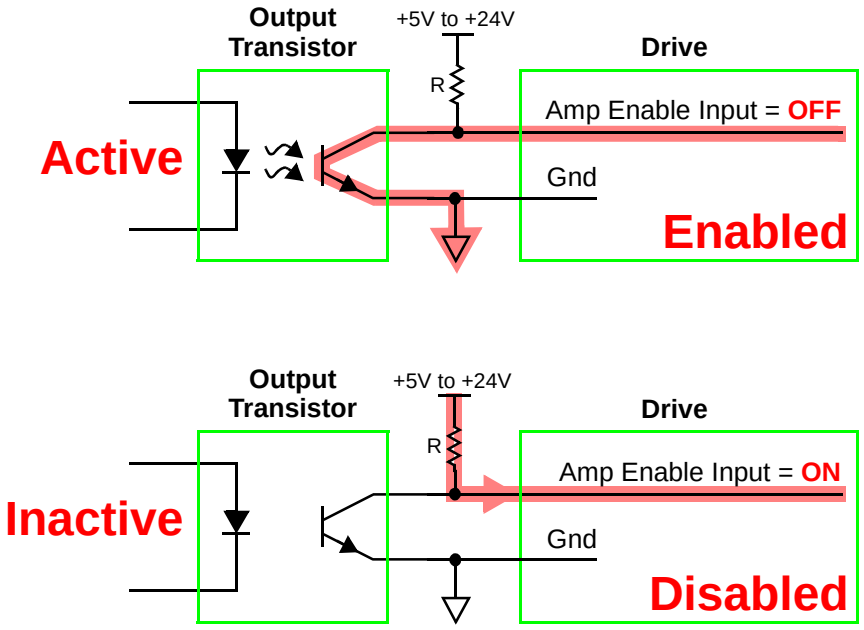


Example 2: Active Low

In the case of an active low drive enable input, the drive is “enabled” when the Amp Enable signal is low.

The diagram below shows how to connect a fail-safe circuit. Normally, the controller’s output transistor is inactive. When the transistor is inactive (OFF), the resistor pulls the Amp Enable Input up to +5V or +24V, disabling the drive. When the transistor is active (ON), the Amp Enable Input is pulled to ground, enabling the drive.

Figure 3-5. Active low wiring.

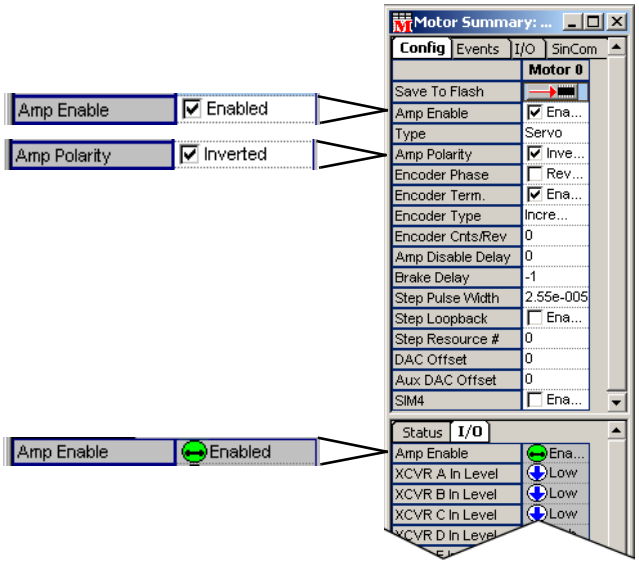


Setting Amp Enable and Polarity from Motion Console

When operating drive amplifiers with Motion Console, you must indicate the following:

- the state of the amplifier: **Enabled** or **Disabled** (unchecked)
- the polarity of the amplifier: **Inverted** or **Normal** (unchecked)

This is done by using the **Amp Enable** and **Amp Polarity** parameters within the **Motor Summary / Config** tab page:



The **Amp Enable** parameter is for use with drives equipped with enable lines. On such drives, the amplifier responds to commands *only* when the **Amp Enable** parameter is set to **Enabled**. If **Amp Enable** is not enabled, the amplifier will not respond to commands.

For fail-safe operation, the Amp Enable logic is determined by the wiring between the controller and drive. In this case, the Amp Polarity should be set to **Inverted**.

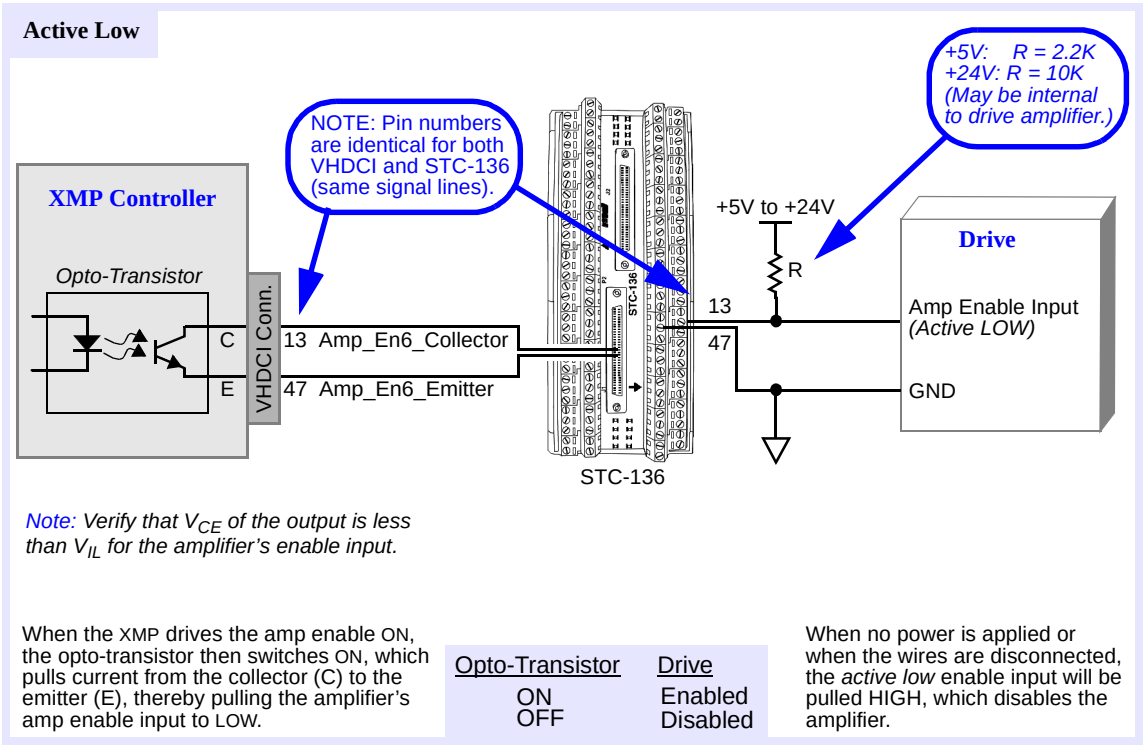
The **Amp Polarity** parameter tells the XMP controller whether the output transistor should be normally inactive, to disable the drive (**Inverted**) or normally active, to disable the drive (Normal). Only the “Inverted” Amp Polarity setting supports fail-safe wiring, and is strongly recommended.

Amp Enable Output

The optically-isolated Amp_Enable outputs provide control of the servo amplifier, allowing the XMP to disable the amplifier under fault conditions.

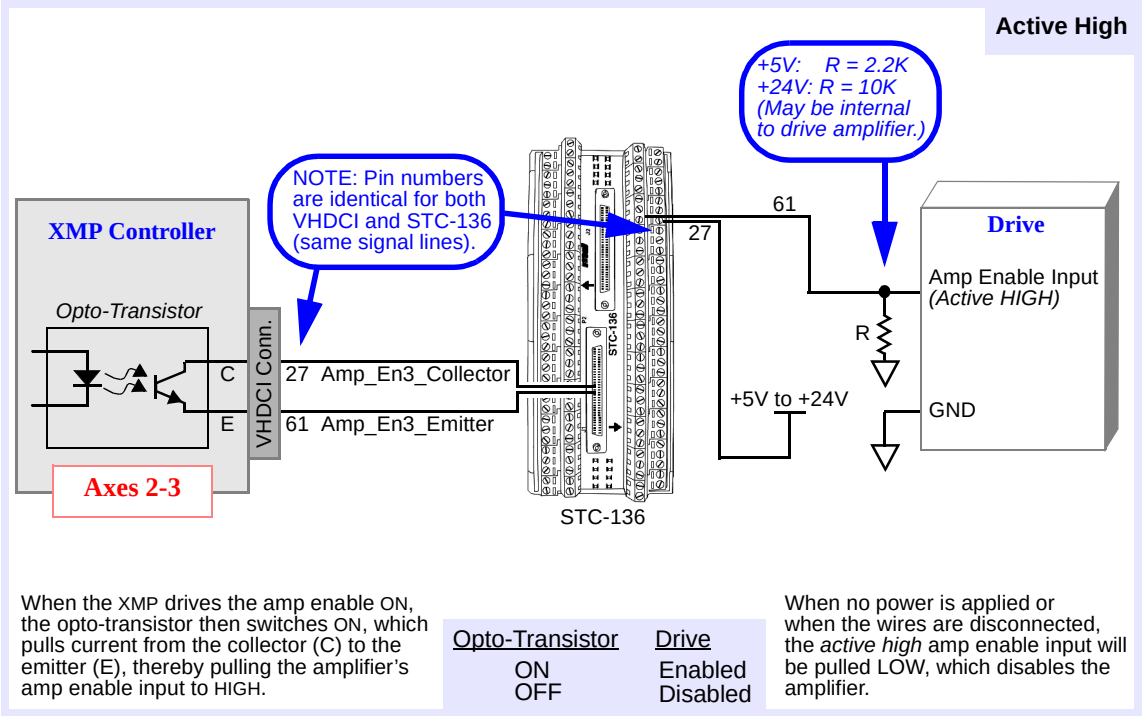
Active LOW Drive Enable Wiring

Figure 3-6. Connect Amp Enable output to drive (active LOW).



Active HIGH Drive Enable Wiring

Figure 3-7. Connect Amp Enable output to drive (+5/24V, active HIGH).



Amplifier Enabling from the MPI

Amplifier enabling is a crucial safety issue in the design of your application. Your system should be designed to anticipate the possibility of a sudden loss of power and/or connectivity to the controller and/or drive.

WARNING!

Make your Amp Enable circuit design SAFE!

When you design your Amp Enable circuit, design it such that when power to the XMP controller is **OFF**, the amplifier is **DISABLED**.

To make a full accounting of your motion system's safety features, you must also address the system's mecahnical dynamics with a disabled drive. (A disabled drive does *not* necessarily mean that all motion will be stopped!) In many cases, the use of automatic, dynamic braking is recommended for high-mass, high-speed loads.

Determining whether an amplifier remains enabled or disabled with the controller powered OFF depends upon several factors:

- Does the drive *have* an Amp Enable line? (Not all drives do. *Check this!*) Is the drive's Amp Enable line internally configured as active-high or active low?
- How is the Amp Enable line wired between the controller and the drive? With power OFF to the controller, does wiring pull the drive's Amp Enable line HIGH or LOW?
- How is the Amp Enable line configured on the controller when it is powered? Is the Amp Enable output transceiver normally ON or OFF? Is the Amp Enable output inverted?
- Does the application software maintain the Amp Enable line in an enabled state when running? Is the controller's position error limit configured for Abort (disables the drive)? Does the application respond to states, disabling or enabling the drive?

Table 3-1. State logic for active high and active low drives.

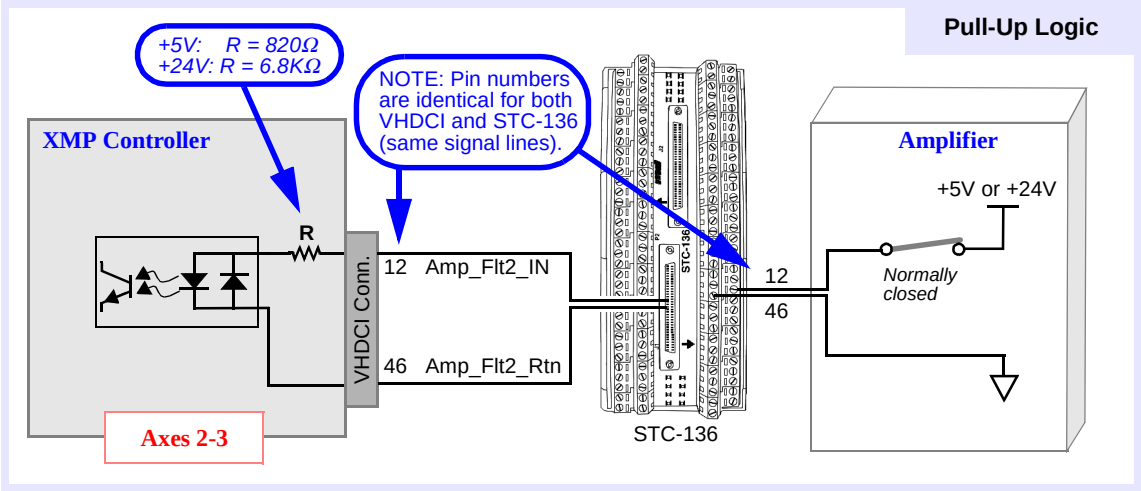
DRIVE	***WIRING***	***STATE***
How is the drive's Amp Enable line internally configured?	***With power to the controller OFF, current to the drive's Amp Enable line is...***	***When power to the controller is OFF, the drive is...***
Active High	High	Enabled
	Low	**Disabled***
Active Low	High	**Disabled***
	Low	Enabled

* Recommended configuration.

Amp Fault Input

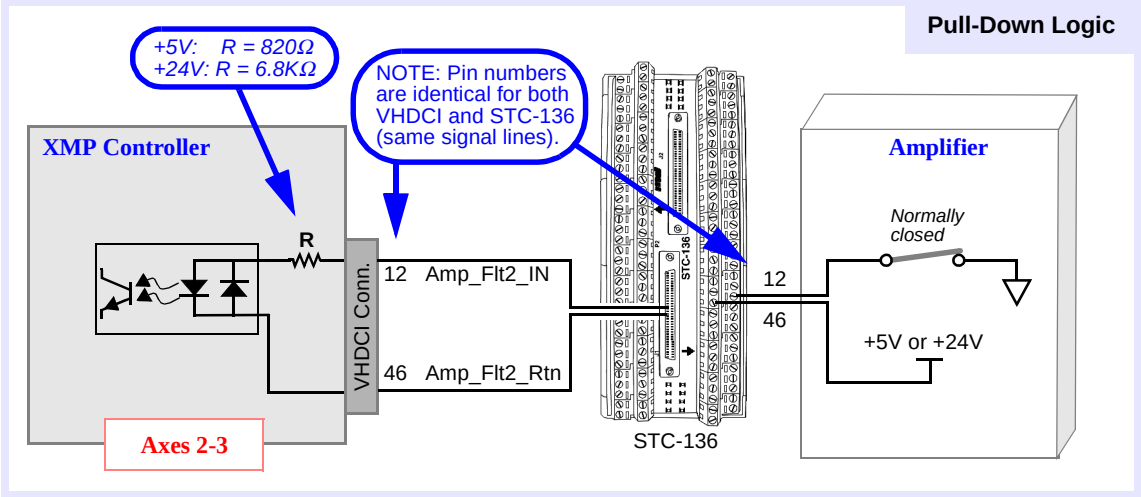
To Pull-up Logic

Figure 3-8. Connect Amp Fault input to amplifier (pull-up logic).



To Pull-down Logic

Figure 3-9. Connect Amp Fault input to amplifier (pull-down logic).



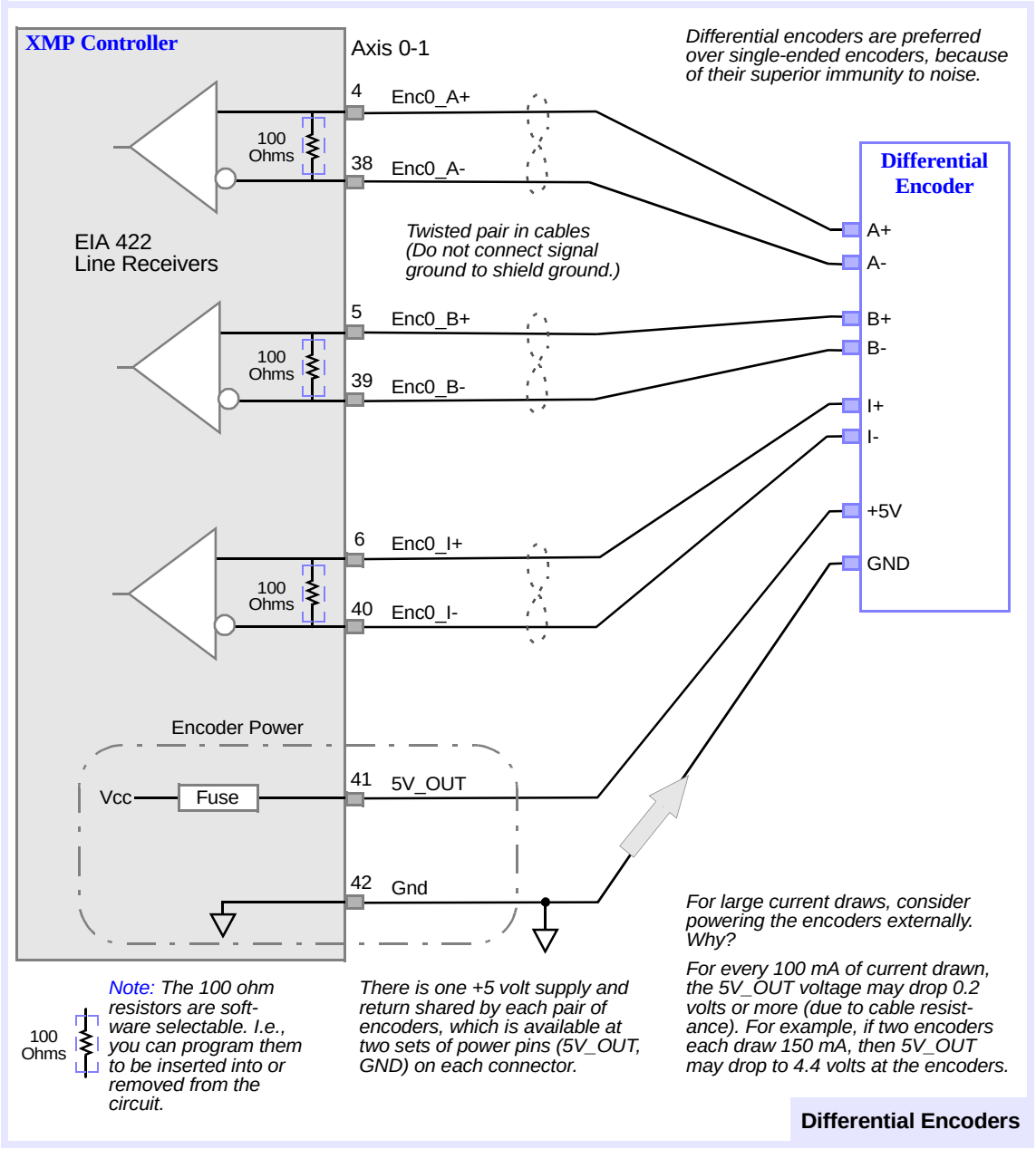
Connect to Digital Quadrature Encoders

Encoder Wiring

Note	The 100 ohm terminations to the receivers are *software-selectable*. If you use the *Scale Interpolation Module*, you must program the 100 ohm terminations off, i.e., remove them from the circuit. To learn more about encoder termination, please refer to *Application Note 206*.

Connect to Differential Encoders

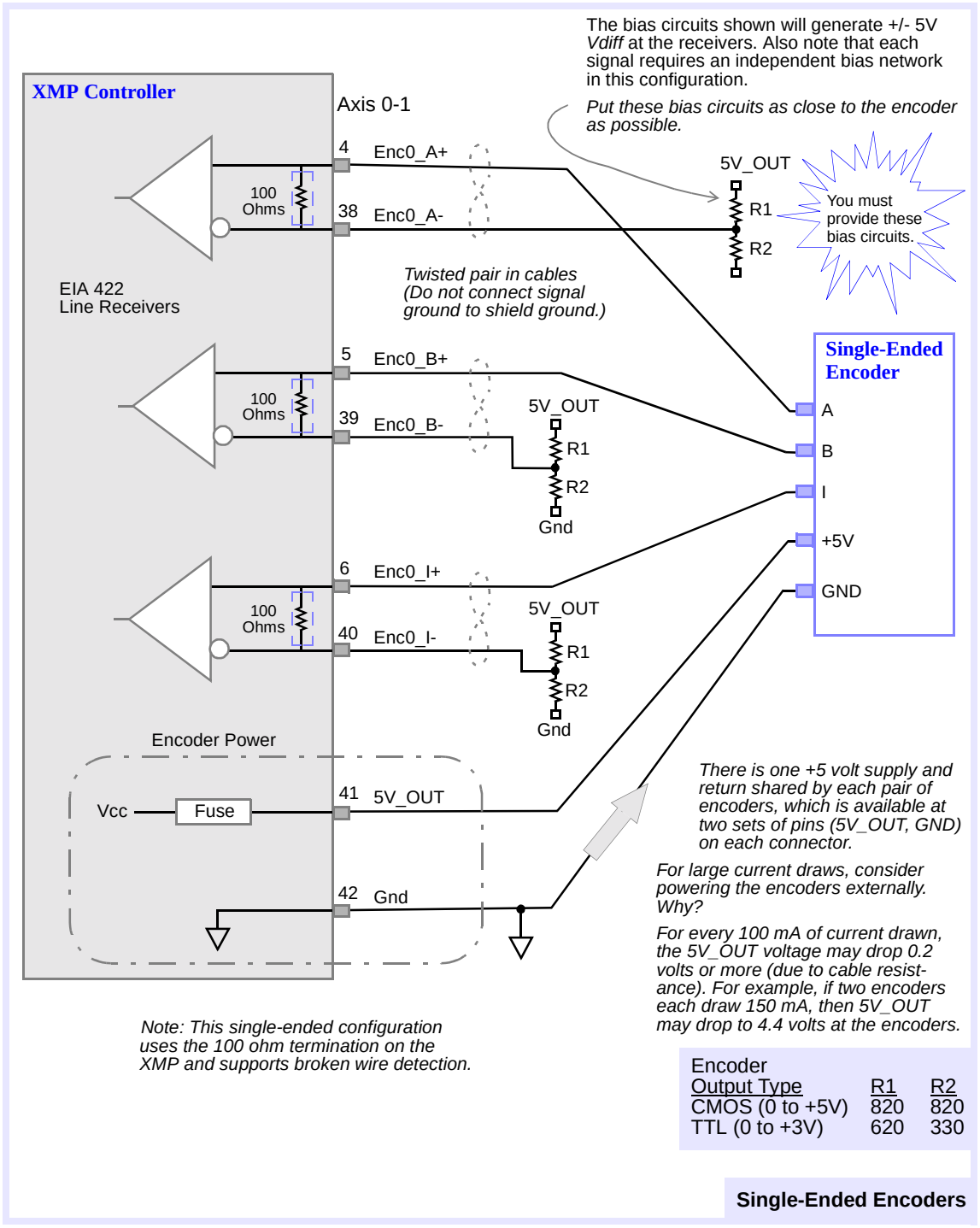
Figure 3-10. Connect to differential encoders (digital).



Connect to Single-ended Encoders

Note	To connect the XMP's differential line receivers to single-ended encoders, you must add a pair of resistors (R1, R2) in the configuration shown, to bias the negative end to the middle of the encoder's output voltage range.

Figure 3-11. Connect to single-ended encoders (digital).



Broken Wire and Illegal State Detection

The encoder inputs (channel A+, A-, B+, B-) are monitored by the FPGA (an on-board logic component). The encoder inputs are sampled at 20mHz. By default, a digital filter is applied to each encoder input. This digital filter requires that an encoder input (channel A+, A-, B+, B-) be stable *for 4 clock cycles* (200 nanoseconds) before a transition is recognized, i.e., encoder input states lasting less than 4 clock cycles are filtered out.

Connect System I/O

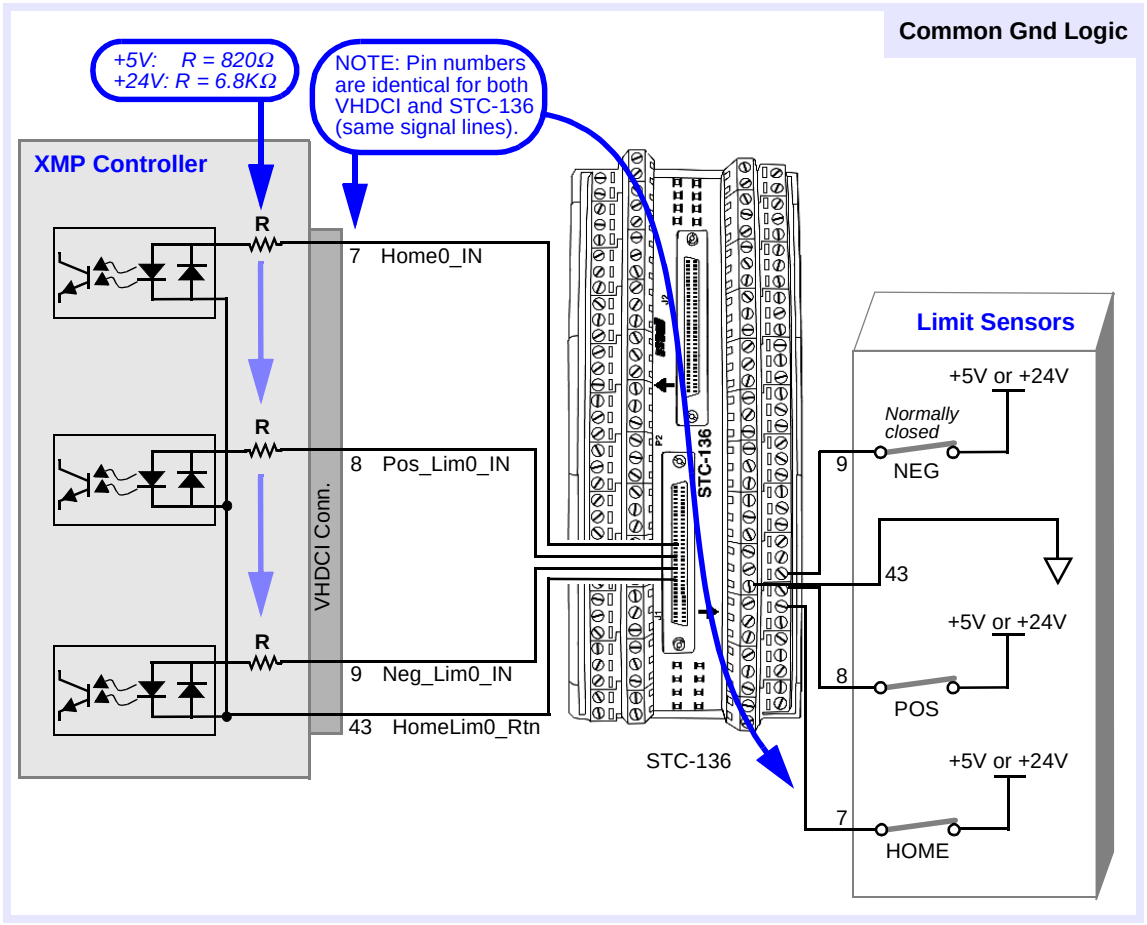
System I/O includes:

- Dedicated inputs per axis:
  - Positive and Negative Limit inputs
  - Home input
- Global system inputs (one per XMP system)
  - System E-Stop input
  - System Reset input
- User I/O (opto-isolated)
  - Individually configurable as input or output
  - There are four user I/Os per 4-axis Motion Block, (or eight user I/Os per board). Each user I/O is individually configurable as input or output.

Limit and Home Sensors

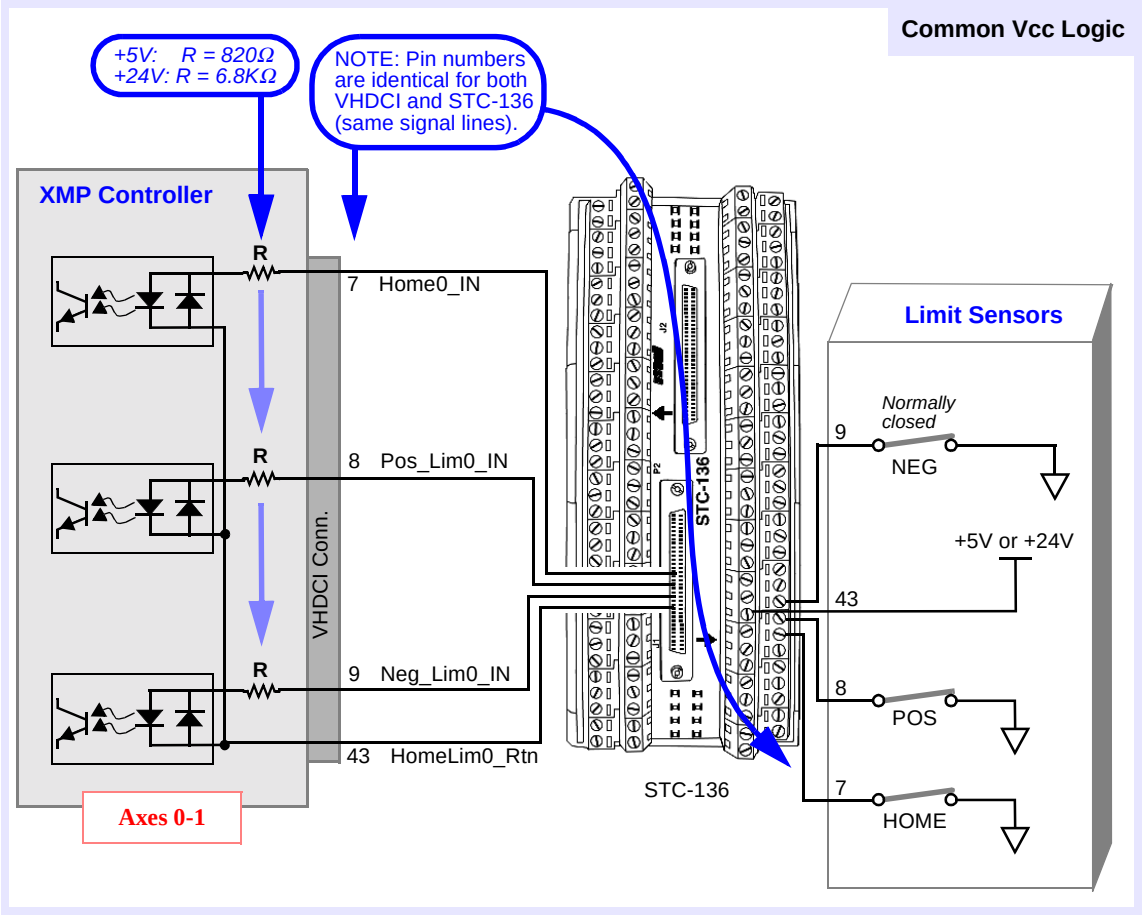
Common Ground Logic

Figure 3-12. Connect home and limit sensors (common ground logic).



Common Vcc Logic

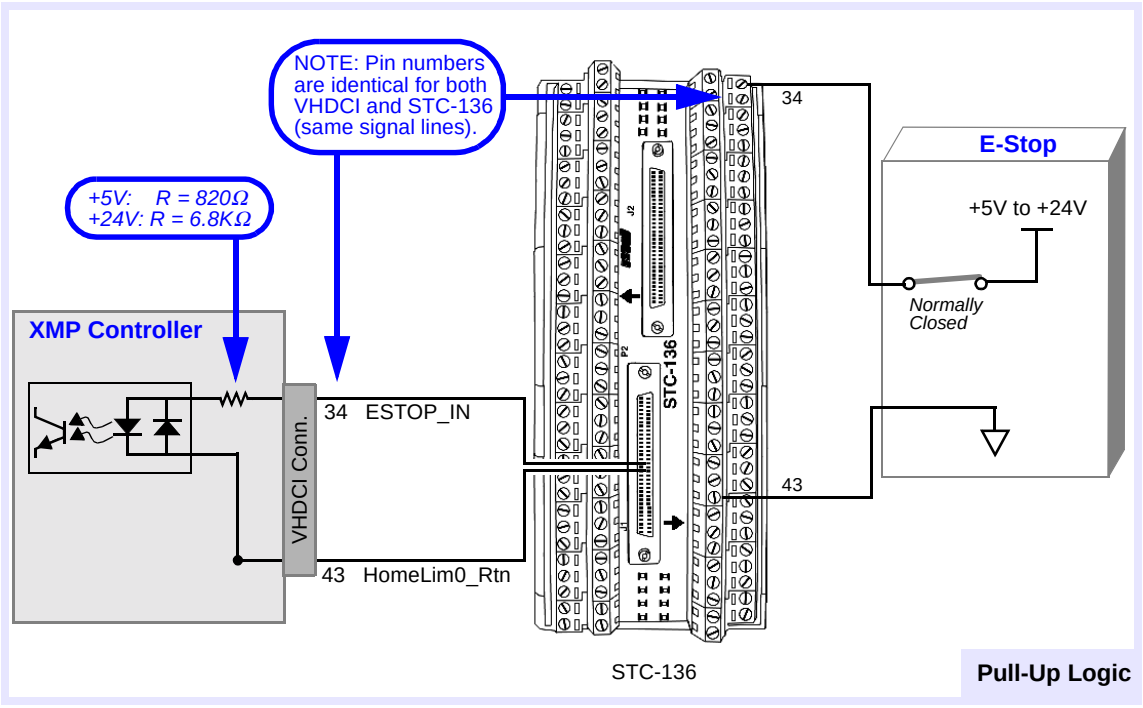
Figure 3-13. Connect home and limit sensors (common Vcc logic).



System E-Stop Input

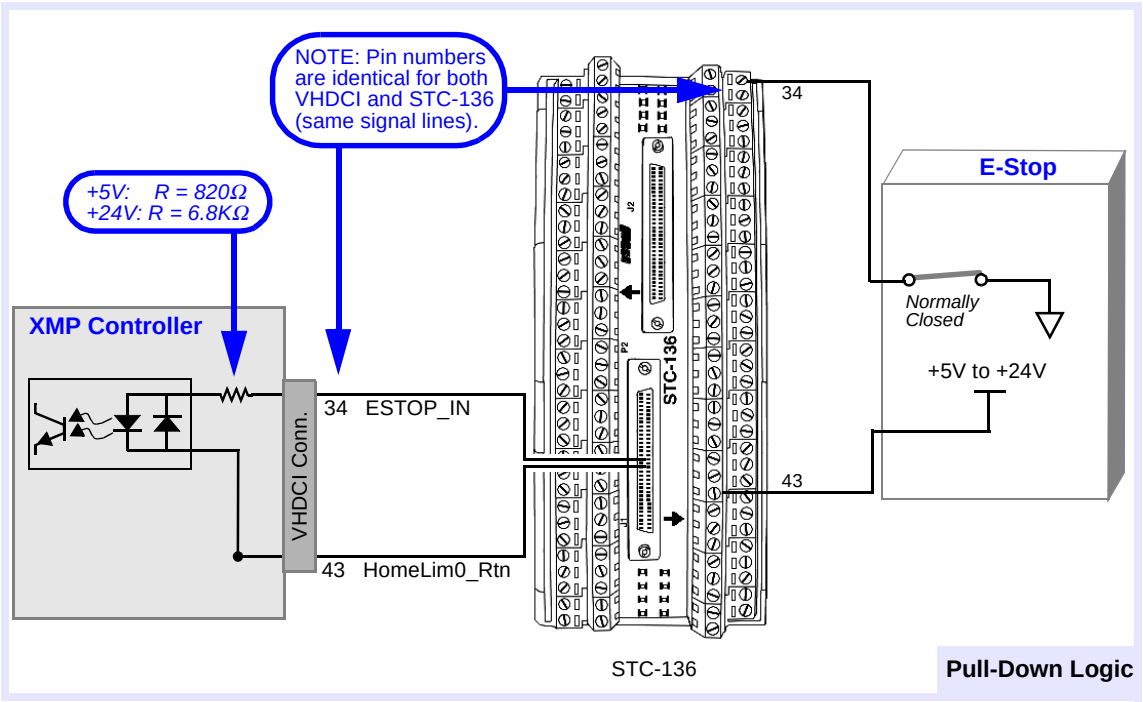
To Pull-up Logic

Figure 3-14. Connect E-stop input to pull-up logic.



To Pull-down Logic

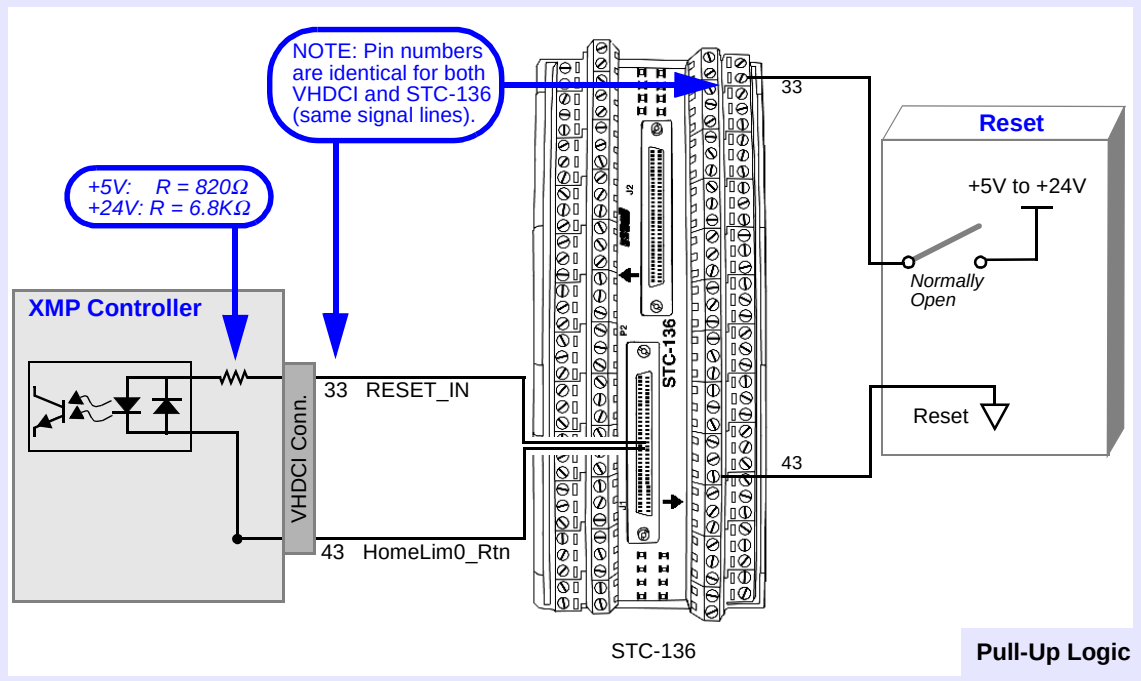
Figure 3-15. Connect E-stop input to pull-down logic.



System Reset Input

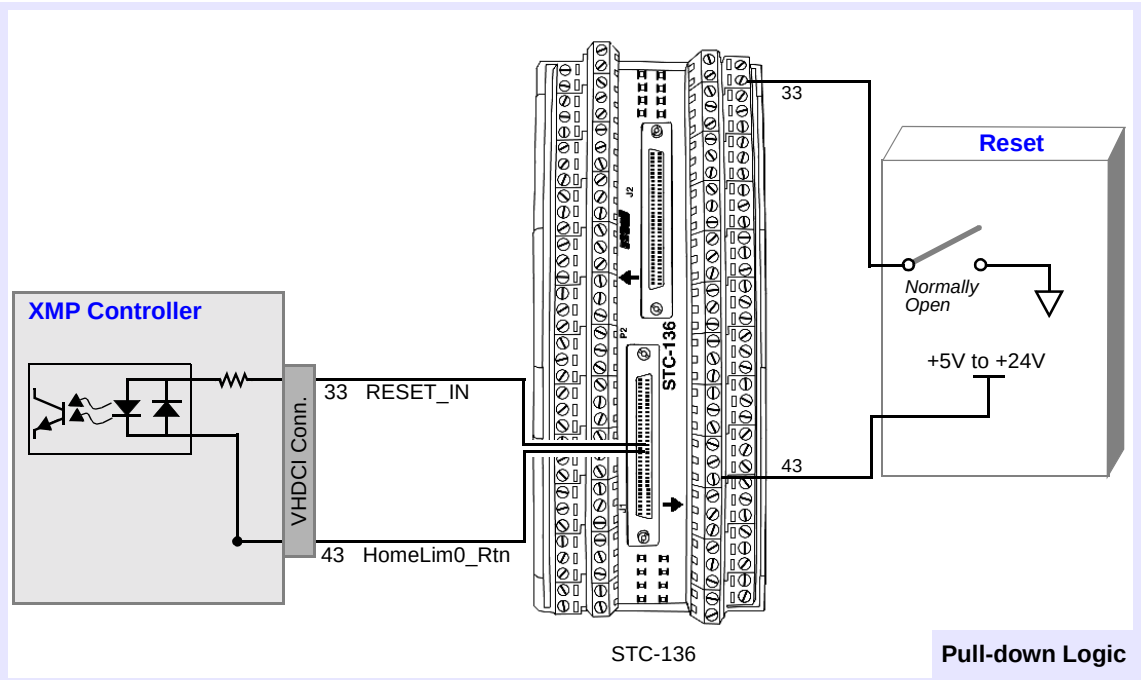
To Pull-up Logic

Figure 3-16. Connect system reset input to pull-up logic.



To Pull-down Logic

Figure 3-17. Connect system reset input to pull-down logic.



Opto-isolated User I/O

There are four (4) user I/Os per 4-axis motion block, and each user I/O is individually configurable as input or output.

Figure 3-18. Connect user I/O input to pull-up logic.

