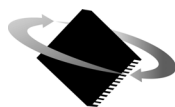


Prodigy[®]/CME PCI User's Guide



P M D

Performance Motion Devices, Inc.
80 Central Street
Boxborough, MA 01719



NOTICE

This document contains proprietary and confidential information of Performance Motion Devices, Inc., and is protected by federal copyright law. The contents of this document may not be disclosed to third parties, translated, copied, or duplicated in any form, in whole or in part, without the express written permission of PMD.

The information contained in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form, by any means, electronic or mechanical, for any purpose, without the express written permission of PMD.

Copyright 1998–2009 by Performance Motion Devices, Inc.

Prodigy, Magellan, ION, Magellan/ION, Pro-Motion, C-Motion, and VB-Motion are registered trademarks of Performance Motion Devices, Inc.



Warranty

PMD warrants performance of its products to the specifications applicable at the time of sale in accordance with PMD's standard warranty. Testing and other quality control techniques are utilized to the extent PMD deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Performance Motion Devices, Inc. (PMD) reserves the right to make changes to its products or to discontinue any product or service without notice, and advises customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

Safety Notice

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage. Products are not designed, authorized, or warranted to be suitable for use in life support devices or systems or other critical applications. Inclusion of PMD products in such applications is understood to be fully at the customer's risk.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent procedural hazards.

Disclaimer

PMD assumes no liability for applications assistance or customer product design. PMD does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of PMD covering or relating to any combination, machine, or process in which such products or services might be or are used. PMD's publication of information regarding any third party's products or services does not constitute PMD's approval, warranty or endorsement thereof.

Related Documents

Magellan[®] Motion Processor User's Guide

Complete description of the Magellan Motion Processor features and functions with detailed theory of its operation.

Magellan[®] Motion Processor Programmer's Command Reference

Descriptions of all Magellan Motion Processor commands, with coding syntax and examples, listed alphabetically for quick reference.

Prodigy[®]/CME Programmer's Reference

Descriptions of all Prodigy/CME product commands, with software architecture overview, command syntax, and examples.

C-Motion[®] Engine Development Tools Manual

Description of the C-Motion Engine, development environment, and software tools.

Pro-Motion[®] User's Guide

User's guide to Pro-Motion, the easy-to-use motion system development tool and performance optimizer. Pro-Motion is a sophisticated, easy-to-use program which allows all motion parameters to be set and/or viewed, and allows all features to be exercised.

Table of Contents

| | |
|--|------------|
| List of Figures | vii |
| Chapter 1. Installation | 9 |
| 1.1 Prodigy Family Overview | 9 |
| 1.2 Prodigy/CME PCI Card Types | 9 |
| 1.3 Software | 10 |
| 1.4 Accessory Products | 12 |
| 1.5 Installation Overview | 13 |
| 1.6 Recommended Hardware | 13 |
| 1.7 Software Installation | 14 |
| 1.8 Preparing the Card for Installation | 14 |
| 1.9 Connection Summary | 16 |
| 1.10 Applying Power | 18 |
| 1.11 First-Time System Verification | 19 |
| 1.12 Developing User Application Code | 23 |
| Chapter 2. Operation | 25 |
| 2.1 Card Function Summary | 26 |
| 2.2 Magellan Motion Processor Functions | 28 |
| 2.3 Magellan-Controlled Functions | 30 |
| 2.4 C-Motion Engine Functions | 35 |
| 2.5 Communications Functions | 39 |
| 2.6 General Card Functions | 43 |
| 2.7 Signal Processing and Hardware Functions | 46 |
| 2.8 Software Libraries | 48 |
| Chapter 3. Accessing Card Resources | 49 |
| 3.1 Resource Addressing | 49 |
| 3.2 Accessing the Communications Ports | 51 |
| 3.3 Accessing On-Card Resources | 55 |
| 3.4 Accessing Magellan-Attached Devices | 56 |
| 3.5 PRP Communication Formats | 58 |
| Chapter 4. Electrical Reference | 61 |
| 4.1 User-Settable Components | 61 |
| 4.2 Connectors | 63 |
| 4.3 Connections Summary—Motor Amplifiers | 70 |
| 4.4 Cables | 73 |
| 4.5 Environmental and Electrical Ratings | 76 |
| 4.6 Mechanical Dimensions | 77 |
| 4.7 User I/O Memory Map | 77 |
| Chapter 5. Interconnect Module | 79 |
| 5.1 IM-1000 Interconnect Module | 79 |
| 5.2 IM-600 Interconnect Module | 81 |
| 5.3 DC-1000 SSI Option Card | 82 |
| Index | 89 |

This page intentionally left blank.

List of Figures

| | | |
|-----|---|----|
| 1-1 | Prodigy/CME PCI Card Component Location | 15 |
| 1-2 | Switch settings | 16 |
| 1-3 | Two Ways to Locate the Code on the Prodigy/CME PCI Card | 24 |
| 2-1 | Prodigy/CME PCI Card Internal Block Diagram | 26 |
| 2-2 | On-card Dual-ported Memory | 35 |
| 2-3 | Overview of C-Motion Engine Architecture | 36 |
| 3-1 | Outgoing and Returning PRP header formats | 50 |
| 3-2 | Example Network Configuration | 53 |
| 3-3 | Example Prodigy/CME PCI Architecture with Ethernet Device Testers | 54 |
| 3-4 | Host Controller & On-card Resources | 55 |
| 3-5 | Host Controller & Magellan-attached Devices | 57 |
| 3-6 | PRP/PCI Packet Format | 58 |
| 3-7 | PRP Message over Serial Format | 59 |
| 4-1 | Components and layout, front of card | 61 |
| 4-2 | Switch Settings by Output Type | 63 |
| 4-3 | Sync I/O connector to three cards | 68 |
| 4-4 | Prodigy/CME mechanical dimensions | 77 |
| 5-1 | IM-1000 location of components | 79 |
| 5-2 | IM-600 location of components | 81 |
| 5-3 | DC-1000 location of components | 84 |
| 5-4 | DC-1000 mounted on Prodigy/CME | 86 |

This page intentionally left blank.

1. Installation

In This Chapter

- ▶ Prodigy Family Overview
- ▶ Prodigy/CME PCI Card Types
- ▶ Software
- ▶ Accessory Products
- ▶ Installation Sequence
- ▶ Recommended Hardware
- ▶ Software Installation
- ▶ Preparing the Card for Installation
- ▶ Connection Summary
- ▶ Applying Power
- ▶ First-Time System Verification
- ▶ Developing User Application Code

1.1 Prodigy Family Overview

The Prodigy® Family of Motion Control Cards provide high-performance control of DC brush, brushless DC, microstepping, and step (pulse & direction) motors. They are available in three form factors; PCI bus, PC/104 bus, and Stand-Alone, and allow multiple forms of communication including PCI bus, PC/104 bus, serial, CANbus, and Ethernet. All Prodigy cards are based on PMD's Magellan® Motion Processors, which perform high speed motion control functions such as profile generation, servo loop closure, pulse & direction signal generation, and many other real-time functions. Certain versions of Prodigy cards include a C-Motion® Engine (CME), which allows the user to offload application code from the host onto the motion card.

The following product selector table summarizes the various members of the Prodigy motion card family and associated Users Guide documentation.

| Prodigy P/N | Bus/Format | C-Motion Engine | User Guide |
|--------------------|-------------------|------------------------|--------------------------------------|
| PR8258x20 | PC/104 | No | Prodigy-PC/104 User's Guide |
| PR8358x20 | PC/104 | Yes | Prodigy/CME PC/104 User's Guide |
| PR9258x20 | PCI | No | Prodigy-PCI User's Guide |
| PR9358x20 | PCI | Yes | Prodigy/CME PCI User's Guide |
| PR13x58x20 | Stand-Alone | Yes | Prodigy/CME Stand-Alone User's Guide |

1.2 Prodigy/CME PCI Card Types

This manual provides a complete User's Guide for the Prodigy/CME PCI cards. For documentation on other members of the Prodigy Family please consult the appropriate documentation.

Prodigy/CME PCI cards come in four different configurations depending on the number of axes supported on the card (from 1 to 4). The following table provides information on the specific card versions that are available. It shows the relationship between card part numbers, Magellan Motion Processor part numbers contained on the card, the number of axes supported, and the type of motors supported. In the motor type column, “all motor types” means DC brush, brushless DC, microstepping, and step (pulse & direction) motors.

| Prodigy/CME PCI Card P/N | Magellan P/N | Number of Axes | Motor Type |
|--------------------------|--------------|----------------|-----------------|
| PR9358120 | MC58120 | 1 | All motor types |
| PR9358220 | MC58220 | 2 | All motor types |
| PR9358320 | MC58320 | 3 | All motor types |
| PR9358420 | MC58420 | 4 | All motor types |

1.2.1 Supported Motor Types

The Prodigy/CME PCI cards support four different motor types; DC Brush, Brushless DC, step motors with a microstepping output, and step motors with a pulse and direction output. Below is a summary of each of these four motor types

DC Brush: output is a single-phase motor command, either in PWM (pulse width modulated), or analog ($\pm 10V$) output format. The output is intended to control DC Brush motors, or brushless DC motors with an amplifier that performs commutation.

Brushless DC: output is multi-phase motor command signals, either in PWM (pulse width modulated), or analog ($\pm 10V$) output format, using Hall-based or sinusoidal commutation. The output is intended to interface with brushless DC amplifiers and motors.

Microstepping: output is multi-phase analog ($\pm 10V$) or PWM (pulse width modulation) waveforms. The output is intended to control 2- or 3-phase step motors using amplifiers which accept this command format.

Pulse & direction: output is standard pulse & direction signals intended to interface with amplifiers which accept this command format.

For complete information on motor output formats and other information, see the *Magellan Motion Processor User's Guide*.

1.3 Software

Four major software packages are provided with the Prodigy/CME PCI cards:

Pro-Motion[®], an interactive Windows-based exerciser & software development tool

C-Motion[®], a C-language library that allows you to create motion applications using the C programming language

C-Motion[®] Engine Development Tools, a set of development resources that allow you to create, download, and monitor programs loaded in the Prodigy/CME PCI card's C-Motion Engine

VB-Motion[®], a set of Active-X objects that lets you create motion applications using the BASIC Programming language

Here is more information on each of these software packages:

1.3.1 Pro-Motion

Pro-Motion is a sophisticated, easy-to-use exerciser program which allows all card parameters to be set and/or viewed, and allows all card features to be exercised. Pro-Motion features include:

- Motion oscilloscope graphically displays processor parameters in real-time
- Axis Wizard to automate axis setup and configuration
- Interactive DC brush and brushless DC tuning
- Project window for accessing motion resources and connections
- Ability to save and load current settings
- Distance and time units conversion
- Motor-specific parameter setup
- Axis shuttle performs continuous back and forth motion between two positions
- C-Motion Engine monitor/debug window
- C-Motion Engine user application code download
- Device window for configuring PMD-based card and digital drive products

Pro-Motion is described in the *Pro-Motion User's Guide*.

1.3.2 C-Motion

C-Motion provides a convenient set of callable routines comprising the code required for controlling the Magellan Motion Processor, whether running on a separate host computer such as a PC, or running on the Prodigy/CME card in the C-Motion Engine. C-Motion includes the following features:

- Magellan axis virtualization
- Ability to communicate to multiple PMD motion cards or modules
- Ability to communicate via PCI bus, PC/104 bus, serial, CANbus, or Ethernet
- Provided as source code, allowing easy compilation & porting onto various run-time environments including PC, microprocessor, embedded card, or C-Motion Engine
- Can be easily linked to any C/C++ application

C-Motion is described in the *Magellan Motion Processor Programmer's Command Reference*.

1.3.3 C-Motion Engine Development Tools

The C-Motion Engine Development Tools includes a source code editor, compiler, linker, and Pro-Motion-based code management & monitoring tools that allow the user to quickly and easily develop applications that will run on a device equipped with a C-Motion Engine. The C-Motion Engine Development Tools includes the following features:

- Complete toolset for creation of user-specific applications running on the motion card
- Open source compiler and C libraries
- Interactive Development Environment
- Supports PCI bus, PC/104 bus, RS232/RS485, CANbus, and Ethernet communications

The C-Motion Engine development tools are described in the *C-Motion Development Tools Manual*.

1.3.4 VB Motion

VB-Motion provides a complete set of methods and properties for developing applications in Visual Basic using a dynamically loaded library (DLL) containing PMD library software. The DLL may also be used from any language capable of calling C language DLL procedures, but no special software support is provided.

VB-Motion includes the following features:

- Magellan axis virtualization
- Ability to communicate to multiple PMD motion cards or Modules
- Ability to communicate via PCI bus, PC/104 bus, serial, CANbus, or Ethernet
- Provided as a single DLL and Visual Basic .NET source code for easy porting onto various PC environments

VB Motion is documented in the *Prodigy/CME Programmer's Reference*.

1.4 Accessory Products

The Prodigy/CME PCI cards can be enhanced with the addition of any or all of the hardware accessory products described in the following table.

| Component Part Number | Description |
|-----------------------|---|
| Cable-1003-01.R | 100-pin high-density ribbon cable. This 3 foot long cable connects the primary motion connector (GP Connector) to an identical connector on a receiving card such as the IM-1000 breakout interconnect card. |
| Cable-1006-01.R | Same as above (Cable-1003-01.R) but 6 feet in length. |
| Cable-3003I-01.R | 68-pin SCSI bracket. This cable plugs into the card's Option Connector and interfaces to a 68-pin ribbon socket through a PCI bracket suitable for connecting to Cable-3003E-01. |
| Cable-3003E-01.R | 68-pin ribbon cable. This 3 foot long cable is a 68-pin ribbon extender cable. It typically connects to Cable-3003I-01.R. |
| Cable-4301-01.R | 10-pin serial header cable. This 1 foot cable plugs into the card's 10-pin serial header connector and provides a single female DB-9 output with dual serial signals. It can be connected to Cable-4355-01.R to provide dual DB-9 serial outputs. |
| Cable-4355-01.R | Dual serial cable. This 5 foot long "Y" cable connects to the DB-9 output of Cable-4301-01.R and provides two female DB-9 connectors suitable for connection to a PC serial port or USB-to-serial converter. |
| Cable-4701-01.R | CANbus header to RJ45 connector. This 1 foot cable plugs into the card's 8-pin CANbus header connector and provides an RJ45 output. |
| Cable-RJ45-02-R | CANbus and Ethernet cable. This 2 meter cable connects to the card's CANbus or Ethernet outputs. It has RJ45 connectors on both ends. |
| TRM-RJ45-02.R | CANbus terminator. This is an RJ45 terminator that may be used to electrically terminate a CANbus network string. |
| Adapt-RJ45T-01.R | CANbus splitter. This Y configuration splitter duplicates the CANbus RJ45 connections into two identical RJ45 connectors, making it easy to connect the card in a daisy chain configuration. |
| Cable-4505-01.R | Ethernet header to RJ45 cable. This 5 foot cable plugs into the card's 10-pin Ethernet header connector and provides an RJ45 output. |
| DC-1000 | Parallel encoder input adaptor. This daughter card module allows parallel-word and other encoders which use the SSI interface format to be directly connected. |
| IM-1000 | Breakout interconnect module provides convenient jack-screw type terminators for the 100-pin cable. Used with Cable-1003-01.R or Cable-1006-01.R. |

| Component Part Number | Description |
|-----------------------|---|
| IM-600 | Breakout interconnect module provides convenient jack-screw type terminators for the 68-pin Option Connector. Used with Cable-3003E-01.R. |
| Adapt-USB232-01.R | This adapter provides USB to serial conversion. It is useful for connecting to the Prodigy/CME's DB-9 serial port from a USB port. |

For information on ordering these accessory products, please contact your PMD representative.

1.5 Installation Overview

- 1 Before using the card, the software must be installed. See Section 1.7, “Software Installation,” on page 14 for instructions on installing the software.
- 2 For a normal installation of a Prodigy/CME PCI card, you will need to configure the card for the specific motor hardware to which it will be connected. See Section 1.8, “Preparing the Card for Installation,” on page 14 for a description of configuring the cards.
- 3 Next, connect the system’s motors, encoders, amplifiers, and sensors to operate the motion hardware. See Section 1.9, “Connection Summary,” on page 16 for a description of the available connections and options for the Prodigy/CME PCI card.
- 4 See Section 5.2, “IM-600 Interconnect Module,” on page 81 if installing the DC-1000.
- 5 Install the Prodigy/CME card into an available PCI slot. Refer to the PC manufacturer’s documentation or web site for additional information regarding the PC’s cover removal, and so on.
- 6 Once this hardware configuration is complete, the final step to finish the installation is to perform a functional test of the finished system. See Section 1.11, “First-Time System Verification,” on page 19 for a description of this procedure.

Once these steps have been accomplished, the installation is complete, and the card is ready for operation.

1.6 Recommended Hardware

To install a Prodigy/CME PCI card, the following hardware is recommended.

- Intel (or compatible) processor, Pentium or better, 300 MB of available disk space, 256MB of available RAM, and a CD-ROM drive. The supported PC operating systems are Windows XP/Vista.
- One to four pulse and direction, PWM, or analog-input amplifiers. The type of amplifier depends on the type of motor being used.
- One to four step, DC brush, or brushless DC motors.

These motors may or may not provide encoder position feedback signals, depending on the type of motor being used. Encoder feedback is a requirement for DC brush and brushless DC motors. For step motors, it’s an option.

- Motion Connectors as required to connect the card to the amplifiers and the motors you have selected.

The GP Connector cable (PMD p/n Cable-1003-01.R, 100-pin high density to dual 50-pin header-type connector) will be required, and for brushless DC motors as well as step motors driven in microstepping mode, the additional Option Connector cable (PMD p/n Cable-3003I-01.R, 68-pin high density connector) will be required. See Section 4.3, “Connections Summary—Motor Amplifiers,” on page 70 for more information on setting up these connections.

1.7 Software Installation

Two CD-ROMs comprise the software distribution for Prodigy/CME cards. All software applications are designed to work with Windows XP/Vista.

- Pro-Motion: The Pro-Motion disk is located in its own Pro-Motion box, and contains the software associated with the Pro-Motion Optimized Motion System Development software.
- PMD Prodigy/CME SDK: The PMD Prodigy/CME SDK disk is located separately, and contains the C-Motion Engine development tools, the C-Motion source libraries, and VB-Motion Libraries.

To install the software:

- 1 Insert the Pro-Motion disk into the CD-ROM drive of your computer.
 - If autorun is enabled, the installation process will begin when the CD-ROM is inserted.
 - If autorun is not enabled, go to the next step.
- 2 On the Start menu, click Run.
- 3 In the Open text box, type D:\setup.exe.
where D: is the drive letter of your computer's CD-ROM drive.
- 4 Follow the on-screen prompts to complete the installation process.

Once installation of Pro-Motion is complete, insert the PMD Prodigy/CME SDK disk, and follow the same procedure above as for Pro-Motion installation.

Upon completion of the installation process for Pro-Motion and PMD Prodigy/CME SDK, the following components will be installed:

- Pro-Motion – an application for communicating to, and exercising the installed card. Refer to the *Pro-Motion User's Guide* for operating instructions.
- C-Motion – source code which may be used for developing motion applications in C/C++ based on the Magellan Motion Processor.
- VB-Motion – Active-X DLLs and example source code which may be used for developing motion applications in Visual Basic based on the Magellan Motion Processor.
- PDF versions of the *Prodigy/CME PCI User's Guide*, *Prodigy/CME Programmer's Reference*, *C-Motion Development Tools Manual*, *Magellan Motion Processor Programmer's Command Reference*, and *Magellan Motion Processor User's Guide*. The Adobe Acrobat Reader is required for viewing these files. If the Adobe Acrobat Reader is not installed on your computer, it may be freely downloaded from <http://www.adobe.com>.

1.8 Preparing the Card for Installation

Figure 1-1 on page 15 shows the location of the resistor packs RS1, RS2, RS3, along with other components such as connectors. The front or top side of the card is shown, with the main- Prodigy/CME PCI connectors at the bottom. All connectors and user-adjustable components are located on the front side of the card. These items are listed in the table below.

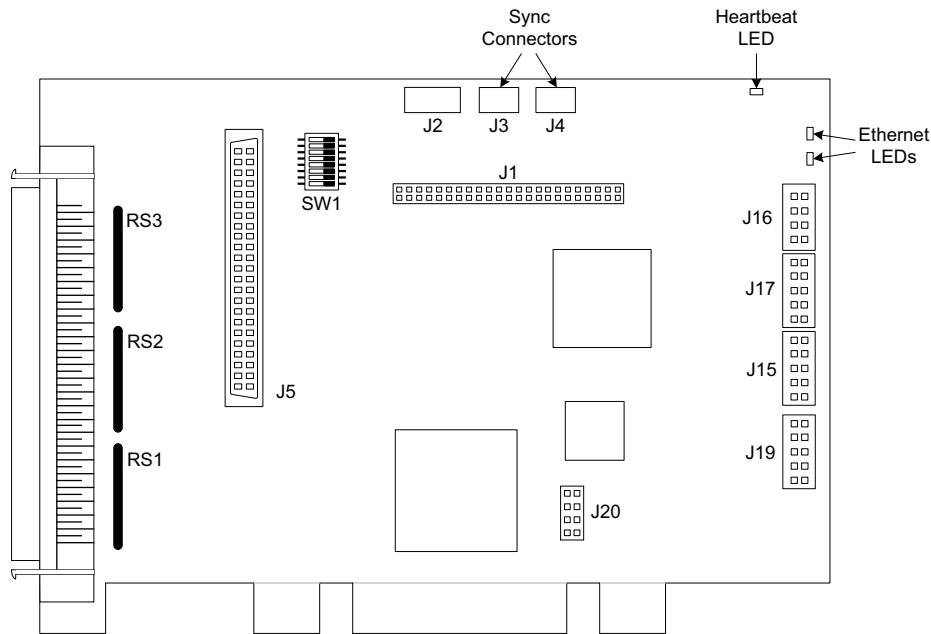


Figure 1-1:
Prodigy/CME
PCI Card
Component
Location

Card Components

The following table describes the components on the card (as shown in Figure 1-1) and their functionality.

| Label | Description |
|-----------|----------------------------|
| RS1 - RS3 | Resistor packs |
| SW1 | Motor output configuration |
| J1 | Extension Connector |
| J2 | Reserved |
| J3, J4 | Synch Connectors |
| J5 | Option Connector |
| J6 | GP Connector |
| J15 | Ethernet Connector |
| J16 | CANbus Connector |
| J17 | Serial Connector |
| J19 | Reserved |
| J20 | Reserved |

1.8.1 Resistor Pack Settings

The Prodigy/CME PCI card has minimal user-adjustable settings. Most settings are software configurable. To prepare the card for installation, the user-specified resistor pack options should be checked, as described in the following table.

| Item | Setting | Description |
|---------------------------------|---|---|
| Resistor packs RS1, RS2, RS3 | Installed; this is the default setting of resistor packs RS1 - RS3. | If differential connections are being used, leave these resistor packs installed. |
| | Removed | If single-ended encoder connections are being used, remove the resistor packs. |

1.8.2 Motor Output Configuration

There are two configurations for setting up the motor output pins on the Prodigy/CME PCI GP Connector: These motor output pins can be set up for use with DC Brush, Brushless DC, and microstepping motors, or the motor

output pins can be set for pulse & direction motors. The default setting is for all motors to be set for DC Brush, Brushless DC, and microstepping.

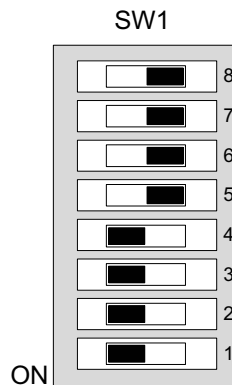
The switch bank SW1 controls these settings, which apply on a per-axis basis. The following table describes the correct switch setting for each output type. Figure 1-2 on page 16 provides a diagram of SW1, and Figure 1-1 on page 15 shows the location of SW-1 on the Prodigy/CME PCI card.

| Axis | Set for DC Brush, Brushless DC, or Microstepping | Set for Pulse & Direction |
|------|--|---------------------------|
| 1 | 1 on, 5 off | 1 off, 5 on |
| 2 | 2 on, 6 off | 2 off, 6 on |
| 3 | 3 on, 7 off | 3 off, 7 on |
| 4 | 4 on, 8 off | 4 off, 8 on |



Power to the card should be turned off when changing the state of the SW1 switches, and care should be taken not to set both switch pairs on (for example both 1 and 5 on, or both 2 and 6 on etc.) or damage may occur to the board.

Figure 1-2:
Switch settings



1.9 Connection Summary

1.9.1 Motor Connections

The following sections summarize the recommended connections for various motor types. DC brush, brushless DC, microstepping, and step (pulse & direction) motors may be connected to the same card, so motor type is allowed to be different on a per-axis basis.

DC Brush Motors

The following table summarizes connections to the Prodigy/CME PCI cards when DC brush motors are used. Between one and four axes may be connected, depending on the specific Prodigy card and application requirements. All connections are made through GP Connector, J6 locatable on the card using Figure 1-1 on page 15.

See Chapter 4, *Electrical Reference*, on page 61 for a detailed list of connections.

| Signal Category | Signal Description |
|--|---|
| Encoder input signals: (per axis) | A quadrature Channel input B quadrature channel input Index pulse channel input |
| Amplifier output signals: (per axis, if PWM sign, magnitude used) | PWM direction PWM magnitude |
| Amplifier output signals: (per axis, if PWM 50/50 used) | PWM magnitude |
| Amplifier output signals: (per axis, if DAC output used) | DAC out |
| Other control signals: (optional per axis) | Home signal input Positive limit switch input Negative limit switch input AxisIn input AxisOut output |
| Miscellaneous signals: | Digital GND, AmpEnable,+5V (for encoder power) |

Brushless DC Motors

The following table summarizes connections to the Prodigy/CME PCI cards when brushless DC motors are used. Between one and four axes may be connected, depending on the specific Prodigy card and application requirements. Connections are made through GP Connector, J6 and Option Connector, J5. Both of these connectors are locatable on the card using Figure 1-1 on page 15.

See Chapter 4, *Electrical Reference*, on page 61 for a detailed list of connections. See Section 4.2.5, “Option Connector,” on page 67 for detailed information regarding the Option Connector.

| Signal Category | Signal Description |
|---|---|
| Encoder input signals: (per axis) | A quadrature channel input B quadrature channel input Index pulse channel input |
| Amplifier output signals: (per axis, if PWM 50/50 used) | PWM magnitude (phase A) PWM magnitude (phase B) PWM magnitude (phase C) |
| Amplifier output signals: (per axis, if DAC output used) | DAC out (phase A) DAC out (phase B) |
| Hall inputs: (Option Con) | Hall (phase A) Hall (phase B) Hall (phase C) |
| Other control signals: (optional per axis) | Home signal channel input Positive limit switch input Negative limit switch input AxisIn input AxisOut output |
| Miscellaneous signals: | Digital GND, AmpEnable,+5V (for encoder power) |

Step Motors

The following table summarizes connections to the Prodigy/CME PCI cards when pulse & direction interface step motors are used. Between one and four axes may be connected, depending on the specific Prodigy card and application requirements. Unless differential pulse & direction output is desired, all connections are made through connector GP Connector, J6. This connector is locatable on the card using Figure 1-1 on page 15.

See Chapter 4, *Electrical Reference*, on page 61 for a detailed list of connections.

| Signal Category | Signal Description |
|--|---|
| Encoder input signals: (optional per axis) | A quadrature channel input B quadrature channel input Index pulse channel input |
| Amplifier output signals: | Pulse Direction |
| Other control signals: (optional per axis) | AtRest signal output Home signal channel input Positive limit switch input Negative limit switch input AxisIn input AxisOut output |
| Miscellaneous signals: | Digital GND, AmpEnable, +5V (for encoder power) |

Microstepping Motors

The following table summarizes connections to the Prodigy/CME PCI cards when microstepping-interface step motors are used. Between one and four axes may be connected, depending on the specific Prodigy card and application requirements. All connections are made through the GP Connector, J6, and Option Connector, J5. Both of these connectors are locatable on the card using Figure 1-1 on page 15.

See Chapter 4, *Electrical Reference*, on page 61 for a detailed list of connections.

| Signal Category | Signal Description |
|---|---|
| Encoder input signals: (optional per axis) | A quadrature channel input B quadrature channel input Index pulse channel input |
| Amplifier output signals: (per axis, if PWM sign, magnitude used) | PWM magnitude (phase A) PWM magnitude (phase B) PWM direction (phase A) PWM direction (phase B) |
| Amplifier output signals: (per axis, if PWM 50/50 used) | PWM magnitude (phase A) PWM magnitude (phase B) |
| Amplifier output signals: (per axis, if DAC output used) | DAC out (phase A) DAC out (phase B) |
| Other control signals: (optional per axis) | Home signal channel input Positive limit switch input Negative limit switch input AxisIn input AxisOut output |
| Miscellaneous signals: | Digital GND, AmpEnable, +5V (for encoder power) |

1.10 Applying Power

Once you have installed the Prodigy/CME PCI card in your PC and have made the necessary connections to your external amplifiers and motor encoders, hardware installation is complete and the card is ready for operation.

