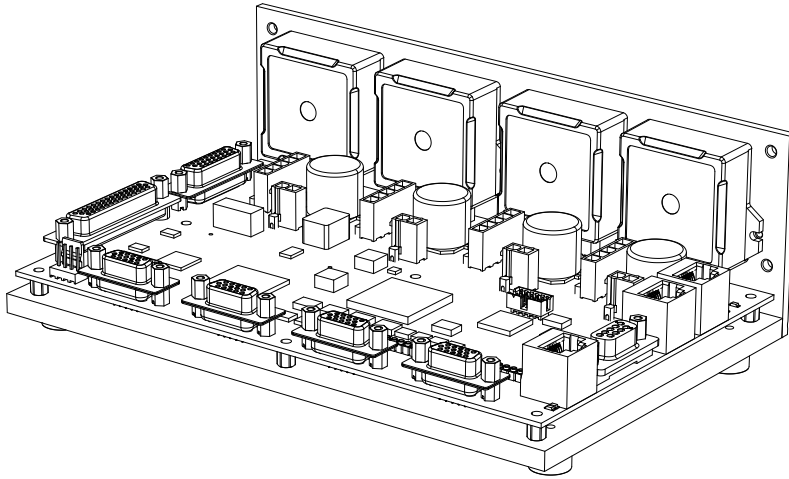


**PERFORMANCE  
MOTION DEVICES**  
MOTION CONTROL AT ITS CORE



# Prodigy<sup>®</sup>/CME Machine-Controller Developer Kit

---

## User Manual

Revision 1.0 / January 2025

**Performance Motion Devices, Inc.**

80 Central Street, Boxborough, MA 01719

[www.pmdcorp.com](http://www.pmdcorp.com)

---



---

## **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 1998–2025 by Performance Motion Devices, Inc.

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

---

## Warranty

Performance Motion Devices, Inc. warrants that its products shall substantially comply with the specifications applicable at the time of sale, provided that this warranty does not extend to any use of any Performance Motion Devices, Inc. product in an Unauthorized Application (as defined below). Except as specifically provided in this paragraph, each Performance Motion Devices, Inc. product is provided “as is” and without warranty of any type, including without limitation implied warranties of merchantability and fitness for any particular purpose.

Performance Motion Devices, Inc. reserves the right to modify its products, and to discontinue any product or service, without notice and advises customers to obtain the latest version of relevant information (including without limitation product specifications) before placing orders to verify the performance capabilities of the products being purchased. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement and limitation of liability.

## Unauthorized Applications

Performance Motion Devices, Inc. products are not designed, approved or warranted for use in any application where failure of the Performance Motion Devices, Inc. product could result in death, personal injury or significant property or environmental damage (each, an “Unauthorized Application”). By way of example and not limitation, a life support system, an aircraft control system and a motor vehicle control system would all be considered “Unauthorized Applications” and use of a Performance Motion Devices, Inc. product in such a system would not be warranted or approved by Performance Motion Devices, Inc.

By using any Performance Motion Devices, Inc. product in connection with an Unauthorized Application, the customer agrees to defend, indemnify and hold harmless Performance Motion Devices, Inc., its officers, directors, employees and agents, from and against any and all claims, losses, liabilities, damages, costs and expenses, including without limitation reasonable attorneys’ fees, (collectively, “Damages”) arising out of or relating to such use, including without limitation any Damages arising out of the failure of the Performance Motion Devices, Inc. product to conform to specifications.

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

Performance Motion Devices, Inc. assumes no liability for applications assistance or customer product design. Performance Motion Devices, Inc. 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 Performance Motion Devices, Inc. covering or relating to any combination, machine, or process in which such products or services might be or are used. Performance Motion Devices, Inc.’s publication of information regarding any third party’s products or services does not constitute Performance Motion Devices, Inc.’s approval, warranty or endorsement thereof.

## Patents

Performance Motion Devices, Inc. may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Patents and/or pending patent applications of Performance Motion Devices, Inc. are listed at <https://www.pmdcorp.com/company/patents>.

---

## Related Documents

### **Prodigy/CME Machine-Controller User Guide**

Complete description of the Prodigy/CME Machine-Controller motion board including information on operation, electrical function, and mechanical dimensions.

### **Magellan Motion Control IC User Guide**

Complete description of the Magellan Motion Control IC features and functions with detailed theory of operations.

### **Atlas Digital Amplifier User Manual**

Description of the Atlas Digital Amplifier electrical and mechanical specifications along with a summary of its operational features.

### **Atlas Digital Amplifier Complete Technical Reference**

Complete technical and mechanical description of the Atlas Digital Amplifier with detailed theory of operations.

### **C-Motion Magellan Programming Reference**

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

### **C-Motion PRP Programming Reference**

Description of Prodigy/CME motion board and ION/CME 500 drive commands with software architecture overview, command syntax, and examples.

### **C-Motion® Engine Development Tools**

Description of the C-Motion Engine hardware resources, development environment, software tools, and command set.

# Table of Contents

<b>Chapter 1. Introduction</b> .....	<b>9</b>
1.1 Prodigy/CME Machine-Controller Developer Kit Overview .....	9
1.2 Guide to this Manual .....	11
1.3 Software Installation.....	12
1.4 Recommended Hardware.....	14
<b>Chapter 2. Quick Start Guide</b> .....	<b>15</b>
2.1 Step #1 — Configuring the Board .....	15
2.2 Step #2 — Making Motion Hardware Connections.....	16
2.3 Step #3 — Applying Power .....	19
2.4 Step #4 — First Time System Verification .....	19
2.5 Next Steps .....	24
<b>Chapter 3. Going Further with Pro-Motion</b> .....	<b>27</b>
3.1 Pro-Motion Screen Layout .....	27
3.2 Project Window .....	29
3.3 Device Control Window .....	30
3.4 Axis Control Window .....	31
3.5 Status Window .....	32
3.6 Monitor Window .....	34
3.7 Command Window.....	35
3.8 Scope Window .....	38
3.9 Project Configuration Save & Restore.....	40
3.10 Configuration Export to C-Motion.....	41
3.11 Pro-Motion Application Notes .....	42
3.12 Troubleshooting Suggestions.....	52
<b>Chapter 4. Communication Port Connections</b> .....	<b>55</b>
4.1 Serial Communications .....	55
4.2 CAN Communications .....	61
4.3 Ethernet Communications.....	63
4.4 Communicating With Attached Devices.....	65
<b>Chapter 5. Software Development</b> .....	<b>69</b>
5.1 Overview of C-Motion .....	69
5.2 Getting Started with C-Motion PRP.....	72
5.3 PC User Code Development .....	74
5.4 C-Motion Engine User Code Development .....	77
<b>Chapter 6. Electrical Reference</b> .....	<b>81</b>
6.1 User-Settable Components .....	81
6.2 Connectors.....	83
6.3 Motor Connections Quick Reference .....	93
6.4 Cables & Accessories .....	94
<b>Appendix A.Installation</b> .....	<b>97</b>
A.1 Developer Kit Assembly .....	97

---

<b>Appendix B.Index Capture Qualification.....</b>	<b>103</b>
<b>Index.....</b>	<b>105</b>

# List of Figures

---

1-1	4-Axis Prodigy/CME Machine Controller Developer Kit	9
1-2	Prodigy/CME Machine-Controller Developer Kit Elements	11
2-1	Components and Layout, Front of Board	16
2-2	Power Connections	18
3-1	Velocity and Acceleration Versus Time for Point-To-Point Trapezoidal Trajectory	51
4-1	Hardware Setup for Communications via RS232	55
4-2	Example Setup for Communications via RS485	57
4-3	Prodigy/CME Machine-Controller RS485 Signal Connection Diagram	58
4-4	Hardware Setup for Communicating via CAN	61
4-5	Hardware Setup for Communicating via Ethernet	63
4-6	PMD Controller with Attached Device Network	65
5-1	PC-Connected to PRP Device Showing Resources	72
5-2	Typical Connection Scheme for PC-Based User Code Development	74
5-3	Recommended Sequence for PC Code Development	76
5-4	Typical Connection Links When User Code Runs on C-Motion Engine	77
5-5	Recommended Sequence for C-Motion Engine Code Development	79
6-1	Components and Layout, Front of Board	81
6-2	Power Connections	86
A-1	Mounting Machine Controller Board onto L-Bracket Base Plate	97
A-2	Thermal Transfer Material Attachment	98
A-3	Atlas Installation into Machine-Controller Board	99
A-4	Compact to Ultra Compact Atlas Format Converter	100
A-5	Attaching Atlas Units to Vertical Plate	100
A-6	L-Bracket Base Mechanical Dimensions	101
A-7	L-Bracket Vertical Plate Mechanical Dimensions	102
B-1	QuadA, QuadB, and Index Signal Qualification Combinations	103

*This page intentionally left blank.*



# 1. Introduction

## In This Chapter

- ▶ Prodigy/CME Machine-Controller Developer Kit Overview
- ▶ Guide to this Manual
- ▶ Software Installation
- ▶ Recommended Hardware

## 1.1 Prodigy/CME Machine-Controller Developer Kit Overview

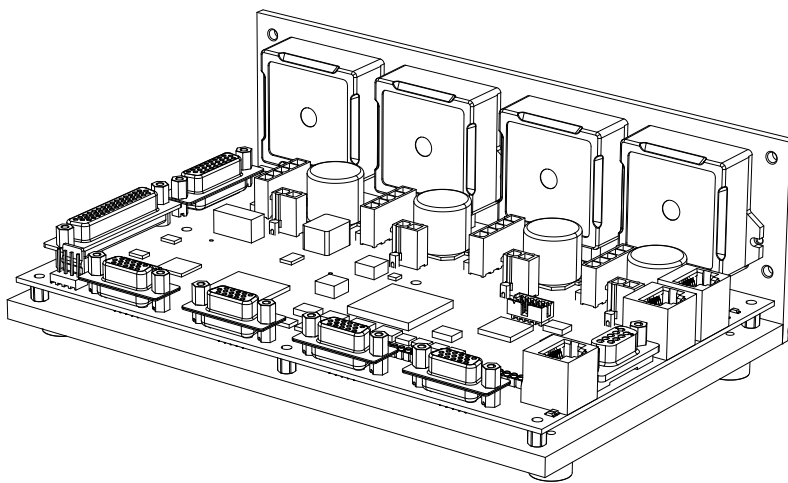


Figure 1-1:  
4-Axis Prodigy/  
CME Machine  
Controller  
Developer Kit

This manual provides complete information on the Prodigy/CME Machine-Controller Developer Kit.

Prodigy/CME Machine-Controller boards provide high-performance control of DC Brush, Brushless DC, and stepper motors. All Prodigy boards are based on PMD's Magellan<sup>®</sup> Motion Control ICs, which perform high speed motion control functions such as profile generation, servo loop closure, pulse & direction signal generation, and many other real-time functions.

For detailed information on the Prodigy/CME Machine-Controller board refer to the *Prodigy/CME Machine-Controller User Guide*. For detailed information on Magellan Motion Control ICs refer to the *Magellan Motion Control IC User Guide*.

### 1.1.1 Developer Kit Part Numbers

There are four different machine controller developer kits, designed to address a range of applications and machine controller configurations. Depending on the developer kit P/N ordered the developer kit may come completely assembled and ready to use out of the box, or some assembly of the developer kit may be required.

The following table provides detailed information on each available machine controller developer kit P/N:

P/N	Motors supported	Comments
PRK33ML44002	DC Brush, Brushless DC, step motor	Complete assembled two axis machine controller developer kit setup with two high power multi-motor Atlas amplifiers
PRK33ML44403	DC Brush, Brushless DC, step motor	Complete assembled three-axis machine controller developer kit setup with three high power multi-motor Atlas amplifiers
PRK33ML44444	DC Brush, Brushless DC, step motor	Complete assembled four axis machine controller developer kit setup with four high power multi-motor Atlas amplifiers
PRK33MK00004	N/A	Miscellaneous parts* needed to assemble a developer kit if Prodigy/CME Machine-Controller board has been purchased separately

\* This P/N is ordered when a Prodigy/CME Machine-Controller board and Atlas units have been ordered separately.

Most users will get started with the Prodigy/CME Machine-Controller board by purchasing one of the three ready-to-go developer kit setups. These represent two, three, and four axis fully loaded setups capable of driving DC Brush, Brushless DC, or step motors with up to 500W of power.

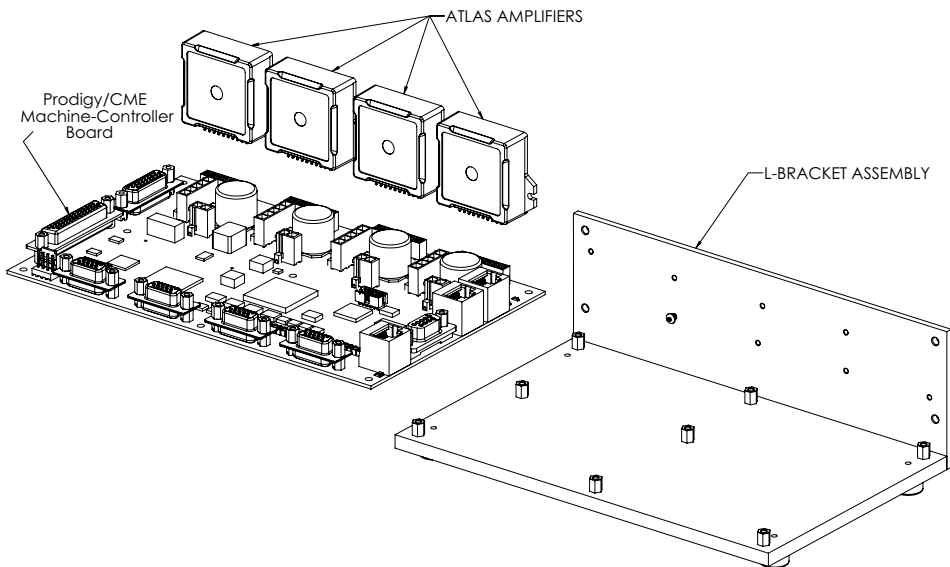
If it is desired that different Atlas amplifiers be installed in the machine controller board, for example low or medium power units, then assembling a DK setup from scratch may be a good option. To do this the machine controller board, desired Atlas units, and miscellaneous parts kit (P/N PRK33MK00004) are ordered separately and the developer kit setup is assembled by the user. See [Appendix A, Installation](#), for detailed instructions on how to assemble the developer kit.

For more information on Atlas amplifiers refer to the *Atlas Digital Amplifier User Manual*.

### 1.1.2 Developer Kit Elements

A typical machine controller developer kit setup consists of these elements:

- Prodigy/CME Machine-Controller board
- L-bracket assembly consisting of a base plate with PCB mounting posts attached to a vertical plate for Atlas mounting
- Up to 4 Atlas Digital Amplifier units
- Miscellaneous cables & accessories to facilitate connection to the application motor hardware and to provide various other electrical connections
- Software SDK (Software Development Kit) and PDFs for all documentation



**Figure 1-2:**  
**Prodigy/CME**  
**Machine-**  
**Controller**  
**Developer Kit**  
**Elements**

### 1.1.3 Developer Kit Cables & Accessories

All Prodigy/CME Machine-Controller Developer Kits come with the following cables and accessories. The quantity of each cable or accessory included with the DK depends on the developer kit P/N ordered. For example if the four axis developer kit was ordered it contains four MC-HW-05 DB15 terminal screw expander boards, one for each axis.

Component Part Number	Description
Cable-5001-01	2-signal HV Power supply cable. This stub cable provides power to the Atlas unit for each axis. This cable may plug into the Prodigy/CME Machine-Controller board's J5 – J8 connectors.
Cable-5002-01	5-signal Motor Drive cable. This stub cable connects to the Motor Drive Connectors.
Cable-RJ45-02	RJ45 to RJ45 connector. This cable connects to the board's Ethernet or CANbus connectors.
Cable-4355-01	Bifurcated serial cable that connects to J23 Serial port and provides two RS232 serial port connections.
Cable-USB-DB9	USB to DB9 serial cable
TRM-RJ45-02	120 ohm CANbus terminator
MC-HW-03	DB44 terminal screw expander board.
MC-HW-04	DB26 terminal screw expander board.
MC-HW-05	DB15 terminal screw expander board.

## 1.2 Guide to this Manual

This manual is designed to help you get your motion hardware setup connected and operating with the Prodigy/CME Machine-Controller board. In addition, this manual shows you how to continue with development of your application by further exercising the connected motor hardware, optimizing the motion system control parameters, and developing software for the production control application.

Here is a summary of the content in the remaining chapters of this manual:

[Chapter 2, Quick Start Guide](#), provides instructions on connecting and verifying proper functioning of your motion hardware with the Prodigy/CME Machine-Controller board.

[Chapter 3, \*Going Further with Pro-Motion\*](#), describes the most frequently used features of Pro-Motion. You will find the content in this chapter helpful if you are new to Pro-Motion and PMD's Magellan Motion Control IC.

[Chapter 4, \*Communication Port Connections\*](#), shows how to set up additional communication options for the Prodigy/CME Machine-Controller, and shows how to configure and connect to additional PMD controllers (if any will be used) so that Pro-Motion can communicate with all of the controllers in your motion system.

[Chapter 5, \*Software Development\*](#), provides an overview of how to develop software application code for your PMD-based control system.

[Chapter 6, \*Electrical Reference\*](#), provides reference information that you may find useful in connecting the machine controller board to your motion hardware.

## 1.3 Software Installation

The software distribution for the machine controller DK is downloaded from the PMD website at the URL: <https://www.pmdcorp.com/resources/software>.

All software applications are designed to work with Microsoft Windows.

To install the software:

- 1 Go to the Software Downloads section of PMD's website located at <https://www.pmdcorp.com/resources/software> and select download for "Developer Kit Software".
- 2 After selecting download you will be prompted to register your DK, providing the serial # for the DK and other information about you and your motion application.
- 3 After selecting submit the next screen will provide a link to the software download. The software download is a zip file containing various installation programs. Select this link and downloading will begin.
- 4 Once the download is complete extract the zip file. There is a *ReadMe.txt* file that may contain additional useful information. When ready execute the **Pro-Motion** install. Pro-Motion is a Windows application that will be used to communicate with and exercise your developer kit.
- 5 You can also extract the following SDK (Software Development Kit) used with the Prodigy/CME Machine-Controller.
  - **C-Motion PRP SDK** – An SDK for creating PC-based applications using .NET (C#, VB) programming languages and for PC-based or downloadable applications in C-language when using PMD products which have a /CME designation

The next few sections give more information on each of these items.

### 1.3.1 Pro-Motion

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

- Motion oscilloscope graphically displays motion control parameters in real-time
- AxisWizard to automate axis setup and configuration
- Project window for accessing motion resources and connections
- Ability to save and load settings

- Distance, time, and electrical units conversion
- Frequency sweep and bode plot analysis tools
- Motor-specific parameter setup
- Axis shuttle performs continuous back and forth motion between two positions
- C-Motion Engine console window
- C-Motion Engine user application code download

### 1.3.2 C-Motion

C-Motion provides a convenient set of callable routines comprising the C language code required for controlling Magellan ICs. C-Motion includes the following features:

- Magellan axis virtualization
- Ability to communicate to multiple PMD motion boards or modules
- Ability to communicate via PC/104 bus, serial, CANbus, Ethernet, SPI (Serial Peripheral Interface), or 8/16 bit parallel bus
- Provided as source code, allowing easy compilation & porting onto various run-time environments including a PC, microprocessor, embedded board, or C-Motion Engine
- Can be easily linked to any C/C++ application

There are three different versions of C-Motion; C-Motion Magellan, C-Motion PRP, and C-Motion PRP II. C-Motion Magellan is used with PMD products that utilize a direct Magellan or Juno formatted protocol. C-Motion PRP is used in systems that support both Magellan/Juno protocol device and PRP (PMD Resource Access Protocol) protocol devices. C-Motion PRP is also used in motion applications that will use the .NET (C#, VB) programming languages. C-Motion PRP II is used with ION/CME N-Series Digital Drives.

C-Motion Magellan is described in the *C-Motion Magellan Programming Reference*, C-Motion PRP is described in the *C-Motion PRP Programming Reference*, C-Motion PRP II is described in the *C-Motion PRP II Programming Reference*.

### 1.3.3 .NET Language Support

A complete set of methods and properties is provided for developing applications in Visual Basic and C# 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, such as Labview, but no special software support is provided.

Includes the following features:

- Magellan axis virtualization
- Ability to communicate to multiple PMD motion boards or modules
- Ability to communicate via PC/104 bus, serial, CAN, Ethernet, SPI, or 8/16-bit parallel bus
- Provided as a single DLL and Visual Basic .NET source code for easy porting onto various PC environments

## 1.4 Recommended Hardware

In addition to the machine controller DK itself, to build a complete functioning system the following hardware is recommended.

- Intel (or compatible) processor, 1 Gbyte of available disk space, 256 MB of available RAM, and a CD ROM drive. The supported PC operating systems are Windows XP, Vista, Windows 7, and Windows 8.
- One to four step motors, DC Brush, or Brushless DC motors. For position control DC Brush and Brushless DC motors must have encoder feedback, while for step motors encoder feedback is optional.
- Cables as required to connect to motor peripherals such as limit switches.
- DC Power Supply with a voltage range of 12V to 56V with sufficient current capacity for the DK setup application. Note that if the setup uses low or medium power Atlas units the supply range is 12V to 48V.

## 2. Quick Start Guide

### ***In This Chapter***

- ▶ Step #1 — Configuring the Board
- ▶ Step #2 — Making Motion Hardware Connections
- ▶ Step #3 — Applying Power
- ▶ Step #4 — First Time System Verification
- ▶ Next Steps

Here are the steps you will execute to set up the Prodigy/CME Machine-Controller so that it can control your motion hardware successfully. If you haven't already installed the software you should do this now. See [Section 1.3, "Software Installation,"](#) for instructions on installing the software.

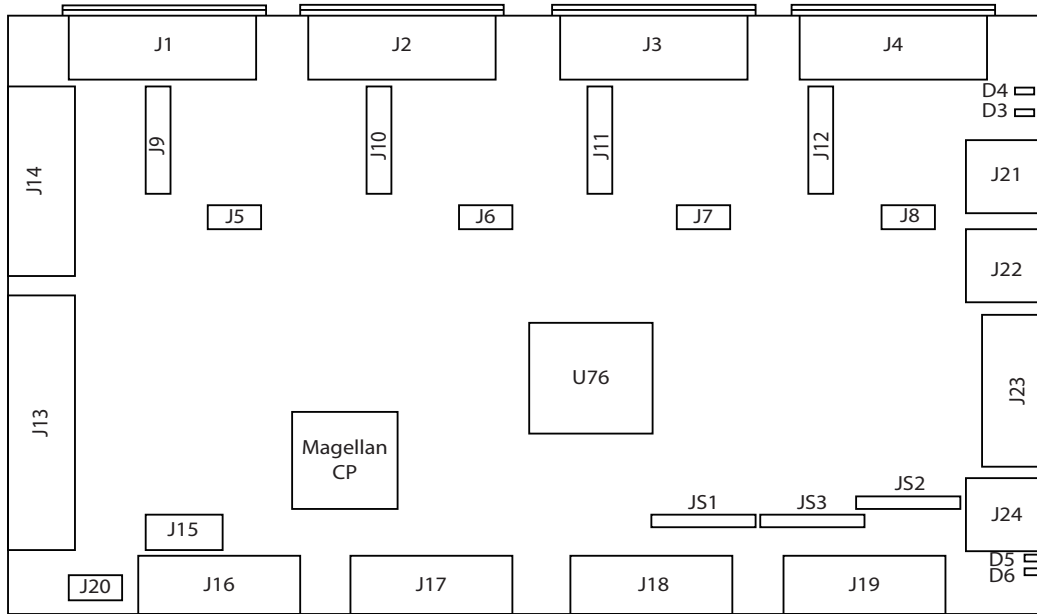
- Step 1** The first step is to configure the board. See [Section 2.1, "Step #1 — Configuring the Board,"](#) for a description.
- Step 2** Next connect your system's encoder(s), motion peripherals, motors, power supply, and PC to the machine controller board. See [Section 2.2, "Step #2 — Making Motion Hardware Connections,"](#) for details.
- Step 3** Next you will provide the board with power. See [Section 2.3, "Step #3 — Applying Power,"](#) for more information.
- Step 4** The final step is to perform a functional test of the finished system. See [Section 2.4, "Step #4 — First Time System Verification,"](#) for a description of this procedure.

Once these steps have been accomplished setup is complete and the board and connected motion hardware are ready for operation.

## 2.1 Step #1 — Configuring the Board

[Figure 2-2](#) illustrates the locations of the principal components of the Prodigy/CME Machine-Controller board.

**Figure 2-1:**  
Components  
and Layout,  
Front of Board



### 2.1.1 Resistor Packs

The user-settable components of the board are listed in the following table:

Item	Setting	Description
Resistor packs JS1, JS2, JS3	Installed; this is the default setting of resistor packs JS1 - JS3	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 inputs may be connected differentially, with two wires for QuadA, QuadB, and Index signals, or with just one wire per signal.

When both single-ended and differential encoders are used on the same board a special arrangement of the connections and resistor packs is needed. Refer to [Section 6.1.1.3, “Using Both Single-Ended and Differential Encoder Connections.”](#) for details.

## 2.2 Step #2 – Making Motion Hardware Connections

The next few sections detail how to make the needed connections between the Prodigy/CME Machine-Controller and your motion hardware, PC, and power supply.

### 2.2.1 Encoder & Motion Peripheral Connections

The following table summarizes the encoder and motion peripheral signal connections to be made to the Prodigy/CME Machine-Controller. J16, J17, J18, and J19 are the connectors for axes 1, 2, 3, and 4 respectively. They are high density female DB15s and may be located on the board using [Figure 2-1](#). Between one and four axes may be connected, depending on the application.



Encoders are required for controlling the position of DC Brush and Brushless DC motors. Encoders are optional for step motors. Quadrature encoders can use a differential wiring scheme where each signal (**A**, **B**, and **Index**) uses a positive (+) and negative (-) connection, or they can use a single-ended scheme with a single connection per signal. If available use of differential connections is highly recommended. If single-ended encoders are used they are connected to the + differential signal. In addition to **QuadA**, **B**, and **Index** signals encoders also typically require 5V and GND connections.

Hall signals are used with Brushless DC motors only. Although they are not required, they should be used if available. Home and position limit sensors are optional.

You can use the provided DB15 terminal screw expander boards to conveniently connect your motion hardware to these connector pins. There is one connector for each axis, and the signals are identical for all axes:

Pin #	Signal Name	Description
<b>J16, J17, J18, J19 - Axis 1-4 Feedback Connectors</b>		
1	QuadA+	Differential A+ quadrature input. (optional for step motor axes)
2	QuadA-	Differential A- quadrature input. (optional for step motor axes)
3	QuadB+	Differential B+ quadrature input. (optional for step motor axes)
4	QuadB-	Differential B- quadrature input. (optional for step motor axes)
5	GND	This is the preferred ground connection for the quadrature and Index signal inputs
6	Index+	Differential Index+ quadrature input. (optional for step motor axes)
7	Index-	Differential Index- quadrature input. (optional for step motor axes)
8	HallA	Hall signal input phase A. not used for DC brush or step motors
9	HallB	Hall signal input phase B. not used for DC brush or step motors
10	HallC	Hall signal input phase C. not used for DC brush or step motors
11	Home	Home signal input (optional)
12	PosLim	Positive position limit input (optional)
13	NegLim	Negative position limit input (optional)
14	+5V	+5V power output which may be used to power the motor's encoder circuitry
15	GND	This is the preferred ground connection for the +5V output along with the Hall, home, and limit input signals.

## 2.2.2 Motor Coil Connections

The following table summarizes the connections from the Prodigy/CME Machine-Controller to the coils of your motor. J9, J10, J11, and J12, known as Motor Drive Connectors, are the motor coil connectors for axes 1, 2, 3, and 4 respectively. They are male Molex mini-Fit Plus style connectors and may be located on the board using [Figure 2-1](#). Each motor drive connector is designed to connect to all available motor types; Brushless DC, DC Brush, step motor. There are four coil connections for each axis and a shield connection. Not every motor type uses all four coil connections.

You may find it convenient to use the provided 5-signal motor drive stub cable sets to connect to your motors. There is one connector for each axis, and the signals are identical for all axes:

Pin #	Signal Name	Description
<b>J9, J10, J11, J12 - Axis 1-4 Motor Drive Connectors</b>		
1	Motor A	A motor drive. Used with all motor types.
2	Motor B	B motor drive. Used with all motor types
3	Motor C	C motor drive. Used with all motor types except DC brush
4	Motor D	D motor drive. Used with step motors only
5	Shield	Connection to motor ground. This shield connection is recommended, however is not required for most motor setups.

The table below provides a quick reference for how motor coils should be connected for each motor type:

Motor type	Motor Drive Signal	Motor Coil Connections
Brushless DC	Motor A	A winding connection
	Motor B	B winding connection
	Motor C	C winding connection
	Shield	(optional) motor shield connection
DC Brush	Motor A	+ winding connection
	Motor B	- winding connection
	Shield	(optional) motor shield connection
Step motor	Motor A	phase A+ winding connection
	Motor B	phase A- winding connection
	Motor C	phase B+ winding connection
	Motor D	phase B- winding connection
	Shield	(optional) shield connection



Shield connections to the motor are not required but are recommended. Not connecting the shield signal for each axis may result in increased EMI (electromagnetic interference), and reduced immunity to ESD (electro static discharge).

## 2.2.3 Communication Connections

While the Prodigy/CME Machine-Controller board can communicate using Ethernet, CANbus, and one of two serial modes (RS232 and RS485), in this first-time installation we will set up the board for RS232 communications. To set up the board for operation in other communication modes, see [Chapter 4, Communication Port Connections](#).

To connect via RS232 use the included USB to DB9 serial cable. This serial cable should be connected to the board's J23 Serial Connector while the opposite end should be connected to one of your computer's USB ports. Figure 2-1 can be used to locate the J23 connector.

## 2.2.4 Power Connections

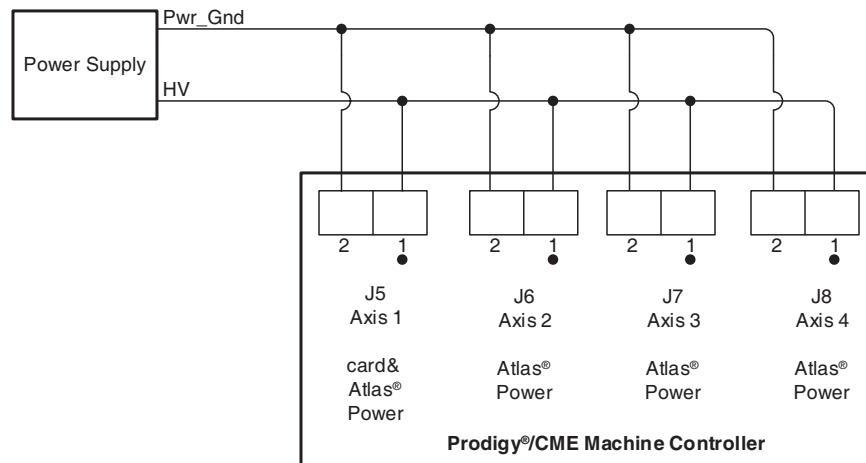


Figure 2-2:  
Power  
Connections

The table below summarizes the power connections from the power supply to the Prodigy/CME Machine-Controller. Each motor driven by an Atlas amplifier has its own power provided to it. Although most applications will power each axis at the same voltage from a single common supply, different voltages may be connected for different axes if desired.

All connections are made through the Motor Power Connectors, which are male two-conductor Molex Mini-Fit Plus 2-style connectors. You may find it convenient to use the provided two-pin power stub cable sets to connect the power supply to these power connectors.