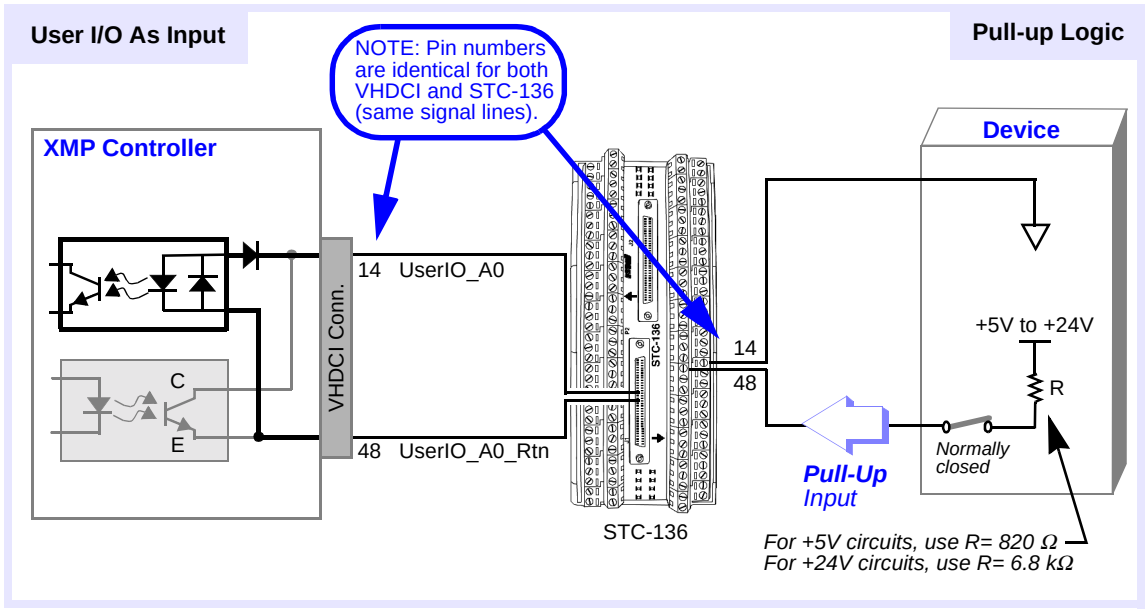
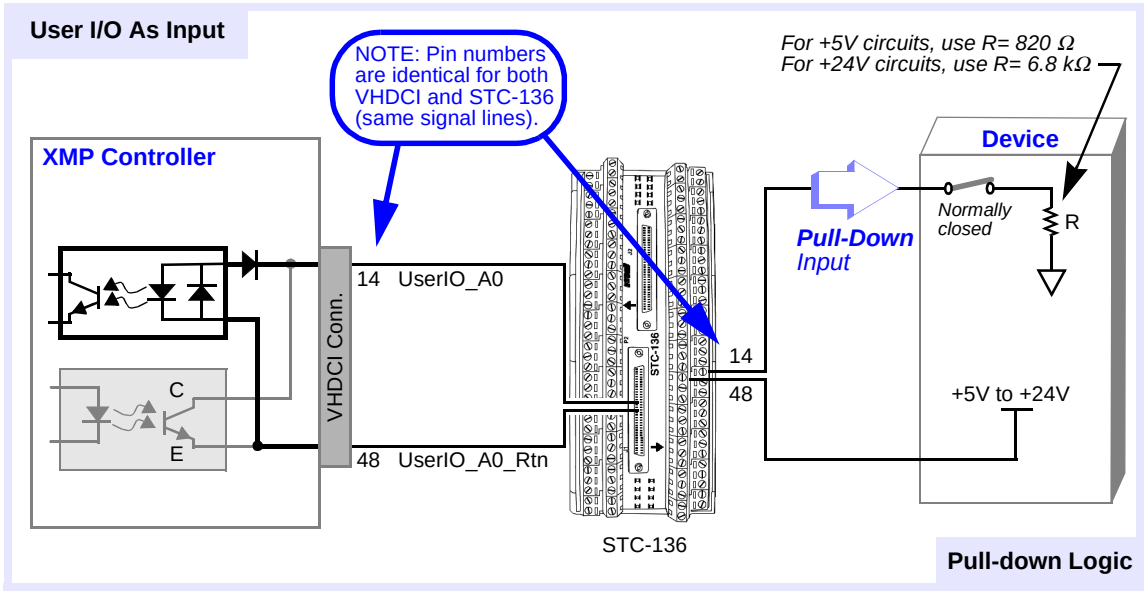


Figure 3-19. Connect user I/O input to pull-down logic.



As User I/O Outputs

Figure 3-20. Connect user I/O as outputs to pull-up logic.

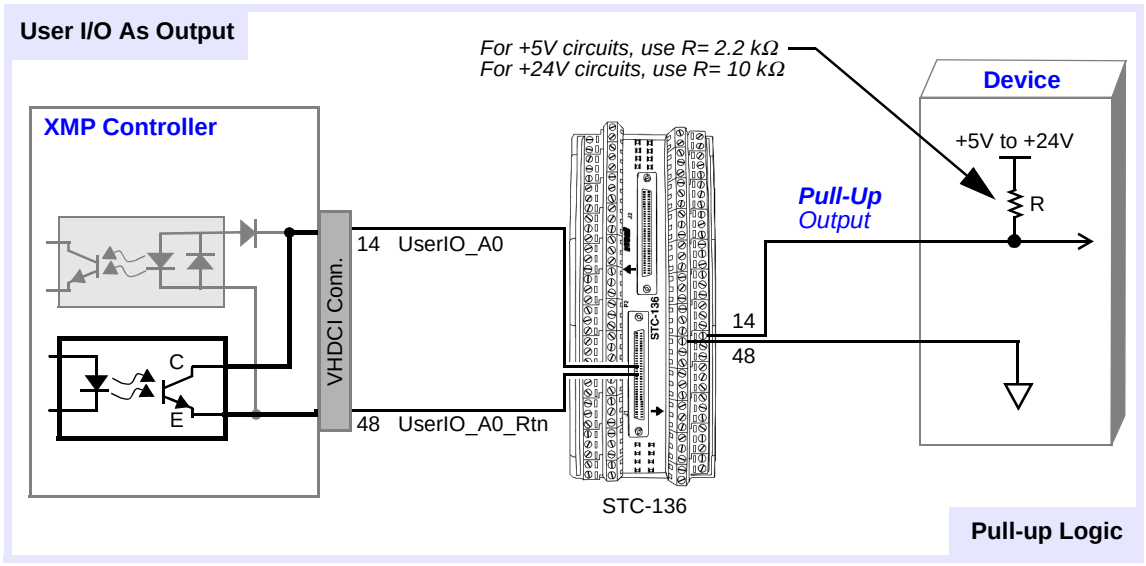
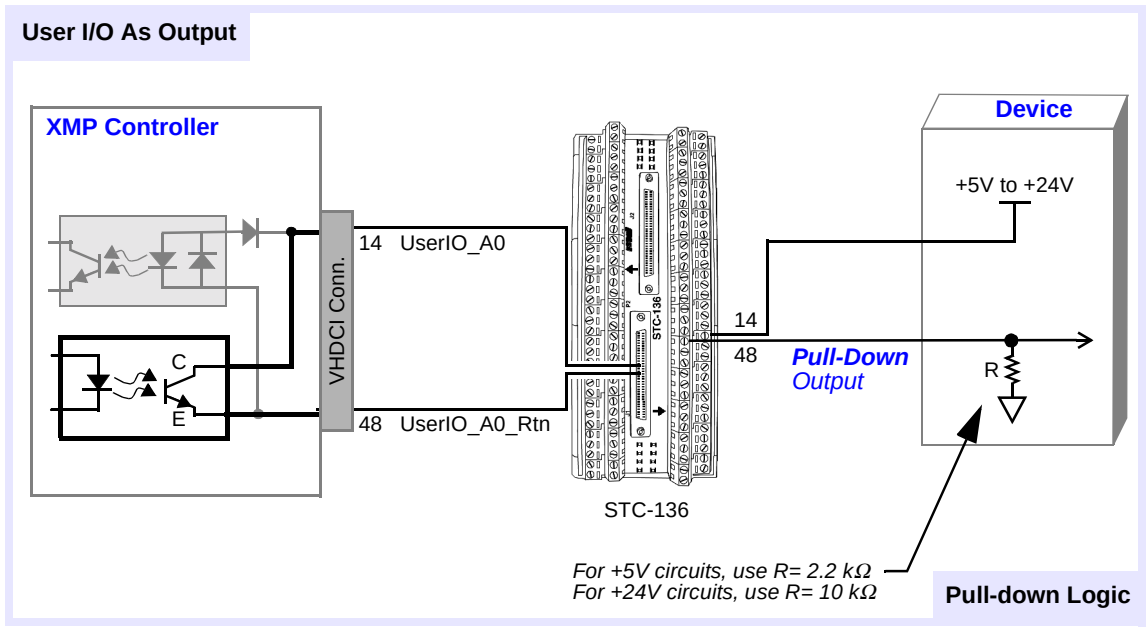


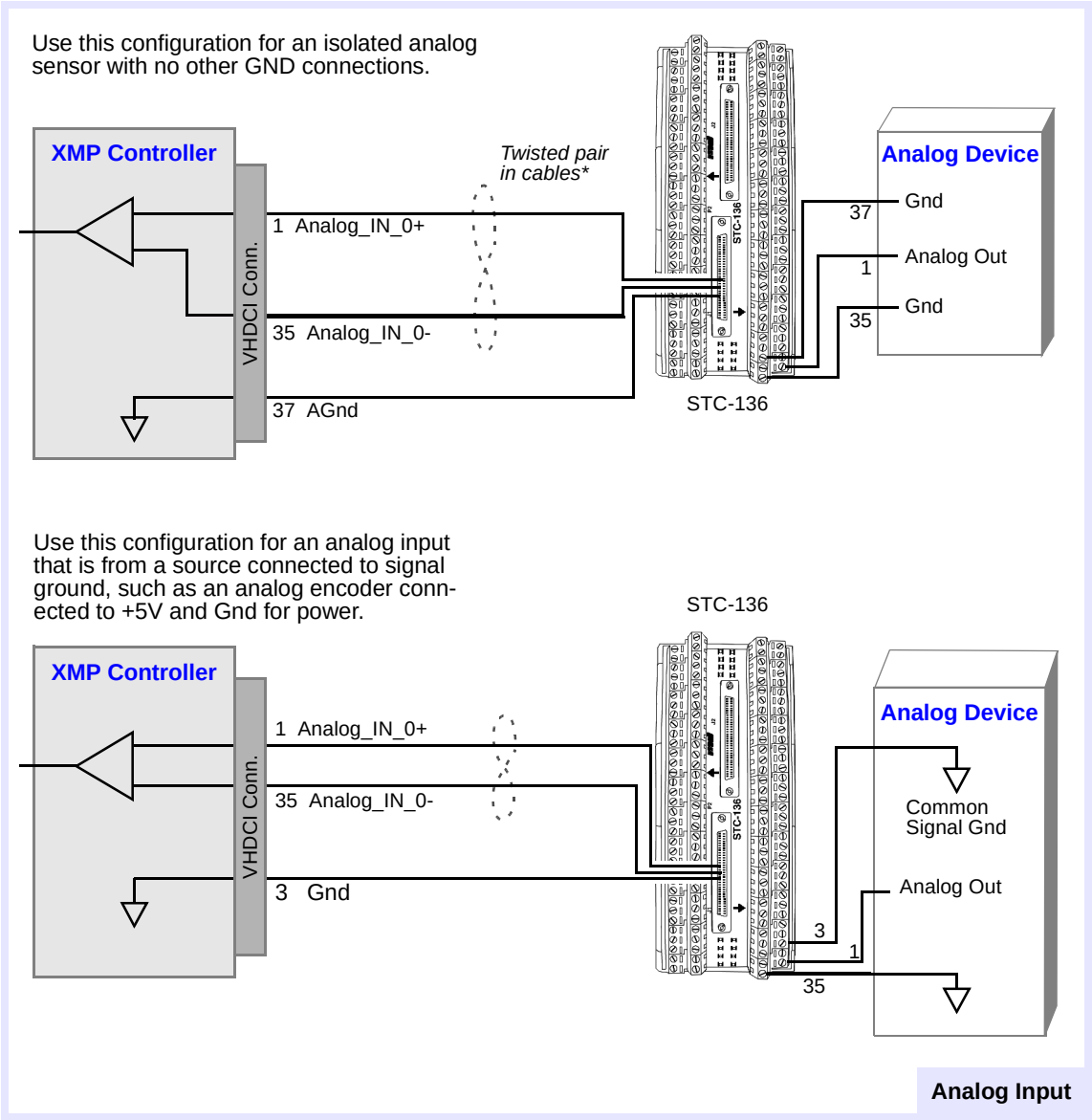
Figure 3-21. Connect user I/O as outputs to pull-down logic.



Analog Inputs

There are eight (8) analog inputs per XMP system.

Figure 3-22. Connect analog inputs.



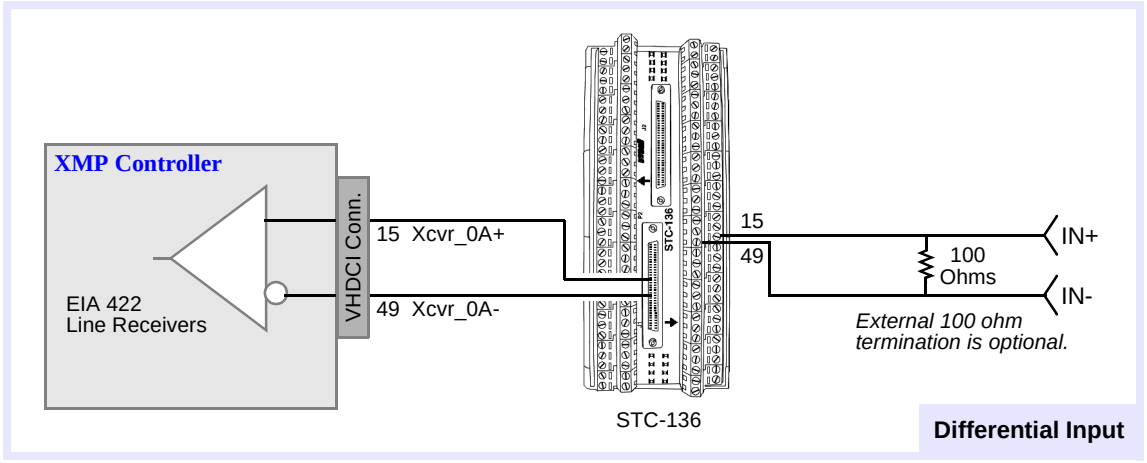
Analog Input

Connecting XMP Transceiver I/O as a Capture Input

There are twelve (12) I/Os for each 4-axis motion block.

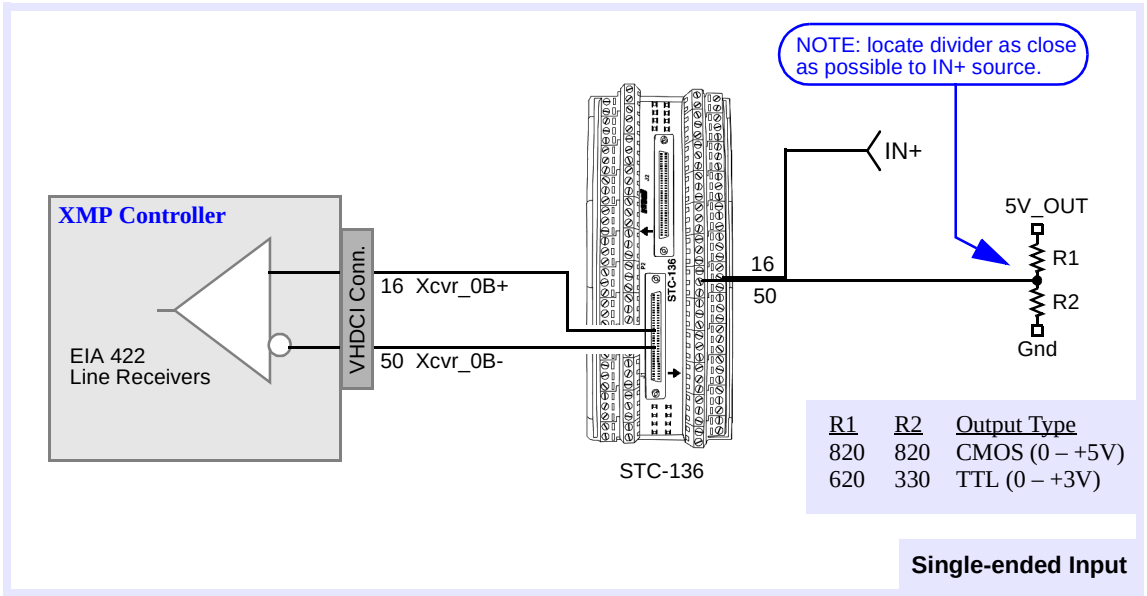
Connect Transceiver as Differential Capture Input

Figure 3-23. Connect transceiver as differential input.



Connect Transceiver as Single-ended Capture Input

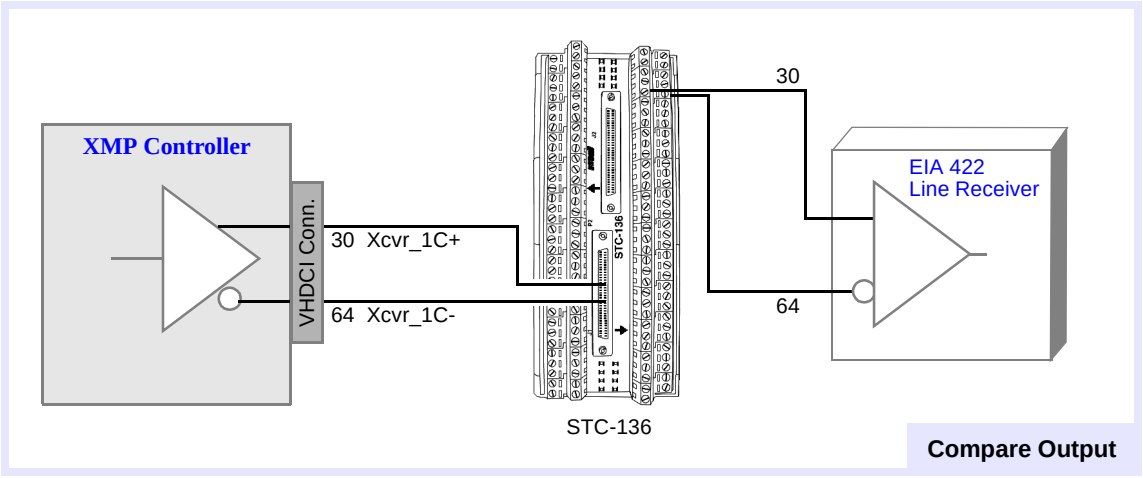
Figure 3-24. Connect transceiver as single-ended input.



Transceiver as a Compare Output

Connect Transceiver as Differential Compare Output

Figure 3-25. Connect transceiver as differential compare output.



Connect Two XMP Boards (12-16 Axes)

To create systems with more than eight (8) axes, use two XMP boards, with one board configured as the main, and the other board configured as the expansion.

Typical Connections

Figure 3-26. Main/expansion axis assignments: CPCI.

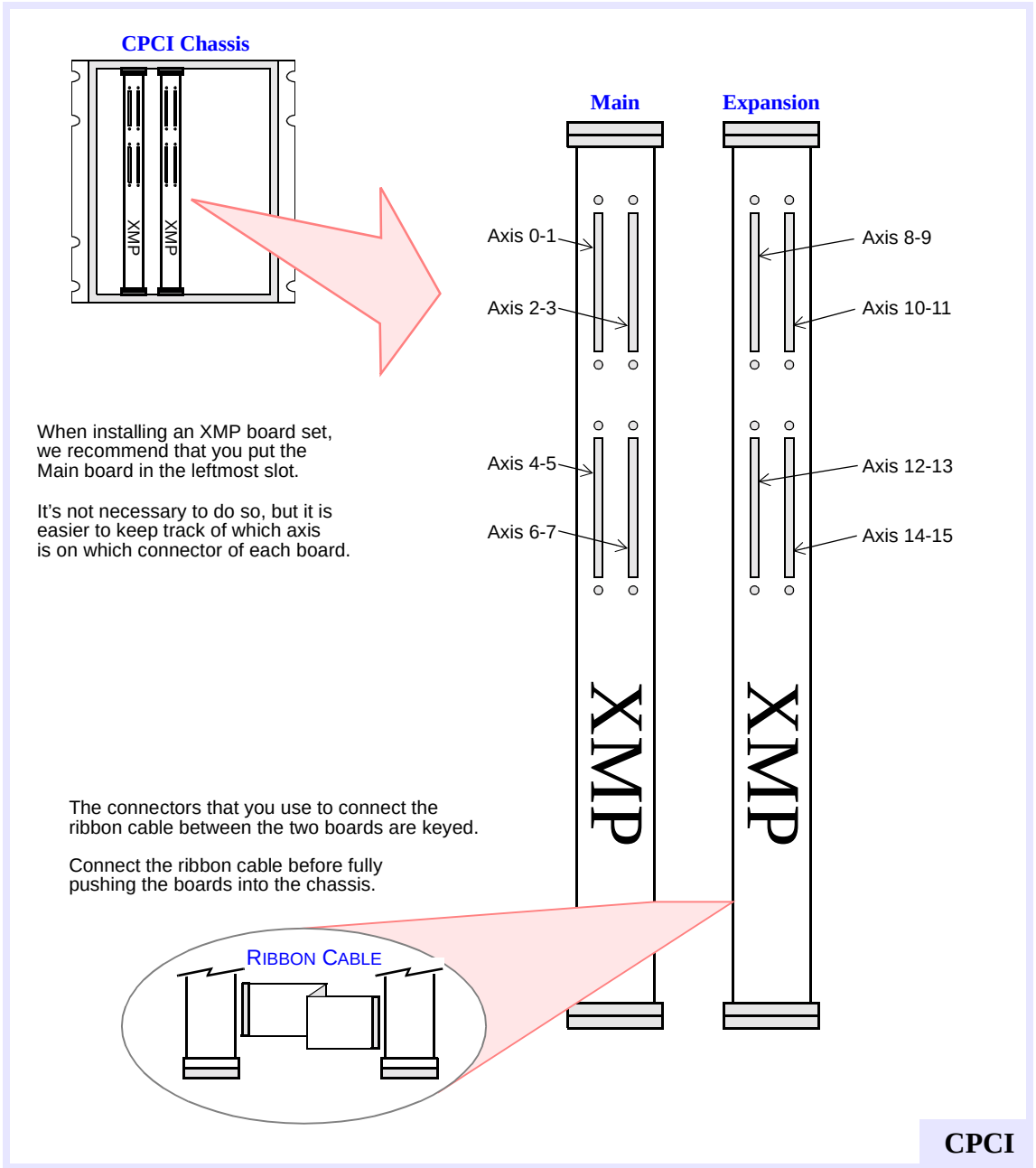
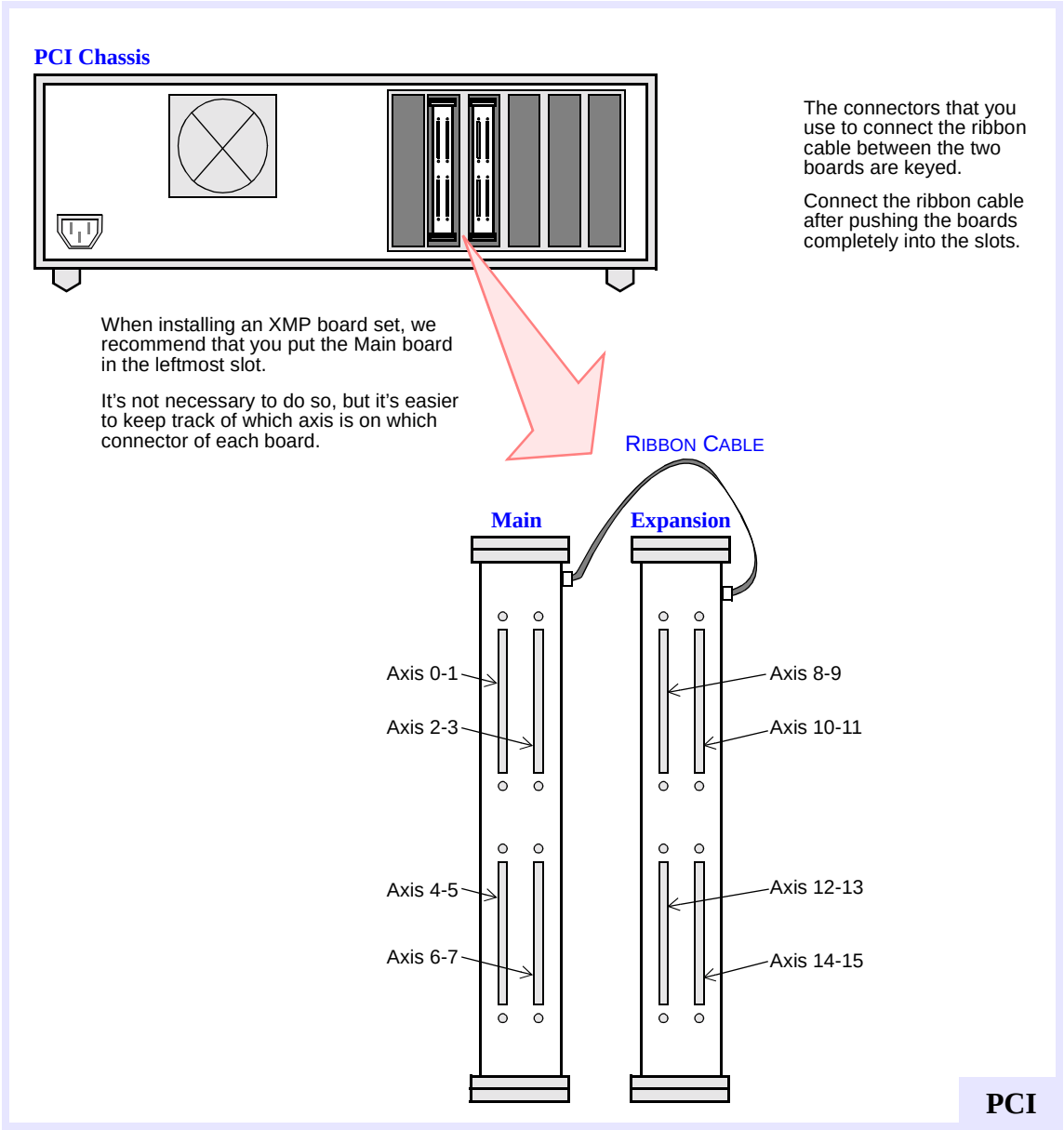


Figure 3-27. Main/expansion axis assignments: PCI.



Configuring Analog Connections

This section provides more general guidance for configuring analog controllers with various hardware (servo amps; encoders, etc.).

XMP-PCI, XMP-CPCI

Connections to Servo Motors

XMP-series controllers can control brush servo motors, brushless servo motors, or linear brush/brushless motors. Basic connections require an analog output signal (from the controller to the amplifier) and an encoder input (from the motor to the controller).

Most amplifiers support either Velocity mode (voltage control) Torque mode (current control) or both. The XMP controller can be used with either servo motor/amplifier package.

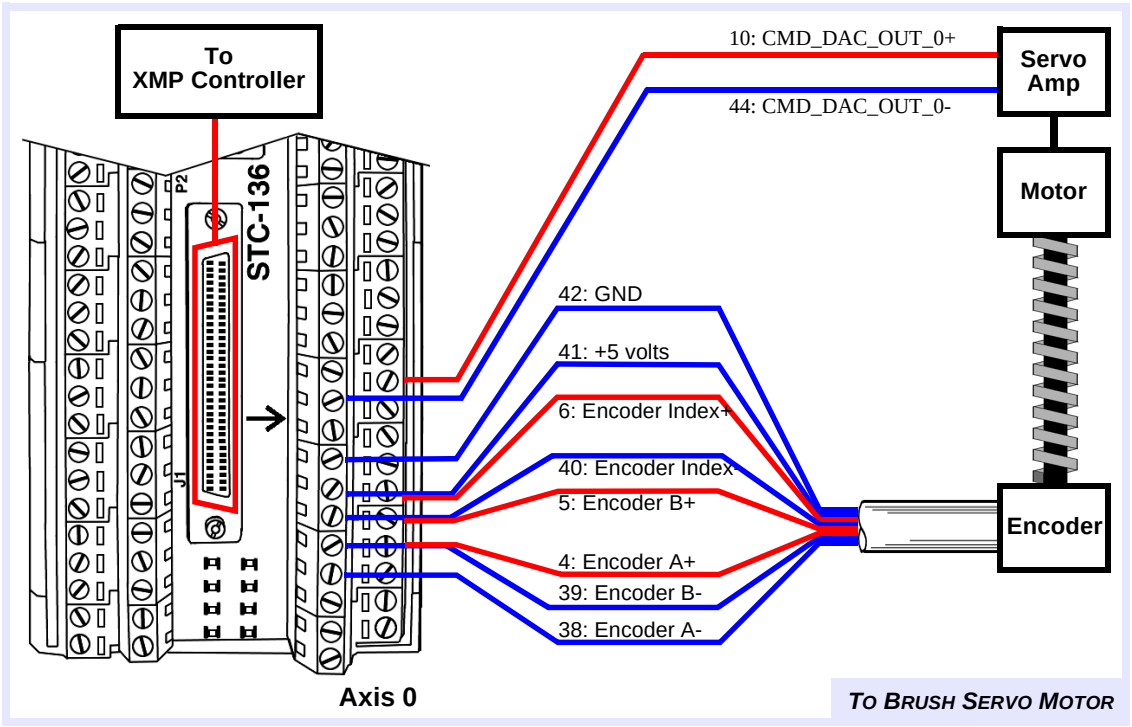
XMP-series controllers accept RS-422 compatible (0V to +5V, 40mA max) encoder input from either differential or single-ended encoders. Differential encoders are preferred due to their excellent noise immunity. The connections for a single-ended encoder are identical to a differential encoder except that references must be connected to channel A- and channel B-.

The controller reads the index pulse (either single-ended or differential ended). Typically, there is one index pulse per revolution of the encoder (rotary type), which can be used for homing. Encoder signals are read in quadrature. Every line on the encoder will produce a rising edge and a falling edge on channels A+ and B+ which is interpreted by the XMP controller as four encoder counts.

Brush Servo Motors

The minimum required connections to brush-type servo are: Analog signal (± 10V), +5V, Signal Ground, Encoder Channel A+, Encoder Channel B+. Typical connections for a brush servo motor with a differential encoder follow.

Figure 3-28. Typical brush servo motor connections.

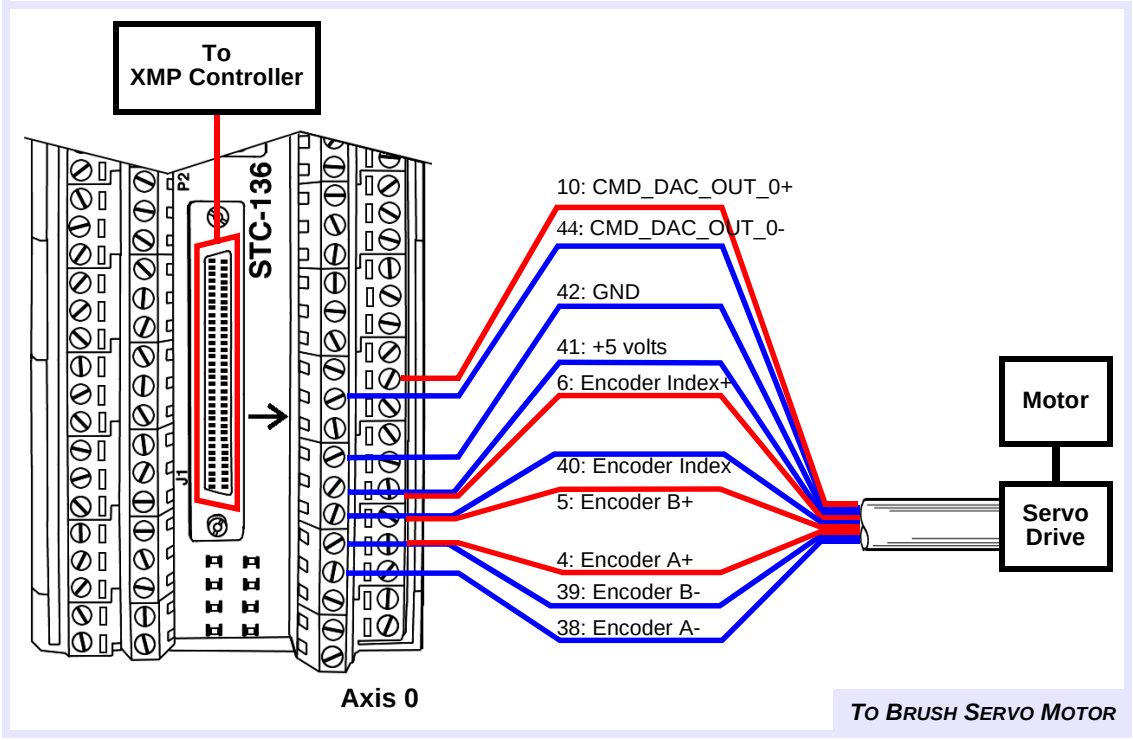


Note	**Any unused lines should be left unconnected.**

Brushless Servo Motors

Typical connections for a brushless servo motor with a differential encoder are:

Figure 3-29. Typical brushless servo motor connections.



Typical brushless servo motor drives of this configuration are “intelligent.” If single ended encoders are used, external references will need to be supplied.

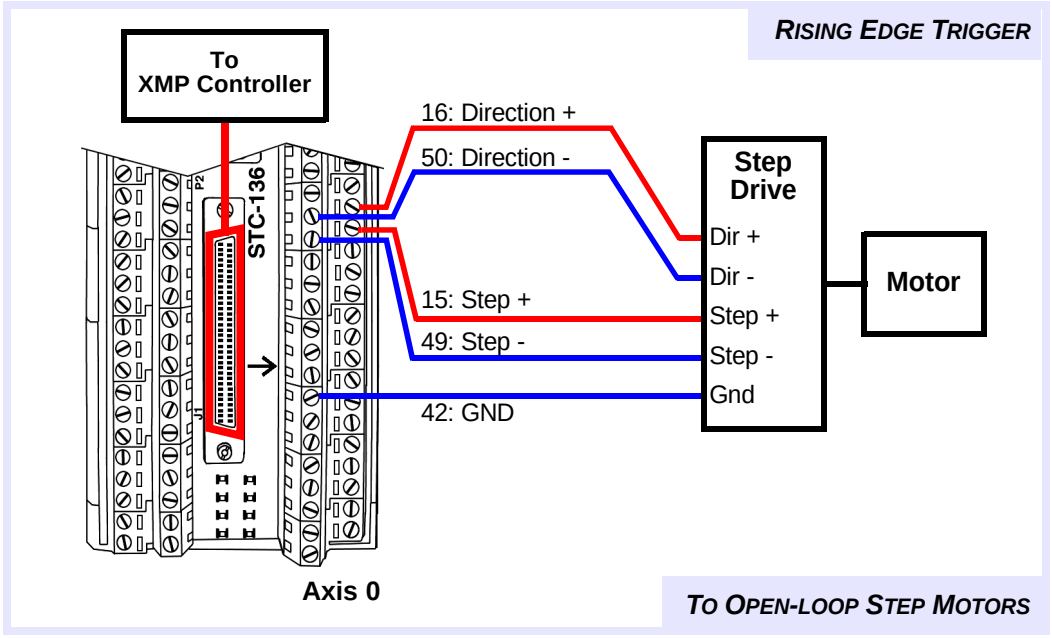
Step-and-Direction Controlled Servo Motors

Some brushless servos are controlled by step-and-direction pulses. With this scheme, the position information is communicated by step pulses, and the PID loop is handled internally by the drive itself.

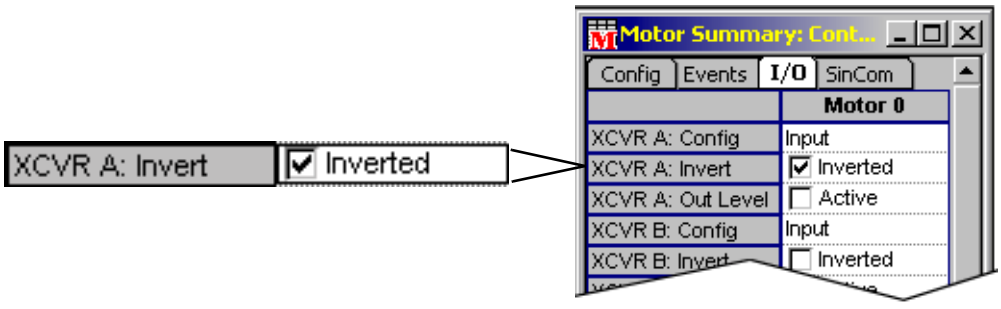
Open-loop Step Motors

XMP controllers can control step motors in open-loop (no feedback) configurations. The XMP can also be configured for STEP LOOPBACK. When this is enabled, the motor Position Error Limit and Axis Settling will depend on the output steps being read back into the actual position register. If STEP LOOPBACK is disabled, an external feedback device (e.g. encoder) will be needed for determining position error limits and axis settling. In either case, the output is generated regardless of actual position.

Figure 3-30. Typical open-loop step motor connections for step drives triggering on the rising edge.



For stepper drives that trigger on the falling edge, invert the step output transceiver. This can be done from Motion Console by enabling the **XCVR A: Invert** (or **XCVR B: Invert**, etc.) parameter in the **Motor Summary / I/O** page.



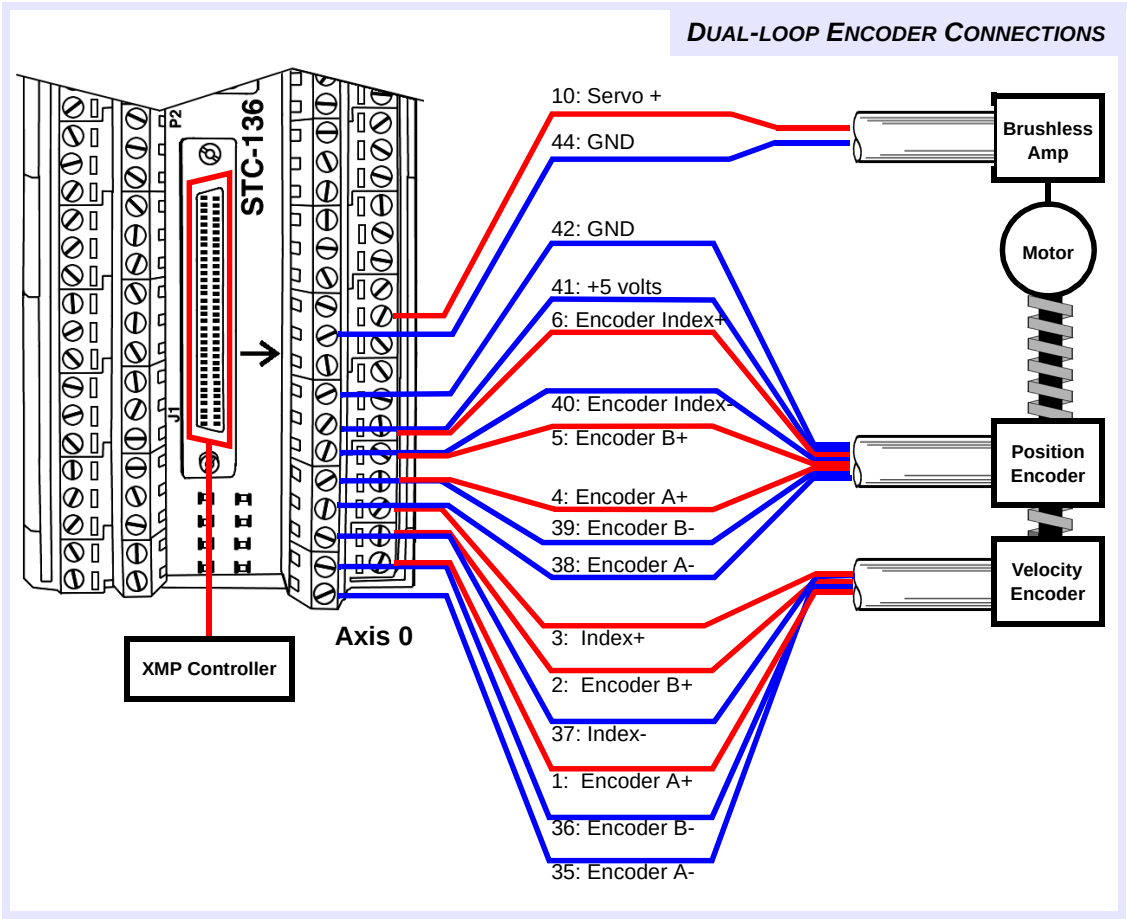
NOTE: to determine whether your drive triggers on the rising or falling step edge, consult the drive manufacturer's manual.

Connections for Dual-loop Control

XMP-series controllers can be configured for dual-loop control. In dual-loop control, the *velocity information* for the PID derivative term (Kd) is derived from a rotary *encoder on the motor* shaft, and the *position information* for the PID proportional and integral terms is derived from an *encoder on the load* itself.

The axis that will be used for the rotary encoder is configurable through software and can be any axis that is not controlling a motor. For example, if axis 0 is configured for velocity feedback and axis 1 is configured for positional feedback, your system would be connected as shown in the next figure.

Figure 3-31. Dual-loop encoder connections with differential encoders.



MPI Support for Steppers

Support for stepper motor control is included within the MPI (Motion Programming Interface).

Transceivers

All transceiver and user I/O are part of the “Motor” object.

Each hardware axis has three (3) transceivers dedicated to it, labeled XCVR**A**, XCVR**B**, and XCVR**C**. The XCVR**A**, XCVR**B**, and XCVR**C** transceivers support:

Table 3-2. All three transceivers support input/output, normal/inverted.