After power up, the card will be in a reset condition. In this condition, no motor output will be applied. Therefore, the motors should remain stationary. If the motors move or jump, power down the card and check the amplifier and encoder connections. If anomalous behavior is still observed, call PMD for assistance. Complete PMD contact information is listed on the last page of this manual.

When using a Prodigy/CME PCI card there are some combinations of motor output signal and amplifier type where undesired motion may occur when the card is powered up. In particular, if the connected amplifier is a PWM 50/50 amplifier, the motor will receive 100% power because the Prodigy/CME PCI card defaults to a different PWM signal encoding, PWM sign/magnitude, on reset. To avoid this situation, use AmpEnable as an enable/disable signal for the amplifier, and set the motor type for each axis before enabling the amplifier. See Chapter 2, *Operation*, on page 25 for more information on this function.



To minimize the risk of unintended machine motion during powerup, care should be taken to control the overall powerup sequence of the Prodigy/CME card and connected system motion amplifiers. The recommended sequence is to first powerup the Prodigy/CME card, followed by powerup of the motion amplifiers. The time delay to ensure stable operation of the control electronics before powering up the amplifiers varies depending on the system's power supplies and configuration, but a typical safe figure is 500 mSec - 1,000 mSec.



1.11 First-Time System Verification

The first time system verification procedure summarized below involves initializing each axis of the system and bringing it under stable control capable of making trajectory moves. While there are many additional capabilities that Pro-Motion and the Prodigy motion cards provide, bringing each axis to this level will create a consistent foundation for further, successful exploration and development. Here is a summary of the steps that will be used during first time system verification. Each of these steps will be described below in a separate manual section.

- 1 Initiate Pro-Motion and confirm communication between the PC and the card via the PCI port.
- 2 Run Pro-Motion's Axis wizard for each axis of your system to initialize parameters such as encoder direction and safe servo parameters (if using a servo motor).
- 3 Execute a simple trajectory profile on each axis demonstrating that it is operating correctly and under stable control.

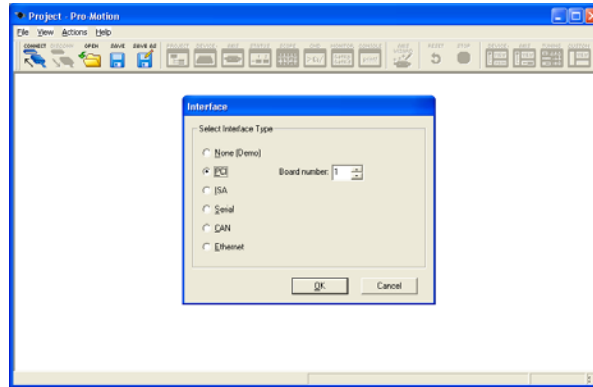
During this first time system setup you may find it useful to refer to other PMD manuals including the *Pro-Motion User's Guide* for complete information on the Pro-Motion application. You may also want to refer to the *Magellan Motion Processor User's Guide* to familiarize yourself with operation of the Magellan Motion Processor, which lies at the heart of all Prodigy Motion cards.

1.11.1 Initiate Pro-Motion

Establishing communications between the PC and the Prodigy/CME PCI card is very simple:

- 1 On the Start menu, click the Pro-Motion application.

When Pro-Motion is launched you will be prompted with an Interface selection window. A typical screen view when first launching Pro-Motion appears below.



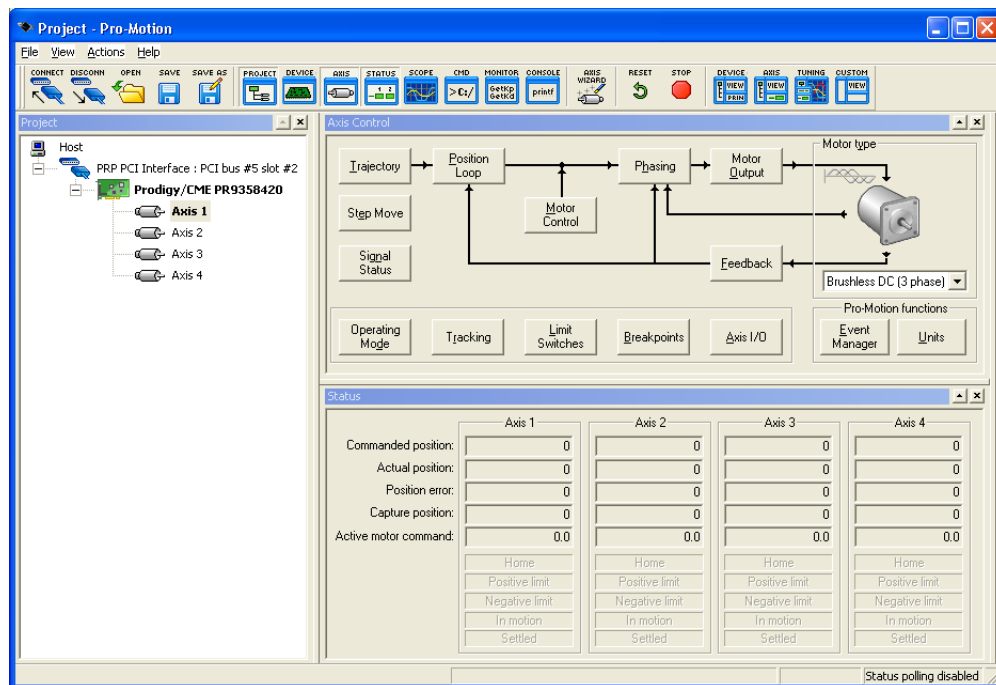
- 2 Click the Connect icon on the toolbar.

Alternatively, on the File menu, click Connect.

The purpose of the Interface dialog box is to indicate to Pro-Motion how your Prodigy/CME PCI card is connected to the PC. It provides various selectable communication options such as PCI, serial, CANbus, Ethernet.

- 3 Click PCI, and then click OK.

If PCI bus communication is correctly established, a set of object graphics loads into the Project window to the left, as shown in the following figure.



For example for a four axis Prodigy card, you see the card name next to an icon of a card, and below that you see four axis icons, one for each available axis of the motion card. Highlighting (single clicking) either the card icon or one of the axis icons with the mouse is used to select specific cards or axes, and is useful later on in the first time system verification.

If PCI bus communications are not correctly established, a dialog box appears indicating that a “Error Opening Port” error has occurred. If this is the case, power down the PC, reseal your card, and repeat from step 1 above. If after repeated attempts a connection can still not be established, call PMD for assistance.

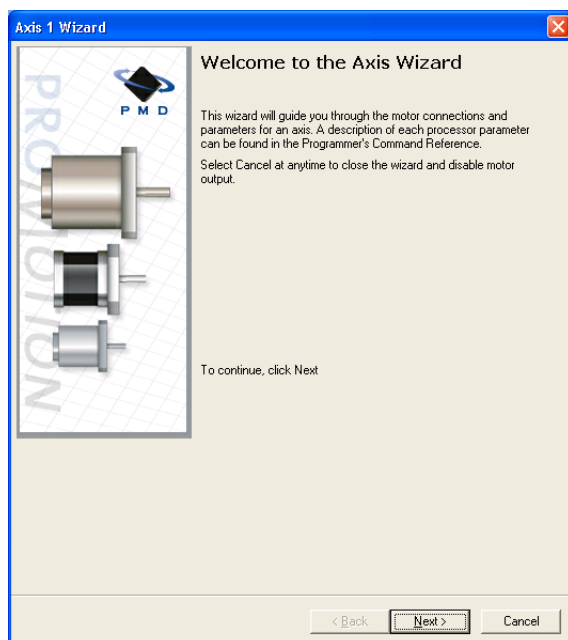
1.11.2 Initializing Motion Axes

The next step to verify the correct operation of the system is to initialize each axis of the motion system sequentially, thereby verifying correct amplifier connection, encoder feedback connections (if an encoder is used), and other motion functions. All of this can be conveniently accomplished using Pro-Motion’s Axis Wizard function. This versatile and easy to use tool initializes all supported motor types including step, DC Brush, and Brushless DC.

To operate the axis wizard:

- 1 Click to select the axis icon that you would like to initialize (normally this would be Axis #1) in the Project window to the left of the screen.
- 2 With this icon highlighted, click the Axis Wizard toolbar button.

The Axis Wizard initialization window appears.



- 3 Click Next and follow the Axis Wizard instructions for each page of the axis initialization process.

A typical axis wizard sequence takes 3-5 minutes. If you have specific questions about the Axis Wizard, refer to *Pro-Motion User's Guide* for detailed information on the axis wizard.

Upon a normal completion of the Axis Wizard, the axis will be ready to make a controlled move. For step motors this means the pulse & direction connections are working properly, and for servo motors this means the encoder and amplifiers connections have been validated, and stable (but not necessarily optimal, see caution below for more information) servo tuning parameters have been loaded into the Prodigy/CME card’s Magellan Motion Processor. Depending on the signals connected, this may also mean that limit switches, and other hardware connections are functioning properly.

The most common reasons for the Axis Wizard to not complete normally are an inability to auto-tune the servo motor, or problems determining the correct commutation sequence for Brushless DC motors when commutated by

the Magellan Motion Processor. Should this happen, it is possible to perform a manual tuning or commutation setup if desired. Refer to the *Pro-Motion User's Guide* for more information, or call PMD for technical assistance.



The Axis Wizard auto tuning routine, which is used with servo motors, is designed to provide stable, but not optimal, parameters for motion. Pro-Motion provides a wealth of functions including a high speed hardware trace oscilloscope that can assist you in determining optimal servo parameters. Values provided by the axis wizard during auto tuning may or may not be safe for your system, and it is up to the user to determine if and when they should be used.

1.11.3 Performing a Simple Trajectory Move

The last step in first time system verification is to perform a simple move for each axis.

To perform a simple move:

- 1 In the Project Window, select the motion axis that you would like to move by clicking the corresponding icon.
- 2 Click the Axis view button on the far right of the toolbar.

Alternatively, click Axis View on the Axis menu.

Your screen organization changes to give easy access to windows that are used while exercising the motion axes.

- 3 Click the Trajectory button in the Axis Control window.

The Trajectory dialog box appears.

- 4 In the Profile mode list, select Trapezoidal.

- 5 Enter motion profiles for deceleration, acceleration, velocity, and destination position (Position 1) that are safe for your system and will demonstrate proper motion.

Pro-Motion provides various selectable units for distance and time, but defaults to units of encoder counts (or pulses for step motors) for distance and seconds for time. This means the default units for velocity are counts/sec, and the default units for acceleration and deceleration are counts/sec². So for a motor that has 2,000 counts per rotation, to perform a symmetric trapezoidal move of 25 rotations with a top speed of 5 rotations per second and with an acceleration time of two seconds, the parameters in the Trajectory dialog box would be set as follows:

Deceleration: 5,000 counts/sec²

Acceleration: 5,000 counts/sec²

Velocity: 10,000 counts/sec

Position 1: 0 counts

Position 2: 50,000 counts

- 6 Click Go and confirm that the motion occurred in a stable and controlled fashion.

Congratulations! First time system verification for this axis is now complete. You should now initialize all of the axes in your system. Go to Section 1.11.2, “Initializing Motion Axes,” on page 21 and repeat the steps.



1.12 Developing User Application Code

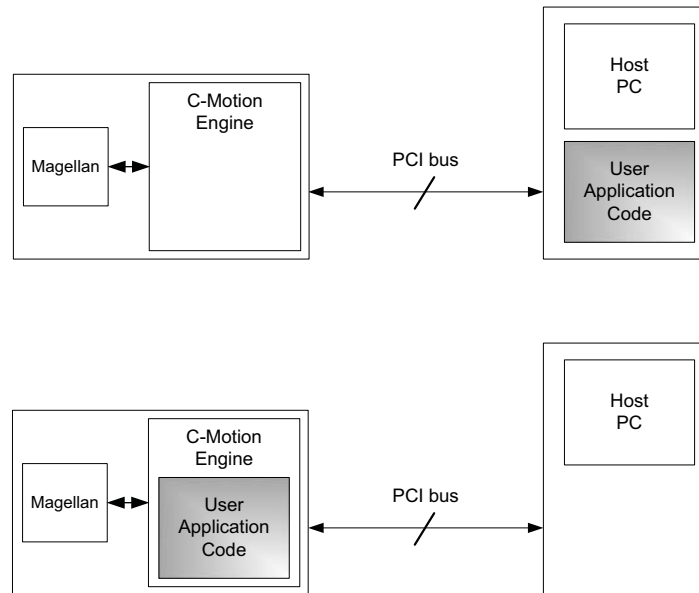
Pro-Motion provides an intuitive, convenient, graphical interface to setup and exercise motion controllers such as Prodigy/CME cards or ION[®] Digital Drives. Eventually though, you will begin to design application-specific software code that will serve as your system controller. PMD supports two standard languages to accomplish this; Visual Basic, through the set of Active-X libraries known as VB-Motion, and C/C++, through the source code-based system known as C-Motion.

For more information on VB-Motion and C-Motion, please consult the *Magellan Motion Processor Programmer's Command Reference*.

1.12.1 Architecture

Figure 1-3 shows two ways to locate your application code with the Prodigy/CME PCI cards.

**Figure 1-3:
Two Ways to
Locate the
Code on the
Prodigy/CME
PCI Card**



When located on a host controller, the user's code communicates via the PCI bus to the Prodigy/CME card. Either VB-Motion or C-Motion can be used to communicate to the card, and the choice of software tools to compile and debug C code is typically determined by the developer. The advantages of a 'host-centered' machine controller approach are that software sequences can be centralized, and the user's code has convenient access to the PC's keyboard, mouse, or touch screen user interface facilities.

When located in the Prodigy/CME PCI card, the user's code communicates directly to the resources available on the card such as the Magellan Motion Processor. This has speed advantages both in communicating with those resources, and in real time code execution predictability.

Another feature of locating code on the card is that the C-Motion Engine can be programmed to receive or send commands to the Prodigy/CME's serial, CANbus, or Ethernet links. In this way the user's application code, downloaded onto the card, forms a local machine controller that can be used to control devices attached to the Prodigy/CME card, thereby unburdening PC-based software, and the PC's network connection hardware, from this task.

By supporting application code on the host controller as well as downloaded directly on the card, the user is provided with multiple options for optimizing the control architecture of his machine, and locating his software on the hardware platform that will best match his machine's operational and performance requirements.

1.12.2 C-Motion Engine

The *C-Motion Development Tools Manual* provides a complete description of how to create C-Motion code that can be downloaded onto the Prodigy/CME PCI's C-Motion Engine.

The C-Motion Engine development environment operates on the PC. Code is edited, compiled, linked, downloaded, and monitored via programs that reside on the PC. Systems which have high level PC-based code concurrently sending commands to the Prodigy card can locate that code on the same PC as the one used for C-Motion code development, or on a separate PC.

All of these considerations and much more are discussed in the *C-Motion Development Tools Manual*, which includes a convenient Getting Started section that introduces the C-Motion Engine IDE (Integrated Development Environment) and walks you through an example session resulting in code being downloaded and executed on the Prodigy/CME PCI card.

2. Operation

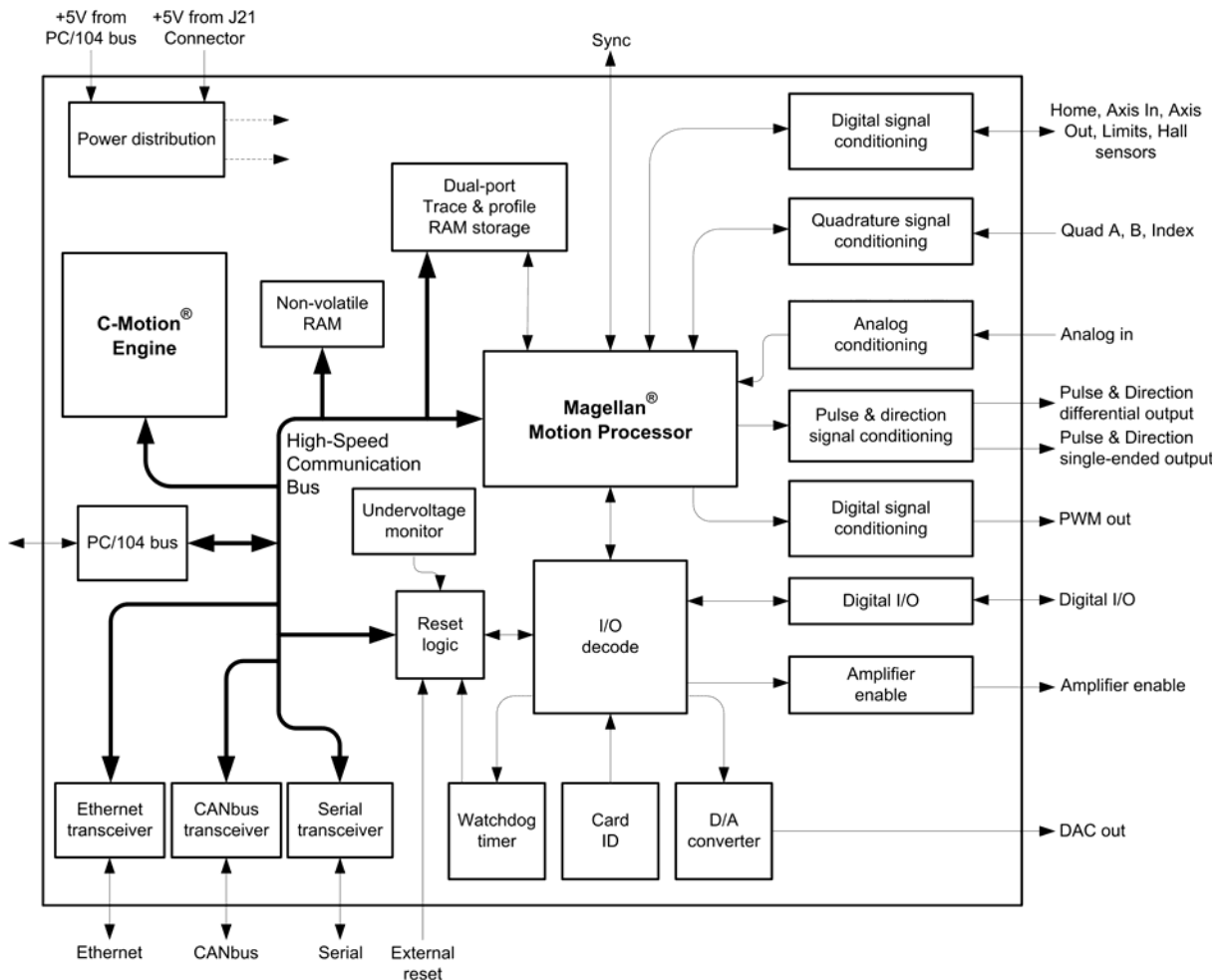
In This Chapter

- ▶ Card Function Summary
- ▶ Magellan Motion Processor Functions
- ▶ Magellan-Controlled Functions
- ▶ C-Motion Engine Functions
- ▶ Communications Functions
- ▶ General Card Functions
- ▶ Signal Processing and Hardware Functions
- ▶ Software Libraries

The Prodigy/CME PCI products are high-performance card-based motion controllers for DC brush, brushless DC, pulse & direction, and microstepping motors. These cards are based on Magellan Motion Processors, which perform motion command interpretation and numerous other real-time functions. The ‘/CME’ series Prodigy cards include a powerful C-Motion Engine module which allows C-Motion application code to be downloaded and executed directly on the motion card, improving run time performance, enabling distributed control architectures, and allowing a complete machine controller to be hosted on a single card.

The following diagram provides a functional block diagram of the Prodigy/CME PCI card:

**Figure 2-1:
Prodigy/CME
PCI Card
Internal Block
Diagram**



2.1 Card Function Summary

The Prodigy/CME PCI card functions can be broken down into six overall categories:

Magellan Motion Processor functions - These are user-accessible functions which reside in the Magellan Motion Processor. Included are profile generation, DC brush and brushless DC loop closure, microstep generation and much more. These functions are accessed through the Magellan Motion Processor's command set, which allows for sophisticated control of the motion axes and related hardware.

Magellan-controlled functions - These are user-accessible functions which are controlled by the Magellan Motion Processor, but which reside outside the Magellan chip on various portions of the card circuitry. These functions include general purpose digital I/O, a watch-dog timer, and many other capabilities.

C-Motion Engine functions - The C-Motion Engine is a self-contained, high performance code execution unit that allows C-Motion code to be downloaded and executed on the Prodigy/CME PCI card. It can communicate with various resources on the card including the Magellan Motion Processor, the card's serial, CANbus, and Ethernet ports, and other on-card peripherals such as the dual-ported RAM.

General card functions - These are user-accessible general purpose card resources not specifically controlled by the Magellan Motion Processor. Examples include card reset, card default parameter restore, and dual-ported and non-volatile RAM access.

Signal processing & hardware functions - A substantial portion of the card performs signal conditioning and other functions associated with safety-related signal processing. These functions are a fixed feature of the card and are not user-accessible.

Communications functions - The Prodigy/CME PCI card provides sophisticated resource addressing capabilities which allow an external host controller, or the onboard C-Motion Engine, to address various on-card and card-connected resources. In addition, various direct management features are provided for the Prodigy/CME PCI card's communication ports (two serial ports, a CANbus port, and an Ethernet port).

Section 2.1.1, "Card Access Basics," on page 27 provides an introduction to general card access issues that will be useful while reading the remainder of this chapter.

Section 2.2, "Magellan Motion Processor Functions," on page 28 through Section 2.7, "Signal Processing and Hardware Functions," on page 46 describe each of these specific functional areas.

Section 2.8, "Software Libraries," on page 48 provides an overview of accessing Prodigy/CME PCI card functions via software libraries.

2.1.1 Card Access Basics

Access to the Prodigy/CME PCI card from the PCI bus, serial, CANbus, or Ethernet ports is provided by a protocol called the *PMD Resource access Protocol* (PRP). This easy-to-use yet powerful system utilizes actions, resources, and addresses to access the Prodigy/CME PCI card's functions. Various card functions are organized into resources, and resources process actions sent to them. Actions can send information, request information, or command specific events to occur. Addresses allow access to a specific resource on the card, or connected to the card, via the serial, CANbus, or Ethernet connections.

A basic communication to the Prodigy/CME PCI card consists of a 16-bit PRP header, and an optional message body. The message body contains data associated with the specified PRP action, but some actions do not require a message body. After a PRP communication is sent to the card, a return communication is sent by the Prodigy/CME PCI card which consists of a PRP header and an optional return message body. The return message body may contain information associated with the requested PRP action, or it may contain error information if there was a problem processing the requested action.

There are five different resource types supported by the Prodigy/CME PCI card. The **Device** resource indicates functionality that is addressed to the entire card, the **MotionProcessor** resource indicates a Magellan Motion Processor, the **CMotionEngine** resource indicates the C-Motion Engine, the **Memory** resource indicates the dual-ported RAM and the non-volatile RAM (Random Access Memory), and the **Peripheral** resource indicates a communications connection.

There are ten different PRP actions including **Command**, which is used to send commands to resources such as the Magellan Motion Processor, **Send** and **Receive**, which are used to communicate using the serial, CANbus, and Ethernet ports, **Read** and **Write**, which are used to access memory-type devices such as the on-card dual-ported RAM, and the non-volatile RAM, and **Set** and **Get**, which are used to load or read parameters.

Chapter 3, *Accessing Card Resources*, on page 49 describes all of these constructs in more detail. In the subsequent sections of this chapter a summary of the PRP actions that are required to access each card function are included with the descriptions of the card functions themselves.



Prodigy/CME card users who write their own software drivers from scratch will need to become familiar with the details of the PRP system and other aspects of card access. The majority of users however will prefer to use C-Motion or VB-Motion, which already encapsulate the details of the card access protocol, and provide convenient and easy to use C and BASIC-callable routines to access the card. For more information see the *Prodigy/CME Programmer's Reference*.

2.2 Magellan Motion Processor Functions

The Magellan Motion Processor block pictured in Figure 2-1 on page 26 forms the core of Prodigy PCI cards. Here is an overview of the functions provided, or managed, by the Magellan Motion Processor on the Prodigy/CME PCI cards:

- Profile generation
- Motor output signal generation (PWM, analog, and pulse & direction)
- Quadrature encoder processing and index capture
- DC brush and brushless DC servo loop closure
- Breakpoint processing
- AxisIn and AxisOut signal processing
- Trace
- Motion error detection, tracking windows, and at-settled indicator
- Limit switches
- Access to various on-card resources such as parallel I/O and dual-port trace & profile RAM

The Magellan Motion Processor interfaces with motion hardware components such as feedback encoders, motor out signal generation hardware, and others, directly through its own pin connections, and through various signal conditioning circuitry. See Section 2.7, “Signal Processing and Hardware Functions,” on page 46 for a complete description of on-card hardware interconnections and signal management.

The Magellan instruction set is very flexible and powerful. The following example, which would be used to set up and execute a simple trapezoidal profile, illustrates just a small part of the overall command set:

```

SetProfileMode Axis1, trapezoidal           // set profile mode to trapezoidal for axis 1
SetPosition Axis1, 12345                    // load a destination position for axis 1
SetVelocity Axis1, 223344                  // load a velocity for axis 1
SetAcceleration Axis1, 1000                // load an acceleration for axis 1
SetDeceleration Axis1, 2000               // load a deceleration for axis 1
SetUpdateMask Axis1, Profile              // specify that an update of profile parameters only
                                           // is to occur
Update Axis1                               // Double buffered registers are copied into

```

// the active registers, thereby initiating the move

Magellan instructions are encoded in packets, which are sent to and from the Magellan Motion Processor. The Magellan processes these packets, performs requested functions, and returns requested data. Within the Prodigy/CME PCI card, the Magellan uses its high-speed parallel-word communications mode to connect to the card's communications bus, which allows the Magellan to be controlled via the C-Motion Engine, or via an external host controller connected to the Prodigy/CME PCI card by PCI bus, serial, CANbus, or Ethernet port.

Two manuals describe how the Magellan Motion Processor operates and how it is programmed: the *Magellan Motion Processor User's Guide*, and the *Magellan Motion Processor Programmer's Command Reference*. These documents also describe VB-Motion, and C-Motion, which are the software libraries that are used to send commands to the Magellan chip and exercise its many functions.

2.2.1 Accessing the Magellan Motion Processor

To send and receive command packets to the Magellan Motion Processor the PRP action **Command** is used. The Magellan command packet is loaded into the PRP message body, and the return PRP message body contains the return packet provided by the Magellan. A return without error indicates that the command was processed successfully. If an error occurred while the Magellan was processing the command, the message body is loaded with the specific error that occurred. For more information on Magellan command packet formats and return packet formats see the *Magellan Motion Processor Programmer's Command Reference*.

In addition to accessing the on-card Magellan, it is also possible to access Magellan Motion Processors that are connected via the Prodigy/CME PCI card's attached networks such as the CANbus, Serial, or Ethernet networks. An additional PRP action, **Open**, sent to either the **Device** or the **Peripheral** resource, is used to establish a connection to such a motion processor. See Section 3.4, "Accessing Magellan-Attached Devices," on page 56 for more information on connecting to motion processors on attached networks.

For complete information on the format and function of these, and other PRP actions, refer to the *Prodigy/CME Programmer's Reference*.

2.2.1.1 Magellan Reset

Although a reset occurs automatically during power-up, it is sometimes desirable to reset the Magellan Motion Processor explicitly through a user-initiated action. The PRP action **Reset**, when sent to the MotionProcessor resource, causes a reset of the Magellan Motion Processor. This reset affects the Magellan Motion Processor, and a number of Magellan-controlled signals. Note however that this type of reset is different than a full card reset initiated via the PRP action **Reset**. See Section 2.6.2, "Reset," on page 43 for a description.

After a Magellan reset occurs, some of the Prodigy/CME PCI card's output signals will be driven to known states. These are summarized in the following table:

| Signal Name | State |
|----------------|-----------|
| AxisOutI-4 | High |
| PWMMagIA-4C | Low |
| PWMSignIA-4B | Low |
| DACIA-DAC4B | No change |
| DigitalOut0-7 | No change |
| AmpEnableI-4 | No change |
| DAC On/Off | No change |
| Watchdog Timer | No change |

2.2.2 Connections & Associated Signals

The Prodigy/CME PCI card's Magellan Motion Processor has extensive signal connections associated with it to allow interfacing to various motion peripherals. See the *Magellan Motion Processor User's Guide* as well as Section 2.7, "Signal Processing and Hardware Functions," on page 46 for a complete description of the signals managed by the Magellan Motion Processor.