Axis 1 motor power should always be provided whether or not an Atlas is installed at that axis. Axis 1 is the power connection from which the board logic power is derived using onboard DC-DC converters. Note that under some circumstances it may be desirable to provide just the +5V board logic power. This can be done via connector J20, however power should only be applied at J20 if no power is applied to Atlas #1. For more information on J20 refer to [Section 6.2.2.5, “+5V Power Connector.”](#)

Signal Name	Pin	Description
<b>J5, J6, J7, J8 - Axis 1-4 Power Connections</b>		
HV1	J5, 1	DC power to the board and Axis 1 Atlas amplifier. Voltage range is 12-56V if high power Atlas installed, 12-48V if low or medium power Atlas installed.
GND	J5, 2	Ground
HV2	J6, 1	DC power to the Axis 2 Atlas amplifier. Voltage range is 12-56V if high power Atlas installed, 12-48V if low or medium power Atlas installed.
GND	J6, 2	Ground
HV3	J7, 1	DC power to the Axis 3 Atlas amplifier. Voltage range is 12-56V if high power Atlas installed, 12-48V if low or medium power Atlas installed.
GND	J7, 2	Ground
HV4	J8, 1	DC power to the Axis 4 Atlas amplifier. Voltage range is 12-56V if high power Atlas installed, 12-48V if low or medium power Atlas installed.
GND	J8, 2	Ground

## 2.3 Step #3 – Applying Power

Once you have made your motion hardware, communication, and power connections, hardware installation is complete and the board is ready to be provided with power. As indicated in [Section 2.2.4, “Power Connections,”](#) the input voltage range is 12 – 56V or 12 – 48V depending on the Atlas amplifier types installed.

When power is applied, the Prodigy/CME Machine-Controller’s green power LED should light, and the green “HB” (Heart Beat) LED should blink slowly. These LEDs are locatable as D3 and D4 using [Figure 2-1](#). If the LEDs do not light, recheck the connections.

After power up the motors should remain stationary. If the motors move or jump, power down the board and check the motor and encoder connections. If anomalous behavior is still observed, call PMD for assistance.

## 2.4 Step #4 – First Time System Verification

The first time system verification procedure summarized below has two overall goals. The first is to connect the machine controller board with the PC that is being used so that they are communicating properly, and the second is to initialize each axis of the system and bring it under stable control capable of making trajectory moves. While there are many additional capabilities that Pro-Motion and the machine controller board provide, these steps will create a foundation for further, successful exploration and development.

During this first time system setup you may find it useful to refer to the *Magellan Motion Control IC User Guide* to familiarize yourself with operation of the Magellan Motion Control IC, which lies at the heart of all PMD motion boards.

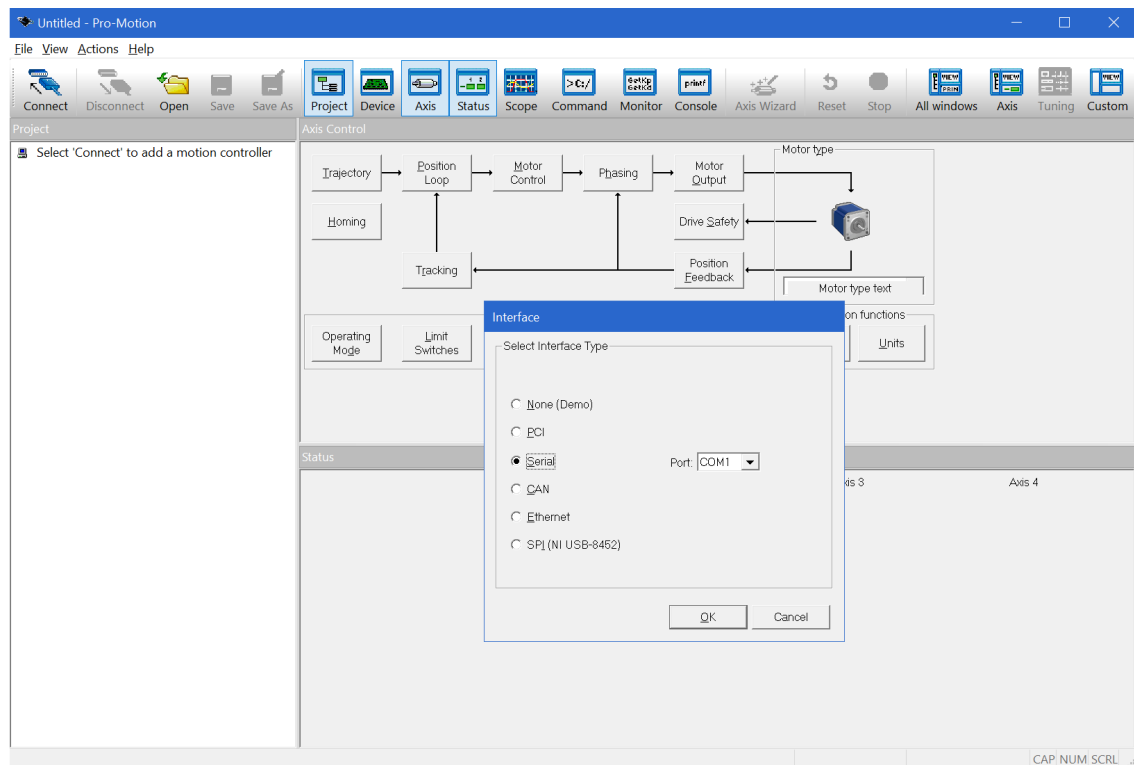
## 2.4.1 Establishing Communications

The first step in the first time system verification sequence is to establish serial communications, which is done as follows:

- 1 Make sure the machine controller board is powered and connected to the PC via the USB to DB9 serial cable.
- 2 Launch the Pro-Motion application.

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

The purpose of the Interface dialog box is to indicate to Pro-Motion how your PMD controller is connected to the PC. It provides various selectable communication options such as serial, CAN, Ethernet.



- 3 Click Serial and view the available COM ports listed in the Port field. If you know which COM port your USB serial cable is connected to, select it.

If you are not sure which of the listed COM ports is the correct one, you may use the following procedure:

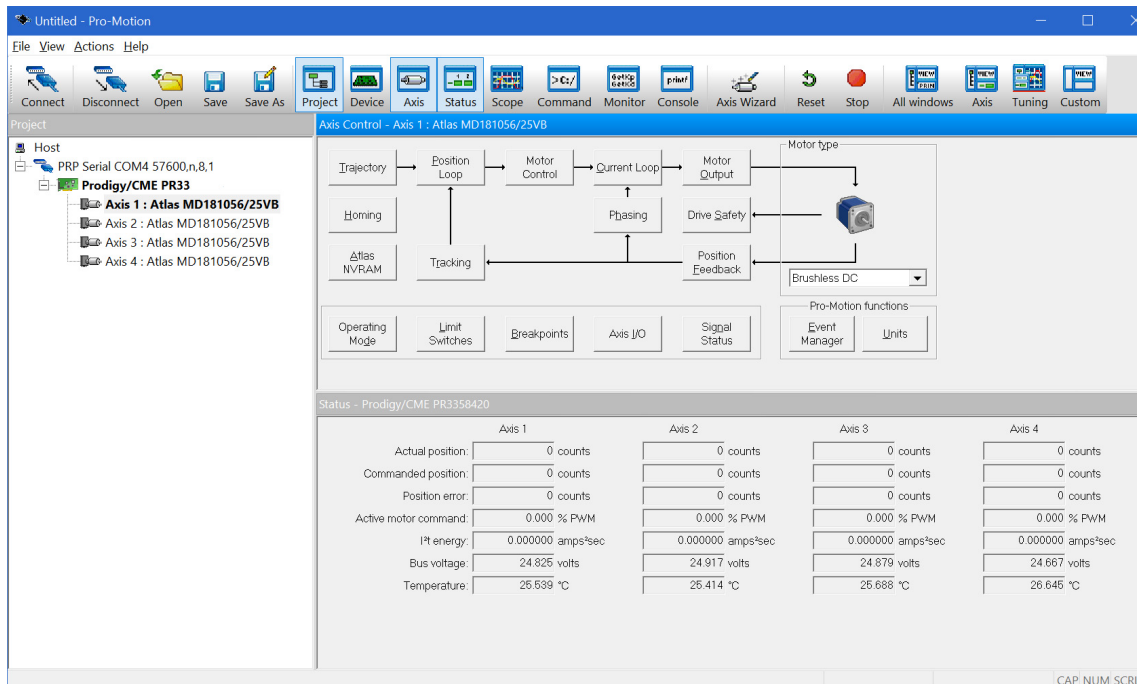
- First, unplug the serial cable from your USB port and exit the Interface dialog box. Now re-enter the Interface dialog box by clicking “Connect” which is an icon at the far left of the top icon bar. Select Serial, and view the COM port list. Record the list of COM ports.

- Next plug the serial cable back into the USB port and once again exit and re-enter the Interface dialog box. Select Serial and now when you view the COM port list you should see a new COM port listed. This is the COM port that is connected via the serial cable provided with the DK.
- Select this COM port and hit the OK button.

The Serial Port dialog box displays with default communication values of 57,600 baud, no parity, 1 stop bit, and point-to-point protocol.

- 4 Click OK without changing any of these settings.

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



If serial communications are not correctly established, a message appears indicating that an error has occurred. If this is the case, recheck your connections and repeat from step 1.

## 2.4.2 Running the Axis Wizard

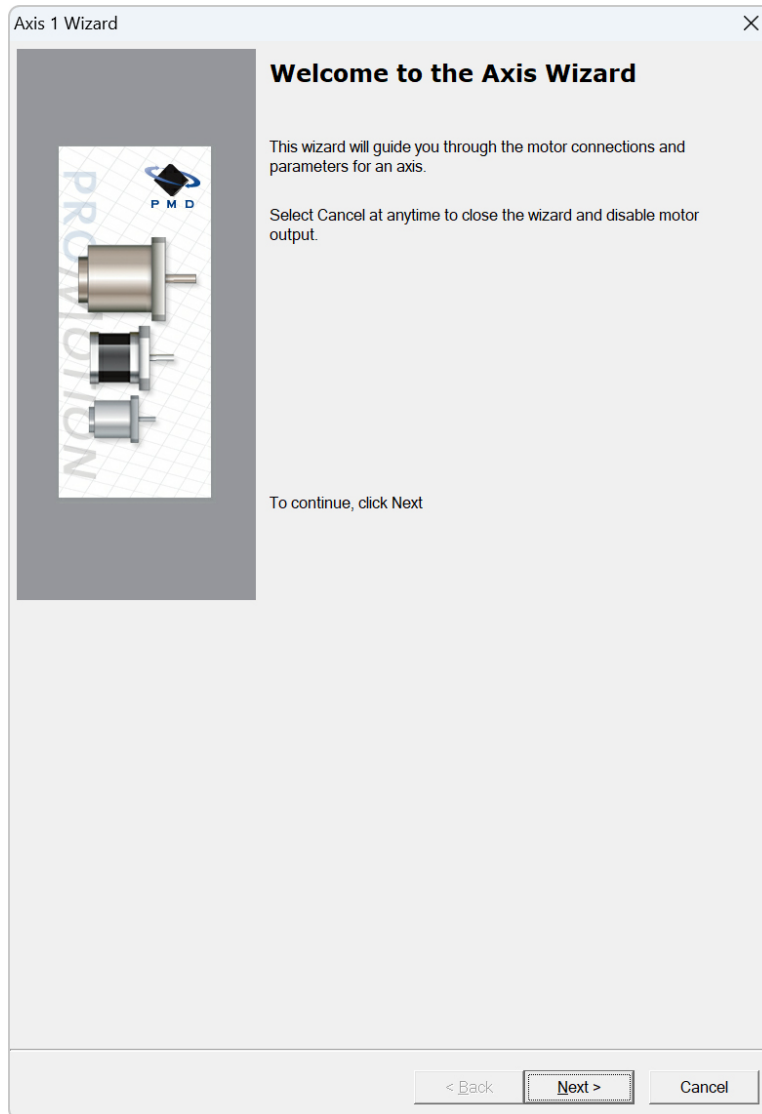
The next step is to initialize each axis in your system, thereby verifying correct motor connections, and other connections. All of this can be conveniently accomplished using Pro-Motion's Axis Wizard.

Before selecting the Axis Wizard icon however you should select the axis # to initialize. This is accomplished via the Project window, which is located to the left, by selecting the desired axis. After clicking on the desired axis it will become highlighted and the Axis Control window title will change to reflect the newly selected axis #. If the axis you want to operate on is already highlighted there is no need to select the axis.

Once the desired axis to initialize has been selected you should:

- 1 Click the Axis Wizard toolbar button which is located right of center in the row of icons toward the top of the window.

The Axis Wizard initialization window appears.

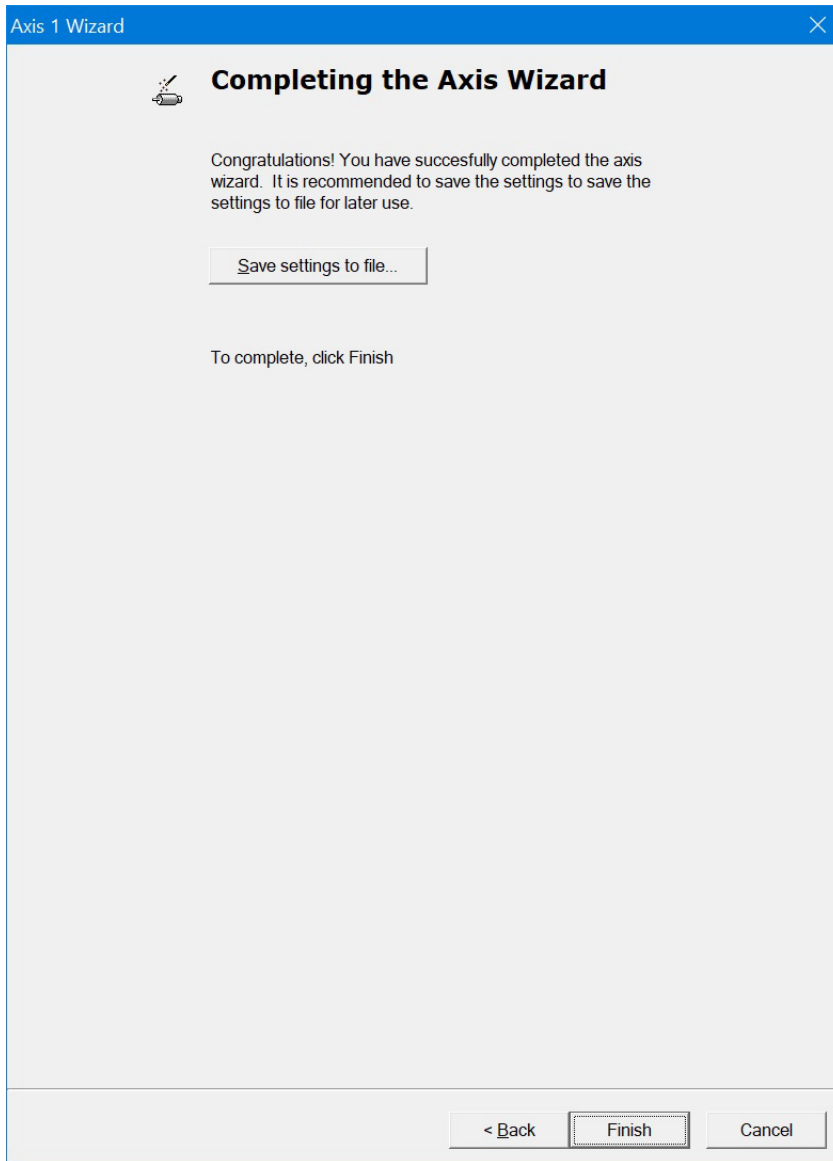


- 2 Click Next and follow the Axis Wizard instructions for each page of the axis setup process. A typical axis wizard sequence takes 5-10 minutes depending on the motor type and number of motion peripherals such as limit switches your system uses. Although rare, if you have any problems while going through the Axis Wizard you may find it helpful to view [Section 3.12, “Troubleshooting Suggestions.”](#)

The Atlas units installed in the developer kit setup are multi-motor type which allows the motor type to be programmed electronically. In fact this selection is one of the very first screens you will encounter in the Axis Wizard sequence, asking you to choose between Brushless DC, DC Brush, and step motor types.

The motor choice determines the various Axis Wizard screens you will step through to initialize and check out the axis setup. For example only Brushless DC motors will show screens related to Hall signal and commutation verification.

- The last Axis wizard screen allows you to save the various control parameters you have specified while in the Axis Wizard. This is convenient because it allows you to quickly load your control settings and make them active without the need to re-run the Axis Wizard.



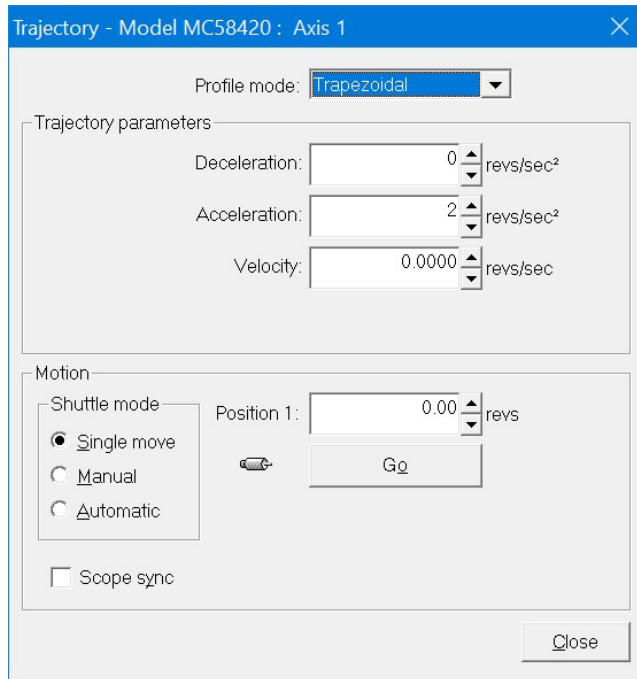
- To save the Axis Wizard settings select "Save settings to file..." and specify a destination directory and file name. The Axis Wizard will create the file with your parameters loaded into it. For more information on saving and restoring configuration settings see [Section 3.9, "Project Configuration Save & Restore."](#)
- When you have specified a file name and saved your settings select Finish at the bottom of the screen. You will now be returned to the main Pro-Motion screen.

### 2.4.3 Exercising The Motor

The next step is to exercise the motor and verify that it is reacting correctly to programmed commands.

This is accomplished by having the motor make a simple point-to-point move. To perform a point-to-point move:

- 1 Click the Trajectory button in the Axis Control window. The Trajectory dialog box appears, shown below.



- 2 In the Profile mode list, select Trapezoidal, and in the Motion box select Single move.
- 3 Enter motion profile settings for deceleration, acceleration, velocity, and destination position (Position 1) that are safe for your system and will demonstrate proper motion. The units of these parameters should match the units you selected earlier in the Axis Wizard setup process. If you would like to change the units you can do this by going to the Axis Control window and clicking the Units box which is to the far lower right. You should then exit and re-enter the Trajectory dialog box for the units change to be visible.
- 4 Click Go and confirm that the motion occurred in a stable and controlled fashion.

Congratulations! Quick setup for this axis is now complete.

At this point you have the choice of exploring Pro-Motion features and further exercising the motor you have just initialized, or selecting a new axis to initialize via the Axis Wizard. To select a new axis to initialize follow the instructions in [Section 2.4.2, “Running the Axis Wizard.”](#) Select the next axis # to initialize in the Project window, click the Axis Wizard button, and repeat setup for this new axis using the Axis Wizard.

## 2.5 Next Steps

When ready, to continue exercising the motion system and to begin optimizing and developing your control application refer to the subsequent chapters of this manual. Here is an overview of these chapters:

[Chapter 3, \*Going Further with Pro-Motion\*](#), shows you the most frequently used features of Pro-Motion. You will find the content in this chapter helpful if you are new to Pro-Motion and PMD’s Magellan Motion Control IC.

[Chapter 4, \*Communication Port Connections\*](#), shows you how to set up alternate communication channels for the PMD controller you installed in this quick setup guide, and shows you how to connect additional PMD controllers (if any will be used) so that Pro-Motion can communicate with all PMD controllers in the motion system.

[Chapter 5, \*Software Development\*](#), provides an overview of how to develop software application code to control your PMD-based control system.



[Chapter 6, \*Electrical Reference\*](#), provides reference information that you may find useful in connecting the machine controller board to your motion hardware. For additional information on the machine controller board you can also refer to *Prodigy/CME Machine-Controller User Guide*.

*This page intentionally left blank.*

# 3. Going Further with Pro-Motion

## In This Chapter

- ▶ Pro-Motion Screen Layout
- ▶ Project Window
- ▶ Device Control Window
- ▶ Axis Control Window
- ▶ Status Window
- ▶ Monitor Window
- ▶ Command Window
- ▶ Scope Window
- ▶ Project Configuration Save & Restore
- ▶ Configuration Export to C-Motion
- ▶ Pro-Motion Application Notes
- ▶ Troubleshooting Suggestions

In this chapter we provide more information on Pro-Motion to help familiarize you with its most commonly used features.

## 3.1 Pro-Motion Screen Layout

The screenshot displays the Pro-Motion software interface. The main window is titled "Axis Control - Axis 1 : Atlas MD181056/25VB". The interface is divided into several sections:

- Menu Bar:** File, View, Actions, Help.
- Toolbar:** Connect, Disconnect, Open, Save, Save As, Project, Device, Axis, Status, Scope, Command, Monitor, Console, Axis Wizard, Reset, Stop, All windows, Axis, Tuning, Custom.
- Project Tree:** Host, PRP Serial COM4 57600,n,8,1, Prodigy/CME PR33, Axis 1 : Atlas MD181056/25VB, Axis 2 : Atlas MD181056/25VB, Axis 3 : Atlas MD181056/25VB, Axis 4 : Atlas MD181056/25VB.
- Axis Control Diagram:** A block diagram showing the control loop: Trajectory → Position Loop → Motor Control → Current Loop → Motor Output → Motor type (Brushless DC). Other components include Homing, Atlas NVRAM, Tracking, Phasing, Drive Safety, Position Feedback, Operating Mode, Limit Switches, Breakpoints, Axis I/O, Signal Status, Event Manager, and Units.
- Status Table:** A table showing real-time data for four axes.

	Axis 1	Axis 2	Axis 3	Axis 4
Actual position:	0 counts	0 counts	0 counts	0 counts
Commanded position:	0 counts	0 counts	0 counts	0 counts
Position error:	0 counts	0 counts	0 counts	0 counts
Active motor command:	0.000 % PWM	0.000 % PWM	0.000 % PWM	0.000 % PWM
I <sup>t</sup> energy:	0.000000 amps*sec	0.000000 amps*sec	0.000000 amps*sec	0.000000 amps*sec
Bus voltage:	24.825 volts	24.917 volts	24.879 volts	24.667 volts
Temperature:	25.539 °C	25.414 °C	25.688 °C	26.645 °C

The screen capture above shows Pro-Motion as it appears after first launching and connecting to a PMD controller. In this particular screen capture the connected PMD controller is a four axis Prodigy/CME Machine-Controller. For information on connecting to a PMD controller after launching Pro-Motion see [Section 2.4.1, “Establishing Communications.”](#)

Pro-Motion follows the general form of Windows-based applications with a menu at the very top of the application window. In the case of Pro-Motion the available menu functions are *File*, *View*, *Actions*, and *Help*. In addition there is a tool bar with icons allowing the user to access the most commonly used Pro-Motion functions with a single click. Here is a description of these clickable icons in the order that they appear from left to right and grouped by function:



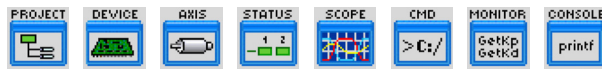
#### Connect, Disconnect

These toolbar icons let you connect or disconnect to PMD controllers in your motion hardware setup.



#### Open, Save, Save As

These toolbar icons let you call up and save your setup configuration. [Section 3.9, “Project Configuration Save & Restore.”](#) describes these “project file” mechanisms in more detail.



#### Project, Device Control, Axis Control, Status, Scope, Command, Monitor, and CME Console

Each member in this group of icons represents a Pro-Motion window that, when clicked, opens if not yet being displayed or closes if being displayed. We will discuss the functions provided by many of these windows later on.



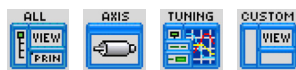
#### Axis Wizard

This icon starts the Axis Wizard. The Axis Wizard is the recommended way to establish and verify connections between the PMD controller and each axis of the motion hardware setup. See [Section 2.4.2, “Running the Axis Wizard.”](#) for more on the Axis Wizard.



#### Reset, Stop

The Reset button will immediately reset the currently selected device in the Project window. The Stop button will immediately stop the motion of the axis selected in the Project window.

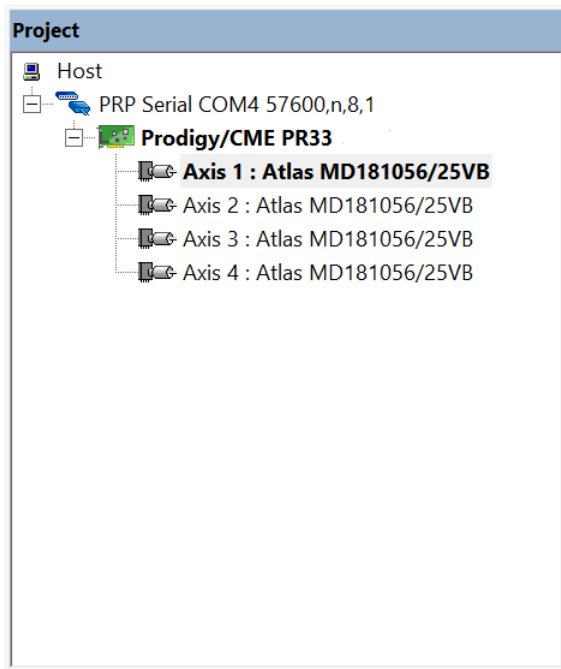


#### All, Axis, Tuning, Custom

This group of icons controls how the Pro-Motion windows are arranged. They provide convenient pre-programmed arrangements of windows designed to make specific tasks easier. The default arrangement is Axis.

Windows can also be arranged manually and saved as a custom window arrangement. To store the arrangement of windows in your Pro-Motion session use the *View/Save Custom View* menu function at the very top of the Pro-Motion screen. Thereafter, selecting the Custom icon will present the windows in this saved custom view scheme.

## 3.2 Project Window



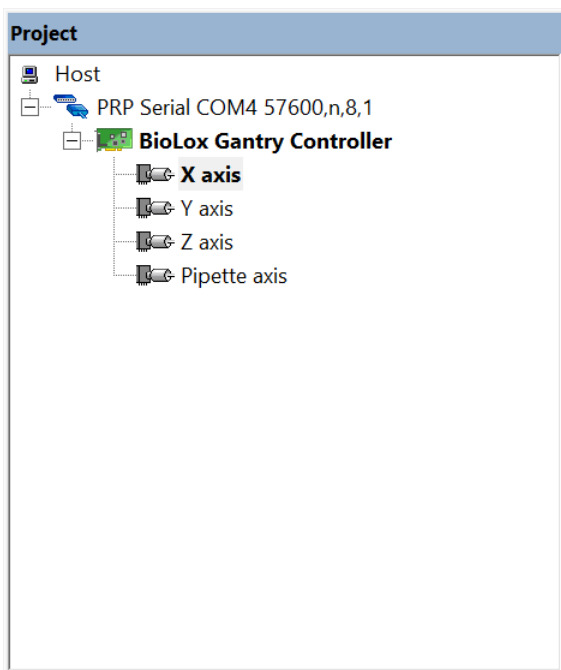
The Project window shows all PMD controllers presently connected to Pro-Motion. It presents these connections using a Windows Tree View scheme.

In addition to displaying an icon and part number for each connected PMD controller it displays a 'motor' icon for each axis within that controller. So a Prodigy board with four active axes would have four such motor icons displayed. This is shown in the screen capture above. Single axis devices such as an N-Series ION drive would have two such icons displayed, one for the primary and one for the auxiliary encoder input. For Magellan axes that use an Atlas amplifier a special version of the motor icon is displayed and the part number of the attached Atlas is displayed next to the motor icon.

In addition to displaying the addressable PMD devices in the present Pro-Motion session, the Project window is used to select which axes are actively being processed by Pro-Motion windows. For example in the screen capture above, axis #1 is highlighted, which means axis-specific windows such as the Axis Control window will program the settings for axis #1. To change the currently selected axis simply click on the desired axis icon in the Project window.

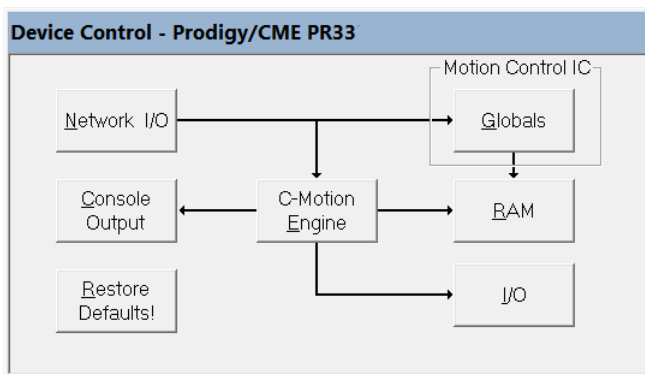
### 3.2.1 Device & Axis Label Customization

A very useful feature of the Project window is that the device and axis labels generated automatically by Pro-Motion can be customized to suit the application. This is shown in the screen capture below, where the Prodigy/CME Machine-Controller labels from the previous Project window have been changed to reflect their purpose and function.



To change the device or axis labels slowly click twice on the existing text, and then type in your desired new entry.

## 3.3 Device Control Window



The Device Control window allows you to view and update 'device level' parameters. Devices are PMD controllers such as motion IC developer kit boards, Prodigy boards, and ION Drives. Different products have different device-level functions available and the Device Control window displays the available functions for the selected device as clickable boxes. To select a particular device from the Project window click any axis controlled by that device. For example for a four-axis Prodigy/CME Machine-Controller clicking the icon for any of its four axes will result in the machine controller being selected in the Device Control window.

The list below briefly describes the clickable function boxes you may see in the Device Control window:

*Network I/O* – The Network I/O module of the Device Control window allows you to view and set various parameters for each of the communication ports supported by the connected-to device.

*Motion Control IC Globals* - Clicking this box lets you set the Magellan or Juno IC's cycle time and lets you view various characteristics of the motion IC such as the IC family name, supported motor type(s), number of supported axes, and the version #.

*NVRAM* – Some Motion Control ICs support non-volatile memory which can contain initialization command sequences. For those products this function allows you to view, erase, and download script file content or the current Pro-Motion configuration to the Magellan IC's NVRAM.

*RAM* – Allows you to view the size of the PMD controller's RAM used for motion trace and User Defined Profile Mode storage. You can also change the usable size of the RAM with this function.

*Analog Input* – Allows you to view current reading(s) from the PMD controller's general purpose analog input function.

*C-Motion Engine* – This function allows you to view, erase, and download a .bin user code memory image to the C-Motion Engine. In addition this function allows you to manually reset, start, or stop code execution, and set whether user code begins execution automatically upon power-up or only after manual start.

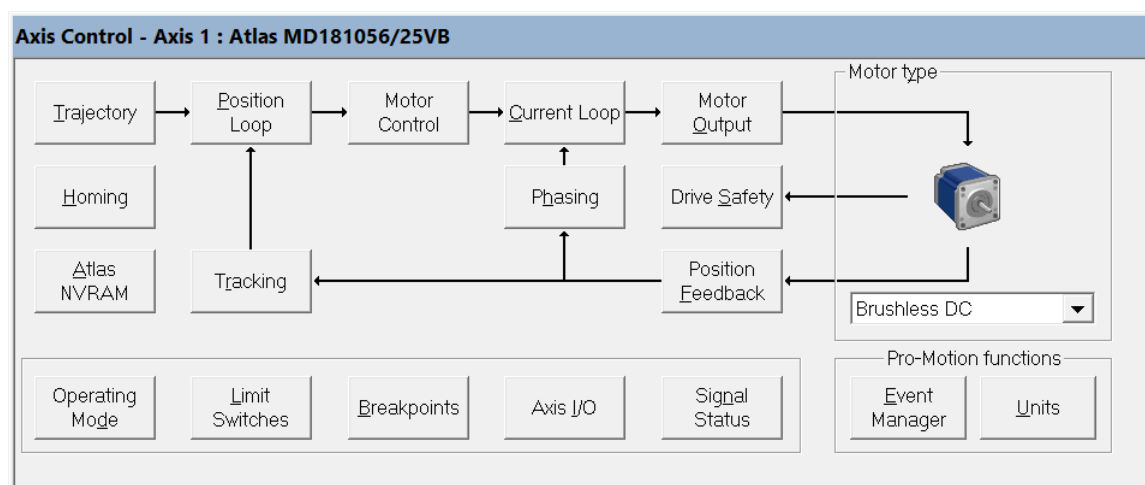
*Console* – This function lets you specify the console communication channel and parameters used in connection with user code running on the C-Motion Engine.