Transceiver	*Input or Output?*	*Normal or Inverted?*
XCVR**A**	Yes	Yes
XCVR**B**	Yes	Yes
XCVR**C**	Yes	Yes

Additionally, the A and B transceivers support more features than the C transceivers do:

Table 3-3. A and B transceivers support more features.

Transceiver	*Step or Dir?*	*CW or CCW?*	*Quad A or Quad B?*	*Normal or Inverted?*	*Compare?*
XCVR**A**	Yes	Yes	Yes	Yes	No
XCVR**B**	Yes	Yes	Yes	Yes	No
XCVR**C**	No	No	No	Yes	Yes

Note that only sensible XCVR**A**/XCVR**B** configurations are permitted. For example, if XCVR**A** is configured for *Step*, then XCVR**B** should be configured for *Dir*. If XCVR**A** is configured for *Dir*, then XCVR**B** should be configured for *Step*. Refer to the next table.

Table 3-4. Transceiver configurations that are supported.

XCVRA	*XCVRB*	*XCVRC*
Step	Dir	
Dir	Step	
CW	CCW	
CCW	CW	
QuadA	QuadB	
QuadB	QuadA	

Step/Dir & CW/CCW Specifications

Table 3-5. Step/dir and CW/CCW specifications.

Pulse width range for Step, CW & CCW	100 nsec to 25.5 microsec
Maximum duty cycle must be less than	50% (or 25.5 microsec)
Minimum separation between Dir edge and rising Step edge (or between CW and CCW edges)	125 nanosec
Maximum step output rate	2.5 MHz

The maximum separation between the Dir edge and the rising Step edge depends upon the sample rate and the commanded motion. A rule of thumb is that the Dir edge occurs at the start of the DSP's trajectory calculator (when the command velocity is non-zero), and the first Step occurs when the command position increments the first whole count. The time separation can be estimated from the commanded acceleration (Accel):

$$Time = \sqrt{\frac{2}{Accel}}$$

For example, if acceleration is 100,000 cts/sec^2, then the separation between the Dir edge and the first Step edge is

$$Time = \sqrt{\frac{2}{100,000}}$$ which computes to Time = 4.5 milliseconds

The minimum separation between the Dir edge and the first Step edge is 125 nanoseconds. Note that the XMP increments its counter on each *rising edge* of the Step or CW signal.

Figure 3-32. Step/dir (and CW/CCW) and motor motion .

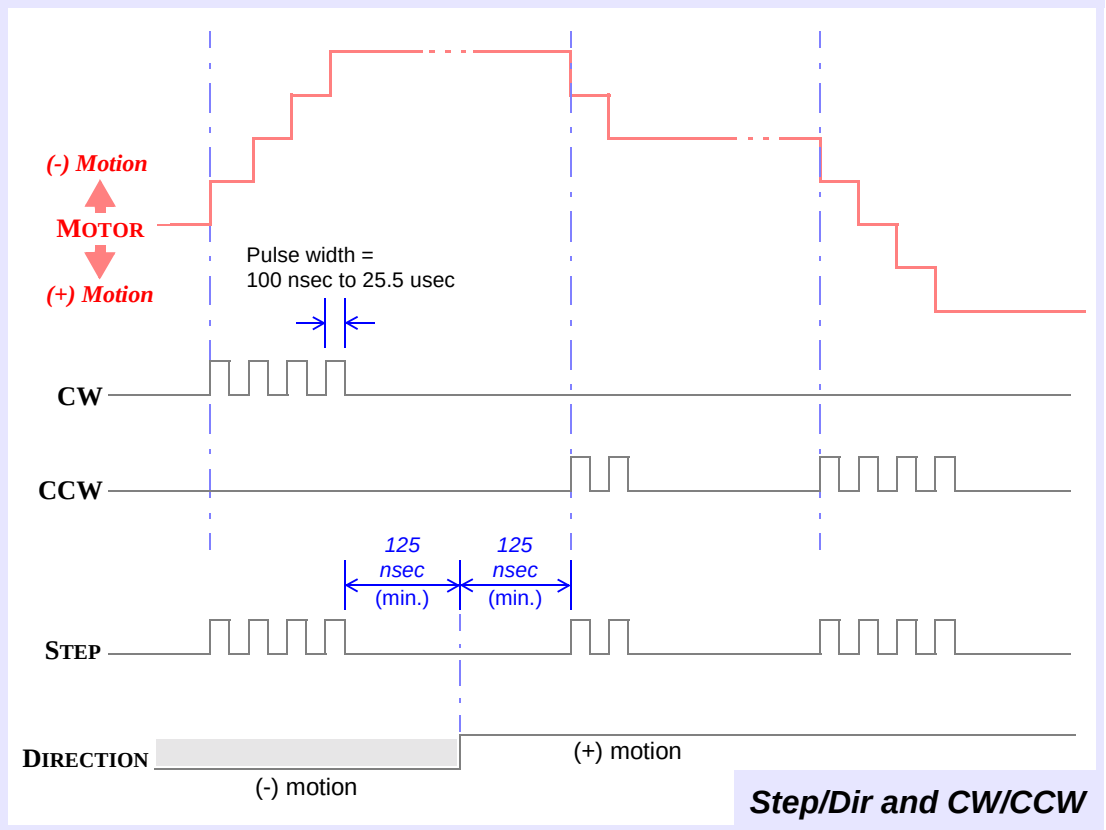
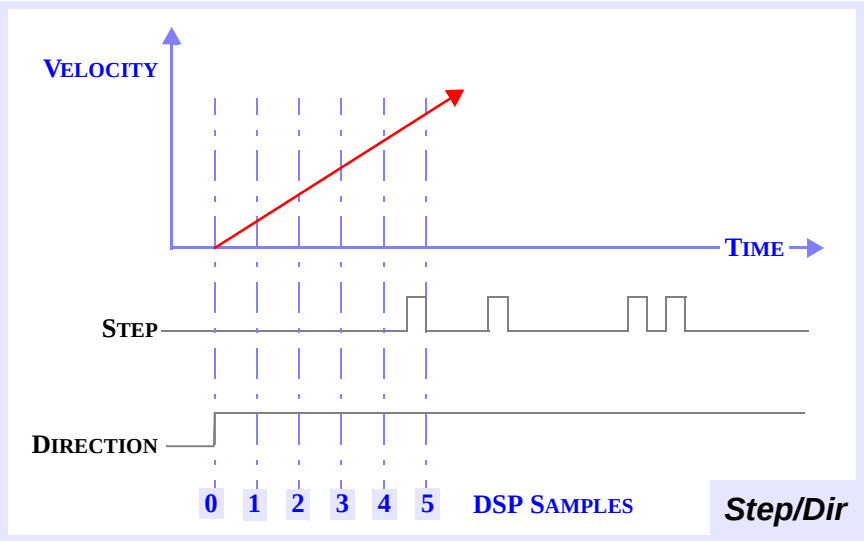


Figure 3-33. Step/dir and velocity.



Stepper Loopback

When the *Loopback* feature is enabled, the Step/Dir (or CW/CCW) logic is routed back into the encoder inputs. Note that the DSP doesn't use the feedback for control. There are two to three samples of latency between when the DSP's command position is updated and when the actual position (loopback) is updated. Also, loopback is not affected by “inverted” configurations.

The position error limit is still valid for loopback operations. If Loopback is not enabled, you can connect the encoder inputs to actual encoders.

HELPFUL TIP: Stepper loopback is very useful for motor simulation. When real servo motors are not available, the controller’s stepper motor and loopback configurations make it possible to develop software.

Closed Loop Steppers

Currently, there is no explicit support for closed loop stepper configurations. But, it is possible to correct the final position based on the actual position via application software, since the encoder inputs are valid with Step/Dir and CW/CCW configurations.

Stepper Configuration using *Motion Console*

To configure a Stepper using *Motion Console*, you must set the following parameters:

Table 3-6. Parameters used to configure a stepper in Motion Console.

Object	*MoCon Parameter*	*Setting*
Motor	Type	Stepper
	Stepper pulse width	2.55 (10^{-5})
	Stepper loopback	Yes
	Transceiver **A** config	Step or CW
	Transceiver **B** config	Dir or CCW
Filter	Algorithm	None

This is a blank page.

CHAPTER 4
XMP-ANALOG I/O

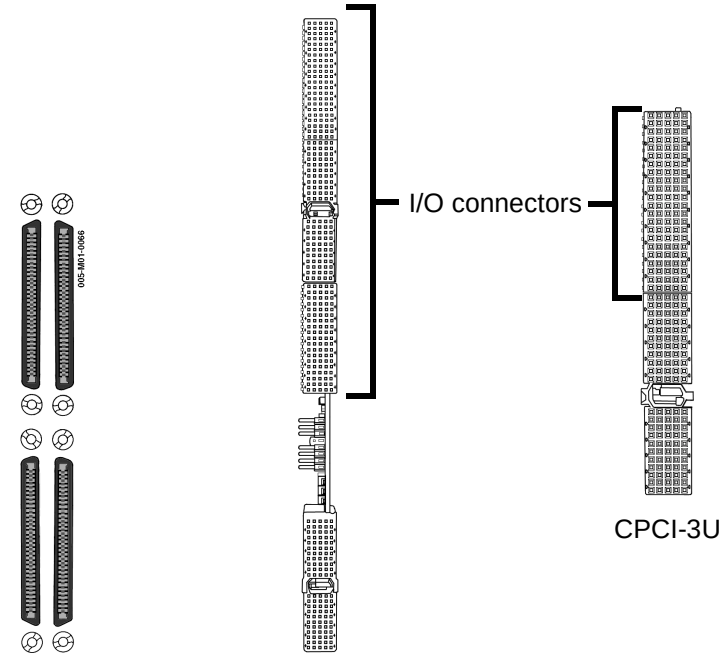
Introduction

This chapter describes analog I/O pin-outs for the XMP-PCI, XMP-CPCI-6U and XMP-CPCI-3U controllers. The first section of this chapter describes connector hardware. The second section of this chapter describes the actual I/O configuration, including pin-outs, organized by form factor.

SERCOS I/O is described in Chapter 5 of this manual.

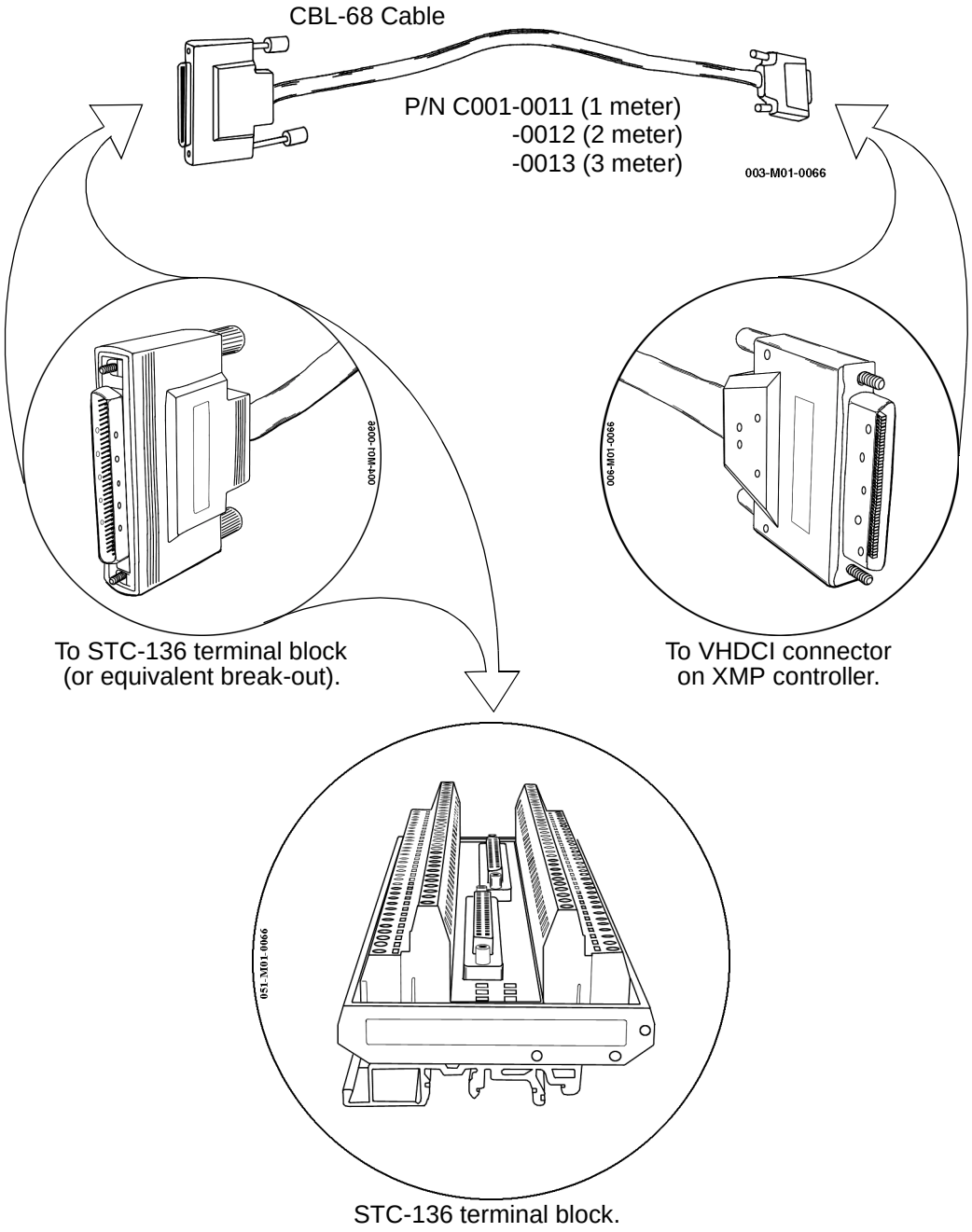
Identifying Analog I/O Hardware

Your analog XMP controller's I/O hardware type can be easily identified by examining the connection between the XMP controller and drive:



VHDCI Cabling and Terminal Block

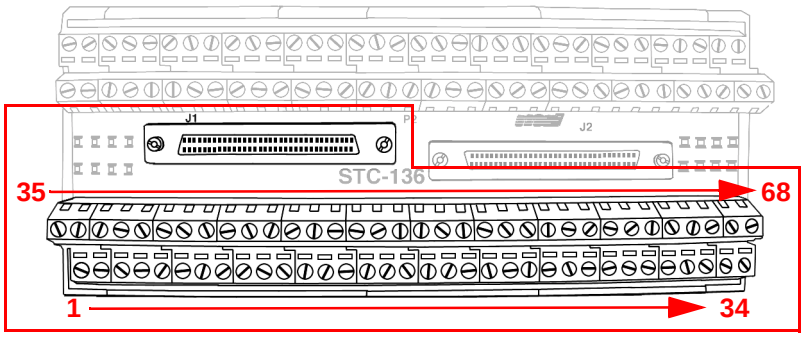
XMP-PCI and XMP-CPCI use the same 68-pin cable for all front panel (VHDCI) I/O connections.



Motion Engineering, Inc.

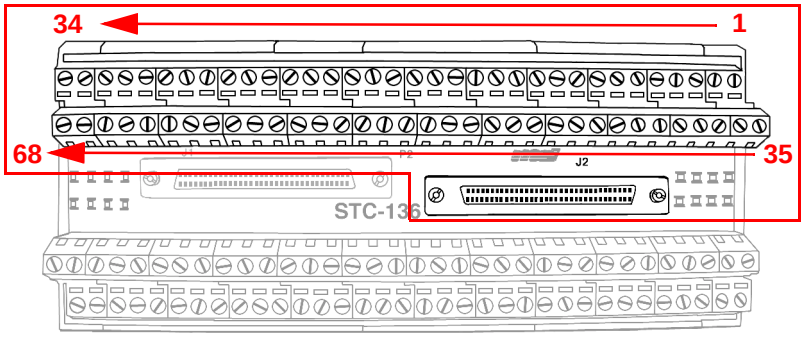
STC-136 Terminal Connector Blocks

All VHDCI connectors used with XMP controllers are 68-pin types. Corresponding to this, terminal blocks having 68 terminals per side are preferred. MEI recommends the STC-136, which can accommodate up to two cables. These use screw terminals for analog wiring.



J1 Terminals

J2 Terminals



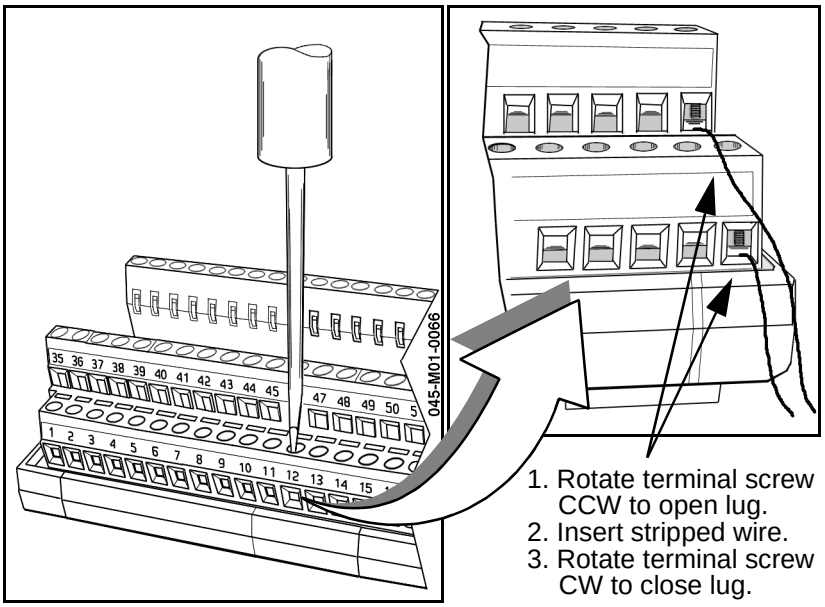
WARNING! Many "controller problems" can be traced to faulty wiring. As with all analog wiring, it is the user's responsibility to verify that wiring is correctly connected per specifications BEFORE applying power. Verify the following when wiring terminal blocks:

At the Terminal Block

- Verify all wiring is clearly and correctly labeled and color-coded for easy identification. Assembly personnel should have clear and concise wiring diagrams available when wiring.
- Connect wires to terminal blocks by referencing terminal numbers, *not* by a wire's placement relative to another wire.
- Strip wires appropriately: removing too much insulation will leave bare wires exposed and susceptible to short-circuit; removing too little insulation will prevent electrical contact inside the lug. Do not pinch or crimp wires when stripping—they will break!
- Do not insert wire above a terminal block's lug. When the lug is tightened, it will *not* close on the wire, failing to make contact. Verify that debris does not enter the lug with the bare wire; it may prevent complete closure of the lug.

• Verify that lugs are tight. Remove all loose wire strands.

• Discard all terminal blocks with damaged traces.

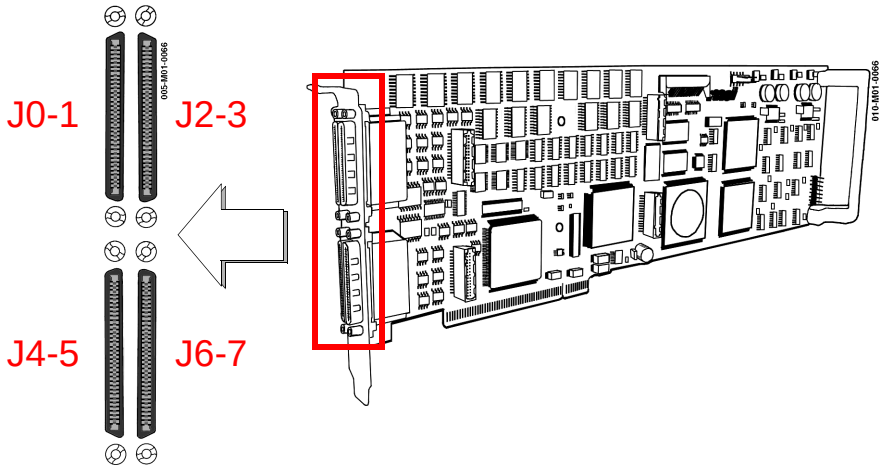


At the Connectors

• Verify that connector pins are straight BEFORE attempting to connect cables. Bent pins can short-circuit, or ruin a connector! If connectors resist, do not force them. Check for problems first.

• Observe electrostatic discharge (ESD) precautions while handling connectors and terminal blocks. High-level ESD can damage circuitry over cables.

XMP-PCI



All motion drive I/O for XMP-PCI controllers is performed via VHDCI connectors on the rear end panel of the controller. Each VHDCI connector is connected to a motion drive via cable. Pin-outs are described below:

Axis 0-1, VHDCI Connector (PCI)

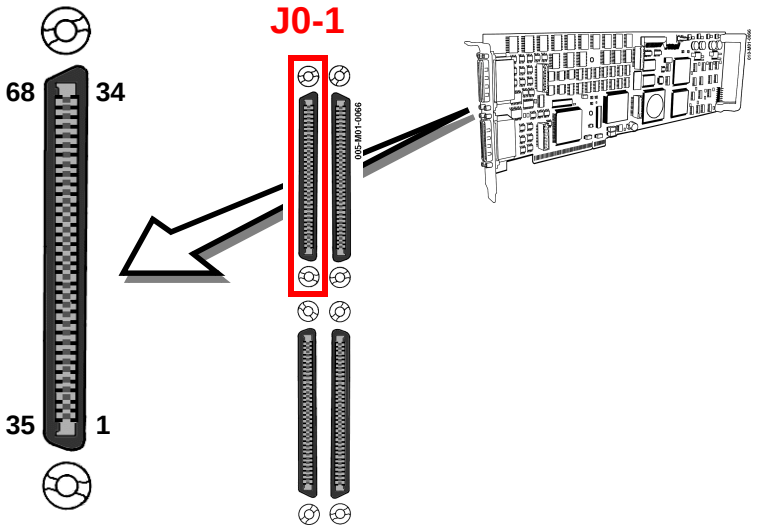


Table 4-1. Connector for axes 0 and 1 (PCI main board).

Pin	*Signal*	*Signal*	*Pin*
1	Analog_IN_0+	Analog_IN_0-	35
2	Analog_IN_1+	Analog_IN_1-	36
3	Gnd	AGnd	37
4	Enc0_A+	Enc0_A-	38
5	Enc0_B+	Enc0_B-	39
6	Enc0_I+	Enc0_I-	40
7	Home0_IN	5V_OUT	41
8	Pos_Lim0_IN	Gnd	42
9	Neg_Lim0_IN	HomeLim0_Rtn	43
10	Cmd_Dac_OUT_0+	Cmd_Dac_OUT_0-	44
11	Aux_Dac_OUT_0+	Aux_Dac_OUT_0-	45
12	Amp_Flt0_IN	Amp_Flt0_Rtn	46
13	Amp_En0_Collector	Amp_En0_Emitter	47
14	UserIO_A0	UserIO_A0_Rtn	48
15	Xcvr0A+	Xcvr0A-	49
16	Xcvr0B+	Xcvr0B-	50
17	Xcvr0C+	Xcvr0C-	51
18	Enc1_A+	Enc1_A-	52
19	Enc1_B+	Enc1_B-	53
20	Enc1_I+	Enc1_I-	54

Table 4-1. Connector for axes 0 and 1 (PCI main board).

Pin	*Signal*	*Signal*	*Pin*
21	Home1_IN	5V_OUT	55
22	Pos_Lim1_IN	Gnd	56
23	Neg_Lim1_IN	HomeLim1_Rtn	57
24	Cmd_Dac_OUT_1+	Cmd_Dac_OUT_1-	58
25	Aux_Dac_OUT_1+	Aux_Dac_OUT_1-	59
26	Amp_Flt1_IN	Amp_Flt1_Rtn	60
27	Amp_En1_Collector	Amp_En1_Emitter	61
28	Gnd	Gnd	62
29	Xcvr1A+	Xcvr1A-	63
30	Xcvr1B+	Xcvr1B-	64
31	Xcvr1C+	Xcvr1C-	65
32	UserIO_A1	UserIO_A1_Rtn	66
33	RESET_IN	UserIO_A2	67
34	ESTOP_IN	UserIO_A2_Rtn	68

Axis 2-3 VHDCI Connector (PCI)

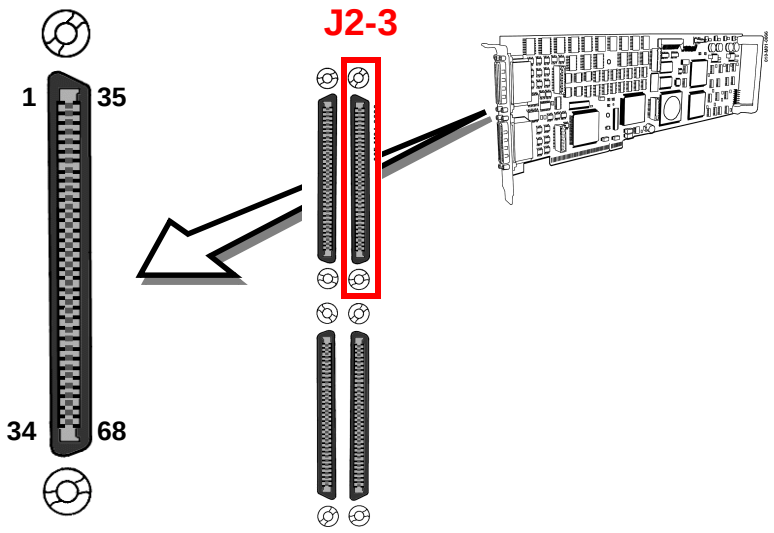


Table 4-2. Connector for axes 2 and 3 (PCI main board).

Pin	*Signal*	*Signal*	*Pin*
1	EncA_A+	EncA_A-	35
2	EncA_B+	EncA_B-	36
3	EncA_I+	EncA_I-	37
4	Enc2_A+	Enc2_A-	38
5	Enc2_B+	Enc2_B-	39
6	Enc2_I+	Enc2_I-	40
7	Home2_IN	5V_OUT	41
8	Pos_Lim2_IN	Gnd	42
9	Neg_Lim2_IN	HomeLim2_Rtn	43
10	Cmd_Dac_OUT_2+	Cmd_Dac_OUT_2-	44
11	Aux_Dac_OUT_2+	Aux_Dac_OUT_2-	45
12	Amp_Flt2_IN	Amp_Flt2_Rtn	46
13	Amp_En2_Collector	Amp_En2_Emitter	47
14	UserIO_A3	UserIO_A3_Rtn	48
15	Xcvr2A+	Xcvr2A-	49
16	Xcvr2B+	Xcvr2B-	50
17	Xcvr2C+	Xcvr2C-	51
18	Enc3_A+	Enc3_A-	52
19	Enc3_B+	Enc3_B-	53
20	Enc3_I+	Enc3_I-	54
21	Home3_IN	5V_OUT	55
22	Pos_Lim3_IN	Gnd	56

Table 4-2. Connector for axes 2 and 3 (PCI main board).

Pin	***Signal***	***Signal***	***Pin***
23	Neg_Lim3_IN	HomeLim3_Rtn	57
24	Cmd_Dac_OUT_3+	Cmd_Dac_OUT_3-	58
25	Aux_Dac_OUT_3+	Aux_Dac_OUT_3-	59
26	Amp_Flt3_IN	Amp_Flt3_Rtn	60
27	Amp_En3_Collector	Amp_En3_Emitter	61
28	Gnd	Gnd	62
29	Xcvr3A+	Xcvr3A-	63
30	Xcvr3B+	Xcvr3B-	64
31	Xcvr3C+	Xcvr3C-	65
32	Gnd	AGnd	66
33	Analog_IN_2+	Analog_IN_2-	67
34	Analog_IN_3+	Analog_IN_3-	68

Axis 4-5 VHDCI Connector (PCI)

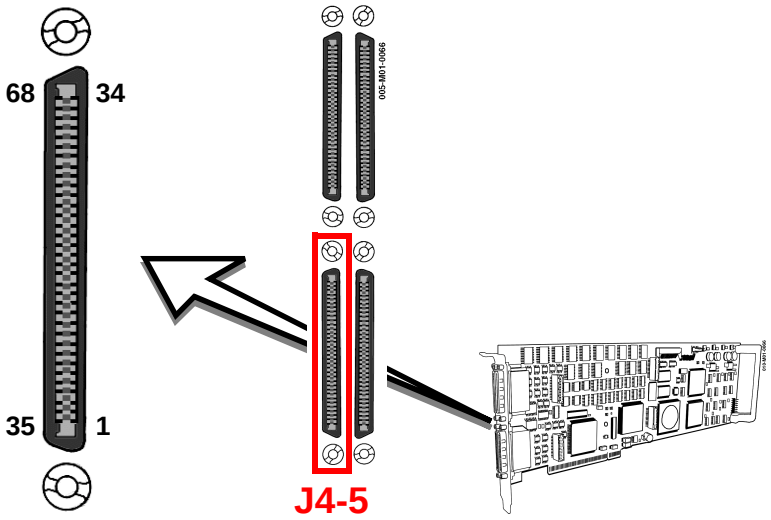


Table 4-3. Connector for axes 4 and 5 (PCI main board).

Pin	*Signal*	*Signal*	*Pin*
1	UserIO_B0	UserIO_B0_Rtn	35
2	UserIO_B1	UserIO_B1_Rtn	36
3	Gnd	Gnd	37
4	Enc4_A+	Enc4_A-	38
5	Enc4_B+	Enc4_B-	39
6	Enc4_I+	Enc4_I-	40
7	Home4_IN	5V_OUT	41
8	Pos_Lim4_IN	Gnd	42
9	Neg_Lim4_IN	HomeLim4_Rtn	43
10	Cmd_Dac_OUT_4+	Cmd_Dac_OUT_4-	44
11	Aux_Dac_OUT_4+	Aux_Dac_OUT_4-	45
12	Amp_Flt4_IN	Amp_Flt4_Rtn	46
13	Amp_En4_Collector	Amp_En4_Emitter	47
14	UserIO_B2	UserIO_B2_Rtn	48
15	Xcvr4A+	Xcvr4A-	49
16	Xcvr4B+	Xcvr4B-	50
17	Xcvr4C+	Xcvr4C-	51
18	Enc5_A+	Enc5_A-	52
19	Enc5_B+	Enc5_B-	53
20	Enc5_I+	Enc5_I-	54
21	Home5_IN	5V_OUT	55
22	Pos_Lim5_IN	Gnd	56

Table 4-3. Connector for axes 4 and 5 (PCI main board).

Pin	***Signal***	***Signal***	***Pin***
23	Neg_Lim5_IN	HomeLim5_Rtn	57
24	Cmd_Dac_OUT_5+	Cmd_Dac_OUT_5-	58
25	Aux_Dac_OUT_5+	Aux_Dac_OUT_5-	59
26	Amp_Flt5_IN	Amp_Flt5_Rtn	60
27	Amp_En5_Collector	Amp_En5_Emitter	61
28	Gnd	Gnd	62
29	Xcvr5A+	Xcvr5A-	63
30	Xcvr5B+	Xcvr5B-	64
31	Xcvr5C+	Xcvr5C-	65
32	Gnd	AGnd	66
33	Analog_IN_4+	Analog_IN_4-	67
34	Analog_IN_5+	Analog_IN_5-	68

Axis 6-7 VHCDI Connector (PCI)

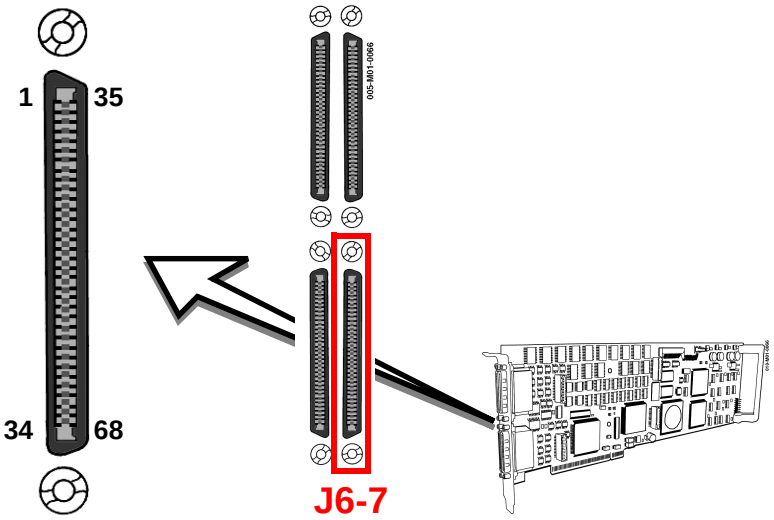


Table 4-4. Connector for axes 6 and 7 (PCI main board).

Pin	*Signal*	*Signal*	*Pin*
1	Analog_IN_6+	Analog_IN_6-	35
2	Analog_IN_7+	Analog_IN_7-	36
3	Gnd	AGnd	37
4	Enc6_A+	Enc6_A-	38
5	Enc6_B+	Enc6_B-	39
6	Enc6_I+	Enc6_I-	40
7	Home6_IN	5V_OUT	41
8	Pos_Lim6_IN	Gnd	42
9	Neg_Lim6_IN	HomeLim6_Rtn	43
10	Cmd_Dac_OUT_6+	Cmd_Dac_OUT_6-	44
11	Aux_Dac_OUT_6+	Aux_Dac_OUT_6-	45
12	Amp_Flt6_IN	Amp_Flt6_Rtn	46
13	Amp_En6_Collector	Amp_En6_Emitter	47
14	UserIO_B3	UserIO_B3_Rtn	48
15	Xcvr6A+	Xcvr6A-	49
16	Xcvr6B+	Xcvr6B-	50
17	Xcvr6C+	Xcvr6C-	51
18	Enc7_A+	Enc7_A-	52
19	Enc7_B+	Enc7_B-	53
20	Enc7_I+	Enc7_I-	54
21	Home7_IN	5V_OUT	55
22	Pos_Lim7_IN	Gnd	56

Table 4-4. Connector for axes 6 and 7 (PCI main board).

Pin	*Signal*	*Signal*	*Pin*
23	Neg_Lim7_IN	HomeLim7_Rtn	57
24	Cmd_Dac_OUT_7+	Cmd_Dac_OUT_7-	58
25	Aux_Dac_OUT_7+	Aux_Dac_OUT_7-	59
26	Amp_Flt7_IN	Amp_Flt7_Rtn	60
27	Amp_En7_Collector	Amp_En7_Emitter	61
28	Gnd	Gnd	62
29	Xcvr7A+	Xcvr7A-	63
30	Xcvr7B+	Xcvr7B-	64
31	Xcvr7C+	Xcvr7C-	65
32	EncB_A+	EncB_A-	66
33	EncB_B+	EncB_B-	67
34	EncB_I+	EncB_I-	68

Axis 8-9 VHDCI Connector (PCI)

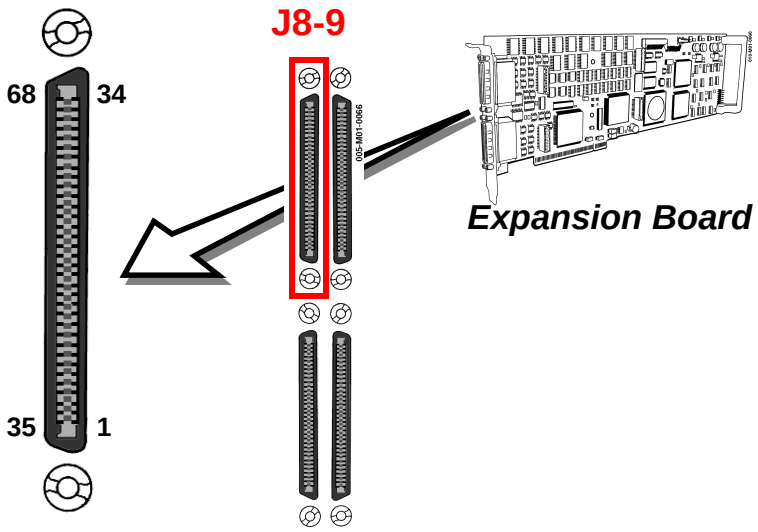


Table 4-5. Connector for axes 8 and 9 (PCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
1	Not Connected	Not Connected	35
2	Not Connected	Not Connected	36
3	Gnd	AGnd	37
4	Enc8_A+	Enc8_A-	38
5	Enc8_B+	Enc8_B-	39
6	Enc8_I+	Enc8_I-	40
7	Home8_IN	5V_OUT	41
8	Pos_Lim8_IN	Gnd	42
9	Neg_Lim8_IN	HomeLim8_Rtn	43
10	Cmd_Dac_OUT_8+	Cmd_Dac_OUT_8-	44
11	Aux_Dac_OUT_8+	Aux_Dac_OUT_8-	45
12	Amp_Flt8_IN	Amp_Flt8_Rtn	46
13	Amp_En8_Collector	Amp_En8_Emitter	47
14	UserIO_C0	UserIO_C0_Rtn	48
15	Xcvr8A+	Xcvr8A-	49
16	Xcvr8B+	Xcvr8B-	50
17	Xcvr8C+	Xcvr8C-	51
18	Enc9_A+	Enc9_A-	52
19	Enc9_B+	Enc9_B-	53
20	Enc9_I+	Enc9_I-	54

Table 4-5. Connector for axes 8 and 9 (PCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
21	Home9_IN	5V_OUT	55
22	Pos_Lim9_IN	Gnd	56
23	Neg_Lim9_IN	HomeLim9_Rtn	57
24	Cmd_Dac_OUT_9+	Cmd_Dac_OUT_9-	58
25	Aux_Dac_OUT_9+	Aux_Dac_OUT_9-	59
26	Amp_Flt9_IN	Amp_Flt9_Rtn	60
27	Amp_En9_Collector	Amp_En9_Emitter	61
28	Gnd	Gnd	62
29	Xcvr9A+	Xcvr9A-	63
30	Xcvr9B+	Xcvr9B-	64
31	Xcvr9C+	Xcvr9C-	65
32	UserIO_C1	UserIO_C1_Rtn	66
33	Not Connected	UserIO_C2	67
34	Not Connected	UserIO_C2_Rtn	68

Axis 10-11 VHCDI Connector (PCI)

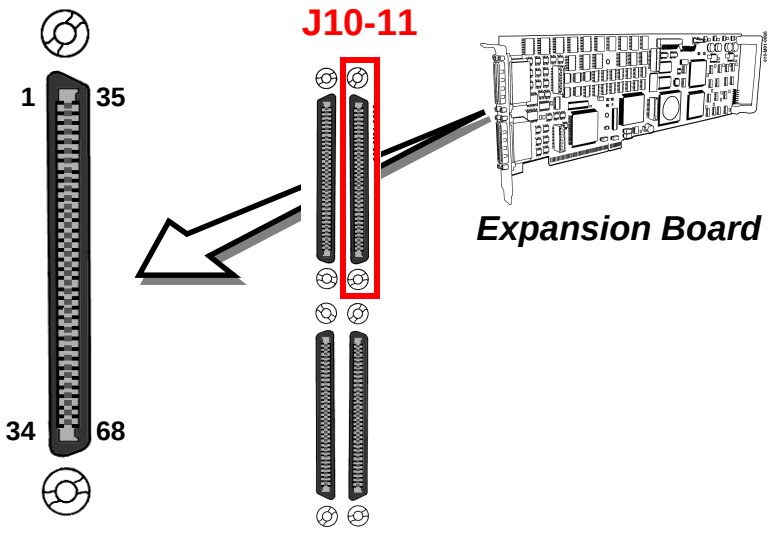


Table 4-6. Connector for axes 10 and 11 (PCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
1	EncC_A+	EncC_A-	35
2	EncC_B+	EncC_B-	36
3	EncC_I+	EncC_I-	37
4	Enc10_A+	Enc10_A-	38
5	Enc10_B+	Enc10_B-	39
6	Enc10_I+	Enc10_I-	40
7	Home10_IN	5V_OUT	41
8	Pos_Lim10_IN	Gnd	42
9	Neg_Lim10_IN	HomeLim10_Rtn	43
10	Cmd_Dac_OUT_10+	Cmd_Dac_OUT_10-	44
11	Aux_Dac_OUT_10+	Aux_Dac_OUT_10-	45
12	Amp_Flt10_IN	Amp_Flt10_Rtn	46
13	Amp_En10_Collector	Amp_En10_Emitter	47
14	UserIO_C3	UserIO_C3_Rtn	48
15	Xcvr10A+	Xcvr10A-	49
16	Xcvr10B+	Xcvr10B-	50
17	Xcvr10C+	Xcvr10C-	51
18	Enc11_A+	Enc11_A-	52
19	Enc11_B+	Enc11_B-	53
20	Enc11_I+	Enc11_I-	54
21	Home11_IN	5V_OUT	55
22	Pos_Lim11_IN	Gnd	56

Table 4-6. Connector for axes 10 and 11 (PCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
23	Neg_Lim11_IN	HomeLim11_Rtn	57
24	Cmd_Dac_OUT_11+	Cmd_Dac_OUT_11-	58
25	Aux_Dac_OUT_11+	Aux_Dac_OUT_11-	59
26	Amp_Flt11_IN	Amp_Flt11_Rtn	60
27	Amp_En11_Collector	Amp_En11_Emitter	61
28	Gnd	Gnd	62
29	Xcvr11A+	Xcvr11A-	63
30	Xcvr11B+	Xcvr11B-	64
31	Xcvr11C+	Xcvr11C-	65
32	Gnd	AGnd	66
33	Not Connected	Not Connected	67
34	Not Connected	Not Connected	68

Axis 12-13 VHDCI Connector (PCI)

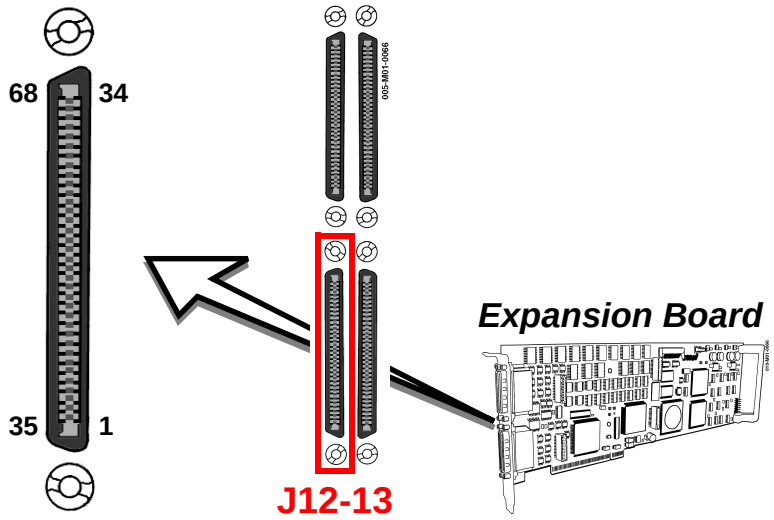


Table 4-7. Connector for axes 12 and 13 (PCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
1	UserIO_D0	UserIO_D0_Rtn	35
2	UserIO_D1	UserIO_D1_Rtn	36
3	Gnd	Gnd	37
4	Enc12_A+	Enc12_A-	38
5	Enc12_B+	Enc12_B-	39
6	Enc12_I+	Enc12_I-	40
7	Home12_IN	5V_OUT	41
8	Pos_Lim12_IN	Gnd	42
9	Neg_Lim12_IN	HomeLim12_Rtn	43
10	Cmd_Dac_OUT_12+	Cmd_Dac_OUT_12-	44
11	Aux_Dac_OUT_12+	Aux_Dac_OUT_12-	45
12	Amp_Flt12_IN	Amp_Flt12_Rtn	46
13	Amp_En12_Collector	Amp_En12_Emitter	47
14	UserIO_D2	UserIO_D2_Rtn	48
15	Xcvr12A+	Xcvr12A-	49
16	Xcvr12B+	Xcvr12B-	50
17	Xcvr12C+	Xcvr12C-	51
18	Enc13_A+	Enc13_A-	52
19	Enc13_B+	Enc13_B-	53
20	Enc13_I+	Enc13_I-	54
21	Home13_IN	5V_OUT	55
22	Pos_Lim13_IN	Gnd	56

Table 4-7. Connector for axes 12 and 13 (PCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
23	Neg_Lim13_IN	HomeLim13_Rtn	57
24	Cmd_Dac_OUT_13+	Cmd_Dac_OUT_13-	58
25	Aux_Dac_OUT_13+	Aux_Dac_OUT_13-	59
26	Amp_Flt13_IN	Amp_Flt13_Rtn	60
27	Amp_En13_Collector	Amp_En13_Emitter	61
28	Gnd	Gnd	62
29	Xcvr13A+	Xcvr13A-	63
30	Xcvr13B+	Xcvr13B-	64
31	Xcvr13C+	Xcvr13C-	65
32	Gnd	AGnd	66
33	Not Connected	Not Connected	67
34	Not Connected	Not Connected	68

Axis 14-15 VHDCI Connector (PCI)

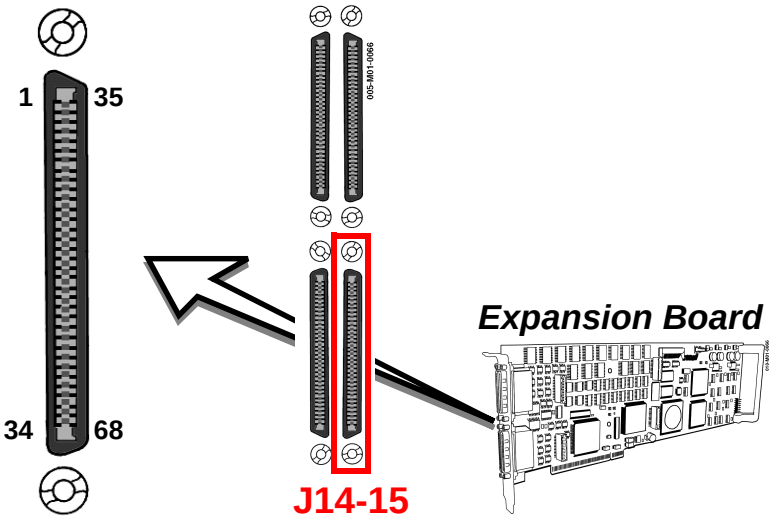


Table 4-8. Connector for axes 14 and 15 (PCI expansion board).