2.3 Magellan-Controlled Functions

There are a number of motion control hardware features on the Prodigy/CME PCI card that are controlled through the Magellan Motion Processor, but are not located within the Magellan Motion Processor itself. Because of this they are not specifically described in the *Magellan Motion Processor User's Guide*, or the *Magellan Programmer's Command Reference*, instead being described here, in subsequent sections.

These Magellan-controlled card functions are:

- General Purpose Digital I/O
- Amplifier Enable
- DAC Output Enable
- Watchdog Timer
- Undervoltage Monitor
- Reset Monitor
- Card ID
- Dual-Port Trace & Profile RAM

These motion hardware control functions are accessed via the Magellan's **ReadIO** and **WriteIO** commands, which provide control of hardware peripherals external to the Magellan Motion Processor. In addition, many of the Prodigy PCI card motion hardware control functions can be accessed using VB-Motion and C-Motion 'named' commands specific to that function. See the subsequent feature-specific sections for a description of both access methods.

The **ReadIO** and **WriteIO** commands are similar to any other Magellan packet commands in that they are accessed via the PRP action command. See Section 2.2.1, "Accessing the Magellan Motion Processor," on page 29 for detailed information.

2.3.1 General-Purpose Digital I/O

In addition to numerous special-purpose digital signals which are input or output to the card, and which are directly processed by the Magellan Motion Processor such as *AxisIn*, *AxisOut*, *Home*, and *QuadA*, the Prodigy/CME PCI cards also support eight general-purpose inputs, and eight general-purpose outputs. These signals provide a convenient way of accessing additional application-specific general-purpose digital I/O.



The general purpose digital inputs are TTL-compatible with typical input range of 0~5.5V. The absolute maximum input range is -0.5~7V.

The general purpose digital outputs are 5V TTL-compatible with an output sink/source current of 24mA.

The eight inputs and eight outputs are read using the Magellan Motion Processor's **ReadIO** command and written using the **WriteIO** command, with an I/O address of 0. This is illustrated in the following table, along with the bit locations of the input and output signals.

| I/O Address | Bit Location | Signals |
|-------------|--------------|---------------|
| 0 | 0 - 7 | DigitalOut0-7 |
| | 8 - 15 | DigitalIn0-7 |

To read all eight general-purpose digital I/O signals, a **ReadIO** command is performed at address offset 0. The 16-bit read word returns the current output values (set using the **WriteIO** command) in bits 0 - 7, while bits 8 - 15 hold the digital values corresponding to the signal levels at the connector for those inputs. To write new signal values to the eight digital outputs, a **WriteIO** command to address offset 0 is sent, and the values on bits 0-7 will be output to the signal connections. The values of bits 8 - 15 are ignored.

Example

To write the value 0xAA to bits 0 - 7, the command **WriteIO** is used. If the signal pattern 0x55 is present on the eight input connections, then the command **ReadIO 0** will return the value 0x55AA. The upper eight bits reflect the present value of the input signals, while the lower eight bits reflect the 8-bit value being output.

In addition to the low-level **ReadIO** and **WriteIO** commands, the following commands are also supported in C-Motion and VB-Motion: **WriteDigitalOutput**, **ReadDigitalInput**, and **ReadDigitalOutput**. These commands provide a clearer and simpler interface to the Prodigy/CME PCI card's general purpose I/O signals by handling the byte shifting.

2.3.1.1 Connections & associated signals

The general-purpose I/O are direct digital inputs and outputs. There are no associated connections required for these signals to function properly, however, one or more of the digital grounds must be connected. Digital inputs are pulled up through 4.7 kOhm resistors to 5V. The power-up default value for all general-purpose digital outputs is low.

See Chapter 4, *Electrical Reference*, on page 61 for a complete description of the pinout connections to and from the card.

2.3.2 Amplifier Enable

The signals **AmpEnable 1-4** provide four digital outputs which may be used as amplifier enable signals. They can also be used as general-purpose digital outputs. They can be read or written using the Prodigy/CME PCI card Magellan's **ReadIO** and **WriteIO** commands.

These outputs are read using the **ReadIO** command, and written to using the **WriteIO** command, using an address of 1, as shown in the following table:

| I/O Address | Bit Location | Signals |
|-------------|--------------|---|
| 1 | 0-3 | Amplifier enable outputs (0-3) |
| | 4-6 | Unused |
| | 7 | DAC enable status (1 = enabled; 0 = disabled) |
| | 8-11 | Change mask for bits 0-3; amplifier enable outputs (1 = change; 0 = don't change) |
| | 12-14 | Unused |
| | 15 | Change mask for DAC enable (1 = change; 0 = don't change) |

To read the status of the amplifier enable outputs, the command **ReadIO** is used at address 1. The values currently being output will appear in bits 0 - 3. To write values to the amplifier enable output signals, the **WriteIO** command is used with an address of 1. An amplifier enable signal will be changed only if the corresponding change mask bit is set. The change mask bits are bits 8 - 11, the values to be loaded are bits 0 - 3.

Examples

```
WriteIO Address1, 0x0505      // write 0x0505 to I/O Address 1 to enable Amplifiers 1 & 3, don't change 2 & 4
WriteIO Address1, 0x0400      // write 0x0400 to I/O Address 1 to disable Amplifier 3, don't change 1, 2, & 4
WriteIO Address1, 0x0100      // write 0x0100 to I/O Address 1 to disable Amplifier 1, don't change 2, 3, & 4
```

In addition to the low-level **ReadIO** and **WriteIO** commands, the following commands are also supported in C-Motion and VB-Motion: **SetAmplifierEnable** and **GetAmplifierEnable**. These commands provide a clearer and simpler interface by handling the bit shifting.

2.3.2.1 Connections & associated signals

AmpEnable1-4 are direct digital outputs. There are no associated connections required for these signals to function properly, however, one or more of the digital grounds must be connected. The power-up default value for all amplifier enable signals is low (disabled). See Chapter 4, *Electrical Reference*, on page 61 for a complete description of the pinout connections to and from the card.

2.3.3 DAC Output Enable

In addition to the amplifier enable outputs, there is a dedicated card function which allows the DAC output signals to be shunted to 0 volts for safety purposes (DAC disabled), or to be controlled by the Magellan Motion Processor (DAC enabled). This shunting occurs at a hardware level outside the motion processor itself, and provides an additional safety layer to control the motor command.

The status of the DAC output enable function can be read using the **ReadIO** command, and the DAC output enable status can be set using the **WriteIO** command, with an address of 1. The table in Section 2.3.2, "Amplifier Enable," on page 31 shows this. To read the status of the DAC output enable function, **ReadIO** is used. The value currently in use will appear in bit 7. A value of 1 indicates DAC output is enabled, meaning that the voltage being output by the DACs is controlled by the motion processor. A value of 0 indicates that it is disabled, meaning that the voltage being output by the DAC is forced to 0.0 volts.

To enable or disable the DAC enable function, the **WriteIO** command is used. The change mask bit located at bit 15 must be loaded with a 1. Bit 8 must be loaded with a value of 0 to disable, or a value of 1 to enable output.

The powerup default value for DAC Output Enable is disabled. In addition, the DAC Output Enable is disabled upon a card reset, or via the external **Reset** signal. See Section 2.6.2, "Reset," on page 43 for more information.

Examples

```
WriteIO Address1, 0x8080      // write 0x8080 to I/O Address 1 to enable all DACs,
                               // don't change amplifier enables
WriteIO Address1, 0x8000      // write 0x8000 to I/O Address 1 to disable all DACs,
                               // don't change amplifier enables
```

In addition to the low-level **ReadIO** and **WriteIO** commands, the following commands are also supported in C-Motion and VB-Motion: **SetDACOutputEnable** and **GetDACOutputEnable**. These commands provide a clearer and simpler interface by handling the bit shifting.

2.3.4 Card Watchdog Timer

To enhance the overall safety of the card, a watchdog function has been included. When enabled, the watchdog will automatically trigger a card reset if communication to it should be lost. See Section 2.6.2, “Reset,” on page 43 for a description of the actions and signal changes that occur after a card reset.

To enable the card watchdog timer, the **WriteIO** command is used to send a value of 0x5562 to address 4. Once enabled, the watchdog timer will time out, causing a hard reset, if another write to address 4 with a value of 0x5562 is not received within 104 mSec. As long as a watchdog value is written to address 4 within the 104-mSec interval, no reset will occur and motion operations will proceed normally. Once enabled, the watchdog mechanism cannot be stopped until a reset or power cycle occurs.

After powerup or card reset, if no command is sent to the watchdog address, then the watchdog will remain disabled. The watchdog is disabled by default at power-up. When the watchdog timer times out and triggers a reset, it also disables itself.

In addition to the low-level **ReadIO** and **WriteIO** commands, the following command is also supported by C-Motion and VB-Motion: **SetWatchDog**. This command provides a clearer and simpler interface by automatically sending the value 0x5562.

2.3.5 Undervoltage Monitor

To enhance reliability under a variety of electrical conditions, an undervoltage detection circuit has been included. This circuit triggers a hard reset when the voltage has dropped to an unsafe level. Resetting the Prodigy/CME PCI card will have the result of setting all motor command outputs to zero, thus allowing the motors to come to a safe stop. An undervoltage condition is detected when the 3.3V internal supply on the card drops below 2.7V. See Section 2.3.6, “Reset Monitor,” on page 33 to determine if a reset was caused by an undervoltage condition.

2.3.6 Reset Monitor

During normal operations, the Prodigy/CME PCI card is only reset during power-up. There are, however, several other ways that the Prodigy/CME PCI card can be reset. See Section 2.6.2, “Reset,” on page 43 for a complete description of how the card can be reset. A reset serves the purpose of initializing values and bringing the Prodigy/CME PCI card to a known and consistent state.

To determine the cause of a card reset, special instructions to read the reset source have been provided. The command **ReadIO** with an address of 2 should be used. The following table details the encoding of this I/O address word.

| I/O Address | Bit Location | Signals |
|-------------|--------------|---|
| 2 | 0-10 | <i>Reserved</i> |
| | 11 | C-Motion Engine user application code fault. A 1 value in this bit indicates an instruction or address access fault. |
| | 12 | Commanded reset. A 1 value in this bit indicates a card-level reset commanded via the PRP action Reset. |
| | 13 | Undervoltage detection: a 1 value in this bit indicates a reset caused by undervoltage detection. |
| | 14 | External signal: a 1 value in this bit indicates a reset caused by the external Reset signal, pin 91 of the GP Connector (J6) |
| | 15 | Card Watchdog timeout: a 1 value in this bit indicates a reset caused by the card watchdog timeout. See Section 2.3.4, “Card Watchdog Timer,” on page 33 for a description. |

Once a reset condition has occurred, the reset status stored at address 2 (described in the preceding table) can be cleared by a **WriteIO** command to address 2 with a value of zero (0).

Example

To determine that a reset has occurred, and to determine the cause of the reset, the command **ReadIO** is used. Assuming that a watchdog timer event has occurred, the value returned would be 0x8000. To clear the reset monitor word, the command **WriteIO** is sent to address 2 with a value of zero (0).

In addition to the low-level **ReadIO** and **WriteIO** commands, a **GetResetCause** command is also supported by C-Motion and VB-Motion. This command returns the cause and also clears the reset condition.

2.3.7 Card ID

This feature allows the user to query the card for a Card ID. This may be helpful for verifying the type of Prodigy Motion Card in situations where multiple cards of varying types are installed.

2.3.7.1 ReadIO and WriteIO commands

To read the Card ID, the **ReadIO** command is used with an address of 0xFF. The encoding of the bits returned is detailed in the following table:

| Address | Bit Location | Signals |
|---------|--------------|---|
| 0xFF | 0-3 | Major card revision: This nibble encodes the major card revision. This value can range from 0 to 15. |
| | 4-7 | Minor card revision: This nibble encodes the minor card revision. This value can range from 0 to 15. |
| | 8-11 | Card generation: This nibble encodes the card generation. This value can range from 8 to 15. (0 to 7 are reserved for older motion card families.) |
| | 12-14 | Card type: This nibble encodes the card type and has one of the following values: 0 = ISA Bus 1 = PCI Bus 2 = CompactPCI 3 = PC/I04 4 = MIPS 5 = RS232 6 = CAN 7 = Standalone |
| | 15 | 0 = Standard Prodigy card 1 = Prodigy/CME PCI card |

Example

To read the Card ID, the command **ReadIO** is used. For example, the value 0xf805 would be interpreted as: Prodigy/CME PCI card, card generation 8, card revision 5.0.

In addition to the low-level **ReadIO** command, a **ReadCardID** command is also supported by C-Motion and VB-Motion. This command returns the Card ID in the format described above.

2.3.8 Dual-Ported RAM (Trace Buffer)

The Prodigy/CME PCI card has 64 KBytes of on-card dual-ported memory (DPRAM) which has one 'port' interfaced to the Motion Processor, and the other 'port' interfaced to the card's high speed internal communications bus, allowing two paths of communication. Figure 2-2 on page 35 shows this configuration.

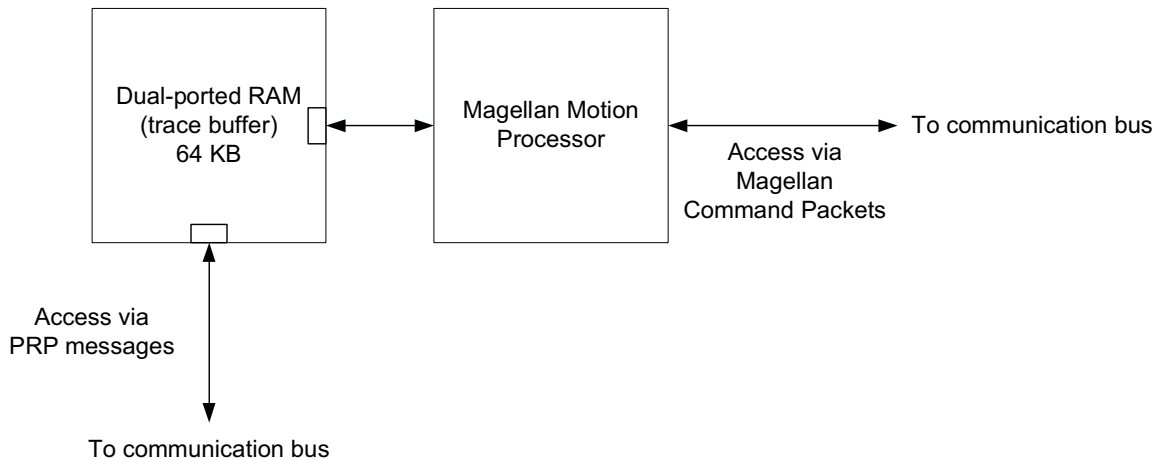


Figure 2-2:
On-card Dual-ported Memory

The dual-ported RAM trace buffer is a powerful feature that allows various Magellan Motion Processor parameters and registers to be continuously captured and stored to a memory buffer. The captured data may be downloaded to the C-Motion Engine or to an off-card host using the Prodigy/CME PCI card's PCI bus, serial, CANbus, or Ethernet communication channels. Magellan data traces are useful for optimizing DC brush and brushless DC servo performance, verifying trajectory behavior, capturing sensor data, or to assist with any type of monitoring where a precise time-based record of the system's behavior is required. For more information on how to set up a trace within the Magellan Motion Processor, see the "Trace Capture" section of the *Magellan Motion Processor User's Guide*.

2.3.8.1 Accessing the dual-ported RAM

To access the contents of the trace buffer via the Magellan port the Magellan's built in memory buffer commands are used. The Magellan provides a sophisticated command set that lets you set up, monitor, and read from the trace buffer. See the *Magellan Motion Processor User's Guide* for more information on these commands. In this read configuration, the Magellan Motion Processor stores data to the DPRAM autonomously, and the host controller reads the data using the Magellan Motion Processor as well.

An alternate path for reading the trace buffer after it has been written to by the Magellan Motion Processor is via the Prodigy/CME PCI card's high speed communication bus. In this mode the dual-ported RAM is referred to as a resource, and is accessed via the PMD Resource Access Protocol. See Section 2.6.4, "Dual-Ported RAM (Trace Buffer)," on page 44 for more information on this access method.

The contents of the dual-ported RAM are volatile. They are not saved during power-down of the card.



2.4 C-Motion Engine Functions

The C-Motion Engine on the Prodigy/CME PCI card allows C-Motion code to be downloaded and executed on the card. The C-Motion Engine is a powerful and flexible engine that can be used to:

- Operate Prodigy motion cards in a standalone mode
- Offload time-critical code from the host to the motion card
- Create a complete machine controller that communicates via PCI bus, serial, CANbus, or Ethernet to a cell controller or other high level controller

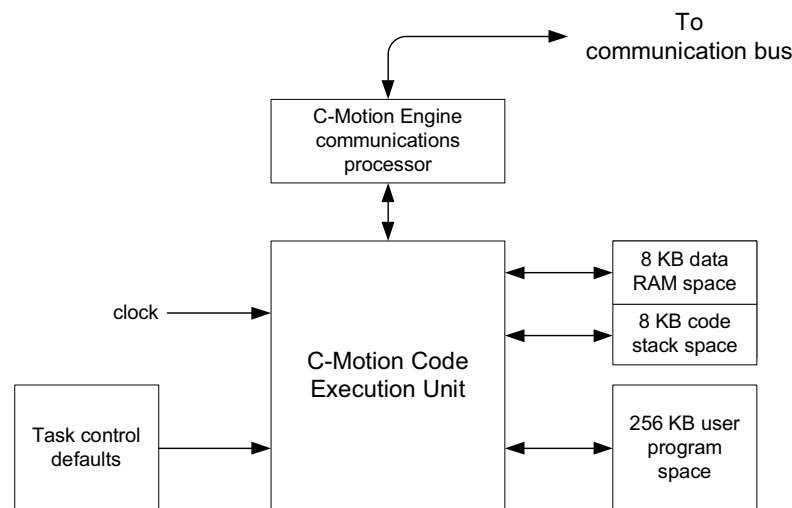
- Extend the functionality of the Magellan Motion Processor with higher level functions such as contouring, macros, or other complex behaviors
- Lower system cost by combining a motherboard function with a dedicated motion card function in a single-card format.

2.4.1 C-Motion Engine Hardware Configuration

The C-Motion Engine is a self-contained module that provides non-volatile RAM space to store downloaded user application code, RAM space for ‘scratch’ data variable storage, and connections to the communication bus allowing the C-Motion Engine to send and receive messages through the network ports, communicate with the Magellan Motion Processor, and access other on-card resources such as the dual-ported RAM.

Creating, compiling, downloading, and verifying a specific user C-Motion application on a Prodigy/CME PCI card is accomplished with the C-Motion Engine development system, described in *C-Motion Development Tools Manual*. The outcome of such a development sequence is a downloadable code image, run on the C-Motion Engine, that contains the user application code and that is executed by the C-Motion Engine on the Prodigy/CME PCI card. Figure 2-3 provides an overview of the architecture of the C-Motion Engine.

**Figure 2-3:
Overview of
C-Motion
Engine
Architecture**



The following table provides an operational overview of the capabilities and resources provided by the C-Motion Engine:

| Resource | Specification |
|--|---------------|
| MIPS (millions of instructions per second) | 96 |
| User program space (stored in flash) | 256KB |
| User data RAM space | 8 KB |
| User code stack space | 8 KB |

2.4.2 Powerup & Operation

Upon reset or power up the C-Motion Engine initializes itself and checks to see whether execution of user application code, if downloaded, should automatically begin. If the factory default settings have not been changed, user application code automatically begins executing and continues until the card is powered down or until a specific ‘stop executing’ command is given. If this default value is changed, then the C-Motion Engine will hold in a wait state, and code execution will not occur automatically.

While there are numerous safety checks and features built into the C-Motion Engine system, application code developed for the C-Motion Engine is C-based, and thus there are limits to code size, RAM usage, and stack usage

that should be observed during run time operation of downloaded C-Motion code. The table above provides these numerical limits.

For user downloaded code that does not correctly observe these limits, or for files that have become corrupted, there are a number of fault conditions that can occur while the C-Motion Engine is executing downloaded user application code. These very serious run-time faults include instruction errors - indicating that an unknown instruction was encountered during execution of the user's code, and address faults - indicating that either a program space or RAM space access limit was violated. If either of these conditions occur, the C-Motion Engine will immediately halt user code execution, and reset the Prodigy/CME PCI card. This C-Motion Engine-initiated reset is identical to the reset that occurs after sending a PRP **Reset** action, except that the cause of the reset is recorded as 'C-Motion Engine user code Fault' rather than 'commanded' reset. See Section 2.6.2, "Reset," on page 43 for more information on the Reset command. See Section 2.3.6, "Reset Monitor," on page 33 for information on retrieving the reset cause.

Whether or not user application code is running, after reset or power up, the C-Motion Engine begins processing PRP actions sent to it. These commands are typically sent from a host controller. The supported commands include functions such as checking the downloaded user application code version stored in the C-Motion Engine, and sending and receiving messages to the user code loaded onto the C-Motion Engine.

For additional guidelines on managing run-time usage of the C-Motion Engine see the *C-Motion Development Tools Manual*.

2.4.3 Task Control

The primary purpose of the C-Motion Engine is to execute user application code that has been downloaded to it using the C-Motion Engine development system.

In a production environment, this code will typically automatically start upon power up, and run continuously while the system is in operation. For debugging however, there are a number of additional controls.

At any point in time it is possible to stop or restart execution of the C-Motion Engine user application code. To access this function the PRP action **Command** is sent to the **CMotionEngine** resource.

Extreme caution should be applied when stopping or starting user application code running on the C-Motion Engine, as depending on the specific application code, this may cause unexpected or unsafe motion. It is the responsibility of the user to determine whether stopping or restarting of user application code is safe and appropriate.



Whether or not the user application code automatically executes upon powerup or reset can also be controlled. The two options are operation under manual mode, in which case the user's code will not begin execution until an explicit start command is given, and auto-start, where the code automatically begins execution from power up or reset. The PRP action **Set** sent to the **CMotionEngine** resource allows setting of the user code start mode.

In addition to these functions, it is also possible to determine whether the user application code is presently running or not. This status information may be useful during code debugging, or to help diagnose problems. This capability is accessed via a **Get** action sent to the **CMotionEngine** resource.

For a detailed description of the supported Prodigy/CME commands see the *Prodigy/CME Programmer's Reference*.

2.4.4 Sending Messages to/from User Application Code

A common function of user application code running on the C-Motion Engine is to parse command messages sent to it by a host controller. For example a user might write code for the C-Motion Engine that responds to an "Extend

Robot Arm” command sent by the host controller, and then send a series of commands to the Magellan Motion Processor to execute this motion sequence. At the end of the motion sequence the user application code might send an “Arm Extended” message confirming the movement sequence has completed.

One method of achieving this is to use the Prodigy/CME PCI card’s peripheral mechanism described in Section 3.2.1, “Peripheral Connections,” on page 52 to open, and operate, a low-level communications link via the PCI bus, serial, CANbus, or Ethernet link. This method has the advantage of giving relatively direct control over the communication traffic. The disadvantage is that the user has to implement specific send and receive communications in the host controller, and the C-Motion Engine needs to have similar code implemented that can process these messages.

Another method that may be more convenient, particularly during early debugging of the user’s application code, is to use a capability of the PRP system to connect directly to the user application code on the C-Motion Engine. Messages sent and received by the C-Motion Engine from a host controller are stored in a special buffer, and can be easily read or written to by the user application code. In addition, PMD’s Pro-Motion application supports a simple way of entering, sending, and/or receiving such messages. This makes it easy to manually enter commands from Pro-Motion and exercise the user application code which is programmed to parse these messages.

To utilize this approach, a PRP **Send** or **Receive** action is sent to the **CMotionEngine** resource. To receive these messages within the C-Motion Engine a special ‘user packet’ peripheral is opened. For more information see the *Prodigy/CME Programmer’s Reference*.

In addition to these communications commands, when sent to the **CMotionEngine** resource, the PRP action **NOP** performs a basic connection check. The message body is empty. A return without a PRP error code indicates that the C-Motion Engine is accessible and processing commands.

2.4.5 Connecting to the C-Motion Engine Code During Development

To develop code that is downloaded to the C-Motion Engine, a communications link is required between the C-Motion Engine and the PC-based C-Motion Engine Development Environment. This link contains the information required for code downloads, as well as other information utilized during application debugging.

While serial, CANbus, or Ethernet can all be used as the communications link during development, for the Prodigy/CME PCI cards, typically this link is the PCI bus, because it is more convenient and faster than the network-based connections.

Selecting which Prodigy/CME PCI communication channel will be used for code download is specified via the C-Motion development system. For more information see the *C-Motion Development Tools Manual*.

2.4.6 Debug Console Window

During development, the user can use procedure calls similar to **printf()** from the downloaded application on the C-Motion Engine to send messages to the PC Development Environment for display in a special console window. These console messages may be useful for checking code progress, displaying internal variables, or for other code development-related purposes.

The default console channel is the PCI bus, however this can be changed using the PRP action **Set** with the **Device** resource specified as Serial, CANbus or Ethernet/UDP.

2.4.7 Downloading and Verifying User Application Code

The C-Motion Engine development system is used to create, compile, and download user application code. The development system can download the file image for the current code project being worked on, or a specific named

file can be downloaded. Downloaded files images end with a “.bin” extension. Only one code image file may be downloaded into the C-Motion Engine at a time. Downloading a new image automatically erases the previous code image.

There are times when it may be useful to read specific characteristics of a code file that has been downloaded into the C-Motion Engine. For example a host controller in a production environment may want to confirm that the host application code version actually loaded on the C-Motion Engine matches the expected production code version. To accomplish this, the PRP action **Get** is used, specifying a resource ID of **CMotionEngine**. Using this command the file name of the downloaded user application code, the checksum of the downloaded file, the date & time of file creation, and the version number of the C-Motion Engine itself (loaded by PMD at the Prodigy/CME factory) can be retrieved.

For complete information on the format and function of these, and other PRP actions, refer to the *Prodigy/CME Programmer's Reference*.

2.4.8 C-Motion Engine Heartbeat LED

The Prodigy/CME PCI card utilizes an LED, locatable using Figure 1-1 on page 15, to provide visual confirmation of C-Motion Engine activity. Two different states can be distinguished, user application code running, and user application code not running.

User application code running means that a file has been downloaded and is actively being executed by the C-Motion Engine. This is indicated by a steady on/off blinking of the LED, once per second.

If no user application code has been downloaded, or if code execution has been halted by the user or for some other reason, the LED changes to a ‘chirp’ indication, with a blinking pattern consisting of very brief on, followed by one second off.

2.5 Communications Functions

The Prodigy/CME PCI card provides four different connection types, PCI bus, Serial, CANbus, and Ethernet. The PCI bus is used primarily to receive and return command messages from the host PC, while the Serial, CANbus, and Ethernet ports are general purpose programmable network ports. Access to the PCI bus controller is handled automatically by the PRP packet processing system and is not generally accessed using low level port commands.

Access to the Serial, CANbus, or Ethernet network communication ports is provided via a peripheral connection. A peripheral is a resource, and is utilized by various PRP actions to send and receive messages to communications connections. Basic access to either the Serial, CANbus, or Ethernet port is accomplished by opening a peripheral with the detailed connection parameters that will be used during communications associated with that peripheral connection, and then sending and receiving messages via that peripheral resource address.

For example to create an Ethernet TCP peripheral connection, the IP Address and port number is provided. If the connection is successfully established, that peripheral is used as the reference for any future communications through that connection.

In the subsequent sections the operational characteristics of the PCI bus, Serial (Serial 1 & Serial 2), CANbus, and Ethernet network communication ports will be detailed. See Section 3.2.1, “Peripheral Connections,” on page 52 for more information on the **Peripheral** resource and how it is used.

2.5.1 PCI Bus

Prodigy/CME PCI cards support the PCI bus interface commonly used in PC-based computers. In the PCI bus scheme, card addressing occurs automatically, but may utilize a card ID for distinguishing cards of identical type. For example if two Prodigy/CME PCI cards are installed in the same PC frame, they can be distinguished by the slot # ID.

The Prodigy/CME PCI card is operated as a PCI bus slave, and can not be operated as a PCI bus master. In typical operation the host PC sends commands to the card using the PRP format, and the card responds, again using the PRP header format to return messages. See Section 3.5.1, “PRP messages over PCI bus,” on page 58 for more information on how the PRP message is transferred on the PCI bus.

To send and receive general format message between the host PC and the C-Motion Engine user application code, the user packet mechanism described in Section 2.4.4, “Sending Messages to/from User Application Code,” on page 37 should be used.

In addition to the standard PRP messages and user packet messages, it is possible for the Magellan Motion Processor to send out asynchronous messages using the PCI bus’ interrupt hardware. These messages are called events, and special C-callable routines are provided to receive events in the host PC or in the user code downloaded to the C-Motion Engine. For more information on processing events see the *Prodigy/CME Programmer Reference*.