*Restore Defaults* – This function reverts the PMD controller's parameters to their default conditions. Any changes that may occur as a result of this operation occur in the device's NVRAM. Only after a reset or power cycle will NVRAM values become the active settings used by the PMD controller. Particular care should be taken when using this function if communication parameters have previously been altered because restoring them to their default values may mean that Pro-Motion no longer uses the correct settings to communicate with the PMD controller. For a detailed list of NVRAM-stored defaults that may be affected by this operation refer to the **DeviceSetDefault** command description in the user manual for the product you are using.

*Digital I/O* – This module allows device-level digital I/O registers to be read and set by the user.

## 3.4 Axis Control Window

The Axis Control window allows you to view and update motion control parameters for the axis selected in the Project window. Each selectable box (technically these are Windows buttons) in the Axis Control window results in a dialog box being opened letting you access a sub-set of the motion control functions provided.



The Axis Control window presents these selectable boxes such that the overall control flow for the motor type selected is evident. For example the boxes and control flow arrows displayed for a Brushless DC motor are different than for a step motor reflecting the fact that Brushless motors are controlled differently than step motors.

Some of the boxes at the bottom and left hand side of the Axis Control window are not connected via the control flow arrows. These provide access to motion control settings such as for limit switches, breakpoints, axis I/O, signal status, event management, units, homing, and Atlas NVRAM (for axes which use an Atlas amplifier).

The box labeled 'Operating Mode' provides control of whether major axis control modules are active. These modules are trajectory, position loop, current loop, and motor output. While normally enabled, there may be circumstances, for example if a motion error occurs, where some modules will get disabled for safety reasons and may need to be manually re-enabled.

Underpinning the control flow arrows, selectable boxes, and Operating Mode status in the Axis Control window is the control architecture of PMD's Magellan Motion Control IC, which is the motion controller at the heart of all PMD position control products including Magellan Motion Control IC DK boards, Prodigy boards, and ION drives. The reference manual that describes this is the *Magellan Motion Control IC User Guide*. For example if you want to know what trajectory profile modes are available and exactly how they function this manual contains that information. The same applies for the other selectable boxes and associated functions within the Axis Control window.

## 3.5 Status Window

The screen capture below shows the Status window for a Prodigy/CME Machine-Controller, which can provide up to four axes of control. The Status window displays all axes for the device selected in the Project window. For example if one of the axes in a four-axis Prodigy/CME Machine-Controller is selected all four axes of that controller will display in the Status window.

Status - Prodigy/CME PR33				
	Axis 1	Axis 2	Axis 3	Axis 4
Actual position:	0.000 revs	0 revs	0 mm	0 mm
Commanded position:	0.000 revs	0 revs	0 mm	0 mm
Position error:	0.000 revs	0 revs	0 mm	0 mm
Active motor command:	0.000 % PWM	0.000 % PWM	0.000 % PWM	0.000 % PWM
I <sup>2</sup> t energy:	0.000000 amps <sup>2</sup> sec	0.000000 amps <sup>2</sup> sec	0.000000 amps <sup>2</sup> sec	0.000000 amps <sup>2</sup> sec
Bus voltage:	24.767 volts	24.868 volts	24.860 volts	24.599 volts
Temperature:	32.465 °C	32.414 °C	33.703 °C	33.730 °C

For each axis of the selected PMD controller the value of several parameters are continuously displayed in the Status window. Note that some PMD products, particularly those without an amplifier, will not display all of these parameters. Here are brief descriptions of the displayed parameters:

*Actual position* – This is the actual position of the motor as measured by an encoder or by Hall sensors if Halls are programmed to provide position feedback.

*Commanded position* – This is the instantaneous commanded position from the trajectory generator.

*Position Error* – This parameter shows the difference between the commanded and the actual position. For servo-controlled motors (DC Brush and Brushless DC) this value measures how accurately the position loop is maintaining the commanded position. For step motors, if an encoder is present, this represents the amount that the actual motor lags the commanded position. A positive value means the commanded position is greater than the actual position, and vice versa for a negative value.



*Active motor command* – This parameter shows the torque (current) command being sent to the current loop/amplifier. If there is no current loop present (or active), rather than units of amps this parameter will have units of % PWM (Pulse Width Modulation).

*Bus voltage* – For PMD products which include an amplifier function this parameter indicates the current measured value of the DC supply voltage.

*Bus current return* – For PMD products which include an amplifier function this parameter continually indicates the current flow from HV to ground in the amplifier's switching bridge. Note that this current value does not include current used to power internal logic of the PMD controller.

*Temperature* – For PMD products which include an amplifier function this parameter indicates the instantaneous temperature of the amplifier bridge.

The quantities displayed in the Status window are displayed using the user-selected units. For example in the screen capture above axes 1 and 2 use position units of revolutions, and axes 3 and 4 use units of millimeters. To change Pro-Motion display units for a particular axis select the 'Units' box in the Axis Control window.

### 3.5.1 Status Polling

By default Pro-Motion continually polls the device selected in the Project window so that the Status window shows up-to-date values for all axes of that device. In addition, regardless of what device is selected in the Project window, Pro-Motion regularly polls all connected devices to report on safety events such as motion error, overcurrent, etc...

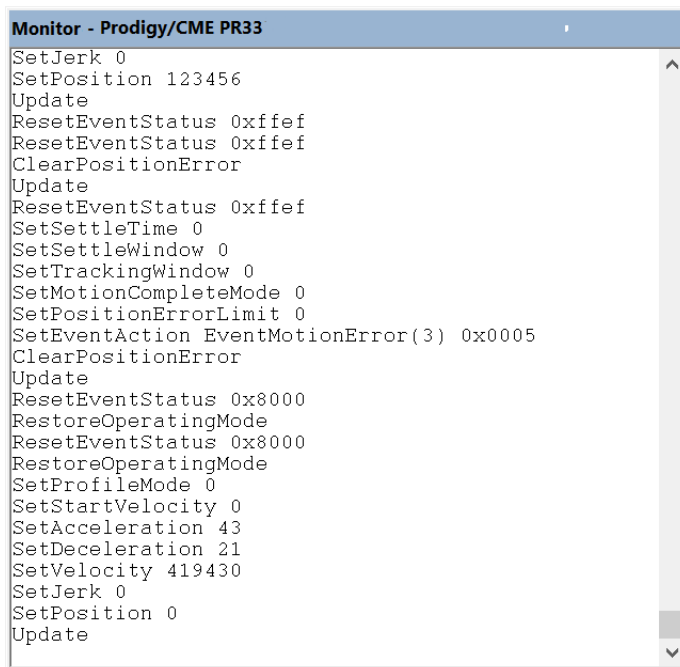
By default this background status polling is enabled, however there are situations where it may become disabled. The first is after some types of communication errors between Pro-Motion and the connected PMD controller. To avoid continually generating communication error warnings automatic polling may be disabled by Pro-Motion. If this happens Pro-Motion will display this change in a dialog box that pops up after the communication error.

Automatic polling may also be manually disabled by the user. This can be accomplished via the *View/Status Polling* menu function. The main reason for manually disabling background polling is when using the Monitor window. Background polling results in a large number of messages being displayed in the Monitor window, thereby making it much harder for the user to focus on specific transactions of interest. Disabling automatic polling solves this problem. Note that after status polling is disabled it should eventually be re-enabled to restore safety monitoring.

Disabling status polling may result in erroneous or unsafe operation of the motion controller being unreported by Pro-Motion. It is therefore always recommended that Pro-Motion status polling be enabled during regular operation of the motion controller.



## 3.6 Monitor Window



```

Monitor - Prodigy/CME PR33
SetJerk 0
SetPosition 123456
Update
ResetEventStatus 0xffef
ResetEventStatus 0xffef
ClearPositionError
Update
ResetEventStatus 0xffef
SetSettleTime 0
SetSettleWindow 0
SetTrackingWindow 0
SetMotionCompleteMode 0
SetPositionErrorLimit 0
SetEventAction EventMotionError(3) 0x0005
ClearPositionError
Update
ResetEventStatus 0x8000
RestoreOperatingMode
ResetEventStatus 0x8000
RestoreOperatingMode
SetProfileMode 0
SetStartVelocity 0
SetAcceleration 43
SetDeceleration 21
SetVelocity 419430
SetJerk 0
SetPosition 0
Update
  
```

Pro-Motion supports the ability to view communications between Pro-Motion and a connected PMD controller. These communications are in the form of hexadecimal-coded packets encoding short command-oriented transactions sent by the PC and responded to by the PMD controller. Packets can transmit messages which have meanings such as “Set the Profile Destination Position to 123,456” or “What is the current encoder position?” Transmitted command packets are displayed in the monitor as ASCII mnemonics. For example these two command functions are displayed in the monitor with the mnemonics **SetPosition** and **GetActualPosition**.

While many users will not need to concern themselves with the format of command mnemonics, software developers in particular may find the Monitor window traffic useful to learn how Pro-Motion commands the PMD controller to achieve certain functions.

Here is some additional information about the Monitor window that you may find helpful:

- To open the Monitor window click on the corresponding icon on the top menu bar.
- Displayed commands have zero, one, two, or three arguments. For details on command and argument formats refer to the *C-Motion Magellan Programming Reference*.
- Arguments that are prefaced with a “0x” (for example 0x1234) are being displayed in hexadecimal. Arguments without an 0x (for example 1234) are being displayed in decimal.
- To control which command packets are displayed right-click from within the Monitor window and select one or both of the ‘filter’ settings. *Filter Gets* will avoid displaying traffic due to regular background queries from Pro-Motion to the device. Note that an alternative to disabling all Get commands may be to disable automatic status polling. See [Section 3.5.1, “Status Polling.”](#) for information on this. *Filter Reads* will avoid displaying traffic containing scope trace data.
- You can clear the content of the Monitor window using right-click and *Clear*.

- You can save the content of the Monitor window to a text file using right-click and *Save As...*

The command packets that are displayed in the Monitor window consist only of Magellan Motion Control IC packets. If the PMD controller is a PRP Device, MotionProcessor (Magellan) command packets will be displayed but packets associated with Device, Peripheral, Memory, or CMotionEngine resource command traffic will not. For more information on PRP Device resource types refer to [Section 5.2, “Getting Started with C-Motion PRP.”](#)



## 3.7 Command Window

```

Command - Prodigy/CME PR33
For a list of commands press Tab
> #Axis 1
> #Axis 1
> #Axis 1
> GetVersion
0x58420031
> SetProfileMode 0
> SetStartVelociny
Error, syntax error in command string.
> SetStartVelocity 12345
> SetAcceleration 43
> GetAcceleration
0x0000002b 43
> SetVelocity 5000
> GetVelocity
0x00001388 5000
> SetPosition 123456
> GetPosition
0x0001e240 123456
> ClearPositionError
> SetSettleWindow 50
> SetSettleTime 100
> SetTrackingWindow 123
> Update
>

```

Pro-Motion supports a command line interface known as the Command window that allows the user to directly type in and send commands to the attached PMD controller.

The Command window has a DOS command line style interface with a “>” command prompt. All Magellan Motion Control IC commands are accepted, and as described in [Section 3.6, “Monitor Window.”](#) are expected to be in ASCII mnemonic format.

Here is some additional information on the Command window that you may find helpful:

- To open the Command window click on the corresponding icon on the top menu bar.
- Both the command mnemonic and associated argument values are entered on a single line followed by the Enter key. If the expected number of arguments are not entered an error message will display.
- Command mnemonics are not case sensitive.
- Entered arguments are separated from the command mnemonic and from each other (if the command accepts more than one argument) by spaces.
- All arguments are numerical. Arguments can be provided in decimal format (for example 1234) or in hexadecimal format by prefacing with “0x” (for example 0x1234).

- A facility for viewing the list of available commands is activated by hitting the Tab key. As characters are typed in, whenever the Tab key is pressed the command mnemonics that match the characters typed in are displayed in a “Select a Command” window that pops up. If Tab is pressed at the prompt a list of all available commands is displayed. To select a command from the list highlight the command and press Enter. The selected command appears at the command prompt (>) and the ‘Select a Command’ window closes.
- Upon entering the Command window typed-in commands are sent to the axis presently selected by the Project window. This can be changed while inside the Command window by entering a “#” followed without a space by “Axis n” (n being the axis number to change to) at the command prompt. For example after entering the command line “#Axis 3” all subsequent typed-in commands will apply to axis 3 of the PMD controller. The addressed axis can be changed by further #Axis commands, or by exiting the Command window, at which point the active Pro-Motion axis returns to the axis selected in the Project window.
- The *C-Motion Magellan Programming Reference* provides detailed syntax and argument definitions for each enterable command. Go to the alphabetized Instruction Reference section of this manual to lookup specific commands. The command syntax is indicated at the top with argument definitions provided just below. Note however that the *Axis* argument, which is shown as the first argument, should not be entered when typed into the Command window in the command line. Doing so will result in an error indicating there are too many arguments. As indicated above the addressed axis is specified using a separate #Axis command.
- If there is an error in processing a typed-in command a message will display indicating the nature of the error. An example of this in the form of a “Error, syntax error in command string” message can be seen in the Command window screen capture above. Commands that request information will return the requested value on the next command line. Commands that do not request information will display the command prompt at the next line.

### 3.7.1 Command Window Scripts

```

Command - Prodigy/CME PR33
> '-----
> ' Example script
> ' This script loads S-curve profile parameters
> ' and Servo gains into the motion IC
> '-----
> '
> SetProfileMode 0           ' Set to trapezoidal profile
> SetStartVelocity 0         ' Set start velocity value
> SetVelocity 10000          ' Set velocity value
> SetAcceleration 500000     ' Set acceleration value
> SetDeceleration 800000    ' Set deceleration value
> SetJerk 1234567            ' Set jerk value
> SetPosition 123456         ' Set destination position
> '
> ' Now set some position loop gain parameters
> '
> SetPositionLoop 0 123      ' Set Kp Proportional gain
> SetPositionLoop 1 25       ' Set Ki Integrator gain
> SetPositionLoop 2 10000000 ' Set Ilim Integrator limit
> SetPositionLoop 3 4560     ' Set Kd Derivative gain
> SetPositionLoop 4 1        ' Set Dtime Derivative time
> SetPositionLoop 5 32767    ' Set KOut Output gain
> '
> ' Get a few values
> '
> GetPositionLoop 0          'Get Kp
0x0000007b 123
> GetPosition                'Get Destination position
0x0001e240 123456
>

```

In addition to the features described above the Command window supports a very useful feature which is the ability to execute a series of commands stored in a script file. Command window scripts do not support conditional statements or branching, instead they act more as macros executing a fixed sequence of pre-programmed commands. As such they are generally used just for sending parameter initialization sequences.

Command window script files are text files containing only ASCII characters. Within Windows the Notepad program is a common way to edit text files, but any editor that supports a text file format can be used.

From a Command window session, to execute a script the “<” character is entered followed without a space by the script file name, for example <c:\scripts\TestScript.txt. Alternatively entering just the “<” character following by the Enter key will call up a Windows Open dialog box. An alternate way to access this same dialog box is right-clicking from the Command window and selecting *Specify Script File*.

Here is information on the formatting of script text files:

*File type* – Script files are ASCII text files

*Comments* – Comments are content embedded in the script file that are not executed, but rather are used by the script developer to clarify its content. Comments may appear anywhere within a line and result in all subsequent characters to the end of the line being ignored during script processing. The character recognized as beginning a comment is the single quote '.

*Blank lines, leading spaces and tabs* – Blank lines are ignored as are leading spaces or tabs.

*#Axis commands* – Scripts may contain #Axis commands anywhere in the script. This is the mechanism by which a script can send commands to different axes on a device that supports multiple axes.

*Commands and Newlines* – Only one command and its associated arguments (if any are needed) are recognized per line. Recognized characters marking the end of the line are \R (ASCII 13) and \N (ASCII 10).

*Argument delimiters* – Arguments must be separated from adjacent commands or arguments by either a space or a tab. Commas are not an allowed delimiter.

Here is some additional information on Command window scripts that you may find helpful:

- If during script processing an error is encountered script processing will halt.
- Nesting of scripts is allowed, meaning scripts are allowed to contain command lines that execute other scripts. If using this feature caution should be exercised to not have a script call itself.
- The primary intended purpose of Command window scripts is to load Magellan Motion Control IC parameter settings so that they don't have to be typed in manually. While not specifically prohibited, script commands that result in axis motion being initiated or modified is not recommended and should instead be done via the Pro-Motion Axis Control window, or via user-written application code.

### 3.7.2 Motion IC Settings Export to Script

Pro-Motion provides the ability to export the Magellan Motion Control IC settings for a selected device to a Command window script.

The configuration information that is saved to a script file consists of Magellan IC settings such as position loop gains, safety settings, signal sense, etc. Multi-axis products save settings for each axis. For example a four-axis Prodigy/CME Machine-Controller would have the configuration information saved for all four axes.

Here is some additional information that you may find helpful:

- To save the configuration to a Command window script file use the menu function *File/Export Motion IC Settings as Script*. A default file name is provided which can be changed. The specified script file will be created and displayed in Windows Notepad for convenience.
- Saved configuration script files are compatible with the Command window script facility and can be loaded and executed from the Command window. In addition the content can be edited or copied and incorporated into other script files as desired.
- Configuration scripts are ASCII-formatted files and therefore cannot be sent directly to the PMD controller using a standalone terminal or PC-based terminal emulator such as Procomm.exe. Scripts are parsed by the Command window and converted into hexadecimal-coded packets, which are then sent to the PMD controller.
- For script file exports there are generally more parameters saved than are actually used in a given application. If desired, the user can therefore review the saved configuration data and remove unused setting entries.



Only Magellan Motion Control IC configuration settings are exported to scripts via this export function. Non MotionProcessor resource configuration settings are not exported as part of this function. For more information on PRP Device resources refer to [Section 5.2, "Getting Started with C-Motion PRP."](#)

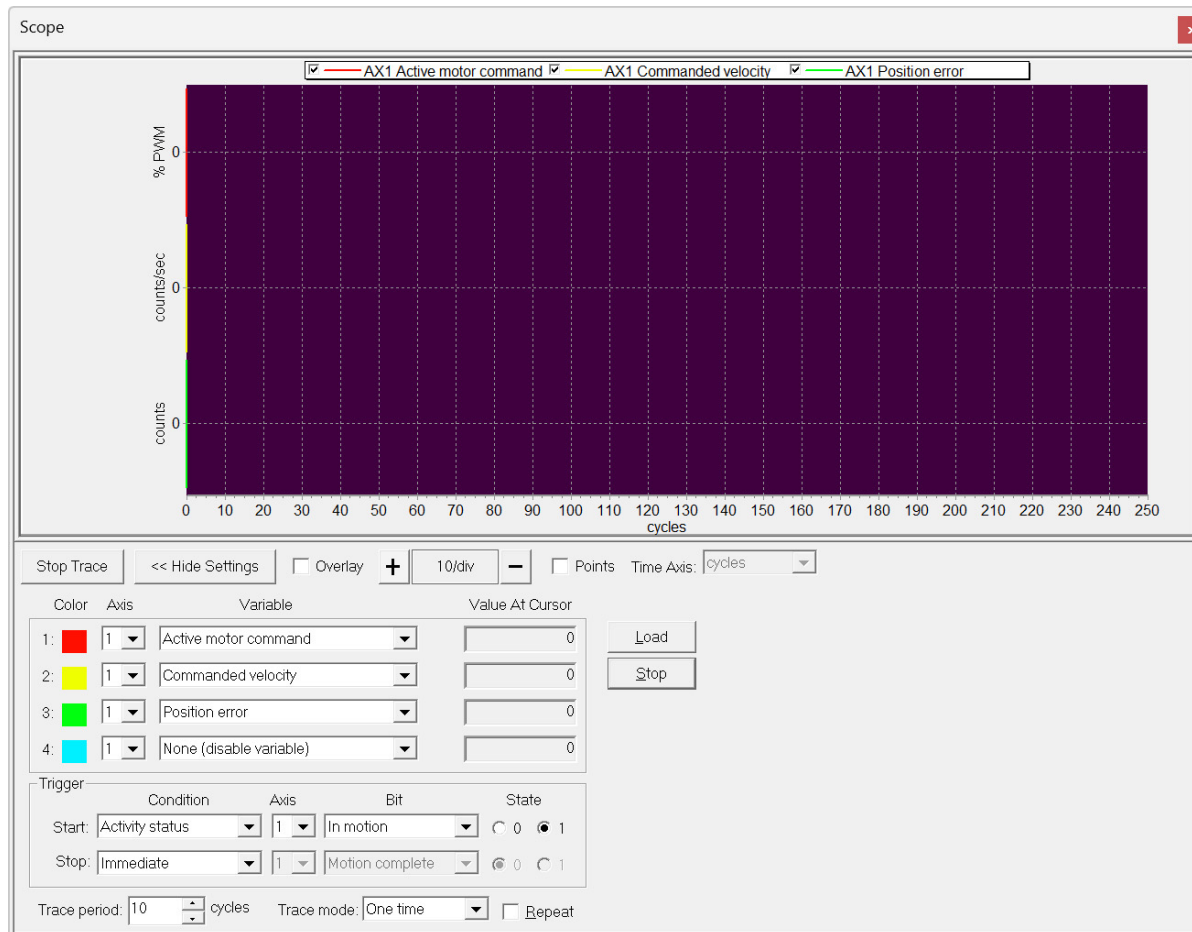
## 3.8 Scope Window

Pro-Motion's Scope window ties directly to the Magellan Motion Control IC feature called trace, which provides the ability to capture and store in hardware memory up to four motion registers simultaneously, at up to 20 ksamples/second.

Once a trace operation is specified the Magellan IC captures the motion values at a programmed time interval and stores these values in its local RAM memory. Pro-Motion then sends commands to retrieve the data from the Magellan IC and display this data graphically. In addition to being displayed, traced data can be captured to a file for import to spreadsheets or other graphing and analysis software. This is accomplished using the *File/Export Trace* menu function.

Common traced variables include the motor output command, position error, commanded position, commanded velocity, actual position, and others. In total there are over 100 different selectable trace variables.

To access the Pro-Motion scope function click the icon at the top bar labeled *Scope*. An example screen capture of the Scope window is shown below.



Here are some of the key settable fields of the Scope window:

*Trace Variables* – The core of the trace and scope function is the list of motion registers that will be traced and graphed. These are shown as Variables 1 through 4 with graph colors red, yellow, green, and blue respectively. For each trace variable click the down arrow which in turn displays a list of trace categories such as Commanded Trajectory, Feedback, Position Loop, etc... Selecting one of these categories then shows the specific available traceable motion variables. Note that for multi-axis devices variables can be traced from separate axes.

*Time Axis* – The top portion of the Scope window graphs captured data. Up to four variables can be graphed at the same time. The horizontal scale is time, with selectable units via the Time Axis field of cycles, milliseconds, and seconds.

*Trigger Controls* – Similar to a regular oscilloscope the conditions by which trace data collection can start or end is settable. Use the down arrow key to see the list of available trigger registers, and the associated bit or signal for each register. For examples of setting trigger controls see [Section 3.11.1, “Application Note — Determining the Optimum](#)

[Trajectory Acceleration,” Section 3.11.2, “Application Note — Tuning the Position Loop,”](#) and [Section 3.11.3, “Application Note — Determining Acceleration Feedforward Gain.”](#)

*Trace Period & Trace Mode* – The data capture trace period is expressed in cycles, which are 51.2  $\mu$ Sec per cycle. So specifying a period of 1 cycle captures data at  $\sim$ 20 kHz. Trace mode can be set to *One-time* or *Rolling Buffer*. One-time capture fills the trace buffer only once beginning when the trigger start condition is satisfied. One time capture is recommended unless a particular reason for selecting rolling buffer exists.

The most common use of rolling buffer mode is to display trace data leading up to the moment an event occurs. For example with rolling buffer mode selected, if *Immediate* is set as the trigger start condition and the *Event Status* register selected with *Motion Error* as the programmed bit and the state set to 1, after a motion error occurs the captured and displayed data will show the trace data prior to the occurrence of the motion error. For more information on motion errors refer to the *Magellan Motion Control IC User Guide*.



When using rolling buffer mode caution should be exercised to avoid overflowing the buffer, which can result in the graphed data not matching the actual traced data. This can occur if the rate of data being put into the trace buffer by the Magellan IC is faster than the rate at which Pro-Motion can request it.

*Trace Length* – The total number of trace samples can be specified via this setting. Setting this value may help you better manage how your data displays or how it exports to a file. The programmed value represents the number of data set captures. For example if this value is set to 500 and one variable is traced a total of 500 data values will be captured, if two variables are traced a total of 1,000 data values will be captured etc...

[Section 3.11.1, “Application Note — Determining the Optimum Trajectory Acceleration,”](#) [Section 3.11.2, “Application Note — Tuning the Position Loop,”](#) and [Section 3.11.3, “Application Note — Determining Acceleration Feedforward Gain.”](#) provide specific examples of how the Scope window can be used to achieve various useful motion application development functions.

### 3.8.1 Trace Buffer Display Optimization

The speed of the scope display update is related to the size of the buffer that fills with data in the Magellan IC, and the speed with which the Windows PC can retrieve this data from the Magellan IC. For higher speed interfaces such as CAN or Ethernet optimizing these settings is usually not needed. However for serial interfaces, improving the communication speed may be useful. To set the baud rate to a new value see [Section 4.1.2, “Changing the PMD Controller RS232 Settings.”](#)

Another way to speed up serial communications is to reduce the latency between message packet sends. This is done via the following sequence; First, open the Windows Device Manager by typing “Device Manager” in the Windows search box. Next, find Ports (COM & LPT) from the list and click on it, then right-click the USB Serial Port (COMn) of the serial port you are using and then select Properties. Click on the Port Settings tab and then select the Advanced button. Change the field for Latency Timer (msec) to a value of 1. Click OK on both Windows and close the Device Manager.

## 3.9 Project Configuration Save & Restore

Pro-Motion project files allow you to store a motion control configuration to a named file and later recall this saved configuration. You can save a project at any time while working with Pro-Motion by specifying *File/ Save Project* or *File/ Save Project As* from the Pro-Motion session menu.



Project files save the control parameters and communication settings for all axes of all PMD controllers connected to Pro-Motion and displayed in the Project window. The files that Pro-Motion uses to save and restore project configuration information have an extension of *.PMD*. These project files are not user-readable. They encode the project configuration information in a format that is written and read by Pro-Motion, but is not intended to be read or edited by the user.

The specific configuration information that is saved via the *File/Save Project* function is all of the PMD controller's configuration, control, and communication settings including all Magellan Motion Control IC settings. The main exceptions are content that is stored in NVRAM such as Magellan initialization commands, device SetDefaults settings, and *.bin* C-Motion Engine user code programs. Most of the Pro-Motion windows also have their content saved via the project save mechanism. This includes the Trajectory and Units dialog box settings of the Axis Control window, and the Scope window settings. Finally, the operating mode of the PMD controller is saved - in other words whether axes are enabled, the position loops are active, amplifier output is enabled, etc...

.PMD project files are not user-readable. They encode the project configuration information in a proprietary format that can be read (and written) by Pro-Motion, but are not intended to be read or edited by the user.



### 3.9.1 Project Configuration Restore

To call up a saved project select *File/Open Project*. You will be asked to specify a project file. Once you specify a file the content will be parsed by Pro-Motion and sent to the appropriate PMD controller(s).

Once saved configuration settings are loaded Pro-Motion will attempt to return each axis of the system to the operating mode it had when the project was saved. Along those lines, if appropriate, dialog boxes will appear asking whether the motor is ready to be energized. In the case of a Brushless DC motor energizing means the commutation is initialized and depending on the motor control mode setting the current loop and the position loop may be enabled. For DC Brush motors, similarly, the position loop and current loop may be enabled depending on the control systems in the project file. For step motors the drive current or holding current will be applied.

With communication connections and controller settings restored, after the project open operation the system should be restored to the same condition it had when the project was saved. At this point further development work on the project can occur and subsequent configuration changes stored, if desired, via the *File/Save Project* function. Alternatively a new project file can be created by saving the configuration via *File/Save Project As*.

Selecting File/Open Project to restore a saved configuration should be done with the connected PMD controller(s) in a reset condition or when in a stable non-moving state. Systems that have axes that may move when de-energized (such as vertical axes subject to the force of gravity) should only be restored from a reset condition. Failure to observe these guidelines may result in damage to the controlled mechanics.



## 3.10 Configuration Export to C-Motion

Pro-Motion provides the ability to export the Pro-Motion motion control setup as a C-Motion source code file, suitable for input to PMD's user application code building system. To learn more about how this export operation can be used to help with user application code building see the *readme.txt* file in the SDK you will be using for code development. For more information on C-Motion SDKs see [Section 5.1.4, "C-Motion SDKs."](#)

Here is some additional information that you may find helpful:

- To save the configuration to a C-Motion file use the menu function *File/Export Motion IC Settings as C-Motion*. A default file name is provided which can be changed. The specified C-Motion source code file will be created and displayed in Windows Notepad for convenience.
- For C-Motion exports there are generally more parameters saved than are actually used in a given application. If desired, the user can therefore review the saved configuration data and remove unused setting entries.

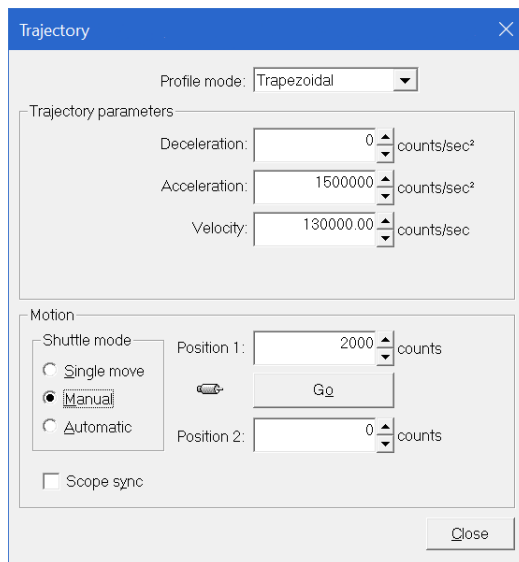
## 3.11 Pro-Motion Application Notes

In the next few sections we provide a few examples of Pro-Motion Application Notes illustrating how Pro-Motion can help characterize and optimize the operation of your motion control system.

### 3.11.1 Application Note – Determining the Optimum Trajectory Acceleration

Many different control parameter optimizations can be explored using the scope function's extensive list of traceable variables. In the sequence below we will provide an example of an optimization session that shows how to determine the profile acceleration setting. Our goal is to determine what maximum acceleration the motor and attached mechanisms can achieve without exceeding the safe current limit specification for the motor. In this example we will assume the motor is driven with a position loop, which is the normal positioning control mode for DC Brush and Brushless DC motors.

- 1 Start by opening up the Trajectory dialog box, which can be accessed from the Axis Control window. Specify Trapezoidal profile mode.



- 2 In the Shuttle mode box at the lower left select Manual.
- 3 Enter an acceleration value, a velocity value, and two destination positions. A deceleration of 0 commands the deceleration rate to match the acceleration rate. The two entered positions are the positions the motion shuttle will alternate between.

- 4 Next open up the scope window and enter the following settings:
  - Trace variable 1 - Active motor command
  - Trace variable 2 - Commanded velocity
  - Trace variable 3 - Position error
  - Trace Period: 10 cycle
  - Trace mode: One-time
  - Start trigger condition Activity Status register, In motion bit, State = 1
  - Stop trigger condition - Immediate
- 5 Review the trace length field located in the middle bottom of the screen and increase or decrease as desired. 250 to 500 is a typical number because it generally displays one oscilloscope screen-full without 'scrolling'. But most PMD products offer far more storage than this. Larger traces may be useful if data will be exported for analysis in a spreadsheet or other analysis tool. To export traced data to a file use the Pro-Motion *File/Export Trace* menu function.
- 6 Select the Start Trace button in the upper left of the trace window. The data graph display should not change. If it immediately begins to graph data wait till graph display completes and select Start Trace again. When in the proper state the graph area should go blank and although the Start Trace has been activated trace capture (or graphing) does not begin.
- 7 From the Trajectory dialog box select Go. You should see the trace data graphing area immediately begin to display captured data and continue till a full buffer has been captured. The reason this is the case is that by pressing Go the Magellan trajectory generator is activated and the Activity Status Register In Motion bit goes from 0 (false) to 1 (true).