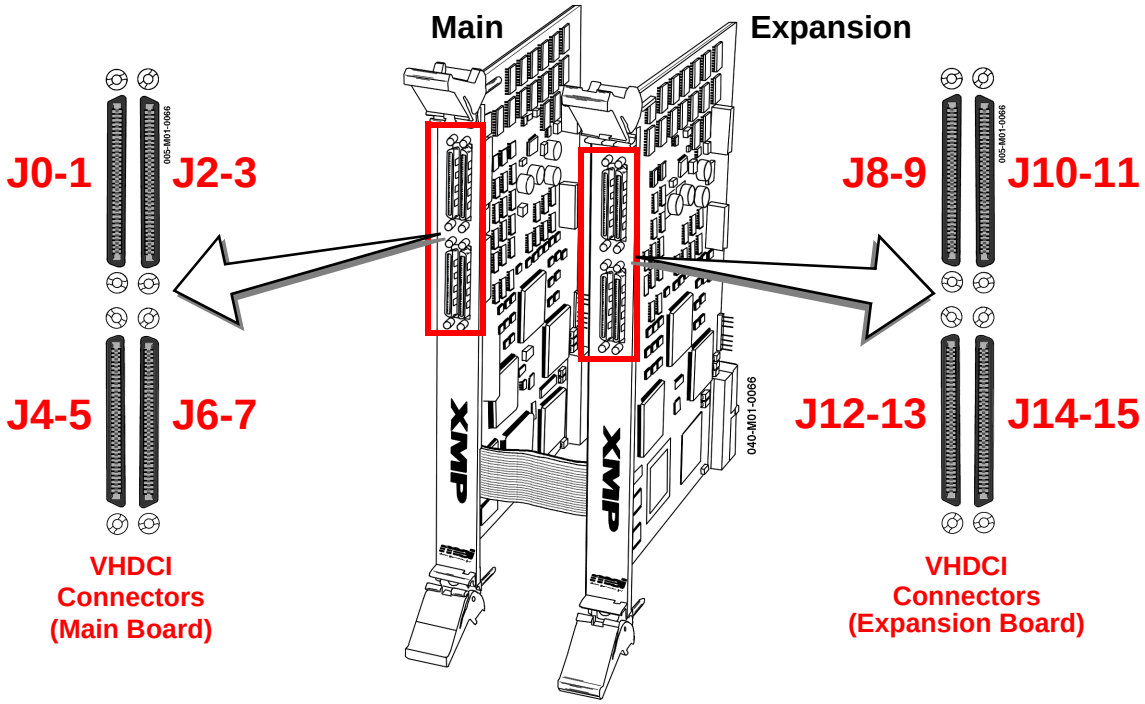
Pin	*Signal*	*Signal*	*Pin*
1	Not Connected	Not Connected	35
2	Not Connected	Not Connected	36
3	Gnd	AGnd	37
4	Enc14_A+	Enc14_A-	38
5	Enc14_B+	Enc14_B-	39
6	Enc14_I+	Enc14_I-	40
7	Home6_IN	5V_OUT	41
8	Pos_Lim14_IN	Gnd	42
9	Neg_Lim14_IN	HomeLim14_Rtn	43
10	Cmd_Dac_OUT_14+	Cmd_Dac_OUT_14-	44
11	Aux_Dac_OUT_14+	Aux_Dac_OUT_14-	45
12	Amp_Flt14_IN	Amp_Flt14_Rtn	46
13	Amp_En14_Collector	Amp_En14_Emitter	47
14	UserIO_D3	UserIO_D3_Rtn	48
15	Xcvr14A+	Xcvr14A-	49
16	Xcvr14B+	Xcvr14B-	50
17	Xcvr14C+	Xcvr14C-	51
18	Enc15_A+	Enc15_A-	52
19	Enc15_B+	Enc15_B-	53
20	Enc15_I+	Enc15_I-	54
21	Home15_IN	5V_OUT	55
22	Pos_Lim15_IN	Gnd	56

Table 4-8. Connector for axes 14 and 15 (PCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
23	Neg_Lim15_IN	HomeLim15_Rtn	57
24	Cmd_Dac_OUT_15+	Cmd_Dac_OUT_15-	58
25	Aux_Dac_OUT_15+	Aux_Dac_OUT_15-	59
26	Amp_Flt15_IN	Amp_Flt15_Rtn	60
27	Amp_En15_Collector	Amp_En15_Emitter	61
28	Gnd	Gnd	62
29	Xcvr15A+	Xcvr15A-	63
30	Xcvr15B+	Xcvr15B-	64
31	Xcvr15C+	Xcvr15C-	65
32	EncD_A+	EncD_A-	66
33	EncD_B+	EncD_B-	67
34	EncD_I+	EncD_I-	68

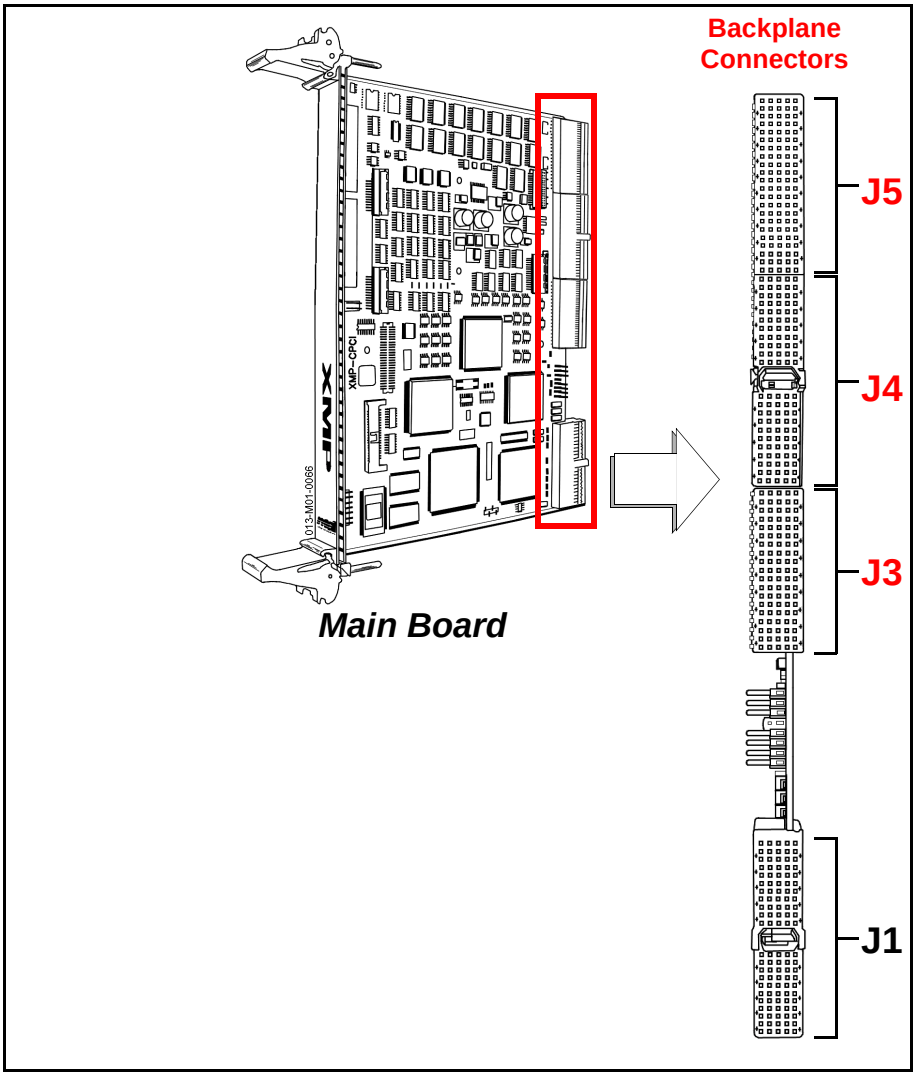
XMP-CPCI-6U

VHDCI Connectors, XMP-CPCI-6U



Many XMP-CPCI-6U boards (main and expansion) feature VHDCI connectors on their front panels. I/O for these connectors is identical to those for XMP-PCI controllers. For pin-outs, please refer to the XMP-PCI section above in this chapter.

CPCI Backplane Connectors, XMP-CPCI-6U



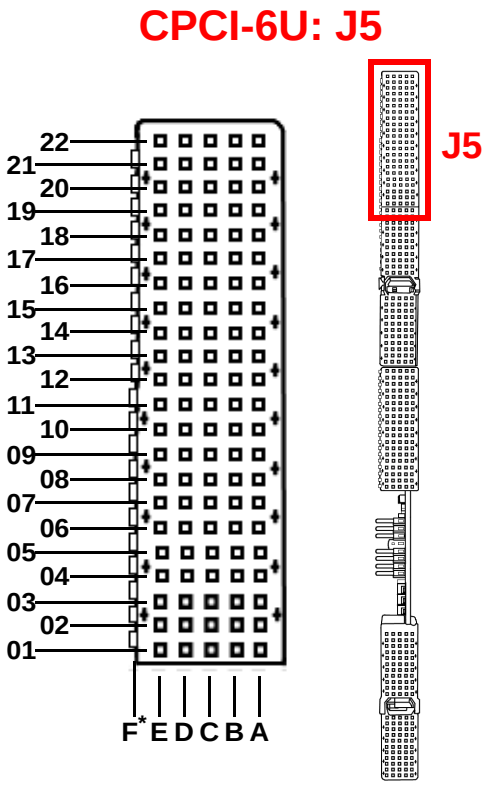
On CPCI-6U controllers using backplane I/O, drive signals are conveyed through connectors J3, J4 and J5.

For main boards, axes 0-3 are controlled through connector J5. Axes 4-7 are controlled through connector J4. User I/O is provided through connector J3.

On expansion boards, axes 8-11 are controlled through connector J5. Axes 12-15 are controlled through connector J4. User I/O is provided through connector J3.

On both main and expansion boards, connector J1 is reserved for computer data bussing. For more information, see Chapter 2.

CPCI-6U Main Board, CPCI Backplane Connector J5



* Column “F” is gold-plated grounding strip on outside edge of shroud.

Table 4-9. J5 backplane connector (CPCI main board).

Pin	***Signal***	***Signal***	***Pin***
A1	Analog_IN_0+	Analog_IN_3+	D1
A2	Analog_IN_1+	Analog_IN_2+	D2
A3	Enc0_A+	Xcvr3C+	D3
A4	Enc0_B+	Xcvr3B+	D4
A5	Enc0_I+	Xcvr3A+	D5
A6	Cmd_Dac_OUT_0+	Amp_En3_Collector	D6
A7	Aux_Dac_OUT_0+	Amp_Flt3_IN	D7
A8	Amp_Flt0_IN	Cmd_Dac_OUT_3+	D8
A9	Amp_En0_Collector	Aux_Dac_OUT_3+	D9
A10	Xcvr0A+	Enc3_I+	D10
A11	Xcvr0B+	Enc3_B+	D11
A12	Xcvr0C+	Enc3_A+	D12
A13	Enc1_A+	Xcvr2C+	D13
A14	Enc1_B+	Xcvr2B+	D14
A15	Enc1_I+	Xcvr2A+	D15
A16	Cmd_Dac_OUT_1+	Amp_En2_Collector	D16

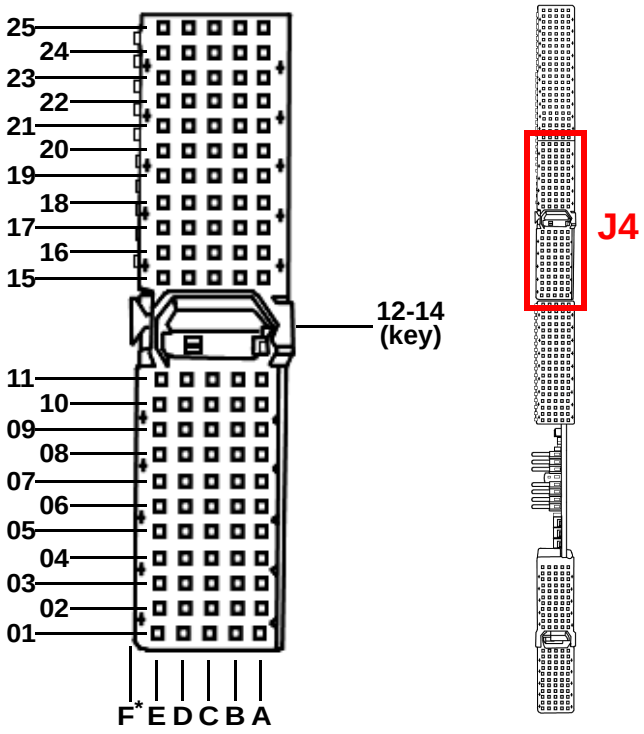
Table 4-9. J5 backplane connector (CPCI main board).

Pin	*Signal*	*Signal*	*Pin*
A17	Aux_Dac_OUT_1+	Amp_Flt2_IN	D17
A18	Amp_Flt1_IN	Cmd_Dac_OUT_2+	D18
A19	Amp_En1_Collector	Aux_Dac_OUT_2+	D19
A20	Xcvr1A+	Enc2_I+	D20
A21	Xcvr1B+	Enc2_B+	D21
A22	Xcvr1C+	Enc2_A+	D22
B1	Analog_IN_0-	Analog_IN_3-	E1
B2	Analog_IN_1-	Analog_IN_2-	E2
B3	Enc0_A-	Xcvr3C-	E3
B4	Enc0_B-	Xcvr3B-	E4
B5	Enc0_I-	Xcvr3A-	E5
B6	Cmd_Dac_OUT_0-	Amp_En3_Emitter	E6
B7	Aux_Dac_OUT_0-	Amp_Flt3_Rtn	E7
B8	Amp_Flt0_Rtn	Cmd_Dac_OUT_3-	E8
B9	Amp_En0_Emitter	Aux_Dac_OUT_3-	E9
B10	Xcvr0A-	Enc3_I-	E10
B11	Xcvr0B-	Enc3_B-	E11
B12	Xcvr0C-	Enc3_A-	E12
B13	Enc1_A-	Xcvr2C-	E13
B14	Enc1_B-	Xcvr2B-	E14
B15	Enc1_I-	Xcvr2A-	E15
B16	Cmd_Dac_OUT_1-	Amp_En2_Emitter	E16
B17	Aux_Dac_OUT_1-	Amp_Flt2_Rtn	E17
B18	Amp_Flt1_Rtn	Cmd_Dac_OUT_2-	E18
B19	Amp_En1_Emitter	Aux_Dac_OUT_2-	E19
B20	Xcvr1A-	Enc2_I-	E20
B21	Xcvr1B-	Enc2_B-	E21
B22	Xcvr1C-	Enc2_A-	E22
C1	AGnd	Gnd	F1
C2	Gnd	Gnd	F2
C3	Home0_IN	Gnd	F3
C4	Pos_Lim0_IN	Gnd	F4
C5	Neg_Lim0_IN	Gnd	F5
C6	HomeLim0_Rtn	Gnd	F6
C7	HomeLim3_Rtn	Gnd	F7
C8	Neg_Lim3_IN	Gnd	F8
C9	Pos_Lim3_IN	Gnd	F9
C10	Home3_IN	Gnd	F10
C11	Not Connected	Gnd	F11
C12	5V_OUT	Gnd	F12
C13	Home1_IN	Gnd	F13
C14	Pos_Lim1_IN	Gnd	F14
C15	Neg_Lim1_IN	Gnd	F15

Table 4-9. J5 backplane connector (CPCI main board).

Pin	*Signal*	*Signal*	*Pin*
C16	HomeLim1_Rtn	Gnd	F16
C17	HomeLim2_Rtn	Gnd	F17
C18	Neg_Lim2_IN	Gnd	F18
C19	Pos_Lim2_IN	Gnd	F19
C20	Home2_IN	Gnd	F20
C21	Gnd	Gnd	F21
C22	5V_OUT	Gnd	F22

CPCI-6U: J4



* Column “F” is gold-plated grounding strip on outside edge of shroud.

Table 4-10. J4 backplane connector (CPCI main board).

Pin	*Signal*	*Signal*	*Pin*
A1	Enc4_A+	Xcvr7C+	D1
A2	Enc4_B+	Xcvr7B+	D2
A3	Enc4_I+	Xcvr7A+	D3
A4	Cmd_Dac_OUT_4+	Amp_En7_Collector	D4
A5	Aux_Dac_OUT_4+	Amp_Flt7_IN	D5
A6	Amp_Flt4_IN	Aux_Dac_OUT_7+	D6
A7	Amp_En4_Collector	Cmd_Dac_OUT_7+	D7
A8	Xcvr4A+	Enc7_I+	D8
A9	Xcvr4B+	Enc7_B+	D9
A10	Xcvr4C+	Enc7_A+	D10
A11	Enc5_A+	Xcvr6C+	D11
A12 A13 A14	Not Connected	Not Connected	D12 D13 D14
A15	Enc5_B+	Xcvr6B+	D15

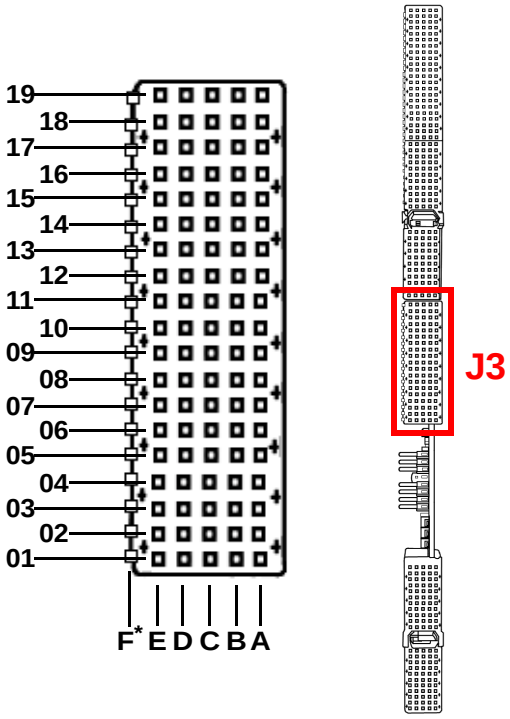
Table 4-10. J4 backplane connector (CPCI main board).

Pin	*Signal*	*Signal*	*Pin*
A16	Enc5_I+	Xcvr6A+	D16
A17	Cmd_Dac_OUT_5+	Amp_En6_Collector	D17
A18	Aux_Dac_OUT_5+	Amp_Flt6_IN	D18
A19	Amp_Flt5_IN	Aux_Dac_OUT_6+	D19
A20	Amp_En5_Collector	Cmd_Dac_OUT_6+	D20
A21	Xcvr5A+	Enc6_I+	D21
A22	Xcvr5B+	Enc6_B+	D22
A23	Xcvr5C+	Enc6_A+	D23
A24	Analog_IN_4+	Analog_IN_7+	D24
A25	Analog_IN_5+	Analog_IN_6+	D25
B1	Enc4_A-	Xcvr7C-	E1
B2	Enc4_B-	Xcvr7B-	E2
B3	Enc4_I-	Xcvr7A-	E3
B4	Cmd_Dac_OUT_4-	Amp_En7_Emitter	E4
B5	Aux_Dac_OUT_4-	Amp_Flt7_Rtn	E5
B6	Amp_Flt4_Rtn	Aux_Dac_OUT_7-	E6
B7	Amp_En4_Emitter	Cmd_Dac_OUT_7-	E7
B8	Xcvr4A-	Enc7_I-	E8
B9	Xcvr4B-	Enc7_B-	E9
B10	Xcvr4C-	Enc7_A-	E10
B11	Enc5_A-	Xcvr6C-	E11
B12 B13 B14	Not Connected	Not Connected	E12 E13 E14
B15	Enc5_B-	Xcvr6B-	E15
B16	Enc5_I-	Xcvr6A-	E16
B17	Cmd_Dac_OUT_5-	Amp_En6_Emitter	E17
B18	Aux_Dac_OUT_5-	Amp_Flt6_Rtn	E18
B19	Amp_Flt5_Rtn	Aux_Dac_OUT_6-	E19
B20	Amp_En5_Emitter	Cmd_Dac_OUT_6-	E20
B21	Xcvr5A-	Enc6_I-	E21
B22	Xcvr5B-	Enc6_B-	E22
B23	Xcvr5C-	Enc6_A-	E23
B24	Analog_IN_4-	Analog_IN_7-	E24
B25	Analog_IN_5-	Analog_IN_6-	E25
C1	5V_OUT	Gnd	F1
C2	Gnd	Gnd	F2
C3	Home4_IN	Gnd	F3
C4	Pos_Lim4_IN	Gnd	F4
C5	Neg_Lim4_IN	Gnd	F5
C6	HomeLim4_Rtn	Gnd	F6
C7	HomeLim7_Rtn	Gnd	F7
C8	Neg_Lim7_IN	Gnd	F8

Table 4-10. J4 backplane connector (CPCI main board).

Pin	*Signal*	*Signal*	*Pin*
C9	Pos_Lim7_IN	Gnd	F9
C10	Home7_IN	Gnd	F10
C11	5V_OUT	Gnd	F11
C12	Not Connected	Gnd	F12
C13		Gnd	F13
C14		Gnd	F14
C15		Gnd	F15
C16	Home5_IN	Gnd	F16
C17	Pos_Lim5_IN	Gnd	F17
C18	Neg_Lim5_IN	Gnd	F18
C19	HomeLim5_Rtn	Gnd	F19
C20	HomeLim6_Rtn	Gnd	F20
C21	Neg_Lim6_IN	Gnd	F21
C22	Pos_Lim6_IN	Gnd	F22
C23	Home6_IN	Gnd	F23
C24	Gnd	Gnd	F24
C25	AGnd	Gnd	F25

CPCI-6U: J3



* Column “F” is gold-plated grounding strip on outside edge of shroud.

Table 4-11. J3 backplane connector (CPCI main board).

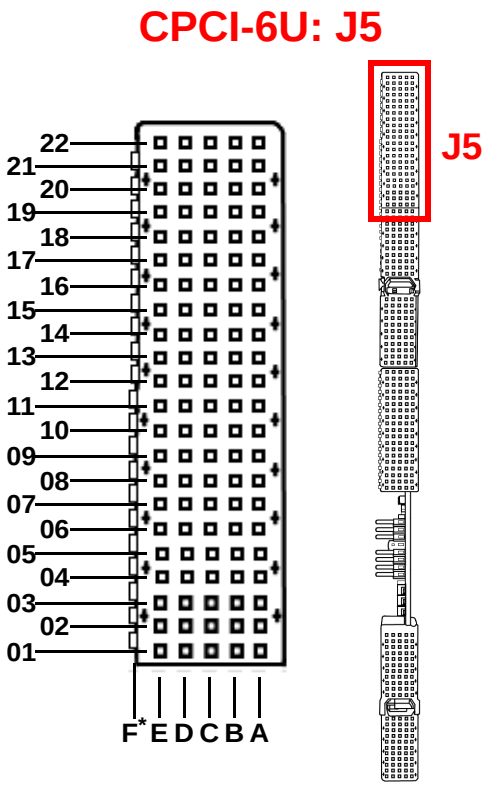
Pin	***Signal***	***Signal***	***Pin***
A1			D1
A2			D2
A3			D3
A4			D4
A5			D5
A6	Not Connected	Not Connected	D6
A7			D7
A8			D8
A9			D9
A10			D10
A11			D11
A12			D12
A13	UserIO_A1	UserIO_A2_Rtn	D13
A14	ESTOP_IN	UserIO_B1	D14
A15	UserIO_A3	UserIO_B3	D15

Table 4-11. J3 backplane connector (CPCI main board).

Pin	*Signal*	*Signal*	*Pin*
A16	UserIO_A0	UserIO_B2	D16
A17	EncA_I+	EncB_I+	D17
A18	EncA_B+	EncB_B+	D18
A19	EncA_A+	EncB_A+	D19
B1	Not Connected	Not Connected	E1
B2			E2
B3			E3
B4			E4
B5			E5
B6			E6
B7			E7
B8			E8
B9			E9
B10			E10
B11			E11
B12			E12
B13	UserIO_A1_Rtn	UserIO_A2	E13
B14	RESET_IN	UserIO_B1_Rtn	E14
B15	UserIO_A3_Rtn	UserIO_B3_Rtn	E15
B16	UserIO_A0_Rtn	UserIO_B2_Rtn	E16
B17	EncA_I-	EncB_I-	E17
B18	EncA_B-	EncB_B-	E18
B19	EncA_A-	EncB_A-	E19
C1	Gnd	Gnd	F1
C2	Not Connected	Gnd	F2
C3	Gnd	Gnd	F3
C4	Not Connected	Gnd	F4
C5	Gnd	Gnd	F5
C6	Not Connected	Gnd	F6
C7		Gnd	F7
C8		Gnd	F8
C9		Gnd	F9
C10		Gnd	F10
C11	UserIO_B0	Gnd	F11
C12	UserIO_B0_Rtn	Gnd	F12
C13	Gnd	Gnd	F13
C14	Gnd	Gnd	F14
C15	Gnd	Gnd	F15
C16	Gnd	Gnd	F16
C17	Gnd	Gnd	F17
C18	Gnd	Gnd	F18
C19	Gnd	Gnd	F19

XMP-CPCI-6U Expansion Board, Backplane Connectors

CPCI-6U Expansion Board, CPCI Backplane Connector J5: Axes 8 - 11



* Column “F” is gold-plated grounding strip on outside edge of shroud.

Table 4-12. J5 backplane connector (CPCI expansion board).

Pin	***Signal***	***Signal***	***Pin***
A1	Analog_IN_0+	Analog_IN_3+	D1
A2	Analog_IN_1+	Analog_IN_2+	D2
A3	Enc8_A+	Xcvr11C+	D3
A4	Enc8_B+	Xcvr11B+	D4
A5	Enc8_I+	Xcvr11A+	D5
A6	Cmd_Dac_OUT_8+	Amp_En11_Collector	D6
A7	Aux_Dac_OUT_8+	Amp_Flt11_IN	D7
A8	Amp_Flt8_IN	Cmd_Dac_OUT_11+	D8
A9	Amp_En8_Collector	Aux_Dac_OUT_11+	D9
A10	Xcvr8A+	Enc11_I+	D10
A11	Xcvr8B+	Enc11_B+	D11
A12	Xcvr8C+	Enc11_A+	D12
A13	Enc9_A+	Xcvr10C+	D13

Table 4-12. J5 backplane connector (CPCI expansion board).

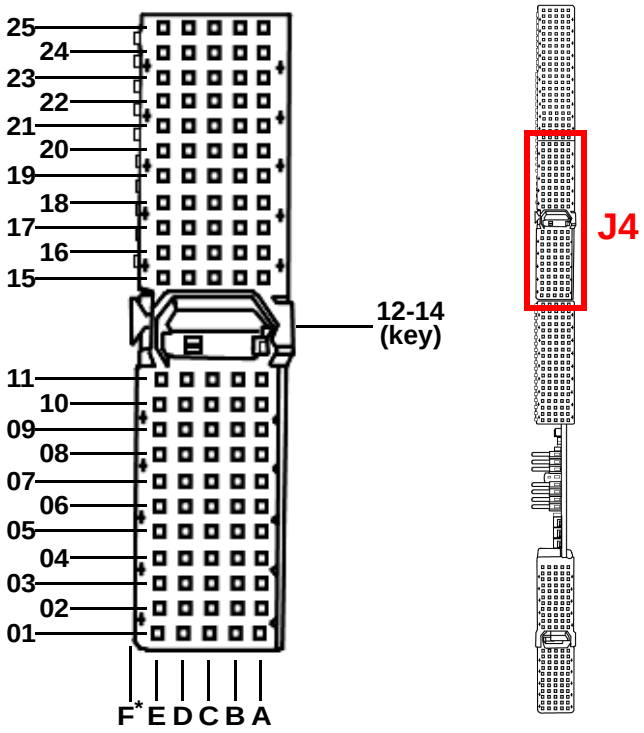
Pin	*Signal*	*Signal*	*Pin*
A14	Enc9_B+	Xcvr10B+	D14
A15	Enc9_I+	Xcvr10A+	D15
A16	Cmd_Dac_OUT_9+	Amp_En10_Collector	D16
A17	Aux_Dac_OUT_9+	Amp_Flt10_IN	D17
A18	Amp_Flt9_IN	Cmd_Dac_OUT_10+	D18
A19	Amp_En9_Collector	Aux_Dac_OUT_10+	D19
A20	Xcvr9A+	Enc10_I+	D20
A21	Xcvr9B+	Enc10_B+	D21
A22	Xcvr9C+	Enc10_A+	D22
B1	Analog_IN_0-	Analog_IN_3-	E1
B2	Analog_IN_1-	Analog_IN_2-	E2
B3	Enc8_A-	Xcvr11C-	E3
B4	Enc8_B-	Xcvr11B-	E4
B5	Enc8_I-	Xcvr11A-	E5
B6	Cmd_Dac_OUT_8-	Amp_En11_Emitter	E6
B7	Aux_Dac_OUT_8-	Amp_Flt11_Rtn	E7
B8	Amp_Flt8_Rtn	Cmd_Dac_OUT_11-	E8
B9	Amp_En8_Emitter	Aux_Dac_OUT_11-	E9
B10	Xcvr8A-	Enc11_I-	E10
B11	Xcvr8B-	Enc11_B-	E11
B12	Xcvr8C-	Enc11_A-	E12
B13	Enc9_A-	Xcvr10C-	E13
B14	Enc9_B-	Xcvr10B-	E14
B15	Enc9_I-	Xcvr10A-	E15
B16	Cmd_Dac_OUT_9-	Amp_En10_Emitter	E16
B17	Aux_Dac_OUT_9-	Amp_Flt10_Rtn	E17
B18	Amp_Flt9_Rtn	Cmd_Dac_OUT_10-	E18
B19	Amp_En9_Emitter	Aux_Dac_OUT_10-	E19
B20	Xcvr9A-	Enc10_I-	E20
B21	Xcvr9B-	Enc10_B-	E21
B22	Xcvr9C-	Enc10_A-	E22
C1	AGnd	Gnd	F1
C2	Gnd	Gnd	F2
C3	Home8_IN	Gnd	F3
C4	Pos_Lim8_IN	Gnd	F4
C5	Neg_Lim8_IN	Gnd	F5
C6	HomeLim8_Rtn	Gnd	F6
C7	HomeLim11_Rtn	Gnd	F7
C8	Neg_Lim11_IN	Gnd	F8
C9	Pos_Lim11_IN	Gnd	F9
C10	Home11_IN	Gnd	F10
C11	Not Connected	Gnd	F11
C12	5V_OUT	Gnd	F12

Table 4-12. J5 backplane connector (CPCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
C13	Home9_IN	Gnd	F13
C14	Pos_Lim9_IN	Gnd	F14
C15	Neg_Lim9_IN	Gnd	F15
C16	HomeLim9_Rtn	Gnd	F16
C17	HomeLim10_Rtn	Gnd	F17
C18	Neg_Lim10_IN	Gnd	F18
C19	Pos_Lim10_IN	Gnd	F19
C20	Home10_IN	Gnd	F20
C21	Gnd	Gnd	F21
C22	5V_OUT	Gnd	F22

CPCI-6U Expansion Board, Rear Panel Connector J4: Axes 12 - 15

CPCI-6U: J4



* Column “F” is gold-plated grounding strip on outside edge of shroud.

Table 4-13. J4 backplane connector (CPCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
A1	Enc12_A+	Xcvr15C+	D1
A2	Enc12_B+	Xcvr15B+	D2
A3	Enc12_I+	Xcvr15A+	D3
A4	Cmd_Dac_OUT_12+	Amp_En15_Collector	D4
A5	Aux_Dac_OUT_12+	Amp_Flt15_IN	D5
A6	Amp_Flt12_IN	Aux_Dac_OUT_15+	D6
A7	Amp_En12_Collector	Cmd_Dac_OUT_15+	D7
A8	Xcvr12A+	Enc15_I+	D8
A9	Xcvr12B+	Enc15_B+	D9
A10	Xcvr12C+	Enc15_A+	D10
A11	Enc13_A+	Xcvr14C+	D11
A12 A13 A14	Not Connected	Not Connected	D12 D13 D14

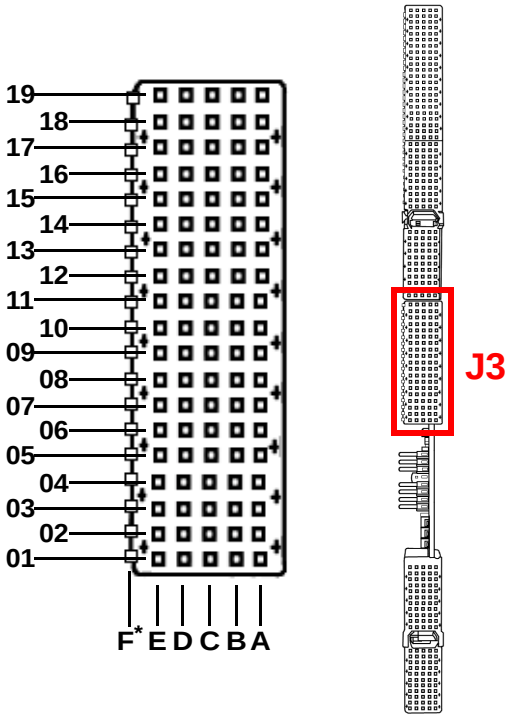
Table 4-13. J4 backplane connector (CPCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
A15	Enc13_B+	Xcvr14B+	D15
A16	Enc13_I+	Xcvr14A+	D16
A17	Cmd_Dac_OUT_13+	Amp_En14_Collector	D17
A18	Aux_Dac_OUT_13+	Amp_Flt14_IN	D18
A19	Amp_Flt13_IN	Aux_Dac_OUT_14+	D19
A20	Amp_En13_Collector	Cmd_Dac_OUT_14+	D20
A21	Xcvr13A+	Enc14_I+	D21
A22	Xcvr13B+	Enc14_B+	D22
A23	Xcvr13C+	Enc14_A+	D23
A24	Analog_IN_4+	Analog_IN_7+	D24
A25	Analog_IN_5+	Analog_IN_6+	D25
B1	Enc12_A-	Xcvr15C-	E1
B2	Enc12_B-	Xcvr15B-	E2
B3	Enc12_I-	Xcvr15A-	E3
B4	Cmd_Dac_OUT_12-	Amp_En15_Emitter	E4
B5	Aux_Dac_OUT_12-	Amp_Flt15_Rtn	E5
B6	Amp_Flt12_Rtn	Aux_Dac_OUT_15-	E6
B7	Amp_En12_Emitter	Cmd_Dac_OUT_15-	E7
B8	Xcvr12A-	Enc15_I-	E8
B9	Xcvr12B-	Enc15_B-	E9
B10	Xcvr12C-	Enc15_A-	E10
B11	Enc13_A-	Xcvr14C-	E11
B12 B13 B14	Not Connected	Not Connected	E12 E13 E14
B15	Enc13_B-	Xcvr14B-	E15
B16	Enc13_I-	Xcvr14A-	E16
B17	Cmd_Dac_OUT_5-	Amp_En14_Emitter	E17
B18	Aux_Dac_OUT_5-	Amp_Flt14_Rtn	E18
B19	Amp_Flt13_Rtn	Aux_Dac_OUT_14-	E19
B20	Amp_En13_Emitter	Cmd_Dac_OUT_14-	E20
B21	Xcvr13A-	Enc14_I-	E21
B22	Xcvr13B-	Enc14_B-	E22
B23	Xcvr13C-	Enc14_A-	E23
B24	Analog_IN_4-	Analog_IN_7-	E24
B25	Analog_IN_5-	Analog_IN_6-	E25
C1	5V_OUT	Gnd	F1
C2	Gnd	Gnd	F2
C3	Home12_IN	Gnd	F3
C4	Pos_Lim12_IN	Gnd	F4
C5	Neg_Lim12_IN	Gnd	F5
C6	HomeLim12_Rtn	Gnd	F6
C7	HomeLim15_Rtn	Gnd	F7

Table 4-13. J4 backplane connector (CPCI expansion board).

Pin	*Signal*	*Signal*	*Pin*
C8	Neg_Lim15_IN	Gnd	F8
C9	Pos_Lim15_IN	Gnd	F9
C10	Home15_IN	Gnd	F10
C11	5V_OUT	Gnd	F11
C12	Not Connected	Gnd	F12
C13		Gnd	F13
C14		Gnd	F14
C15		Gnd	F15
C16	Home13_IN	Gnd	F16
C17	Pos_Lim13_IN	Gnd	F17
C18	Neg_Lim13_IN	Gnd	F18
C19	HomeLim13_Rtn	Gnd	F19
C20	HomeLim14_Rtn	Gnd	F20
C21	Neg_Lim14_IN	Gnd	F21
C22	Pos_Lim14_IN	Gnd	F22
C23	Home14_IN	Gnd	F23
C24	Gnd	Gnd	F24
C25	AGnd	Gnd	F25

CPCI-6U: J3



* Column “F” is gold-plated grounding strip on outside edge of shroud.

Table 4-14. J3 backplane connector (CPCI expansion board).

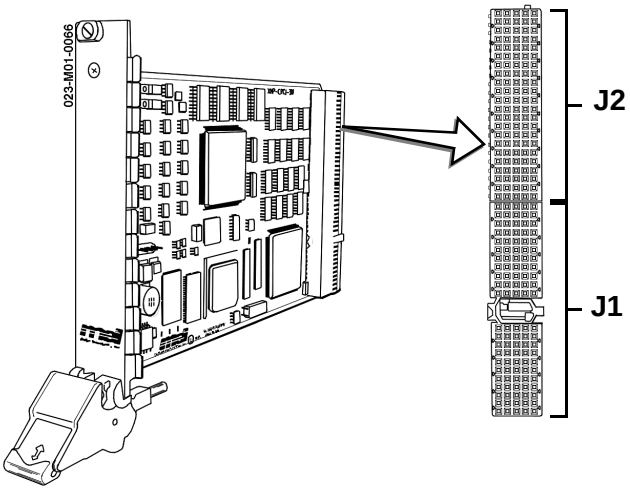
Pin	***Signal***	***Signal***	***Pin***
A1			D1
A2			D2
A3			D3
A4			D4
A5			D5
A6	Not Connected	Not Connected	D6
A7			D7
A8			D8
A9			D9
A10			D10
A11			D11
A12			D12
A13	UserIO_C1	UserIO_C2_Rtn	D13
A14	Not Used	UserIO_D2	D14
A15	UserIO_C3	UserIO_D4	D15

Table 4-14. J3 backplane connector (CPCI expansion board).