2.5.2 Serial 1 & Serial 2

Prodigy/CME PCI Motion Cards provide asynchronous serial communications in either RS232 or RS485 mode. Access to the serial port controller is managed using peripheral connections. See Section 3.2.1, “Peripheral Connections,” on page 52 for more information on creating and using peripheral connections.

In RS232 mode two serial ports are supported, referred to as Serial 1, and Serial 2. While some applications will not need to use two serial ports, the second port may be useful during C-Motion application code debugging, or to communicate with various serial devices connected to the machine. In RS485 mode, a single serial port is supported, referred to as Serial 1. Also in RS485 mode, the serial port may be operated in either half duplex or full duplex mode. Pin #1 of the J21 Serial Connector selects whether RS232 or RS485 communications mode is used. If left open (the default condition), the card operates the serial port in RS232 mode. If closed (tied to ground) the serial port is operated in RS485 mode. Note that a change in the status of this pin takes effect only after a power on or reset of the card.

Both Serial Port 1 and Serial Port 2 can be operated at various communication settings as shown in the following table. All settable serial port parameters can be programmed separately for Serial 1 and Serial 2. For RS232 communications each serial controller only allows one peripheral to be established at a time. For example if a peripheral is opened to establish RS232 communications via Serial 1, another peripheral may be opened to establish RS232 communication via Serial 2. However if a third peripheral is then opened to establish a new connection with Serial 1, the original Serial 1 peripheral will automatically be closed.

Settings & default values for Serial 1 and Serial 2:

| Parameter | Range | Serial 1 Default | Serial 2 Default |
|-------------|-----------------|------------------|------------------|
| Baud rate | 1200 to 460,800 | 57,600 | 115,200 |
| Parity | none, even, odd | none | none |
| # Data bits | 5, 6, 7, 8 | 8 | 8 |
| # stop bits | 1, 2 | 1 | 1 |

To create a serial port peripheral connection, the above parameters are specified in the PRP action **Open** with the resource set to **Device**. Messages to and from the Serial port are transmitted via the **Send** and **Receive** actions sent to the **Peripheral** resource. Whenever a new serial port peripheral is opened its previous function is canceled. By default Serial 1 listens for PRP communications and Serial 2 is the console port.

After a reset or at power-up, the card retrieves default information for the serial ports. To change these default values, the PRP action **Set** is send with a resource of **Device**. Section 2.6.3, “Setting Card Defaults,” on page 44 describes this further.

For detailed information on PRP action formats and function, refer to the *Prodigy/CME Programmer’s Reference*.

2.5.2.1 Connections & associated signals

J17 comprises a special 10-pin dual header connector used to connect one or both serial ports. See Section 1.4, “Accessory Products,” on page 12 and Section 4.2.6, “Serial Connector,” on page 67 for a detailed signal description of the Serial connector.

2.5.3 CANbus Communications

The Prodigy/CME PCI card provides a general purpose CANbus port compatible with the CAN 2.0B standard which may be operated at various communication rates from 10,000 to 1,000,000 bps (bits per second). In addition, each CANbus devices is assigned two CAN identifiers (also called a addresses); one for transmission of messages, and one for reception of messages.

The following table summarizes this information along with the factory defaults for these values:

| Parameter | Range | Default |
|----------------------|-------------------------|-----------|
| Baud rate | 10,000 to 1,000,000 bps | 1,000,000 |
| Host send address | 0 - 0x800 | 0x580 |
| Host receive address | 0 - 0x800 | 0x600 |

To create a CANbus peripheral conversation, the baud rate, send address and receive address are specified in the PRP message body of an **Open** action sent to the **Device** resource. Messages to and from the CANbus port are transmitted via the **Send** and **Receive** actions sent to the **Peripheral** resource.

After a reset or at power-up, the card retrieves default information for the CANbus port. To change these default values, the PRP **Set** action is used with a resource of **Device**. See Section 2.6.3, “Setting Card Defaults,” on page 44 for more information.

For detailed information on PRP action formats and function, refer to the *Prodigy/CME Programmer’s Reference*.

2.5.3.1 Connections & associated signals

J16 is an 8-pin dual header that comprises the CANbus network connection. See Section 4.2.8, “CAN Connectors,” on page 68 for a detailed signal description of the CAN connectors.

2.5.4 Ethernet Communications

The Prodigy/CME PCI cards support two different Ethernet protocols, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP is typically used for secure communications to the card, while UDP is typically used for non-critical applications such as data logging, or for the Pro-Motion console window. See Section 2.4.6, “Debug Console Window,” on page 38 for more information on the C-Motion Engine console window.

By convention, the 32 bit values for IP Address, Net mask, and Gateway are shown in Dotted Quad Notation. In this notation each of the four numbers are separated by dots, and denote a decimal value for each byte of the four byte word



The table below shows the range and default settings for the Ethernet controller of the Prodigy/CME PCI card:

| Parameter | Range | Default |
|---------------------|---------------------------|---------------|
| IP address | 0.0.0.0 – 255.255.255.255 | 192.168.2.2 |
| Net mask | 0.0.0.0 – 255.255.255.255 | 255.255.255.0 |
| Gateway | 0.0.0.0 – 255.255.255.255 | 0.0.0.0 |
| PRP Listen TCP Port | 0 - 65,535 | 40100 |

Each physical hardware device on an Ethernet network is assigned one IP address, however, a given IP address can have multiple ports. This is useful because it allows user application code running on the C-Motion Engine to open up multiple peripheral connections by using different port numbers.

To create an Ethernet/TCP or Ethernet/UDP peripheral conversation, the IP Address and port are specified in the PRP message body of an **Open** command sent to the **Device** resource. To transfer messages via this peripheral connection the PRP actions **Send** and **Receive** sent to the **Peripheral** resource are used.

After a reset or at power-up, the card retrieves default information for the Prodigy/CME PCI Ethernet port. To change these default values, the PRP **Set** command is sent to the **Device** resource. See Section 2.6.3, “Setting Card Defaults,” on page 44 for a description.

For detailed information on PRP action formats and function, refer to the *Prodigy/CME Programmer’s Reference*.

2.5.4.1 TCP Connection Keep-Alive

All prodigy/CME card TCP connections, including the PRP communications port, use a ‘keep-alive’ mechanism to detect whether a connection is still valid.

The keep-alive mechanism is a standard part of the TCP protocol specification, and is useful for preventing the prodigy/CME card from leaving connections open if the host has not properly closed a connection. This may occur, for example, if the host has been physically disconnected, or otherwise stops functioning.

The default keep-alive parameters for the Prodigy/CME cards are:

| Parameter | Range | Default |
|---------------|--------------------|------------|
| idle time | 0 - 65,535 seconds | 60 seconds |
| interval time | not settable | 30 seconds |
| retry count | not settable | 2 |

The *idle time* is the amount of time after the last message on the port that must occur for a ‘keep-alive’ message to be sent. When sent, the keep-alive message requests the host connection to acknowledge that it is still functioning properly. If it provides this acknowledgment, the *idle time* counter is reset to 0. If it does not, a second ‘keep-alive’ message will be sent after the *interval time*, and this will be repeated a total of *retry count* number of times. If the host ultimately does not correctly respond, the Prodigy/CME connection will automatically be closed.

Note that all of these functions are handled automatically by the Prodigy/CME card’s TCP processing system, and should in turn also automatically be handled by the host’s TCP system. No user action is required to initiate or monitor these automatic TCP ‘keep-alive’ messages.

To change the default value of the keep-alive idle time a **Set** command is sent to the **Device** resource. A value of 0 indicates that the keep-alive mechanism should be disabled.

For detailed information on PRP action formats and function, refer to the *Prodigy/CME Programmer’s Reference*.

2.5.4.2 Ethernet status LED

Using Figure 1-1 on page 15 two Ethernet status LEDs are locatable, one amber in color, and one grey in color. These LEDs provide information on the status of the Ethernet link. A solid green LED indicates that a link exists (there is a transceiver connected on the ‘other side’ of the connection, and a blinking green LED means that data is being transmitted. The Amber LED indicates that a 100 Mbps (mega bits per second) network is in use. No Amber LED indicates that the network is at 10 Mbps.

2.5.4.3 Connections & associated signals

The Ethernet connector is a 10-pin dual header and is located at J24.

2.5.5 Cable Connections

The serial, CANbus, and Ethernet connectors of the Prodigy/CME PCI card are dual header style, which are suitable for connection to bracket converters for use in a typical PCI slot frame, however they are not suitable for directly connecting to typical serial, CANbus, and Ethernet networks. PMD provides a number of cable options to accomplish this conversion either during prototyping, or for your production products. The subsequent sections detail typical cable connections for these networks.

2.5.5.1 RSR232 serial cable connections

Serial 1 and Serial 2 may be accessed using PMD's off-the-shelf cable accessories. Cable-4301-01.R has a 10-pin header that plugs into the Prodigy/CME PCI Serial Connector (J17) and converts to a single female DB-9 output with dual serial signals. Cable-4355-01.R is a "Y" cable that plugs into this cable, and converts to two DB-9 connectors suitable for connection to a PC serial port or USB-to-serial converter.

2.5.5.2 CANbus cable connections

The Prodigy/CME's CANbus port may be accessed using PMD's off-the-shelf cable accessories. Cable-4701-01.R has an 8-pin header that plugs into the Prodigy/CME PCI CANbus Connector (J16) and converts to an RJ45 output. This RJ45 then plugs into Adapt-RJ45T-01.R, which is a small splitter that duplicates the CANbus RJ45 connections into two identical RJ45 connectors.

Cable-RJ45-02-R is a standard straight-thru RJ45 cable that plugs into a network socket. If additional nodes will be added then no terminator is used. If this node will be the last node on the CANbus network, TRM-RJ45-02.R should be inserted into one of the two RJ45 sockets.

2.5.5.3 Ethernet cable connections

The Ethernet port may be accessed using PMD's off-the-shelf cable accessories. Cable-4505-01.R has a 10-pin header that plugs into the Prodigy/CME PCI Ethernet Connector (J15) and converts to an RJ45 output. This cable can then be directly plugged into the Ethernet network socket.

2.6 General Card Functions

2.6.1 Connection Check

When sent to the **Device** resource, this PRP action **NOP** performs a basic connection check to the Prodigy/CME PCI card. A return without a PRP error code indicates that the Prodigy/CME PCI card is accessible and processing commands.

2.6.2 Reset

Although a reset occurs automatically during power-up, it is sometimes desirable to reset the Prodigy/CME PCI card explicitly through a user-initiated command or action. This can be done in one of two ways. The first is via the PRP action **Reset** sent to the **Device** resource. The second method is via the external signal **Reset**, located at pin 91 of the GP Connector.

After a card reset occurs the Magellan Motion Processor and the C-Motion Engine modules will be reset, and many of the Prodigy/CME PCI card's output signals will be driven to known states. These are summarized in the following table:

| Signal Name | State |
|-------------|-------|
| AxisOutI-4 | High |

| | |
|----------------|-----------|
| PWMMagIA-4C | Low |
| PWMSignIA-4B | Low |
| DACIA-DAC4B | 0.0 volts |
| DigitalOut0-7 | Low |
| AmpEnableI-4 | Low |
| DAC On/Off | Off |
| Watchdog Timer | Disabled |

In addition, upon a card reset all card default parameters are reloaded. See Section 2.6.3, “Setting Card Defaults,” on page 44 for more information on default values.

2.6.2.1 Connections & associated signals

The reset feature has an external signal input, **Reset**, associated with it. This active low signal is located on pin 91 of the GP Connector. It is pulled up through a 4.7 kOhm resistor to 5 V. In addition, all of the signals in the table above are affected by a card reset.

2.6.3 Setting Card Defaults

There are a number of user-settable parameters that are saved by the card in non-volatile RAM, and that are utilized after a power up, or after a card reset. The following table shows these parameters, and provides the initial factory default values:

| Parameter | Factory default value |
|--------------------------------------|--|
| Ethernet Communications | |
| IP Address | 192.168.2.2 |
| Net Mask | 255.255.255.0 |
| Gateway | 0.0.0.0 |
| PRP Port | 40100 |
| Serial Communications | |
| Serial 1 settings | 57600, no parity, 8 data bits, 1 stop bit |
| Serial 2 settings | 115200, no parity, 8 data bits, 1 stop bit |
| RS485 duplex | Full |
| CANbus Communications | |
| Baud Rate | 1,000,000 |
| Send Address | 0x580 |
| Receive Address | 0x600 |
| Task Control & User Application Code | |
| Auto start (y/n) | Yes |
| Debug/Console channel | PCI bus |

To change the default values used by the card the PRP action **Set** sent to the **Device** resource is used. To read back the current default parameters stored in the card the action **Get** is used. For detailed information on the format of these PRP messages consult the *Prodigy/CME Programmer's Reference*.

2.6.4 Dual-Ported RAM (Trace Buffer)

The Prodigy/CME PCI cards have 64 KBytes of on-card dual-ported memory (DPRAM) which is interfaced to the Magellan Motion Processor as well as directly to the Prodigy/CME PCI card's communications bus. Typically, the dual-ported RAM is used to capture trace data from the Magellan Motion Processor. In this mode, its dual port configuration allows for high bandwidth access via the communications bus using PRP actions without affecting Magellan command traffic. For more information on using the Magellan's trace capability, see the *Magellan Motion Processor User's Guide*.

The Magellan Motion Processor can be programmed to store information into the trace buffer using virtual buffers of various sizes. Thus to correctly read the contents of the dual-ported RAM when used in this mode, you must know the memory organization of the Magellan's trace function. See the *Magellan Motion Processor User's Guide* for details.

Another potential use of this RAM buffer is as a general purpose memory or register-based communications portal. Since the dual-ported RAM is a resource addressable through the PRP system, the originator of the data, as well as the retriever of the data, can be any device on, or external to, the card. While there are several possible ways to utilize the dual-ported RAM in this way, the most common configuration is for the user application code of the C-Motion Engine to collect and store data to the dual-ported RAM buffer, and to have an external host controller read it.

To read or write to the dual-ported RAM using the PRP system, a resource address must first be obtained by sending the **Open** action to the **Memory** resource. This resource address is used for all further access to the RAM. To write data to the dual-ported RAM, the action **Write** is sent to the memory resource. To read the contents of the dual-ported RAM, the action **Read** is used. Note that byte-sized memory operations are not supported to the dual-ported RAM.

The contents of the dual-ported RAM are volatile. They are not saved during power-down of the card.



For complete information on the format and function of these, and other PRP actions, refer to the *Prodigy/CME Programmer's Reference*.

2.6.5 Non-volatile Memory

The Prodigy/CME PCI cards have a general purpose 4,094 byte memory that retains its contents after a card power down or reset. This memory is useful for storing parameters that are set only occasionally, and stay with the card, such as machine calibration information.

Accessing the non-volatile memory is accomplished in the same manner as accessing the dual-ported RAM, except that the NVRAM memory type is specified, instead of the DPRAM memory type. Addresses are specified from 0 to 4,093. When writing to this memory, a typical write takes 30 μ Secs, however under certain circumstances it can take much longer, up to several 100 mSec. Read speed is the same as for other Memory resources, and takes just a few nanoseconds. As for the dual-ported RAM, byte-size memory operations are not supported by the non-volatile memory. The smallest memory unit that can be accessed is 16 bits.

The non-volatile memory can be rewritten a limited number of times. The worst case write limit cycle is 100,000 times for a given memory address, but in typical operation, the limit is much higher. As a general guideline, to avoid erase/write cycle limit problems, the non-volatile RAM should not be used for general purpose scratch RAM, and should only be used to store permanent or semi-permanent parameters.

For complete information on the format and function of these, and other PRP actions, refer to the *Prodigy/CME Programmer's Reference*.

The typical write time to the non-volatile RAM is 30 μ Sec, however it may take as long as several 100 mSec. If other portions of the user application code, or any other PRP-connected device, depends on these values having been written, it is recommended that you ensure that the write operation has completed by adding code that explicitly checks the value, or by waiting a fixed period of time after the NVRAM write operation



2.7 Signal Processing and Hardware Functions

These functions are implemented in hardware and are not directly user-programmable. The following sections are organized into related groups of signals, and provide information which may be helpful when connecting the motion system.

2.7.1 Home, AxisIn, AxisOut, Limits, Hall Sensors

These signals are conditioned by the card, and then input or output directly to the Magellan Motion Processor. The *Magellan Motion Processor User's Guide* explains the functions provided in connection with these various signals. Most of the signals are optional, and are connected depending on the nature of the application.

These signals are named *Home I-4*, *AxisIn I-4*, *AxisOut I-4*, *PosLim I-4* (positive direction limit input), *NegLim I-4* (negative direction limit input), and *HallIA-4C* (12 signals in all).

2.7.1.1 Connections & associated signals

These signals are single-ended digital inputs to the card, with the exception of *AxisOut*, which is a single-ended output. There are no associated connections required for these signals to function properly, however, one or more of the digital grounds must be connected. The input signals are pulled up through 4.7 kOhm resistors to 5V. The default power-up value for all *AxisOut* signals is high.

See Chapter 4, *Electrical Reference*, on page 61 for a complete description of the pinout connections to and from the card.

2.7.2 QuadA, QuadB, Index

These signals provide position feedback to the motion controller which is used to track motor position. For DC brush and brushless DC motors, they are required for proper operation. For microstepping or step (pulse & direction) motors, they are optional.

The encoder-processing circuitry provides a multi-stage digital filter of the *QuadA*, *QuadB*, and *Index* signals for each axis. This provides additional protection against erroneous noise spikes, thus improving reliability and motion integrity. These signals are named *QuadA1+* through *QuadB4-* (16 signals), and *Index1+* through *Index4-* (8 signals).

2.7.2.1 Connections & associated signals

These signals can be connected in one of two ways. Single-ended means that only one wire per signal is used, while differential means two wires encode each signal (labeled + and -). Differential transmission is generally recommended for the highest level of reliability, because it provides greater noise immunity than a single-ended connection scheme.

If single-ended connections are used, only the + signal is connected, and the - signal should be left floating. For example, in connecting to the A quadrature input, *QuadA1+* connects to the signal, and *QuadA1-* remains floating. If differential connections are used, both the + and - signals are used. Differential or single-ended termination must be selected through resistor pack installation. See the table in Section 1.8, "Preparing the Card for Installation," on page 14 for details.

When using the system with differential connections, the polarity of the differential signal can be reversed by swapping the + and - connections. This may be useful for altering the motor and/or encoder direction; however, this same function can also be accomplished through commands to the Prodigy/CME PCI card. See the *Magellan Motion Processor User's Guide* for more information. Associated connections supported by the card are the +5V output signals. These are provided as a convenience, as they are generally connected to a corresponding input on the encoder to power its internal circuitry. As was the case for the digital input signals, one or more of the digital grounds must also be connected.

See Chapter 4, *Electrical Reference*, on page 61 for a complete description of the pinout connections to and from the card.

2.7.3 Analog Input

The *Analog I-8* signals provide general purpose input of up to eight analog signals. The voltages present at these various connections do not directly affect the Prodigy/CME PCI card's behavior. However, they can be read through the Prodigy/CME PCI card, thus providing a convenient way of importing analog signal levels which may be acted upon by the user's application code located on the host PC. These signals are read using the Magellan command **ReadAnalog**. For more information on reading the value of these analog inputs, see the *Magellan Motion Processor User's Guide*. The minimum allowed input voltage is 0.0V, and the maximum allowed input voltage is 3.3V. To determine the numerical value that will be read by the Prodigy/CME PCI card given a specific voltage, the following formula is used:

$$\text{ReadValue} = \text{AnalogVoltage} * 65,536 / 3.3\text{V}$$

Conversely, given a read value, the voltage at the connection is calculated as:

$$\text{AnalogVoltage} = \text{ReadValue} * 3.3\text{V} / 65,536$$

2.7.3.1 Connections & associated signals

For analog voltages to be read correctly, in addition to the analog signal itself, *AnalogGND* (analog ground) must be connected.

2.7.4 Pulse & Direction

For pulse & direction applications, these signals provide a stream of pulse and direction data, and are compatible with a wide variety of off-the-shelf step motor amplifiers. These signals are generated by the Magellan Motion Processor and are named *Pulse I-4* and *Direction I-4*. The default value at power-up and reset for all pulse and direction output signals is: pulse signal is high; direction signal is low.

2.7.4.1 Connections & associated signals

Both single-ended and differential line driver versions of these signals are output from the Prodigy/CME PCI card. There are no associated connections required for these signals to function properly, however, one or more of the digital grounds must be connected. See Chapter 4, *Electrical Reference*, on page 61 for a complete description of the pinout connections to and from the card.

2.7.5 PWM Out

For DC brush, brushless DC or microstepping motors, these signals provide PWM (pulse width modulated) motor command signals when the motor output mode is set to **PWMSignMagnitude** or **PWM5050Magnitude**. The number of signals per axis varies, depending on factors such as the motor type, the number of phases of the motor, and the motor drive method (sign/magnitude or 50/50). See Chapter 4, *Electrical Reference*, on page 61 for complete connection tables for various motor configurations.

These signals are named *PWMMagIA-4C* (12 signals) and *PWMSignIA-4B* (8 signals).

2.7.5.1 Connections & associated signals

These signals are generated by the Magellan Motion Processor. There are no associated connections required for these signals to function properly, however, one or more of the digital grounds must be connected. See Chapter 4, *Electrical Reference*, on page 61 for a complete description of the pinout connections to and from the card.

2.7.6 DAC Out

For DC brush, brushless DC or microstepping motors, this is the analog motor command when the motor output mode is set to DAC Offset Binary (digital-to-analog converter). These signals are named *DAC1A - DAC4B* (8 signals), and vary between -10V and +10V. The number of signals per axis depends upon the motor type. See Section 4.5, “Environmental and Electrical Ratings,” on page 76 and the *Magellan Motion Processor User’s Guide* for more information.

2.7.6.1 Connections & associated signals

For analog voltages to be output correctly, *AGND* (motor command ground) must be connected. See Chapter 4, *Electrical Reference*, on page 61 for a complete description of the pinout connections to and from the card.

2.8 Software Libraries

PMD provides software libraries in C/C++ as well as Visual Basic to access all of the functions provided by the Prodigy/CME PCI cards. In addition, a special library of commands is used to access Magellan Motion Processor functions called VB-Motion, and C-Motion.

The *Prodigy/CME Programmer’s Reference*, in addition to providing detailed format and function descriptions of the PRP messages associated with the Prodigy/CME PCI cards, also provides the equivalent C and Visual Basic library calls. There is often, but not always, a direct correspondence between PRP messages and C language software commands as they will be used for user application code to be executed in the host controller, or in the C-Motion Engine.

2.8.1 C Language Code Running on the C-Motion Engine

One of the most powerful features of the software libraries provided for the Prodigy/CME PCI card is that the code sequence used to accomplish a specific function such as accessing the dual-ported RAM, or sending commands to the Magellan Motion Processor, does not change whether the code is compiled for execution on the host controller, or for execution on the C-Motion Engine.

This allows motion control applications developed using C-Motion on a host computer to be easily ported to the Prodigy/CME PCI card. The C-Motion Engine provides a C programming environment for motor control without requiring the time-consuming development of the entire embedded framework. Only the part unique to a specific motor control application must be provided.

For a complete description of the C-language commands supported by the Prodigy/CME PCI card see the *Prodigy/CME Programmer’s Reference*. For a complete description of Magellan C-Motion commands, see the *Magellan Motion Processor Programmer’s Command Reference*. For a detailed description of how to develop, download, and monitor user application code for the C-Motion Engine see the *C-Motion Development Tools Manual*.

3. Accessing Card Resources

In This Chapter

- ▶ Resource Addressing
- ▶ Accessing the Communications Ports
- ▶ Accessing On-Card Resources
- ▶ Accessing Magellan-Attached Devices
- ▶ PRP Communication Formats

3.1 Resource Addressing

The Prodigy/CME PCI card's various features and capabilities are organized into groups called 'resources.' For example, to communicate with the Prodigy/CME PCI card's Magellan Motion Processor, the host controller sends a command to the **MotionProcessor** resource.

To address different types of resources, the Prodigy/CME PCI cards provides a general purpose resource addressing scheme called the *PMD Resource access Protocol* (PRP). PRP is a sophisticated general purpose addressing scheme that allows:

- one or more host controllers to communicate with Prodigy/CME PCI cards
- Prodigy/CME PCI cards to communicate with each other
- Prodigy/CME PCI cards to communicate with other PMD products such as IONs and non-/CME Prodigy cards
- Prodigy/CME PCI cards to communicate with user-designed hardware, or other off-the-shelf hardware.

3.1.1 PMD Resource Access Protocol (PRP)

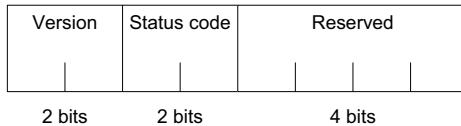
The core of the PMD Resource access Protocol is a header that accompanies all PRP communications. Figure 3-1 on page 50 shows the format of the resource access protocol header. The PRP header is a single 16-bit word divided into five fields. Normally, the PRP header is immediately followed by a message body, but there are certain communications that do not require a message body. PRP headers are used in both the outgoing, and the response packet. The returned PRP header is 1 byte in length, consisting of the version field, the status code field, and 4 reserved bits.

**Figure 3-1:
Outgoing and
Returning PRP
header formats**

Outgoing PRP Header



Return PRP Header



The majority of communications to/from the host controller and the Prodigy/CME PCI card use the PRP header. Exceptions are 'low-level' communications sent or received directly from the Serial, CANbus, or Ethernet ports. See Section 3.2, "Accessing the Communications Ports," on page 51 for more information.

PRP header field descriptions:

Version - This two bit field encodes the version of PRP being used. The value of this field for the Prodigy/CME PCI card should always be 1 (binary 01) unless documentation included with your Prodigy/CME PCI card indicates otherwise.

Status code - For PRP commands being sent out, this 2-bit field should contain the value 2. When received, a return value of 0 indicates that this message is a normal response to an outgoing PRP command, a return value of 1 indicates that an error occurred during PRP command processing, and a value of 3 indicates that this is an asynchronous event message originated by the Prodigy/CME card or by a device attached to the card. Each of these different response status codes may have information loaded in the PRP message body. See the *Prodigy/CME Programmer's Reference* for more information.

Action - This 4-bit field contains an action identifier that is used to process PRP messages. See Section 3.1.3, "PRP Actions," on page 51 for a summary of the PRP actions supported by the Prodigy/CME PCI card. This field is not used in the return PRP header.

Resource - This 3-bit field encodes the specific resource being addressed. See the table in Section 3.1.2, "PRP Resources," on page 51 for the complete resource map of the Prodigy/CME PCI card. This field is not used in the return PRP header.

Address - This 5-bit field encodes the address of the particular resource being communicated to. Fixed addresses allow on-card resources to be addressed. See the table in Section 3.1.2, "PRP Resources," on page 51 for a resource map of these addresses. Automatically assigned addresses are used to access attached devices, and are also used to create peripheral connections, which are communication 'conversations' between the Prodigy/CME PCI card and another device. This field is not used in the return PRP header.

The following sections provide general information on the PRP system. For a detailed description of the PRP header, resources, and supported actions, see the *Prodigy/CME Programmer's Reference*.

3.1.2 PRP Resources

The Prodigy/CME PCI card supports five different resource types, each of which is identified by a specific Resource ID. In addition each available resource has an Address. The following table summarizes these Resource IDs and Addresses for the on-card Prodigy/CME PCI resources:

| Resource Name | Resource ID | Address | Comments |
|-----------------|-------------|---------|---|
| Device | 0 | 0 | The Device resource indicates a Prodigy/CME PCI card. |
| CMotionEngine | 1 | 0 | The CMotionEngine resource indicates a C-Motion Engine. |
| MotionProcessor | 2 | 0 | The MotionProcessor resource indicates a Magellan Motion Processor. |
| Memory | 3 | 0 | The Memory resource indicates a dual-ported RAM (random access memory). |
| Peripheral | 4 | 0 | The Peripheral resource indicates a communications connection. A peripheral address of 0 indicates a 'null' peripheral. See Section 3.2.1, "Peripheral Connections," on page 52 for more information. |

3.1.3 PRP Actions

The Prodigy/CME PCI card supports ten different actions, each of which is identified by a specific Action ID. In addition, many actions have sub-actions associated with them that are loaded into the PRP message body. For more information on actions, sub-actions, and the exact format by which the message body should be loaded, see the *Prodigy/CME Programmer's Reference*. The following table summarizes the Action IDs for the Prodigy/CME PCI cards:

| Name | Action ID | Used by Resource | Comments |
|---------|-----------|-------------------------------------|--|
| NOP | 0 | All | The No Operation (NOP) command is used to verify connection to a given resource. |
| Reset | 1 | Device, MotionProcessor | Resets the specified resource. |
| Command | 2 | MotionProcessor, CMotionEngine | Sends a command to the specified resource. |
| Open | 3 | Peripheral, Device | Creates a new addressable resource. Resource addresses created using the Open action are not fixed, they are assigned automatically at the time a connection is requested. |
| Close | 4 | Peripheral, Device, MotionProcessor | Closes a resource created by Open, freeing the automatically-assigned address. Used when a resource is no longer needed. |
| Send | 5 | Peripheral, C-Motion Engine | Sends a message to the specified resource. |
| Receive | 6 | Peripheral, C-Motion Engine | Receives a message from the specified resource |
| Write | 7 | Memory | Writes a data word to a memory resource |
| Read | 8 | Memory | Reads a data word from a memory resource |
| Set | 9 | Device, CMotionEngine | Sets parameters for a specified resource |
| Get | 10 | Device, CMotionEngine | Get parameters for a specified resource |

3.2 Accessing the Communications Ports

Figure 2-1 on page 26 shows an internal block diagram of the Prodigy/CME PCI card. Routing of communications within the card is accomplished via a high-speed communications bus. This bus interconnects resources within the card including the Magellan Motion Processor, the C-Motion Engine, and the Dual-ported RAM. External communications use this same bus to access the serial, CANbus, or Ethernet communication ports. This means that

commands to card resources may originate from outside the card via the serial, CANbus, or Ethernet ports, or internally, via the C-Motion Engine.

