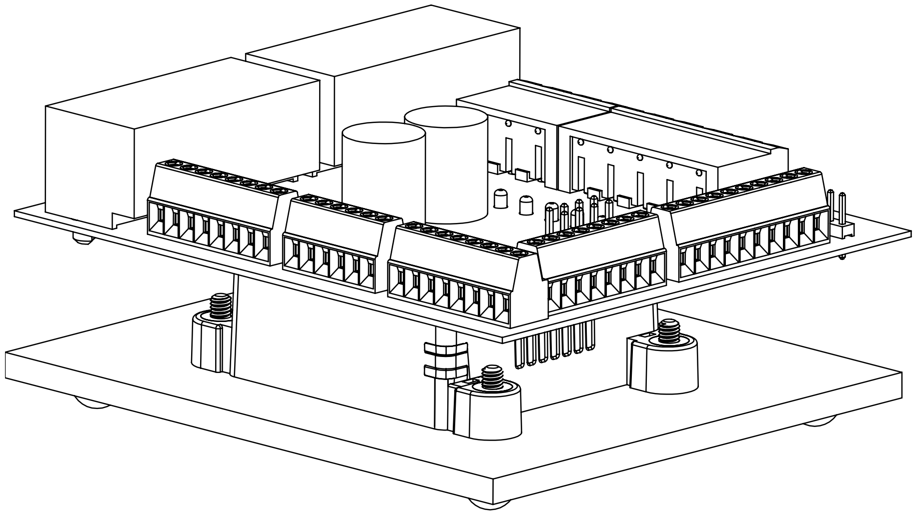


ION®/CME N-Series Digital Drive Developer Kit

User Manual

Revision 1.1/ July 2025

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

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

ION/CME N-Series Digital Drive User Manual

Complete description of the N-Series ION family of drives including complete mechanical, electrical, and software specifications.

Magellan Motion Control IC User Guide

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

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® Engine Development Tools Manual

Complete guide to developing software for PMD controller products. Includes examples on how to use the C-Motion Magellan, C-Motion PRP, and C-Motion PRP II C-language libraries, and details the recommended code development sequence for microcontroller-based, PC-based , and C-Motion Engine-based systems.

C-Motion PRP II Programming Reference

Describes C-Motion language function calls and associated PRP-formatted packets along with data types for the ION/CME N-Series ION Digital Drives.

Table of Contents

List of Figures

This page intentionally left blank.

1. Introduction

In This Chapter

1.1 ION/CME N-Series Digital Drive Overview

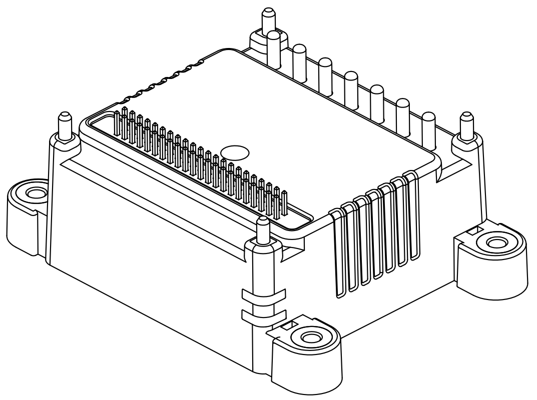


Figure 1-1: N-Series ION Drive

This manual provides a complete user guide for the ION/CME N-Series Digital Drive Developer Kit.

ION/CME N-Series Digital Drives are single-axis motion controllers with integrated power electronics and network communications. Various models are available to drive DC Brush, Brushless DC, and step motors. Their very compact size, range of power output levels, and high level of connectivity make them an ideal solution for embedded or distributed motion control applications that require high performance in a small envelope.

ION N-Series Drives are based on PMD’s Magellan Motion Control IC and perform profile generation, encoder position feedback, position servo compensation, step motor stall detection, brushless DC motor commutation, microstep generation, digital current/torque control, and more. All members of the ION family have integrated, high-power drive stages which protect from overcurrent, undervoltage, overvoltage, overtemperature, and short-circuit faults.

N-Series ION host communication options include Ethernet, CAN, RS232/RS485, and SPI (Serial Peripheral Interface). Each drive also supports an additional CAN and SPI expansion network for connecting to other N-Series ION or to other peripherals. All ION/CME N-Series Drives include a C-Motion Engine, allowing user application code to be downloaded and executed directly in the drive, along with NVRAM memory and trace memory for permanent and temporary storage of control parameters and performance trace results.