To manage the data being displayed you can use the scope's - or + buttons to expand or reduce the horizontal scale. Alternatively if you would like to speed up the trace buffer update, see [Section 3.8.1, "Trace Buffer Display Optimization,"](#) for suggestions.

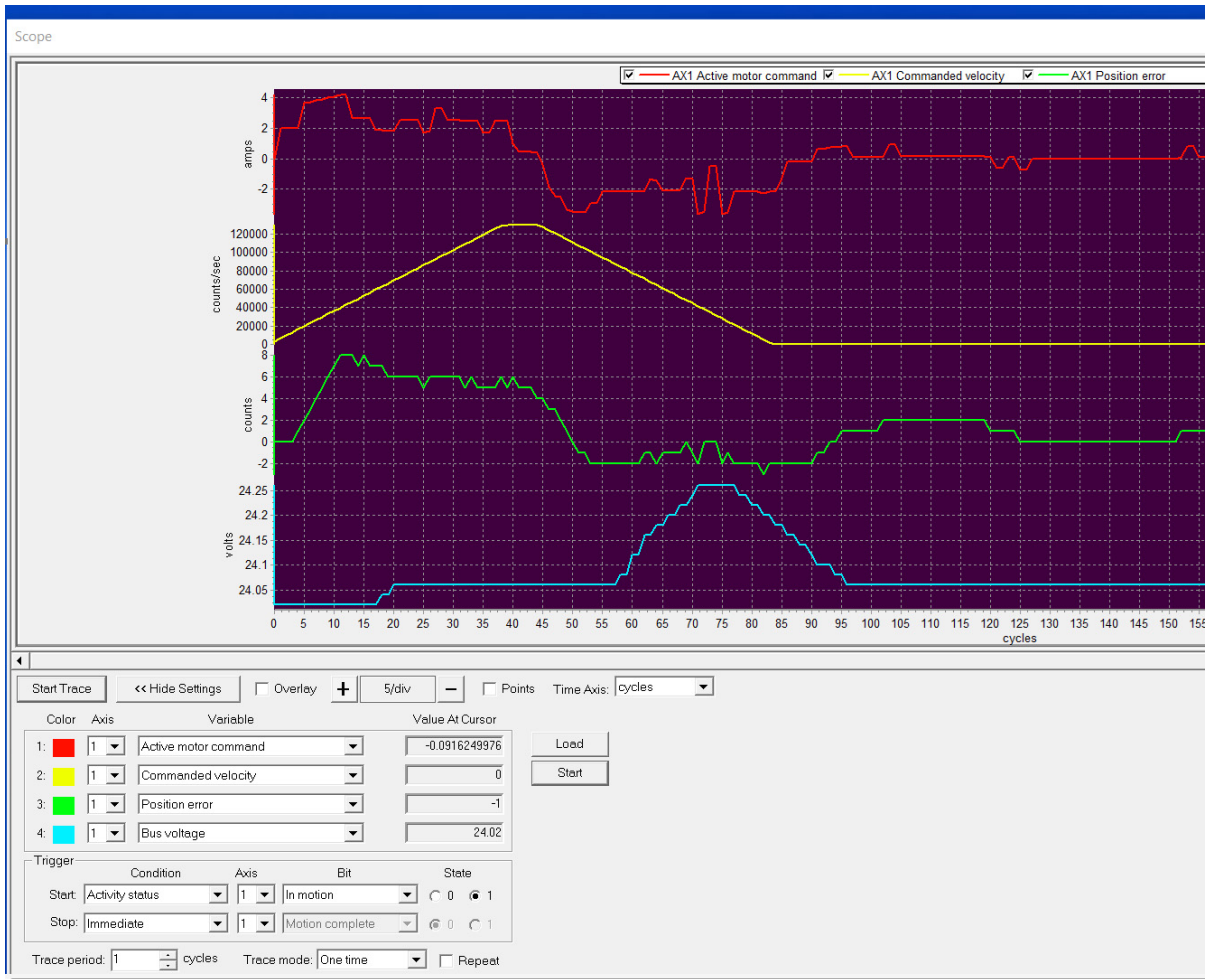
Based on the resultant graph display you may also want to change the shuttle move distance so that the entire trapezoidal move is displayed.

The goal for setting an optimal trajectory acceleration is to increase it just until the PMD controller's programmed current limit (generally set equal to the maximum current specification on the motor data sheet) is exceeded. When this happens, even though the trajectory generator asks for more torque to achieve the requested acceleration, the torque output saturates resulting in the actual motor position rapidly falling behind the commanded position.

In this trial and error approach you will iterate by increasing or decreasing the setting being explored, re-run the profile and repeat until you find an optimum value. This process is made easier with the Manual shuttle feature of the Trajectory dialog box which uses two different programmed positions.

Each time you hit the "Go" button the commanded position switches from one to the other. Using the shuttle also has the benefit of creating alternating positive and negative direction moves, which is useful to confirm there are no unexpected differences in the movement based on the direction of motion.

The image below shows an example of what a trace may look like when optimizing the acceleration:



This screen capture shows data from a motor with a current drive limit of 4 amps, optimized to accelerate as fast as possible without exceeding the current drive limit. The resultant point-to-point trapezoidal move traverses 300 counts (~ 25 motor shaft degrees) in approximately 4 milliseconds. Note that despite this aggressive move the on-the-fly position error never exceeds 8 encoder counts, and the motor settles to within 2 counts error after 85 cycles, which corresponds to 4.25 mSec.

### 3.11.2 Application Note – Tuning the Position Loop

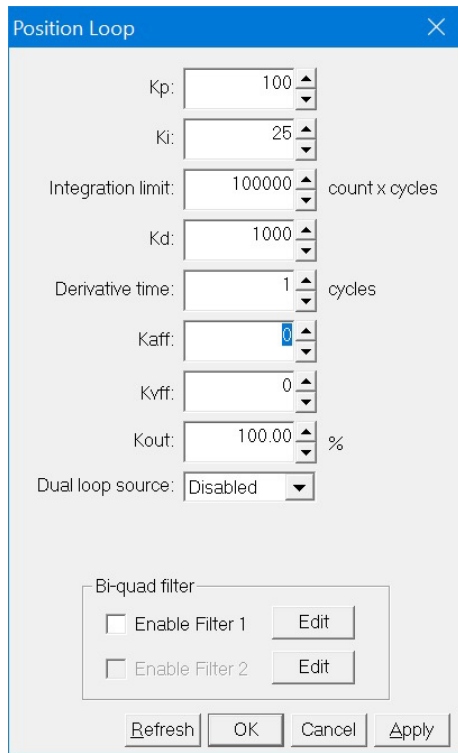
This application note provides an approach to developing PID position loop gain settings. Note that this section only applies to servo motors (DC Brush, Brushless DC) and motors using a closed loop stepper method. The results you obtain may not be completely optimum, but should work well in a large majority of cases.

Many users will first experience manually tuning their position loop while using the Axis Wizard. All servo motors (DC Brush or Brushless DC) which use the position loop require position loop gain settings. If you are tuning the position loop from the Axis Wizard make sure to enable the scope display by hitting the “Display Scope” button, after which you may skip to [Section 3.11.2.2, “Position Loop Tuning.”](#)

If you are tuning your position loop from the Axis Window, either because you didn’t set up the motor configuration using the Axis Wizard, or because you want to re-tune gain parameters for your motor, continue reading from the next section to set up the scope window for a tuning session.

### 3.11.2.1 Scope Window Tuning Setup

- 1 To set up a tuning session the motor should be energized. Open the position loop dialog box by clicking the Position Loop box in the Axis Control window. The diagram below shows what this dialog box looks like:



- 2 Next, click the Tuning button on the right side of the top icon bar of Pro-Motion. After doing so the trace scope will be displayed along with a Step Response dialog box. You may want to adjust the location and size of the Scope window to fit your monitor size.
- 3 In the Scope window select *Active Motor Command* for the first variable to trace, select *Commanded Position* as the second variable, and select *Actual Position* as the third variable.

### 3.11.2.2 Position Loop Tuning

With the scope properly set up for a tuning session (or if you are tuning from the Axis Wizard where the scope is already set up for tuning) the following instructions apply.

- 1 Specify a step move distance. A typical move distance for motors with an encoder is 50 or 100 counts. For motors using Halls to track the position a typical move distance is 5 to 10 counts.
- 2 Set Ki (Integration gain), Integration limit, Kaff (Acceleration feedforward gain), and Kyff (Velocity feedforward gain) to zero. Set Kp (Proportional gain) to a small value, typically 10 to 50 for motors with encoders and 100 to 500 for motors with Halls only. Set the Kd (Derivative gain) to a value 10 times greater than the Kp value you entered. Set the derivative time to 1 and set KOut to 100%.
- 3 Click the right arrow key. This should result in the motor quickly (in a single instantaneous step) moving by the programmed amount, and shortly thereafter the trace scope window should begin to display data. If this doesn't happen review the instructions above and try again.
- 4 If the motor oscillates in an uncontrolled manner set Kp to a lower value and click the right arrow key again. If at any point you are concerned that the motor or power supply may be over-stressed you can hit the "stop" icon in the tool bar to immediately disable the servo loop.

Once the position loop is at least somewhat stable, you will begin a repeating sequence consisting of:

- View the scope display to determine whether the response is underdamped, critically damped, or overdamped (see below for instructions on how to do this)
- Adjust the gain settings accordingly
- Make another step move

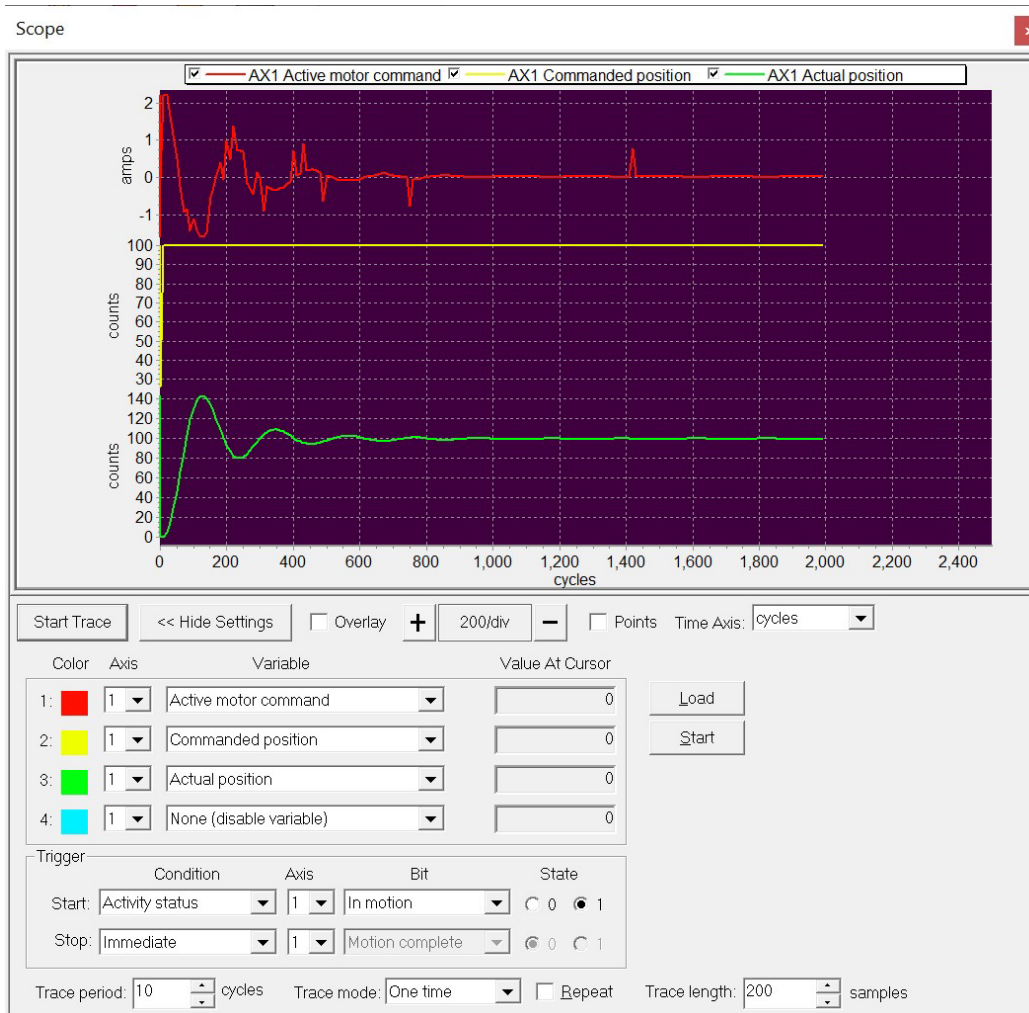
The next three sections will show you how to characterize the step move response which will in turn indicate how gain settings should be adjusted.



During servo tuning a certain amount of oscillation is not unusual, but if at any point you become concerned that the motor or power supply may be over-stressed hit the “stop” icon in the tool bar to immediately disable the servo loop.

### 3.11.2.3 Underdamped Response

The screen capture below shows a typical underdamped response.



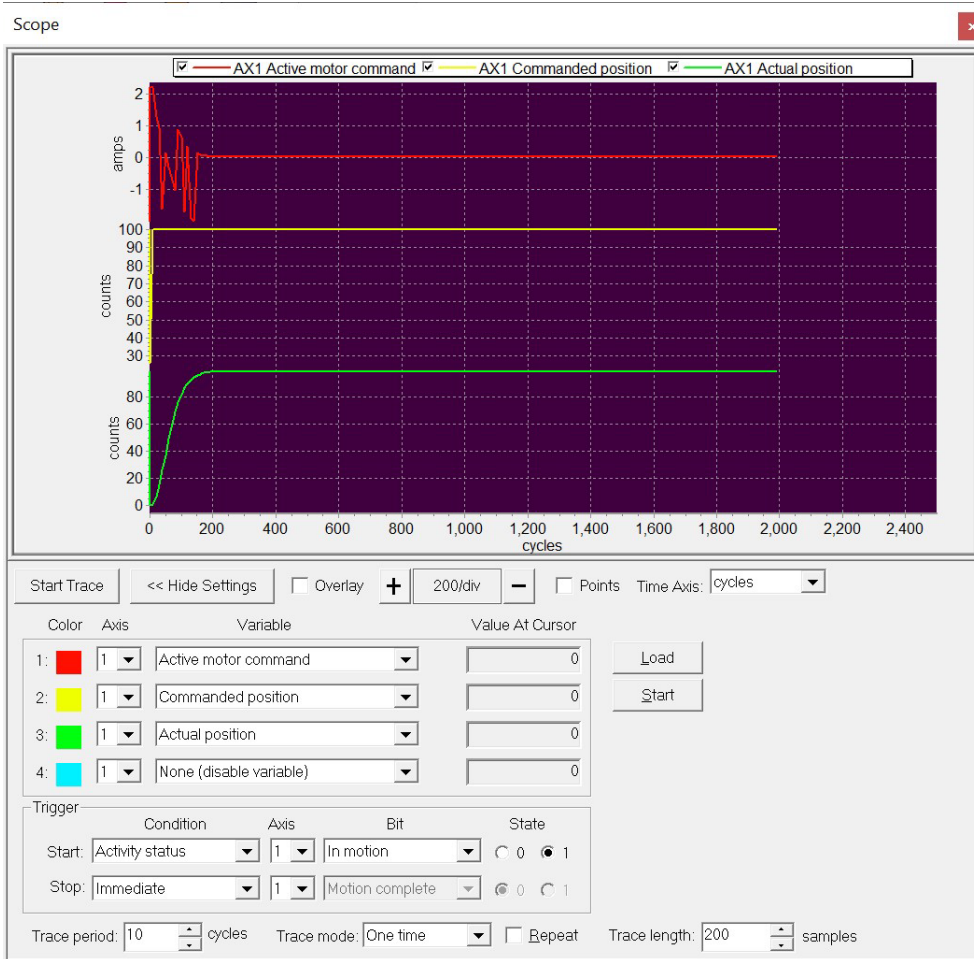


The yellow trace shows that the commanded position instantly changes from a position of 0 to a position of 100 (in your setup this jump will equal whatever you entered into the step response dialog box). The green trace shows the resultant actual motor location.

From this trace, although the actual position does eventually settle to a position value of 100, it overshoots significantly and oscillates several times. Whenever the actual axis position overshoots the commanded position the system is considered underdamped, and to correct for this you should increase the derivative gain (Kd). The goal of the Kd setting is to determine a value that results in a critically damped response, as shown in the next section.

### 3.11.2.4 Critically Damped Response

The screen capture below shows a critically damped response.

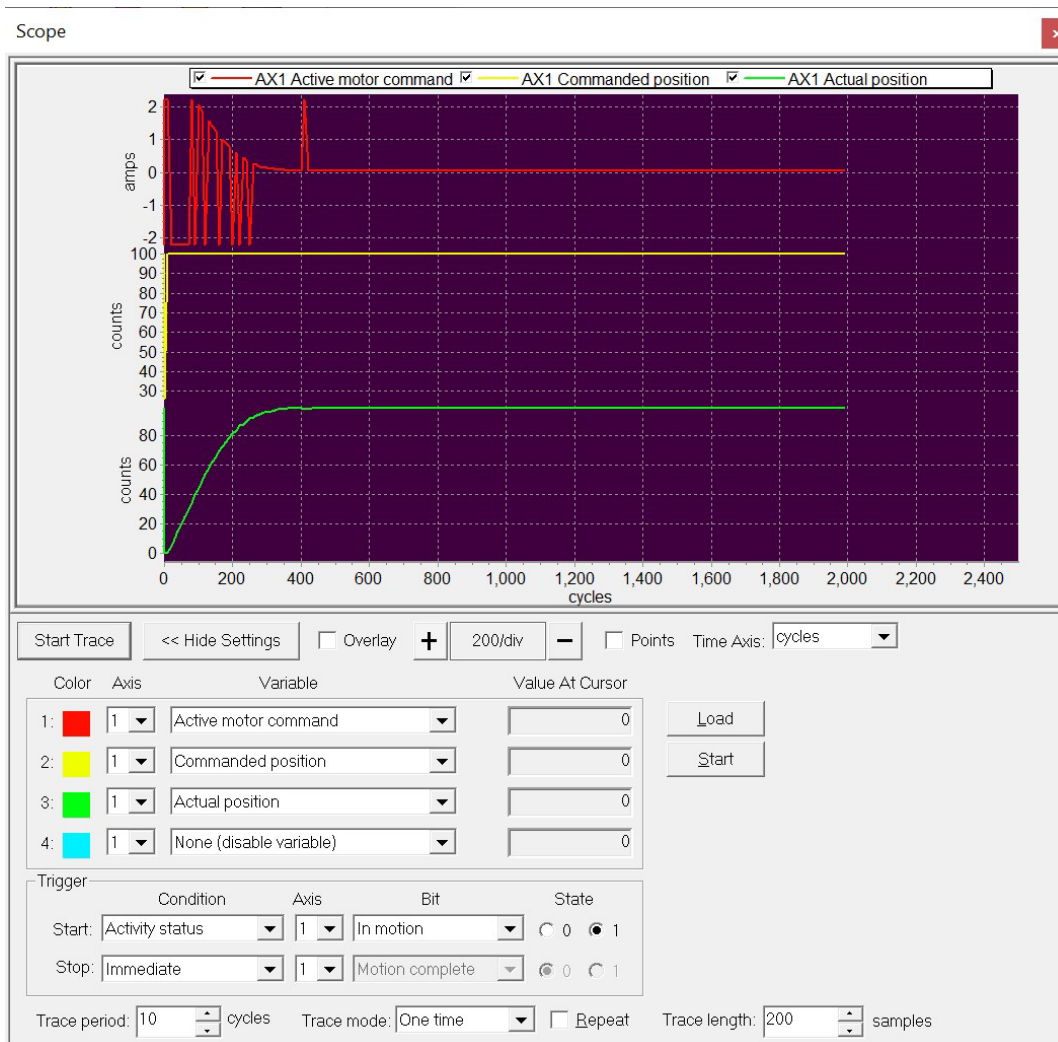


In this trace the actual position does not overshoot that commanded position, and it settles quickly for this particular motor in 10 mSec to the commanded position of 100.

It's worth noting that many Kd settings could result in a response that appears critically damped. The distinction between critically damped and overdamped is somewhat subjective when using this manual trial and error process. In general you should try to find the smallest setting of Kd which still provides a critically damped response.

### 3.11.2.5 Overdamped Response

The screen capture below shows an overdamped response.



This can be seen by comparing with the critically damped response in the previous section. An overdamped system will be stable but needlessly slow in its response to changes in the position command.

If the response is overdamped you should reduce  $K_d$ , trying to find the smallest value of  $K_d$  that still gives a critically damped response.



For purposes of illustration in the above three example graphs the  $K_p$  setting was 100 and the  $K_d$  settings were 2,000, 6,000, and 25,000 for the underdamped, critically damped, and overdamped responses respectively.

### 3.11.2.6 Getting To Final Gain Settings

Your overall position loop tuning goal is to set the  $K_p$  to as high a value as possible while still finding associated  $K_d$  values that can create a critically damped response. Higher  $K_p$  values will result in more accurate tracking and faster responses to command position changes.



As you try higher and higher  $K_p$  values, at some value of  $K_p$  you will find that there is no value of  $K_d$  that can create a stable and critically damped response. At this point you should reduce the  $K_p$  setting by 35-50% and determine the associated critically damped  $K_d$  setting. Backing down from the 'borderline'  $K_p$  value is important for improving stability, accommodating small differences in controller, motor, and attached hardware behavior, and for handling changes that may occur to the system over time.

### 3.11.2.7 Setting $K_i$ , $Ilimit$ , and $Dtime$

Once  $K_p$  and  $K_d$  have been set, you can enter a value of  $K_i$  (Integration gain) to improve tracking accuracy. Typically,  $K_i$  settings are 1/10 to 1/2 of the  $K_p$  value. Smaller motors typically use relatively large ratios, and larger motors with larger loads typically use smaller  $K_i$  ratios. Higher  $K_i$  settings will increase tracking accuracy during and after the motor move is complete but can also reduce system stability. So you should set  $K_i$  to the smallest value that can achieve your goals for final or dynamic tracking while not distorting the critically damped response the  $K_p$  and  $K_d$  settings created.

The Integration limit can be set to a high value, perhaps 10,000,000. The Magellan IC PID engine has built in anti-windup logic so it is very unlikely this integration limit will ever be reached.

The final 'core' PID setting, known as derivative time, can often be left at one, meaning one sample time. Setting it to higher values means the period at which the derivative contribution is calculated is increased by a multiple of the sample time. This can be useful to lower the frequency of  $K_d$  'chatter' to below audible, should there be any in the system. Another benefit to increasing the derivative time may be to increase the influence of the  $K_d$  term. The impact of the  $K_d$  on the PID is a direct multiple of the derivative time. For example if the derivative time is set to 10 and the  $K_d$  value is 100 this would be an equivalent impact as a  $K_d$  of 1,000 with a derivative time of 1.

### 3.11.2.8 Frequency Based Tools & Analysis

The above process represents a popular and relatively straightforward approach toward tuning the PID loop. There are more sophisticated approaches however both for determining PID parameters and for verifying the behavior of a given position loop controller. Pro-Motion provides frequency-based tools known as Bode plots to support such methods.

Bode plots can be used in different ways and can help characterize your mechanical system as well as determine important characteristics of the operating control loop such as the bandwidth and phase margin. The Pro-Motion Bode plot tool can be accessed from the *View/BodePlots* Pro-Motion menu selection.

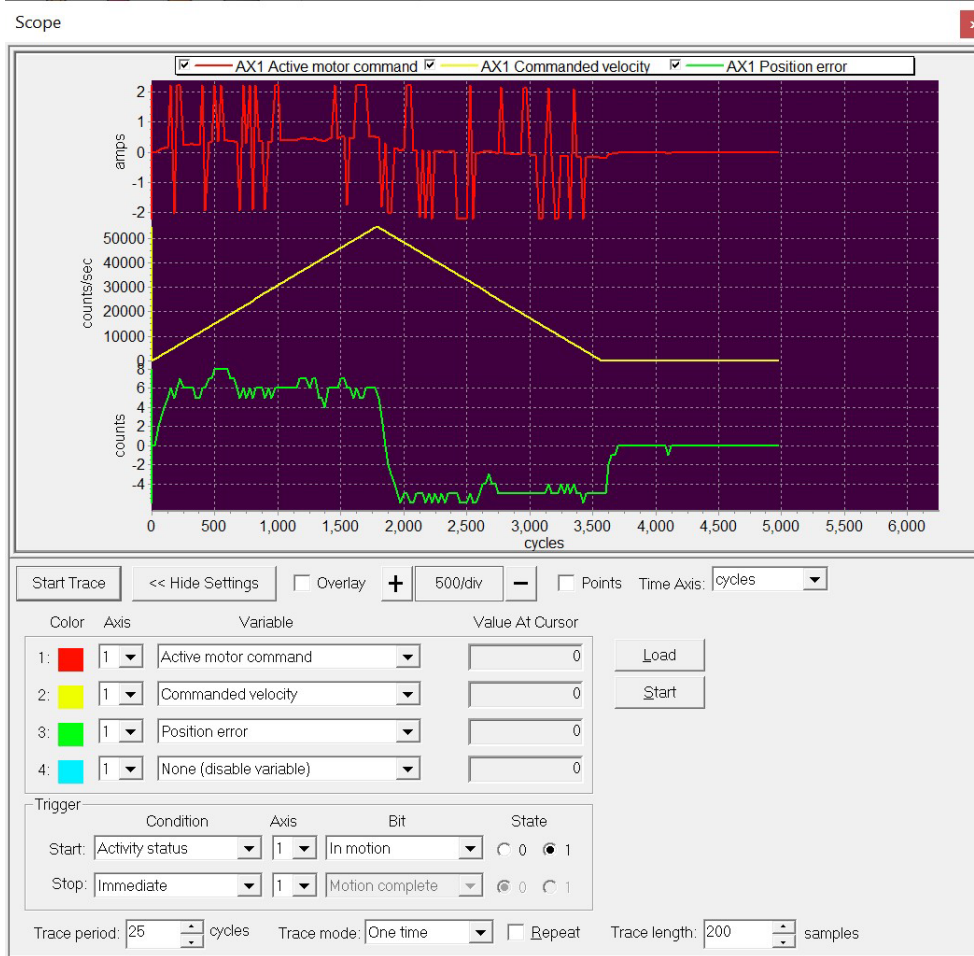
One output of a Bode-based system characterization may be the identification of natural resonances in the mechanical system. While a properly tuned PID can help reduce the impact of this, the Magellan IC also provides two general purpose biquad filters in the position loop. Biquad filters can be used to create low pass and notch filters. For more information on biquads as well as the Magellan Position PID loop refer to the *Magellan Motion Control IC User Guide*.

## 3.11.3 Application Note — Determining Acceleration Feedforward Gain

The following optimization session using the Scope window shows how to determine the acceleration feedforward gain.

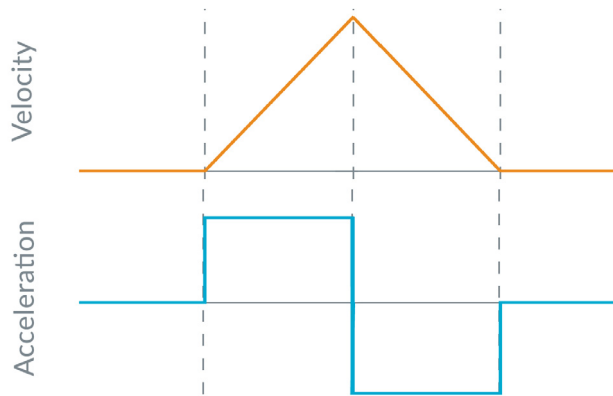
The graph below shows a complete point-to-point profile move using the critically damped system settings from [Section 3.11.2, "Application Note — Tuning the Position Loop."](#) The yellow line represents the commanded trajectory velocity, and the green line represents the position error (the difference between the commanded and the actual position). Notice that despite the fact that the axis is accelerating and decelerating quite aggressively (the entire move

duration is less than 200 mSec) the motion is stable throughout and the tracking accuracy is quite good even during the motion, varying from +8 counts maximum position error to -5 counts.



Nevertheless Acceleration feedforward (Aff for short) can reduce the dynamic tracking error even further. Acceleration feedforward works by adding a torque command proportional to commanded acceleration. PMD's Magellan IC also supports velocity feedforward, which similarly adds to the motor command proportional to the commanded velocity. Velocity feedforward is used less often than acceleration feedforward, typically to compensate for fluid friction such as lubricated bearings or in fluid pumping applications.

The reason Aff improves tracking accuracy is that from the position error graph above we can see that the form of the position error mimics the form of the trajectory's commanded acceleration. This can be seen in [Figure 3-1](#). Therefore by 'pre-injecting' this feedforward value into the position PID loop the servo is required to do less work, resulting in better tracking accuracy.



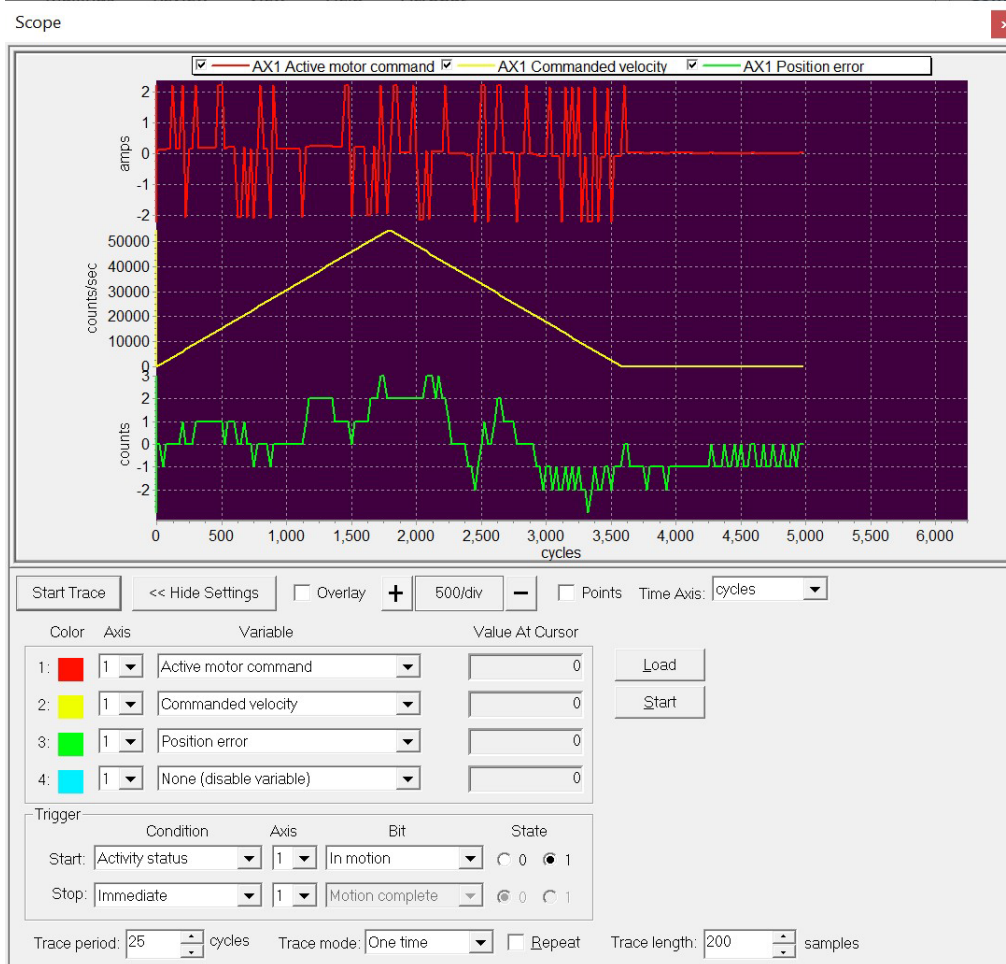
**Figure 3-1:**  
Velocity and  
Acceleration  
Versus Time for  
Point-To-Point  
Trapezoidal  
Trajectory

This general session for optimizing Acceleration (or Velocity) feedforward uses the same setup as described in [Section 3.11.2, “Application Note — Tuning the Position Loop.”](#)

As for all trial and error optimizations with the Pro-Motion scope function you will iterate by increasing or decreasing the setting being explored, re-run the profile and repeat until you find an optimum value. This is most easily accomplished using the Manual shuttle mode of the Trajectory dialog box. See [Section 3.11.2, “Application Note — Tuning the Position Loop.”](#) for details.