A powerful feature of the Prodigy/CME PCI card's communication bus is that it can process multiple communication requests simultaneously. For example the C-Motion Engine can be used to send motion commands to the on-card Magellan Motion Processor, while an external host controller can communicate to the same Magellan via the PCI bus port to monitor progress of the moves, and display results on a remote capture/analysis program such as PMD's Pro-Motion software.



While it is allowed to have more than one communications channel send motion commands to the Prodigy/CME PCI card's Magellan Motion Processor and another channel send monitoring-related requests, it is not advisable to have multiple communication channels send motion commands to the Magellan at the same time. This may result in unexpected or unsafe motion.

3.2.1 Peripheral Connections

The Prodigy/CME PCI card provides three different programmable network connection types, Serial, CANbus, and Ethernet. These communication resources are represented in PRP by a construct called a peripheral connection. A peripheral is a resource (resource ID: 4), and is provided, or utilized, by various PRP actions to send and receive messages to network connections.

Obtaining access to either the Serial, CANbus, or Ethernet port is accomplished via the PRP **Open** action. This action opens a peripheral by specifying a sub-action of **OpenSerial**, **OpenCAN**, or **OpenTCP** or **OpenUDP**, as well as the detailed connection parameters that will be used during communications with that specific peripheral connection. Each new open peripheral connection receives an automatically assigned address. The application code that requests the new peripheral connection must record that provided address for future use, and it is this address that is used within the PRP message to reference the newly created peripheral connection.



Automatically assigned addresses generally increment by one each time they are assigned, however this should not be assumed. The exact return address value should be stored and only that value should be used to reference that specific peripheral. Each **Peripheral** Open action is specific to the type of connection being requested. For example there is an 'open peripheral' command for CANbus communications, one for Serial communications, and so on.

To send a message to the new peripheral connection, a timeout parameter and the desired message is loaded into the PRP message body and the **Send** action is specified. To retrieve messages from that peripheral connection the **Receive** action is used. The **Receive** action requires that an amount of time (the communication timeout) be loaded into the message body. If a message is not received in the specified amount of time an error is returned.

Example

Figure 3-2 on page 53 shows a network configuration. The host controller needs to initiate, send and receive a message to/from a specific device connected on the CANbus port.

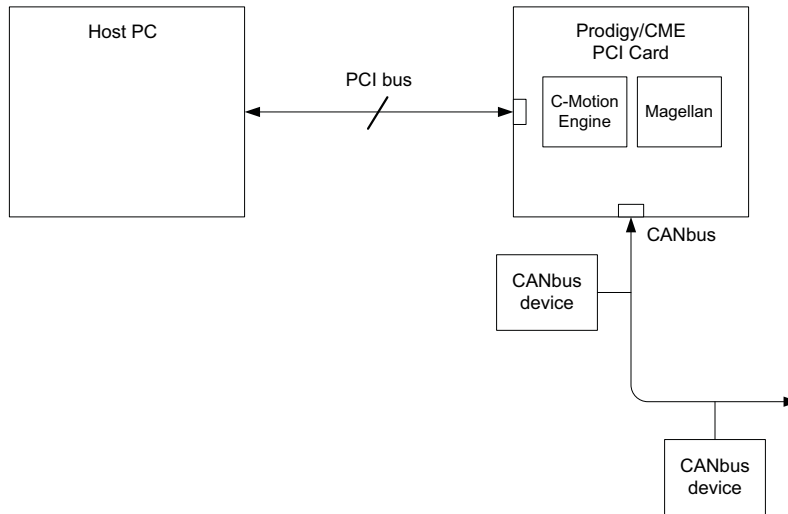


Figure 3-2:
Example
Network
Configuration

This sequence will be accomplished in three steps. PRP messages will be assembled that comprise the **Open** action, and then in turn the **Send** and **Receive** actions. As for all PRP messages, the resource ID, the address, and the Action ID must be specified. In this example, to start the sequence the Resource is a 4 specifying a **Peripheral**, the Action ID for **Open** is 2, and the PRP address will be loaded with a 0. The message body is loaded with a code for the sub-action **OpenCAN** as well as the detailed CANbus communications parameters to establish the connection such as baud rate, send address, and receive address. For the exact message body of this and all PRP actions refer to the *Prodigy/CME Programmer's Reference*.

To simplify the nomenclature for PRP messages, a shorthand will be used which contains the mnemonic 'PRP,' the Resource ID, the string 'Addr,' the resource address, and the PRP action. Any additional information pertaining to the error code or message body will be contained in the comment for the message.



```

NewPeriphID = PRP Device, Addr 0, Open // Send a request to open a connection to the Prodigy/CME
// PCI card being addressed via the CANbus port.
// The Message body contains the 'CANbus' as well as the
// detailed CANbus parameters for that connection.
// The Address of the new Peripheral connection is contained in
// the body of the return message.

PRP Peripheral, Addr NewPeriphID, Send // Send a message, contained in the message body, to the CAN
// bus connection created using the Open command. Note that
// the Peripheral address provided in the previous command is
// used to identify which Peripheral connection the message
// should be sent to.

PRP Peripheral, Addr NewPeriphID, Receive // Receive a message. The string will be contained in the message
// body of the return message, from the CANbus Peripheral
// connection.
    
```

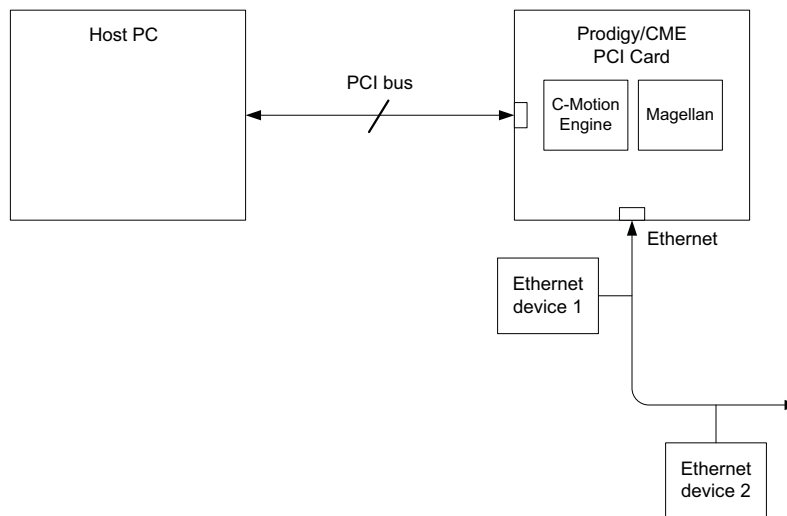
Sending a message to the **Peripheral** resource at address 0 has special meaning as the ‘null’ peripheral. This peripheral address indicates ‘no peripheral,’ and may be useful in certain situations for disabling console messages, or disabling the Magellan’s event output mechanism.

3.2.2 Managing Automatically Assigned Addresses

Although most network configurations are created once and left in place while the machine is operating, there may be circumstances where peripheral connections are made, and are then no longer needed. If this is the case, these connections should be ‘closed,’ thereby freeing that automatically assigned **Peripheral** address. This is accomplished via the PRP action **Close**.

Figure 3-3 shows a Prodigy/CME PCI card being used as a production Ethernet device exerciser.

Figure 3-3:
Example
Prodigy/CME
PCI
Architecture
with Ethernet
Device Testers



In this example the Prodigy/CME card sends messages to two different Ethernet/TCP addresses and then closes those peripheral connections to allow the process to be repeated indefinitely.

```

PeriphID1 = PRP Device, Addr 0, Open           // Open a peripheral connection to Ethernet device 1
PRP Peripheral, Addr PeriphID1, Send          // Send a test string to Ethernet device 1
PeriphID2 = PRP Device, Addr 0, Open           // Open a peripheral connection to Ethernet device 2
PRP Peripheral, Addr PeriphID2, Send          // Send a test string to Ethernet device 2
PRP Peripheral, PeriphID2, Close              // Close peripheral connection for Ethernet device 2
PRP Peripheral, PeriphID1, Close              // Close peripheral connection for Ethernet device 1

```

For complete information on the format and function of these, and other PRP commands, refer to the *Prodigy/CME Programmer's Reference*.

3.3 Accessing On-Card Resources

The most common use of the Prodigy/CME PCI card is to process commands to resources located on the card itself. Typically this means communicating with the on-card Magellan Motion Processor, but it can also mean communicating with other Prodigy/CME PCI on-card resources such as the dual-ported RAM. Figure 3-4 shows this configuration.

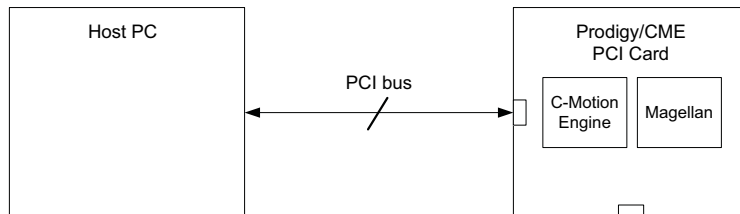


Figure 3-4:
Host Controller
& On-card
Resources

Accessing on-card resources is straightforward using the PRP system. The on-card resource and the address are looked up using the table in Section 3.1.2, “PRP Resources,” on page 51 and then a PRP message corresponding to the desired action is sent using those on-card resource and address values. The following example illustrates:

Example 1: A Host Controller wants to set the position of Axis 3 of the Prodigy/CME PCI’s on-card Magellan Motion Processor to a value of 0x123456.

From the table in Section 3.1.2, “PRP Resources,” on page 51 to communicate with the on-card Magellan Motion Processor, a PRP message is sent to Resource ID 2 (corresponding to the **MotionProcessor** resource), to address 0 (corresponding to the Prodigy/CME PCI’s on-card Magellan), with an action ID of 2 (corresponding to the **Command** action). The message body is loaded with the Magellan packet corresponding to “Set Position, #3 0x123456,” which is the 3-word sequence 0x210, 0x0012, 0x3456.

```
PRP MotionProcessor, Addr 0, Command // Send a command to the on-card Magellan Motion Processor. Message
// body contains Magellan Command “SetPosition #3, 0x123456”
```

Upon processing of this command, the host would receive a PRP message back. A zero in the status field would indicate that no error occurred. If this is the case the message body will be empty. If an error did occur, then the PRP status field would contain a 1, and the message body would contain the specific error code that occurred.

Example 2: The Host Controller reads the 32-bit word value of address 0x100 from the Prodigy/CME PCI’s dual-ported RAM

Here is the command that would be sent:

```
PRP Memory, Addr 0, Read // Send a ‘Read’ action (ID: 7) to the PRP Memory Resource (ID: 3),
// address 0 (the address of the Prodigy/CME PCI on-card dual-
// ported RAM). The PRP message body contains a sub-action of 0,
// specifying a 32 bit word read, followed by 0x100 the address of the
// 32 bit word to read from the dual-ported RAM.
```

Upon successfully processing this command, the host would receive the 32-bit contents of memory location 0x100 in the message body.

Note that the PRP message to send a Magellan command packet did not use a sub-action code in the message body, while the **Read** command sent to the dual-ported RAM did. Whether or not a sub-action is required, and what the codes are for various sub-actions, is action-specific, and sometimes resource-specific. *The Prodigy/CME Programmer’s Reference* provides exact message body information for each PRP action and (if applicable) sub-action.

Note also that in this discussion of sending PRP messages to the Prodigy/CME PCI card, the specific communications channel to the card (PCI bus, serial, CANbus, Ethernet) was not discussed. That is because the PRP header and message body are the same regardless of how transmission occurs. Depending on the network type used, those varying communication links will send, receive, and process errors for packet communications in different ways, but from the ‘perspective’ of the PRP system, those message transmission details are handled automatically. This is a very powerful characteristic of the PRP system, and we will see additional examples of similar ‘access virtualization’ as we discuss more advanced network topologies.



To actually send a PRP command over a PCI bus, serial, CANbus, or Ethernet network, specific protocols must be observed. See Section 3.5, “PRP Communication Formats,” on page 58 for this network-specific (serial, CANbus, or Ethernet) information.

3.4 Accessing Magellan-Attached Devices

Section 3.2.1, “Peripheral Connections,” on page 52 provided information on how general purpose messages can be sent to, or received from, any of the Prodigy/CME PCI card’s network ports. This level of low-level access can be useful to communicate with a wide variety of custom-created or off-the-shelf products that are capable of communicating on that bus.

If the attached device is a PMD device however, such as an ION or non-/CME Prodigy PCI card (PR825xx20 family and PR925xx20 family) then it is possible to integrate these devices into the PRP access network so that they can be communicated to without using low-level peripheral send and receive commands.



IONs and non-/CME Prodigy PCI cards are referred to as ‘Magellan-Attached’ devices because the network directly connects to the Magellan Motion Processor, and utilizes the Magellan’s own communication protocols for receiving commands and returning data along the communication link.

Figure 3-5 on page 57 provides an example of a setup where a Prodigy/CME PCI card is connected via Ethernet to a host controller, and where a second network of CANbus-connected IONs is attached to the Prodigy/CME PCI card’s CANbus port.

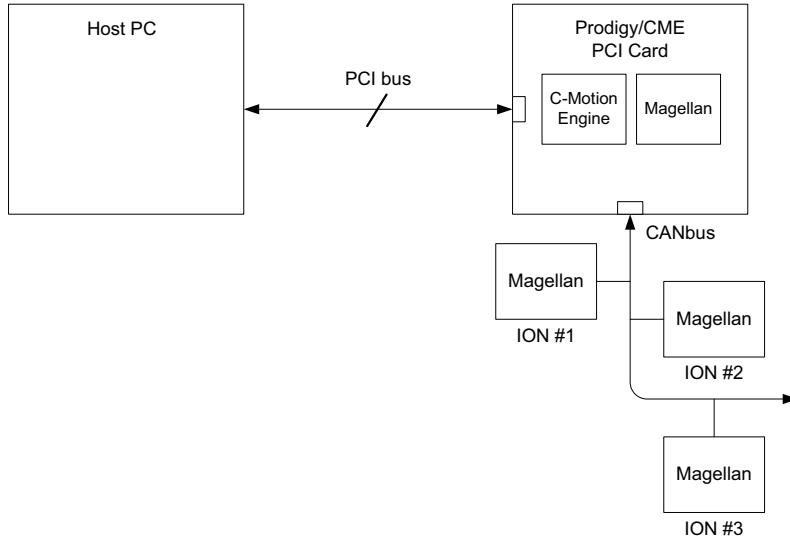


Figure 3-5:
Host Controller
& Magellan-
attached
Devices

In this ‘bridge’ configuration, the host controller (or the Prodigy/CME PCI card’s C-Motion Engine) can access the CANbus-connected IONs very similarly to the way in which the on-card Magellan is addressed. To accomplish this however, a different method is used to create access to the Magellan compared to the method described for on-card Magellan access in (see Section 3.3, “Accessing On-Card Resources,” on page 55 for details).

Rather than sending PRP messages to the on-card **MotionProcessor** resource, first a raw peripheral connection is opened, and then the **Open** action with a sub-action of **MotionProcessor** is used to open a new **MotionProcessor** resource. The following PRP code sequence illustrates:

```

NewPeriphID = PRP Device, Addr 0, Open      // Open a CANbus peripheral connection to connect to ION #1
NewMtnProcID = PRP Peripheral, Addr NewPeriphID, Open
// Use the opened peripheral connection to create a new
// MotionProcessor resource using the Open action with
// sub-action specifying MotionProcessor. A new MotionProcessor
// resource address is loaded into the message body.
PRP MotionProcessor, Addr NewMtnProcID, Command
// Send Magellan command packet to ION #1's Magellan Motion
// Processor using the standard method of communicating with
// MotionProcessor resources.
    
```

The action sequence to communicate with ION #2 and subsequent devices is similar, except the unique CANbus parameters to address that particular ION should be used, and the unique automatically assigned PeriphID and MagellanID values should be recorded and used for resources addressing.

Note that as the command sequence above shows, after the new **MotionProcessor** resource addresses are created using the **Open** action, subsequent Magellan commands to the CANbus IONs are identical in format to commands to the on-card Magellan. This illustrates a very powerful feature of the PRP system which is that it allows resources to be addressed transparently by the host controller (or C-Motion Engine module), making it easy to create and access networks of PMD products.

Note also that the **Open** action can be used with different resource types. In Section 3.2.1, “Peripheral Connections,” on page 52 it was used with a resource type of **Device** to open a peripheral connection. In the above example it was used with a resource type of **Peripheral** to open a connection to a Magellan Motion Processor.

3.5 PRP Communication Formats

The following sections discuss how PRP messages should be formatted on the Serial, CANbus, and Ethernet networks so that they can be properly interpreted by the Prodigy/CME PCI card.

3.5.1 PRP messages over PCI bus

By default, the Prodigy/CME PCI card is set up to receive PRP command messages from a host controller on the PCI bus. This section describes how this transfer occurs.

PCI (Peripheral Component Interconnect) bus utilizes up to six address spaces, indexed via base address registers (often called BARs), for data transfer to/from cards installed on the bus. An important feature of the PCI bus is that the actual base addresses (BARs) are assigned automatically, so that unlike the older ISA bus, they need not be entered manually by the user or 'keyed' into the card using dip-switches or jumpers.

Prodigy/CME cards use two of these six address spaces, BAR 2, and BAR 5. BAR 2 is 32 16-bit words in length, and is used to synchronize communications between the host PC and the Prodigy/CME card. BAR 5 is 1,024 16-bit words in length, and is used to hold the PRP/PCI message packet. Of the four other address spaces, two (BAR 0 and BAR 1) are used by the PLX driver that is utilized in the Prodigy/CME cards, and two (BAR 3 and BAR 4) are unused.

The following table shows the registers used by the BAR 2 and BAR 5 buffers in the PRP system:

| Action | Function |
|--------------|--|
| BAR 2 + 0x10 | The low bit of this one word (16 bit) register is set high (1) by the PC to indicate to the Prodigy/CME card that a PRP message is ready to be read. |
| BAR 2 + 0x12 | The low bit of this one word (16 bit) register is set high (1) by the Prodigy/CME card when a PRP responses packet is ready to be read by the host PC. |
| BAR 5 + 0x00 | This 1,024 word buffer holds the PRP message for both outgoing and incoming PRP messages using the format shown in Figure 3-6. |

3.5.1.1 PRP/PCI message format

Figure 3-6 shows the format of the PRP/PCI packet that should be loaded into, or read from, BAR 5. The first four bytes contain a length field which holds the length, in bytes, of the PRP message (PRP header and message body). The PRP header and message body is loaded beginning directly after the length field.

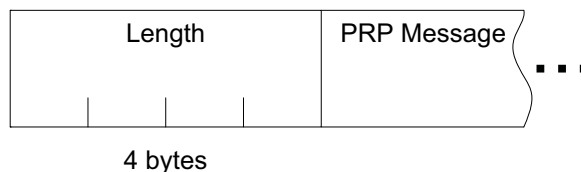


Figure 3-6:
PRP/PCI Packet
Format

Return PRP/PCI messages are formatted in the same way.

3.5.1.2 PCI packet processing

An error-free PRP/PCI communication sequence from the host PC to the Prodigy/CME PCI card consists of the following sequence:

- The host PC writes a PRP/PCI packet using the format detailed in Figure 3-6 to the buffer beginning at BAR 5.
- The host PC sets the low bit of BAR 2 + 0x10 high.

- The host PC polls the low bit of BAR 2 + 0x12. If it is low, no return message is ready, and the host should repeat this step.
- If it is high, then the host reads the return PRP/PCI message from the buffer beginning at BAR 5.
- The host PC sets the low bit of BAR 2 + 0x12 to 0.
- Now that the data has been read, other processing of the loaded PRP message can occur per the user's application.

The return message must be received within a fixed amount of time determined by the host PC. Correctly setting this 'timeout window' may depend on factors such as the speed of the PC's CPU, but 100 mSec is a typical safe value.

For additional information on PRP/PCI packet processing, see the *Prodigy/CME Programmer's Reference*.

3.5.2 PRP messages over Serial

The Prodigy/CME PCI card can also receive PRP command messages from a host controller on Serial port 1 or Serial port 2. This section describes the format of these packets, which transfer PRP messages over a serial protocol.

Figure 3-7 shows the Serial packet protocol that must be used for all PRP messages. This serial header is 32 bits, and is broken into three fields as follows:

Address: the first byte is an address field, used only with RS485 communications. For RS232 communications, this byte is not included in the packet.

Checksum: The second byte is a simple 8-bit sum checksum of all bytes in the packet payload (excluding the serial header, i.e., the address, checksum, and length fields). From Figure 3-7 this means only the PRP header and message body contribute to this checksum value

Length: The third byte is an 8 bit length field. The length indicates the number of bytes in the packet payload. The packet payload is defined as everything in the serial packet after the length field. That is, the 2 byte PRP header plus however many bytes are contained in the PRP message body. For example if the PRP message body is 6 bytes in size, the value filled into the length field should be 8 (2 bytes for PRP header + 6 bytes for the PRP message body).

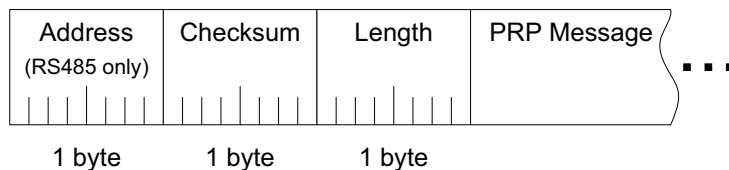


Figure 3-7:
PRP Message
over Serial
Format

Return PRP messages should be formatted in the same way.

3.5.2.1 Serial packet processing

An error-free Serial/PRP communication sequence from the host controller to the Prodigy/CME PCI card consists of a full outgoing packet transmission with the correct checksum and specified number of bytes received by the Prodigy/CME PCI card, and a full packet response with correct checksum and length received at the host controller. The return message must be received within a fixed amount of time determined by the host controller. Correctly setting this 'timeout window' may depend on factors such as baud rate, but 100 mSec is a typical safe value.

If the host controller receives a response packet with an incorrect checksum, or does not receive the complete response (communications timeout), then the original message should be resent.

If the Prodigy/CME PCI card receives a packet with an incorrect checksum, then a packet with an error code indicating this is returned to the host controller.

If the Prodigy/CME PCI card does not receive the specified number of bytes within 100mSecs (the Prodigy/CME PCI timeout value) of beginning of packet reception, the incoming message is ignored, and no message is sent to the host controller.

3.5.3 PRP messages over CANbus

If the Prodigy/CME PCI card is set up to process PRP command messages from a host controller over CANbus, a specific format for the packets must be followed. This section describes the format of how PRP messages are carried over CANbus.

Since native CANbus communications can not be larger than 8 bytes, hosting the PRP system, which can support shorter as well as longer messages, requires additional layers to manage data segmentation and desegmentation. The protocol that is used by the Prodigy/CME PCI cards to accomplish this is very similar to the SDO (Service Data Object) protocol of the CANopen standard.

The details of this protocol are extensive enough that they are not described here, but are available in the *Prodigy/CME Programmer's Reference*.

3.5.3.1 CANbus packet processing

Unlike the serial protocols, the SDO based CANbus protocol has a robust error checking and retransmission mechanism built in that corrects for garbled or otherwise unusable transmissions.

Nevertheless, if a host controller does not receive the complete response packet within a specific time window (communications timeout), then the original message should be resent.

3.5.4 PRP messages over Ethernet

The existence of ports, and the broad range of packet lengths that are supported with the Ethernet protocol, makes sending PRP messages very simple. For both sent and received messages the PRP message is simply loaded as the 'payload' of the Ethernet message. The only convention that must be observed is that the host controller's destination TCP port must be equal to the Prodigy/CME PCI card's TCP default value set using the **SetDefault** command. Note that the UDP protocol may not be used for PRP communications to/from the Prodigy/CME PCI card.

3.5.4.1 Ethernet packet processing

Unlike the serial protocols, Ethernet TCP packets have a robust error checking and retransmission mechanism built in that corrects for garbled or otherwise unusable transmissions.

4. Electrical Reference

In This Chapter

- ▶ User-Settable Components
- ▶ Connectors
- ▶ Connections Summary—Motor Amplifiers
- ▶ Cables
- ▶ Environmental and Electrical Ratings
- ▶ Mechanical Dimensions
- ▶ User I/O memory Map

4.1 User-Settable Components

Figure 4-1 illustrates the locations of the principal components of the Prodigy/CME PCI cards. The user-settable components of the card are listed in the following table:

| Component | Function |
|----------------------------------|--|
| Resistor packs RS1, RS2, and RS3 | Sets the encoder termination. |
| SW1 DIP switch | Sets the GP Connector Motor Output Configuration |

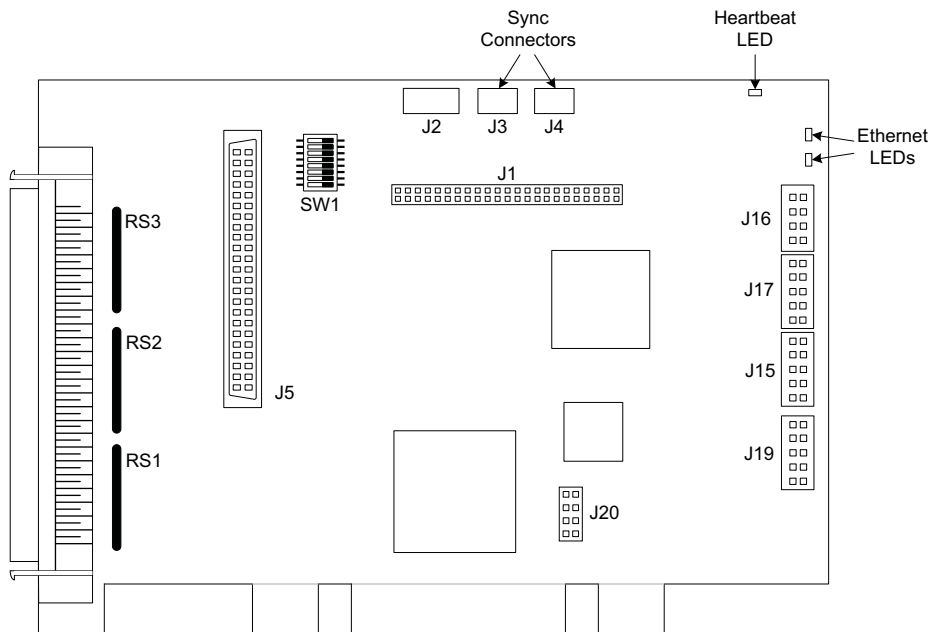


Figure 4-1:
Components and layout,
front of card

4.1.1 Encoder Connections and Resistor Packs

Encoder inputs may be connected differentially, with two wires for *QuadA*, *QuadB*, and *Index* signals, or with just one wire per signal. If differential connections are being employed, resistor packs RS1, RS2, and RS3 should remain installed. If single-ended encoders are used, remove all three resistor packs, and connect encoder signals to the positive encoder input only. The negative input may remain unconnected.

The following table shows the relationship between the encoder input mode and resistor packs:

| Item | Setting | Description |
|---------------------------------|---|---|
| Resistor packs RS1, RS2, RS3 | Installed; this is the default setting of resistor packs RS1 - RS3. | If differential connections are being used, leave the resistor packs installed. |
| | Removed | If single-ended encoder connections are being used, remove the resistor packs. |

Encoder connections are detailed in the following tables. All connections are made through the J6 GP Connector.