The N-Series IONs are PCB (printed circuit board) mounted and are packaged in a plastic and metal solderable module measuring 1.48" x 1.48" x 0.66" (37.6 mm x 37.6 mm x 16.8 mm). They come in three power levels; 75 watts, 350 watts, and 1,000 watts and utilize a 44 pin (2 x 22) 50-mil header for signal connections and a 7-pin high current connector for the DC bus and motor connections.

There are 36 different ION/CME N-Series Digital Drives in all, consisting of the combinations of four motor types (step motor, Brushless DC, DC Brush, Multi-Motor), three host interfaces (Serial, CAN/SPI, Ethernet), and three power levels (low, medium, high). Note that multi-motor units allow the motor type, either Brushless DC, DC Brush, or step motor, to be user programmed. For a complete list of N-Series ION part numbers refer to Section 6.3, "N-Series ION Unit Part Numbers and Configurations."

1.2 Developer Kit Overview

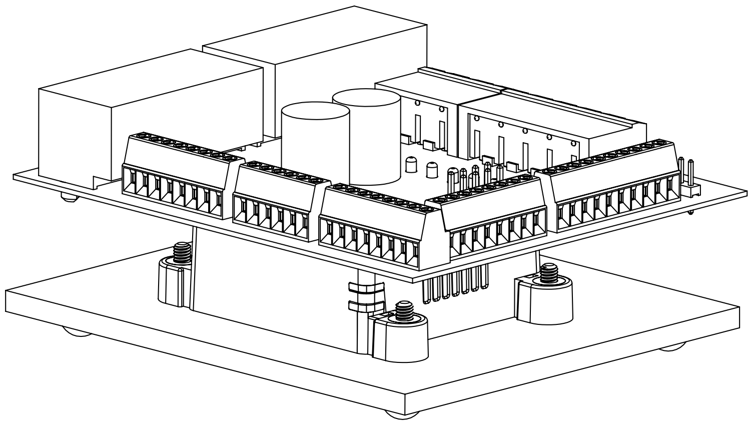


Figure 1-2: N-Series ION Developer Kit

The ION/CME N-Series Digital Drive Developer Kit is an integrated board/software package that serves as a mechanical, electrical, and software design tool for prototyping and building systems with N-Series ION drives.

The core of the developer kit is an interconnect board that provides convenient connections to your PC, the motor hardware, and other peripherals that will be used in your application.

As shown in Figure 1-2 the N-Series ION is mounted in between a metallic heat sink base (which also serves the purpose of providing a footing for bench-top use) and the DK interconnect board which provides the connectors. Alternatively you can remove the metallic heat sink base and mount the N-Series ION with attached DK interconnect board directly on your machine hardware. For exact dimensions of the N-Series ION refer to Section 6.11, "Physical Dimensions."

1.3 Developer Kit Part Numbers

1.3.1 Pre-Assembled Developer Kits

Nine fully assembled developer kit variations support all members of the N-Series ION drive family, representing three different host interface types (serial, CAN/SPI, Ethernet) and three different power levels (low, medium, high). These N-Series ION developer kits are installed with a multi-motor ION unit type.

DK P/N	ION P/N Installed	Host Interface	Power Level
DK481S0056/02	DD481S0056/02	Serial	Low
DK481S0056/06	DD481S0056/06	Serial	Medium
DK481S0056/18	DD481S0056/18	Serial	High
DK481C0056/02	DD481C0056/02	CAN/SPI	Low
DK481C0056/06	DD481C0056/06	CAN/SPI	Medium
DK481C0056/18	DD481C0056/18	CAN/SPI	High
DK481D0056/02	DD481D0056/02	Ethernet	Low
DK481D0056/06	DD481D0056/06	Ethernet	Medium
DK481D0056/18	DD481D0056/18	Ethernet	High