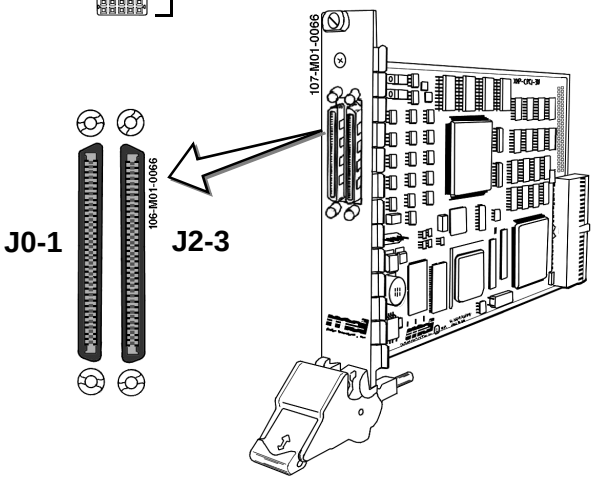
Pin	*Signal*	*Signal*	*Pin*
A16	UserIO_C0	UserIO_D3	D16
A17	EncC_I+	EncD_I+	D17
A18	EncC_B+	EncD_B+	D18
A19	EncC_A+	EncD_A+	D19
B1	Not Connected	Not Connected	E1
B2			E2
B3			E3
B4			E4
B5			E5
B6			E6
B7			E7
B8			E8
B9			E9
B10			E10
B11			E11
B12			E12
B13	UserIO_C1_Rtn	UserIO_C2	E13
B14	Not Used	UserIO_D2_Rtn	E14
B15	UserIO_C3_Rtn	UserIO_D4_Rtn	E15
B16	UserIO_A0_Rtn	UserIO_D3_Rtn	E16
B17	EncC_I-	EncD_I-	E17
B18	EncC_B-	EncD_B-	E18
B19	EncC_A-	EncD_A-	E19
C1	Gnd	Gnd	F1
C2	Not Connected	Gnd	F2
C3	Gnd	Gnd	F3
C4	Not Connected	Gnd	F4
C5	Gnd	Gnd	F5
C6	Not Connected	Gnd	F6
C7		Gnd	F7
C8		Gnd	F8
C9		Gnd	F9
C10		Gnd	F10
C11	UserIO_D14	Gnd	F11
C12	UserIO_D1_Rtn	Gnd	F12
C13	Gnd	Gnd	F13
C14	Gnd	Gnd	F14
C15	Gnd	Gnd	F15
C16	Gnd	Gnd	F16
C17	Gnd	Gnd	F17
C18	Gnd	Gnd	F18
C19	Gnd	Gnd	F19

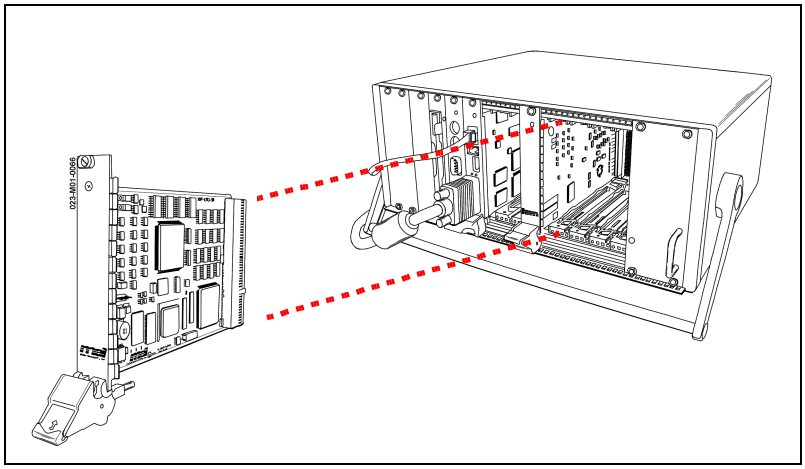
XMP-CPCI-3U

Backplane Connectors



VHDCI Connectors





The XMP-CPCI-3U boards have two hardware stuff options for bringing out Motor I/O: VHDCI connectors on the front panel or Backplane connectors on the rear panel. The I/O for these connectors is decribed below.

Axis 0-1, VHDCI Connector (CPCI)

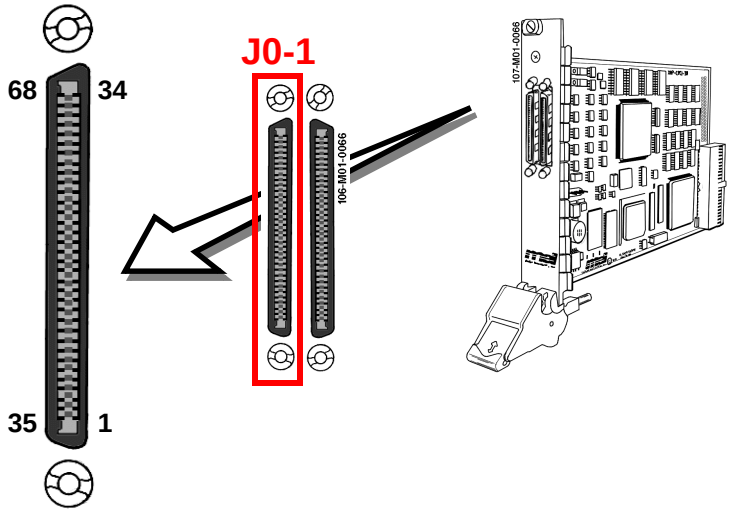


Table 4-15. Connector for axes 0 and 1.

Pin	*Signal*	*Signal*	*Pin*
1	No Connect	No Connect	35
2	No Connect	No Connect	36
3	Gnd	Gnd	37
4	Enc0_A+	Enc0_A-	38
5	Enc0_B+	Enc0_B-	39
6	Enc0_I+	Enc0_I-	40
7	Home0_IN	5V_OUT	41
8	Pos_Lim0_IN	Gnd	42
9	Neg_Lim0_IN	HomeLim0_Rtn	43
10	Cmd_Dac_OUT_0+	Cmd_Dac_OUT_0-	44
11	No Connect	Gnd	45
12	Amp_Flt0_IN	Amp_Flt0_Rtn	46
13	Amp_En0_Collector	Amp_En0_Emitter	47
14	UserIO_A0	UserIO_A0_Rtn	48
15	Xcvr0A+	Xcvr0A-	49
16	Xcvr0B+	Xcvr0B-	50
17	Xcvr0C+	Xcvr0C-	51
18	Enc1_A+	Enc1_A-	52
19	Enc1_B+	Enc1_B-	53
20	Enc1_I+	Enc1_I-	54

Table 4-15. Connector for axes 0 and 1.

Pin	*Signal*	*Signal*	*Pin*
21	Home1_IN	5V_OUT	55
22	Pos_Lim1_IN	Gnd	56
23	Neg_Lim1_IN	HomeLim1_Rtn	57
24	Cmd_Dac_OUT_1+	Cmd_Dac_OUT_1-	58
25	No Connect	Gnd	59
26	Amp_Flt1_IN	Amp_Flt1_Rtn	60
27	Amp_En1_Collector	Amp_En1_Emitter	61
28	Gnd	Gnd	62
29	Xcvr1A+	Xcvr1A-	63
30	Xcvr1B+	Xcvr1B-	64
31	Xcvr1C+	Xcvr1C-	65
32	UserIO_A1	UserIO_A1_Rtn	66
33	RESET_IN	UserIO_A2	67
34	ESTOP_IN	UserIO_A2_Rtn	68

Axis 2-3, VHDCI Connector (CPCI)

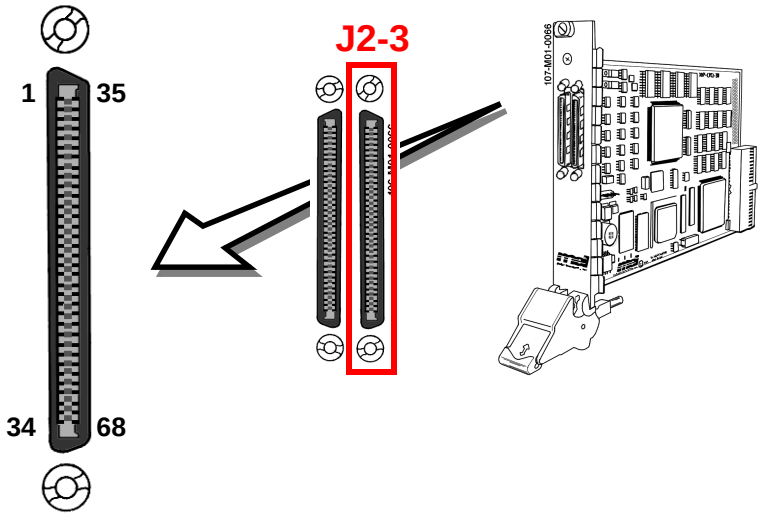


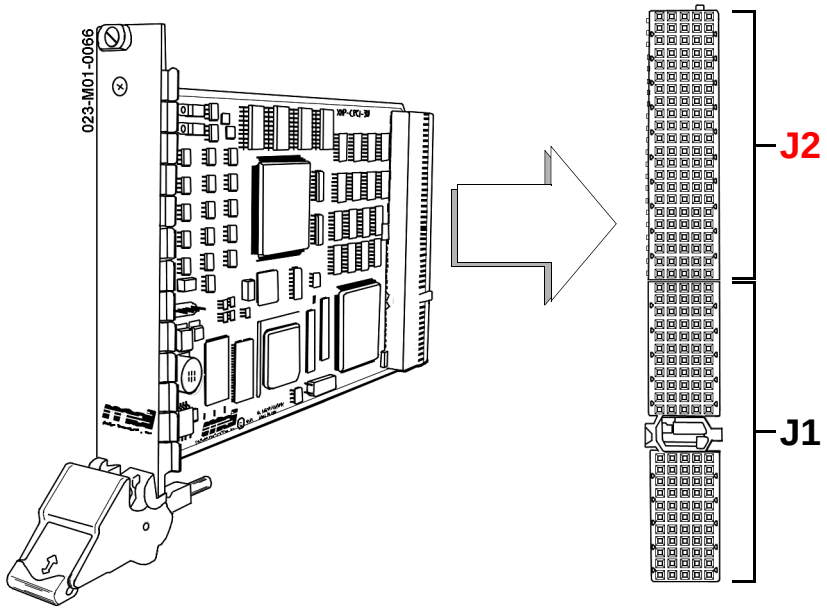
Table 4-16. Connector for axes 2 and 3.

Pin	*Signal*	*Signal*	*Pin*
1	EncA_A+	EncA_A-	35
2	EncA_B+	EncA_B-	36
3	EncA_I+	EncA_I-	37
4	Enc2_A+	Enc2_A-	38
5	Enc2_B+	Enc2_B-	39
6	Enc2_I+	Enc2_I-	40
7	Home2_IN	5V_OUT	41
8	Pos_Lim2_IN	Gnd	42
9	Neg_Lim2_IN	HomeLim2_Rtn	43
10	Cmd_Dac_OUT_2+	Cmd_Dac_OUT_2-	44
11	No Connect	Gnd	45
12	Amp_Flt2_IN	Amp_Flt2_Rtn	46
13	Amp_En2_Collector	Amp_En2_Emitter	47
14	UserIO_A3	UserIO_A3_Rtn	48
15	Xcvr2A+	Xcvr2A-	49
16	Xcvr2B+	Xcvr2B-	50
17	Xcvr2C+	Xcvr2C-	51
18	Enc3_A+	Enc3_A-	52
19	Enc3_B+	Enc3_B-	53
20	Enc3_I+	Enc3_I-	54
21	Home3_IN	5V_OUT	55
22	Pos_Lim3_IN	Gnd	56

Table 4-16. Connector for axes 2 and 3.

Pin	*Signal*	*Signal*	*Pin*
23	Neg_Lim3_IN	HomeLim3_Rtn	57
24	Cmd_Dac_OUT_3+	Cmd_Dac_OUT_3-	58
25	No Connect	Gnd	59
26	Amp_Flt3_IN	Amp_Flt3_Rtn	60
27	Amp_En3_Collector	Amp_En3_Emitter	61
28	Gnd	Gnd	62
29	Xcvr3A+	Xcvr3A-	63
30	Xcvr3B+	Xcvr3B-	64
31	Xcvr3C+	Xcvr3C-	65
32	Gnd	Gnd	66
33	No Connect	No Connect	67
34	No Connect	No Connect	68

CPCI-3U, Backplane Connectors, Rear Panel J2: Motor I/O

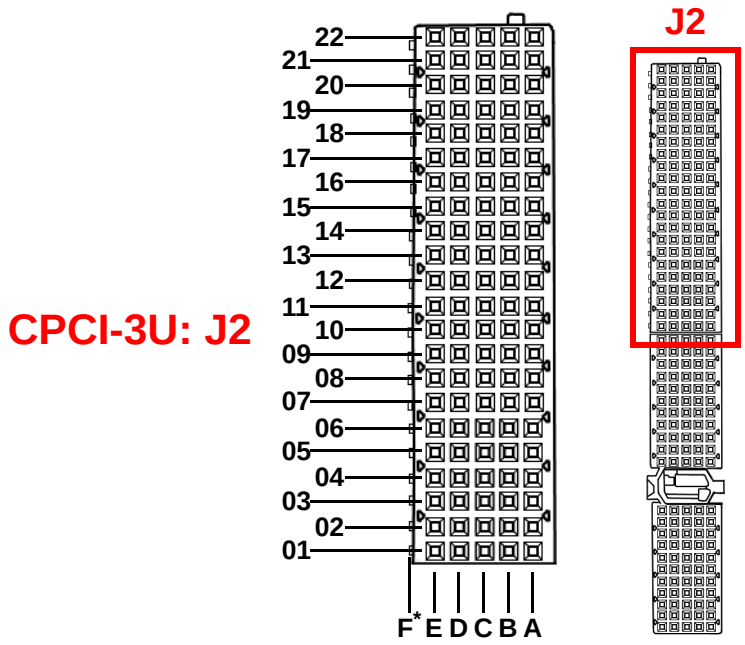


On CPCI-3U controllers using backplane I/O, Motor I/O is controlled through connector J2. Connector J1 is reserved for computer data bussing. For more information about computer data bussing, see Chapter 2.

CPCI-3U boards *do not* use expansion boards.

Backplane connector pins are referred to by numbered rows and lettered columns as shown below.

User I/O, Backplane Connector (CPCI) :



* Column “F” is gold-plated grounding strip on outside edge of shroud.

NOTE: the CPCI-XMP does not use the 64 bit extension.

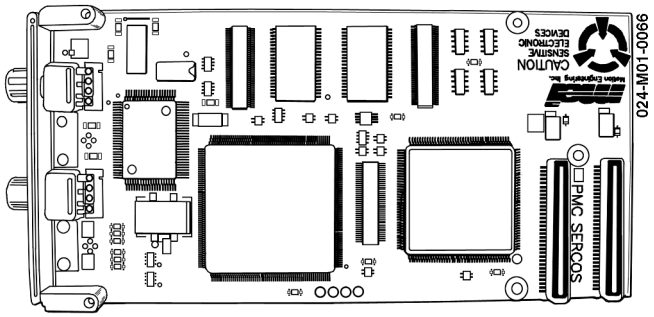
Table 4-17. CPCI backplane connector.

Pin	*A*	*B*	*C*	*D*	*E*	*F*
22	Reserved	Reserved	Reserved	Reserved	Reserved	GND
21	DAC_0+	DAC_1+	XCVR0A+	DAC_2+	DAC_3+	GND
20	DAC_0_RTN	DAC_1_RTN	XCVR0A-	DAC_2_RTN	DAC_3_RTN	GND
19	ENC0_A+	ENC0_A-	XCVR0B+	ENC2_A+	ENC2_A-	GND
18	ENC0_B+	ENC0_B-	XCVR0B-	ENC2_B+	ENC2_B-	GND
17	ENC0_I+	ENC0_I-	XCVR1A+	ENC2_I+	ENC2_I-	GND
16	ENC1_A+	ENC1_A-	XCVR1A-	ENC3_A+	ENC3_A-	GND
15	ENC1_B+	ENC1_B-	XCVR1B+	ENC3_B+	ENC3_B-	GND
14	ENC1_I+	ENC1_I-	XCVR1B-	ENC3_I+	ENC3_I-	GND
13	P2_A13	P2_B13	XCVR2A+	P2_D13	P2_E13	GND
12	P2_A12	P2_B12	XCVR2A-	P2_D12	P2_E12	GND
11	P2_A11	P2_B11	XCVR2B+	P2_D11	P2_E11	GND
10	P2_A10	P2_B10	XCVR2B-	DIR3+	DIR3-	GND
9	HOME_0_IN	HOMELIM_0_RTN	XCVR3A+	HOME_2_IN	HOMELIM_2_RTN	GND
8	HOME_1_IN	HOMELIM_1_RTN	XCVR3A-	HOME_3_IN	HOMELIM_3_RTN	GND
7	AMP_FLT_0_IN	AMP_FLT_0_RTN	XCVR3B+	AMP_FLT_2_IN	AMP_FLT_2_RTN	GND
6	P2_A6	P2_B6	XCVR3B-	P2_D6	P2_E6	GND
5	USERIO_A0	USERIO_A0_RTN	XCVR0C+	USERIO_A2	USERIO_A2_RTN	GND
4	USERIO_A1	USERIO_A1_RTN	XCVR0C-	USERIO_A3	USERIO_A3_RTN	GND

Table 4-17. CPCI backplane connector.

Pin	*A*	*B*	*C*	*D*	*E*	*F*
3	AMP_EN_0_COLL	AMP_EN_0_EMIT	P2_C3	AMP_EN_2_COLL	AMP_EN_2_EMIT	GND
2	P2_A2	P2_B2	P2_C2	P2_D2	P2_E2	GND
1	Reserved	Reserved	Reserved	Reserved	Reserved	GND

XMP-SERCOS-PMC



XMP-SERCOS-PMC controllers use SERCOS I/O. For more information regarding SERCOS, please see Chapter 5 of this manual.

Transceiver (XCVR) and User I/O Details

This section lists the signal names (at the connector pins) and the Motion Block FPGA functions associated with each.

XCVR Functions

The XCVR signals are used for many different functions. The XMP-PCI and XMP-CPCI each offer three XCVRs (A,B,C) per motor, while the XMP-CPCI-3U offers six XCVRs (A,B,C,D,E,F) per motor. The various XMP boards and XCVR combinations are listed in Table 4-18.

Table 4-18. XCVRs for various XMP products.

Product	***Boards***	***Motion Blocks***	***Motors***	***XCVRs per Motor***	***XCVRs per Block***	***XCVRs Total***
XMP-PCI	2	4	16	3 (A-C)	12	48
XMP-CPCI	2	4	16	3 (A-C)	12	48
XMP-CPCI-3U (non-standard)	1	1	4	6 (A-F)[a]	23	23
XMP-CPCI-3U (standard)	1	1	4	3 (A-C)	12	12

a. On non-standard XMP-CPCI-3U controller, Axis 2 XCVR D is not connected to rear I/O, giving a total of 23 XCVRs. Standard -3U controller has 12 XCVRs.

These multi-purpose XCVR signals may be configured for several different input or output functions. The available XCVR functions for one motor are detailed below; the XCVRs for all other motors are the same.

Note	An option to invert any output function is also available.

Table 4-19. XCVR A functions.

MOTOR 0 XCVR A Functions (connector signal pair XCVR0A+ and XCVR0A-)		
Function	***Signal Direction***	***Comments***
OUTPUT	OUTPUT	General purpose output
STEP	OUTPUT	STEP signal for MOTOR 0 stepper
DIR	OUTPUT	DIRECTION signal for MOTOR 0 stepper
CW	OUTPUT	CLOCKWISE signal for MOTOR 0 stepper
CCW	OUTPUT	COUNTER-CLOCKWISE signal for MOTOR 0 stepper
QUADA	OUTPUT	QUADRATURE A signal for MOTOR 0 stepper
QUADB	OUTPUT	QUADRATURE B signal for MOTOR 0 stepper
INPUT	INPUT	General purpose input -or- To CAPTURE EVENT 0 input lookup table

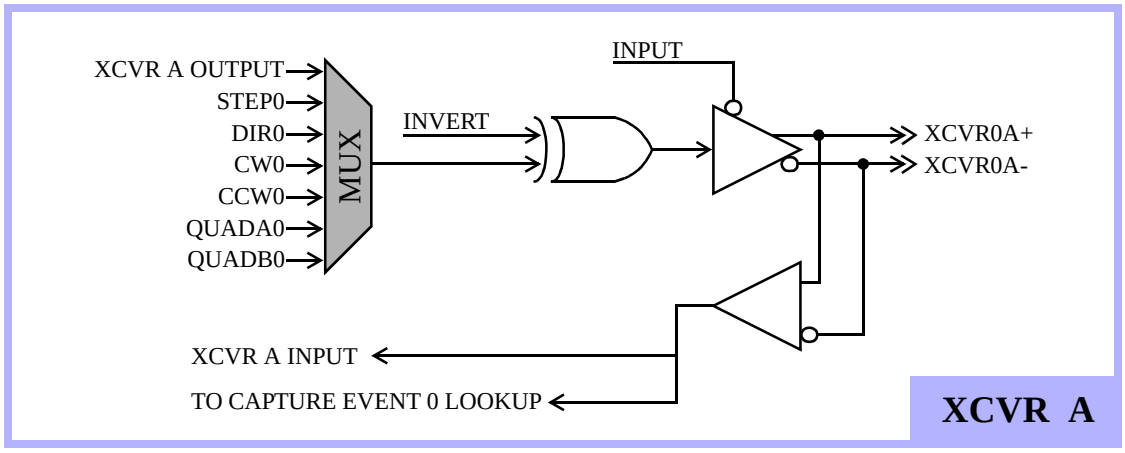


Table 4-20. XCVR B functions.

MOTOR 0 XCVR B Functions (connector signal pair XCVR0B+ and XCVR0B-)		
Function	***Signal Direction***	***Comments***
OUTPUT	OUTPUT	General purpose output
STEP	OUTPUT	STEP signal for MOTOR 0 stepper
DIR	OUTPUT	DIRECTION signal for MOTOR 0 stepper
CW	OUTPUT	CLOCKWISE signal for MOTOR 0 stepper
CCW	OUTPUT	COUNTER-CLOCKWISE signal for MOTOR 0 stepper
QUADA	OUTPUT	QUADRATURE A signal for MOTOR 0 stepper
QUADB	OUTPUT	QUADRATURE B signal for MOTOR 0 stepper
INPUT	INPUT	General purpose input -or- To CAPTURE EVENT 0 input lookup table

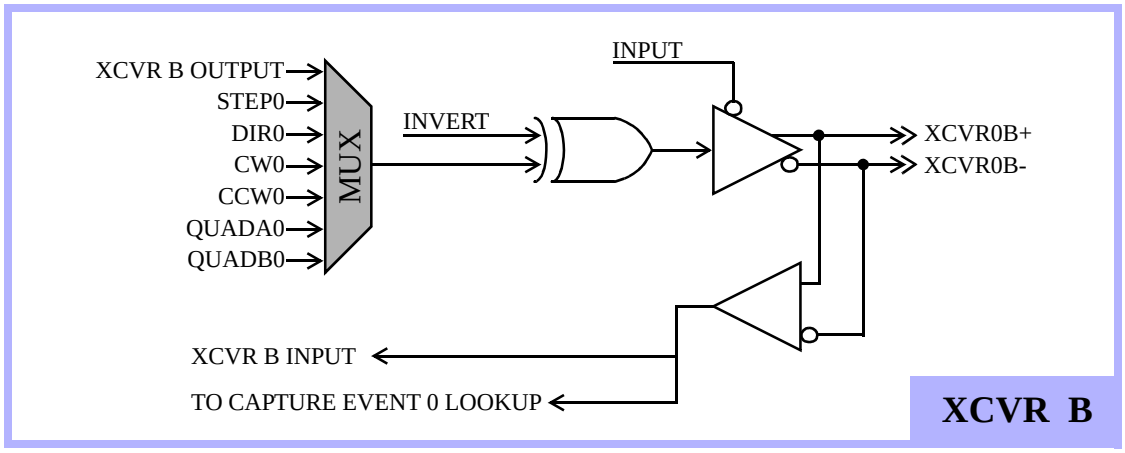
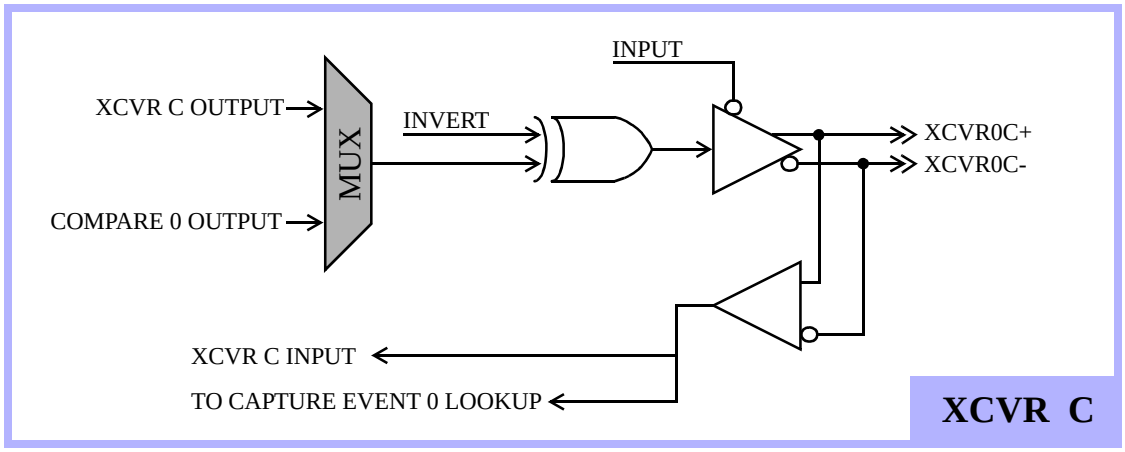


Table 4-21. XCVR C functions.

MOTOR 0 XCVR C Functions (connector signal pair XCVR0C+ and XCVR0C-)		
Function	***Signal Direction***	***Comments***
OUTPUT	OUTPUT	General purpose output
COMPARE	OUTPUT	Compare output signal for MOTOR 0
INPUT	INPUT	General purpose input -or- To CAPTURE EVENT 0 input lookup table



Note	XCVR D, E and F are available only on the XMP-CPCI-3U board. They are not available on the XMP-PCI or XMP-CPCI.

Table 4-22. XCVR D functions.

MOTOR 0 XCVR D Functions (connector signal pair XCVR0D+ and XCVR0D-)		
Function	***Signal Direction***	***Comments***
OUTPUT	OUTPUT	General purpose output
STEP	OUTPUT	STEP signal for MOTOR 0 stepper
DIR	OUTPUT	DIRECTION signal for MOTOR 0 stepper
CW	OUTPUT	CLOCKWISE signal for MOTOR 0 stepper
CCW	OUTPUT	COUNTER-CLOCKWISE signal for MOTOR 0 stepper
QUADA	OUTPUT	QUADRATURE A signal for MOTOR 0 stepper
QUADB	OUTPUT	QUADRATURE B signal for MOTOR 0 stepper
INPUT	INPUT	General purpose input

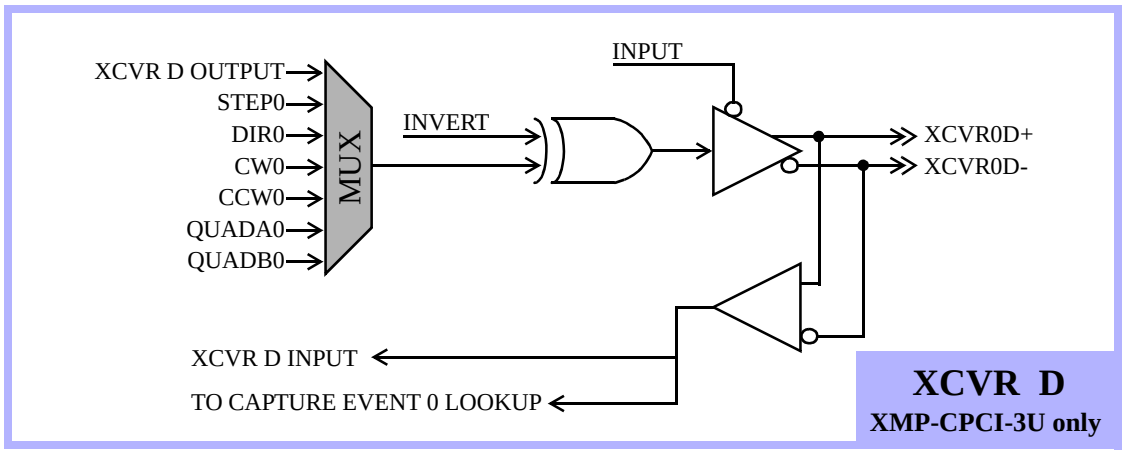


Table 4-23. XCVR E functions.

MOTOR 0 XCVR E Functions (connector signal pair XCVR0E+ and XCVR0E-)		
Function	***Signal Direction***	***Comments***
OUTPUT	OUTPUT	General purpose output
STEP	OUTPUT	STEP signal for MOTOR 0 stepper
DIR	OUTPUT	DIRECTION signal for MOTOR 0 stepper
CW	OUTPUT	CLOCKWISE signal for MOTOR 0 stepper
CCW	OUTPUT	COUNTER-CLOCKWISE signal for MOTOR 0 stepper
QUADA	OUTPUT	QUADRATURE A signal for MOTOR 0 stepper
QUADB	OUTPUT	QUADRATURE B signal for MOTOR 0 stepper

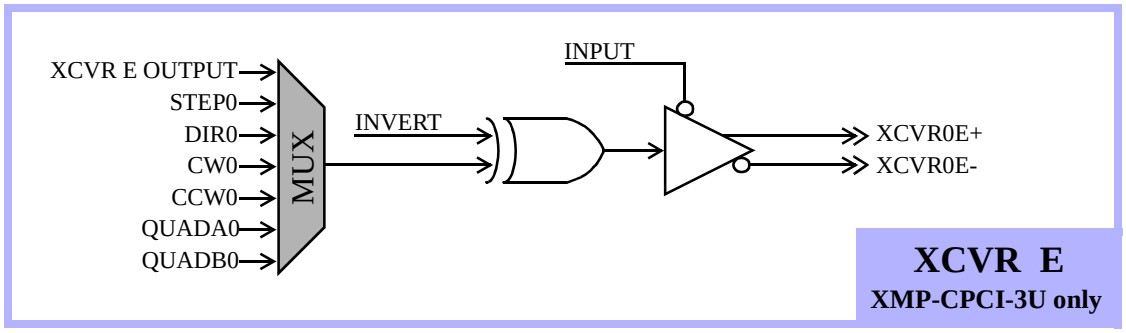
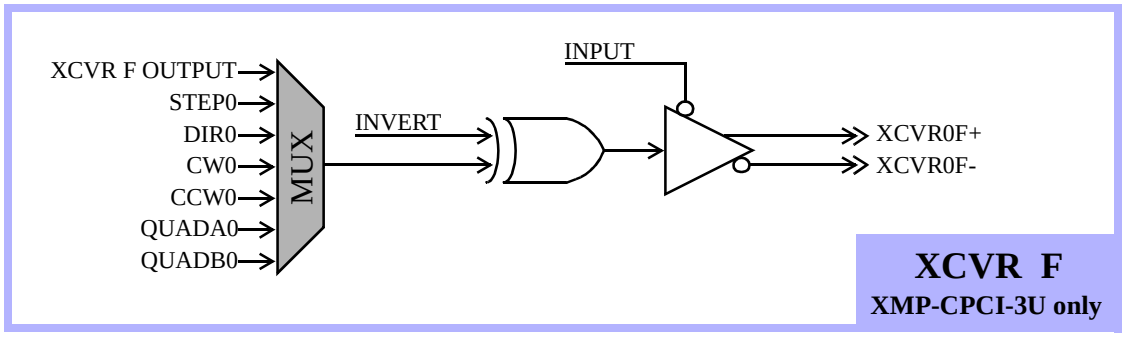


Table 4-24. XCVR F functions.

MOTOR 0 XCVR F Functions (connector signal pair XCVR0F+ and XCVR0F-)		
Function	***Signal Direction***	***Comments***
OUTPUT	OUTPUT	General purpose output
STEP	OUTPUT	STEP signal for MOTOR 0 stepper
DIR	OUTPUT	DIRECTION signal for MOTOR 0 stepper
CW	OUTPUT	CLOCKWISE signal for MOTOR 0 stepper
CCW	OUTPUT	COUNTER-CLOCKWISE signal for MOTOR 0 stepper
QUADA	OUTPUT	QUADRATURE A signal for MOTOR 0 stepper
QUADB	OUTPUT	QUADRATURE B signal for MOTOR 0 stepper



User I/O

There is one USER I/O signal associated with each motor (4 per motion block, 8 for an 8-axis board, 16 for a 16-axis system). These general purpose, opto-isolated signals may be used as either an input or an output. Selecting between input or output is done via the electrical polarity of the external circuit: the polarity of an input circuit disables the output (the output transistor cannot turn on with $V_{collector} < V_{emitter}$), while the reverse polarity enables the output and disables the input (due to a blocking diode). The circuit design guarantees all components remain within operating limits through the full range of possible values (i.e., with V_{in} = 24 V, both input and output parts are within spec).

Table 4-25. User I/O.

Board	***Motion Block***	***Available functions (choose one)***	***Signal Names at Connector***
Main Board	Motion Block 0	MOTOR0 USER INPUT ***OR*** MOTOR0 USER OUTPUT	USERIO_A0 ***AND*** USERIO_AO_RTN
		MOTOR1 USER INPUT ***OR*** MOTOR1 USER OUTPUT	USERIO_A1 ***AND*** USERIO_A1_RTN
		MOTOR2 USER INPUT ***OR*** MOTOR2 USER OUTPUT	USERIO_A2 ***AND*** USERIO_A2_RTN
		MOTOR3 USER INPUT ***OR*** MOTOR3 USER OUTPUT	USERIO_A3 ***AND*** USERIO_A3_RTN
	Motion Block 2	MOTOR4 USER INPUT ***OR*** MOTOR4 USER OUTPUT	USERIO_B0 ***AND*** USERIO_B0_RTN
		MOTOR5 USER INPUT ***OR*** MOTOR5 USER OUTPUT	USERIO_B1 ***AND*** USERIO_B1_RTN
		MOTOR6 USER INPUT ***OR*** MOTOR6 USER OUTPUT	USERIO_B2 ***AND*** USERIO_B2_RTN
		MOTOR7 USER INPUT ***OR*** MOTOR7 USER OUTPUT	USERIO_B3 ***AND*** USERIO_B3_RTN

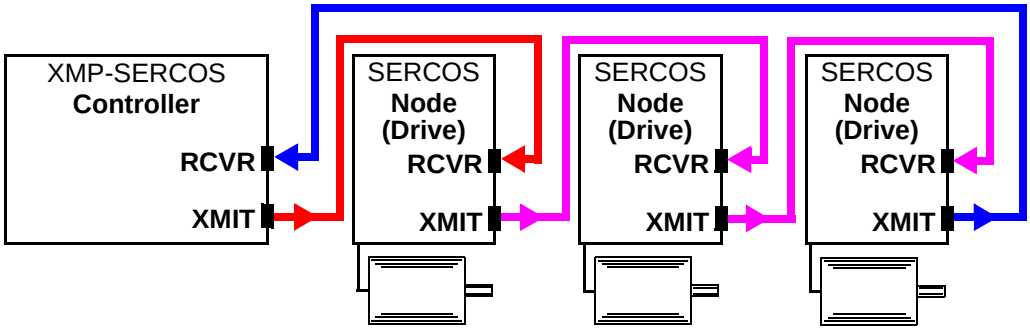
Table 4-25. User I/O.

Board	***Motion Block***	***Available functions (choose one)***	***Signal Names at Connector***
Expansion Board	Motion Block 1	MOTOR8 USER INPUT ***OR*** MOTOR8 USER OUTPUT	USERIO_C0 ***AND*** USERIO_C0_RTN
		MOTOR9 USER INPUT ***OR*** MOTOR9 USER OUTPUT	USERIO_C1 ***AND*** USERIO_C1_RTN
		MOTOR10 USER INPUT ***OR*** MOTOR10 USER OUTPUT	USERIO_C2 ***AND*** USERIO_C2_RTN
		MOTOR11 USER INPUT ***OR*** MOTOR11 USER OUTPUT	USERIO_C3 ***AND*** USERIO_C3_RTN
	Motion Block 3	MOTOR12 USER INPUT ***OR*** MOTOR12 USER OUTPUT	USERIO_D0 ***AND*** USERIO_D0_RTN
		MOTOR13 USER INPUT ***OR*** MOTOR13 USER OUTPUT	USERIO_D1 ***AND*** USERIO_D1_RTN
		MOTOR14 USER INPUT ***OR*** MOTOR14 USER OUTPUT	USERIO_D2 ***AND*** USERIO_D2_RTN
		MOTOR15 USER INPUT ***OR*** MOTOR15 USER OUTPUT	USERIO_D3 ***AND*** USERIO_D3_RTN

CHAPTER 5
XMP CONTROLLER-MOTION DRIVE I/O: SERCOS

Introduction

SERCOS (SErial Real-time COmmunication System) is an optical networking standard for motion controllers and drives. It confers a number of advantages over electrically based, analog forms of control. Most notably, its fiber optic mode of communication reduces noise problems seen in electrical systems, while offering very high signal throughput. SERCOS devices are serially connected (i.e., “daisy-chained”) in a ring configuration via “nodes,” each node having its own dedicated receiving and transmitting ports. Data is sent in “telegrams,” which contain either control or status information. These are sent by the controller (master) then received, processed, and relayed by each node (slave) in the ring.



Normally configured XMP-SERCOS systems connect up to eight (8) nodes. An additional four nodes may be connected for a maximum total of twelve (12) nodes. This lends tremendous flexibility to SERCOS-based motion control systems. In the network example above, a single ring is featured with three separate drives, each having its own dedicated node. Almost any combination of nodes and drives is possible; however, the final configuration and number of drives within the system is limited by the required cycle time of each node and the controller. To determine the optimal SERCOS configuration for your motion control application, please contact Motion Engineering.

Besides parsing control data and sending feedback, each drive’s on-board processor offers additional capabilities. Depending on the manufacturer, SERCOS drives have the ability to close a position, velocity, and/or torque loop, execute internal homing procedures, as well as latch position based on digital inputs.

Despite the fact that the XMP-SERCOS controller has a different physical interface than the standard XMP controller, the same MPI is used for both controllers. The complexity of the XMP-SERCOS controller's interface has been hidden by the MPI and by the firmware running on the XMP. The same development tools can be used for both controllers. The XMP-SERCOS controller requires an initialization procedure before telegrams can be sent to amplifiers. Once the initialization procedure has successfully executed, XMP-SERCOS controllers act the same as standard, analog XMP controllers.

Design Limitations

Despite its many advantages, SERCOS does not confer "something for nothing." XMP-SERCOS controllers share some of the same limitations as analog controllers:

- XMP-SERCOS controllers cannot obtain more from a drive than it's designed to deliver. When selecting a SERCOS drive, the designer must scale their selection to the hardware-software requirements of the system. Different drive systems feature different capabilities and processing speeds. SERCOS errors arise when a drive is commanded to do something it cannot do.

- Although most SERCOS drives share a number of standardized user identification numbers (IDNs), most drive manufacturers assign their own unique IDNs. If you change drives, IDNs can be expected to change also.

- The selection of cable length and material (acrylic vs. glass), transceiver wavelength and power, connector types, etc. all bear upon the success of a SERCOS network. Hardware flaws at any point in the SERCOS ring will affect the performance of the entire system.

Controller-Drive Compatibility

Depending upon your choice of motion drives to be used with the XMP-SERCOS controller, you should consider some or all of the below parameters:

- Optical power
- Data rate
- Cycle time
- IDNs

Optical wavelength and power

Motion Engineering's XMP-SERCOS controller uses Agilent Technologies HFBR-1505A/2505A transceivers for transmitting and receiving data.[1] Transceivers used on industry SERCOS drives may differ, depending upon the manufacturer.

Transmitter

The XMP-SERCOS controller transmitter , (HFBR-1505) incorporates an LED, transmitting at a nominal wavelength of 650 nm (± 10 nm). Optical peak output power (P_t) is software adjustable from -18.0 to -5.5 dBm. NOTE: power is affected by temperature: higher transmitter temperatures yield slightly decreased power.

Receiver

The Agilent HFBR-2505A receiver's peak input power varies from -20 dBm (Level Logic HIGH) to -42 dBm (Level Logic LOW). Data corruption may result if the receiver is overdriven (inputted optical power too high), or underdriven (inputted optical power too low). These conditions are discussed in the next section.

Data Rate, Cycle Times and Configuring SERCOS

The number of devices capable of being operated by one XMP-SERCOS controller will depend upon the number and cycle times of drives and the number of networked nodes. It may prove advantageous to consolidate two or more drives into a lesser or greater number of nodes, depending upon the task(s) to be accomplished. To receive guidance on optimizing your SERCOS network, please contact Motion Engineering.

1. Agilent Technologies is a subsidiary of Hewlett-Packard. Data sheets for HFBR-1505A/2505A transceivers are available from Agilent via their web site: www.agilent.com.

IDNs

SIDNs and PIDNs

IDNs, or Identification Numbers, are numbers identifying data that can be transmitted between SERCOS nodes and hosts. Design of a successful SERCOS network requires a thorough knowledge of all IDNs used.