Encoder connections when using differential encoder input:

| Signal | J6 (GP Connector) Pin Connections | | | |
|---------|-----------------------------------|--------|--------|--------|
| | Axis 1 | Axis 2 | Axis 3 | Axis 4 |
| QuadAn+ | J6-1 | J6-19 | J6-51 | J6-69 |
| QuadAn- | J6-2 | J6-20 | J6-52 | J6-70 |
| QuadBn+ | J6-3 | J6-21 | J6-53 | J6-71 |
| QuadBn- | J6-4 | J6-22 | J6-54 | J6-72 |
| Indexn+ | J6-5 | J6-23 | J6-55 | J6-73 |
| Indexn- | J6-6 | J6-24 | J6-56 | J6-74 |
| Vcc | J6-7 | J6-25 | J6-57 | J6-75 |
| GND | J6-8 | J6-26 | J6-58 | J6-76 |

Encoder connections when using single-ended encoder input:

| Signal | J6 (GP Connector) Pin Connections | | | |
|--------|-----------------------------------|--------|--------|--------|
| | Axis 1 | Axis 2 | Axis 3 | Axis 4 |
| QuadAn | J6-1 | J6-19 | J6-51 | J6-69 |
| QuadBn | J6-3 | J6-21 | J6-53 | J6-71 |
| Indexn | J6-5 | J6-23 | J6-55 | J6-73 |
| Vcc | J6-7 | J6-25 | J6-57 | J6-75 |
| GND | J6-8 | J6-26 | J6-58 | J6-76 |

4.1.2 Motor Output Configuration

There are two configurations for setting up the motor output pins on the Prodigy/CME PCI GP Connector: These motor output pins can be set up for use with DC Brush, Brushless DC, and microstepping motors, or the motor output pins can be set for pulse & direction motors. The default setting is for all motors to be set for DC Brush, Brushless DC, and microstepping.

The switch bank SW1 controls these settings, which apply on a per-axis basis. The following table describes the switch setting for each output type. Figure 4-2 on page 63 provides a diagram of SW1. See Figure 4-1 on page 61 for SW1 location on the Prodigy/CME PCI card. In the following table, the individual switches are numbered from left to right.

| Axis | SW1 Set for Servo Motor | SW1 Set for Pulse & Direction |
|------|-------------------------|-------------------------------|
| 1 | 1 on, 5 off | 1 off, 5 on |
| 2 | 2 on, 6 off | 2 off, 6 on |
| 3 | 3 on, 7 off | 3 off, 7 on |
| 4 | 4 on, 8 off | 4 off, 8 on |

Power to the card should be turned off when changing the state of the SW1 switches, and care should be taken not to set both switch pairs on (for example both 1 and 5 on, or both 2 and 6 on etc.) or damage may occur to the board.

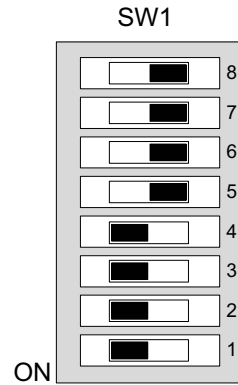


Figure 4-2:
Switch
Settings by
Output Type

4.2 Connectors

There are 15 user-accessible connectors on the Prodigy/CME PCI card. See Figure 4-1 on page 61 for the specific locations of the connectors on the card. The connectors and their functions are outlined in the following table:

| Connector Name | Connector # | Functionality |
|----------------|-------------|---|
| GP | J6 | Provides various motion connections to external amplifiers and motor elements. |
| Option | J5 | Used with brushless DC and microstepping motors, the Option Connector connector provides additional connections to external amplifier and motor components. |
| Serial | J17 | Dual serial port for serial RS232 or RS485 connections. |
| Sync I/O (x2) | J3, J4 | Synchronizes additional cards within the same motion system. |
| CAN | J16 | Provides connection to a CAN 2.0B network. |
| Ethernet | J15 | Provides connection to an Ethernet network |
| Extension | J1 | Allows the DC-1000 card to be installed on the Prodigy/CME PCI card. |

4.2.1 GP Connector

The GP Connector (J6 in Figure 4-1 on page 61) connects to various motion peripherals such as encoders, amplifiers, etc. The GP Connector is comprised of a single 100-pin high density connector, which is connected to with Cable-1003-01.R. See Section 4.2.11, “Connector Parts Reference,” on page 70 for detailed information on Prodigy/CME PCI cable accessories.

4.2.2 GP Connector Using DC Brush Motors

| Pin | Connection | Description | Pin | Connection | Description |
|-----------|-------------|--|-----|-------------|--|
| J6 | | | | | |
| 1 | QuadA1+ | Quadrature A+ encoder input (axis 1) | 51 | QuadA3+ | Quadrature A+ encoder input (axis 3) |
| 2 | QuadA1- | Quadrature A- encoder input (axis 1) | 52 | QuadA3- | Quadrature A- encoder input (axis 3) |
| 3 | QuadB1+ | Quadrature B+ encoder input (axis 1) | 53 | QuadB3+ | Quadrature B+ encoder input (axis 3) |
| 4 | QuadB1- | Quadrature B- encoder input (axis 1) | 54 | QuadB3- | Quadrature B- encoder input (axis 3) |
| 5 | Index1+ | Index+ input (axis 1) | 55 | Index3+ | Index+ input (axis 3) |
| 6 | Index1- | Index- input (axis 1) | 56 | Index3- | Index- input (axis 3) |
| 7 | Vcc | +5V | 57 | Vcc | +5V |
| 8 | GND | Ground | 58 | GND | Ground |
| 9 | PosLim1 | Pos. direction limit switch input (axis 1) | 59 | PosLim3 | Pos. direction limit switch input (axis 3) |
| 10 | NegLim1 | Neg. direction limit switch input (axis 1) | 60 | NegLim3 | Neg. direction limit switch input (axis 3) |
| 11 | Home1 | Home input (axis 1) | 61 | Home3 | Home input (axis 3) |
| 12 | GND | Ground | 62 | GND | Ground |
| 13 | AxisOut1 | AxisOut output (axis 1) | 63 | AxisOut3 | AxisOut output (axis 3) |
| 14 | PWMMag1A | PWM magnitude output (axis 1) | 64 | PWMMag3A | PWM magnitude output (axis 3) |
| 15 | PWMSign1A | PWM sign output (axis 1) | 65 | PWMSign3A | PWM sign output (axis 3) |
| 16 | AxisIn1 | AxisIn input (axis 1) | 66 | AxisIn3 | AxisIn input (axis 3) |
| 17 | DAC1A | Analog mtr cmd output (axis 1), $\pm 10V$ | 67 | DAC3A | Analog mtr cmd output (axis 3), $\pm 10V$ |
| 18 | AGND | Ground for analog motor command | 68 | AGND | Ground for analog motor command |
| 19 | QuadA2+ | Quadrature A+ encoder input (axis 2) | 69 | QuadA4+ | Quadrature A+ encoder input (axis 4) |
| 20 | QuadA2- | Quadrature A- encoder input (axis 2) | 70 | QuadA4- | Quadrature A- encoder input (axis 4) |
| 21 | QuadB2+ | Quadrature B+ encoder input (axis 2) | 71 | QuadB4+ | Quadrature B+ encoder input (axis 4) |
| 22 | QuadB2- | Quadrature B- encoder input (axis 2) | 72 | QuadB4- | Quadrature B- encoder input (axis 4) |
| 23 | Index2+ | Index+ input (axis 2) | 73 | Index4+ | Index+ input (axis 4) |
| 24 | Index2- | Index- input (axis 2) | 74 | Index4- | Index- input (axis 4) |
| 25 | Vcc | +5V | 75 | Vcc | +5V |
| 26 | GND | Ground | 76 | GND | Ground |
| 27 | PosLim2 | Pos. direction limit switch input (axis 2) | 77 | PosLim4 | Pos. direction limit switch input (axis 4) |
| 28 | NegLim2 | Neg. direction limit switch input (axis 2) | 78 | NegLim4 | Neg. direction limit switch input (axis 4) |
| 29 | Home2 | Home input (axis 2) | 79 | Home4 | Home input (axis 4) |
| 30 | AxisOut2 | AxisOut output (axis 2) | 80 | AxisOut4 | AxisOut output (axis 4) |
| 31 | PWMMag2A | PWM magnitude output (axis 2) | 81 | PWMMag4A | PWM magnitude output (axis 4) |
| 32 | PWMSign2A | PWM sign output (axis 2) | 82 | PWMSign4A | PWM sign output (axis 4) |
| 33 | AxisIn2 | AxisIn input (axis 2) | 83 | AxisIn4 | AxisIn input (axis 4) |
| 34 | DAC2A | Analog mtr cmd output (axis 2), $\pm 10V$ | 84 | DAC4A | Analog mtr cmd output (axis 4), $\pm 10V$ |
| 35 | AGND | Ground for analog motor command | 85 | AGND | Ground for analog motor command |
| 36 | DigitalIn0 | General purpose digital input 0 | 86 | DigitalIn4 | General purpose digital input 4 |
| 37 | DigitalIn1 | General purpose digital input 1 | 87 | DigitalIn5 | General purpose digital input 5 |
| 38 | DigitalIn2 | General purpose digital input 2 | 88 | DigitalIn6 | General purpose digital input 6 |
| 39 | DigitalIn3 | General purpose digital input 3 | 89 | DigitalIn7 | General purpose digital input 7 |
| 40 | AmpEnable1 | Amplifier enable signal (axis 1) | 90 | AmpEnable3 | Amplifier enable signal (axis 3) |
| 41 | DigitalOut0 | General purpose digital output 0 | 91 | DigitalOut4 | General purpose digital output 4 |
| 42 | DigitalOut1 | General purpose digital output 1 | 92 | DigitalOut5 | General purpose digital output 5 |
| 43 | DigitalOut2 | General purpose digital output 2 | 93 | DigitalOut6 | General purpose digital output 6 |
| 44 | DigitalOut3 | General purpose digital output 3 | 94 | DigitalOut7 | General purpose digital output 7 |
| 45 | AmpEnable2 | Amplifier enable signal (axis 2) | 95 | AmpEnable4 | Amplifier enable signal (axis 4) |
| 46 | Reset | Hardware reset input | 96 | AnalogGND | Gnd for general purpose analog inputs |
| 47 | Analog1 | General purpose analog input 1 | 97 | Analog5 | General purpose analog input 5 |
| 48 | Analog2 | General purpose analog input 2 | 98 | Analog6 | General purpose analog input 6 |
| 49 | Analog3 | General purpose analog input 3 | 99 | Analog7 | General purpose analog input 7 |
| 50 | Analog4 | General purpose analog input 4 | 100 | Analog8 | General purpose analog input 8 |

4.2.3 GP Connector Using Brushless DC or Microstepping Motors

| Pin | Connection | Description | Pin | Connection | Description |
|-----------|-------------|--|-----|-------------|--|
| J6 | | | | | |
| 1 | QuadA1+ | Quadrature A+ encoder input (axis 1) | 51 | QuadA3+ | Quadrature A+ encoder input (axis 3) |
| 2 | QuadA1- | Quadrature A- encoder input (axis 1) | 52 | QuadA3- | Quadrature A- encoder input (axis 3) |
| 3 | QuadB1+ | Quadrature B+ encoder input (axis 1) | 53 | QuadB3+ | Quadrature B+ encoder input (axis 3) |
| 4 | QuadB1- | Quadrature B- encoder input (axis 1) | 54 | QuadB3- | Quadrature B- encoder input (axis 3) |
| 5 | Index1+ | Index+ input (axis 1) | 55 | Index3+ | Index+ input (axis 3) |
| 6 | Index1- | Index- input (axis 1) | 56 | Index3- | Index- input (axis 3) |
| 7 | Vcc | +5V | 57 | Vcc | +5V |
| 8 | GND | Ground | 58 | GND | Ground |
| 9 | PosLim1 | Pos. direction limit switch input (axis 1) | 59 | PosLim3 | Pos. direction limit switch input (axis 3) |
| 10 | NegLim1 | Neg. direction limit switch input (axis 1) | 60 | NegLim3 | Neg. direction limit switch input (axis 3) |
| 11 | Home1 | Home input (axis 1) | 61 | Home3 | Home input (axis 3) |
| 12 | GND | Ground | 62 | GND | Ground |
| 13 | AxisOut1 | AxisOut output (axis 1) | 63 | AxisOut3 | AxisOut output (axis 3) |
| 14 | n.c. | No connection | 64 | n.c. | No connection |
| 15 | n.c. | No connection | 65 | n.c. | No connection |
| 16 | AxisIn1 | AxisIn input (axis 1) | 66 | AxisIn3 | AxisIn input (axis 3) |
| 17 | DAC1A | Phase A analog mtr cmd output (axis 1), ±10V | 67 | DAC3A | Phase A analog mtr cmd output (axis 3), ±10V |
| 18 | AGND | Ground for analog motor command | 68 | AGND | Ground for analog motor command |
| 19 | QuadA2+ | Quadrature A+ encoder input (axis 2) | 69 | QuadA4+ | Quadrature A+ encoder input (axis 4) |
| 20 | QuadA2- | Quadrature A- encoder input (axis 2) | 70 | QuadA4- | Quadrature A- encoder input (axis 4) |
| 21 | QuadB2+ | Quadrature B+ encoder input (axis 2) | 71 | QuadB4+ | Quadrature B+ encoder input (axis 4) |
| 22 | QuadB2- | Quadrature B- encoder input (axis 2) | 72 | QuadB4- | Quadrature B- encoder input (axis 4) |
| 23 | Index2+ | Index+ input (axis 2) | 73 | Index4+ | Index+ input (axis 4) |
| 24 | Index2- | Index- input (axis 2) | 74 | Index4- | Index- input (axis 4) |
| 25 | Vcc | +5V | 75 | Vcc | +5V |
| 26 | GND | Ground | 76 | GND | Ground |
| 27 | PosLim2 | Pos. direction limit switch input (axis 2) | 77 | PosLim4 | Pos. direction limit switch input (axis 4) |
| 28 | NegLim2 | Neg. direction limit switch input (axis 2) | 78 | NegLim4 | Neg. direction limit switch input (axis 4) |
| 29 | Home2 | Home input (axis 2) | 79 | Home4 | Home input (axis 4) |
| 30 | AxisOut2 | AxisOut output (axis 2) | 80 | AxisOut4 | AxisOut output (axis 4) |
| 31 | n.c. | No connection | 81 | n.c. | No connection |
| 32 | n.c. | No connection | 82 | n.c. | No connection |
| 33 | AxisIn2 | AxisIn input (axis 2) | 83 | AxisIn4 | AxisIn input (axis 4) |
| 34 | DAC2A | Phase A analog mtr cmd output (axis 2), ±10V | 84 | DAC4A | Phase A analog mtr cmd output (axis 4), ±10V |
| 35 | AGND | Ground for analog motor command | 85 | AGND | Ground for analog motor command |
| 36 | DigitalIn0 | General purpose digital input 0 | 86 | DigitalIn4 | General purpose digital input 4 |
| 37 | DigitalIn1 | General purpose digital input 1 | 87 | DigitalIn5 | General purpose digital input 5 |
| 38 | DigitalIn2 | General purpose digital input 2 | 88 | DigitalIn6 | General purpose digital input 6 |
| 39 | DigitalIn3 | General purpose digital input 3 | 89 | DigitalIn7 | General purpose digital input 7 |
| 40 | AmpEnable1 | Amplifier enable signal (axis 1) | 90 | AmpEnable3 | Amplifier enable signal (axis 3) |
| 41 | DigitalOut0 | General purpose digital output 0 | 91 | DigitalOut4 | General purpose digital output 4 |
| 42 | DigitalOut1 | General purpose digital output 1 | 92 | DigitalOut5 | General purpose digital output 5 |
| 43 | DigitalOut2 | General purpose digital output 2 | 93 | DigitalOut6 | General purpose digital output 6 |
| 44 | DigitalOut3 | General purpose digital output 3 | 94 | DigitalOut7 | General purpose digital output 7 |
| 45 | AmpEnable2 | Amplifier enable signal (axis 2) | 95 | AmpEnable4 | Amplifier enable signal (axis 4) |
| 46 | Reset | Hardware reset input | 96 | AnalogGND | Ground for general purpose analog inputs |
| 47 | Analog1 | General purpose analog input 1 | 97 | Analog5 | General purpose analog input 5 |
| 48 | Analog2 | General purpose analog input 2 | 98 | Analog6 | General purpose analog input 6 |
| 49 | Analog3 | General purpose analog input 3 | 99 | Analog7 | General purpose analog input 7 |
| 50 | Analog4 | General purpose analog input 4 | 100 | Analog8 | General purpose analog input 8 |

4.2.4 GP Connector Using Step (Pulse & Direction) Motors

| Pin | Connection | Description | Pin | Connection | Description |
|-----------|-------------|--|-----|-------------|--|
| J6 | | | | | |
| 1 | QuadA1+ | Quadrature A+ encoder input (axis 1) | 51 | QuadA3+ | Quadrature A+ encoder input (axis 3) |
| 2 | QuadA1- | Quadrature A- encoder input (axis 1) | 52 | QuadA3- | Quadrature A- encoder input (axis 3) |
| 3 | QuadB1+ | Quadrature B+ encoder input (axis 1) | 53 | QuadB3+ | Quadrature B+ encoder input (axis 3) |
| 4 | QuadB1- | Quadrature B- encoder input (axis 1) | 54 | QuadB3- | Quadrature B- encoder input (axis 3) |
| 5 | Index1+ | Index+ input (axis 1) | 55 | Index3+ | Index+ input (axis 3) |
| 6 | Index1- | Index- input (axis 1) | 56 | Index3- | Index- input (axis 3) |
| 7 | Vcc | +5V | 57 | Vcc | +5V |
| 8 | GND | Ground | 58 | GND | Ground |
| 9 | PosLim1 | Pos. direction limit switch input (axis 1) | 59 | PosLim3 | Pos. direction limit switch input (axis 3) |
| 10 | NegLim1 | Neg. direction limit switch input (axis 1) | 60 | NegLim3 | Neg. direction limit switch input (axis 3) |
| 11 | Home1 | Home input (axis 1) | 61 | Home3 | Home input (axis 3) |
| 12 | GND | Ground | 62 | GND | Ground |
| 13 | AxisOut1 | AxisOut output (axis 1) | 63 | AxisOut3 | AxisOut output (axis 3) |
| 14 | Pulse1 | Pulse output (axis 1) | 64 | Pulse3 | Pulse output (axis 3) |
| 15 | Direction1 | Direction output (axis 1) | 65 | Direction3 | Direction output (axis 3) |
| 16 | AxisIn1 | AxisIn input (axis 1) | 66 | AxisIn3 | AxisIn input (axis 3) |
| 17 | AtRest1 | Atrest indicator output (axis 1) | 67 | AtRest3 | Atrest indicator output (axis 3) |
| 18 | GND | Ground | 68 | GND | Ground |
| 19 | QuadA2+ | Quadrature A+ encoder input (axis 2) | 69 | QuadA4+ | Quadrature A+ encoder input (axis 4) |
| 20 | QuadA2- | Quadrature A- encoder input (axis 2) | 70 | QuadA4- | Quadrature A- encoder input (axis 4) |
| 21 | QuadB2+ | Quadrature B+ encoder input (axis 2) | 71 | QuadB4+ | Quadrature B+ encoder input (axis 4) |
| 22 | QuadB2- | Quadrature B- encoder input (axis 2) | 72 | QuadB4- | Quadrature B- encoder input (axis 4) |
| 23 | Index2+ | Index+ input (axis 2) | 73 | Index4+ | Index+ input (axis 4) |
| 24 | Index2- | Index- input (axis 2) | 74 | Index4- | Index- input (axis 4) |
| 25 | Vcc | +5V | 75 | Vcc | +5V |
| 26 | GND | Ground | 76 | GND | Ground |
| 27 | PosLim2 | Pos. direction limit switch input (axis 2) | 77 | PosLim4 | Pos. direction limit switch input (axis 4) |
| 28 | NegLim2 | Neg. direction limit switch input (axis 2) | 78 | NegLim4 | Neg. direction limit switch input (axis 4) |
| 29 | Home2 | Home input (axis 2) | 79 | Home4 | Home input (axis 4) |
| 30 | AxisOut2 | AxisOut output (axis 2) | 80 | AxisOut4 | AxisOut output (axis 4) |
| 31 | Pulse2 | Pulse output (axis 2) | 81 | Pulse4 | Pulse output (axis 4) |
| 32 | Direction2 | Direction output (axis 2) | 82 | Direction4 | Direction output (axis 4) |
| 33 | AxisIn2 | AxisIn input (axis 2) | 83 | AxisIn4 | AxisIn input (axis 4) |
| 34 | AtRest2 | Atrest indicator output (axis 2) | 84 | AtRest4 | Atrest indicator output (axis 4) |
| 35 | GND | Ground | 85 | GND | Ground |
| 36 | DigitalIn0 | General purpose digital input 0 | 86 | DigitalIn4 | General purpose digital input 4 |
| 37 | DigitalIn1 | General purpose digital input 1 | 87 | DigitalIn5 | General purpose digital input 5 |
| 38 | DigitalIn2 | General purpose digital input 2 | 88 | DigitalIn6 | General purpose digital input 6 |
| 39 | DigitalIn3 | General purpose digital input 3 | 89 | DigitalIn7 | General purpose digital input 7 |
| 40 | AmpEnable1 | Amplifier enable signal (axis 1) | 90 | AmpEnable3 | Amplifier enable signal (axis 3) |
| 41 | DigitalOut0 | General purpose digital output 0 | 91 | DigitalOut4 | General purpose digital output 4 |
| 42 | DigitalOut1 | General purpose digital output 1 | 92 | DigitalOut5 | General purpose digital output 5 |
| 43 | DigitalOut2 | General purpose digital output 2 | 93 | DigitalOut6 | General purpose digital output 6 |
| 44 | DigitalOut3 | General purpose digital output 3 | 94 | DigitalOut7 | General purpose digital output 7 |
| 45 | AmpEnable2 | Amplifier enable signal (axis 2) | 95 | AmpEnable4 | Amplifier enable signal (axis 4) |
| 46 | Reset | Hardware reset input | 96 | AnalogGND | Ground for general purpose analog inputs |
| 47 | Analog1 | General purpose analog input 1 | 97 | Analog5 | General purpose analog input 5 |
| 48 | Analog2 | General purpose analog input 2 | 98 | Analog6 | General purpose analog input 6 |
| 49 | Analog3 | General purpose analog input 3 | 99 | Analog7 | General purpose analog input 7 |
| 50 | Analog4 | General purpose analog input 4 | 100 | Analog8 | General purpose analog input 8 |

4.2.5 Option Connector

When the Prodigy/CME PCI card is used with either brushless DC or microstepping motors, the Option Connector (labeled J5 in Figure 4-1 on page 61) provides additional signals for multi-phase motor output and input of signals such as Hall sensors. The Option Connector is a single high density 68-pin SCSI-style connector.

| Pin | Connection | Description | Pin | Connection | Description |
|-----------|----------------------|--|-----|--------------------|--|
| J5 | | | | | |
| 1 | PWMMag1A | Phase A PWM magnitude output (axis 1) | 35 | PWMMag1B | Phase B PWM magnitude output (axis 1) |
| 2 | PWMMag1C / PWMSign1B | Phase C PWM magnitude output (axis 1) / Phase B PWM sign output (axis 1) | 36 | GND | Ground |
| 3 | PWMSign1A | Phase A PWM sign output (axis 1) | 37 | GND | Ground |
| 4 | PWMMag2A | Phase A PWM magnitude output (axis 2) | 38 | PWMMag2B | Phase B PWM magnitude output (axis 2) |
| 5 | PWMMag2C / PWMSign2B | Phase C PWM magnitude output (axis 2) / Phase B PWM sign output (axis 2) | 39 | GND | Ground |
| 6 | PWMSign2A | Phase A PWM sign output (axis 2) | 40 | GND | Ground |
| 7 | PWMMag3A | Phase A PWM magnitude output (axis 3) | 41 | PWMMag3B | Phase B PWM magnitude output (axis 3) |
| 8 | PWMMag3C / PWMSign3B | Phase C PWM magnitude output (axis 3) / Phase B PWM sign output (axis 3) | 42 | GND | Ground |
| 9 | PWMSign3A | Phase A PWM sign output (axis 3) | 43 | GND | Ground |
| 10 | PWMMag4A | Phase A PWM magnitude output (axis 4) | 44 | PWMMag4B | Phase B PWM magnitude output (axis 4) |
| 11 | PWMMag4C / PWMSign4B | Phase C PWM magnitude output (axis 4) / Phase B PWM sign output (axis 4) | 45 | GND | Ground |
| 12 | PWMSign4A | Phase A PWM sign output (axis 4) | 46 | GND | Ground |
| 13 | Hall1 A | Phase A Hall sensor input (axis 1) | 47 | Hall1 B | Phase B Hall sensor input (axis 1) |
| 14 | Hall1 C | Phase C Hall sensor input (axis 1) | 48 | GND | Ground |
| 15 | Hall2 A | Phase A Hall sensor input (axis 2) | 49 | Hall2 B | Phase B Hall sensor input (axis 2) |
| 16 | Hall2 C | Phase C Hall sensor input (axis 2) | 50 | GND | Ground |
| 17 | Hall3 A | Phase A Hall sensor input (axis 3) | 51 | Hall3 B | Phase B Hall sensor input (axis 3) |
| 18 | Hall3 C | Phase C Hall sensor input (axis 3) | 52 | GND | Ground |
| 19 | Hall4 A | Phase A Hall sensor input (axis 4) | 53 | Hall4 B | Phase B Hall sensor input (axis 4) |
| 20 | Hall4 C | Phase C Hall sensor input (axis 5) | 54 | GND | Ground |
| 21 | Axis 1 pulse + | Axis 1 differential pulse + | 55 | Axis 1 pulse - | Axis 1 differential pulse - |
| 22 | Axis 2 pulse + | Axis 2 differential pulse + | 56 | Axis 2 pulse - | Axis 2 differential pulse - |
| 23 | Axis 3 pulse + | Axis 3 differential pulse + | 57 | Axis 3 pulse - | Axis 3 differential pulse - |
| 24 | Axis 4 pulse + | Axis 4 differential pulse + | 58 | Axis 4 pulse - | Axis 4 differential pulse - |
| 25 | Axis 1 direction + | Axis 1 differential direction + | 59 | Axis 1 direction - | Axis 1 differential direction - |
| 26 | Axis 2 direction + | Axis 2 differential direction + | 60 | Axis 2 direction - | Axis 2 differential direction - |
| 27 | Axis 3 direction + | Axis 3 differential direction + | 61 | Axis 3 direction - | Axis 3 differential direction - |
| 28 | Axis 4 direction + | Axis 4 differential direction + | 62 | Axis 4 direction - | Axis 4 differential direction - |
| 29 | Vcc | +5V | 63 | Vcc | +5V |
| 30 | AGND | Ground for analog motor command | 64 | AGND | Ground for analog motor command |
| 31 | DAC1B | Phase B analog mtr cmd output (axis 1), ±10V | 65 | DAC1A | Phase A analog mtr cmd output (axis 1), ±10V |
| 32 | DAC2B | Phase B analog mtr cmd output (axis 2), ±10V | 66 | DAC2A | Phase A analog mtr cmd output (axis 2), ±10V |
| 33 | DAC3B | Phase B analog mtr cmd output (axis 3), ±10V | 67 | DAC3A | Phase A analog mtr cmd output (axis 3), ±10V |
| 34 | DAC4B | Phase B analog mtr cmd output (axis 4), ±10V | 68 | DAC4A | Phase A analog mtr cmd output (axis 4), ±10V |

4.2.6 Serial Connector

There are two serial connectors, Serial Connector (J17) is a 10-pin dual header-style connector and provides signals for two serial ports in RS232 mode, or a single serial port in RS485 mode. The following section provides information for this connector, and provides pinouts when operated in RS232 mode, RS485 full duplex mode, and RS485 half duplex mode.

| Pin | Connection | RS232 | RS485 Full Duplex | RS485 Half Duplex |
|------------|-------------------------------|--|---|---|
| J17 | | | | |
| 1 | RS485Select | Open (default) selects RS232 mode for both Serial1 and Serial2 | Tie to ground selects RS485 mode for both Serial1 and Serial2 | Tie to ground selects RS485 mode for both Serial1 and Serial2 |
| 2 | RS485Rcv ⁺ | no connect | Positive (non-inverting) receive input | no connect |
| 3 | Sr1Xmt | Serial 1 transmit output | no connect | no connect |
| 4 | Sr12Rcv/RS485Rcv ⁻ | Serial 2 receive input | Negative (inverting) receive input | no connect |

| Pin | Connection | RS232 | RS485 Full Duplex | RS485 Half Duplex |
|-----|-------------------------------|--------------------------|--|---|
| 5 | Srl1Rcv | Serial 1 receive input | no connect | no connect |
| 6 | Srl2Xmt/RS485Xmt ⁺ | Serial 2 transmit output | Positive (non-inverting) transmit output | Positive (non-inverting) transmit/receive |
| 7 | No connect | no connect | no connect | no connect |
| 8 | RS485Xmt ⁻ | no connect | Negative (inverting) transmit output | Negative (inverting) transmit/receive |
| 9 | GND | Ground | Ground | Ground |
| 10 | GND | Ground | Ground | Ground |

4.2.7 Sync I/O Connector

The two Sync I/O connectors located on the Prodigy/CME PCI card (J3 and J4 in Figure 4-1 on page 61) allow for the synchronization of multiple Prodigy/CME PCI cards within a single system. This configuration enables operation within the same cycle period. If multiple cards are installed, yet not inter-connected, any additional cards would begin working after the first card (master) was initialized. However, none of the axes would be synchronized. With Sync I/O activated, the servo loops of all slave cards are synchronized to the servo loop of the master card. This allows for precise synchronization of all implemented axes.