In the case of determining the best Aff value, you will find that when you are below the best Aff value further increasing the gain decreases the average position error, and when you are above the best value further increasing the Aff gain increases the position error. You are looking for that ‘best’ Aff value where any increase or a decrease in gain value increases the position error.

After adding an optimized Aff gain to the PID loop the results are shown below. Although the tracking is not perfect, the magnitude of the position error is halved during the motion.



One caution with acceleration feedforward is that it is load specific. If you tune the Aff value by increasing and decreasing till the net position error during motion is minimized, you will find that if the load on the motor changes you will need to re-tune the Aff value to achieve a similar result. This means systems that typically carry a variable load may not be good candidates for using an acceleration feedforward gain.

## 3.12 Troubleshooting Suggestions

If the first-time verification sequence detailed in [Section 2.4, "Step #4 — First Time System Verification,"](#) was successful problems while using Pro-Motion are not common. Nevertheless below is a list of suggested corrections for issues users may encounter while exercising their motion hardware.

**Index Capture Isn't Being Recognized.** Especially during Axis Wizard setup, if Index signal capture is not occurring at all, or not once per rotation, the problem may be related to signal qualification with the QuadA and QuadB signals. A remedy may be to invert both the A and B signal using the 'Encoder Test' screen in the Axis Wizard. Depending on your encoder's signal output format however additional signal manipulation may be needed. Refer to [Appendix B, Index Capture Qualification](#), for details.

**The Motor Stops Responding.** If a motor previously moving and responding to your commands stops responding the most likely reason is that the operating mode is not set correctly. Although most such 'transitions' from normal operation to an error/protection state result in a dialog box appearing, to check the operating mode select the

Operating Mode box in the Axis Control window and adjust the settings if needed. See [Section 3.4, “Axis Control Window,”](#) for more on the Operating Mode dialog box.

**The Step Motor Stops Responding.** In addition to checking the operating mode as detailed above, step motors controlled via microstepping mode may stop moving because the motor command has been set to zero. This may occur during safety related actions or when certain command functions are sent. To check and restore the motor command setting select the Motor Control box in the Axis Control window.

**Events Aren’t Being Reported.** As the Magellan Motion Control IC executes its control settings some occurrences representing a change in the control system state may occur. These changes are called events and include items such as motion error, over temperature, breakpoints, and more. To check the Pro-Motion settings for which events will result in a dialog box popping up click the Event Manager box in the lower right of the Axis Control window. For detailed information on Magellan events refer to the *Magellan Motion Control User Guide*.

**An Overvoltage Event Occurred.** If during a motion profile an overvoltage event is indicated the most likely reason is that during deceleration of the motor the inertia of the motor and load resulted in net generation of energy. Depending on the design of the power supply you are using this can result in the HV voltage rising and eventually exceeding the overvoltage limit. Remedies are to reduce the profile’s rate of deceleration or add more capacitance to the HV supply.

**An Undervoltage Event Occurred.** If during a motion profile an undervoltage event is indicated the most likely reason is that during acceleration the power supply was not able to deliver the needed amount of current, or was not able to deliver an increase in current fast enough. Undervoltage (and overvoltage) events may also occur if the current loop or position loop is unstable. Remedies are to increase the current output limit/rating of the power supply, add more capacitance to the HV supply, or reduce the profile’s rate of acceleration or re-tune the servo loops.

**An Overcurrent Event Occurred.** If during a motion profile an overcurrent event is indicated the most likely reason is that during a rapid trajectory deceleration the inertia of the motor and load resulted in the HV voltage rising rapidly, which, in combination with a motor with very low coil resistance can sometimes result in excess bus current. Remedies are to reduce the profile’s rate of deceleration, use the shunt feature of the PMD controller (if it has one), or add more capacitance to the HV supply.

**A Motion Error Event Occurred.** If during a motion profile a motion error event is indicated there are a few potential reasons. Motion error, also called position error, means the difference between the commanded position and the actual motor position has exceeded a user-specified threshold. Potential reasons are that the axis motion was blocked, that the position loop is not well tuned, that the commanded profile velocity exceeds the achievable motor velocity, or that an aggressive profile has temporarily resulted in a motion error during the move. Remedies depend on the cause, and are generally straightforward once the cause is determined. To change the position error threshold use the Tracking box in the Axis Control window.

**Communication Errors Are Occurring.** Communication errors between Pro-Motion and the controller are rare, but depending on the connection type and motion setup may occur. The 3-pin Programming Port for example is susceptible to noise if high current moves are commanded because it uses a UART signal scheme which is not a differential. For whatever communication link is used, you should try to determine if there are specific operating conditions that result in communication errors. If signal noise may be at fault consider switching to a link that uses a differential signaling scheme such as RS232, CAN, or Ethernet.

Depending on the type of amplifier used some of the events described above may not be reportable by the control system. For example if Atlas amplifiers are used Overvoltage and Undervoltage event reporting will occur correctly, but if user-designed or non-PMD amplifiers are used these events will not be detected or reported by Pro-Motion.



*This page intentionally left blank.*

# 4. Communication Port Connections

## In This Chapter

- ▶ Serial Communications
- ▶ CAN Communications
- ▶ Ethernet Communications
- ▶ Communicating With Attached Devices

In this chapter we provide additional information on Pro-Motion communication connections including how to set up and operate PMD controllers for RS232, RS485, CAN, and Ethernet communications and how to set up an ‘attached device’ network.

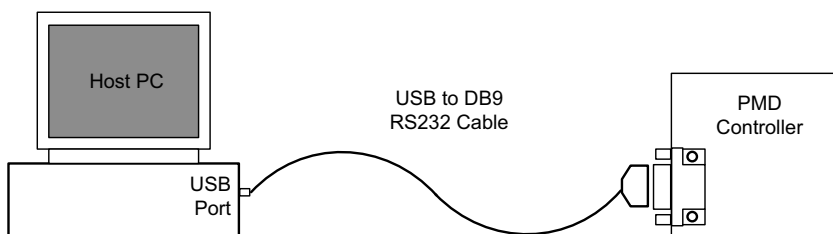
This information may be useful to change the connection mode originally set up in [Chapter 2, Quick Start Guide](#), or to add additional PMD controllers to the existing Pro-Motion controller setup.

## 4.1 Serial Communications

Prodigy/CME Machine-Controllers support RS232, RS422, and RS485 communications. For detailed information on the machine controller’s serial interfaces refer to the *Prodigy/CME Machine-Controller User Guide*.

In the following sections, although we have already connected via RS232 in the quick start guide, for the sake of completeness we will detail how to connect from a PC to the PMD controller both for RS232 and RS485.

### 4.1.1 RS232 Communications



**Figure 4-1:**  
Hardware Setup for Communications via RS232

For the PC to communicate via RS232 you will use the USB to DB9 serial converter cable (PMD P/N Cable-USB-DB9) which is included with machine controller DKs. The machine controller directly accepts this cable at its J23 connector.

Although in this RS232 setup description we will only use Serial1, machine controllers provide two RS232 interfaces, Serial1 and Serial2. Note that only Serial1 can be used to communicate with Pro-Motion. Serial2 has a few potential uses including as a console port, or as a general purpose serial connection used by user application code running in the C-Motion Engine.

Two serial-related cables are provided with the machine controller DK as listed below. One is a single RS232 USB to DB9 cable and the other is a bifurcated cable that provides RS232 DB9 connections for Serial1 as well as Serial2.

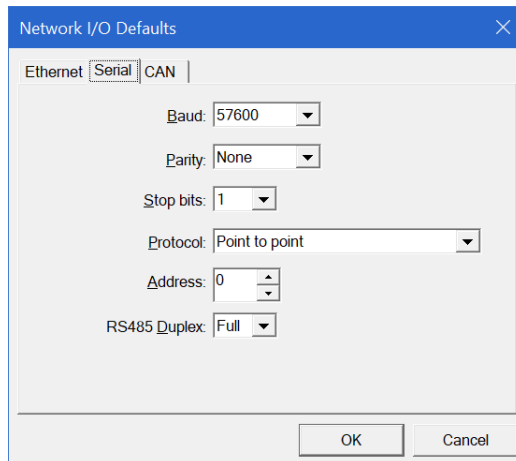
Vendor	P/N	Description
PMD	Cable-USB-DB9	Male DB9 to USB serial converted cable.
PMD	Cable-4355-01.R	Male DB9 dual serial to two single-channel female DB9 cable.

For detailed wiring information on Cable-4355-01.R refer to [Section 6.4.4, “Cable-4355-01.R.”](#)

## 4.1.2 Changing the PMD Controller RS232 Settings

Here is the Pro-Motion sequence to change the RS232 parameter settings in the PMD controller.

- 1 With Pro-Motion communications via the RS232 port functioning properly, click the Device Control toolbar icon. The Device Control window appears.
- 2 Click Network I/O. The Network I/O Defaults dialog box appears.
- 3 Click the Serial tab. This window appears with data entry fields for the default serial port setting such as the baud rate. This is shown below with default values visible.



- 4 Enter the desired com settings in the corresponding data fields. The Protocol field should be set to point-to-point and the Address and RS485 Duplex fields can be left unchanged.
- 5 Click OK to store as the power on default in the PMD controller's NVRAM.
- 6 Disconnect RS232 communication using the Disconnect toolbar icon.
- 7 Power down the PMD controller and then after a pause of a few seconds repower it.

The PMD controller has now been programmed to use the new RS232 communications settings.



### 4.1.3 Setting Pro-Motion for RS232 Settings

If the communication settings of the PMD controller have been changed the settings used by Pro-Motion must be set to the same settings.

When changing the Serial1 parameters special care should be taken since this is the channel being used to communicate from Pro-Motion to the PMD controller. If for some reason communication is lost and cannot be re-established after the RS232 settings are changed, it is recommended that you try to connect via CAN or Ethernet ports, if available, by setting Pro-Motion to communicate at their default settings. For the default communication settings for CAN and Ethernet refer to *the Prodigy/CME Machine-Controller User Guide*.



To set Pro-Motion's RS232 parameters:

- 1 With no RS232 connection to the PMD controller operating, click the Connect toolbar icon.
- 2 Select Serial from the list of options.
- 3 Select the comm port that the RS232 connection is located at and select OK.
- 4 Enter the same RS232 settings as were programmed into the PMD controller previously.
- 5 When complete, click OK.

If RS232 communication is successful a graphical icon representing the PMD controller on the network will be loaded into the Project window. If communication is not successful, a Communications Timeout Error dialog box appears. If this happens recheck the connections and retry to establish communications with the PMD controller unit.

With RS232 communications established, a useful optimization of the RS232 connection may be to reduce the USB latency between message packet sends. If this was already done for your PC during quick setup then there is no need to set this again. If not, the next paragraph describes how this can be done:

First, open the Windows Device Manager by typing "Device Manager" in the Windows search box. Next, find Ports (COM & LPT) from the list and click on it, then right-click the USB Serial Port (COMn) of the serial port you are using and then select Properties. Click on the Port Settings tab and then select the Advanced button. Change the field for Latency Timer (msec) to a value of 1. Click OK on both windows and close the Device Manager. The driver is now optimized for use with the machine controller drive.

### 4.1.4 RS485 Communications

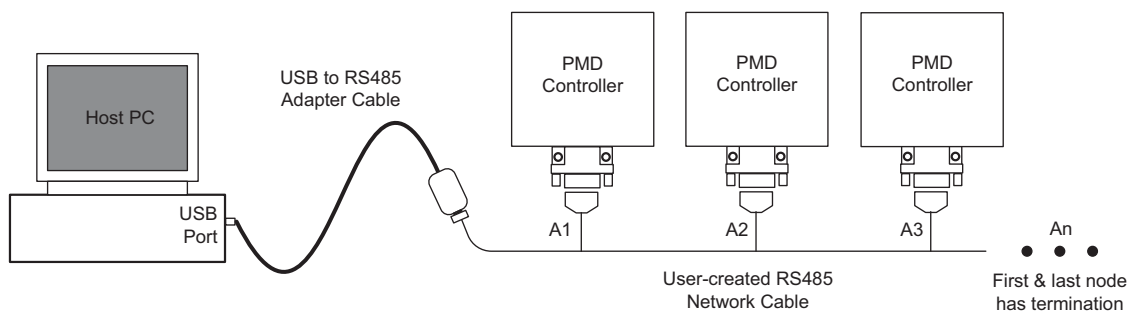
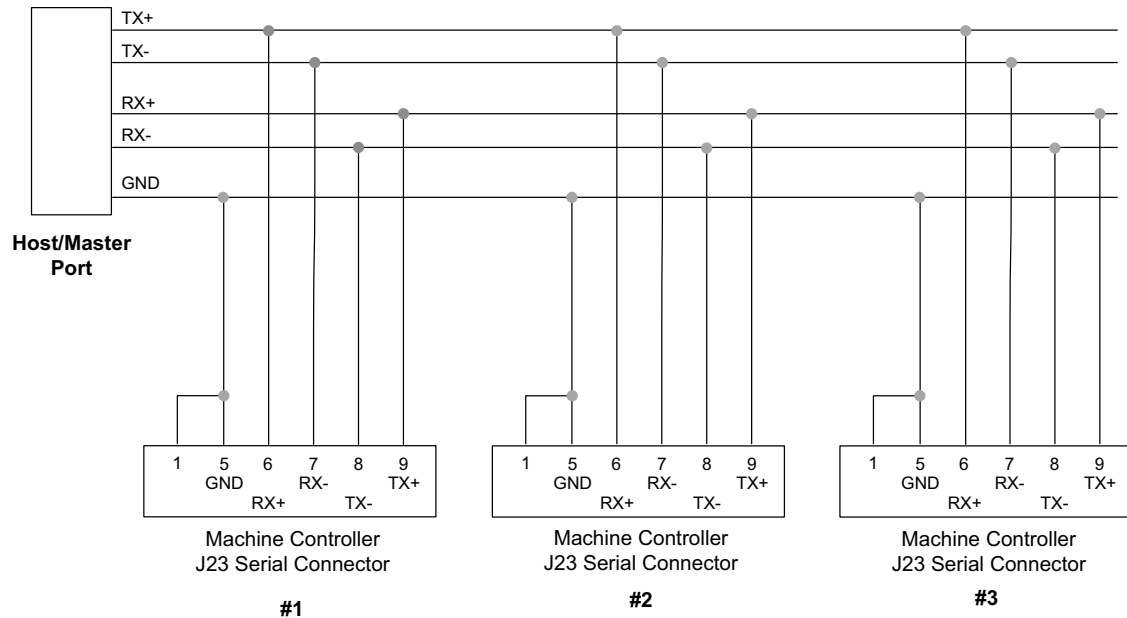


Figure 4-2:  
Example Setup  
for  
Communica-  
tions via  
RS485

Prodigy/CME Machine-Controller boards support RS485 communications via the Serial1 interface which is accessed at J23. Communicating by RS485 is typically less of a 'plug and play' process than communicating by RS232 however, and requires the user to create their own cables. For more information on the machine controller board's RS485 interface refer to the *Prodigy/CME Machine Controller User Guide*.

**Figure 4-3:  
Prodigy/CME  
Machine-  
Controller  
RS485 Signal  
Connection  
Diagram**

RS422 is essentially a subset of RS485 which uses a similar differential electrical interface but has only a single node connection. We will therefore focus on RS485 and show how to wire a network of RS485-connected machine controller.



RS485 is a master slave system. When connecting to a PC running Pro-Motion the PC functions as the master, and each PMD controller in the network functions as a slave. [Figure 4-3](#) shows the wiring scheme that is used for a full duplex (four-wire) connection. Four wire is recommended over two-wire because it has higher reliability. If two-wire (half duplex) is being used the receive signals (Rx+, Rx-) are not connected.

As can be seen from the wiring diagram, in addition to the RS485 signals themselves the machine controller board uses a ground connection to pin 1 of its J23 serial connector. A ground at pin 1 signals to the machine controller board that it should use RS485 rather RS232 communications.

For reliable communications the last unit on both sides of the network cable should provide termination. For most systems 120 ohms of termination resistance is recommended. Additional details such as what wire type to use, wire shielding, twisting differential wire pairs, length of wire etc. will not be detailed here but are important so a reference text on RS485 networks should be consulted.

To prepare the PC to function as the RS485 master an RS422/485 USB adaptor or RS485 card should be purchased and installed. Follow the product directions to install the software drivers. If properly installed and visible to Windows, Pro-Motion should recognize the RS485 port and be able to communicate with it automatically. For convenience the following table lists a vendor and P/N for such an adaptor product:

Vendor	P/N	Description
Advantech	BB-USOPTL4	Isolated high retention USB to RS422/485 converter (USB cable included)

### 4.1.5 Setting Up the PMD Controller for RS485 Communications

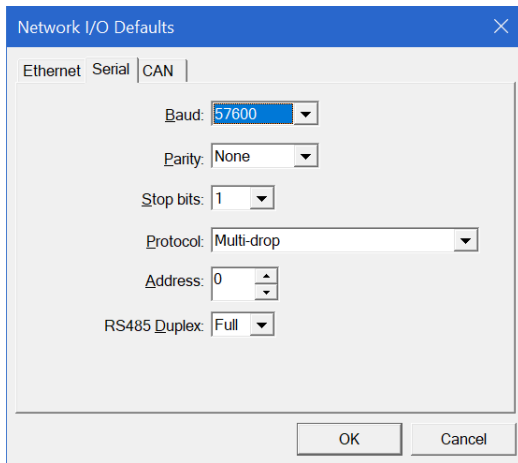
Each PMD controller unit to be installed in the RS485 network needs to be programmed with network settings. These settings can be programmed using Pro-Motion communicating to the PMD controller via the RS232 USB to DB9 cable.

The table below provides a summary of the available RS485 network settings as well as their default values.

Parameter	Default Setting	Setting Options	Comment
Address	0	0-31	Each PMD controller on the RS485 network should be programmed with a different node address.
Duplex (full/half)	full duplex	half duplex full duplex	If using a four wire connection scheme full duplex should be selected. Two wire schemes should select half duplex.

Here is the Pro-Motion sequence used to set RS485 parameters in the PMD controller.

- 1 With Pro-Motion RS232 communications via the USB to DB9 cable functioning properly, click the Device Control toolbar icon. The Device Control window appears.
- 2 Click Network I/O. The Network I/O Defaults dialog box appears.
- 3 Click the Serial tab. This window appears with data entry fields for the default serial port setting such as the baud rate, protocol, address, etc. This is shown below with default values visible.



- 4 Enter the desired communication settings in the corresponding data fields. The protocol should be set to Multi-drop and an address should be entered. The RS485 Duplex field should be set to match the actual wiring scheme used.
- 5 Click OK to store as the power on default in the PMD controller's NVRAM.
- 6 Disconnect RS232 communications using the Disconnect toolbar button.
- 7 Power down the PMD controller.
- 8 With the PMD controller powered off unplug the USB to DB9 RS232 cable and connect the RS485 cables according to [Figure 4-3](#).
- 9 Repower the PMD controller.

The PMD controller has now been programmed with new communications settings and is ready for operating via RS485 when powered up.

## 4.1.6 Setting Up Pro-Motion for RS485 Communications

With a RS422/RS485 USB adapter or other RS485 network card installed in the PC and drivers loaded, the instructions below will connect the host PC running Pro-Motion via RS485 to the PMD controller programmed for RS485 communications.

To set Pro-Motion RS485 parameters:

- 1 With no RS232 connection to the PMD controller operating, click the Connect toolbar icon.
- 2 Select Serial from the list of options.
- 3 Set the PC comm port that the RS485 connection is located at and press OK
- 4 Enter the RS485 communication settings. The programmed baud rate should match the RS485 parameters entered in the previous section, the protocol should be programmed to multi-drop, and the address/node ID of the PMD controller you are connecting to should be entered.
- 5 When complete, click OK.

If RS485 communication is successful a graphical icon representing the PMD controller on the network will be loaded into the Project window. If communication is not successful, a Communications Timeout Error dialog box appears. If this happens, recheck the connections, and retry to establish communications with the PMD controller.

If after rechecking your connections communication is still not correctly established you may want to connect via RS232 and check your RS485 settings. This can be accomplished by re-installing the USB to DB9 cable in J23, powering up the machine controller, and selecting Connect from the top icon menu bar. Set the communication settings to the same address, baud rate, and parity as set in [Section 4.1.5, “Setting Up the PMD Controller for RS485 Communications”](#) and set the protocol to multi-drop. Communications should re-establish and you can now check or change your RS485 settings using the instructions in [Section 4.1.5, “Setting Up the PMD Controller for RS485 Communications”](#) and then retry to connect using the RS485 cable as detailed in [Section 4.1.6, “Setting Up Pro-Motion for RS485 Communications.”](#)

## 4.1.7 Setting Up Multiple Units on an RS485 Network

[Section 4.1.5](#) and [Section 4.1.6](#) show how to program a single PMD controller and configure Pro-Motion so that correct communication are established. This is useful to verify that the RS485 network cabling and RS485 communication settings are correct.

To set up multiple PMD controllers on the RS485 network a typical sequence used is as follows:

- Program every unit that will be on the network, one by one, using the procedure in [Section 4.1.5, “Setting Up the PMD Controller for RS485 Communications”](#) however do not yet install the RS485 cables and do not execute the last step powering the PMD controller on.
- Install all of the programmed PMD controller units into the RS485 network, connect the RS485 network to the PC, and provide power to all of the PMD controllers on the RS485 network.
- Configure Pro-Motion to connect to each unit on the network, one-by-one, using the procedure described in [Section 4.1.6, “Setting Up Pro-Motion for RS485 Communications.”](#) So initially one unit will be connected on the network, then a second, etc. until all units are connected

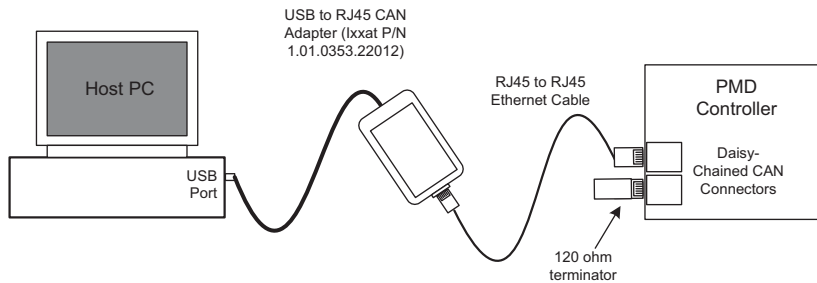
If successful, at the end of this procedure the Project window will display the entire list of units on the network, and by selecting (clicking on) a particular unit you can talk to that PMD controller directly. Once the network is installed

and configured in Pro-Motion you can save this network configuration using the *File/Save Project* menu function. To recall this configuration use the *File/Open Project* function.

## 4.2 CAN Communications

The Prodigy/CME Machine-Controller supports the CANbus 2.0 communication standard. For more information on the machine controller board's CAN interface refer to the *Prodigy/CME Machine-Controller User Guide*.

### 4.2.1 CAN Hardware Setup



**Figure 4-4:**  
Hardware  
Setup for  
Communicat-  
ing via CAN

For your PC to communicate via CAN you will typically use a USB to CAN converter which provides an RJ45 interface. The machine controller board directly accepts the RJ45 CAN connector at either J21 or J22. Either socket can be used because each signal is daisy chained from one connector to the other.

For reliable communications units on both ends of the CAN network should provide termination. For most systems 120 ohms of termination resistance is recommended. For connection to a single machine controller the TRM-RJ05-02 termination device, which is included with the DK, can be inserted into the unused J21 or J22 CAN connector.

For the host PC side of the network, if the CAN adapter being used has two daisy chained RJ45 connectors then a TRM-RJ05-02 can be installed in the unused connector. Alternatively some CAN adapters provide software settable or jumper settable engagement of termination. To order more TRM-RJ45-02 devices contact your local PMD representative.

Although not recommended for the production application, CAN network communication will often function correctly even if only one end of the network is terminated. So for bench-top checkout and application development use of only one terminator may be acceptable.



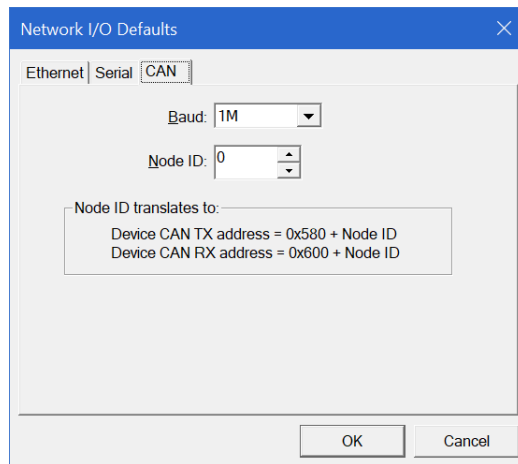
The following table provides a list of commercially available products that you may find useful for CAN communications:

Vendor	P/N	Description	Included with Machine Controller DK
lxxat	1.01.0281.12002	USB to CAN connector, V2 IXCAN, RJ45 interface	No
CableWholesale	10x8-56101	6' Cat 6 RJ45 to RJ45 connector cable	Yes

## 4.2.2 Setting Up the PMD Controller for CAN Communications

Here is the Pro-Motion sequence used to set up a PMD controller to connect via its CAN host interface.

- 1 With the USB to DB9 RS232 cable connection in place and operating, click the Device Control toolbar icon. The Device Control window appears.
- 2 Click Network I/O. The Network I/O Defaults dialog box appears.
- 3 Click the CAN tab. This window appears with data entry fields for the bit rate and the node ID. This is shown below with default values visible.



- 4 Enter the desired baud rate and Node ID in the corresponding data fields. Node IDs may have a value from 0 to 127. The CAN transmit address is derived from the selected Node ID by adding 0x580 and the CAN receive address is derived by adding 0x600. For example if a Node ID of 5 is specified the CAN transmit address will be 0x585 and the CAN receive address will be 0x605.

Although the PMD controller may be programmed with a Node ID of 0, it is recommended that 0 not be used as one of the programmed network addresses. The reason is that the default Node ID of PMD controllers is 0, and therefore keeping the 0 address reserved allows an unprogrammed PMD controller to be directly plugged into the CAN network and communicated with at Node ID 0.

- 5 Click OK to store as the power on default in the PMD controller's NVRAM
- 6 Disconnect RS232 communications using the Disconnect toolbar icon.
- 7 Power down the PMD controller and then after a pause of a few seconds repower the PMD controller.

The PMD controller has now been programmed to be ready for CAN communications.

## 4.2.3 Setting Up Pro-Motion For CAN Communications

With a CANbus 2.0 adapter and driver installed in the PC, the instructions below will connect the host PC running Pro-Motion via CAN to the PMD controller programmed for CAN communications.

To setup Pro-Motion to communicate by CAN:

- 1 Click the Connect toolbar button.
- 2 Select CAN, and then click OK.
- 3 Enter the same baud rate and Node ID as was programmed into the PMD controller previously.

- 4 When complete, click OK.

If CAN communication is successful, a set of graphical icons representing your PMD controller will be loaded into the Project window. If communication is not successful, after about 30 seconds, a Communications Timeout Error dialog box appears. If this happens recheck your connections and retry to establish CAN communications. If desired, you can reconnect via the RS232 connections to check or set the CAN settings as detailed in [Section 4.2.2, “Setting Up the PMD Controller for CAN Communications.”](#)

## 4.2.4 Setting Up Multiple Units on a CAN Network

[Section 4.2.2](#) and [Section 4.2.3](#) show how to program a single PMD controller and configure Pro-Motion so that CAN communication is established.

To set up multiple units on the CAN network a typical sequence used is as follows:

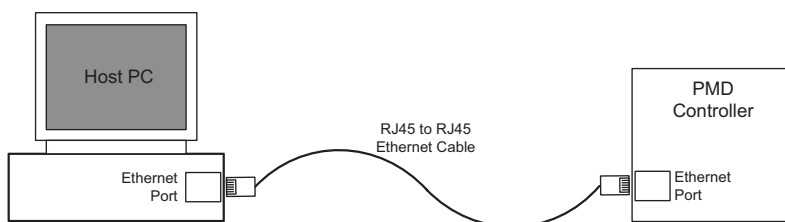
- Program every unit that will be on the network, one by one, using the procedure in [Section 4.2.2, “Setting Up the PMD Controller for CAN Communications.”](#) however do not power the PMD controller on. Note that all units on the network must use the same baud rate.
- Install all of these programmed units into the CAN network and provide power to all PMD controllers on the network.
- Configure Pro-Motion to connect to each unit on the network, one-by-one, using the procedure described in [Section 4.2.3, “Setting Up Pro-Motion For CAN Communications.”](#) So initially one unit will be connected on the network, then a second, etc. until all units are connected.

If successful, at the end of this procedure the Project window will display the entire list of units on the network, and by selecting (clicking on) a particular unit you can talk to that PMD controller directly. Once the network is installed and configured in Pro-Motion you can save this network configuration using the *File/Save Project* menu function. To recall this configuration use the *File/Open Project* function.

## 4.3 Ethernet Communications

Prodigy/CME Machine-Controller boards support 100 Base-T Ethernet Communications. For detailed information on the machine controller’s Ethernet interface refer to *Prodigy/CME Machine-Controller User Guide*.

### 4.3.1 Ethernet Hardware Setup



**Figure 4-5:**  
Hardware Setup for Communicating via Ethernet

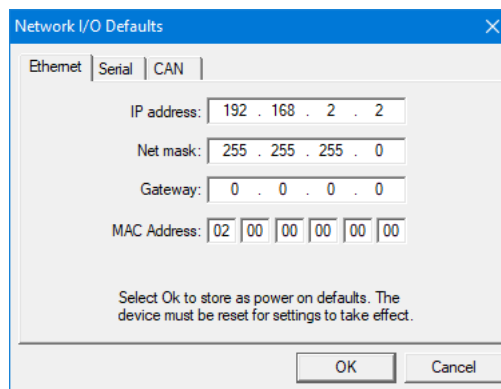
To connect to the machine controller via Ethernet you will use the included RJ45 to RJ45 cable which plugs into J24 on the machine controller board. There are two common ways to set your PC up to communicate with a PMD controller via Ethernet. The first is to install an Ethernet NIC (Network Interface Card) in the PC, and the second is to plug the PMD controller Ethernet connection into a spare connector on the same network your PC is connected to.

The NIC approach has the advantage that IP addresses are guaranteed to not conflict with other devices on the network, but both approaches can work. To determine the most suitable approach consult with your network administrator.

### 4.3.2 Setting Up the PMD Controller for Ethernet Communications

Here is the Pro-Motion sequence used to set up the PMD controller to connect via its Ethernet host interface.

- 1 With the USB to DB9 RS232 cable connection in place and operating, click the Device Control toolbar icon. The Device Control window appears.
- 2 Click Network I/O. The Network I/O Defaults dialog box appears.
- 3 Click the Ethernet tab. This window appears with data entry fields for the IP Address, the Net Mask, and the gateway. This is shown below with default values visible.



- 4 Enter the IP Address in the corresponding data field as well as the net mask and gateway if this is required for your network.



For a typical installation you will not change the netmask and gateway default values, but you must specify a valid, unique IP Address. If you are not sure what IP addresses are free and available for your Ethernet network contact your system administrator.

- 5 Click OK to store as the power on default in the PMD controller's NVRAM
- 6 Next disconnect RS232 communication using the Disconnect icon in the top icon menu bar.
- 7 Power down the PMD controller and then after a pause of a few seconds repower it.

The PMD controller has now been programmed to be ready for Ethernet communications.

### 4.3.3 Setting Up Pro-Motion for Ethernet Communications

With an Ethernet cable installed between the PC and the PMD controller, the instructions below will connect the host PC running Pro-Motion via Ethernet.

To setup Pro-Motion to communicate by Ethernet:

- 1 Click the Connect toolbar icon.