There are two types of IDNs. **SIDNs** (standard IDNs) are defined by the SERCOS specification. These IDNs represent data that is commonly used in drive implementations. It should be noted that it is up to the manufacturer to decide which SIDNs it will support. **PIDNs** (product IDNs) are defined by the manufacturer. These IDNs represent data used in the drive for features not discussed in the SERCOS specification. PIDNs vary widely between drives and manufacturers.

Drive Initialization

During initialization of a SERCOS network, many drives require the setting of certain IDNs. Please consult the drive manufacturer's documentation concerning required IDN settings.

SERCOS Connection Hardware

SERCOS Cabling

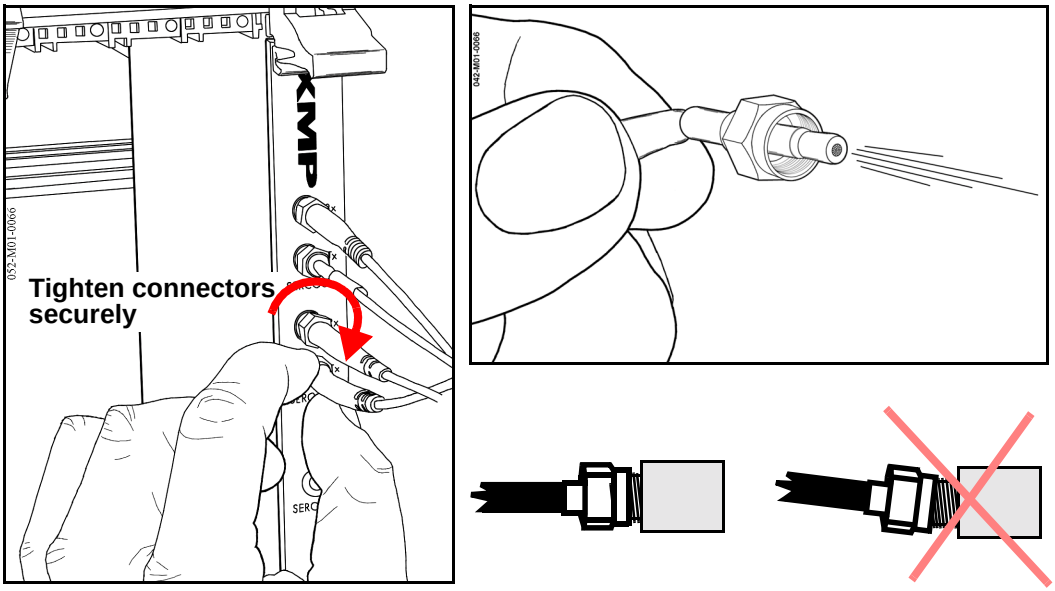
Variations in cable length and material have a profound influence upon power attenuation within networks:

- **Fiber optic cable length**— the longer the cable, the more optical power it absorbs.
- **Fiber optic cable material (glass vs. acrylic)**— different materials absorb different quantities of light, and react differently to changes in temperature, bend radius, etc.
- **Temperature**— affects transceiver and cable characteristics.
- **Cable movement / bend radius**— as fiber optic cables are moved between motion components, their bend radii will vary. This, in turn, can attenuate optical signals. Generally, the smaller the bend radius, the higher the power attenuation. Cables attain their maximum optical efficiency when they are straight.

Specifications for SERCOS fiber-optic cable are found in Section 5.3.3 of the SERCOS interface standard (IEC 64191); however, network designers should consult their vendors for exact cable characteristics. Depending upon the cable material used (glass vs acrylic), length and bend radius, performance will vary. Section 5.3.5 of IEC 61491 specifies system data of the optical transmission path. Conforming cables must meet this specification.

SERCOS Connectors

XMP-SERCOS controllers utilize screw-on connectors having 1/4 - 36 UNS 2A threads. Typically, transceivers use molded plastic threads, while many cable connectors are steel; therefore, use caution to prevent cross-threading and stripping.

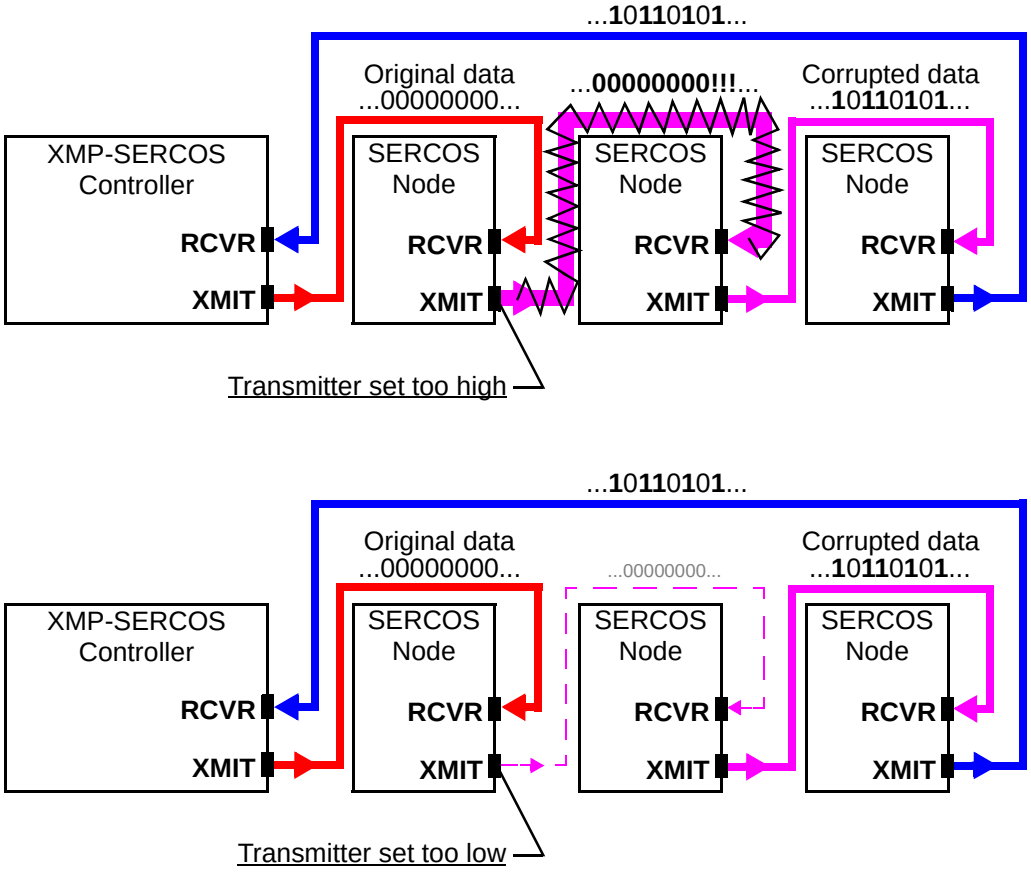


Loose connectors do not efficiently couple the fiber-optic cable's light energy to a transceiver and can cause data corruption. Under no circumstances should connectors be left loose. (This includes situations where a transceiver's power output is too high and can be reduced by loosening the connector; this is *not* a reliable way to adjust power output!)

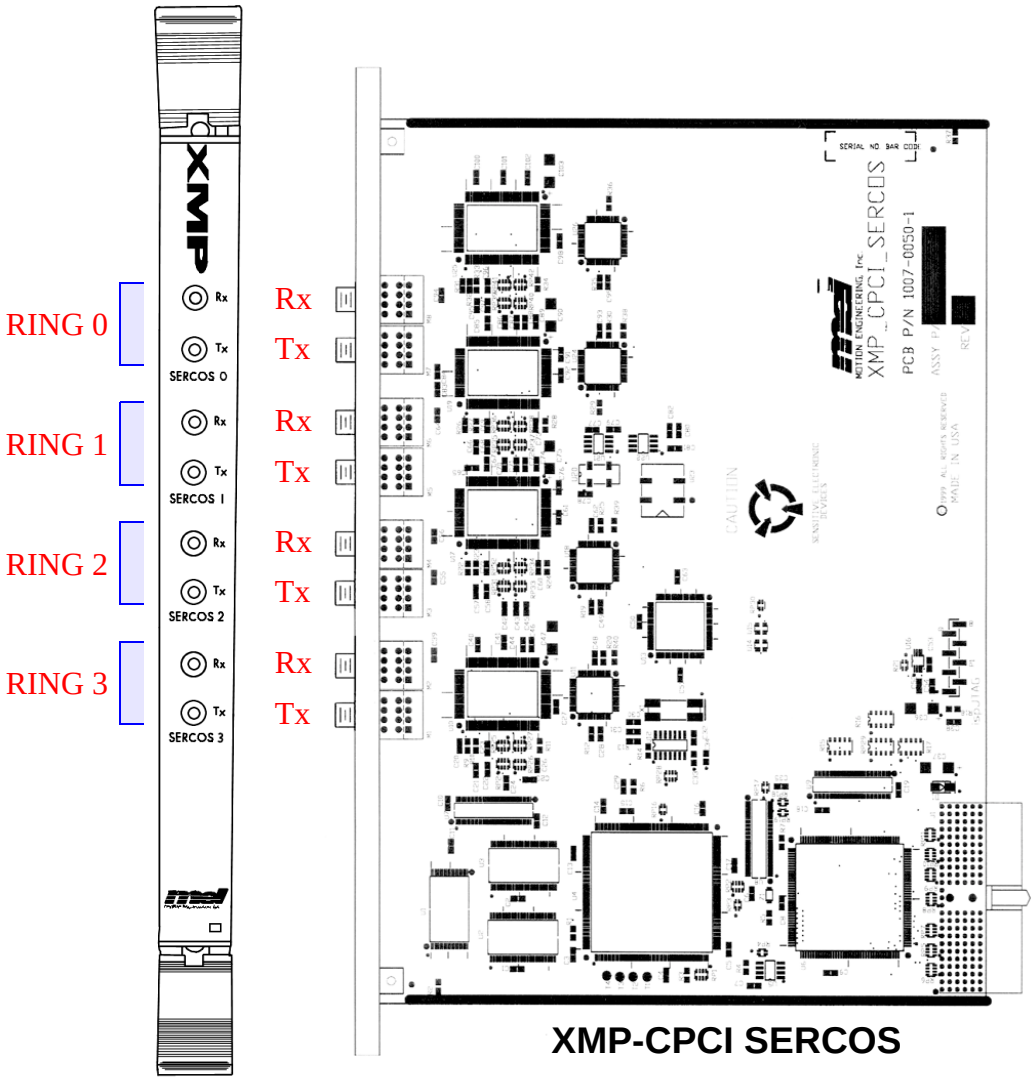
Setting Transmitter Power

An integral part of building a SERCOS ring is establishing optimal transmitting power for each device in the ring. Starting at the XMP controller, an optical control signal is transmitted at a specified power level. The first device in the ring receives this signal, copies and processes it, then generates a new transmission. The next device in the ring receives this secondary transmission, processes it, and so on. This relay continues through every device until the controller becomes the last device in the ring. The controller processes the telegrams it has received, calculates new control commands, then transmits new telegrams during the next data cycle. The process repeats until halted by the controller, or an interruption in the ring causes a fault.

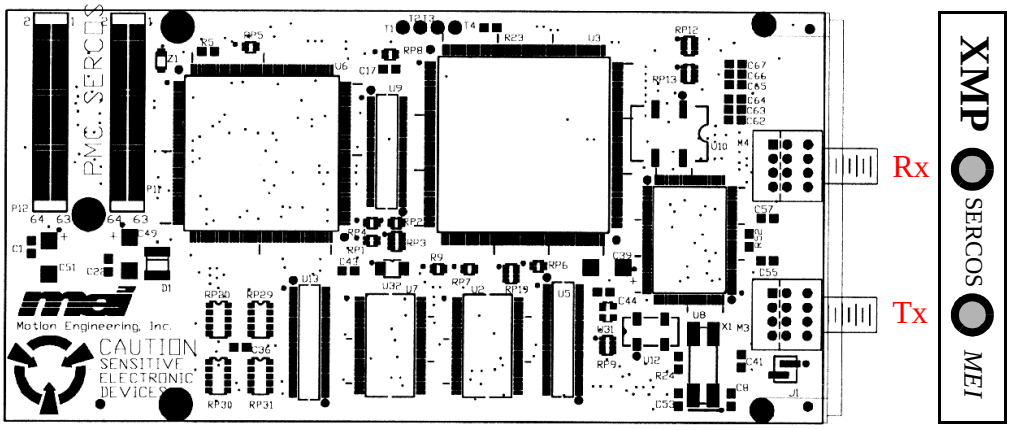
When building SERCOS networks, it is crucial to match the power of each device's transmitter with the downstream device's receiver. If power levels are set too low or too high, data corruption can quickly result--a common cause of errors and problems in SERCOS networks.



XMP-SERCOS Connector Ports



XMP-CPCI SERCOS



XMP-PMC SERCOS

Connector ports on XMP-SERCOS controllers are labeled “Tx” (transmitter) and “Rx” (receiver). The diagram above shows connector locations for both the XMP-SERCOS-PMC and XMP-CPCI-SERCOS controllers. The XMP-SERCOS-PCI-S1 and -S2 controllers are not shown.

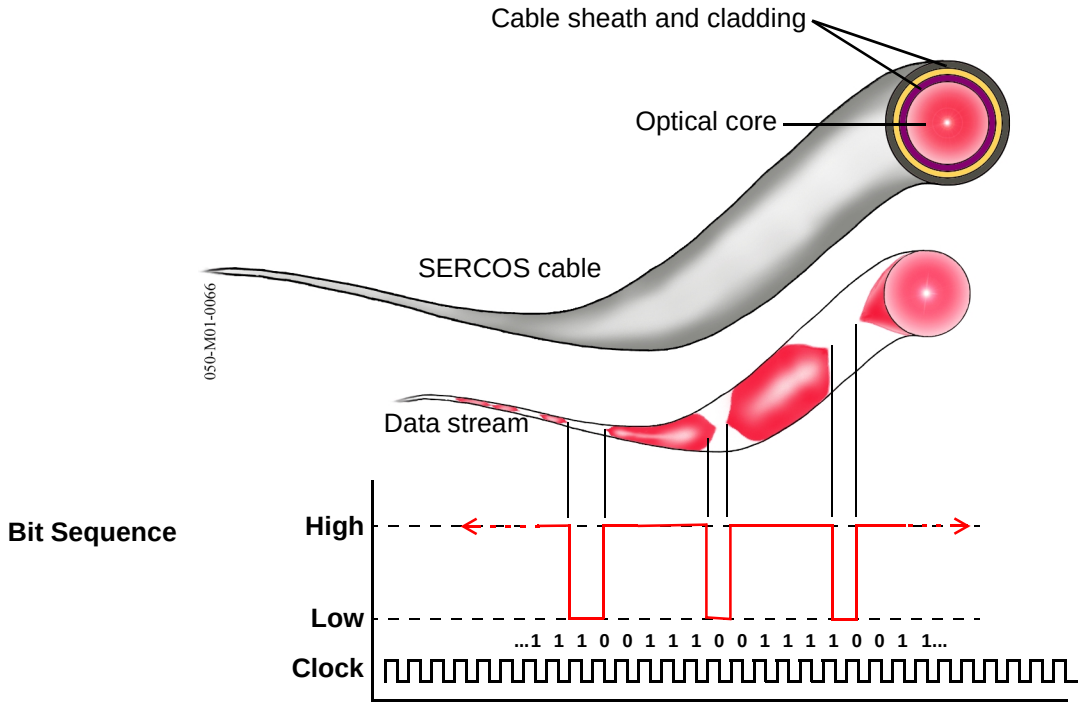
SERCOS cables are connected by inserting them gently into their ports and rotating the metal ferrule until snug. The XMP controller and all SERCOS devices should be configured in a ring (or “daisy chain”) such that the “Tx” port of one device is plugged into the “Rx” of a neighboring device in the ring.

Internal SERCOS Controller Hardware and Signal Structure

Both PMC and CPCI XMP-SERCOS controllers are designed to be firmware- and software-compatible with the standard MPI/XMP libraries, while providing control to the SERCOS ASIC. (All of the motion blocks normally found in the XMP are available on these controllers.) The XMP-SERCOS-PMC and XMP-SERCOS-CPCI controllers each contain SERCOS ASICs, opto couplers, and control logic sufficient to control 1 to 4 independent SERCOS rings. Each ring is capable of independently supporting 2, 4, 8, 10, or 16MHz data rates.

Bit Coding

SERCOS signals are conveyed optically as a series of discretely modulated light pulses from each device’s transmitter to the adjoining device’s receiver. Information is NRZI-coded (Non Return to Zero Inverted), with a retrievable clock signal. Information is encoded using a **0** where there is a transition, and a **1** where there is no transition. To ensure that clock pulses remain retrievable from the transmitted signal, zeros are forced (bit-stuffed) into the data stream every eight bits.



A full description of NRZI coding and SERCOS is included within Section 5.5 of the publication entitled “Electrical Equipment of Industrial Machines--Serial Data Link for Real-time Communication Between Controls and Drives” from the International Electrotechnical Commission.

For Assistance with SERCOS Networks...

Because of variation in motion control drives and designs, no two SERCOS networks are exactly the same. Motion Engineering's Applications Department has accrued a breadth of experience in the design and troubleshooting of SERCOS networks and is ready to assist you. When contacting MEI, please be prepared to provide the following information:

System

1) Number of drives in system.

2) Type(s) of cable (glass, plastic, etc....).

3) Cable lengths.

Controller

1) Model and hardware version.

2) Firmware version.

3) Software (MPI) version.

4) Sample code demonstrating problem (if any). Sample application can be an MEI application.

Drive

1) Drive manufacturer.

2) Drive model.

3) Drive firmware version.

4) Drive operation mode (position, velocity, torque, etc....).

5) Contact information for drive manufacturer.

This is a blank page.

CHAPTER 6
XMP ARCHITECTURE

Introduction

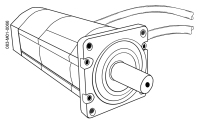
This chapter presents a general description of the XMP controller's architecture. It is intended to help users better understand the XMP's inner workings, and also explains how hardware functions with external components such as host computers, drives, and encoders. Many components discussed in this chapter are detailed in other sections of the manual. To learn more about a specific component, please refer to the index.

Architectural Overview

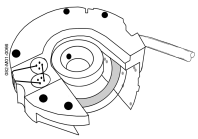
The XMP controller is a remarkably integrated device, designed to link a host computer with industry-standard motion drives. While many motion controllers heavily task the host computer's CPU to calculate motion paths and execute commands, the XMP performs much like a computer-within-a-computer to free the host of these tasks. Because the controller dedicates itself 100 percent to the task of motion control, the host computer is tasked only with communicating high-level commands; it remains free to oversee other tasks. Meanwhile, your machine's motion drive is ensured the safest, most rapid and precise motion control possible. The XMP controller achieves this feat through a fully integrated architecture, consolidating all essential control components and subsystems onto one board.

Computer Hosts— XMP controllers can be hosted by both PCI and CompactPCI (CPCI) computers via a 32-bit direct memory interface. Remote hosting via an Ethernet (TCP/IP) connection is also available.

Digital Signal Processor (DSP)— The hardware centerpiece of the XMP is the SHARC DSP (*S*uper *H*arvard *AR*chitecture *C*omputer *D*igital *S*ignal *P*rocessor). By taking full advantage of the DSP's horsepower, efficiency is optimized, by splitting the motion tasks between the host and XMP.



Motor Control— Each controller is capable of operating up to 16 axes (independently configurable to control a servo or stepper motor) at a sample rate of 5 KHz, or 8 axes at a sample rate of 10 kHz. Included with each axis is a high speed compare output, and a high speed capture or registration input. Each axis can also be configured to sinusoidally commutate servo motors via two analog outputs. Full tuning capability is offered with PID and advanced, post-PID filtering options for each axis, including gain-scheduling and dual-loop control. Stepper output rates can be as high as 2.5 MHz. This direct-digital pulse train can control an open or closed loop motor, enabling frequency-controlled servo operation. Axes can be configured for step-direction or clockwise-counterclockwise operation.



Motion Feedback— Motion feedback can be standard quadrature from an incremental encoder (10 MHz max, 40 MHz count rate), ±10 volt analog differ-

ential signal, or analog encoder signal routed to a 4096x Scale Interpolation Module (SIM4).

Motion Programming Interface (MPI)— The MPI (Motion Programming Interface) has two built-in software objects for easy real-time motion application control: the Event Manager and Program Sequencer. The Event Manager communicates to the XMP via the bus interface and interrupts. Multi-threaded applications can wake or put to sleep multiple threads based on user-defined events or standard MPI events, such as emergencies. The Event Manager facilitates efficient multi-threaded applications. The Program Sequencer allows onboard autonomous programs to execute instructions during each DSP sample period. Onboard XMP-handled programs require no host intervention for optimal host-processor efficiency and guaranteed time response.

Safety— Dedicated, DSP-based motion control optimizes safety: the controller's on-board watchdog timers constantly monitor handshaking, and a dedicated, general purpose E-stop input is at the ready. Each axis also has configurable, optically-isolated software and hardware limits up to 24 volts.

Clock— A phase-locked loop (PLL) clock generator provides the master clock to the DSP, PCI interface, state machines and FPGAs.

Analog Power Supply— Clean analog power is generated on ±15V power rails from Vcc. Each 8-axis controller has its own analog power supply.

XMP Components

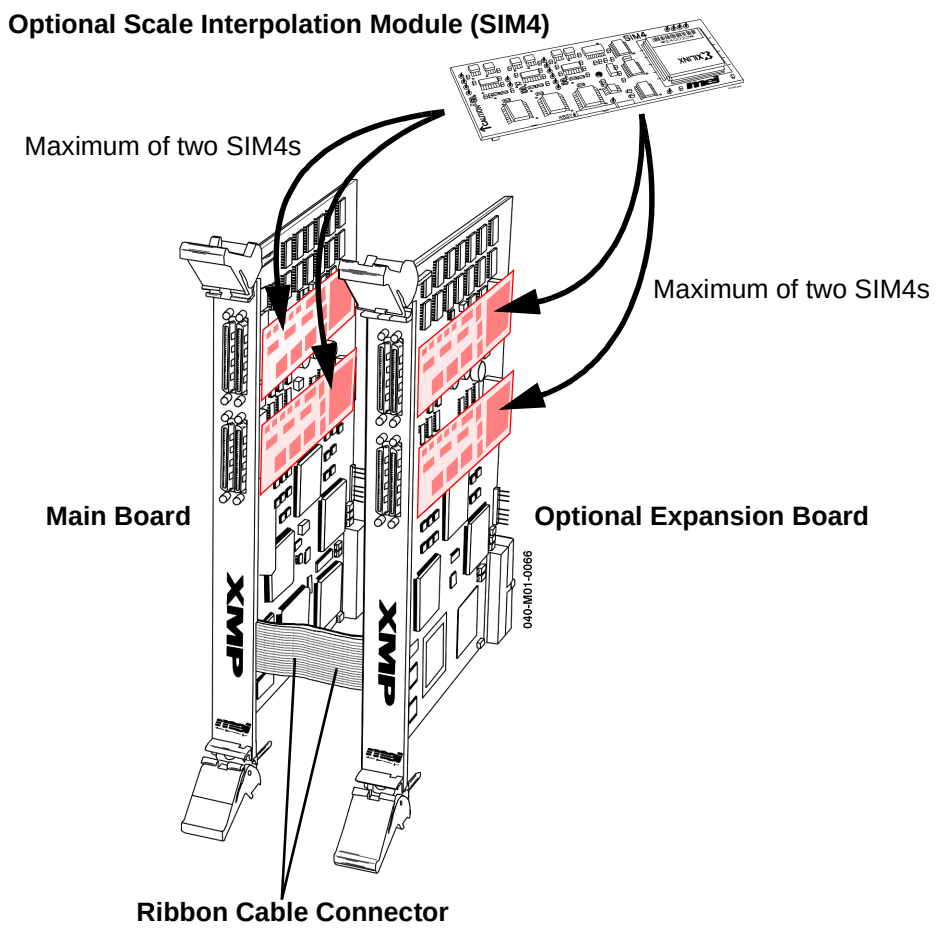
Three key components comprise the XMP family: main boards, expansion boards, and SIM4 modules. Every XMP controller starts with one main board. Other components are added as needs dictate.

XMP main board— Every XMP controller consists of at least this component; in some cases (XMP-CPCI-3U and XMP-SERCOS-PMC) it is the *only* component. Each main board includes a SHARC DSP processor subsystem with up to two motion blocks each.

XMP expansion board (optional)— Connects directly to one main board (XMP-PCI and XMP-CPCI-6U only) via a ribbon cable. Adds up to two more motion blocks, with 4- and 8-axis models available.

SIM4 scale interpolation module (optional)— Plugs directly into main and/or expansion boards (XMP-PCI and XMP-CPCI-6U only) via mezzanine connectors. Up to two SIM4s may be added to each main or expansion board.

Figure 6-1. XMP boards and SIM4 modules.



Host Bus

Host Bus Interface

Data bussing between XMP controllers and hosting computers is similar in all cases, regardless of whether the controller is configured for PCI, CPCI, or PMC busses, or the choice of operating systems. The host communicates with the XMP using direct memory reads/writes over a 32-bit bus. The full DSP internal memory (128k bytes), external SRAM, and FLASH memory are accessible to the host, in a 32-bit flat memory window of 8 Mbytes in the PCI (or CPCI) memory map. The host bus interface includes system registers for host control of *Reset*, *Write* and *Interrupt* enables, system configuration, control and status.

Encoder Inputs

Each motion block provides up to five encoder interfaces (4 axes plus 1 auxiliary), with the following features:

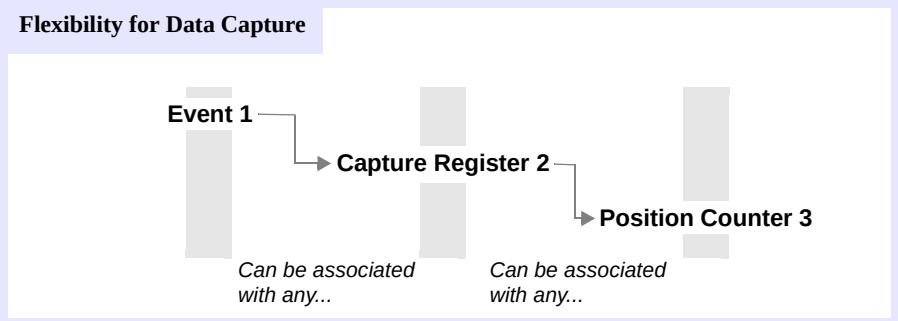
- 20 MHz maximum count rate (5 MHz quadrature cycle)
- Digital noise filtering
- Broken wire and illegal state detection

Having five (5) encoders allows an encoder to each axis, plus a fifth encoder for master-slave, master-cam or dual-loop operations. Each motion block has features for position capture and position compare. Digital noise filtering is enabled by default and limits the maximum count rate to 10 MHz (2.5 MHz quadrature cycle).

Position Capture

Each motion block maintains a set of ten 32-bit data capture registers (in hardware), used to store latched positions. The ten data capture registers are shared among the five encoder interfaces of each motion block. A specific capture register can be associated with any encoder (position) counter. Each capture register can be triggered by one of the five events (one from each of the five encoder interfaces). One event can trigger any number of captures, allowing probe capture of multiple axes.

Figure 6-2. Events/Data Capture Registers/Position Counters



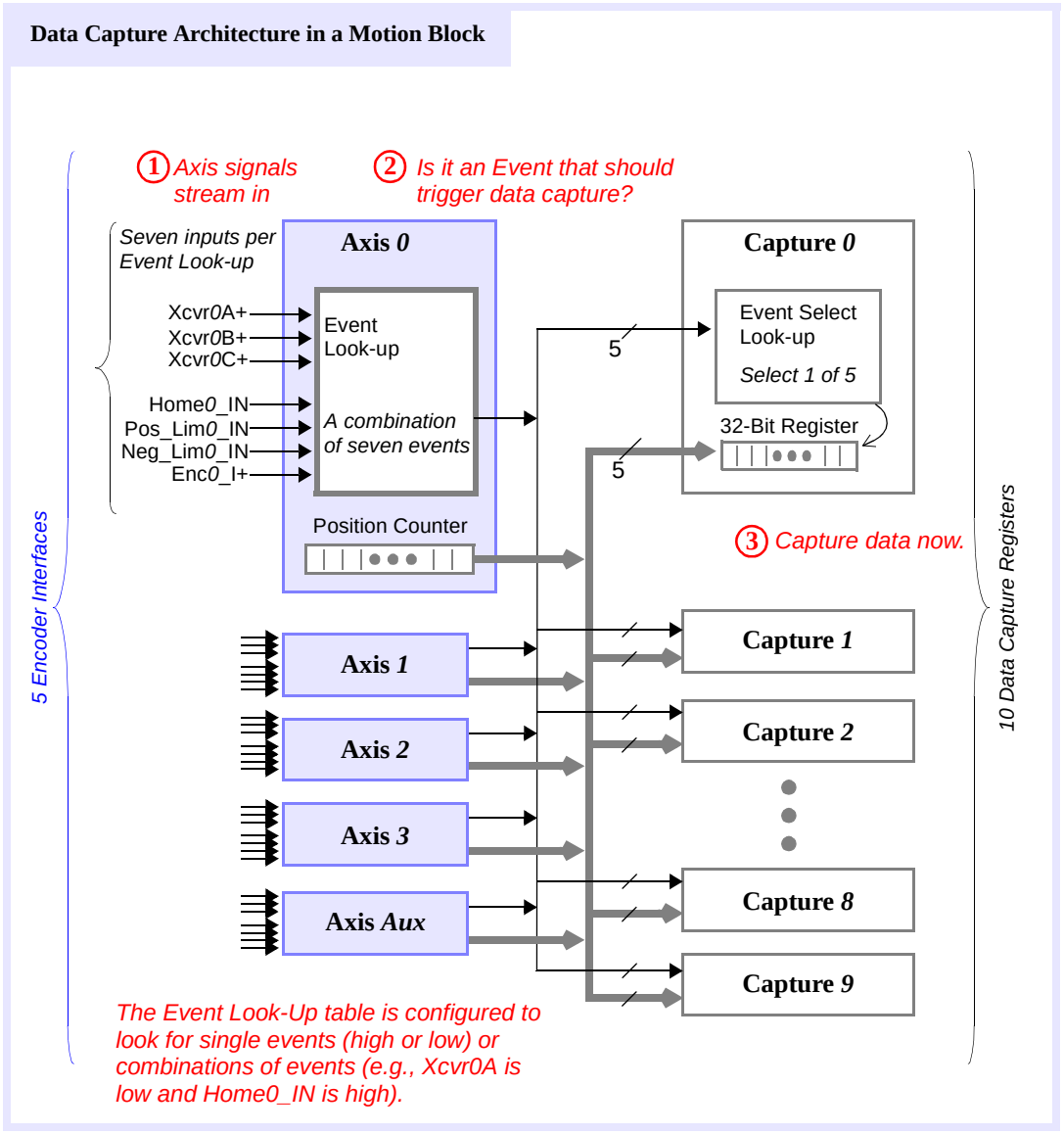
The free-running, 32-bit position counter for any axis can be latched on the edge of an input. The latency for position capture is sub-microsecond, because the registers are implemented in hardware. The input source and polarity are configurable. The capture event can trigger on any of:

- Twelve available RS-422 transceiver inputs per motion block
- Dedicated *Index_in* (per axis)
- Dedicated *Home_in* (per axis)

Additionally, the capture event trigger can be enabled or disabled with software.

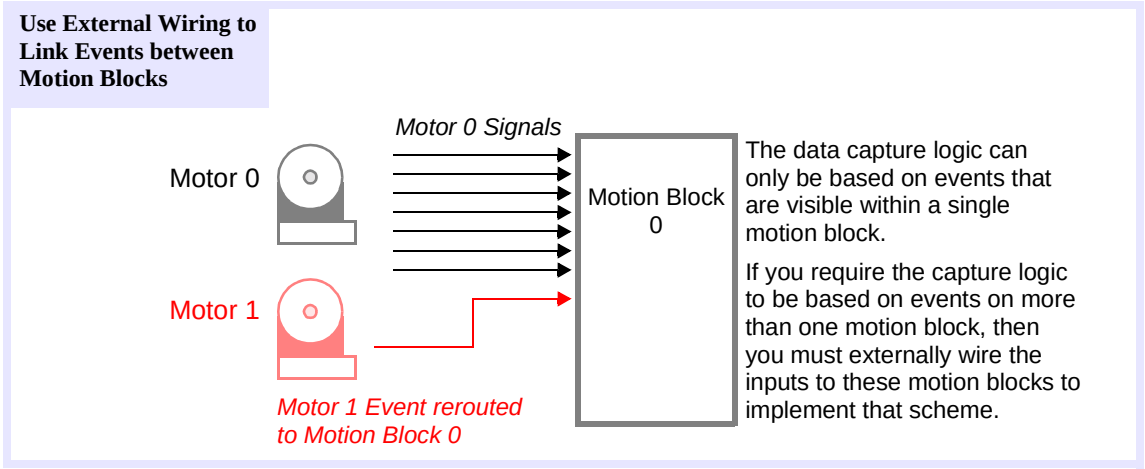
Capture logic may be triggered from any one of the five Event look-up tables in a given motion block FPGA (one Event look-up table per axis, plus one spare table). Each Event look-up table has seven inputs. These signals are the three Xcvr signals for that axis plus Home, Pos_LIM, Neg_LIM and Index. The Event look-up table can be configured to use either polarity of any one input or combinations of inputs (for example, capture data if Home is *low* and Index is *high*).

Figure 6-3. Event processing for data capture.



Although the Capture 0 register is normally used to capture the encoder (position) count for Axis 0, you can also use Capture 0 to capture the encoder counts for the other axes in that motion block (Axis 1,2,3), individually or as a group. Note that Capture events cannot cross between motion blocks, although you can work around this by externally wiring the axis inputs among multiple motion blocks.

Figure 6-4. External wiring between motion blocks.



Position Compare

Each motion block has a hardware register file of sixteen 32-bit registers which are used to provide compare setpoints. The sixteen registers are shared among the four axes. Buffer modes can be set up as FIFO or circular, so that compare events can occur faster than the servo update rate. Compare latency is less than 1 µsec.

The compare event can set a *Position_Compare* output on any of the twelve available RS-422 transceiver outputs. The compare event status is also available to the DSP. The trigger event is software-configurable and will occur when the value loaded in the position compare register is either:

greater than the actual position
equal to the actual position
less than the actual position

In addition, the equality condition can be either transparent or latched. Note that the latched condition can be cleared. The *Position_Compare* output polarity is programmable and the compare output can be disabled. The register can also be read back by the host.

Scale Interpolation Modules (SIM4)

For encoders with sinusoidal outputs, the scale interpolation module provides increased position resolution for up to four axes. These outputs produce one cycle of sine and cosine analog signals for each encoder's *line pair*. At every sample, the scale interpolation module reads the sine and cosine levels and determines the angular position within the line pair. The sine and cosine outputs are also routed to the standard quadrature inputs, providing coarse position information.

The quadrature inputs generate 4 counts for every line pair, and the 12-bit interpolated value generates 4096 counts for each line pair. The full interpolated position is obtained by combining the *number of quadrant counts* divided by 4, with the position between two encoder lines. Essentially, the 12-bit scale interpolation provides a resolution increase of 1024 over the quadrature counters. To maintain accurate phase information, the sine and cosine signals are captured with simultaneous-sampling A/D converters.

Position Compare

To implement the position compare feature, the SIM4 compares the current position to a position latched in the FPGA. The A/D converters convert continuously, with a 10µsec latency.

- 10 compare registers are available per SIM4 (4 axes).
- Can compare two positions per servo cycle on 4 axes, or can compare 10 positions per servo cycle on 1 axis.
- Delay from compare event to compare = 10µsec.

Position Capture

To implement the position capture feature, the SIM4 latches the full interpolated position when the user-supplied latch signal is pulsed. Note that different axes can be latched independently of each other.

- 10 capture registers are available per SIM4 (4 axes)
- Can capture two times per servo cycle on 4 axes, or can capture 10 times per servo cycle on 1 axis
- Delay from capture event to capture = 5 µsec

For detailed information about SIM4 modules, please see *Application Note 206* from Motion Engineering.

Digital-to-Analog Converter (DAC) Subsystem

The command link from the XMP controller to analog input servo drive is the DAC. (In contrast, stepper drives receive commands from the XMP controller via transceivers; DACs are not used for steppers.) The XMP controller can have up to two DACs per axis. DACs are separated into primary and auxiliary DACs. One channel is used for standard servo control ("primary") and the other channel is an auxiliary, OR both channels are used for sinusoidal commutation.

Using the firmware's default settings, motors are mapped to DACs according to the table below. The firmware provides for a fully configured, 16-motor controller (consisting of a linked main and expansion board) by mapping all 16 motors to 32 DACs (16 primary, plus 16 auxiliary). If a controller consists only of one main board (8 motors), the expansion hardware to support the additional motor-DAC mappings (motors 8-15, DACs 8-15, and AuxDacs 8-15) does not exist, and the firmware ignores them.

Table 6-1. Default firmware motor-DAC assignments.

16-axis controllers (main plus expansion board)

8-axis controllers (main board only)

Motor	DAC (Primary)	AuxDac (Auxiliary[a])
Motor 0	0	0
Motor 1	1	1
Motor 2	2	2
Motor 3	3	3
Motor 4	4	4
Motor 5	5	5
Motor 6	6	6
Motor 7	7	7
Motor 8	8	8
Motor 9	9	9
Motor 10	10	10
Motor 11	11	11
Motor 12	12	12
Motor 13	13	13
Motor 14	14	14
Motor 15	15	15

Default firmware maps **Motors 0-15** to **DACs 0-15**, **AuxDacs 0-15**, as shown at left on all XMP controllers (both 8- and 16-axis models); however...

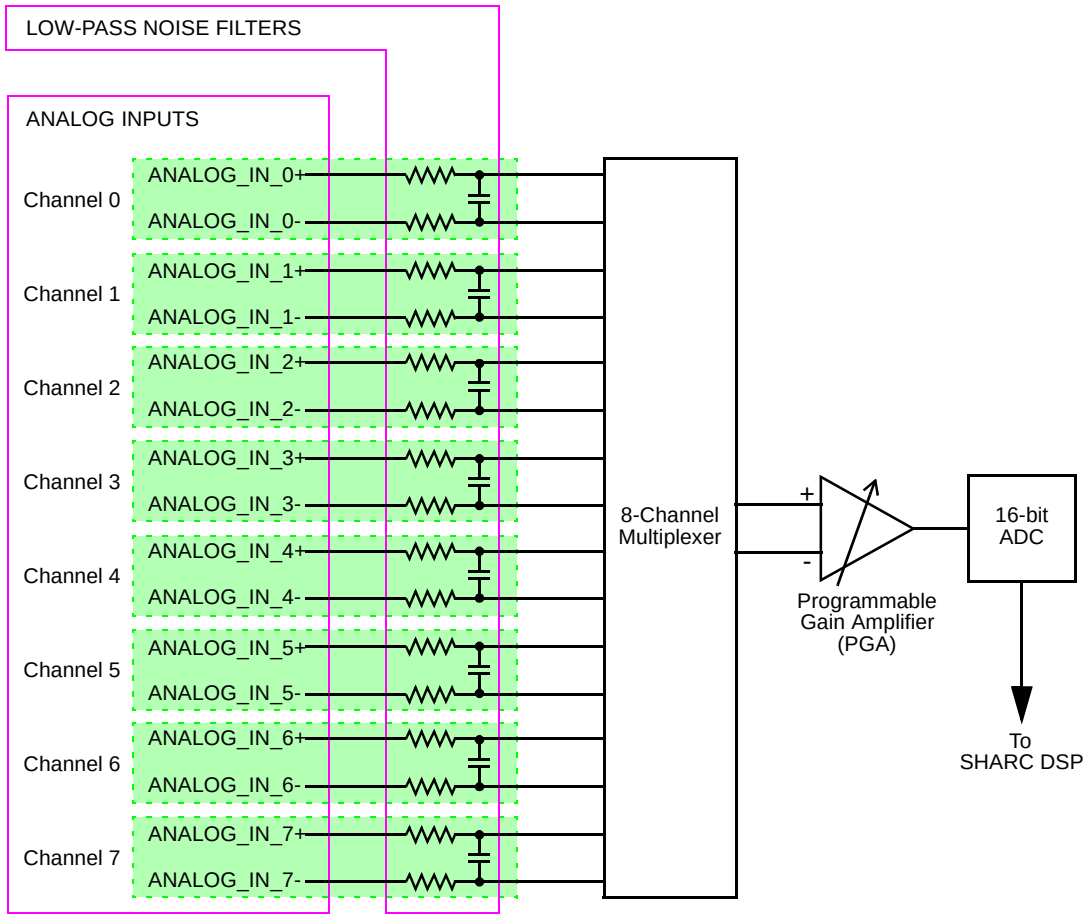
motion block components and hardware for **Motors 8-15**, **DACs 8-15**, and **AuxDacs 8-15** exist only on 16 axis XMP controllers.

[a] One primary DAC is required for servo drives that perform motor commutation. One primary DAC and one auxiliary DAC is required for systems where the controller performs motor commutation.

Analog Inputs

Analog signals, from such external devices as optical sensors and LVDTs, can be fed to the XMP controller. They are wired through dedicated analog lines (e.g., ANALOG_IN_0+ and _0-), then converted to digital values by the XMP controller's **analog-digital converter (ADC)**. The ADC used in XMP controllers is an Analog Devices 16-bit AD976A.

The figure below outlines the analog input subsystem for the XMP controller.



The XMP controller accepts analog inputs having a maximum dynamic range of +10V or -10V, and converts these to 16-bit digital values within the ADC. 17 kHz **noise filtering** at the inputs removes unwanted voltage spikes. Because the highest digital resolution is attained when analog inputs are scaled to 10V, it is advantageous to rescale lower voltage inputs via the integrated **programmable gain amplifier (PGA)**. Gain factors are individually selectable at *gain select* = 1, 2, 4, and 8, for ±10V, ±5V, ±2.5V, and ±1.25V inputs, respectively.