To connect two or more Prodigy/CME PCI cards for synchronization, a Sync I/O cable (one cable for each set of two cards) is required. This cable may be connected to either of the two Sync I/O connectors on the cards. Both connectors function as either an input or output; the two sync connectors are wired in parallel. For more information on synchronizing multiple Prodigy/CME PCI cards, see the *Magellan Motion Processor User's Guide*.

The pinouts for the Sync I/O connectors are defined in the following table:

| Pin Number | Signal |
|---------------|--|
| J3, J4 | |
| 1 | Sync-in or sync-out pin, depending on whether the card is master or slave. |
| 2 | GND |

The following diagram shows three synchronized Cards.

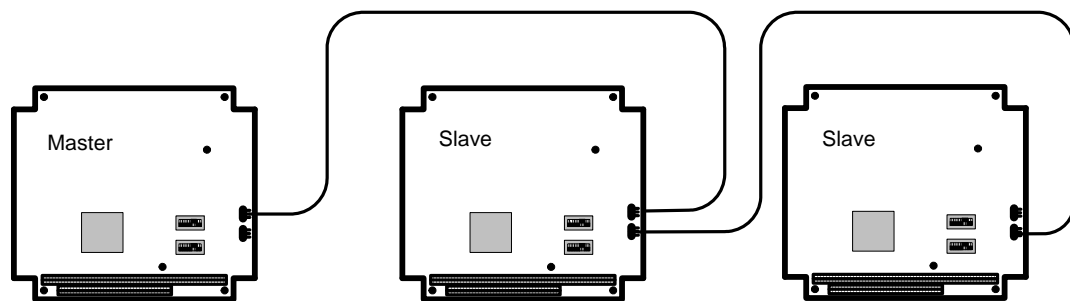


Figure 4-3:
Sync I/O
connector to
three cards

4.2.8 CAN Connectors

The Prodigy/CME PCI card's controller area network (CAN) transceivers are designed for use with CAN controllers, or with equivalent devices. They are intended for use in applications employing the CAN serial communication physical layer in accordance with the ISO 11898 standard. The transceiver provides differential transmit and differential receive capability to/from a CAN controller at speeds up to 1 Mbps.

The CAN connector (J16) is an 8-pin header-style connector. Termination at each end of the cable run is generally recommended unless cable lengths are very short and speed is slow. ISO-11898 requires 120 ohm termination at each end of the bus. Note that it is up to the customer to verify their network topology and operating parameters.

See Section 2.5.3, “CANbus Communications,” on page 41 for more information on the functionality of the CANbus port.

| Pin Number | Signal | Description |
|------------|-------------|---|
| J16 | | |
| 1 | Vcc | +5V |
| 2 | CAN+ | Positive CAN signal connection |
| 3 | no connect | reserved, do not connect |
| 4 | CAN- | Negative CAN signal connection |
| 5 | no connect | reserved, do not connect |
| 6 | CAN- | Negative CAN signal connection (either this signal or signal 4 can be used for CAN-) |
| 7 | GND | Ground |
| 8 | termination | optional termination - connecting this pin with pin 6 will provide 120 ohm termination on the CAN signals |

4.2.9 Ethernet Connector

The Prodigy/CME PCI's Ethernet transceivers are designed for use with 10/100 base-TX Ethernet and support both TCP and UDP protocols. See Section 2.5.4, “Ethernet Communications,” on page 41 for more information on the functionality of the Ethernet port.

The Ethernet Connector (J15) is an 10-pin 2mm header-style connector with pinouts as follows:

| Pin Number | Signal | Description |
|------------|-------------|---|
| J15 | | |
| 1 | EthernetTx+ | Ethernet differential transmit positive |
| 2 | EthernetTx- | Ethernet differential transmit negative |
| 3 | EthernetRx+ | Ethernet differential receive positive |
| 4 | no connect | no connect |
| 5 | no connect | no connect |
| 6 | EthernetRx- | Ethernet differential receive negative |
| 7 | no connect | no connect |
| 8 | no connect | no connect |
| 9 | no connect | no connect |
| 10 | no connect | no connect |

4.2.10 Extension Connector

This is a 46-pin connector (J1 in Figure 4-1 on page 61), which connects the DC-1000 expansion card to the Prodigy/CME PCI. The DC-1000 is a Synchronous Serial Interface (SSI) for absolute encoders, and connects to the Prodigy/CME PCI card's parallel input connector. See Section 4.2, “Connectors,” on page 63 for detailed information about the DC-1000, its functionality, and installation instructions.

4.2.11 Connector Parts Reference

The following table is supplied as a reference only. “(V)” indicates the vertical mounting configuration of the Prodigy/CME PCI card, while “(H)” indicates the horizontal configuration.

| Label | Description | Card Part Number | Connector Mate |
|-------|---------------------|---|--|
| J6 | GP Connector | AMP 2-5178238-9 or 3M NI02A0-52B2PC | AMP 2-5175677-9 (plug) AMP 1-176793-0 (shell) or 3M 101A0-6000EC (plug) 3M 103A0-A200-00 (shell) |
| J5 | Option Connector | Molex 1587-0205 | Molex 70498-4068 or Harting 60 06 068 5440 |
| J3,J4 | Synch Connector | Molex 53261-0271 | Molex 51021-0200 |
| J17 | Serial Connector | Molex 87832-1020 | Molex 51110-1051 (housing) and 50394-8100 (crimp pin) |
| J16 | CAN Connector | Molex 87832-0820 | Molex 51110-0851 (housing) and Molex 50394-8100 (crimp pin) |
| J15 | Ethernet Connector | Molex 87832-1020 | Molex 51110-1051 (housing) and 50394-8100 (crimp pin) |
| J1 | Extension Connector | Samtec TMM-123-01-L-D-SM | |

4.3 Connections Summary—Motor Amplifiers

The Prodigy/CME PCI card supports four methods of output to motor amplifiers, as described in the following table:

| Motor Output Method | Signal Output |
|---------------------|--|
| DAC | Analog signals from the onboard D-A converters. |
| PWM sign-magnitude | Pulse width modulated signals with separate magnitude and sign signals per output phase. |
| PWM 50/50 | Pulse width modulated square-wave signals with a single PWM signal per output phase. |
| Pulse & Direction | One signal representing pulse information, and a signal representing direction. |

In addition, each motor axis may have one, two, or three output phases associated with it. For DC brush motors, the number of phases is one. For multi-phase motors such as brushless DC or microstepping motors, the number of phases can be two or three, depending on the output waveform programmed into the Prodigy/CME PCI card. For more information, see the *Magellan Motion Processor User's Guide*.

The following tables provide convenient summaries of amplifier connections for common configurations of motor output method and motor type. These outputs should be connected from the designated connector pins to the appropriate amplifier inputs. Note that the names of the pins will vary among amplifiers. Common names are shown.

4.3.1 DC Brush Motor Connections

To connect to DC Brush motors the GP Connector (J6) is used.

| Motor Output Method | Connection Name | Amplifier Input Connection Name* | Axis 1 | Axis 2 | Axis 3 | Axis 4 |
|----------------------|-----------------|----------------------------------|--------|--------|--------|--------|
| | | | | | | |
| DAC | DACI-4A | Ref+ or V+ | J6-17 | J6-34 | J6-67 | J6-84 |
| | AGND | Ref- or GND | J6-18 | J6-35 | J6-68 | J6-85 |
| PWM sign/magnitude** | PWMMagI-4A | PWM magnitude | J6-14 | J6-31 | J5-64 | J6-81 |
| | PWMSignI-4A | PWM direction | J6-15 | J6-32 | J6-65 | J6-82 |
| | GND | Ground | J6-8 | J6-26 | J6-58 | J6-76 |

* Names of amplifier connections vary. Common names are shown.

** Both the Magnitude and Sign connections are used if the Magellan's PWM Sign Magnitude mode is selected. If PWM50/50 mode is selected, then only the Magnitude connections are used.

4.3.2 Brushless DC Motor Connections

To connect to Brushless DC motors using internal commutation (provided by the Magellan Motion Processor), two sinusoidal phase signals are driven per axis via the Option Connector (J5).

| Motor Output Method | Connection Name | Amplifier Input Connection Name* | Axis 1 | Axis 2 | Axis 3 | Axis 4 |
|---------------------|-----------------|----------------------------------|--------|--------|--------|--------|
| | | | | | | |
| DAC | DACI-4A | Ref1+ or V1+ | J5-65 | J5-66 | J5-67 | J5-68 |
| | DACI-4B | Ref2+ or V2+ | J5-31 | J5-32 | J5-33 | J5-34 |
| | AGND | Ref- or GND | J5-30 | J5-64 | J5-30 | J5-64 |
| PWM 50/50 | PWMMagI-4A | PWM phase 1 | J5-1 | J5-4 | J5-7 | J5-10 |
| | PWMMagI-4B | PWM phase 2 | J5-35 | J5-38 | J5-41 | J5-44 |
| | PWMMagI-4C | PWM phase 3 | J5-2 | J5-5 | J5-8 | J5-11 |
| | GND | Ground | J5-36 | J5-39 | J5-42 | J5-45 |

* Names of amplifier connections vary. Common names are shown.

4.3.3 Microstepping Motor Connections

To connect to step motors in microstepping mode, two sinusoidal phase signals are driven per axis via the Option Connector (J5).

| Motor Output Method | Connection Name | Amplifier Input | Axis 1 | Axis 2 | Axis 3 | Axis 4 |
|----------------------|-----------------|------------------|--------|--------|--------|--------|
| | | Connection Name* | | | | |
| DAC | DAC1-4A | Ref1+ or V1+ | J5-65 | J5-66 | J5-67 | J5-68 |
| | DAC1-4B | Ref2+ or V2+ | J5-31 | J5-32 | J5-33 | J5-34 |
| | AGND | Ref- or GND | J5-30 | J5-64 | J5-30 | J5-64 |
| PWM sign/magnitude** | PWMMag1-4A | PWM magnitude | J5-1 | J5-4 | J5-7 | J5-10 |
| | PWMSign1-4A | PWM direction | J5-3 | J5-6 | J5-9 | J5-12 |
| | PWMMag1-4B | PWM magnitude | J5-35 | J5-38 | J5-41 | J5-44 |
| | PWMSign1-4B | PWM direction | J5-2 | J5-5 | J5-8 | J5-11 |
| | GND | Ground | J5-36 | J5-39 | J5-42 | J5-45 |

* Names of amplifier connections vary. Common names are shown.

**Both the Magnitude and Sign connections are used if the Magellan's PWM Sign Magnitude mode is selected. If PWM50/50 mode is selected, then only the Magnitude connections are used.

4.3.4 Step (Pulse & Direction) Motor Connections

To connect to step motors using single-ended pulse & direction input amplifiers the GP Connector (J6) is used, and for differential drive the Pulse & Direction Connector (J7) as well as the GP Connector (J6) are used.

| Motor Output Method | Connection Name | Amplifier Input | Axis 1 | Axis 2 | Axis 3 | Axis 4 |
|----------------------------------|-----------------|-------------------|--------|--------|--------|--------|
| | | Connection Name* | | | | |
| Pulse & direction (single-ended) | Pulse1-4 | Pulse or step | J6-14 | J6-61 | J6-28 | J6-62 |
| | Direction1-4 | Direction | J6-15 | J6-63 | J6-30 | J6-64 |
| | GND | Ground | J6-8 | J6-51 | J6-16 | J6-52 |
| Pulse & direction (differential) | Pulse1-4+ | Pulse + or step + | J5-21 | J5-22 | J5-23 | J5-24 |
| | Pulse1-4- | Pulse - or step - | J5-55 | J5-56 | J5-57 | J5-58 |
| | Direction1-4+ | Direction + | J5-25 | J5-26 | J5-27 | J5-28 |
| | Direction1-4- | Direction - | J5-59 | J5-16 | J5-12 | J5-8 |
| Pulse & direction | AtRest 1-4 | At Rest | J6-17 | J6-34 | J6-67 | J6-84 |

* Names of amplifier connections vary. Common names are shown.

4.4 Cables

The following table provides a summary of the standard cable accessories available with Prodigy/CME PCI cards.

| Component Part Number | Description |
|-----------------------|---|
| Cable-1003-01.R | 100-pin high density ribbon cable. This 3 foot long cable connects the primary motion connector (GP Connector) to an identical connector on a receiving card such as the IM-1000 breakout interconnect card. |
| Cable-1006-01.R | Same as above (Cable-1003-01.R) but 6 feet in length. |
| Cable-3003I-01.R | 68-pin SCSI bracket. This cable plugs into the card's Option Connector and interfaces to a 68-pin ribbon socket through a PCI bracket suitable for connecting to Cable-3003E-01 |
| Cable-3003E-01.R | 68-pin ribbon cable. This 3 foot long cable is a 68-pin ribbon extender cable. It typically connects to Cable-3003I-01.R. |
| Cable-4301-01.R | 10-pin serial header cable. This 1 foot cable plugs into the card's 10-pin serial header connector and provides a single female DB-9 output with dual serial signals. It can be connected to Cable-4355-01.R to provide dual DB-9 serial outputs. |
| Cable-4355-01.R | Dual serial cable. This 5 foot long "Y" cable connects to the DB-9 output of Cable-4301-01.R and provides two female DB-9 connectors suitable for connection to a PC serial port or USB-to-serial converter. |
| Cable-4701-01.R | CANbus header to RJ45 connector. This 1 foot cable plugs into the card's 8-pin CANbus header connector and provides an RJ45 output. |
| Cable-RJ45-02.R | CANbus and Ethernet cable. This 2 meter cable connects to the card's CANbus or Ethernet outputs. It has RJ45 connectors on both ends. |
| TRM-RJ45-02.R | CANbus terminator. This is an RJ45 terminator that may be used to electrically terminate a CANbus network string. |
| Adapt-RJ45T-01.R | CANbus splitter. This Y configuration splitter duplicates the CANbus RJ45 connections into two identical RJ45 connectors, making it easy to connect the card in a daisy chain configuration. |
| Cable-4505-01.R | Ethernet header to RJ45 cable. This 5 foot cable connects to the card's 10-pin Ethernet header connector and provides an RJ45 output. |
| DC-1000 | Parallel encoder input adaptor. This daughter card module allows parallel-word and other encoders which use the SSI interface format to be directly connected. |
| IM-1000 | Breakout interconnect module provides convenient jack-screw type terminators for the 100-pin cable. Used with Cable-1003-01.R or Cable-1006-01.R. |
| Adapt-USB232-01.R | This adapter provides USB to serial conversion. It is useful for connecting to the Prodigy/CME's DB-9 serial port from a USB port. |

The following sections provide detailed information on each of these cables.

4.4.1 Cable-1003-01.R

PMD Part #: Cable-1003-01.R

Description: High density 100-pin connector to 100-pin connector ribbon cable.

Length: 3.0 feet

Cable: 50 pair, 30 AWG, shielded round

Wiring: Straight through

4.4.2 Cable-1006-01.R

PMD Part #: Cable-1006-01.R

Description: High density 100-pin latch connector to 100-pin latch connector ribbon cable.

Length: 6.0 feet

Cable: 50 pair, 30 AWG, shielded round

Wiring: Straight through

4.4.3 Cable-3003I-01.R

PMD Part #: Cable-3003I-01.R

Description: SCSI high density 68-pin male to 68-pin female connector with PCI "L" bracket cable

Length: 1.75 feet

Cable: 68 wire, 0.025" pitch, 30 AWG, ribbon cables

Wiring: Straight through

4.4.4 Cable-3003E-01.R

PMD Part #: Cable-3003E-01.R

Description: High density 68-pin male to 68-pin male SCSI cable

Length: 3.0 feet

Cable: 34 pair, 30 AWG, shielded

Wiring: Straight through

4.4.5 Cable-4301-01.R

PMD Part #: Cable-4301-01.R

Description: 10-pin dual header (J17) to DB-9 cable

Length: 1.0 feet

Cable: 9 conductor, 26 AWG, untwisted

Wiring: See table.

| J17 Pin | Signal | Connects to |
|---------|-----------------------------------|-------------|
| 1 | RS485Rcv ⁺ | 1 |
| 2 | SrI1Xmt | 6 |
| 3 | SrI2Rcv/ RS485Rcv ⁻ | 2 |
| 4 | SrI1Rcv | 7 |
| 5 | SrI2Xmt/ RS485Xmt ⁺ | 3 |
| 6 | SrI2Xmt/ RS485Xmt ⁺ | 8 |
| 7 | RS485Xmt ⁻ | 4 |
| 8 | GND | 9 |
| 9 | GND | 5 |
| 10 | no connect | - |

4.4.6 Cable-4355-01.R

PMD Part #: Cable-4355-01.R

Description: Male DB-9 Dual Serial to two single-channel female DB-9 cable

Length: 5.0 feet

Cable: (2) 3 conductor, 26 AWG, Untwisted, shielded, UL STYLE 2464

Note: Dual female DB-9 ends are labeled "Port1" and "Port2."

Wiring: See table.

| Male DB-9 Pin | Signal | Connects to dual female DB-9 Pin: |
|---------------|-----------------------------------|-----------------------------------|
| 1 | RS485Select | - |
| 2 | Sr1IXmt | Port1-2 |
| 3 | Sr1IRcv | Port1-3 |
| 4 | No connect | - |
| 5 | GND | Port1-5, Port2-5 |
| 6 | RS485Rcv ⁺ | - |
| 7 | Sr12Rcv/ RS485Rcv ⁻ | Port2-3 |
| 8 | Sr12Xmt/ RS485Xmt ⁺ | Port2-2 |
| 9 | RS485Xmt ⁻ | - |

4.4.7 Cable-4701-01.R

PMD Part #: Cable-4701-01.R

Description: 8-pin dual header to RJ45 cable

Length: 1.0 feet

Cable: 4 conductor, 24 AWG, twisted

Wiring: See table.

| J16 Pin | Signal | Connects to: |
|---------|-------------|--------------|
| 1 | Vcc | no connect |
| 2 | CAN+ | 1 |
| 3 | no connect | no connect |
| 4 | CAN- | 2 |
| 5 | no connect | no connect |
| 6 | CAN- | no connect |
| 7 | GND | 3, 7 |
| 8 | termination | no connect |

4.4.8 Cable-RJ45-02-R

PMD Part #: Cable-RJ45-02-R

Description: Male RJ-45 to male RJ-45 cable

Length: 2.0 meters

Cable: 4P, 24 AWG, UTP, Cat 5e

Wiring: straight through

4.4.9 TRM-RJ45-02-R

PMD Part #: TRM-RJ45-02-R

Description: RJ45 CANbus terminator
(120 ohm between pins 1 and 2)

4.4.10 Adapt-RJ45T-01.R

| | | | |
|---|------------|---------------------|---------------------|
| PMD Part #: Adapt-RJ45T-01.R | Pin | Split Pin 1: | Split Pin 2: |
| Description: Modular RJ45 splitter | 1 | 1 | 1 |
| Wiring: See table | 2 | 2 | 2 |
| | 3 | 3 | 3 |
| | 4 | 4 | 4 |
| | 5 | 5 | 5 |
| | 6 | 6 | 6 |
| | 7 | 7 | 7 |
| | 8 | 8 | 8 |

4.4.11 Cable-4505-01-R

| | | | |
|--|----------------|---------------|---------------------|
| PMD Part #: Cable-4505-01.R | J16 Pin | Signal | Connects to: |
| Description: 10-pin dual header to RJ45 cable | 1 | EthernetTx+ | 1 |
| Length: 5.0 feet | 2 | EthernetTx- | 2 |
| Cable: 4 pair, 24 AWG, Cat 5/5e | 3 | EthernetRx+ | 3 |
| Wiring: See table. | 4 | no connect | 4 |
| | 5 | no connect | 5 |
| | 6 | EthernetRx- | 6 |
| | 7 | no connect | 7 |
| | 8 | no connect | 8 |
| | 9 | no connect | - |
| | 10 | no connect | - |

4.5 Environmental and Electrical Ratings

| | |
|---------------------------------|---|
| Storage temperature: | -40 to +125 degrees C (-40° F to +257° F) |
| Operating temperature: | 0 to +70 degrees C (32° F to +158° F) |
| Power requirements: | 4.8V to 5.25V operating range, 0.8A (no outputs on) |
| Supply voltage limits: | -0.3V to +7V |
| Analog (DAC) output range: | -10.0V to +10.0V, \pm 3mA min/axis, short circuit protected |
| Analog input range: | 0 to 3.3V, 1.4 KOhm input impedance |
| Digital I/O voltage range: | 0V to 5V, TTL thresholds, inputs pulled up to 5V through 4.7 kOhm resistors |
| Digital outputs drive capacity: | DC output source or sink current: \pm 50mA |
| CAN communications: | 2.0B compliant, non-isolated, 1 Mbps |
| Serial communications: | RS232 signaling or RS485 Full or Half Duplex (data only) |
| Ethernet communications: | 10/100BASE-TX (10/100 Mbps) |

4.6 Mechanical Dimensions

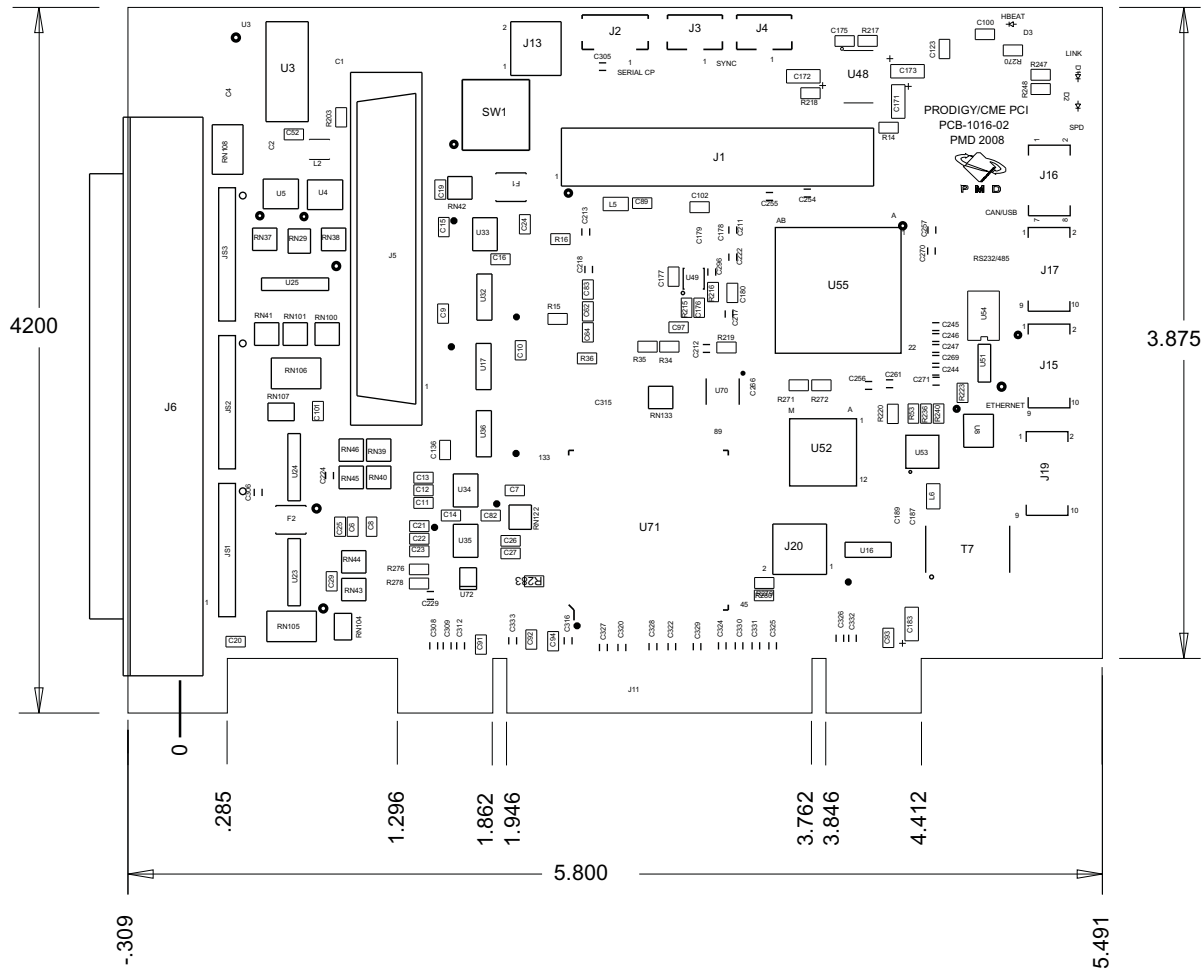


Figure 4-4: Prodigy/CME mechanical dimensions

4.7 User I/O Memory Map

The Magellan Motion Processor reserves the sector from address 1000h to 10FFh in peripheral space for user-defined I/O devices. The Prodigy/CME PCI card uses this sector as shown in the following table:

| Address | Device | Description |
|---------------|---------------------------|---|
| 1000h | General Purpose I/O | Includes the 8 digital inputs and 8 digital outputs |
| 1001h | Amplifier Enable register | Also includes the DAC output enable for information |
| 1002h | Reset Monitor register | See Section 2.3.6, "Reset Monitor," on page 33. |
| 1003h | Reserved | |
| 1004h | Watchdog register | See Section 2.3.4, "Card Watchdog Timer," on page 33. |
| 1005h | Reserved | |
| 1006h - 100Fh | Reserved | |
| 1010h - 101Fh | DC-1000 | Includes the SSI clock, resolution, and absolute position registers |
| 1020h - 10CFh | None | Available for use with custom peripherals over Extension Connector |
| 10D0h - 10DAh | Reserved | |
| 10DBh | Build register | Build number of Prodigy FPGA |
| 10DCh - 10DFh | Reserved | |

| Address | Device | Description |
|---------------|------------|--|
| 10E0h - 10EFh | None | Available for use with custom peripherals over Extension Connector |
| 10F0h - 10FDh | Reserved | |
| 10FEh | Model type | For compatibility with older Magellan Motion Card family |
| 10FFh | Card ID | See Section 2.3.7, "Card ID," on page 34. |

See the *Magellan Motion Processor User's Guide* for more information on peripheral memory space.

5. Interconnect Module

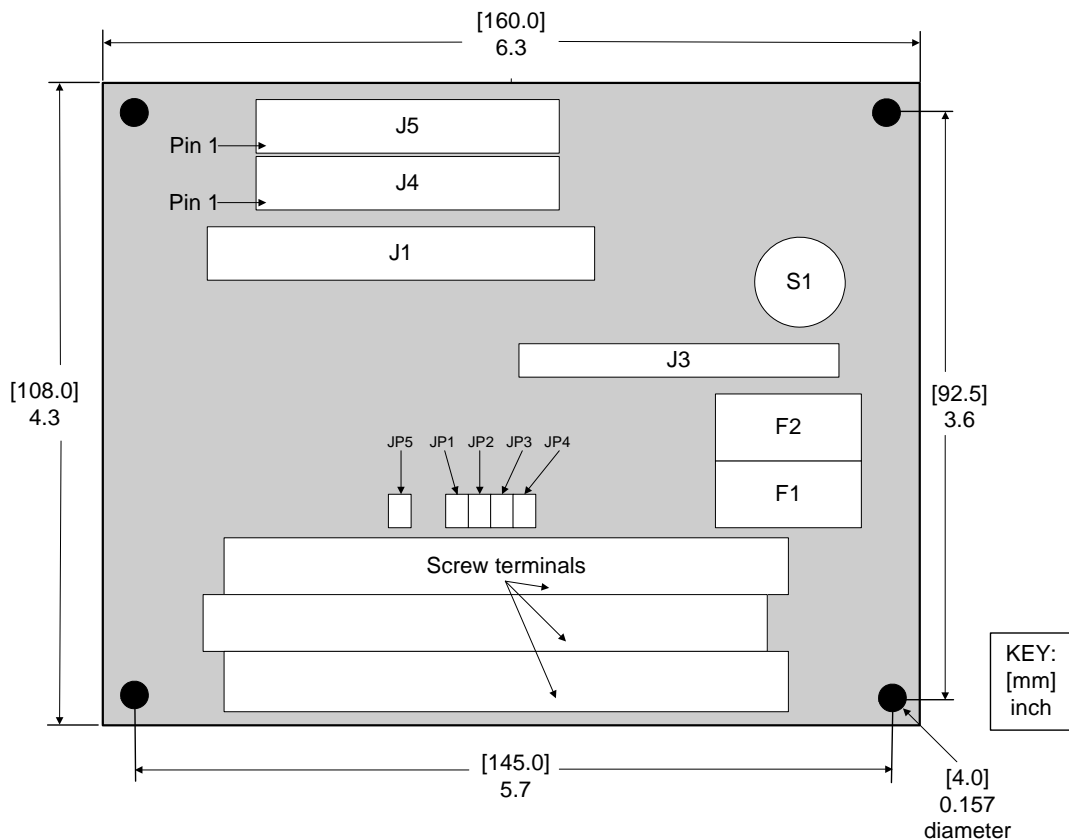
In This Chapter

- ▶ IM-1000 Interconnect Module
- ▶ DC-1000 SSI Option Card

5.1 IM-1000 Interconnect Module

The IM-1000 is an interconnect module which assists in the set up and configuration of Prodigy/CME PCI cards, and also provides complete options for external connections. All wiring to and from the Prodigy/CME PCI card, the amplifiers, power supplies, IOs and encoder feedback may easily be connected to this central point.