- 2 Select Ethernet, and then click OK.
- 3 Enter the same IP Address as was programmed into the PMD controller previously.
- 4 When complete, click OK.

If Ethernet communication is successful, a set of graphical icons representing your PMD controller will be loaded into the Project window. If communication is not successful, after about 30 seconds, a Communications Timeout Error dialog box appears. If this happens recheck your connections and retry to establish Ethernet communications. If desired, you can reconnect via the RS232 connections to check or change the Ethernet settings as detailed in [Section 4.3.2, “Setting Up the PMD Controller for Ethernet Communications.”](#)

### 4.3.4 Setting Up Multiple Units on an Ethernet Network

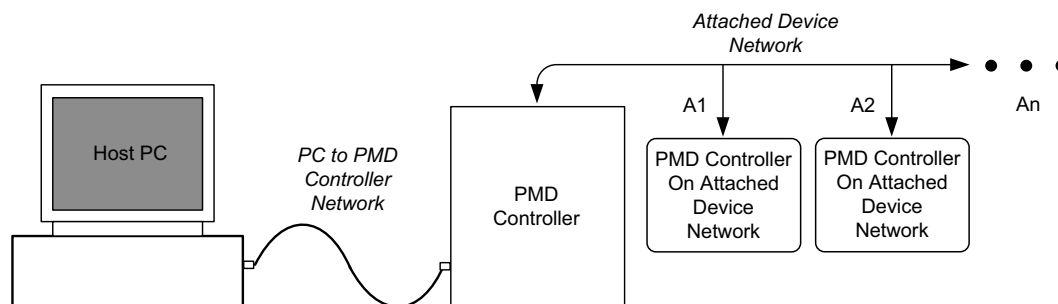
[Section 4.3.2](#) and [Section 4.3.3](#) show how to program a single PMD controller and configure Pro-Motion so that correct communication to the PMD controller is established. This is useful to verify that the Ethernet network cabling and Ethernet communication settings are correct.

To set up multiple units on the Ethernet network via an Ethernet switch a typical sequence used is as follows:

- Program every unit that will be on the network, one by one, using the procedure in [Section 4.3.2, “Setting Up the PMD Controller for Ethernet Communications.”](#) however do not power the PMD controller on. Note that each unit on the network must be set to a unique IP address.
- Install all of these programmed units into the network and provide power to all PMD controllers on the Ethernet network.
- Configure Pro-Motion to connect to each unit on the network, one-by-one, using the procedure described in [Section 4.3.3, “Setting Up Pro-Motion for Ethernet Communications.”](#) So initially one unit will be connected on the network, then a second, etc. until all units are connected

If successful, at the end of this procedure the Project window will display the entire list of units on the network, and by selecting (clicking on) a particular unit you can talk to that unit directly. Once the network is installed and configured in Pro-Motion you can save this network configuration using the *File/Save Project* menu function. To recall this configuration use the *File/Open Project* function.

## 4.4 Communicating With Attached Devices



The figure above shows a PC connecting to a PMD controller which supports what is known as an attached device network. Attached devices are PMD controllers (or non-PMD controllers) connected via a network to the PMD

**Figure 4-6:**  
PMD Controller  
with Attached  
Device  
Network



controller which in turn is connected to the PC. In the case of the Prodigy/CME Machine-Controller one type of attached device network is supported which is CANbus 2.0.

The PMD controllers that can be connected to the Prodigy/CME Machine-Controller CAN attached device network must be Magellan protocol devices. PRP protocol devices are not supported on the CAN machine controller attached device network. To view the list of Magellan protocol PMD products refer to [Section 5.1.4, “C-Motion SDKs.”](#)

#### 4.4.1 Setting Up CAN Network Attached Devices

In the next few sections we will provide an example of connecting Pro-Motion to devices on an attached device network. The devices to be connected are PMD ION 500 Digital Drives with CANbus host interface. ION 500 Digital Drives are intelligent single axis drives with amplification that can drive DC Brush, Brushless DC, or step motors.

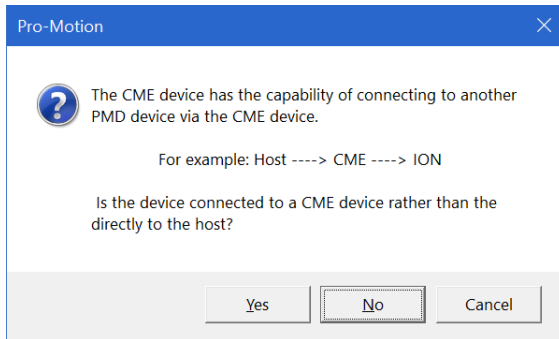
Each ION 500 unit on the attached device network should be programmed with the network baud rate and a unique Node ID. As is the case for regular CAN networks, it is recommended that Node ID assignments reserve Node ID 0 and only use values in the range of 1 – 127. For the sake of brevity the instructions for programming each CAN ION 500 unit to be connected on the PMD controller’s attached device network will not be repeated here. For a description of how to do this refer to the *ION Digital Drive User Manual*.

#### 4.4.2 Setting Up Pro-Motion for Attached Network Communications

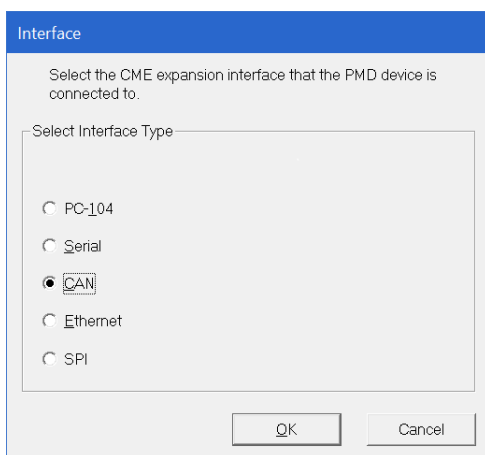
Setting up Pro-Motion for attached network connections first requires a connection be in place with the PC-connected PMD controller. In the instructions below the PC-connected PMD controller is a Prodigy/CME Machine-Controller board and the connection is the RS232 connection that was used with the quick setup instructions.

- 1 Connect the USB to DB9 serial cable to the Prodigy/CME Machine Controller at J23.
- 2 Connect the ION 500 unit at its host CAN connector to the machine controller CAN network connector (J21 or J22) using an RJ45 to RJ45 male/male cable such as listed in the table in [Section 4.2.1, “CAN Hardware Setup.”](#) In addition, you should install at least one 120 ohm terminator (PMD P/N TRM-RJ05-02) in either the ION or machine controller.
- 3 Power up the machine controller and the ION.
- 4 Click the Connect toolbar button.
- 5 Click Serial, select the USB to DB9 RS232 com port that the machine controller is connected to, and then click OK. The Serial Port dialog box displays with default communication values of 57,600 baud, no parity, 1 stop bit, and point-to-point protocol.
- 6 Click OK without changing any of these settings. If RS232 communication is correctly established, a graphic object for the machine controller will load into the Project window.

- Next Click the Connect toolbar icon again. A dialog box will appear as shown below indicating that since you are already connected (via Serial) to the machine controller this connection will be for an attached device network.



- Select Yes. Next an Interface Dialog box will display as shown below:



- Select CAN, and then click OK.
- Enter the same baud rate and Node ID as was programmed into the ION 500 unit previously.
- When complete, click OK. If CAN communication is successful, a set of graphical icons representing the attached device network N-Series ION will be loaded into the Project window.

If communication is not successful, after about 30 seconds, a Communications Timeout Error dialog box appears. If this happens recheck your connections and retry to establish CAN communications.

CAN communication setup is now complete for this attached ION unit.

### 4.4.3 Setting Up Multiple ION Units On the Attached Device Network

[Section 4.4.1](#) and [Section 4.4.2](#) show how to program a single ION 500 and configure Pro-Motion so that it communicates on the attached device network. This is useful to verify that the CAN network cabling and communication settings are correct.

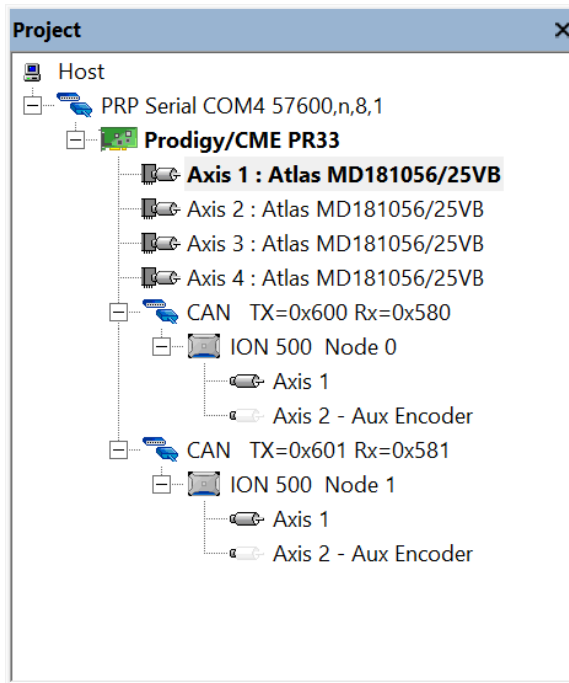
To set up multiple IONs on the CAN attached device network a typical sequence used is as follows:

- Program every unit that will be on the network, one by one, using the procedure in [Section 4.4.1, “Setting Up CAN Network Attached Devices.”](#)
- Install all of these programmed IONs into the network. Be sure to install a 120 ohm terminating resistor in one of the PC-connected PMD controller CAN connectors and in the last ION on the network. You

may use the included TRM-RJ45-02 device for this purpose. To order more TRM-RJ45-02 devices control your local PMD representative.

- Follow the instructions in [Section 4.4.2, “Setting Up Pro-Motion for Attached Network Communications”](#) step 1 through step 6 once, and then follow the instructions step 7 through 11 for each ION unit on the network, one-by-one until all units are connected.

If successful, at the end of this procedure the Project window will display the PC-connected PMD controller as well as the list of units on the attached device network. An example of this is shown in the Project window screen capture below.



This motion setup consists of a four-axis Prodigy/CME Machine-Controller which hosts two CAN ION 500 Digital Drives connected on an attached device network.

The Pro-Motion user can address any of these axes simply by clicking on an axis in the Project window, regardless of whether they are directly connect to the PC running Pro-Motion, or connected to the PC via the machine controller’s attached device network.

The ability of PMD controllers to provide this ‘pass through’ communications capability is a powerful feature of PMD products generally, and the PRP (PMD Resource access Protocol) specifically. It enables the creation of hierarchical communication structures that help spread out and manage motion network traffic.

Once the network is installed and configured in Pro-Motion you can save this network configuration using the *File/Save Project* menu function. To recall this configuration use the *File/Open Project* function.

# 5. Software Development

## In This Chapter

- ▶ Overview of C-Motion
- ▶ Getting Started with C-Motion PRP
- ▶ PC User Code Development
- ▶ C-Motion Engine User Code Development

This chapter provides a general introduction to creating software for PMD controller products. The focus will be on C-Motion, PMD's C-language libraries but PMD also provides .NET language support which enables software to be written in C# or VisualBasic.

For a complete description of PMD's software language support, tools, and design examples refer to the *C-Motion Engine Development Tools Manual*.

## 5.1 Overview of C-Motion

The C-language programming system used to develop user application code for PMD products is called C-Motion. C-Motion is a set of callable C-language routines that provide many features including:

- Axis virtualization
- Ability to communicate to multiple PMD motion ICs, boards, or modules
- Ability to communicate via RS232, RS485, CAN, Ethernet, SPI (Serial Peripheral Interface), or 8/16 bit parallel bus
- Provided as source code, allowing easy compilation & porting onto various run-time environments including PCs, user-designed boards, or C-Motion Engines

Broadly speaking there are two different ways user application code written in C-Motion can be used to control a PMD controller; the user application program can run on a host separate from the PMD controller, or the user application program can run directly on the PMD controller in a code execution module called the C-Motion Engine. All PMD products can support user application code running on a separate host, while only PMD products with a "/CME" designation in their product name contain a C-Motion Engine.

Here is more information on these two different ways of executing the user application code.

### 5.1.1 Host-based Execution of User Code

When located on a host controller the user code communicates via one of the available host interfaces to the PMD controller. Depending on the PMD product being used this may be point-to-point serial, multi-drop serial, CAN, Ethernet, SPI (Serial Peripheral Interface), or PC/104. The format of these communications is one of two packet based protocols depending on the PMD product being used; PRP protocol, which is short for PMD Resource Access

Protocol, or Magellan protocol. The user need not be concerned with the packet format however because these details are handled automatically when code is written in C-Motion.

For more information on the Magellan protocol refer to the *C-Motion Magellan Programming Reference*. For more information on the PRP protocol refer to the *C-Motion PRP Programming Reference*.

## 5.1.2 C-Motion Engine-based Execution of User Code

When executed on the PMD controller's C-Motion Engine the user code communicates internally to the resources available on the controller such as the Magellan Motion Control IC. This has speed advantages both in communicating with those resources and in real time code execution predictability. The software tools used to compile and debug C-Motion code when run on the C-Motion Engine are contained in the SDK (Software Development Kit) provided by PMD.

Executing the code directly on the C-Motion Engine allows the controller to function as a fully standalone controller. In this mode a host controller network communication link is not needed, and one or more of the PMD controller's communication ports or digital I/O ports are typically used to interface to user-operated buttons or a touch screen.

Alternatively, code can be executed on the PMD controller's C-Motion Engine that processes commands which are received from a host network, thus forming an application-specific local controller within a larger system. For example for a device with a C-Motion Engine controlling a three-axis gantry a host may send high level commands which are interpreted to mean "move the gantry to location X, Y, Z". The user code executing on the C-Motion Engine parses these incoming commands and generates axis-specific motions for each of the three controlled motion axes to execute the high level host command.

## 5.1.3 PMD Products & User Code Execution Options

The following table shows specific types of PMD products and options for running the user application code:

PMD Product Type	Application Code Runs On	System Description
PMD Motion Control IC	Microcontroller	<b>User-designed board.</b> A microcontroller sends commands to a PMD motion control IC, both of which are located on a user-designed board. Allows standalone operation.
PMD Motion Control IC	PC, user-designed controller, or other controller	<b>Host-connected user-designed board.</b> A PC, user-designed controller, or other controller sends commands via a network connection to a PMD motion control IC which is located on a user-designed board.
PMD ION Drive or Prodigy Board	PC, user-designed controller, or other controller	<b>Host-connected ION drive or Prodigy board.</b> A PC, user-designed controller, or other controller sends commands via a network connection to a PMD ION drive or Prodigy board.
PMD ION/CME * Drive or Prodigy/CME Board	C-Motion Engine	<b>ION/CME drive or Prodigy/CME board.</b> User application code runs on the PMD controller's C-Motion Engine. Allows standalone operation.

\*PMD products which support a C-Motion Engine have a "/CME" in their product name, for example ION/CME N-Series Digital Drive.

## 5.1.4 C-Motion SDKs

There are three different C-Motion SDKs; C-Motion Magellan, C-Motion PRP, and C-Motion PRP II. All of these SDKs are available from the PMD website at <http://www.pmdcorp.com/resources/software>. Here is more information on each:

- **C-Motion Magellan SDK** – an SDK for creating host-based user applications for PMD products that utilize a Magellan or Juno formatted protocol.
- **C-Motion PRP SDK** – an SDK for creating host-based and downloadable C-Motion Engine-based user code for systems utilizing either a PRP (PMD Resource Access Protocol) protocol device or a Magellan/Juno protocol device. The C-Motion PRP SDK is also used in motion applications that will use the .NET (C#, VB) programming languages.
- **C-Motion PRP II SDK** – This SDK is similar to C-Motion PRP but is used with ION/CME N-Series Digital Drives. Compared to standard C-Motion PRP, C-Motion PRP II supports additional features such as multi-tasking, mailboxes, mutexes, and enhanced event management.

For reference the following table shows the packet protocol and C-Motion SDKs that can be used with each PMD product family:

Product Family	Packet Protocol	C-Motion SDKs
Magellan MC581 I3 Family ICs & DKs	Magellan	C-Motion Magellan, C-Motion PRP*
Magellan MC5x000 Family ICs & DKs	Magellan	C-Motion Magellan, C-Motion PRP*
Juno MC781 I3 Family ICs & DKs	Magellan**	C-Motion Magellan, C-Motion PRP*
ION/CME N-Series	PRP	C-Motion PRP II
ION 500 (except Ethernet)	Magellan	C-Motion Magellan, C-Motion PRP*
ION 500 with Ethernet interface	PRP	C-Motion PRP
ION/CME 500	PRP	C-Motion PRP
ION 3000	Magellan	C-Motion Magellan, C-Motion PRP*
Prodigy PC/104	Magellan	C-Motion Magellan, C-Motion PRP*
Prodigy/CME PC/104	PRP	C-Motion PRP
Prodigy/CME Stand-Alone	PRP	C-Motion PRP
Prodigy/CME Machine-Controller	PRP	C-Motion PRP

\* With this product C-Motion PRP typically only used for .NET support, or if a mix of Magellan protocol and PRP protocol devices are attached.

\*\* Although the Juno IC command set is somewhat different than the Magellan IC command set, the overall packet format is the same and therefore also referred to as a Magellan packet protocol.

From C-Motion Magellan to C-Motion PRP to C-Motion PRP II, each 'higher' level is a functional superset of the previous, and so when using multiple PMD products the highest required SDK should be used. For example for a PC-based application that communicates both with N-Series ION drives (which by themselves require the C-Motion PRP II SDK) and with Prodigy/CME Machine-Controllers (which by themselves require the C-Motion PRP SDK) the C-Motion PRP II SDK should be used.

Because higher SDKs are a superset of the lower level SDKs there may be situations where more than one SDK could be chosen and successfully used in the application. In subsequent sections, when this is the case, we will provide guidance to help with this choice.

## 5.2 Getting Started with C-Motion PRP

The version of C-Motion that is used with the Prodigy/CME Machine-Controller is C-Motion PRP. PRP stands for PMD Resource Access Protocol. While the details of how PRP packets are formatted is not important for most C-Motion PRP users, a general understanding of the underlying architecture is helpful.

PRP packets may be transmitted via serial, CAN, Ethernet TCP/IP, or SPI (Serial Peripheral Interface). PRP device functions are organized into *resources* and resources process *actions* sent to them. Actions can send information, retrieve information, or command specific events to occur. A basic communication to a PRP device consists of a command header and for some communications a message body. The message body, if present, contains data associated with the specified PRP command. The header contains various information used to process the PRP messages.

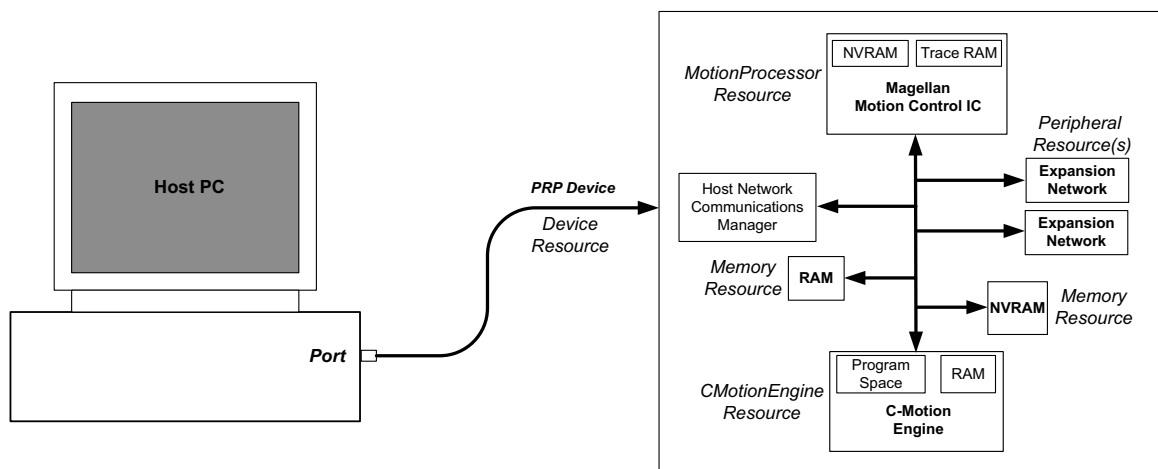
After a PRP communication is sent to a device a return communication is sent by the PRP device which consists of a response header and an optional return message body. The return message body may contain information associated with the requested PRP command, or it may contain error information if there was a problem processing the command.

Depending on the communication channel used PRP packets may also include checksum information. For example when sent via serial, checksum words are included, but when sent via Ethernet TCP/IP checksum words are not included because transmission integrity is handled by other layers of the Ethernet protocol.

PRP is a master/slave system. The host functions as the master and initiates communication sequences which the connected device responds to. The connected device can not initiate messages on its own within the PRP protocol. Note however that some PRP-supported networks, in particular CAN and Ethernet, allow one or more non-PRP protocol connections to be established to support asynchronous communication from the attached device to the host.

### 5.2.1 PRP Resources

**Figure 5-1:**  
PC-Connected  
to PRP Device  
Showing  
Resources



The above diagram shows a generalized PRP device, which is a PMD controller board or module that expects PRP formatted communication packets. All PMD Prodigy/CME boards, ION/CME drives and the Ethernet ION 500 drive are PRP devices. See [Section 5.1.4, “C-Motion SDKs.”](#) for a detailed product list.

There are five different resource types supported by PRP devices. The *Device* resource indicates functionality that is addressed to the entire board or drive module, the *MotionProcessor* resource indicates a Magellan Motion Control IC, the *CMotionEngine* resource indicates the C-Motion Engine, the *Memory* resource indicates RAM or non-volatile RAM (Random Access Memory), and the *Peripheral* resource indicates a communications connection.



Within the C-Motion PRP system C-language handles are used to reference each instance of a resource. For example if the host is communicating with two different PRP devices, each would have its own Device resource handle, and to send commands to one of those PRP devices the corresponding handle would be an argument in the C-Motion call. The same C-language handle approach applies to access Peripheral, MotionProcessor, Memory, and CMotionEngine resources.

## 5.2.2 Peripheral Resources

Peripheral resources are somewhat unique within the PRP system. While the MotionProcessor, Memory, and CMotionEngine resources refer to specific functional modules on the PRP device, Peripherals may be created dynamically because they represent a connection rather than a specific functional module.

For example if a PRP device supports an attached CAN network and that CAN network had 10 attached devices on it, each connection between the PRP device and the attached CAN network device utilizes a separate Peripheral handle. Not all Peripheral connections are created dynamically, for example non multi-drop connection ports such as an RS232 port would be referenced with a single Peripheral handle.

To initially create Peripheral handles one of a few 'open' calls are sent. For example the CMotion call **PMDPeriphOpenCom()** returns a handle to a newly opened Peripheral, with the peripheral type being a serial connection. The arguments specified with this command include serial parameters such as baud rate and parity. Once opened the Peripheral handle is used in calls that take a peripheral handle as an argument, for example commands that send or receive messages.

In addition to communication connections Peripherals can also refer to connections to a specific portion of the PRP device such as the PIO registers, which are used to interact with digital and analog I/O and set various system characteristics of the controller.

## 5.2.3 MotionProcessor Resources

The MotionProcessor resource is the name given for Magellan Motion Control ICs on a PRP device. Accessing the MotionProcessor resource is similar to the procedure detailed above, except the C-language handle is called an Axis handle, and each handle accesses just one axis. For multi-axis products such as Prodigy/CME Machine-Controllers an Axis handle can be created for each active axis, up to four. For single-axis products such as ION/CME drives two axis handles can be created, one for the primary axis and one for the auxiliary axis which provides just an encoder input.

There are numerous callable Motion-Processor routines that take an Axis handle as an argument. MotionProcessor resource commands are described in their own C-Motion manual, called *C-Motion Magellan Programming Reference*.

## 5.2.4 Going Further with C-Motion PRP

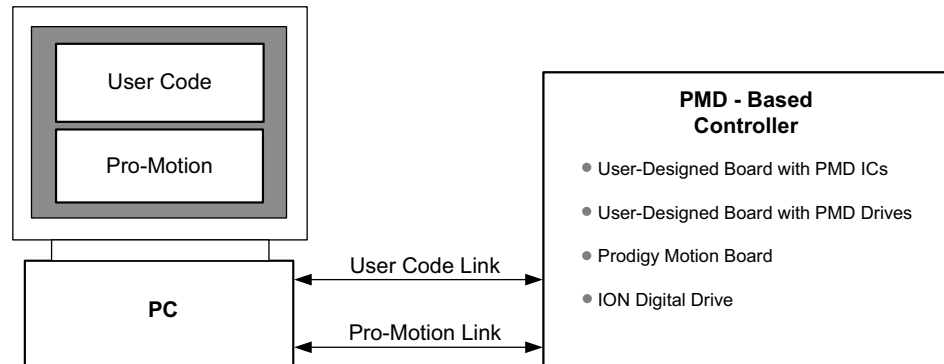
This brief introduction to C-Motion PRP is intended to provide some basic information about C-Motion library calls when user application code communicates with a PRP protocol PMD controller. For additional information including example command sequences and more about software development with PMD products refer to the *C-Motion Engine Development Tools Manual*. For a complete reference of PRP protocol commands refer to the *C-Motion PRP Programming Reference*.

## 5.3 PC User Code Development

PC user code means user code that runs on the PC. This is a popular approach for controlling off-the-shelf PMD board and module products, and also for controlling user-designed boards that contain PMD motion ICs or PMD PCB-mountable drive modules.

In addition to running user code written with C-Motion libraries, running on the PC is the most popular approach when coding in C# and other .NET languages.

**Figure 5-2:**  
Typical  
Connection  
Scheme for PC-  
Based User  
Code  
Development



There are typically two separate communication links used with PC-based code development as shown in [Figure 5-2](#):

### User Code Link

When the PC runs the user application code it uses a communication link to send commands to the PMD controller. This communication channel is called the User Code link. For PRP devices, the User Code link will carry PRP-formatted packets so that the PRP device can correctly interpret them and respond accordingly. For Magellan protocol devices the User Code link will carry Magellan-formatted packets. See [Section 5.1.4, “C-Motion SDKs,”](#) for a list of packet protocol types for various PMD products.

Depending on the communication channel used for the User Code link other devices may also be addressed on the same link, and these devices may or may not be a PMD controller. For example if an Ethernet TCP network is used as the User Code link the PMD controller will use just one IP Address, and other devices, PMD-based or not, may be on the same network as long as they use a different IP address.

### Pro-Motion Link (optional)

The Pro-Motion link uses a separate connection to allow Pro-Motion to communicate with the PMD controller. Although optional, running Pro-Motion and utilizing a Pro-Motion link as part of the code development setup has significant benefits because Pro-Motion can be used to investigate the status of the PMD controller before, during, and after execution of the host user application. For more information on Pro-Motion refer to [Chapter 3, \*Going Further with Pro-Motion\*](#).



When both the user application code and Pro-Motion are running care should be taken to avoid using Pro-Motion to command motions or make changes to the PMD controller that could conflict with commands being sent by the user application code.

### 5.3.1 Link Options for the Prodigy/CME Machine Controller

The communication link options that are available for different types of PMD controllers vary. The table below shows typical link options for User Code and Pro-Motion links when developing PC-based code to control a Prodigy/CME Machine-Controller.

User Code Link	Pro-Motion Link
CAN	Ethernet TCP
RS485	Ethernet TCP
Ethernet TCP port 40100	Serial I
CAN	Serial I

To set up Pro-Motion or PMD controllers for Ethernet TCP or Serial1 connections instructions can be found in [Section 4.3, “Ethernet Communications.”](#) or [Section 4.1, “Serial Communications.”](#) To set up Pro-Motion or PMD controllers for CAN or RS-485 connections instructions can be found at [Section 4.2, “CAN Communications.”](#) or [Section 4.1.4, “RS485 Communications.”](#)

### 5.3.2 Choosing the SDK For PC-Based User Code Development

In most setups where the host code will run on the PC in the production application there won't be a choice of C-Motion library SDK. For example if one or more of the devices being commanded are N-Series IONs, the SDK used for the PC-based user code must be C-Motion PRP II.

The main scenario where a choice may exist is if the setup consists only of a Magellan architecture device, for example a DK58113 board (the developer kit for the MC58113 IC) or a user-designed board with PMD motion control ICs on it. In this case both C-Motion Magellan SDK and C-Motion PRP SDK are viable choices.

In this situation many developers will opt for the C-Motion PRP SDK. The reason is that if in the future a PRP device is added to the controller setup there will be no need to switch SDK types. In other words since C-Motion PRP provides a superset function of C-Motion Magellan it is inherently more ‘future proof’.

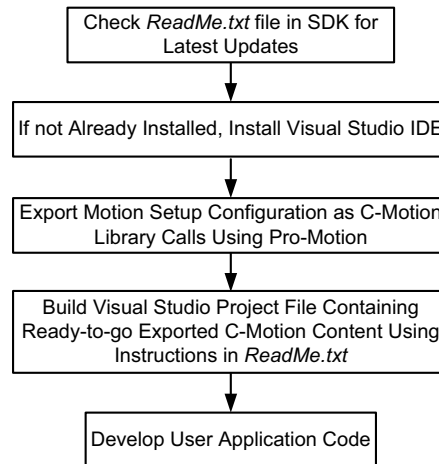
The main reason for choosing C-Motion Magellan is that the size of the C-Motion source code libraries is significantly smaller. While not a major consideration if the user code runs on the PC in the production application, smaller and simpler source code libraries may be important if the user application code will eventually be ported to run on a microcontroller on a user-designed board.

For more information on C-Motion SDKs refer to [Section 5.1.4, “C-Motion SDKs.”](#)

### 5.3.3 Code Development Process

Figure 5-3 shows a flowchart of the recommended process for developing PC-based user application code with PMD products.

**Figure 5-3:**  
Recommended  
Sequence for  
PC Code  
Development



A first step is to review the *ReadMe.txt* file that is included at the top level directory of the SDK you are using. This file will provide up-to-date information on example Visual Studio projects, source code examples, and other resources provided by the SDK.

The next step is to install Visual Studio IDE, if you haven't already installed it. Visual Studio is a software development environment made by Microsoft Corporation. The C-Motion SDK design examples are designed to be used with Visual Studio IDE.

Next, with Pro-Motion connected to the PMD controller(s) in the motion setup and with all other configuration settings such as gain settings and safety settings in place for your motion setup, execute a configuration export to C-Motion as detailed in [Section 3.10, "Configuration Export to C-Motion."](#) The output of this operation will be one or more C-language source files that will be incorporated into your initial user application code Visual Studio project.

The instructions for the final step to building your initial user code development Visual Studio project can be found in the *ReadMe.txt* file for the SDK you are using. This step combines the C-Motion source code library files with your exported configuration source code file(s) into a ready-to-compile and run Visual Studio project.

Before executing your user code project be sure Pro-Motion is running and connected via the Pro-Motion link. As mentioned earlier Pro-Motion can be very helpful as a debugging aid by using its Command window or status screens to confirm that commands from the user application code have resulted in the expected motions or changes in the PMD controller.

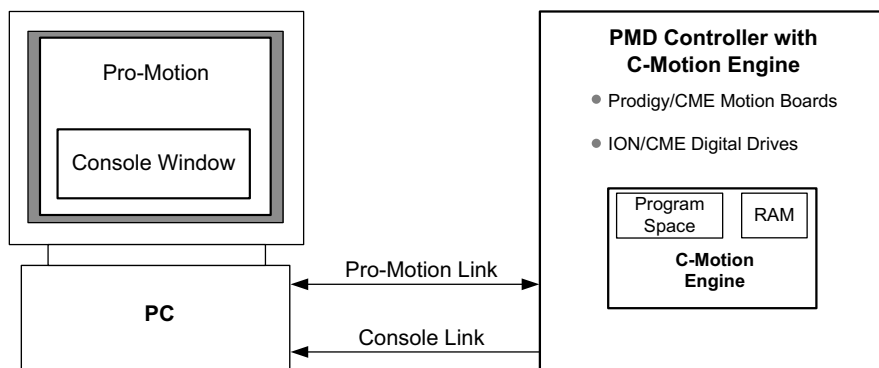
As you proceed with development of your application code the following three manuals will be especially useful. For detailed information on C-Motion PRP refer to *C-Motion PRP Programming Reference*. For detailed information on Magellan Motion Control IC commands refer to the *C-Motion Magellan Programming Reference*. For more detailed information on all aspects of developing software for PMD products refer to the *C-Motion Engine Development Tools Manual*.