1.3.2 Component Developer Kits

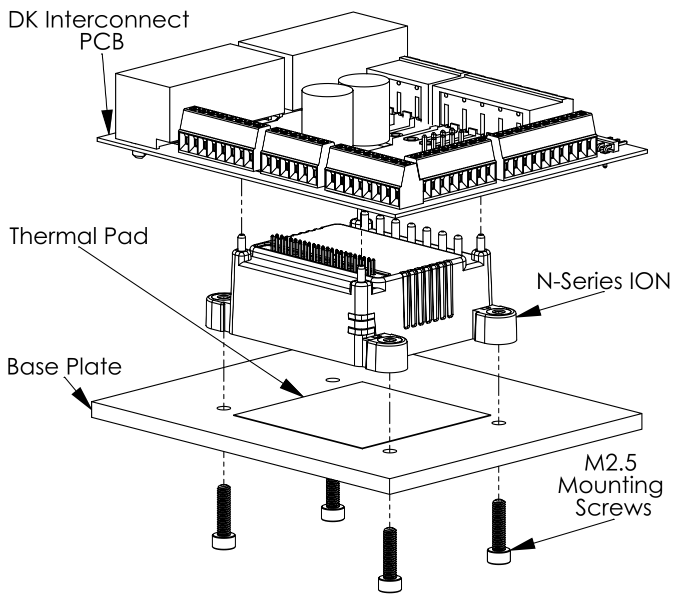


Figure 1-3: N-Series ION DK Showing Component Stack Elements

There may be occasions where it is preferable for the user to assemble an N-Series ION developer kit themselves. The main advantage of this is that it allows developer kit setups to be created with N-Series IONs other than the multi-motor type.

To facilitate this three non pre-assembled component developer kits are available as shown in the table below:

P/N	Host Interface	Comments
DK4X1S	Serial	Supports all serial host interface N-Series ION units
DK4X1C	CAN/SPI	Supports all CAN/SPI host interface N-Series ION units
DK4X1D	Ethernet	Supports all Ethernet host interface N-Series ION units

Each of these component-only DKs contain a DK interconnect PCB, a metallic base plate, and various mounting hardware such as screws and a thermal pad to allow the user to assemble a complete DK setup. Note that these DK P/Ns do not include the N-Series ION unit itself. The N-Series ION must be ordered separately.

For instructions on how to assemble a developer kit setup from a component-only DK refer to Section 6.10, "Component Developer Kit Assembly."

1.4 Developer Kit Components List

An assembled ION/CME N-Series Digital Drive Developer Kit contains the following components:

- Developer Kit interconnect board with N-Series Drive soldered in place
- Metal heat sink base with rubber feet
- USB to 3-pin serial programming cable
- Additional cables or accessories, depending on the host interface type of the ION unit

The following software and design materials are also included with each N-Series ION Developer Kit:

- Pro-Motion Windows-based exerciser
- C-Motion Software Development Kit:
    - A complete toolset for the creation of user-specific applications running on ION/CME or host

- An open-source compiler
- C-Motion libraries

For a detailed list of components for each N-Series ION DK type refer to Section 6.4, "Developer Kit Hardware Contents."

1.5 Guide to this Manual

This manual is designed to help you get your motion hardware setup connected and operating with an N-Series ION Developer Kit. 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 N-Series ION DK.

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 ICs.

Chapter 4, *Communication Port Connections* provides an overview of how to develop software application code for your PMD-based control system.

Chapter 5, *Software Development* provides reference information that you may find useful in connecting the machine controller board to your motion hardware.

Chapter 6, *Reference* provides miscellaneous information on the N-Series ION units and N-Series ION Developer Kits including operational specifications and mechanical dimensions. This chapter also provides a complete electrical reference for the N-Series ION developer kit interconnect boards.

Chapter 7, *DK Interconnect Board Schematics* provides reference schematics for all three N-Series ION DK interconnect PCBs.

1.6 Software Installation

The software distribution for the developer kit 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 "N-Series ION 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 N-Series ION drive:
   - **C-Motion PRP II** - an SDK (Software Development Kit) for creating PC and downloadable user code for N-Series ION drive products. Also supports creating motion applications using the .NET (C#, VB) programming languages for all products.

Here is more information on each of these items.

1.6.1 Pro-Motion

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

- Motion oscilloscope graphically displays processor 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

For more information on Pro-Motion see Chapter 3, *Going Further with Pro-Motion*.

1.6.2 C-Motion

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

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

For more information on C-Motion see Chapter 5, *Software Development*.

1.6.3 .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:

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

1.7 Recommended Hardware

To install an N-Series ION developer kit the following hardware is recommended.