The **multiplexer** provides fault protection as well as channel selection. With no power applied, the switches are OFF. Overvoltage clamps become activated at ±13.5V.

Table 6-2. ADC Subsystem Specifications

Max. Input Voltage	± 10V
Input Ranges	± 1.25V; ± 2.5V; ± 10V; or, ± 5V
Resolution	16 bits
Channels	8
Input Impedance	100kΩ
Signal Bandwidth	DC – 17 kHz
Conversion Time	10µs
Conversion Rate	Once per servo cycle (8 channels)[a]
DC Offset	15mV (max.) on 10V range.
Gain Uncertainty	0.5% (max.) on 10V range.
Nonlinearity	6 LSB

a. Please note that ADC conversion rates vary slightly, depending upon where within the servo cycle an analog change occurs. If an analog input changes near the end of the cycle, the digital conversion may occur sooner after the change than if the change occurs at the very beginning of the servo cycle.

Main Board and Expansion Board

Each XMP controller contains one main control board, capable of operating up to eight (8) axes. The main board contains the SHARC DSP, which processes all data between the host computer and motion hardware. In addition, one optional expansion board may be connected to a main board to add up to eight (8) more axes. Unlike main boards, expansion boards do NOT have their own SHARC DSP; they rely on the main board's SHARC DSP to do all digital processing.

Motion Block

Each main and expansion board has two (2) "motion blocks." Each motion block, in turn, contains four (4) "motor blocks," each of which controls one motor (see "Motor Block" below). Because motion blocks contain multiple motor blocks and are linked to all other motion blocks on the controller, it is possible to freely map one object to another. This provides the XMP controller with unprecedented flexibility and speed in coordinated motion.

A motion block is not a single piece of hardware, but actually consists of one field programmable gate array (FPGA), plus other hardware components. It is useful to think in terms of motion blocks to best understand how the XMP controller works. The motion block serves as the interface between the SHARC processor subsystem and motion I/O signals, and handles the interface to output peripherals (DACs, XCVRs, and opto-I/O), as well as the interface to feedback peripherals. Each motion block subsystem has a high speed serial interface to the SHARC DSP subsystem.

Motor Block

Contained within each motion block are four (4) "motor blocks." Motor blocks represent the most basic unit of motion control within the XMP environment. Each motor block acts as a dedicated gateway between the host computer and motion control components such as motors and encoders.

List of Figures

The next three pages illustrate overall data flow and component architecture of a typical XMP controller, including:

- Figure 6-5. Overall architecture: main and expansion boards.
- Figure 6-6. Motion block detail.
- Figure 6-7. Motor block detail.

A study of these figures will provide some insight into an XMP controller's component layout and data flow.

Figure 6-5. Overall architecture: main and expansion boards.

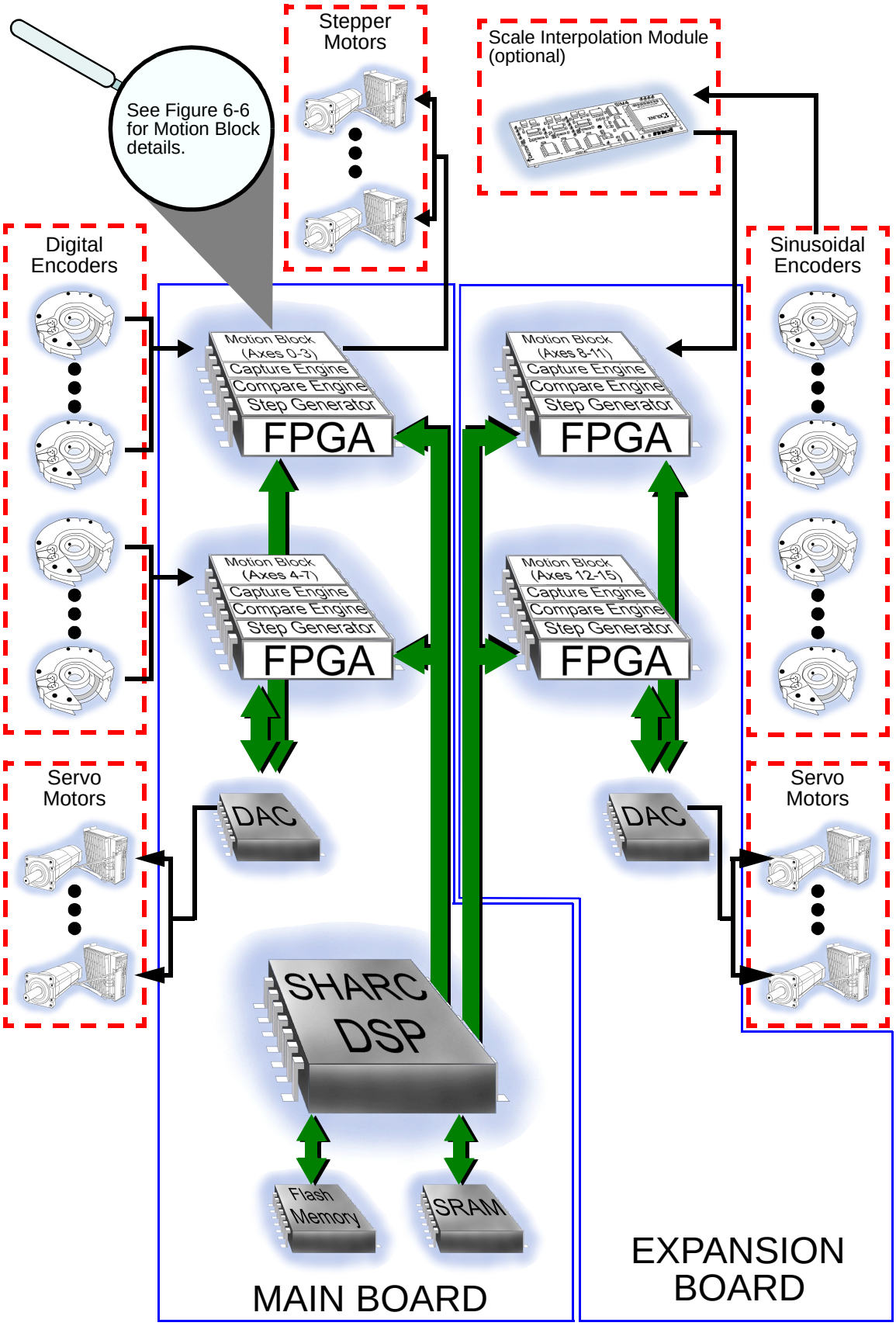


Figure 6-6. Motion block detail.

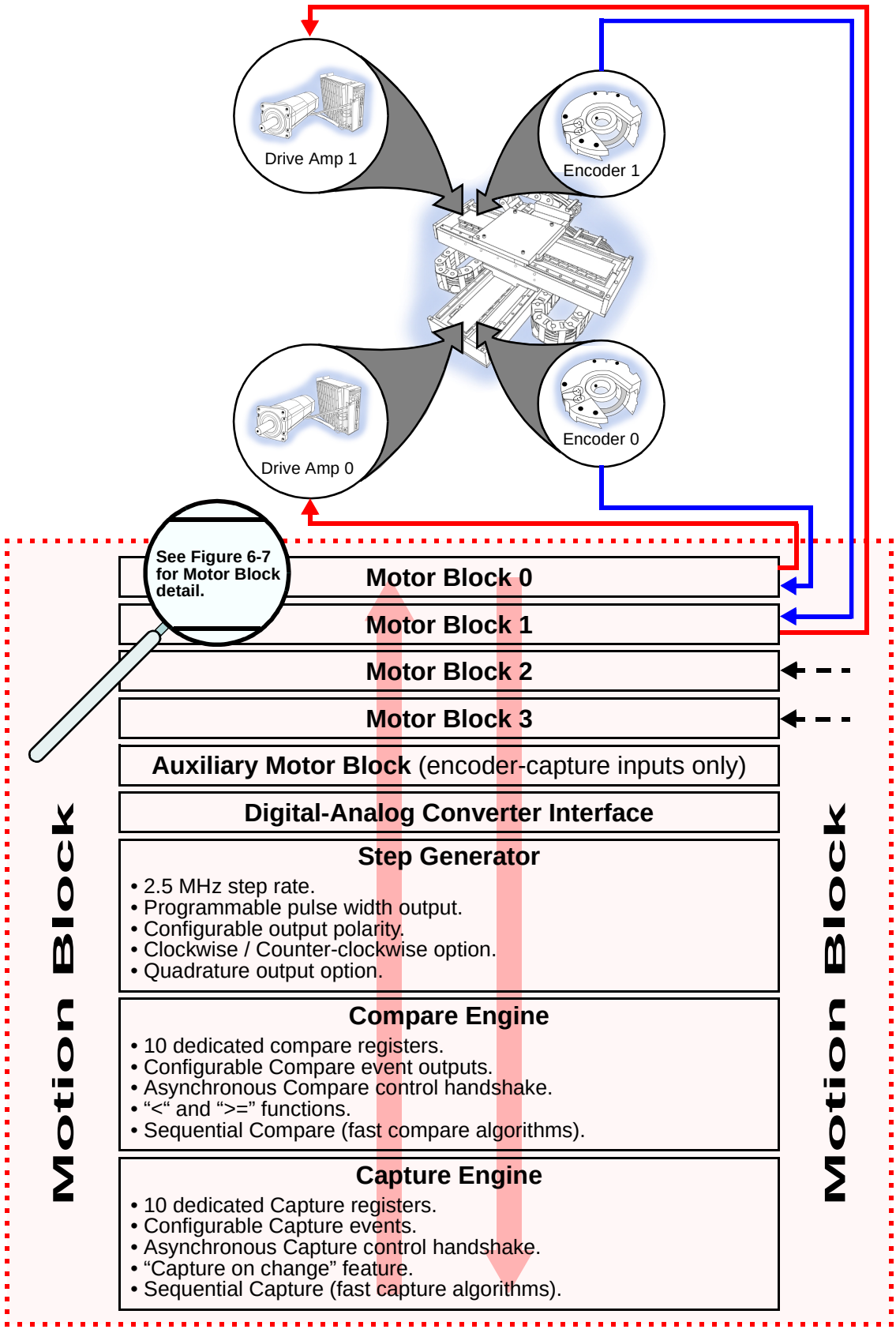
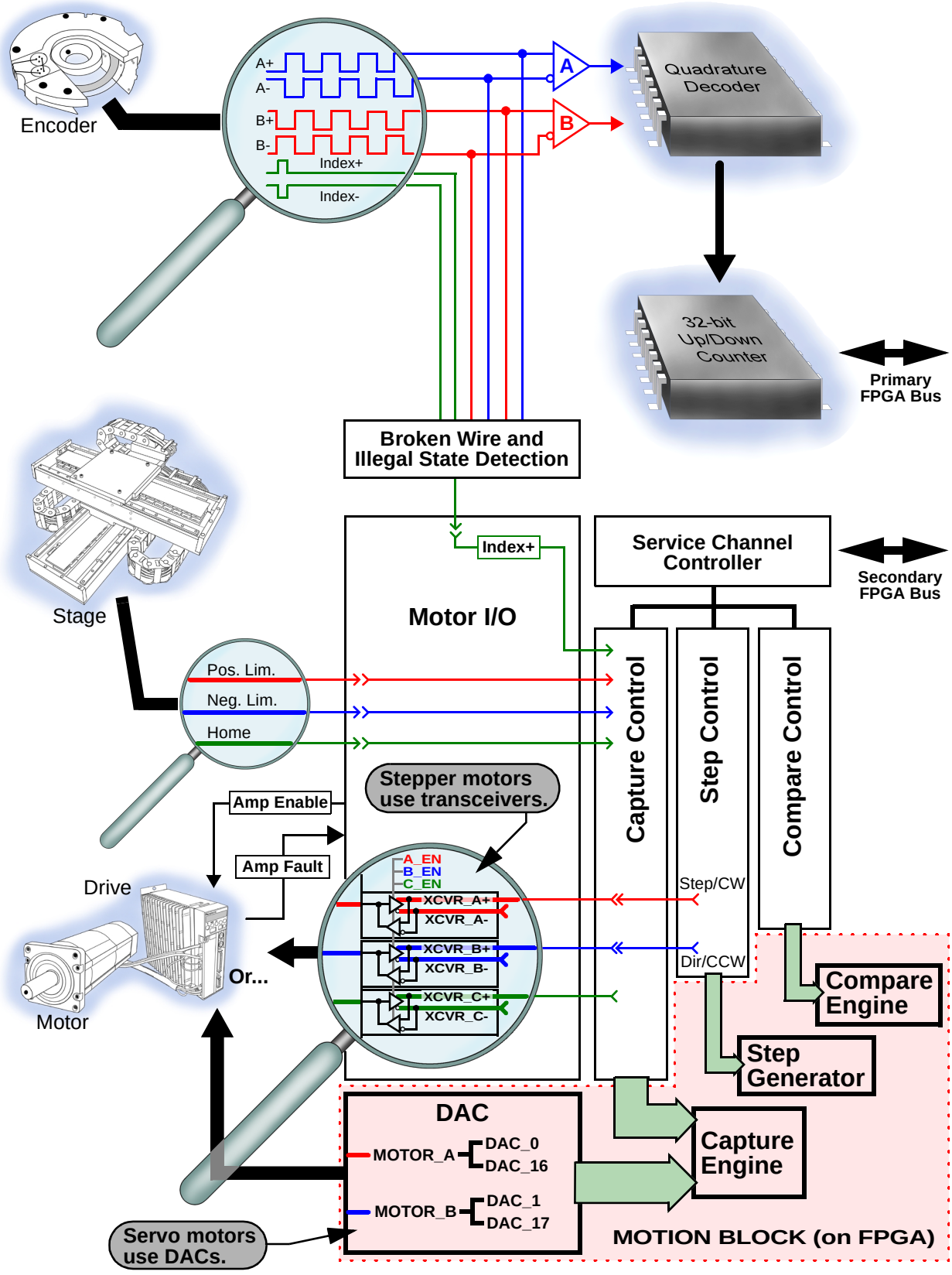


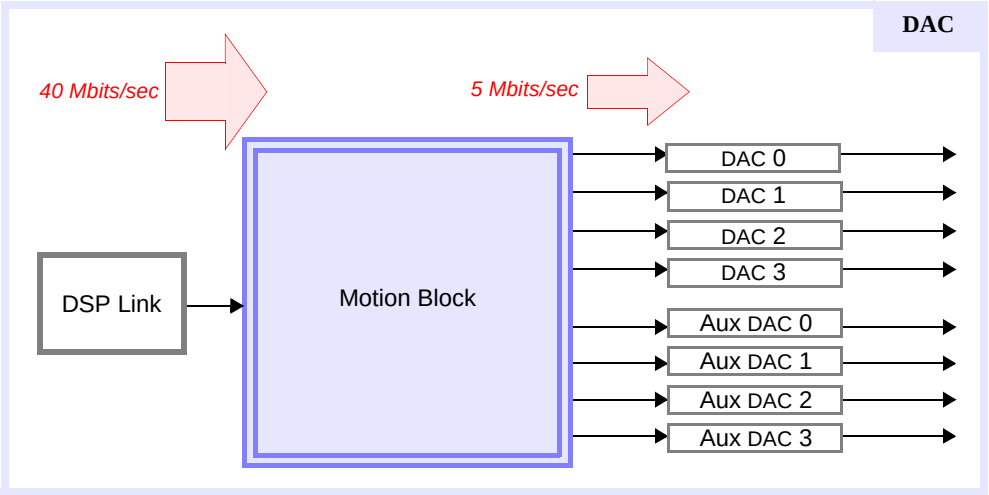
Figure 6-7. Motor block detail.



DAC Circuitry

The D/A converters are 16-bit and DC-specified for high performance.

Figure 6-8. DAC circuitry.



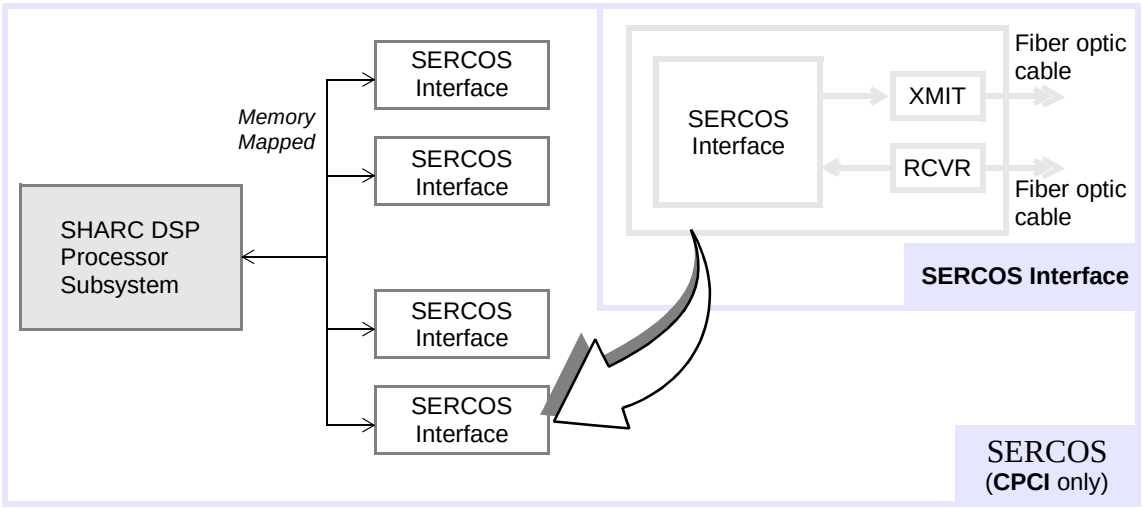
Four pairs of DACs (eight total) are associated with each motion block. The first DAC of the pair is the *Servo Command Output DAC*, the second is an *Auxiliary DAC* for optional two-phase sinusoidal commutation support. The serial stream is shifted out of the first DAC of each pair and into the second DAC. A common latch updates all DACs.

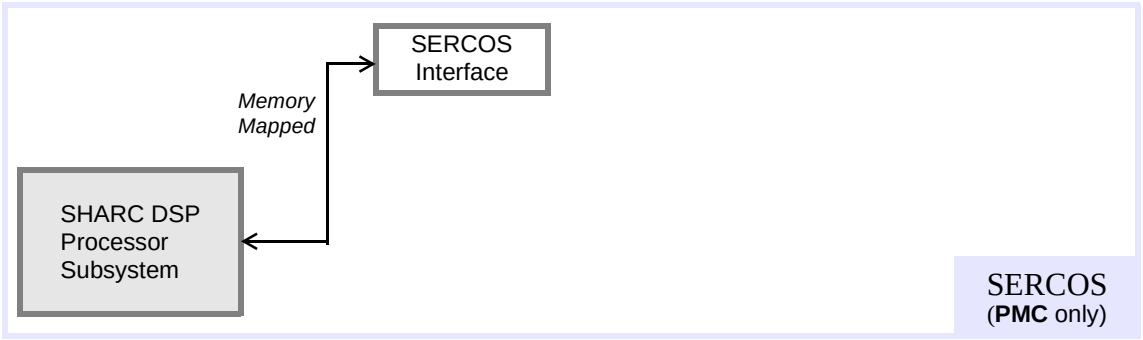
The outputs swing ±10V to provide the *Command Output* voltage. The reference signal for the *Command Output* is each individual DAC's analog ground. Each *Command Output* is capable of driving a load of 10kΩ and 220pF (typical brushless amp input over 6 feet of cable) with a slew rate of 4 V/microsec (20 Volt swing at 20 kHz). See Chapter 3 for specifications on command voltage outputs.

Note	The AD7849BR DAC used on the XMP should not need calibration. The bipolar zero error is specified at ±2 LSBs (0.6 mV) at 25° C. Between -40° C and +85° C, bipolar zero error is ±12 LSBs (3.6 mV). Calibration should not be required. The outputs are fault protected (switched to AGND) for power-on/power-off and emergency conditions.

SERCOS Subsystems

The SERCOS subsystem is an interface between the SHARC processor subsystem and a drive. A drive is generally referred to as a Node. A **Node** is a SERCOS slave representing a drive, I/O block, etc. A SERCOS subsystem can support up to eight nodes. An XMP-SERCOS controller can support up to four SERCOS subsystems.





The SERCOS subsystem consists of a SERCOS interface and a set of fiber optic transceivers. The fiber optic transceivers allow communication with the SERCOS Nodes via a serial data stream. The SERCOS subsystem supports data streams with rates of 2, 4, or 10 MBaud.

The SERCOS interface is responsible for synchronizing data transmission by transmitting a Master Synchronization Telegram (MST) at a defined interval (T_{scyc}). T_{scyc} can be programmed to be a power of 2 of the XMP sample rate.

Each SERCOS subsystem is mapped to a specific location in the SHARC memory space. This allows the SHARC processor to directly access registers and memory in the SERCOS subsystem. This, in turn, allows direct mapping of command and feedback data for each Node. The analog and SERCOS versions of the XMP use the same firmware.

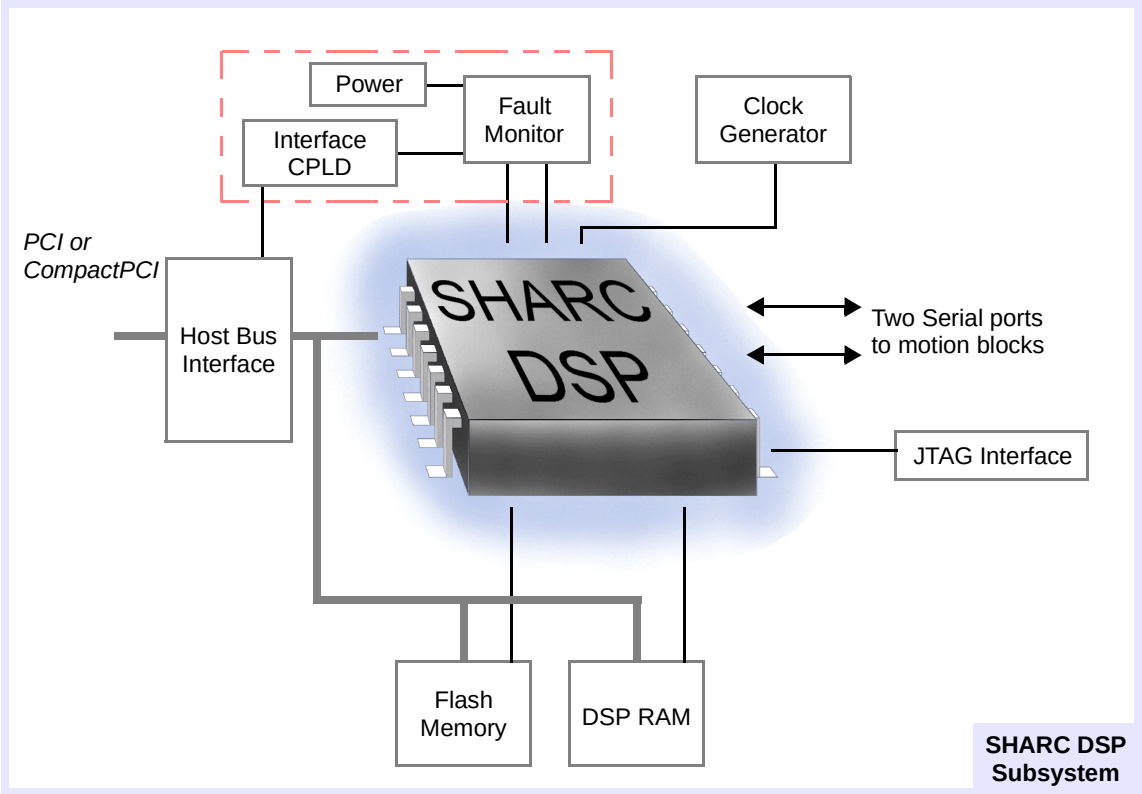
For more information on SERCOS hardware, please refer to Chapter 5.

SHARC DSP Processor Subsystem

The DSP processor subsystem includes the following subsystems:

- SHARC DSP processor
- DSP RAM
- Flash memory
- Host bus interface
- Serial ports to motion blocks
- Fault monitor
- Clock generator
- JTAG interface

Figure 6-9. SHARC DSP processor subsystem.



SHARC DSP Processor

The XMP's main DSP processor is a 32-bit Analog Devices 21061 Floating Point DSP (also known as the SHARC DSP) with a peak performance of 150 MFLOPs. The 21061 provides trajectory planning, coordination, interpolation and position/ velocity loop closure.

DSP RAM

External SRAM is available to the DSP for maintaining message buffers and storing data. The SRAM can be up to 2 Mbytes of zero-wait state RAM. The standard configuration is 64K x 32 bit words (256K bytes).

Flash Memory

An 8-Mbit FLASH memory stores the DSP boot code, FPGA configuration code and nonvolatile system information. The FLASH memory is rated for 100,000 write/erase cycles.

Serial Ports to Motion Blocks

The SHARC DSP communicates with the four peripheral *motion blocks* over two 40 Mbit/sec serial links. All DSP-to-motion block data transfers occur over these two links.

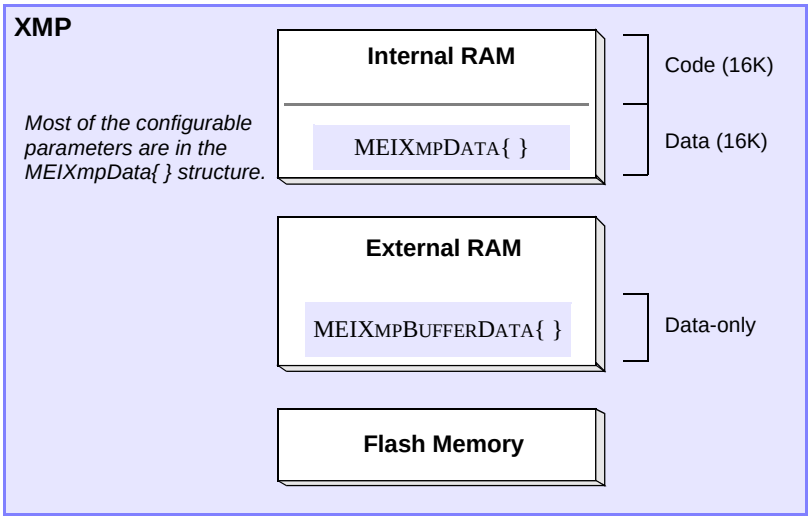
Each SHARC DSP serial port connects to two motion blocks. The motion blocks are the gateways for all of the controller's I/O to the external system.

Memory Organization

The XMP has three memory regions: internal RAM (SRAM), external RAM (also SRAM), and flash memory. Internal RAM is split into two roughly equal sections of about 16K words (32-bit), one for code and the other for data. Another data-only region exists in external RAM. The MEIXmpData structure (defined in *xmp.h*) just about fills the internal RAM region. The MEIXmpBuffer-Data structure is in the external memory region. (Almost all of the configurable parameters are in the MEIXmpData structure.)

Figure 6-10. XMP memory organization.

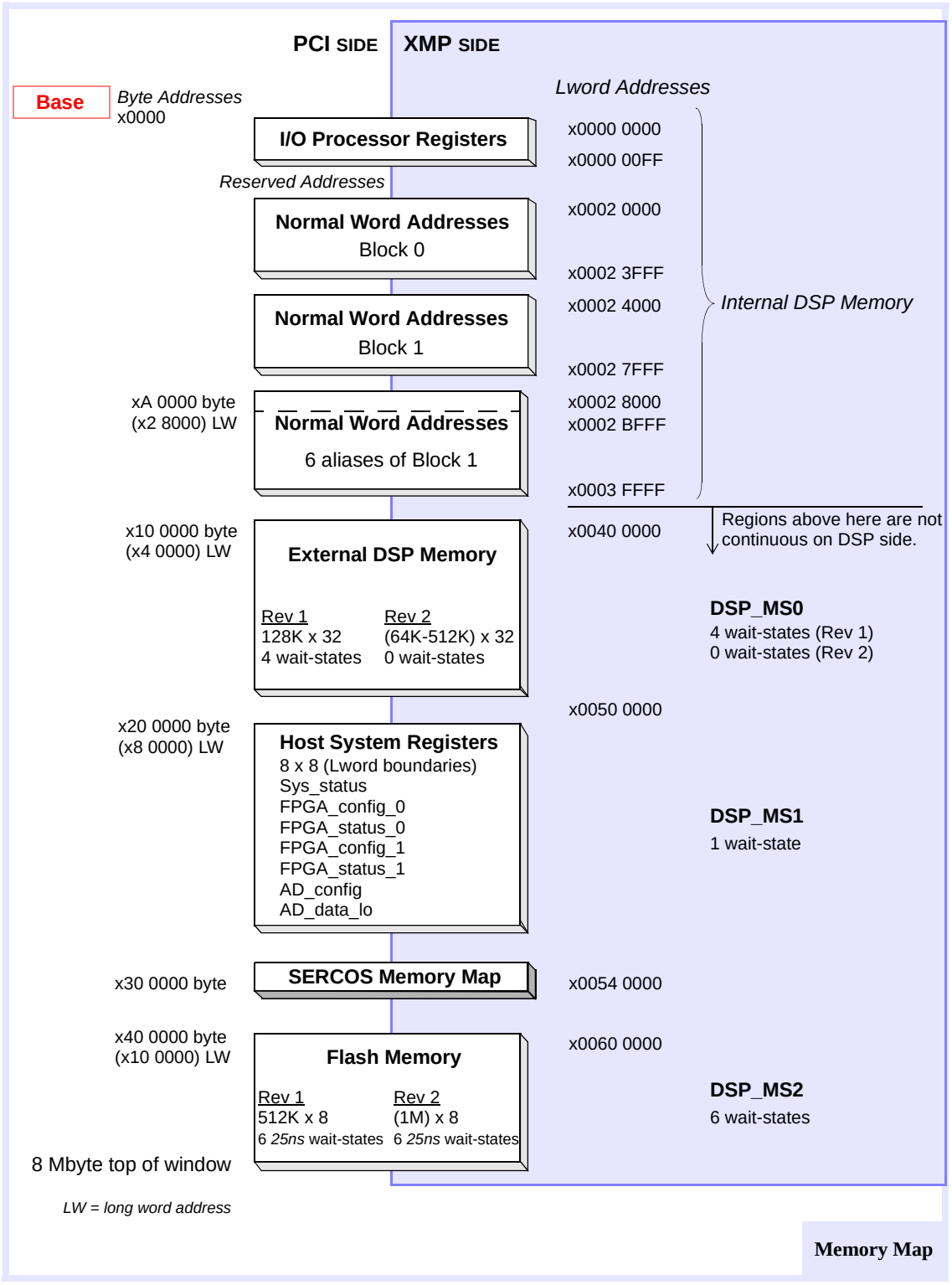
XMP Memory Organization



Immediately after power-on or reset, the SHARC loads a small boot loader (~128 words) from the bottom of flash memory into internal RAM, and then branches to the first instruction. This boot loader then loads the rest of the firm-ware code (~16K words) from the same flash memory page and then jumps to the first firmware instruction. The firmware then copies data into the MEIXmp-Data (internal RAM) and MEIXmpBufferData (external RAM) structures from identical copies stored in flash memory.

Memory Map

Figure 6-11. Memory map of XMP-PCI and XMP-CPCI.



Corrupted Flash Memory?

It is **very difficult** (but not impossible) to corrupt the flash memory, because there are software and hardware locks which make this improbable. If you suspect that the flash memory is being corrupted, make sure that all of your software developers know who is writing updated values to flash memory and when they are doing that. Often, one member of your software developer team is writing updated values to flash, and forgetting to tell the other team members about it.

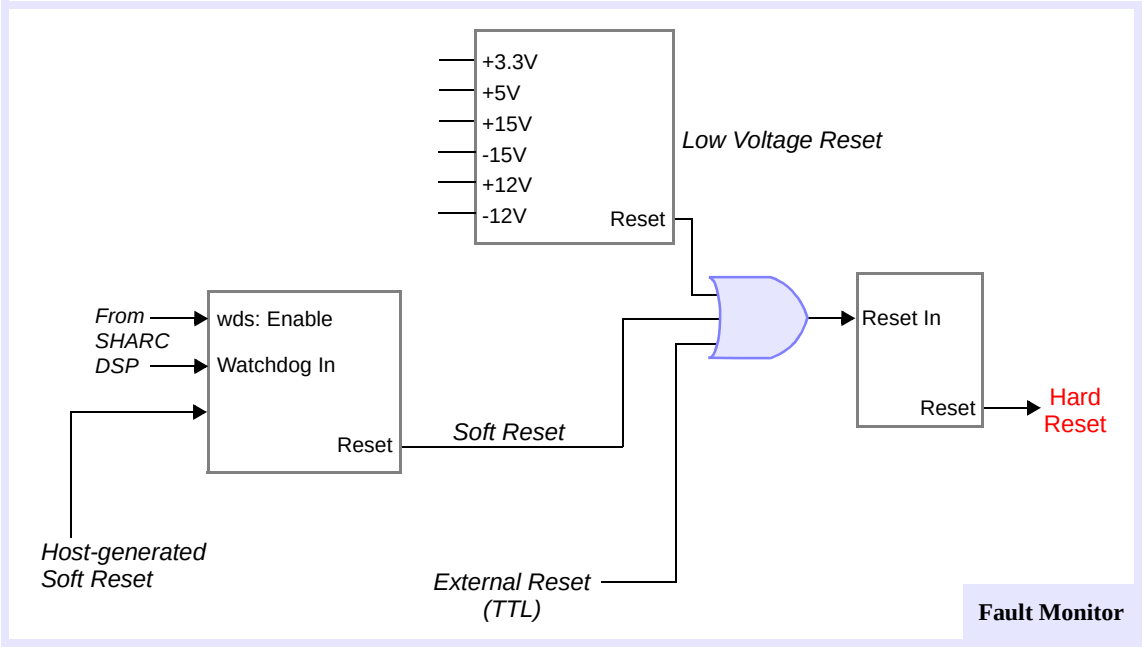
Faults and Resets

The fault monitor circuitry detects faults and controls reset operations.

Resets

XMP system resets can occur due to *hard reset* (hardware reset) or *soft reset* (software reset). The DAC (digital-to-analog converter) outputs are disabled after both *Hard Reset* and *Soft Reset* events.

Figure 6-12. Fault monitor reset scheme.



Hard Reset

Hard resets are caused by:

- Power-up
- Low voltage conditions on either 5V, +15V or -15V, +12V or -12V, 3.3V
- Software reset via the host (through Interface CPLD), or watchdog time-out
- External reset line (TTL) activation (one dedicated input per board)

Hard reset occurs 400ms after power-on or fail events, and 200ms after *software reset* or *external reset* events. The *Hard Reset* lasts 200-400 msec, leaving the DACs with *Vout* switched away from the DAC and tied to the DAC's analog ground. The first write from the SHARC to the DAC will cause this switch (*Vout*) to shift back to the DAC for normal operations.

Soft Reset

Soft resets are caused by a *watchdog timeout* from the DSP's flag output, or when the host writes a key sequence to the host system registers. *External reset* is triggered using an external TTL input. Both software and external resets are active low. To guarantee reset when Vcc is low, the triple OR gate will operate down to Vcc = +1V.

The software reset is handled by a *Watchdog Timer* with an adjustable timeout. The *Watchdog Timer* is intended to catch malfunctioning firmware. The *Watchdog Timer* does not monitor Vcc. The timer only responds to a timeout accompanied by a 200 ms reset. The *Watchdog Timer* is disabled during power-up, and following a reset of the SHARC DSP, allows the DSP to boot and begin instruction execution.

A *Soft Reset will*:

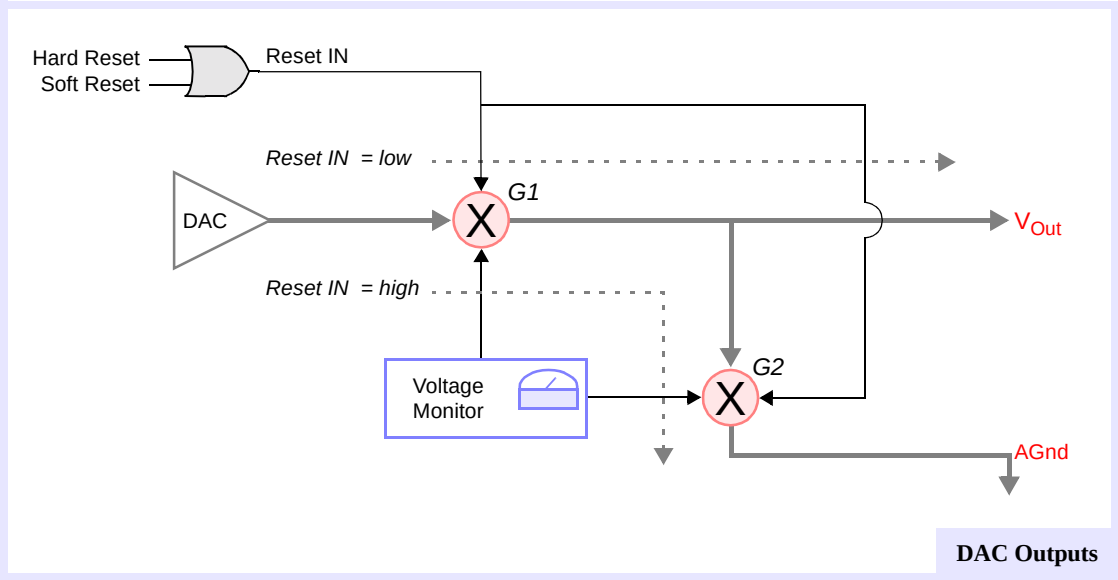
- Cause all analog command outputs to switch away from the DAC to a low impedance path (to their associated analog ground).
- Soft reset the motion blocks, causing them to clear dedicated I/O to their DSP-configured default state, and to reset the position counters and associated registers. *Soft Reset* will not enable the FPGA configuration to be reloaded. Note that it takes a power-on *Hard Reset* to cause the FPGAs to reload configurations.
- Clear the host interface *Reset Status* register and all other host system registers.
- *Not* reset the SHARC DSP. The host driver will soft-boot the DSP.

For the *Soft Reset* watchdog circuitry, a latched reset is used, which provides an added protection of not allowing a malfunctioning SHARC DSP to mistakenly write to the DAC (the *Soft Reset* is held *until* the host clears it, as part of the host's watchdog timeout recovery routine).

Power Fail Control of DAC Outputs

The DAC has embedded analog switches in the output stage. The analog switches G1 and G2 are controlled by the on-chip voltage monitor and by the dedicated reset input pin (*Reset IN*). A low voltage on the *Reset IN* input will cause G1 to open and G2 to close, clamping Vout to AGND via a low impedance path (typically 1K ohms). This condition will remain until the reset input goes high and a valid word is written into the DAC.

Figure 6-13. DAC output protection.



The *Reset IN* input is the active low OR'ed condition of *Hard Reset* or *Soft Reset*.

A 65K ohm resistor from *Vout* to AGND will maintain 0 volts on *Vout* until the supplies reach +1 volt. What happens if we lose an analog power rail? The *Reset IN* input will go low once one of our rails crosses the reset threshold. The *Reset IN* input is valid for *Vdd* and *Vss* to +1.2 volts, and below those voltage levels, the 65K ohm pulldown resistor protects the circuit.

XMP Data Architecture

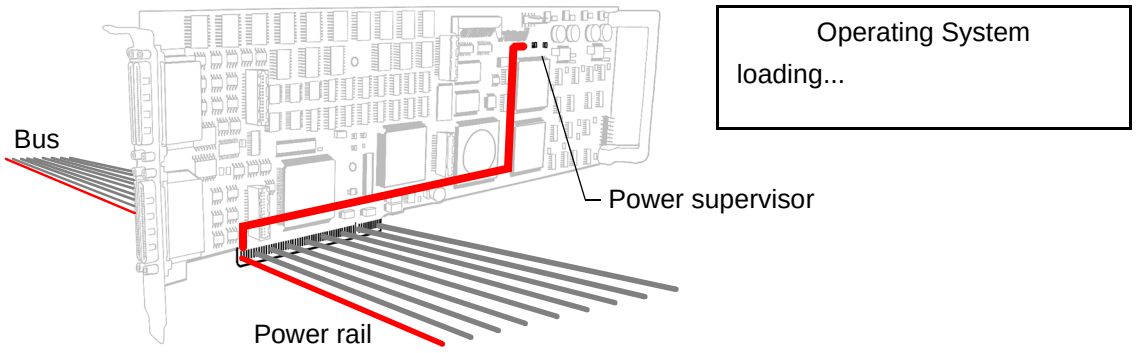
This section provides a dynamic look at the XMP's data architecture, between board components, and between the controller and external devices such as drive amplifiers.

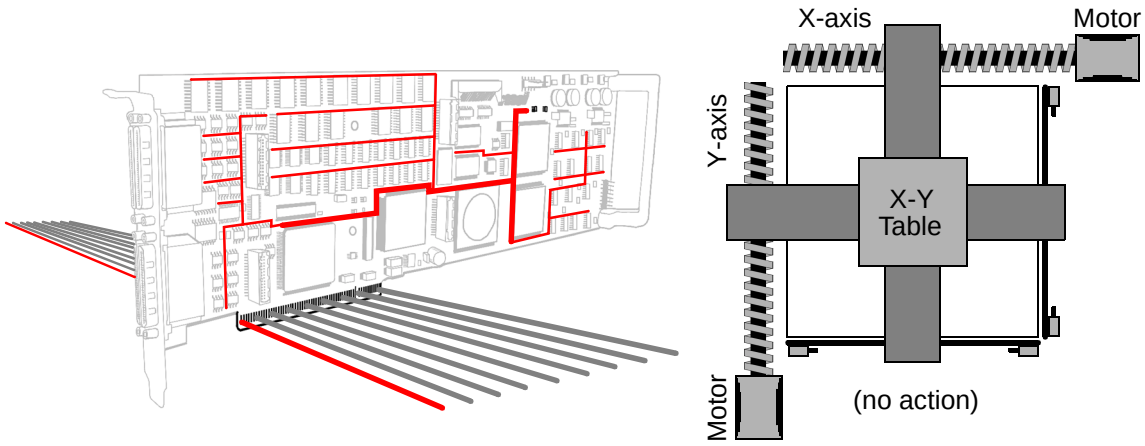
XMP Power-up Sequence

When first powering-up the XMP controller, several actions initialize the controller according to the firmware's settings and prepare it for motion control.