## 5.4 C-Motion Engine User Code Development

C-Motion Engine user code development means the user application code runs on the C-Motion Engine module, which is a code execution module provided on some PMD controller products. PMD controllers which have “/CME” in their product name have a C-Motion Engine module. This includes the Prodigy/CME Machine Controller.

This section provides information on how to develop user code that runs on the C-Motion Engine.

### 5.4.1 Connections Overview



**Figure 5-4:**  
Typical  
Connection  
Links When  
User Code  
Runs on C-  
Motion Engine

[Figure 5-4](#) shows a typical connection scheme when the user code executes on the C-Motion Engine. There can be up to two separate communication links:

#### Pro-Motion Link

The Pro-Motion link connects the PC that runs Pro-Motion to the PMD controller containing the C-Motion Engine. This link allows the user code, once compiled and converted into a *.bin* file format, to be downloaded and eventually executed on the C-Motion Engine. In addition this link allows Pro-Motion to monitor the status of the motion system while the C-Motion Engine-based user application code is executing.

#### Console Link (Optional)

The console link provides a convenient pathway for sending printf statements from the user code running on the C-Motion Engine to a separate monitor or to the Pro-Motion Console window. The console link does not have a protocol as such, directly transmitting whatever ASCII messages are sent using printf commands by the C-Motion Engine user code.

While a console port connection may be useful, particularly in the earlier stages of user code development, not all systems will need or use a console channel link.

## 5.4.2 Link Options for the Prodigy/CME Machine-Controller

The communication link options that are available for different types of PMD controllers vary. The table below provides typical link options for Pro-Motion and Console links when developing code that is running on the Prodigy/CME Machine-Controller's C-Motion Engine.

Pro-Motion Link	Console Link
Ethernet TCP Port 40100	Ethernet UDP
Ethernet TCP Port 40100	Serial2
Ethernet TCP Port 40100	Ethernet UDP
Ethernet TCP Port 40100	Serial2
Ethernet TCP Port 40100	Ethernet UDP
Serial1	Ethernet UDP
Serial1	Serial2
Serial1	Ethernet UDP
Serial1	Serial2

Ethernet TCP is the most common channel for the Pro-Motion link connection, however if the development system is being powered on and off frequently, due to the relatively slow startup of Ethernet, Serial1 may be preferred as the Pro-Motion connection link.

Most developers will select Ethernet UDP as the console channel when operating with Pro-Motion, however there may be situations where an RS232 connection at Serial2 has benefits because it allows monitoring of the program without need for running Pro-Motion by streaming the serial output to a terminal or terminal emulator.

To set up the Pro-Motion and PMD controller connections, refer to [Section 4.1, "Serial Communications,"](#) or [Section 4.3, "Ethernet Communications."](#) To set the console connection type use the Console Output button located in the Pro-Motion Device Control window.

## 5.4.3 Choosing the SDK For C-Motion Engine Code Development

The C-Motion SDK used to develop code that runs on a C-Motion Engine must match the listed SDK in the table in [Section 5.1.4, "C-Motion SDKs."](#) For example for a Prodigy/CME Machine-Controller the SDK is C-Motion PRP, and only this SDK can be used.

## 5.4.4 Code Development Process

Figure 5-5 shows a flowchart of the recommended process for developing user application code that will run on the PMD controller's C-Motion Engine.

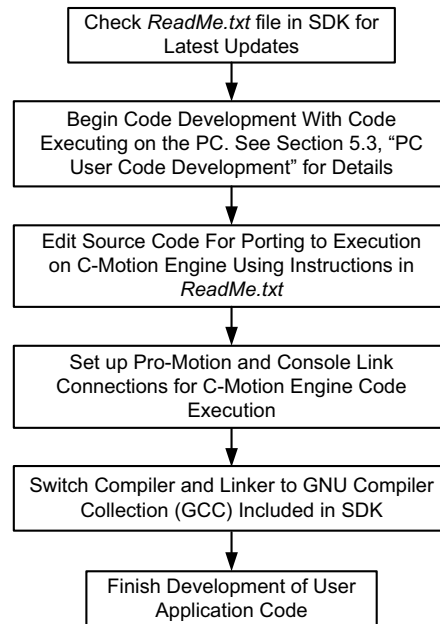


Figure 5-5:  
Recommended  
Sequence for  
C-Motion  
Engine Code  
Development

The recommended process for developing user application code that will run on the C-Motion Engine is to begin code development with the user code running on the PC and then later move code execution to the C-Motion Engine where code development can be completed.

Beginning application development with the user code running on the PC gives you access to resources such as Visual Studio which should make initial code development easier. Refer to [Section 5.3, "PC User Code Development,"](#) for information on connections and the recommended code development process when the user application code runs on the PC.

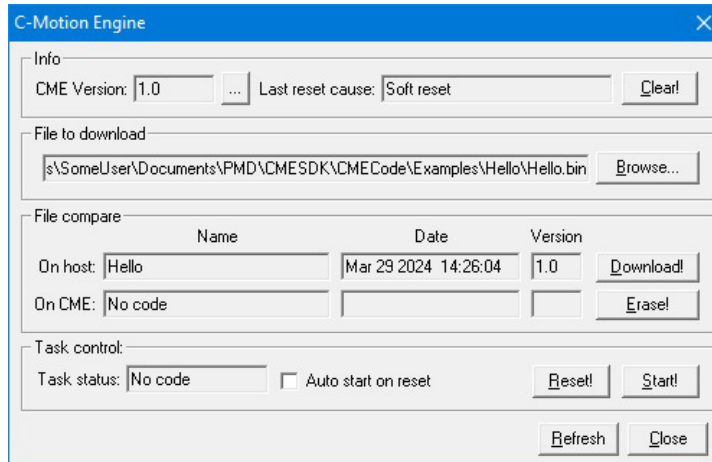
In this two-step approach, the typical focus of code development during the first step (when the code runs on the PC) is motion sequence development, machine performance measurement & optimization, and user interface development.

At some point however you will be ready to port the code to run on the C-Motion Engine. The timing of this switch may be influenced by several factors but a particularly important one is the fact that some features (particularly features provided by CMotion PRP II which is the SDK used with the N-Series ION drive) are not available when compiled on the PC. These features include multi-tasking, mutexes, mailboxes, and event processing. So if your code relies on those features you may decide to port the user code to execute on the C-Motion Engine earlier.

Porting code execution from the PC to the C-Motion Engine will require some code changes reflecting the fact that commands no longer come from the PC, they come from the C-Motion Engine. Nevertheless because of PRP's use of C-language handles to access resources this changeover is generally a straightforward process. For details of this porting process refer to the *The C-Motion Engine Development Tools Manual* and the *ReadMe.txt* file for the SDK you are using.

In addition to making changes to the source code you should also change the link connections from those used when developing code that runs on the PC, to those used when developing code that runs on the C-Motion Engine. For more on that refer to [Section 5.4.2, "Link Options for the Prodigy/CME Machine-Controller."](#)

Once ready to compile the code for the C-Motion Engine, you will no longer use the Visual Studio C-language compiler and instead use the GNU Compiler Collection (GCC) compiler and linker which is included in the C-Motion SDK. The *C-Motion Engine Development Tools Manual* provides detailed step by step information on how to edit, compile, and link the user application code resulting in a *.bin* file that can be downloaded to the C-Motion Engine.



Once a *.bin* file has been generated it can be transferred to the PMD controller's C-Motion Engine using the C-Motion Engine dialog box accessible from the Pro-Motion Device Control window. A screen capture of the C-Motion Engine dialog box is shown above. Pro-Motion 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.

When the code image loaded into the C-Motion Engine is ready to be executed there are two options to achieve this; manual start and auto-start. To manually start code execution leave *Auto start on reset* unselected. If auto-start is selected, after powerup or a reset the user code will begin executing automatically without user intervention.



C-Motion Engine auto-start should only be selected when the user code is mature and stable. Manually starting code execution at each board initialization avoids the possibility that the user code running on the C-Motion Engine contains a flaw resulting in corruption of the board's communication settings, making it difficult or impossible thereafter to communicate with the board via Pro-Motion.



# 6. Electrical Reference

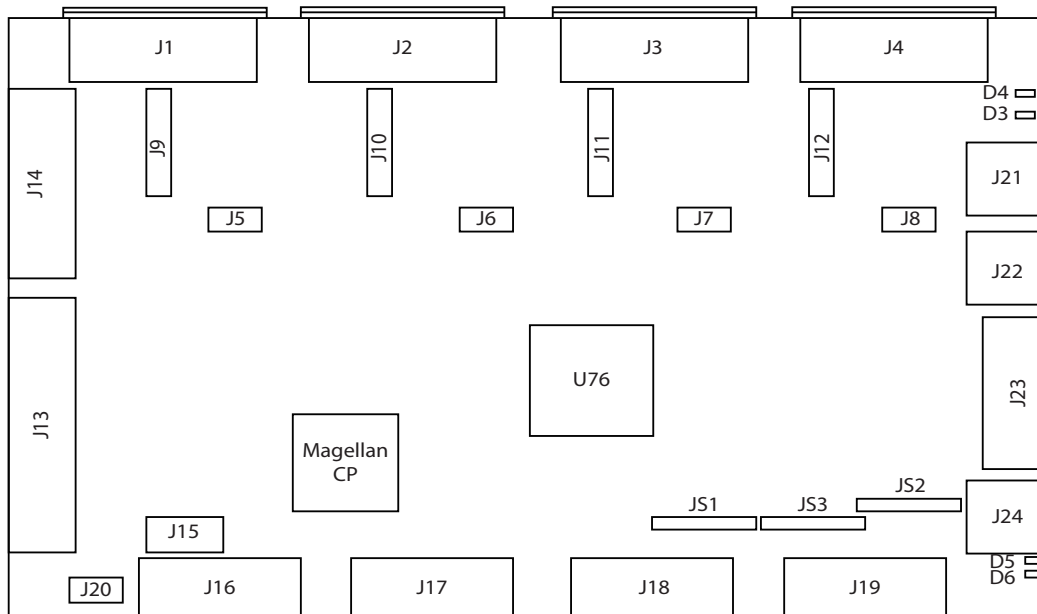
## In This Chapter

- ▶ User-Settable Components
- ▶ Connectors
- ▶ Motor Connections Quick Reference
- ▶ Cables & Accessories

## 6.1 User-Settable Components

[Figure 6-1](#) illustrates the locations of the principal components of the Prodigy/CME Machine-Controller boards. The user-settable components of the board are listed in the following table:

Component	Function
Resistor packs JS1, JS2, and JS3	Sets the encoder termination.



**Figure 6-1:**  
Components  
and Layout,  
Front of Board

### 6.1.1 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 used, resistor packs JS1, JS2, and JS3 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 JS1, JS2, JS3	Installed; this is the default setting of resistor packs JS1 - JS3.	If differential connections are being used, leave the resistor packs installed.
	Removed	If single-ended encoder connections are being used, remove the resistor packs.

### 6.1.1.1 Differential Encoder Connections

Differential encoder connections are detailed in the following table.

Signal	J16 - J19			
	Axis 1	Axis 2	Axis 3	Axis 4
QuadAn+	J16-1	J17-1	J18-1	J19-1
QuadAn-	J16-2	J17-2	J18-2	J19-2
QuadBn+	J16-3	J17-3	J18-3	J19-3
QuadBn-	J16-4	J17-4	J18-4	J19-4
Indexn+	J16-6	J17-6	J18-6	J19-6
Indexn-	J16-7	J17-7	J18-7	J19-7
Vcc	J16-14	J17-14	J18-14	J19-14
GND	J16-15	J17-15	J18-15	J19-15

### 6.1.1.2 Single-Ended Encoder Connections

Single-ended encoder connections are detailed in the following table.

Encoder connections when using single-ended encoder input:

Signal	J16 - J19			
	Axis 1	Axis 2	Axis 3	Axis 4
QuadAn	J16-1	J17-1	J18-1	J19-1
QuadBn	J16-3	J17-3	J18-3	J19-3
Indexn	J16-6	J17-6	J18-6	J19-6
Vcc	J16-14	J17-14	J18-14	J19-14
GND	J16-15	J17-15	J18-15	J19-15

### 6.1.1.3 Using Both Single-Ended and Differential Encoder Connections

When both single-ended and differential encoders are used on the same board a special arrangement of the connections is needed. If Axis 1 or Axis 2 has a single-ended encoder, remove the resistor packs installed on JS1 and JS3. JS1, JS2, and JS3 are the connectors that receive the RS1, RS2, and RS3 resistor packs, respectively. If Axis3 or Axis4 has a single-ended encoder, remove the packs installed on JS2 and JS3.

For any axis that has a differential encoder and the corresponding resistor pack has been removed, a 120 ohm resistor needs to be installed as follows:

	Axis 1	Axis 2	Axis 3	Axis 4
Channel A	JS1-1,2	JS1-5,6	JS2-1,2	JS2-5,6
Channel B	JS1-3,4	JS1-7,8	JS2-3,4	JS2-7,8
Index (Z)	JS3-1,2	JS3-3,4	JS3-5,6	JS3-7,8

Every cell in the table above represents an installed 120 ohm resistor at that location. For example JS1-1,2 implies that one terminal of the resistor is connected to JS1-1 and the other terminal is connected to JS1-2.

## 6.2 Connectors

There are 24 user-accessible connectors on the Prodigy/CME Machine-Controller board. See [Figure 6-1](#) for the specific locations of the connectors on the board. The connectors and their functions are outlined in the following table:

Connector Name	Connector #	Functionality
Axis 1 Atlas	J1	Provides socket for Axis 1 pluggable Atlas unit.
Axis 2 Atlas	J2	Provides socket for Axis 2 pluggable Atlas unit.
Axis 3 Atlas	J3	Provides socket for Axis 3 pluggable Atlas unit.
Axis 4 Atlas	J4	Provides socket for Axis 4 pluggable Atlas unit.
Axis 1 Power	J5	Provides power to the whole board plus Axis 1 Atlas unit. It accepts DC supply in the range of +12 – 56 VDC.
Axis 2 Power	J6	Provides power to the Axis 2 Atlas unit. It accepts DC supply in the range of +12 – 56 VDC.
Axis 3 Power	J7	Provides power to the Axis 3 Atlas unit. It accepts DC supply in the range of +12 – 56 VDC.
Axis 4 Power	J8	Provides power to the Axis 4 Atlas unit. It accepts DC supply in the range of +12 – 56 VDC.
Axis 1 Motor Drive	J9	Provides direct motor control signals for Axis 1.
Axis 2 Motor Drive	J10	Provides direct motor control signals for Axis 2.
Axis 3 Motor Drive	J11	Provides direct motor control signals for Axis 3.
Axis 4 Motor Drive	J12	Provides direct motor control signals for Axis 4.
General I/O	J13	Provides the following general I/O interfaces to the board: 8 channels of bi-directional digital I/O; 4 channels of digital inputs; 4 channels of digital outputs; 8 channels of differential analog.
Amplifier I/O	J14	Provides amplifier enable outputs for 4 axes. Provides 8 <i>AnalogOutput</i> signals for controlling external amplifiers or general purpose $\pm 10V$ analog output. Provides <i>AxisIn</i> and <i>AxisOut</i> connections for all four axes.
Expansion	J15	Provides synchronization of multiple Prodigy Machine-Controller boards within a single system.
Axis 1 Feedback	J16	Provides differential or single-ended motor encoder feedback signals such as <i>Quad A/B</i> , <i>Index</i> , <i>Hall A/B/C</i> , <i>PosLim</i> and <i>NegLim</i> for Axis 1.
Axis 2 Feedback	J17	Provides differential or single-ended motor encoder feedback signals such as <i>Quad A/B</i> , <i>Index</i> , <i>Hall A/B/C</i> , <i>PosLim</i> and <i>NegLim</i> for Axis 2.
Axis 3 Feedback	J18	Provides differential or single-ended motor encoder feedback signals such as <i>Quad A/B</i> , <i>Index</i> , <i>Hall A/B/C</i> , <i>PosLim</i> and <i>NegLim</i> for Axis 3.
Axis 4 Feedback	J19	Provides differential or single-ended motor encoder feedback signals such as <i>Quad A/B</i> , <i>Index</i> , <i>Hall A/B/C</i> , <i>PosLim</i> and <i>NegLim</i> for Axis 4.
+5V Power	J20	Provides logic power to the board when HVI is not available.
CAN1, CAN2	J21, J22	RJ45 connector that provide connection to a CAN 2.0B network.
Serial	J23	DB-9 serial port for RS232 or RS485 connections.
Ethernet	J24	RJ45 connector that provide connection to an Ethernet TCP/IP network

## 6.2.1 Atlas Connectors (J1-J4)

The Atlas Connectors (J1 – J4) are sockets that accept vertical Atlas amplifier units.

### 6.2.1.1 Axis 1 Atlas Socket

Pin	Connection	Description
<b>J1</b>		
1	GND	Power return for HV1, Motor A, Motor B, Motor C and Motor D.
2	GND	Power return for HV1, Motor A, Motor B, Motor C and Motor D.
3	HV1	DC power to axis 1 Atlas amplifier, referenced to GND.
4	HV1	DC power to axis 1 Atlas amplifier, referenced to GND.
5	Motor A1	Axis 1 motor output signal A.
6	Motor A1	Axis 1 motor output signal A.
7	Motor B1	Axis 1 motor output signal B.
8	Motor B1	Axis 1 motor output signal B.
9	Motor C1	Axis 1 motor output signal C.
10	Motor C1	Axis 1 motor output signal C.
11	Motor D1	Axis 1 motor output signal D.
12	Motor D1	Axis 1 motor output signal D.
13	~Enable1	An active-low enable signal.
14	FaultOut1	FaultOut1 provides programmable fault indication.
15	NC	Not Connected.
16	GND	Ground return for ~Enable, FaultOut, or SPI signals.
17	~SPICSI	Atlas SPI bus slave select 1.
18	SPISI	Atlas SPI bus master output, slave input.
19	SPICLK	Atlas SPI bus serial clock.
20	SPISO	Atlas SPI bus master input, slave output.

### 6.2.1.2 Axis 2 Atlas Socket

Pin	Connection	Description
<b>J2</b>		
1	GND	Power return for HV2, Motor A, Motor B, Motor C and Motor D.
2	GND	Power return for HV2, Motor A, Motor B, Motor C and Motor D.
3	HV2	DC power to Axis 2 Atlas amplifier, referenced to GND.
4	HV2	DC power to Axis 2 Atlas amplifier, referenced to GND.
5	Motor A2	Axis 2 motor output signal A.
6	Motor A2	Axis 2 motor output signal A.
7	Motor B2	Axis 2 motor output signal B.
8	Motor B2	Axis 2 motor output signal B.
9	Motor C2	Axis 2 motor output signal C.
10	Motor C2	Axis 2 motor output signal C.
11	Motor D2	Axis 2 motor output signal D.
12	Motor D2	Axis 2 motor output signal D.
13	~Enable2	An active-low enable signal.
14	FaultOut2	FaultOut2 provides programmable fault indication.
15	NC	Not Connected.
16	GND	Ground return for ~Enable, FaultOut, or SPI signals.
17	~SPICS2	Atlas SPI bus slave select 2.
18	SPISI	Atlas SPI bus master output, slave input.
19	SPICLK	Atlas SPI bus serial clock.

Pin	Connection	Description
20	SPISO	Atlas SPI bus master input, slave output.

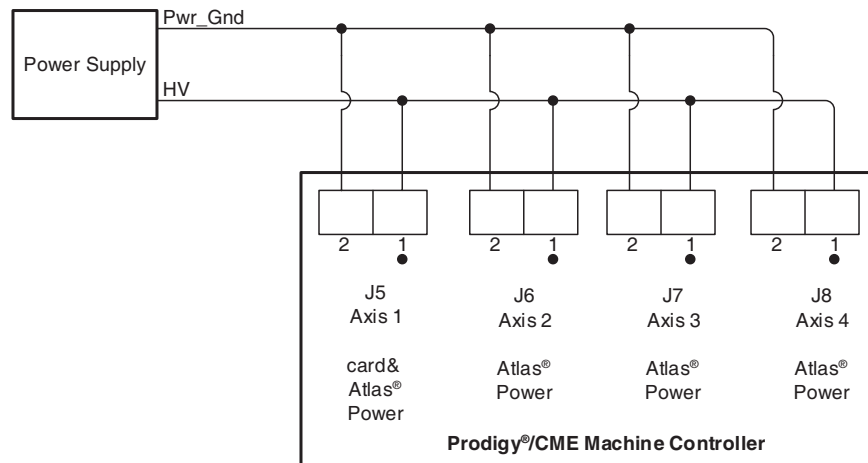
### 6.2.1.3 Axis 3 Atlas Socket

Pin	Connection	Description
<b>J3</b>		
1	GND	Power return for HV3, Motor A, Motor B, Motor C and Motor D.
2	GND	Power return for HV3, Motor A, Motor B, Motor C and Motor D.
3	HV3	DC power to the Axis 3 Atlas amplifier, referenced to GND.
4	HV3	DC power to the Axis 3 Atlas amplifier, referenced to GND.
5	Motor A3	Axis 3 motor output signal A.
6	Motor A3	Axis 3 motor output signal A.
7	Motor B3	Axis 3 motor output signal B.
8	Motor B3	Axis 3 motor output signal B.
9	Motor C3	Axis 3 motor output signal C.
10	Motor C3	Axis 3 motor output signal C.
11	Motor D3	Axis 3 motor output signal D.
12	Motor D3	Axis 3 motor output signal D.
13	~Enable3	An active-low enable signal.
14	FaultOut3	FaultOut3 provides programmable fault indication.
15	NC	Not Connected.
16	GND	Ground return for ~Enable, FaultOut, or SPI signals.
17	~SPICS3	Atlas SPI bus slave select3.
18	SPISI	Atlas SPI bus master output, slave input.
19	SPICLK	Atlas SPI bus serial clock.
20	SPISO	Atlas SPI bus master input, slave output.

### 6.2.1.4 Axis 4 Atlas Socket

Pin	Connection	Description
<b>J4</b>		
1	GND	Power return for HV4, Motor A, Motor B, Motor C and Motor D.
2	GND	Power return for HV4, Motor A, Motor B, Motor C and Motor D.
3	HV4	DC power to Axis 4 Atlas amplifier, referenced to GND.
4	HV4	DC power to Axis 4 Atlas amplifier, referenced to GND.
5	Motor A4	Axis 4 motor output signal A.
6	Motor A4	Axis 4 motor output signal A.
7	Motor B4	Axis 4 motor output signal B.
8	Motor B4	Axis 4 motor output signal B.
9	Motor C4	Axis 4 motor output signal C.
10	Motor C4	Axis 4 motor output signal C.
11	Motor D4	Axis 4 motor output signal D.
12	Motor D4	Axis 4 motor output signal D.
13	~Enable4	An active-low enable signal.
14	FaultOut4	FaultOut4 provides programmable fault indication.
15	NC	Not Connected.
16	GND	Ground return for ~Enable, FaultOut, or SPI signals.
17	~SPICS4	Atlas SPI bus slave select4.
18	SPISI	Atlas SPI bus master output, slave input.
19	SPICLK	Atlas SPI bus serial clock.
20	SPISO	Atlas SPI bus master input, slave output.

## 6.2.2 Power Connectors (J5-J8, J20)



**Figure 6-2:  
Power  
Connections**

The table below summarizes the motor power connections from the power supply to the Prodigy/CME Machine-Controller. Each motor driven by an Atlas amplifier must have separate power provided to it. Although most applications will power each axis at the same voltage from a single common supply, different voltages may be connected if desired. The input voltage range is +12 – 56VDC for high power Atlas units and +12 - 48VDC for low and medium power Atlas units.

When Atlas units are installed power should always be provided to the Axis 1 (J5) Power Connector. In addition to powering an Atlas unit, J5 is the power connection from which the board logic power is derived using onboard DC-DC converters.

If no Atlas units are installed there are two options for powering the board. The first is powering the board through J5 as indicated above. The voltage range in this circumstance is +12 – 56V. The second is powering the board logic directly via J20. See [Section 6.2.2.5, “+5V Power Connector,”](#) for details.

The J5-J8 Motor Power Connectors are male two- conductor Molex Mini-Fit Plus 2-style connectors. If you have ordered the developer kit version of the machine controller you may find it convenient to use the provided two-pin power stub cable sets.

### 6.2.2.1 Axis 1 Power Connector

Pin Connection Description		
J5		
1	HV1	Provides DC power to the board and Axis 1 Atlas amplifier
2	GND	Ground

### 6.2.2.2 Axis 2 Power Connector

Pin Connection Description		
J6		
1	HV2	Provides DC power to the Axis 2 Atlas amplifier
2	GND	Ground

### 6.2.2.3 Axis 3 Power Connector

Pin Connection Description		
J7		
1	HV3	Provides DC power to the Axis 3 Atlas amplifier
2	GND	Ground

### 6.2.2.4 Axis 4 Power Connector

Pin Connection Description		
J8		
1	HV4	Provides DC power to the Axis 4 Atlas amplifier
2	GND	Ground

### 6.2.2.5 +5V Power Connector

A direct +5V power connector (J20) is available which powers just the logic on the board. This can only be used when power to HV1 is not present (J5). J20 uses a Molex 3.00 Pitch Micro-Fit Header, 4-signal, single row, vertical connector. This +5V DC supply should have a minimum current capacity of 1.5A.

Providing +5V power at J20 will not power the Atlas amplifiers. There are two main uses of this direct +5V power connector. The first is to allow powering of the board when no Atlas units are installed. The second is to allow board diagnostics to be performed when Atlas units and motors are connected without powering the Atlas units or motors.

Pin Connection Description		
J20		
1	5V	+5V power
2*	5V	+5V power
3	GND	Ground
4*	GND	Ground

\* These connections are redundant and therefore optional

## 6.2.3 Feedback Connectors (J16-J19)

The Feedback Connectors (J16 – J19 in Figure 3-1) provide connections to various motor feedback signals using 15-pin high density DB connectors. If you have ordered the developer kit version of the machine controller you may find it convenient to use the included DB15 expander boards (PMD P/N: MCH-HW-05) to make these connections.

For Brushless DC motors, the Feedback Connector also connects the Hall effect signals typically used to commutate the motor. The Halls are not used with the DC Brush or step motors.

Vcc (+5V) output is provided in each of the Feedback Connectors. These outputs are typically used to power the encoder and Hall circuitry. The drive capacity of each 5V output is 100 mA.

### 6.2.3.1 Axis 1 Feedback Connector

Pin	Connection	Description
J16		
1	QuadA1+	Axis 1 Quadrature A+ encoder input
2	QuadA1-	Axis 1 Quadrature A- encoder input
3	QuadB1+	Axis 1 Quadrature B+ encoder input
4	QuadB1-	Axis 1 Quadrature B - encoder input

Pin	Connection	Description
5	GND	Ground
6	Index1+	Axis 1 Index+ input
7	Index1-	Axis 1 Index1- input
8	HallA1	Axis 1 Hall A input
9	HallB1	Axis 1 Hall B input
10	HallC1	Axis 1 Hall C input
11	Home1	Axis 1 Home input
12	PosLim1	Axis 1 Positive direction limit switch input
13	NegLim1	Axis 1 Negative direction limit switch input
14	Vcc	+5V
15	GND	Ground

### 6.2.3.2 Axis 2 Feedback Connector

Pin	Connection	Description
<b>J17</b>		
1	QuadA2+	Axis 2 Quadrature A+ encoder input
2	QuadA2-	Axis 2 Quadrature A- encoder input
3	QuadB2+	Axis 2 Quadrature B+ encoder input
4	QuadB2-	Axis 2 Quadrature B- encoder input
5	GND	Ground
6	Index2+	Axis 2 Index+ input
7	Index2-	Axis 2 Index- input
8	HallA2	Axis 2 Hall A input
9	HallB2	Axis 2 Hall B input
10	HallC2	Axis 2 Hall C input
11	Home2	Axis 2 Home input
12	PosLim2	Axis 2 Positive direction limit switch input
13	NegLim2	Axis 2 Negative direction limit switch input
14	Vcc	+5V
15	GND	Ground

### 6.2.3.3 Axis 3 Feedback Connector

Pin	Connection	Description
<b>J18</b>		
1	QuadA3+	Axis 3 Quadrature A+ encoder input
2	QuadA3-	Axis 3 Quadrature A- encoder input
3	QuadB3+	Axis 3 Quadrature B+ encoder input
4	QuadB3-	Axis 3 Quadrature B- encoder input
5	GND	Ground
6	Index3+	Axis 3 Index+ input
7	Index3-	Axis 3 Index- input
8	HallA3	Axis 3 Hall A input
9	HallB3	Axis 3 Hall B input
10	HallC3	Axis 3 Hall C input
11	Home3	Axis 3 Home input
12	PosLim3	Axis 3 Positive direction limit switch input



Pin	Connection	Description
13	NegLim1	Axis 3 Negative direction limit switch input
14	Vcc	+5V
15	GND	Ground

#### 6.2.3.4 Axis 4 Feedback Connector

Pin	Connection	Description
<b>J19</b>		
1	QuadA4+	Axis 4 Quadrature A+ encoder input
2	QuadA4-	Axis 4 Quadrature A- encoder input
3	QuadB4+	Axis 4 Quadrature B+ encoder input
4	QuadB4-	Axis 4 Quadrature B- encoder input
5	GND	Ground
6	Index4+	Axis 4 Index+ input
7	Index4-	Axis 4 Index- input
8	HallA4	Axis 4 Hall A input
9	HallB4	Axis 4 Hall B input
10	HallC4	Axis 4 Hall C input
11	Home4	Axis 4 Home input
12	PosLim4	Axis 4 Positive direction limit switch input
13	NegLim4	Axis 4 Negative direction limit switch input
14	Vcc	+5V
15	GND	Ground

### 6.2.4 Motor Drive Connectors (J9-J12)

The Motor Drive connectors (J9 – J12) provide motor output signals for use with Brushless DC, DC Brush, or step motors. These are Molex 4.20mm Pitch Mini-Fit Plus HCS series, 5-signal, single row, vertical connectors.

#### 6.2.4.1 Axis 1 Motor Connector

Pin	Connection	Description
<b>J9</b>		
1	Motor A1	Axis 1 motor output signal A.
2	Motor B1	Axis 1 motor output signal B.
3	Motor C1	Axis 1 motor output signal C.
4	Motor D1	Axis 1 motor output signal D.
5	Case/Shield	Ground

#### 6.2.4.2 Axis 2 Motor Connector

Pin	Connection	Description
<b>J10</b>		
1	Motor A2	Axis 2 motor output signal A.
2	Motor B2	Axis 2 motor output signal B.
3	Motor C2	Axis 2 motor output signal C.
4	Motor D2	Axis 2 motor output signal D.
5	Case/Shield	Ground

### 6.2.4.3 Axis 3 Motor Connector

Pin	Connection	Description
<b>J11</b>		
1	Motor A3	Axis 3 motor output signal A.
2	Motor B3	Axis 3 motor output signal B.
3	Motor C3	Axis 3 motor output signal C.
4	Motor D3	Axis 3 motor output signal D.
5	Case/Shield	Ground

### 6.2.4.4 Axis 4 Motor Connector

Pin	Connection	Description
<b>J12</b>		
1	Motor A4	Axis 4 motor output signal A.
2	Motor B4	Axis 4 motor output signal B.
3	Motor C4	Axis 4 motor output signal C.
4	Motor D4	Axis 4 motor output signal D.
5	Case/Shield	Ground

## 6.2.5 General I/O Connector (J13)

The General I/O connector provides various I/O connections to the board. There are 8 channels of bi-directional digital I/O; 4 channels of digital input; 4 channels of digital output and 8 channels of differential analog inputs. This is a high density DB-44 connector.