- PC with Intel (or compatible) processor, 1 Gbyte of available disk space and 256 MB of available RAM. The supported PC operating systems is Windows 10.
- One stepper, DC Brush, or Brushless DC motor. This motor may or may not provide encoder position feedback signals, depending on the type of motor being used. Encoder feedback is normally used with DC Brush motors and with Brushless DC motors (although not required because Hall sensors can be used for the position feedback). For step motors, encoders are an option.
- Cables as required to connect to the motor and associated motion hardware such as feedback signals.

- DC Power supply. The N-Series ION drive requires only a single voltage supply. Its internal logic and other circuitry is powered from this input voltage using an internal DC to DC converter. The input voltage range is 12-56V.

2. Quick Start Guide

In This Chapter

Here are the steps to set up the N-Series ION developer kit so that it can control your motion hardware successfully. If you haven't already installed the software you should do this now. See Section 1.6, "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 the system's encoder, motion peripherals, motor, power supply, and PC to the DK board. See Section 2.2, "Step #2 — Making Motion Hardware Connections," for details.

Step 3 Next provide the N-Series ION DK with power. See Section 2.3, "Step #3 — Applying Power," for more information.

Step 4 The final step to 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 N-Series ION and connected motion hardware are ready for operation.

2.1 Step #1 — Configuring the Board

Figure 2-1 shows the location of various components for the three different DK interconnect boards which differ in the host interface type; serial, CAN/SPI, and Ethernet.

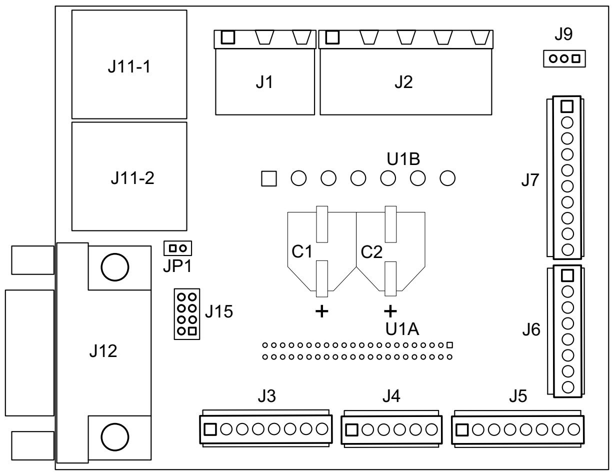


Figure 2-1: Serial Host Interface DK Interconnect Board

Figure 2-2: CAN/SPI Host Interface DK Interconnect Board

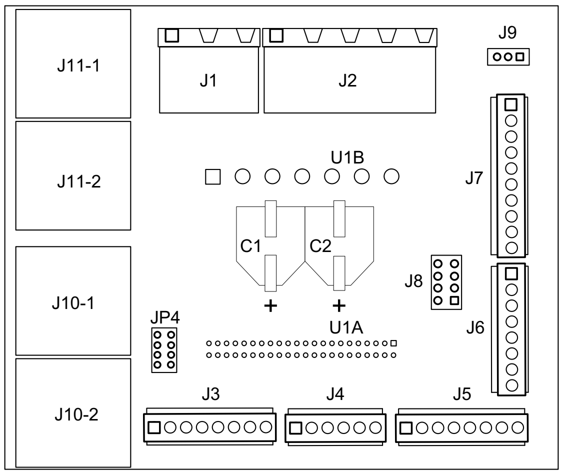
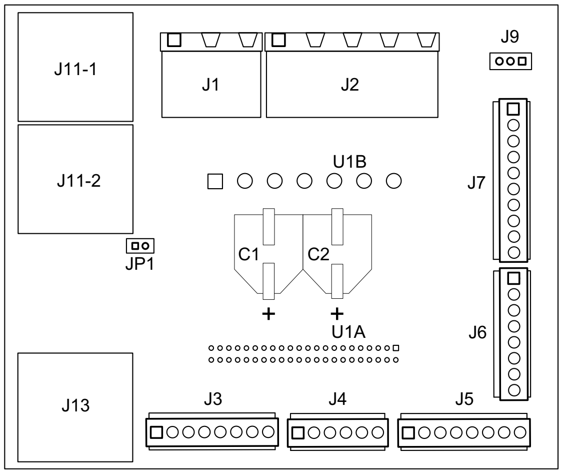


Figure 2-3: Ethernet Host Interface DK Interconnect Board



The following table identifies these components:

Label	Description
J1	HV Power Connector
J2	Motor Drive Connector
J3	Feedback Connector
J4	Hall Signals Connector
J5	Auxiliary Connector
J6	Motion Signals Connector
J7	Indexer Connector
J8	Host SPI Connector (CAN/SPI DK version only)
J9	Programming Connector
J10	Host CAN Connector (CAN/SPI DK version only)
J11	Expansion CAN Connector
J12	Host Serial Connector (Serial DK version only)
J13	Host Ethernet Connector (Ethernet DK version only)
J15	Host Serial Header Connector (Serial DK version only)
JP1	Serial & Ethernet board jumper
JP4	CAN/SPI board jumpers
U1A	N-Series ION Signal Connector pins
U1B	N-Series ION Power Connector pins

Label	Description
C1, C2	HV Capacitors

For complete details on all of these connectors refer to Section 6.1, "Connector Reference."

2.1.1 Jumper Settings

There are no jumper changes that need to be made for the N-Series ION DK boards to be compatible with this quick start chapter. However, for reference the table below shows the available jumper settings of the N-Series ION DK boards:

Here is the jumper setting for the Serial and Ethernet DK boards, which are shown in Figure 2-1 and Figure 2-3 respectively:

Jumper	Label	Factory Default	Description
JP1	N/A	Installed	When installed connects a 120 ohm termination resistor at the Expansion CAN bus. The terminating resistor should be used if the DK board is located at the end of the CAN network bus.

Here are the jumper settings for the SPI/CAN DK board, which is shown in Figure 2-2:

Jumper	Label	Factory Default	Description
JP4-1	E-Term	Installed	When installed connects a 120 ohm termination resistor at the Expansion CAN bus. The terminating resistor should be used if the DK board is located at the end of the CAN network bus.
JP4-2	H-Term	Installed	When installed connects a 120 ohm termination resistor at the Host CAN bus. This terminating resistor should be used if the DK board is located at the end of the CAN network bus.
JP4-3	SynchIn	Not installed	When installed connects pin 4 of the J10-2 Host CAN connector to the N-Series ION's SynchIn signal which is pin 34 of the Signal Connector. This connection allows daisy-chain ID assignment to be used described in the *ION/CME N-Series Digital Drive User Manual*.
JP4-4	SynchOut	Not installed	When installed connects pin 4 of the J10-1 Host CAN connector to the N-Series ION's SynchOut signal which is pin 37 of the Signal Connector. This connection allows daisy-chain ID assignment to be used described in the *ION/CME N-Series Digital Drive User Manual.*

2.1.2 Enable Signal

N-Series IONs require an active *Enable* signal to operate. To accomplish this the Motion Signals Connector (J6) is used. Connect terminal #4 of J6 (indicated on the board as En) to terminal #8 of the same terminal screw connection (indicated on the board as GND) using a short wire.

The following table details the Motion Signals Connector (J6). This connector is an 8-pin terminal screw connection.

Pin #	Signal Name	Description
		J6 — Motion Signal Connector
1	PosLim	Positive position limit input (optional)
2	Neglim	Negative position limit input (optional)
3	Home	Home signal input (optional)
4	Enable	Enable input signal
5	FaultOut	FaultOut signal output

Pin #	Signal Name	Description
		J6 — Motion Signal Connector
6	Reset	Reset signal input (optional)
7	Brake	Brake signal input (optional)
8	GND	Digital ground

2.2 Step #2 — Making Motion Hardware Connections

The next few sections detail how to make the needed connections between the N-Series ION drive DK boards and your motion hardware, PC, and power supply.

2.2.1 Encoder & Motion Peripheral Connections

The following tables summarize encoder and motion peripheral signal connections to the N-Series ION DK boards. All connections are made through either the Feedback Connector (J3), the Motion Signals Connector (J6), or the Hall Signals Connector (J4). Each of these connectors uses terminal screws for easy wire connection.

Encoders are used for 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 (QuadA, QuadB, 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, QuadB, 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.

The following table shows the Feedback Connector (J3) connections.

Pin #	Signal Name	Description
		J3 — Feedback Connector
1	+5V	+5V power output which may be used to power the motor's encoder circuitry
2	GND	This is the preferred ground connection for the quadrature and Index signal inputs
3	QuadA1+	Differential A+ encoder input. *Optional for step motors.*
4	QuadA1-	Differential A- encoder input. *Optional for step motors.*
5	QuadB1+	Differential B+ encoder input. *Optional for step motors.*
6	QuadB1-	Differential B- encoder input. *Optional for step motors.*
7	Index1+	Differential Index+ encoder connection. *Optional for step motors.*
8	Index1-	Differential Index- encoder connection. *Optional for step motors.*

The following table details the Motion Signals Connector (J6) connections.