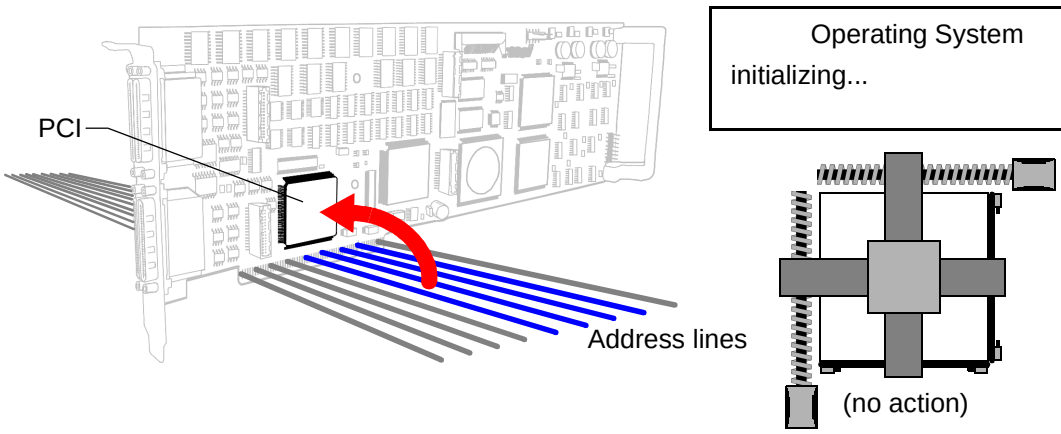
Power is applied

During power-up, the host computer initializes. Power is also supplied via bus lines to the XMP controller and other devices connected to the PCI bus. The XMP controller's power supervisor waits for the bussed power supply to attain voltage thresholds (2.93 for 3.3 VDC power, and 4.63 for 5 VDC power) before powering the controller's components. Until power fully stabilizes, the board is kept in a reset condition.



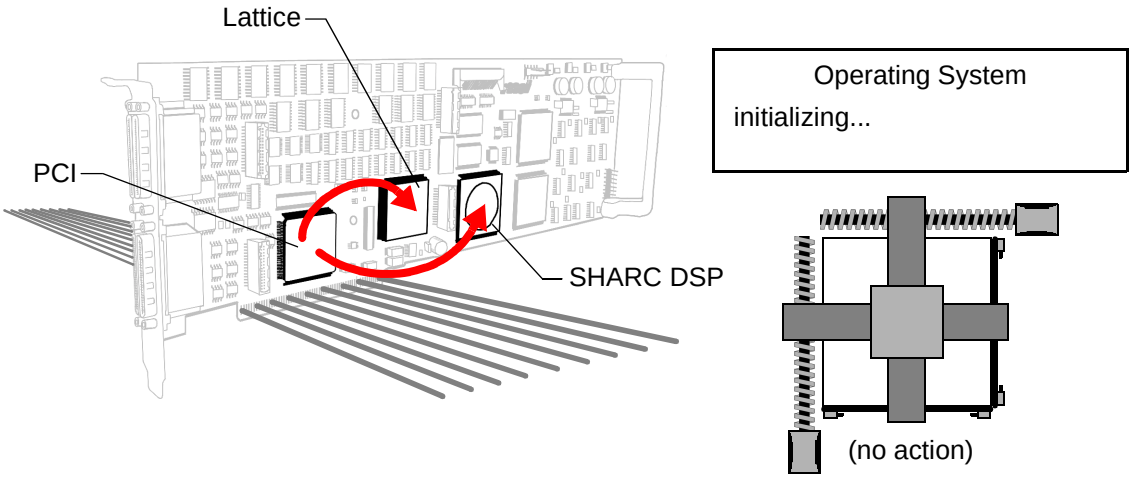


Once initialized, the CPU bios polls the bus and assigns an address to each device (including the XMP controller).



PCI handshakes with SHARC DSP and Lattice

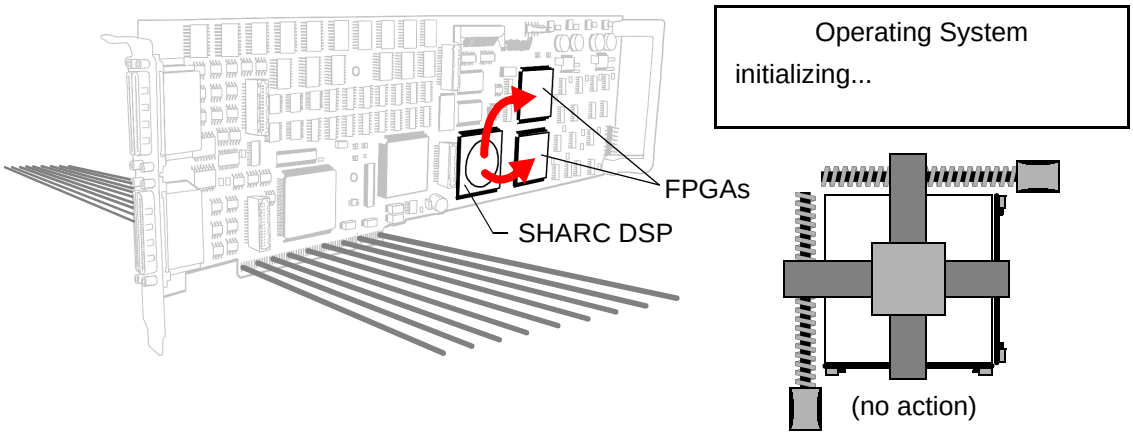
When the XMP controller's PCI receives a bus address from the host computer's CPU bios, it handshakes with the SHARC and Lattice.



FPGAs loaded

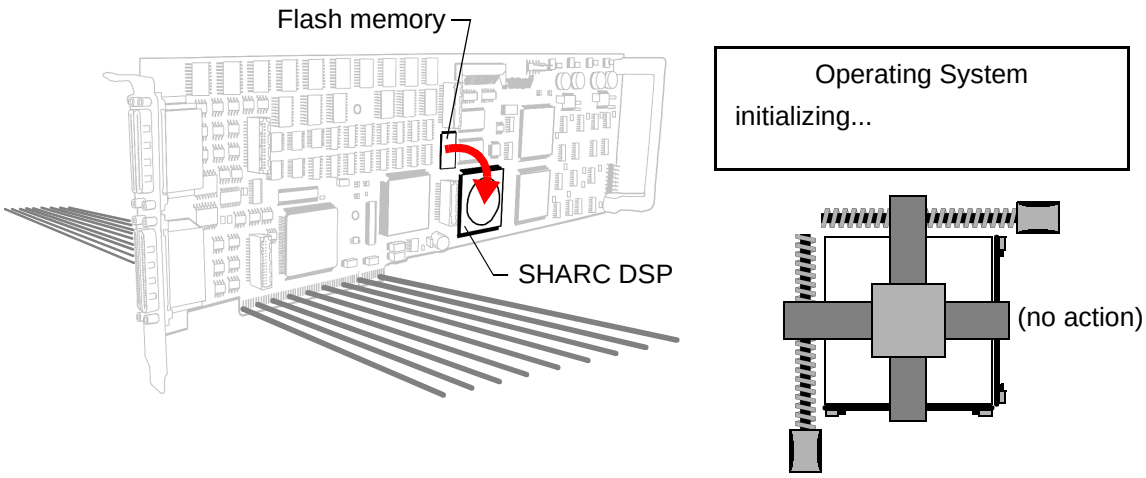
With the FPGA's file format determined, the SHARC loads both FPGAs with the following:

- DSP variables (including default timing parameters)
- FPGA image
- SIM4 FPGA image (if a SIM4 module is installed)



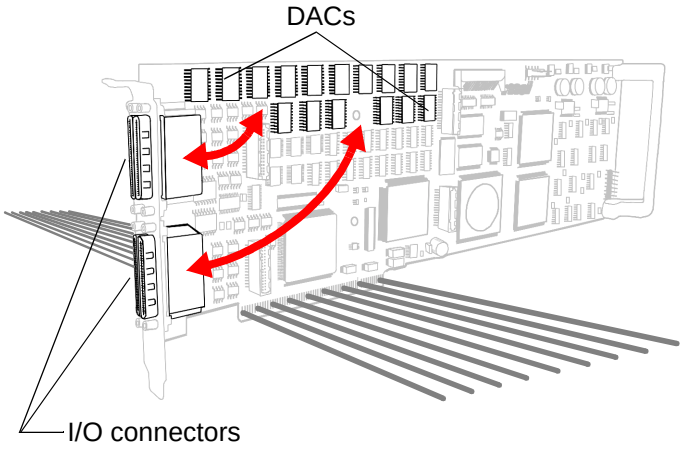
Firmware loaded from flash memory into the DSP

With the FPGAs loaded, the controller is now ready to load the firmware from flash memory to the DSP.

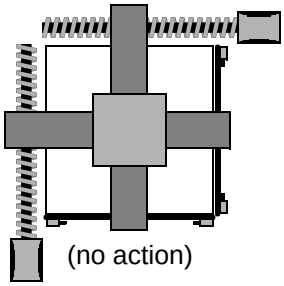


DACs calibrated and activated

At the beginning of the power-up process, the digital-to-analog converters (DACs) are kept grounded (i.e., in an inactive, OFF state) until the controller is ready to switch them in. This avoids erroneous inputs-outputs with the drive. After the DSP is loaded, the DACs are internally calibrated and activated. The system may now accept input from the drives and send commands.



Operating System
initializing...



(no action)

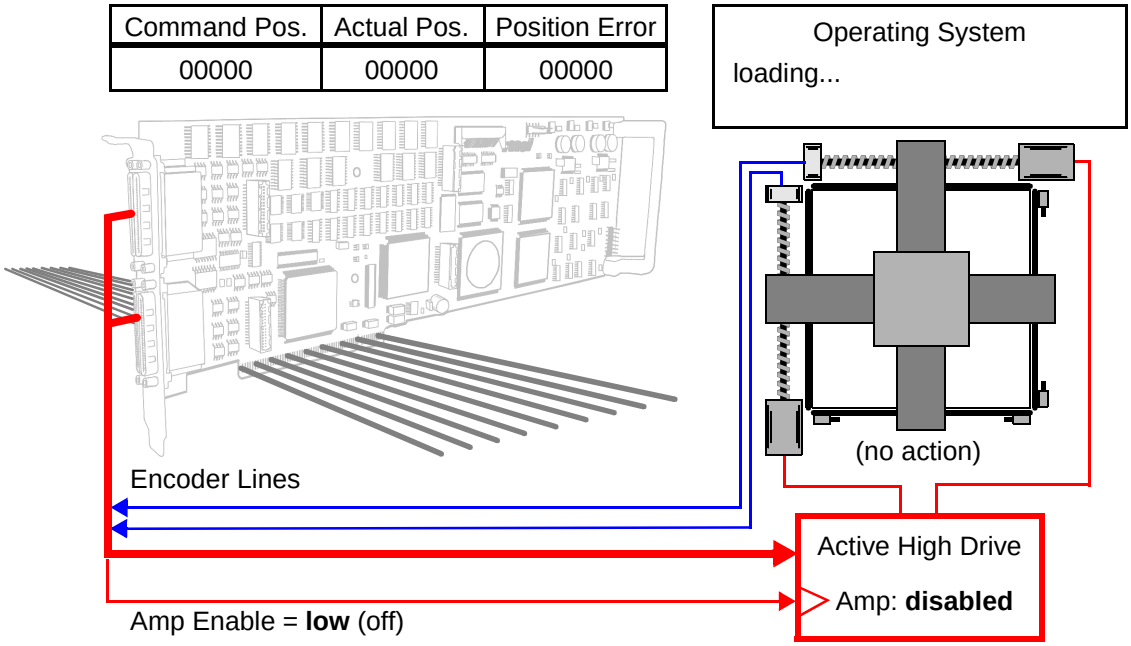
Controller initializes

The controller's boot configuration is based upon parameters stored in flash memory; these determine the controller's (and drive's) start-up behavior. One of the programmer's first tasks is to define what this "hello world" starting state ought to be immediately after power-up. In some instances, it may prove advantageous to keep a default boot configuration in flash memory, then modify it from the application during boot-up. In other instances, it may be better for the application to flash the controller's memory with an entirely new set of parameters during boot-up. The user must determine the best approach.

Controller configuration can be done using Motion Console; however, during development of an application, it is the programmer's responsibility to write code that properly configures the controller for its next session upon exit of the application. The controller should utilize safe amp enabling and hardware I/O to prevent runaway conditions during start-up.

For this example, we will assume the following safety measures:

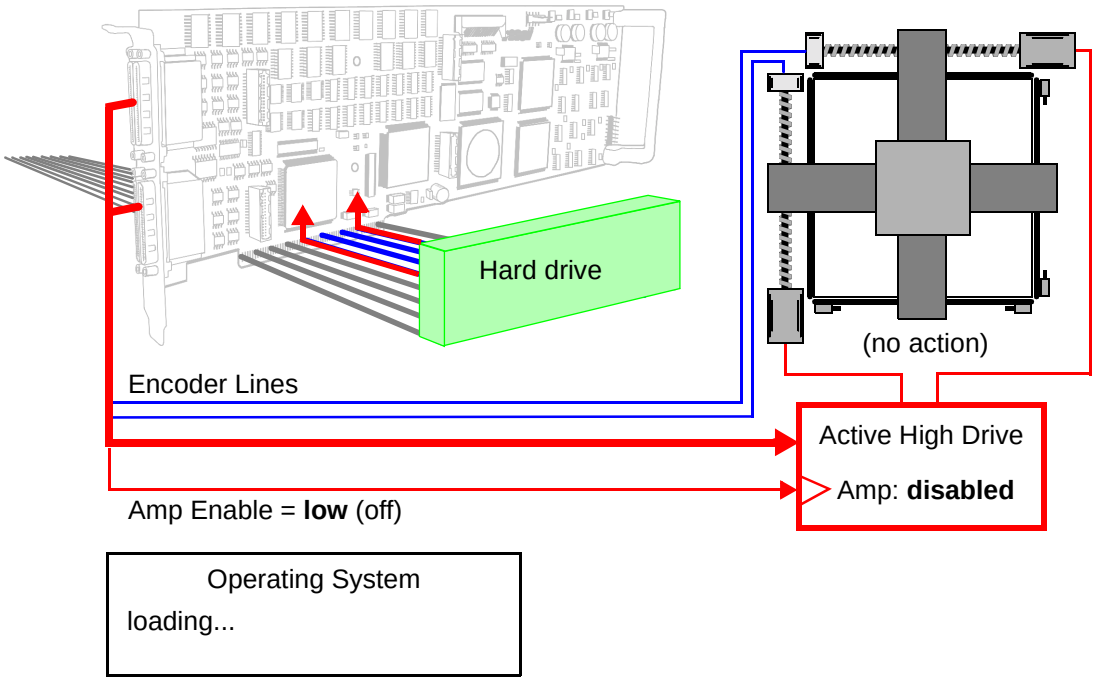
- The drive is configured for active high amp enabling. (That is, the drive cannot be enabled without controller input.)
- The controller is configured to awaken with the Amp Enable line set to **low** (off). This should keep an active high drive disabled.
- The controller is configured to trigger an Abort if the Position Error is greater than some specified value. Because the controller's data registers are all set to zero at start-up (including the Command and Actual Position values), any mechanical movement will produce a Position Error. By triggering an Abort based on Position Error, the controller safeguards against uncommanded moves.[1]



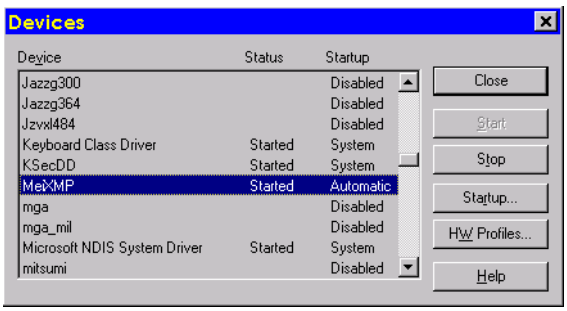
1. **IMPORTANT!** Use of Position Error and other positional inputs as safeguards assumes: a) good data connection between the encoder(s) and controller; b) normal conditions. If a higher level of safety is required, then extra measures should be utilized, such as broken encoder wire detection.

Device Driver and Operating System Loads

At this point, the XMP controller is initialized and ready for use. The next and final phase of the power-up sequence is the loading of the XMP's device driver and operating system on the host computer. This must be completed before any motion control application can be loaded and run.



The XMP controller's device driver is shipped to load manually; however, it can be reconfigured to load automatically. To view the XMP's device driver, access your operating system's **Devices** window. (In WindowsNT™ environments, the XMP controller's device driver is labeled as **MeiXMP**.)

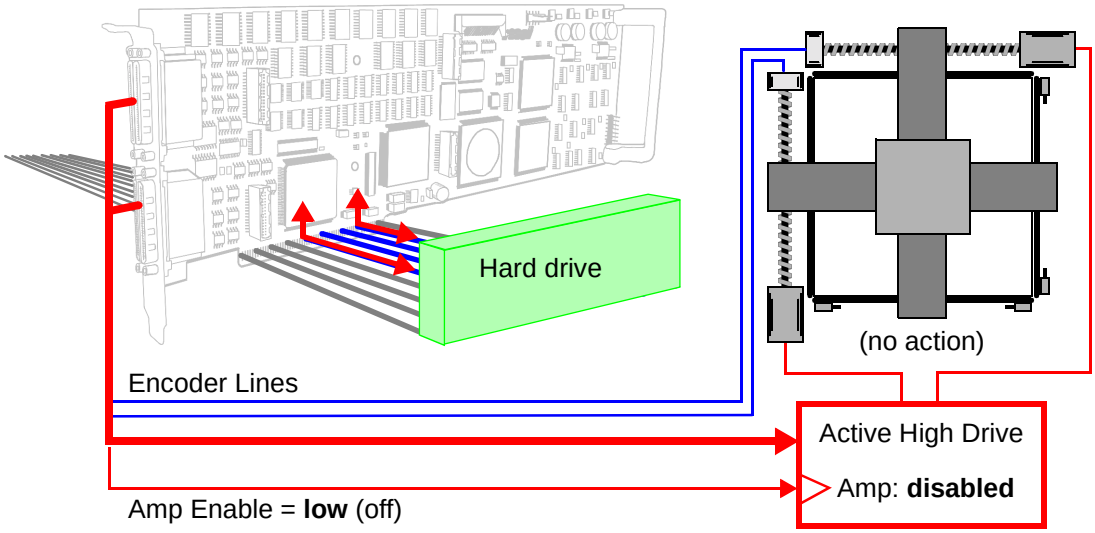


The operating system completes its loading within several seconds of the XMP's initialization.

Load Motion Control Application

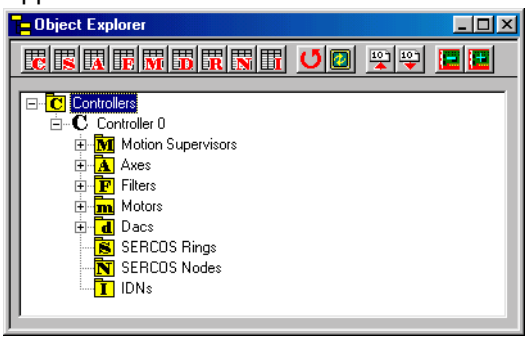
Application boot-up

When a motion control application (e.g., Motion Console) is loaded, it first verifies that the XMP controller is operational. The user interface is displayed and is now available for use.



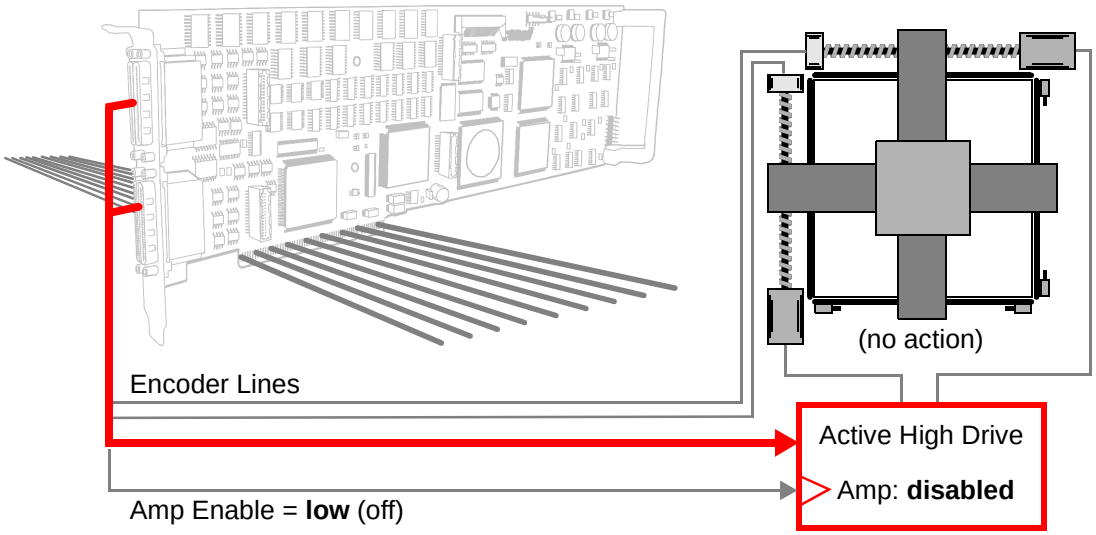
Operating System
C:\MC_XMP_95.exe

Application UI



Create Control Object and Initialize Controller

Before any motion application can move axes, it must create a controller object and initialize communication. This is a standard requirement for any application which intends movement.



MPI

```
#include <stdlib.h>
#include <stdio.h>
    .
    .
    .

/ *Create motion controller object */
control = mpiControlCreate(controlType,
                                    &controlAddress);
meiASSERT(mpiControlValidate(control) == MPIMessageOK);

/ *Initialize motion controller */
returnValue = mpiControlInit(control);
if (returnValue != MPIMessageOK)  {
```

Homing the Motion Control System

Homing is the setting of a coordinate system through detection of a known point in space (the “home” point). A home point can be defined using a limit switch, or by detecting a signal from the encoder such as a home or index pulse. This is typically the first act a controller must perform before running the main application code.

Before homing, objects must be created and the controller must be initialized (see above). One example of a homing sequence is shown below, consisting of:

- Configure a capture to trigger off the home input.
- Configure a home event action.
- Arm the capture.
- Command a velocity move while polling the capture trigger.
- When the trigger changes (at the home point), set the origin.
- Move the axis to the origin.

This homing example is taken from the **home1.c** sample application program, which may be found in the *Sample Applications* manual and should be studied. For sake of clarity, excerpts are included below.

Configure the Capture trigger

After the motion objects are created and the controller is initialized, the MPI must configure and arm a capture. The trigger in this example is a home sensor (limit switch) on a stage.

MPI

```
/* Configure capture for a Home input trigger */
return Value =
                    mpiCaptureConfigGet(capture,
                                                &captureConfig,
                                                 NULL);
msgCHECK(returnValue);

/* Set capture parameters for trigger on Home input */

   captureConfig.trigger.mask = MEIMotorInputHOME;
    captureConfig.trigger.pattern = captureActiveEdge ? MEIMotorInputINDEX : 0;


returnValue =
     mpiCaptureConfigSet(capture,
                                  &captureConfig,
                                 NULL);
msgCHECK(returnValue);
```

Define motion parameters

In preparation for commanding moves, there are several motion parameters which must be defined:.

MPI

```
/* Set-up motion parameters  */
trajectory.velocity          = VELOCITY
trajectory.acceleration      = ACCEL
trajectory.deceleration      = ACCEL;  /* Not used in velocity type move */
trajectory.jerkPercent       = 0.0;    /* Not used in velocity type move */

params.velocity.trajectory = &trajectory;
```

Two types of moves are used for this sample homing routine: a velocity move and a trapezoidal move. The velocity move consists simply of running the axis at a set velocity, hunting for the home point, without regard to its starting and ending positions.

Because a trapezoidal move requires beginning and ending positions (and we don't know the axis coordinates until the home point is detected), the first move *must* be a velocity move. After coordinates are established through detection of a home point, a trapezoidal move may be used to move the axis to the origin.

Configure the home event action

The "home event" is the what we use to detect the home point. The home event could be an input from a limit switch, triggered when the axis runs into it. The home event could be a home or index pulse on the encoder. For this example, the stage's **HOME** switch is used as the trigger, and the trigger's input is configured to be **TRUE** (active edge).

The "home action" is what happens when the home is detected. In this example, the programmer has decided to define the action as a command to **STOP** the axis.

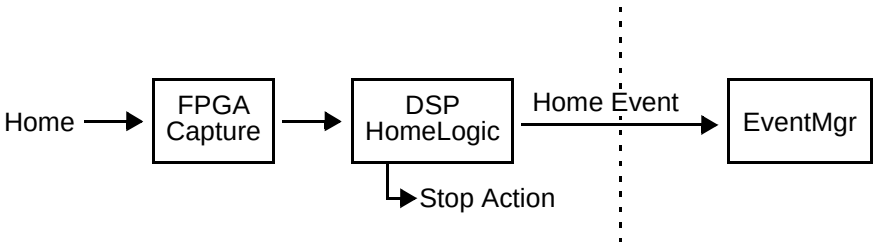
MPI

```
/* Configure Home event action  */
returnValue =
        mpiMotorEventConfigGet(motor,
                               MPIEventTypeHOME,
                                 &eventConfig,
                                 NULL);
msgCHECK(returnValue);

eventConfig.action = MPIActionSTOP;        /* STOP event */
eventConfig.trigger.polarity = TRUE;       /* Active HIGH */

returnValue =
        mpiMotorEventConfigSet(motor,
                               MPIEventTypeHOME,
                                 &eventConfig,
                                 NULL);
msgCHECK(returnValue);
```

This uses a **HOME** input from the encoder as the home event.

The home action is: **STOP**

The trigger's input polarity is **TRUE** (active edge).



Arm the trigger

With the home event and action defined, all that is left is to “arm” the trigger by setting the mpiCaptureArm to TRUE. Recall that earlier, this was set to FALSE to “disarm” the system.

MPI

```
/* Arm the capture  */
returnValue =
        mpiCaptureArm(capture,
                      TRUE);
msgCHECK(returnValue);
```

The capture trigger is armed by setting mpiCaptureArm to **TRUE**.

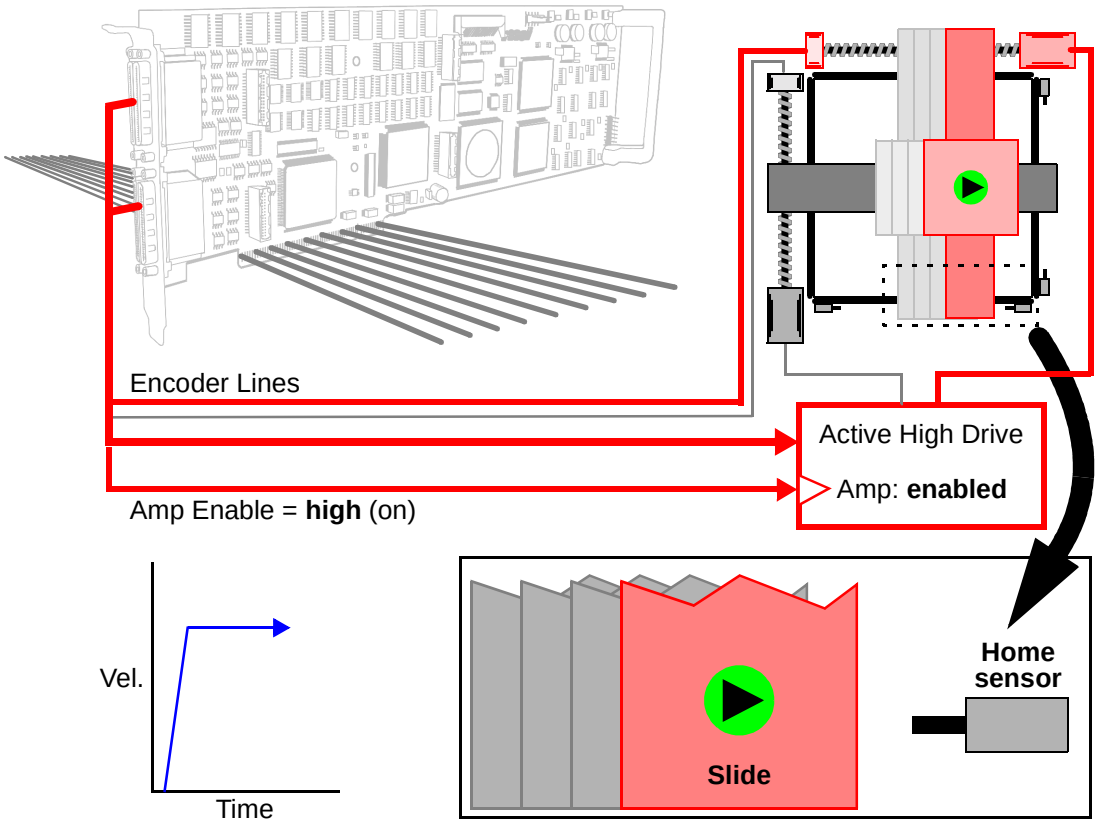
Start a velocity move

With the capture event and action defined, and the trigger armed, we are ready to hunt for the home point. This is accomplished by commanding a velocity move. Before the axis can be commanded, however, the XMP controller must first enable the drive. In this example, it is assumed that the drive is wired to be an active high type; therefore, the Amp Enable line is set to **high** (on). Drive enabling via software must be done as a separate code item and is not shown here:

MPI

```
/* Start a velocity move  */
returnValue =
        mpiMotionStart(motion,
                       MPIMotionTypeVELOCITY);
                       &params);
msgCHECK(returnValue);
```

This is a **VELOCITY** move.



As the leadscrew rotates, the slide approaches the home sensor.

Poll trigger status

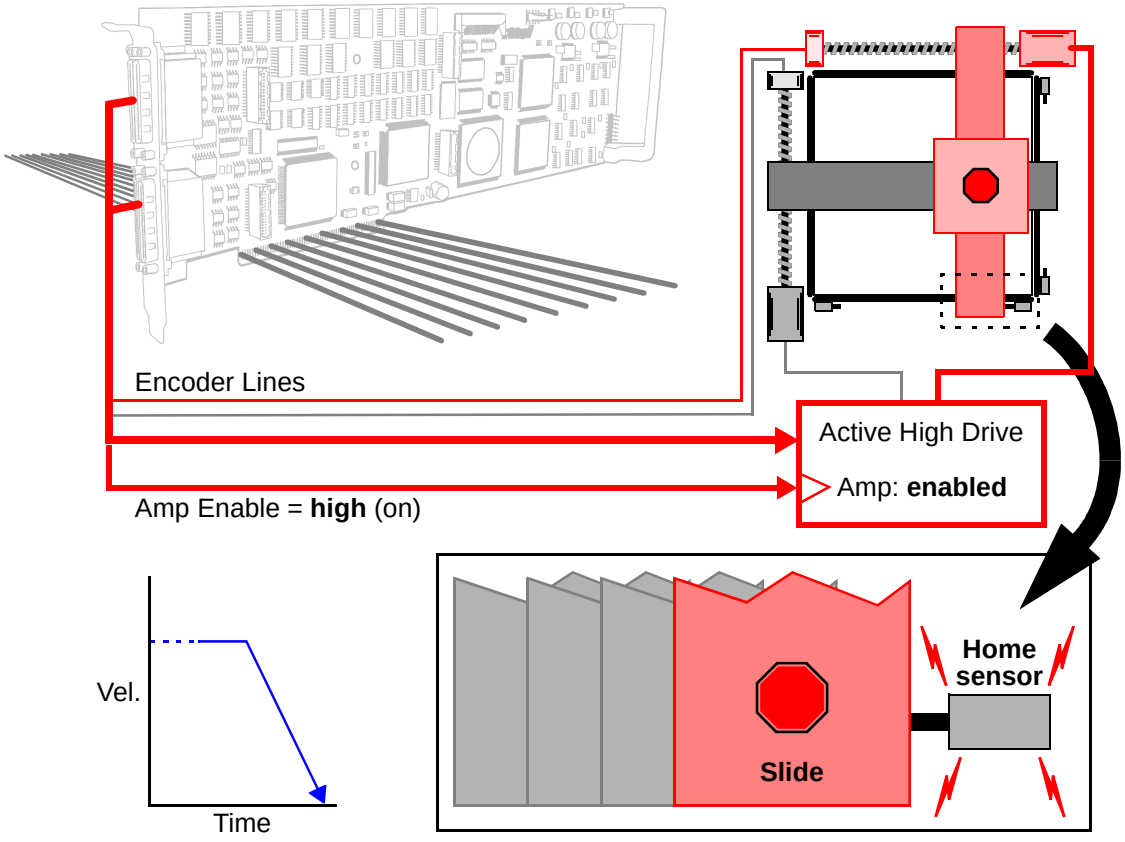
With the trigger armed, the software's next task is to poll the status of the trigger while the axis continues rotating.

MPI

```
/* Poll status until motion done */
motionDone = FALSE;
while (motionDone == FALSE) {
          returnValue =
         mpiCaptureStatus(capture,
                                  &captureStatus,
                                  NULL);
         msgCHECK(returnValue);

         /* Display Home and Capture state */
          returnValue =
                        mpiMotorIoGet(motor,
                                                &io);
                        msgCHECK(returnValue);

          returnValue =
                        mpiMotionStatus(motion,
                                                &status,
                                                NULL);
          msgCHECK(returnValue);

            printf("\rHome:0x%x CaptureState:0x%x MotionStatus:0x%x",
                        io.input & MEIMotorInputHOME,   /* Home input bit */
                           captureStatus.state,              /* Value of Capture bit*/
                         status.eventMask);              /*(1=ARMED, 2=CAPTURED)*/

            if (status.state == MPIStateIDLE)     {
               motionDone = TRUE;
             }
}

if(captureStatus.state == MPICaptureStateCAPTURED)   {
   printf("\nLatched Home Position = %.0lf\n",
             captureStatus.latch[0]);
}
else  {
          printf("\nERROR - Capture Failed. Capture State:  %d\n",
                    captureStatus.state);
}
```

Home event detected

As the axis motor continues turning, the leadscrew rotates until the stage makes contact with the home sensor; this acts as the trigger. When the trigger event is detected, the encoder position is captured and a stop action is initiated.



Define the home position as the origin

When the home pulse is detected in our example, another simultaneous action is initiated: the axis's position is captured by the FPGA. At this point, the axis position is recorded in memory. Meanwhile, the axis slows to a stop. The mpiAxisOriginSet command is then used to set the capture position to zero, making it the new origin of the axis's coordinate system.

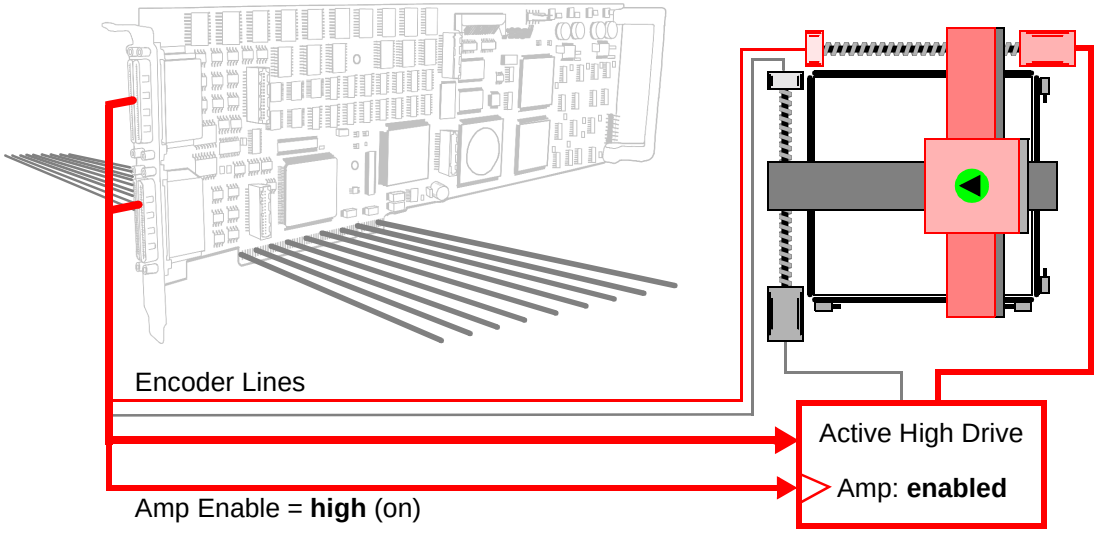
MPI

```
/* Set origin to home position  */
returnValue =
        mpiAxisOriginSet(axis,
                                captureStatus.latch[0]);
msgCHECK(returnValue);
```

Move axis to the (new) origin

Our sample homing routine's last task is to move the axis to the origin. Recall that the home pulse was detected while in motion, and that after detection, the axis coasted to a stop. This means that the axis is now somewhere *beyond* the home point (our new origin).

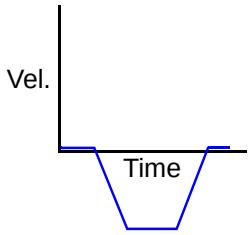
Because we now have a coordinate system, we can perform a backward trapezoidal move from the current position to the intended position (the origin).



MPI

```
/* Move back to home  */
position = 0.0;

params.trapeziodal.trajectory= &trajectory;
params.trapeziodal.position= &position;

returnValue =
        mpiMotionStart(motion,
                       MPIMotionTypeTRAPEZOIDAL,
                          &params);
msgCHECK(returnValue);
```

This is a trapezoidal move.

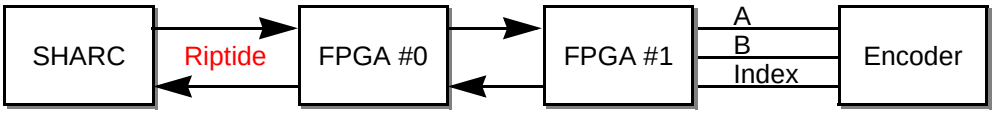


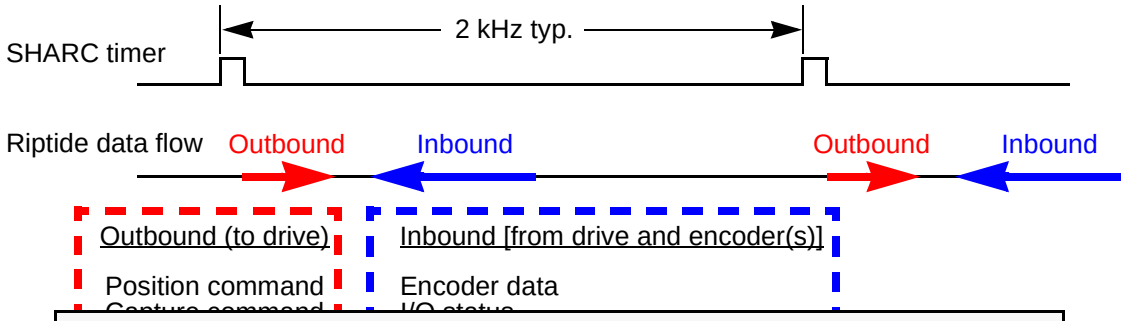
Delete Objects

Finally, the homing routine must delete the objects it has created. This is performed with a number of deletion objects such as **mpiMotionDelete**, **mpiAxisDelete**, **mpiMotorDelete**, etc.

Internal Data Movement via "Riptide"

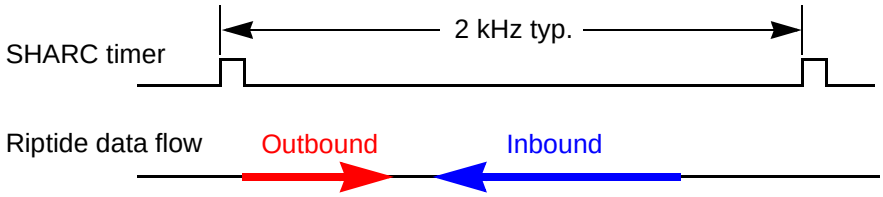
Within a typical SHARC timer cycle, data is moved back and forth between the SHARC and the FPGAs and various feedback devices (e.g., encoders, resolvers, etc.). On the XMP controller, this is performed over a proprietary, dedicated data bus called the "Riptide." The Riptide is the serial backbone of all motion data flow. Within a given SHARC cycle, data moves first from the SHARC to the FPGAs, and then to the drives. Within the same cycle, data is moved in the opposite direction (inbound) from the drive and encoder(s) to the SHARC.





This is a blank page.

As more drives and encoders are added to the motion control system, the required data bandwidth increases across the Riptide, occupying a greater portion of the foreground tasks.



INDEX

US Headquarters

Motion Engineering, Inc.
Santa Barbara, CA
ph +1-805-681-3300 fax +1-805-681-3311

Boston Technical Support Office
Boston, MA
ph +1-978-264-0051 fax +1-978-264-0057

Philadelphia Development Office
Philadelphia, PA
ph +1-215-793-4220 fax +1-215-793-4223

Maryland Development Office
Frostburg, MD
ph +1-301-687-0353

Japan Headquarters

Motion Engineering, KK
Tokyo, Japan
ph +81-3-5540-6431 fax +81-3-5540-6432

Nagoya Technical Support Office
Nagoya, Japan
ph +81-532-45-3511 fax +81-532-45-5415

UK Development Office

Motion Engineering, Ltd.
Bristol, UK
ph +44-117-3179-333 fax. +44-117-3179-303

For Online Support, go to http://support.motioneng.com
www.motioneng.com
info@motioneng.com

Helping you build a better machine, faster.