Pin	Connection	Description	Pin	Connection	Description
<b>J13</b>					
1	AnalogIn1+	Analog input 1+	2	AnalogIn1-	Analog input 1-
3	AnalogIn2+	Analog input 2+	4	AnalogIn2-	Analog input 2-
5	AGND	Ground for analog signals	6	AGND	Ground for analog signals
7	Reserved	Reserved signal	8	Reserved	Reserved signal
9	DigitalIO1	Digital input/output 1	10	DigitalIO2	Digital input/output 2
11	DigitalIO3	Digital input/output 3	12	GND	Ground
13	DigitalOut1	Digital output 1	14	DigitalOut2	Digital output 2
15	DigitalOut3	Digital output 3	16	AnalogIn3+	Analog input 3+
17	AnalogIn4+	Analog input 4+	18	AnalogIn5+	Analog input 5+
19	AnalogIn6+	Analog input 6+	20	AnalogIn7+	Analog input 7+
21	AnalogIn8+	Analog input 8+	22	AGND	Ground for analog signals
23	Reserved	Reserved signal	24	Reserved	Reserved signal
25	DigitalIO4	Digital input/output 4	26	DigitalIO5	Digital input/output 5
27	DigitalIO6	Digital input/output 6	28	DigitalIn1	Digital input 1
29	DigitalIn2	Digital input 2	30	DigitalOut4	Digital output 4
31	AnalogIn3-	Analog input 3-	32	AnalogIn4-	Analog input 4-
33	AnalogIn5-	Analog input 5-	34	AnalogIn6-	Analog input 6-
35	AnalogIn7-	Analog input 7-	36	AnalogIn8-	Analog input 8-
37	AGND	Ground for analog signals	38	GND	Ground
39	GND	Ground	40	DigitalIO7	Digital input/output 7
41	DigitalIO8	Digital input/output 8	42	GND	Ground
43	DigitalIn3	Digital input 3	44	DigitalIn4	Digital input 4

## 6.2.6 Amplifier I/O Connector (J14)

The Amplifier I/O connector provides various amplifier-related signals including amplifier enable outputs, 8 *AnalogOutput* signals, *AxisIn*, and *AxisOut* signals. This is a high density DB-26 connector.

Pin	Connection	Description	Pin	Connection	Description
<b>J14</b>					
1	AmpEnable1	Axis 1 amplifier enable signal	2	AmpEnable2	Axis 2 amplifier enable signal
3	GND	Ground	4	AxisIn1	Axis 1 AxisIn input
5	AxisIn2	Axis 2 AxisIn input	6	AGND	Ground for analog signals
7	AnalogOut1	Analog output 1	8	AnalogOut2	Analog output 2
9	AnalogOut3	Analog output 3	10	AmpEnable3	Axis 3 amplifier enable signal
11	AmpEnable4	Axis 4 amplifier enable signal	12	GND	Ground
13	AxisIn3	Axis 3 AxisIn input	14	AxisIn4	Axis 4 AxisIn input
15	AnalogOut4	Analog output 4	16	AnalogOut5	Analog output 5
17	AnalogOut6	Analog output 6	18	AGND	Ground for analog signals
19	AxisOut1	Axis 1 AxisOut output	20	AxisOut2	Axis 2 AxisOut output
21	AxisOut3	Axis 3 AxisOut output	22	AxisOut4	Axis 4 AxisOut output
23	GND	Ground	24	AGND	Ground for analog signals
25	AnalogOut7	Analog output 7	26	AnalogOut8	Analog output 8

### 6.2.6.1 External Amplifiers

The following tables show Amplifier I/O Connector connections to servo motors such as DC Brush and Brushless DC motors that will be driven by an external amplifier. The output is a single-ended +/-10V analog output intended to connect to a motor amplifier that accepts that command format.

Connection Name	Axis 1	Axis 2	Axis 3	Axis 4
AnalogOut1-4	J15-7	J14-8	J14-9	J14-15

## 6.2.7 Expansion Connector (J15)

This connector supports multi-board synchronization and an external RESET signal. Depending on whether the board is a master or slave, the *SyncOut* only or both *SyncIn* and *SyncOut* signals are used. This is a Molex 2.00 mini-pitch header, vertical, shrouded connector.

Pin	Connection	Description	Pin	Connection	Description
<b>J15</b>					
1	Reserved	Reserved signal	2	Reserved	Reserved signal
3	Reserved	Reserved signal	4	SyncIn	Multiple boards SyncIn signal
5	GND	Ground	6	+5V	Maximum 100mA +5V output
7	Reserved	Reserved signal	8	Reserved	Reserved signal
9	Reserved	Reserved signal	10	SyncOut	Multiple boards SyncOut signal
11	GND	Ground	12	Reset	Active Low hardware reset

## 6.2.8 Serial Connector (J23)

The Serial Connector (J23) provides connections to two serial ports in RS232 mode, or a single serial port in RS485 mode. Electrically these connectors provide access to the same signals; however they have different physical connectors and wiring. The following sections provide information for the serial connector, and provide pinouts when operated in RS232 mode, RS485 full-duplex mode, and RS485 half-duplex mode.

Pin	Connection	RS232	RS485 Full Duplex	RS485 Half Duplex
<b>J23</b>				
1	RS485Select	Open (default) selects RS232 mode for both Serial1 and Serial2	Tie to ground selects RS485 mode	Tie to ground selects RS485 mode
2	Sr1Xmt	Serial 1 transmit output	no connect	no connect
3	Sr1Rcv	Serial 1 receive input	no connect	no connect
4	No connect	no connect	no connect	no connect
5	GND	Ground	Ground	Ground
6	RS485Rcv <sup>+</sup>	no connect	Positive (non-inverting) receive input	no connect
7	Sr12Rcv/RS485Rcv <sup>-</sup>	Serial 2 receive input	Negative (inverting) receive input	no connect
8	Sr12Xmt/RS485Xmt <sup>+</sup>	Serial 2 transmit output	Positive (non-inverting) transmit output	Positive transmit/receive output
9	RS485Xmt <sup>-</sup>	no connect	Negative (inverting) transmit output	Negative transmit/receive output

*Note that pins 2 and 9 are connected to each other on the board as are pins 3 and 6.*

## 6.2.9 CAN Connectors (J21, J22)

The Prodigy/CME Machine-Controller's controller area network (CAN) transceivers are designed 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.

There are two different CAN connectors, J21, and J22, providing electrically identical signals. These two connectors are designed to make it easy to connect the Prodigy/CME Machine-Controller board in a daisy-chain configuration. 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. The CANbus connector is a female RJ45 type connector.

See [Section 4.2, "CAN Communications,"](#) for more information on the setting up and operating the CANbus port with Pro-Motion.

The pinouts for both the J21 and J22 CAN connector are as follows:

Pin Number	Signal	Description
<b>J21, J22</b>		
1	CAN+	Positive CAN signal connection
2	CAN-	Negative CAN signal connection
3	GND	Ground
4	No Connect	Pass-through signal
5	No Connect	Pass-through signal
6	No Connect	Pass-through signal
7	GND	Ground
8	No Connect	Pass-through signal

## 6.2.10 Ethernet Connector (J24)

The Prodigy/CME Machine-Controller's Ethernet transceivers are designed for use with 10/100 base-TX Ethernet and support both TCP and UDP protocols. See [Section 4.3, "Ethernet Communications,"](#) for more information setting up and operating the Ethernet port with Pro-Motion.

The Ethernet connector is an 8-pin female RJ45, and has two status LEDs, green and amber, which 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. Without Amber LED indicates that the network is at 10 Mbps.

The pinouts for the J24 Ethernet connector are as follows:

Pin Number	Signal	Description
<b>J24</b>		
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

## 6.2.11 Connector Parts Reference

The following table is supplied as a reference only.

Label	Description	Connector Part Number	Connector Mate
J1-J4	Atlas Connector	Samtec SSQ-110-01-F-D	Pluggable Atlas Unit
J5-J8	Power Connector	Molex 46015-0203	Molex 39-01-2025
J23	Serial Connector	TE Connectivity -5747150-4	DB-9 Male
J9-J12	Motor Drive Connector	Molex 39-30-2050	Molex 39-01-4051
J22, J28	CAN Connector	(H) Amphenol FRJAE-408 (V) EDAC A00-108-620-450	Male RJ45
J24	Ethernet Connector	(H) Amphenol FRJAE-408 (V) EDAC A00-108-620-450	Male RJ45
J15	Expansion Connector	Molex 87831-1220	Molex 79107-7005
J16-19	Feedback Connector	FCI 10090929-S154VLF	High density DB-15 Male
J13	General I/O Connector	FCI 10090929-S444VLF	High density DB-44 Male
J14	Amplifier I/O Connector	FCI 10090929-S264VLF	High density DB-26 Male

## 6.3 Motor Connections Quick Reference

The following tables provide a quick-reference for Motor Drive connections to different types of motors when the Atlas amplifiers are used.

### 6.3.1 Brushless DC Motor Connections

Connection Name	Axis 1	Axis 2	Axis 3	Axis 4
Motor A	J9-1	J10-1	J11-1	J12-1
Motor B	J9-2	J10-2	J11-2	J12-2
Motor C	J9-3	J10-3	J11-3	J12-3
Case/Shield	J9-5	J10-5	J11-5	J12-5

### 6.3.2 DC Brush Motor Connections

Connection Name	Axis 1	Axis 2	Axis 3	Axis 4
Motor+	J9-1	J10-1	J11-1	J12-1
Motor-	J9-2	J10-2	J11-2	J12-2
Case/Shield	J9-5	J10-5	J11-5	J12-5

### 6.3.3 Step Motor Connections

Connection Name	Axis 1	Axis 2	Axis 3	Axis 4
MotorA+	J9-1	J10-1	J11-1	J12-1
MotorA-	J9-2	J10-2	J11-2	J12-2
MotorB+	J9-3	J10-3	J11-3	J12-3
MotorB-	J9-4	J10-4	J11-4	J12-4
Case/Shield	J1-5	J10-5	J11-5	J12-5

## 6.4 Cables & Accessories

The following table provides a summary of the cables that are available with the Prodigy/CME Machine-Controller.

Component Part Number	Description
Cable-5001-01	2-signal HV Power supply cable. This stub cable provides power to the Atlas unit for each axis. This cable may plug into the Prodigy/CME Machine-Controller board's J5 – J8 connectors.
Cable-5002-01	5-signal Motor Drive cable. This stub cable connects to the Motor Drive Connectors.
Cable-RJ45-02	Ethernet connector. This cable connects to the board's Ethernet connector.
Cable-4355-01	Bifurcated serial cable that connects to J23 Serial port and provides two RS232 serial port connections.
Cable-USB-DB9	USB to DB9 serial cable
TRM-RJ45-02	120 ohm CANbus terminator
MC-HW-03	DB44 terminal screw expander board.
MC-HW-04	DB26 terminal screw expander board.
MC-HW-05	DB15 terminal screw expander board.

The following sections provide detailed information on selected cables from the table above.

## 6.4.1 Cable-5001-01

**PMD Part #:** Cable-5001-01

**Description:** 2-signal HV Power supply stub cable.

**Length:** 1.0 meter

J5-J8	Signal
1	HV
2	Ground

## 6.4.2 Cable-5002-01

**PMD Part #:** Cable-5002-01

**Description:** This stub cable connects to the Motor Drive Connectors.

**Length:** 1.0 meter

**Cable:** 5 conductor, 4.20mm pitch, 16AWG stranded cables

J9-J12 Pin	Signal	Wire Color
1	Motor A+	Red
2	Motor A-	White
3	Motor B+	Black
4	Motor B-	Brown
5	GND	Green

## 6.4.3 Cable-RJ45-02

**PMD Part #:** Cable-RJ45-02

**Description:** Male RJ-45 to male RJ-45 cable wired in a straight-through configuration

**Length:** 2.0 meters

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

### CANbus

#### J21, J22

Pin	Signal	Connects to
1	CAN+	1
2	CAN-	2
3	GND	3
4	pass thru	4
5	pass thru	5
6	pass thru	6
7	GND	7
8	pass thru	

### Ethernet

#### J24 Pin

Pin	Signal	Connects to
1	EthernetTx+	1
2	EthernetTx-	2
3	EthernetRx+	3
4	-	4
5	-	5
6	EthernetRX-	6
7	-	7
8	-	8

### 6.4.4 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, 26AWG, Untwisted, shielded, UL STYLE 2464

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

J21Pin	Signal	Connects to
1	RS485Select	-
2	SrlXmt	Port1-2
3	SrlRcv	Port1-3
4	No connect	-
5	GND	Port1-5, Port2-5
		-
7	Srl2Rcv/ RS485Rcv <sup>-</sup>	Port2-3
8	Srl2Xmt/ RS485Xmt <sup>+</sup>	Port2-2
9	RS485Xmt <sup>-</sup>	-

### 6.4.5 TRM-RJ45-02-R

**PMD Part #:** TRM-RJ45-02-R

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



# A. Installation

A

## In This Appendix

- ▶ Developer Kit Assembly

## A.1 Developer Kit Assembly

### A.1.1 Machine Controller Board Installation

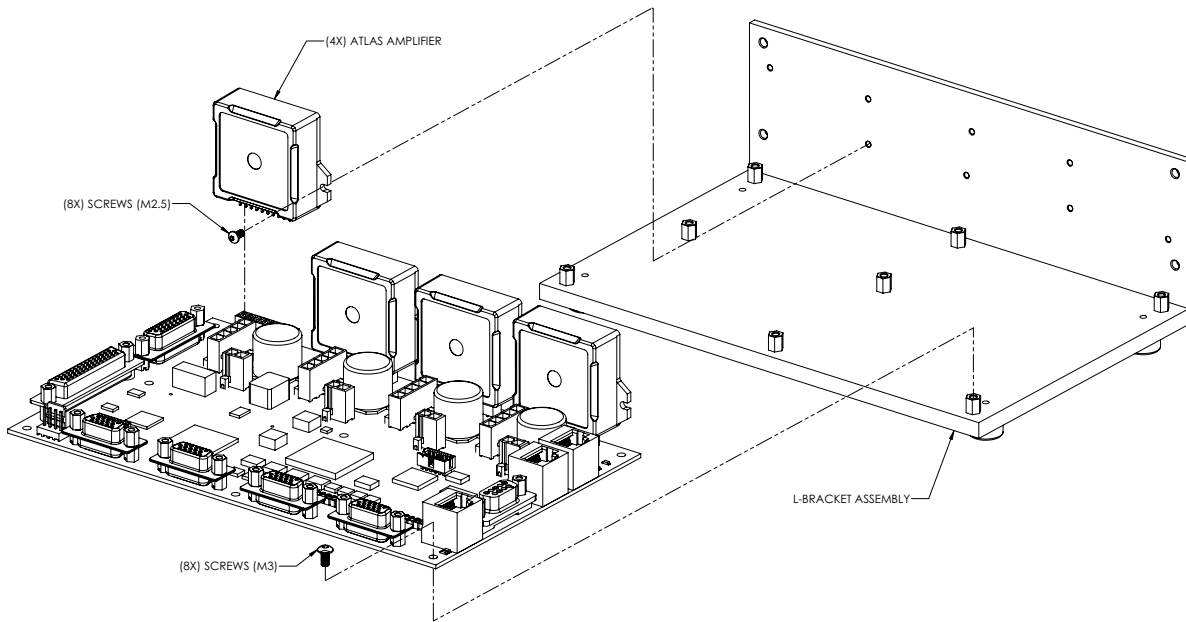


Figure A-1:  
Mounting  
Machine  
Controller  
Board onto  
L-Bracket Base  
Plate

If you are not using an L-bracket, or if the developer kit you ordered already has the machine controller board installed onto the L-bracket hardware, you may skip to [Section A.1.3, “Installing Atlas Units into the Board.”](#)

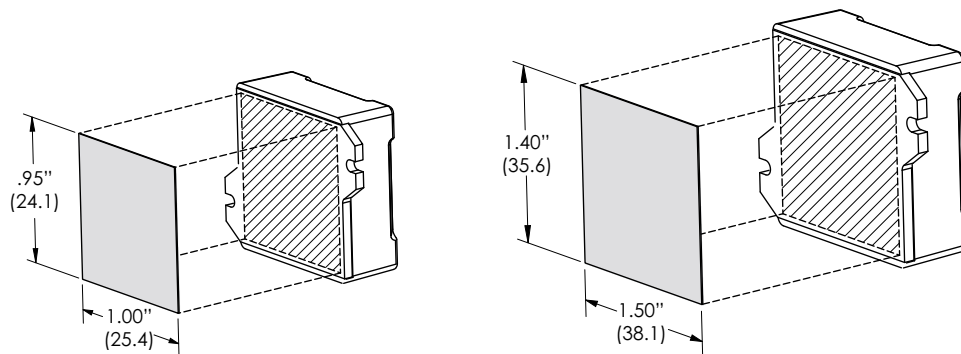
To install the machine controller board to the base plate use eight M3 screws at the base plate post location as shown in [Figure A-1](#). All necessary fasteners should be included with your developer kit hardware. Also included are the Allen keys which you will need for installation of the machine controller and Atlas amplifiers.



It is good practice to wear a grounding strap while handling both the machine controller board and the Atlas units. In addition it is recommended that assembly be undertaken on a surface that dissipates electrostatic charge.

## A.1.2 Atlas Thermal Pad Attachment

**Figure A-2:  
Thermal  
Transfer  
Material  
Attachment**

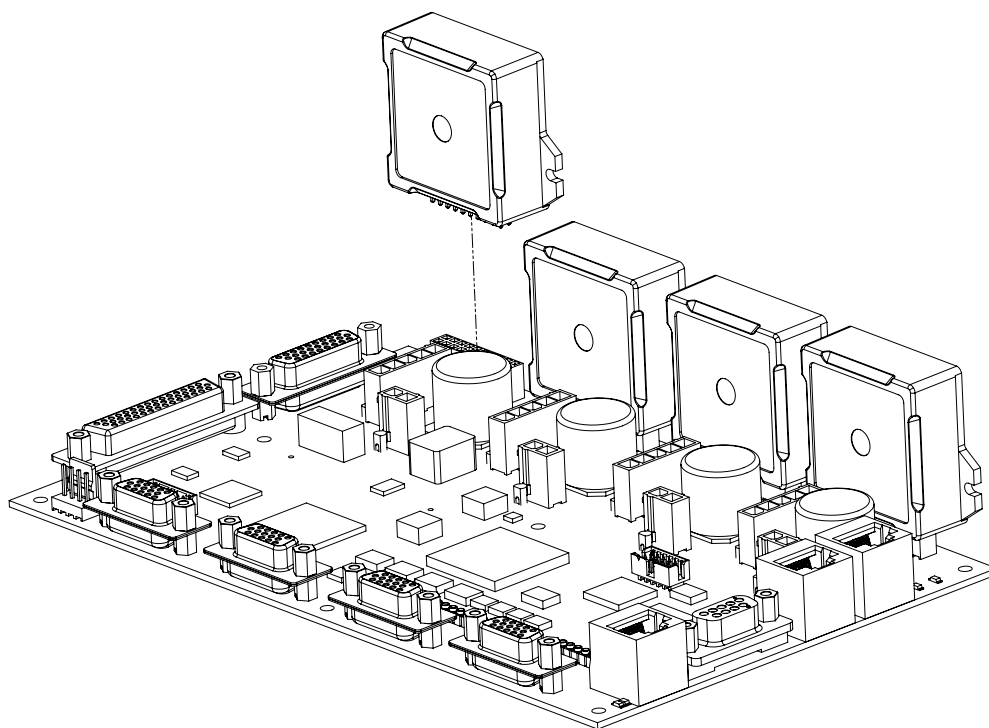


It is very important to have good thermal contact between the Atlas units and the L-bracket heat sink. For this purpose thermal pads will be attached to each Atlas to be installed.

First, locate the thermal pads of matching size. Two sets of thermal pads are included in your developer kit reflecting the two available Atlas package sizes - compact or ultra-compact. As shown in the figure above compact units use the larger thermal pad and ultra compact units use the smaller thermal pad.

Next, carefully remove the thin plastic protective sheets on either side of each pre-cut thermal pad and mount onto the Atlas unit, carefully aligning the pads with the Atlas' metallic backing, and applying finger pressure to adhere the pads to the metal. Note that the dimensions of each pad are not exactly square, so it is best to align the pads in the orientation shown in the diagram. Once pressed in place the pads should stay in place, but if required the pads can be removed and remounted.

## A.1.3 Installing Atlas Units into the Board



**Figure A-3:**  
Atlas  
Installation into  
Machine-  
Controller  
Board

If your machine controller board already has the Atlas amplifiers installed you may skip to [Section 1.3, “Software Installation.”](#)

To install Atlas units into the machine controller board sockets, confirm that the Atlas is oriented correctly, with the metal heat sink surface facing toward the vertical L-bracket plate. Carefully align the Atlas pins to the socket and press firmly down until the Atlas is fully seated in the socket.

If using Atlas units for specific motor types, the motor type of the Atlas should conform to the motor type that will be utilized for that axis. For example if your system has a DC brush motor at axis #1, and a step motor connected at axis #2, you should install a DC brush motor Atlas in the axis #1 socket, and a step motor Atlas in the axis #2 socket. In the figure above the axis #1 socket is the left-most socket and increase to the right. Alternatively by using the multi-motor version of Atlas, the motor type can be user programmed.

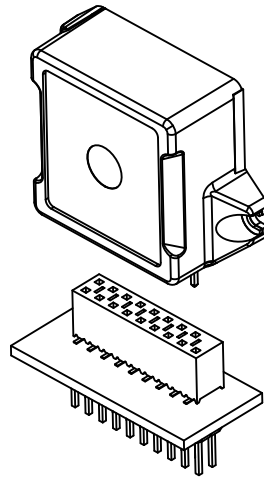
Extreme care should be taken when installing the Atlas into its socket. Failure to orient the Atlas correctly, or mis-alignment of pins may result in damage to the Prodigy/CME Machine-Controller board, Atlas units, or both.



### A.1.3.1 Ultra Compact Atlas Unit Installation

The above instructions provide details on installing compact Atlas vertical units, which are the larger of the two available Atlas sizes. The smaller ultra compact units, which are the package sizes for the low and medium power Atlas units, require the installation of a conversion card before installation into the machine controller board.

**Figure A-4:  
Compact to  
Ultra Compact  
Atlas Format  
Converter**



For any ultra compact Atlas units to be installed onto the machine controller board, first install the smaller thermal pads provided with your developer kit. Other than the size difference, installation of the pad to the Atlas unit is the same as for compact Atlas units.

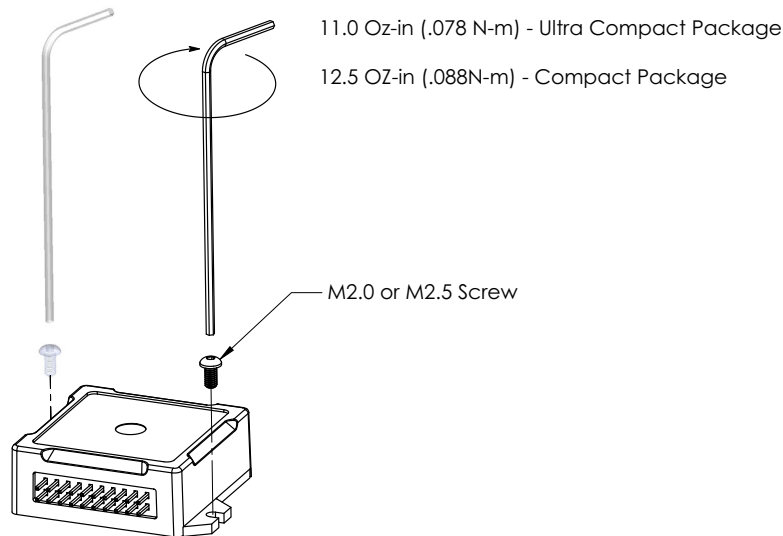
Next, each ultra-compact Atlas unit must be mated to a converter card as shown in the above figure. Before connecting the Atlas to the converter card, care should be taken to insure that they are oriented correctly, and that all pins align correctly without overhang.

Once the ultra-compact Atlas has been properly mated to the converter card, the converter/Atlas assembly can then be inserted into the machine controller board as described in [Section A.1.3, “Installing Atlas Units into the Board.”](#)

Note that the socket installation location of the compact and ultra compact Atlas units on the machine controller board is interchangeable. There is no restriction on the location of compact Atlas units versus the location of ultra compact Atlas units.

## A.1.4 Attaching Atlas Units to the Vertical Plate

**Figure A-5:  
Attaching Atlas  
Units to  
Vertical Plate**

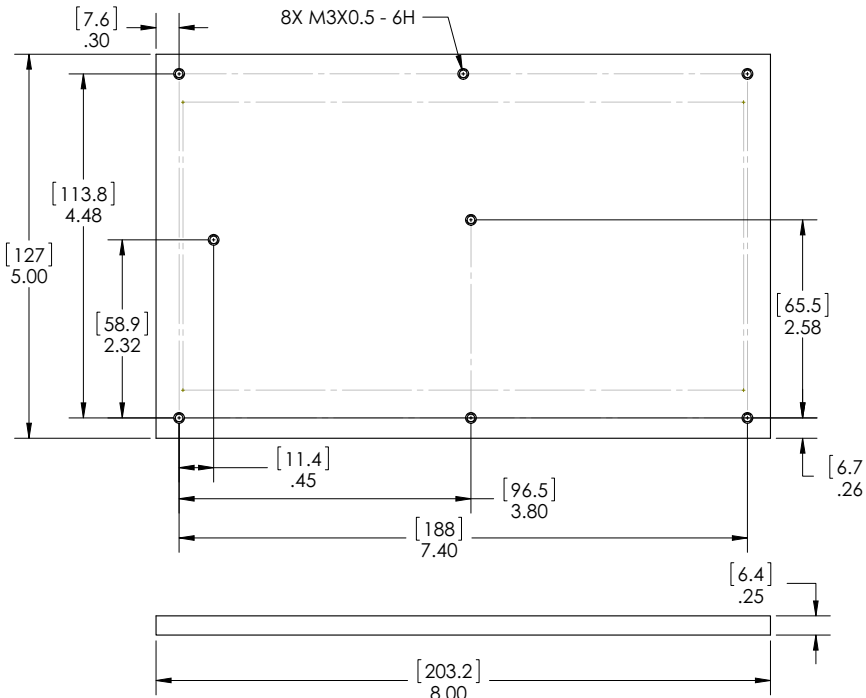


Finally, the Atlas units should be fastened to the vertical plate. Two screws are used to attach each Atlas and [Figure A-5](#) shows how the screws connect to the vertical plate. For compact Atlas units (high power) the M2.5 screws are used, and for ultra compact Atlas units (low and medium power) M2 screws are used.

Note that the mounting tap hole locations in the vertical plate are different for the compact and ultra compact Atlas units. Use only modest force in attached Atlas units to the vertical plate. [Figure A-5](#) shows this, also providing the torque limit specification for both Atlas types.

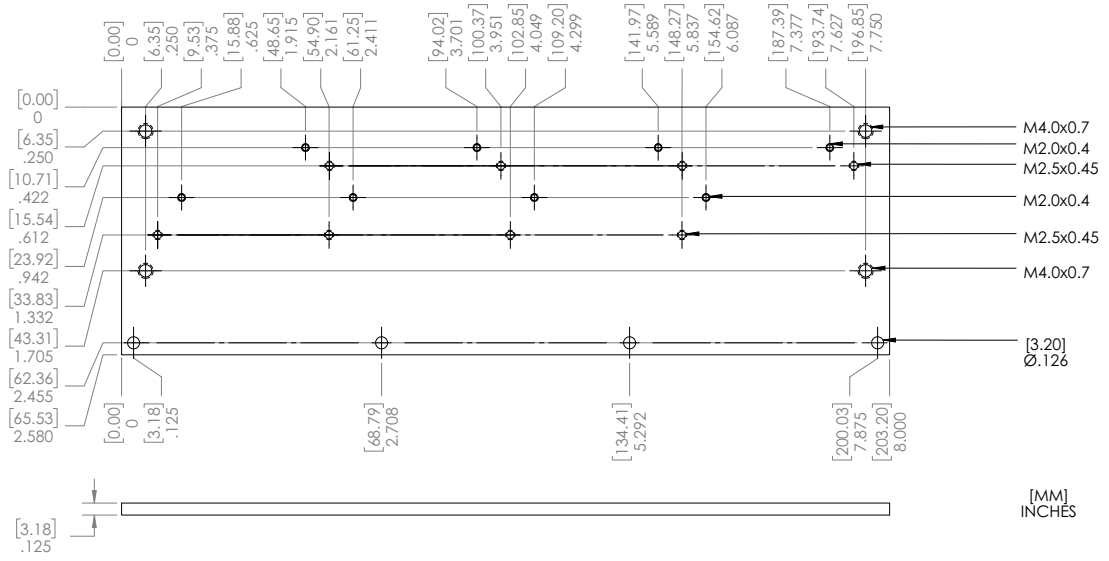
Congratulations! You have now completed mechanical assembly of the L-brackets to the machine controller board and Atlas units.

### A.1.5 L-Bracket Mechanical Dimensions



**Figure A-6:**  
L-Bracket Base  
Mechanical  
Dimensions

**Figure A-7:  
L-Bracket  
Vertical Plate  
Mechanical  
Dimensions**



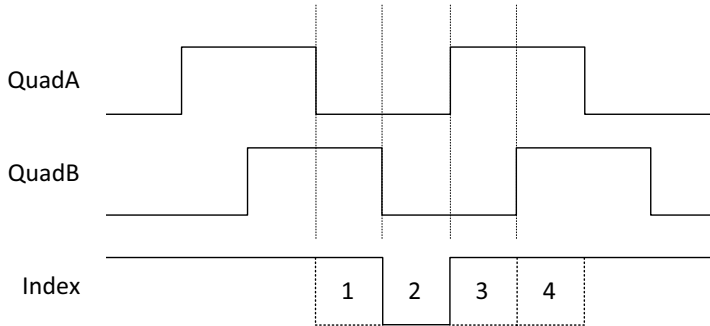
# B. Index Capture Qualification

B

Many PMD controllers qualify the Index signal with the *QuadA* and *QuadB* signals, requiring that all three be in a low state for a capture to occur.

Depending on how the encoder outputs these signals you may therefore need to invert the *Index* signal so that it provides an active low signal, meaning when the encoder is in the index it outputs a low signal, and when it is not in the index it outputs a high signal. Even if the *Index* signal sense is correctly set, one or more of the quadrature signals may also need to be inverted to satisfy the condition of all three signals being low for capture to occur.

For an index pulse that is correctly input as an active low signal the figure below shows the four possible combinations of the relationship between *QuadA*, *QuadB*, and *Index*.



**Figure B-1:**  
QuadA, QuadB,  
and Index  
Signal  
Qualification  
Combinations

The table below shows actions that can be taken to satisfy the Index qualification requirement without changing the encoder's direction interpretation.

Index State #	Action
1	Swap QuadA and QuadB signals* Invert** QuadA signal input
2	No action, correct as is
3	Swap QuadA and QuadB signals Invert QuadB signal input
4	Invert QuadA and QuadB

\*\* Swap QuadA and QuadB signals means rewire the quadrature signals so that channel A signals from the encoder are input into channel B of the N-Series ION, and channel B signals are input into channel A.

\* N-Series ION supports electronic signal inversion so inverting signal inputs is easiest to accomplish via Pro-Motion. Nevertheless if differentially encoded these signals inputs may also be inverted by swapping the + and - differential input wires.

To determine what condition the encoder signals are in the best approach is to command the motor to rotate and view these signals on an oscilloscope, capturing a screen image when the index goes low. This has the additional benefit of confirming that the *Index* signal is functioning and wired into the PMD controller correctly.

Note that it is not recommended to use Pro-Motion's scope trace feature to try to view the *Index* signal because the index pulse duration can be well below the trace sampling interval. A general purpose laboratory oscilloscope with the trigger set on the index pulse is the recommended approach to view the *Index*, *QuadA* and *QuadB* signals.

Alternatively, if an oscilloscope is not available, you can experiment with signal state inversions and re-wiring until capture occurs correctly.



# Index

---

## A

Atlas amplifier  
connectors 83, 84  
electrical installation 98  
sockets 84–85

## C

card  
types 9  
connections  
communications 18  
motor drive 17  
motor feedback 16  
motor power 18, 86  
connectors 83  
amplifier I/O 91  
CAN 92  
Ethernet 93  
expansion 91  
feedback 87  
general I/O 90  
motor drive 89  
parts reference 93  
serial 92

## E

encoder  
settings 81

## H

hardware accessory products 11

## L

L-bracket  
card assembly 97  
installation 97  
mechanical dimensions 101

## P

pinouts 92  
CAN connectors 92  
Ethernet connector 93  
power

applying 19  
connections 18  
PRP  
resources 72

## R

resistor packs 81  
settings 81  
resource  
addressing 72  
resources, PRP 72

## U

user-settable components 15, 81



---

*This page intentionally left blank.*