Figure 5-1:
IM-1000
location of
components



| Connector | Description |
|-----------|-----------------------------|
| F1 | 1 Amp fuse for power supply |
| F2 | 1 Amp fuse for power supply |
| J1 | 100-position connector |

| Connector | Description |
|-----------------|---|
| J3 | OPTO22 connector |
| J4 | 50-position connector |
| J5 | 50-position connector |
| S1 | External hardware reset button |
| Screw terminals | (3) 100-terminal connection blocks for connecting to one 100-position, or two 50-position cables. |

The enclosure of the card is a standard Phoenix DIN rail mounting system. Alternatively, the enclosure may be removed, and the card mounted to other systems via the 3.5 mm mounting holes located on the corners of the card.

The pinout descriptions that follow provide a detailed illustration of the IM-1000's connections. Please observe the differences between the DC brush and brushless DC and stepping motor versions. The screw terminal numbers correspond to the pinout description for Prodigy/CME PCI GP Connector (J6) Cable-5003 Header connectors (see Section 4.4.1, "Cable-1003-01.R," on page 73 for a description). "H1" in the tables below refers to Header1, and "H2" in the tables refers to Header2.

| Header | IM-1000 | Screw Terminal |
|--------|---------|----------------|
| H1 | J4 | 1-50 |
| H2 | J5 | 51-100 |

The following table lists the functions of the remaining screw terminals:

| Screw Terminals | Function |
|-----------------|----------------------------------|
| 101, 103 | 5V from J1 through 1 Amp fuse F2 |
| 102, 104, 105 | GND |
| 106 | External power in |
| 107 | GND |
| 108 | Unused |

The Cable-5003 Header 1 and Header 2 cables supply 5V DC to pins H1-7, H1-25, H2-7, and H2-25 through a 1 Amp fuse to power encoders. When the jumpers JP1, JP2, JP3 and JP4 are installed 1-2, this 5V is passed on to screw terminals 7, 25, 57, and 75, respectively. Alternatively, it is also possible to use an external power supply in the range of 5 - 12 VDC for the encoders. To do this, install jumpers JP1, JP2, JP3 and JP4 on pins 2-3 and connect the external power to screw terminal 106 and its ground to 107. This external power will be limited to 1 Amp by fuse F1.

Connections to J3 (the OPTO22 connector) are as follows:

| H1/H2 | J3 | Description |
|-------|----|-------------------------------|
| | 49 | 5V if JP5 jumper is installed |
| H1-36 | 47 | DigitalIn0 |
| H1-37 | 45 | DigitalIn1 |
| H1-38 | 43 | DigitalIn2 |
| H1-39 | 41 | DigitalIn3 |
| H2-36 | 39 | DigitalIn4 |
| H2-37 | 37 | DigitalIn5 |
| H2-38 | 35 | DigitalIn6 |
| H2-39 | 33 | DigitalIn7 |
| H1-41 | 31 | DigitalOut0 |
| H1-42 | 29 | DigitalOut1 |
| H1-43 | 27 | DigitalOut2 |
| H1-44 | 25 | DigitalOut3 |
| H2-41 | 23 | DigitalOut4 |
| H2-42 | 21 | DigitalOut5 |
| H2-43 | 19 | DigitalOut6 |
| H2-44 | 17 | DigitalOut7 |
| H1-40 | 15 | AmpEnable1 |
| H1-45 | 13 | AmpEnable2 |
| H2-40 | 11 | AmpEnable3 |

| H1/H2 | J3 | Description |
|------------------------|----|-------------|
| H2-45 | 09 | AmpEnable4 |
| H1-13 | 07 | AxisOut1 |
| H1-30 | 05 | AxisOut2 |
| H2-13 | 03 | AxisOut3 |
| H2-30 | 01 | AxisOut4 |
| All even-numbered pins | | GND |

5.2 IM-600 Interconnect Module

The IM-600 is an interconnect module which assists in the set up and configuration of the Option Connector and the Pulse & Direction Connector for all versions of Prodigy cards by providing jack screw 'breakout' functionality, making it easy to hand-connect motion peripherals.

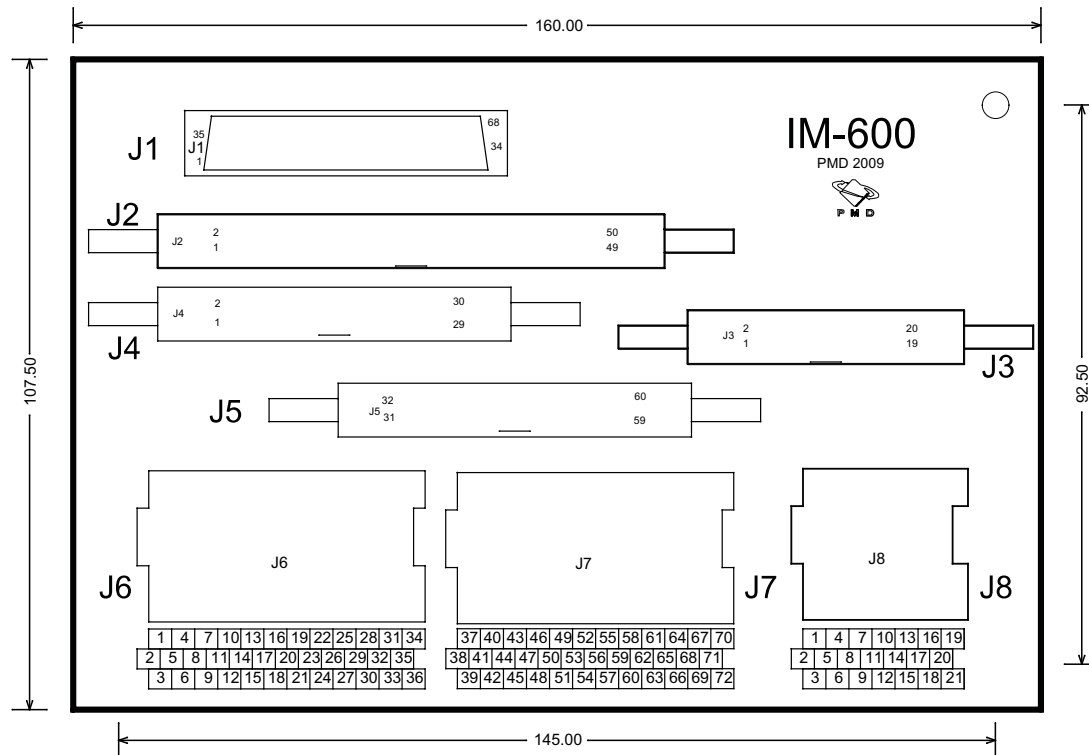


Figure 5-2:
IM-600
location of
components

The following table summarizes the functionality of the various components of the IM-600 module.

| Connector | Description |
|-----------|-------------------------------|
| J1 | 68-pin SCSI-style connector |
| J2 | 50-pin dual header connector |
| J3 | 20-pin dual header connector |
| J4 & J5 | 30-pin dual header connectors |
| J6/J7 | 72-pin jack screw array |
| J8 | 21-pin jack screw array |

The enclosure of the card is a standard Phoenix DIN rail mounting system. Alternatively, the enclosure may be removed, and the card mounted to other systems via the 3.5 mm mounting holes located on the corners of the card.

The table below illustrates the connections from the PC/104, PCI, and Stand-Alone versions of the Prodigy card's Option Connector and Pulse & Direction Connector, to the IM-600's connectors. In all cases the jack screw pins are

wired 'straight through' from the connector input, so that the documentation for the Prodigy card connectors or cable headers can be conveniently used to determine correct pin placement on the IM-600's jack screw pin connections:

| Prodigy Card format | Card Connector & Type | Connect to IM-600 Connector | Screw terminal connections & wiring |
|---------------------|-----------------------------------|-----------------------------|-------------------------------------|
| PC/I04* | J9 - Option Connector | J2 | J6/J7 pins 1-50, straight through |
| | J12 - Pulse & Direction Connector | J3 | J8 pins 1-20, straight through |
| PCI* | J5 - Option Connector | J1 | J6/J7 pins 1-68, straight through |
| Stand-Alone | J5/H1** - Option Connector | J4 | J6/J7 pins 1-30, straight through |
| | J5/H2** - Option Connector | J5 | J6/J7 pins 31-60, straight through |
| | J7 - Pulse & Direction Connector | J3 | J8 pins 1-20, straight through |

* Both CME and non-CME version Prodigy cards

** J5 Option Connector on the Prodigy/CME Stand-Alone card bifurcates via PMD p/n Cable-7003-01.R into two 30-pin dual headers, labeled Header 1 (H1) and Header 2 (H2)

5.3 DC-1000 SSI Option Card

The DC-1000 is an optional expansion card for the Prodigy/CME PCI card. The DC-1000 is a Synchronous Serial Interface (SSI) for absolute encoders, and connects to the Prodigy/CME PCI card's Extension Connector J1 (see Figure 4-1 on page 61). The data of the absolute encoders is read through the DC-1000 interface, and is then transmitted to the Prodigy/CME PCI's Magellan Motion Processor. The electronics of the absolute encoder converts the parallel-format data into serial-format data, and transmits this data to the DC-1000. The DC-1000 converts the incoming data into a 16-bit data word, from Gray code to binary code, and transmits the data to the Prodigy/CME PCI card. The DC-1000 can accept connections of up to four axes, and the data of each axis can be programmed for a read rate of up to 20 kHz.

Some of the major features of the DC-1000 are as follows:

- Up to four axes can be read
- Resolution can be programmed for 10, 12, 13, or 25 bits
- Compatible with single- and multi-turn absolute encoders
- Programmable frequencies of 1.1 MHz, 550 kHz, 275 kHz, and 137.5 kHz
- Default jumper settings can be overwritten with software commands
- Update rates of up to 20 kHz for all four axes
- Automatic recognition and addition of position overflow
- Index search not required
- Absolute position is read at power-on
- Incremental and absolute encoders can be mixed in any combination
- Real-time switching between incremental and absolute modes
- Compatible with any absolute encoders which support the Synchronous Serial Interface (SSI), such as Haidenheim, Stegmann, Thalheim

Unlike incremental encoders, optical encoders always supply the exact absolute position. At power-on, the encoder will report its exact position. This precludes the need for the machine to locate a machine reference. Absolute encoders are grouped into two categories: single-turn and multi-turn. Encoders with a resolution of up to 360 degrees are single-turn encoders. Those which can resolve several turns (normally up to 4,096) are classified as multi-turn encoders. The key components of an absolute encoder consist of an optical code disk, an opto-electronic sensor system, and pulse-shaping circuitry and amplification.

The code discs of most absolute encoders generate position data in Gray code. This differs from binary and BCD code, as Gray code guarantees higher reliability for increased accuracy of data transmission. The requirements for the transmission of absolute position data is one data line for each bit of information. An encoder with 13 bits of resolution would therefore require 13 data lines; an encoder with 25 bits of resolution would require 25 data lines. A system with four encoders would require 100 (4 x 25) data lines to transmit the data to the controller. The SSI interface requires only four data lines, regardless of the encoder's resolution. This is accomplished by converting the parallel data into serial data. Each bit of the parallel data is sent on the data lines +/- synchronously to the +/- clock lines. The clock speed of the DC-1000 is programmable between 1.1 MHz and 137.5 kHz. The clock speed depends on the length of the data cables between encoder and controller.

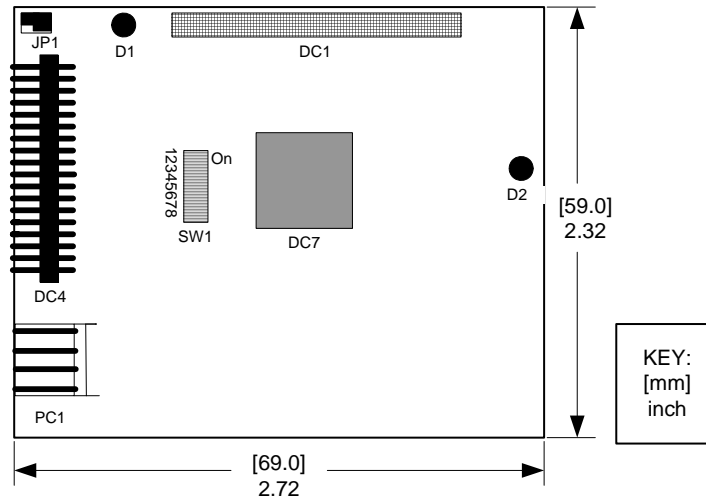
The DC-1000 converts the received Gray code to binary code serial data, and stores it in parallel format in registers. This data can then be read by the Prodigy/CME PCI's Magellan Motion Processor. For a system with four absolute encoders and a cable length of up to 50 meters, the position of every encoder is read 20,000 times per second. This guarantees timely and accurate data. The DC-1000 can be programmed for encoder resolutions of 10, 12, 13 and 25 bits. The transmission speed may be programmed to accommodate cable lengths of greater than 300 meters, and both single-turn encoders and multi-turn encoders may be employed. The Prodigy/CME PCI card will capture the absolute position of single-turn encoders for a full 32 bits. When a roll-over occurs, it will be detected by the card and the correct position will be stored. The direction of motion will also be correctly detected.

5.3.1 DC-1000 Specifications

- Dimensions: 2.75 inches x 2.25 inches
- 40-position mating connector for the Prodigy/CME PCI card
- 2 stand-offs with screws for mounting to Prodigy/CME PCI card
- Encoder mating connector for 5V DC or 12V DC from PC power supply
- Jumper for selecting 5V or 12V supply voltage for the encoders
- 26-position IDC connector and mating stub IDC ribbon cable
- DIP-switches for setting encoder resolution for each axis
- Four position registers (one for each axis) maintain absolute positions
- All positions may be read by the Prodigy/CME PCI card at any time
- Four read and write registers (one for each axis) for programmatically setting resolution and clock speed
- Resolution and clock speed settings can be different for every axis

The following diagram illustrates the location of the DC-1000's main components and connectors. The component side of the card is shown.

Figure 5-3:
DC-1000
location of
components



The names and descriptions of the main components of the DC-1000 are detailed in the following table:

| Label | Name | Description |
|--------|--------------------|---|
| DC1 | Parallel connector | This socket connects to the parallel input connector of the Prodigy/CME PCI card. |
| D1, D2 | Mounting holes | These holes (2) are in direct alignment with the mounting holes on the Prodigy/CME PCI card, and are used for the mounting screws and stand-offs. |
| JP1 | Jumper | This jumper selects either 5V or 12V supply voltage. 5V DC is the default setting. |
| DC4 | Encoder connector | The encoders are connected to this pin block. |
| PC1 | Power connector | Use this pin block to connect a small Molex connector for DC power. |
| SW1 | DIP switches | This block of DIP switches is used to set encoder resolution for each axis. |
| DC7 | PLD | This is the processor for the DC-1000. |

5.3.2 DC-1000 Connections

The pinouts for connector DC4 are outlined in the following table:

| IDC Pin No. | Description | Axis |
|-------------|--------------------------|------|
| 1 | +5 or +12 VDC to Encoder | X |
| 2 | Clock 1+ | X |
| 3 | Clock 1- | X |
| 4 | GND | X |
| 5 | DIN 1+ | X |
| 6 | DIN 1- | X |
| 7 | +5 or +12 VDC to Encoder | Y |
| 8 | Clock 2+ | Y |
| 9 | Clock 2- | Y |
| 10 | GND | Y |
| 11 | DIN 2+ | Y |
| 12 | DIN 2- | Y |
| 13 | No connection | |
| 14 | No connection | |
| 15 | +5 or +12 VDC to Encoder | Z |
| 16 | Clock 3+ | Z |
| 17 | Clock 3- | Z |
| 18 | GND | Z |
| 19 | DIN 3+ | Z |
| 20 | DIN 3- | Z |
| 21 | +5 or +12 VDC to Encoder | W |
| 22 | Clock 4+ | W |

| IDC Pin No. | Description | Axis |
|-------------|-------------|------|
| 23 | Clock 4- | W |
| 24 | GND | W |
| 25 | DIN 4+ | W |
| 26 | DIN 4- | W |

The settings for the jumper (labeled JP1 in Figure 5-3 on page 84) are outlined in the following table:

| Pins Jumpered | Setting |
|---------------|-----------------|
| 1 - 2 | 12V DC |
| 2 - 3 | 5V DC (default) |

5.3.3 DC-1000 Default Parameters

At power-on or after a reset, the system clock will be set to 1.1 MHz, and the default system resolution will be determined by the settings of the DIP switches. Additional parameters are defined in the following tables:

| Switches (0...1) | Resolution |
|------------------|------------|
| 00 | 10 bits |
| 01 | 12 bits |
| 10 | 13 bits |
| 11 | 25 bits |

For the DIP switches, ON = 1; OFF = 0. Default settings are all DIP switches set to OFF (0). The DIP switches are labeled SW1 in Figure 5-3 on page 84.

| DIP Switches | Axis |
|--------------|--------|
| 1 - 2 | X-axis |
| 3 - 4 | Y-axis |
| 5 - 6 | Z-axis |
| 7 - 8 | W-axis |

These values may be overwritten by software.



5.3.4 Installing the DC-1000

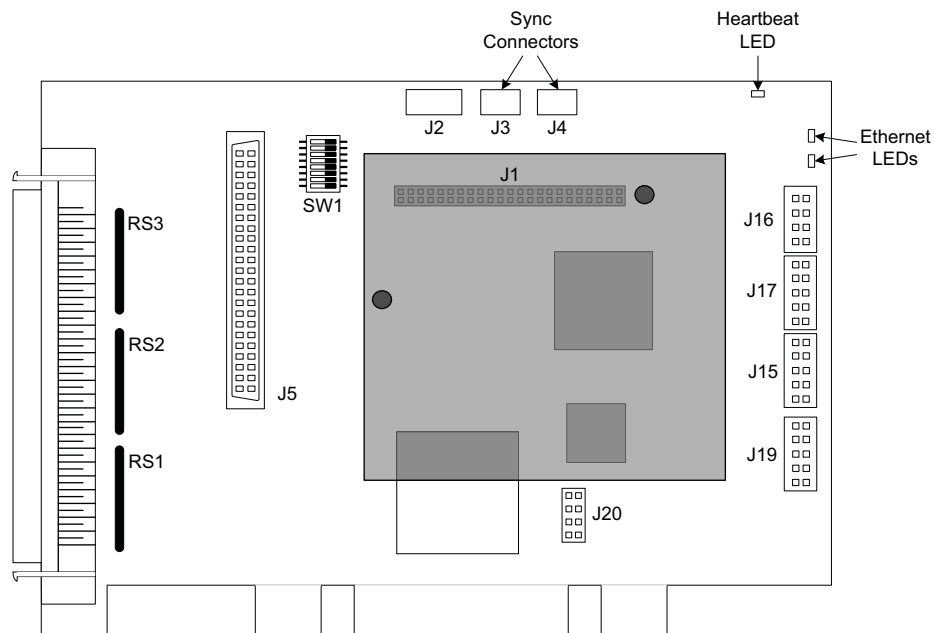
The DC-1000 attaches directly to the Prodigy/CME PCI card, with the component sides of both cards facing each other. The DC-1000's parallel input socket DC1 connects to the Prodigy/CME PCI card's Extension Connector J1 (see Figure 4-1 on page 61).

- 1 To mount the DC-1000 to the Prodigy/CME PCI card, orient the card as shown in Figure 5-4 on page 86.
- 2 Position the DC-1000's parallel connector socket (DC1) over the Prodigy/CME PCI card's Extension Connector. Ensure that the top edge of the mating connectors are aligned. The DC1 connector socket is shorter than the Prodigy/CME PCI card's Extension Connector, and when the top edges of both are properly aligned, three sets of pins on the Extension Connector will be visible at the bottom of the DC-1000's connector when viewed from above the cards. Before pressing the DC-1000 into place, view the cards from the front in the same orientation as Figure 5-4 on page 86. Verify that the two sets of mounting holes are in exact alignment.

- 3 When this alignment is correct, press firmly and evenly to seat the pins of the Prodigy/CME PCI card's Extension Connector into the sockets of the DC-1000's parallel connector.
- 4 Place the two standoffs between the cards so that they are aligned with the two mounting holes, and insert the mounting screws through the mounting holes and standoffs.
- 5 Once the cards are mated, attach the supplied IDC cable to the encoder connector on the DC-1000. Pin 1 of the connector is located nearest the JP1 jumper, and the cable's connector must be oriented so that the red-striped wire of the IDC cable attaches to pin 1. The red marking on the cable will then be located on the left side of the cards.
- 6 To connect DC power to the DC-1000, attach a small Molex plug from the computer's power supply to the 4-pin white connector on the DC-1000 (labeled PC1 in Figure 5-3 on page 84). These are the small power connectors, which are used to connect a floppy drive.

The DC-1000 mounted to the front of the Prodigy/CME PCI card is illustrated in the following figure:

Figure 5-4:
DC-1000
mounted on
Prodigy/CME



5.3.5 DC-1000 Software Commands

The DC-1000 may be operated in mixed-mode. Any or all of the four axes may be configured in any combination of incremental or absolute modes. The following commands are required to configure the absolute encoder mode for the DC-1000.

SetEncoderSource Axis n, mode //Sets the mode to 0 = incremental or 1 = absolute for Axis n

Setting the mode to absolute (parallel) deactivates the incremental mode and setting to incremental deactivates the absolute (parallel) mode.

SetEncoderModulus Axis n, modulus //Sets the modulus (0 to 65535) for Axis n

Modulus settings are required for single-turn absolute encoders in order to capture the full position when the encoder exceeds a full turn. The Prodigy/CME PCI will accumulate the full 32 bits for the position. The modulus must always

be one-half of the encoder's resolution. Assuming an encoder with 12 bits (4,096), the resulting modulus to be programmed would be 2,048.

One more command is required to program the SSI clock and resolution on the DC-1000.

```
SetSSIRegister Axis n, resolution, frequency //This function sets the resolution and clock frequency for Axis n
```

Resolution can be set to one of 4 values: 10 bits, 12 bits, 13 bits or 25 bits. The frequency can also be set to one of 4 values: 1.1 MHz (default), 550 kHz, 275 kHz or 137.5 kHz. Each axis can be programmed independently.

See the *Magellan Motion Processor Programmer's Command Reference* for more information.

This page intentionally left blank.

Index

Numerics

- 100-terminal connection blocks 80
- 26-position IDC connector 83
- 40-position mating connector 83
- 50-position connector 80

A

- absolute encoder 83
- absolute mode 86
- accessing
 - communication ports 51
 - Magellan-attached devices 56
 - on-card resources 55
- accessory products 12
- actions, PRP 51
- Adapt-RJ45T-01.R 76
- AmpEnable 19
- amplifier
 - connections 70
 - enable output signals 32
 - inputs 70
- amplifiers 63
- analog
 - input 47
 - input range 76
 - output range 76
- applying power 18
- automatically assigned addresses 54
- AxisIn sensor 46
- AxisOut sensor 46

B

- BCD code 83
- brushless DC motor connections 17, 71

C

- Cable-1003-01.R 73
- Cable-1006-01.R 74
- Cable-3003-01.R 74
- Cable-3003E-01.R 74
- Cable-4301-01.R 74
- Cable-4355-01.R 74, 75
- Cable-4505-01-R 76

- Cable-4701-01.R 75
- Cable-5003-01.R 73
- Cable-7003-01.R 74
- Cable-RJ45-02-R 75
- cables 73
- CAN connector 81
- CAN serial communication physical layer 68
- CANbus packet processing 60
- card
 - function summary 26
 - types 9
- C-Motion 11, 14
 - engine development environment 11
 - engine development environment features 11
 - features 11
- communication ports, accessing 51
- components table, front of board 15
- connection summary 16, 70
- connector 14
 - functions 63
 - parts reference 70
 - pins 70
- connectors 63
- controller area network transceivers 68
- cycle period 68

D

- DAC 70
 - out 48
 - output enable 32
- data cables, length of 83
- DC brush motor connections 16, 71
- DC-1000
 - connections 84
 - default parameters 85
 - expansion card 82
 - installing 85
 - specifications 83
- differential
 - connections 62
 - encoder input 62
 - encoding 46

receive capability 68
transmit capability 68
digital I/O, general purpose 30
digital outputs drive capacity 76
DIP-switches, DC-1000 83

E

encoder 63
connections 62
feedback 79
inputs 62
mating connector 83
resolutions 83
settings 62
signals 62
environmental and electrical ratings 76
Ethernet connector 69
Ethernet packet processing 60
extension connector 69
external
connections 79
hardware reset button 80

F

first time system verification 19
front diagram 14

G

general-purpose
digital I/O 30
inputs 30
outputs 30
GP connector 63
Gray code 82

H

Hall sensors 67
Home sensor 46

I

IM-1000 79
incremental mode 86
index capture 28
Index signal 46
inputs, general-purpose 30
installation 14
sequence 13
interconnect module 79
ISO 11898 standard 68

L

Limits sensor 46

M

Magellan instruction set 28
microstepping
cards 65
motor connections 18, 72
modulus settings 86
motion processor reset 33
motor
amplifier output methods 70
axis 70
output configuration 15
output method 70
multi-phase
motor output 67
motors 70
multi-turn encoders 83

N

negative direction limit input 46

O

operating temperature 76
optical encoders 83
option connector 67
OPTO22 connector 80
output
phases 70
waveform 70
outputs, general-purpose 30

P

parallel input connector 69
parallel-format data 82
PCI packet processing 58
peripheral connections 52
peripherals 63
Phoenix DIN rail mounting system 80, 81
Phoenix EN rail mounting system 80, 81
PMD Resource Access Protocol 49
polarity 46
positive direction limit input 46
power
requirements 76
power up 18
Prodigy/CME Stand-Alone connectors 14
profile generation 26

Pro-Motion 11, 14
 features 11

PRP 49
 actions 51
 communication formats 58
 resources 51

PRP messages
 CANbus 60
 Ethernet 60
 over PCI bus 58
 over serial 59

PRP/PCI message format 58

pulse & direction 47
 cards 66

PWM
 50/50 70
 out 47
 sign-magnitude 70

Q

QuadA signal 46
QuadB signal 46

R

ReadIO command 31
required hardware 13
reset
 condition 18, 33
 monitor word 34
resistor pack 14
 settings 15, 62
resistor packs 61, 62
resource
 addressing 49
 PRP 51
Resource Access Protocol 49
roll-over 83

S

screw terminal 80
serial
 connector 67
serial packet processing 59
serial-format data 82
servo loop closure 26
shunting 32
signal conditioning 27
single-ended
 connections 46
 encoder input 62
 encoders 62
 single-turn encoders 83
 software
 commands, DC-1000 86
 libraries 48
 packages 10
SSI 69
step (pulse & direction) motor connections 72
storage temperature 76
supply voltage limits 76
sync I/O 68
 cable 68
 connectors 68
 pinouts 68
synchronizing multiple boards 68
Synchronous Serial Interface 69, 82

T

transmission speed 83
TRM-RJ45-02-R 75

U


under-voltage detection circuit 33
undesired motion 19
user I/O memory map 77
user-settable components 61

V

VB-Motion 14
 features 12

W

watchdog function 33
WriteIO command 31

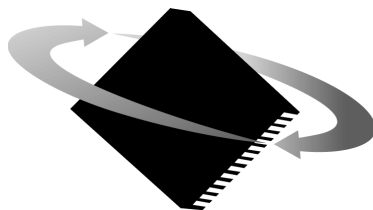


This page intentionally left blank.

For additional information, or for technical assistance,
please contact PMD at (978) 266-1210.

You may also e-mail your request to support@pmdcorp.com

Visit our website at <http://www.pmdcorp.com>



P M D

Performance Motion Devices
80 Central Street
Boxborough, MA 01719