Pin #	Signal Name	Description
		J6 — Motion Signals Connector
1	PosLim	Positive position limit input *(optional)*
2	Neglim	Negative position limit input *(optional)*
3	Home	Home signal input *(optional)*
8	GND	Digital Ground

The following table shows the Hall Signals Connector (J4) connections.

Pin #	Signal Name	Description
		J4 — Hall Signals Connector
1	HallA	Hall Sensor A Input
2	HallB	Hall Sensor B Input
3	HallC	Hall Sensor C Input
4	GND	Ground

2.2.2 Motor Coil Connections

The following table summarizes the motor drive connections from the N-Series ION DK board to the coils (also called windings) of the motor. The Motor Drive Connector, J2, provides these connections and is designed to connect to all supported motor types: Brushless DC, DC Brush, and step motor. There are four motor drive connections and a shield connection. Not every motor type uses all four drive connections however.

All connections are made via J2, the Motor Drive Connector, which is a Würth Elektronik 5 Position Terminal Block Header P/N 691313710005.

Pin #	Signal Name	Description
		J2 — Motor Drive Connector
1	Motor A	A motor drive lead. Used with all motor types.
2	Motor B	B motor drive lead. Used with all motor types
3	Motor C	C motor drive lead. Used with all motor types except DC Brush
4	Motor D/Shunt	D motor drive lead used with step motors only, or shunt output used with DC Brush or Brushless DC motors.
5	Case/shield	Connection to motor case/shield. A shield connection is strongly recommended for most motor setups.

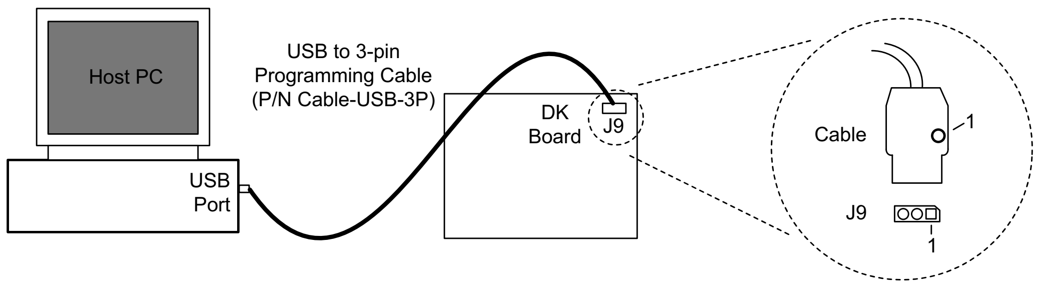
This table shows which leads should be connected for each supported motor type:

Motor Type	DK Board Pin # and Name	Motor Coil Connections
Brushless DC	1, Motor A 2, Motor B 3, Motor C 5, Case/shield	A winding connection B winding connection C winding connection (optional) motor shield connection
DC Brush	1, Motor A 2, Motor B 5, Case/shield	+ winding connection - winding connection (optional) motor shield connection
Step motor	1, Motor A 2, Motor B 3, Motor C 4, Motor D 5, Case/shield	phase A+ winding connection phase A- winding connection phase B+ winding connection phase B- winding connection (optional) shield connection

Shield connections to the motor are not required but are recommended. Not connecting the shield signal may result in increased EMI (electromagnetic interference), reduced immunity to ESD (electrostatic discharge), or electrical noise resulting in motor operation failure.

2.2.3 Communication Connections

Figure 2-4: PC to N-Series ION DK Board Connection



For this quick start installation we will use the N-Series ION's 3-pin serial programming interface for communicating to the PC. For this purpose, a 3-pin programming cable is included with the developer kit. This serial cable (PMD p/n Cable-USB-3P) should be connected to the board's J9 Programming Connector, while the opposite end of the cable is connected to one of the computer's USB ports. Take special care when connecting to the 3-pin connector on the board that pin #1 on the cable (marked with a dot on the cable) aligns with pin #1 on the board connector (also marked with a dot).

If operating the developer kit in high EMI environments or at high motor current levels, shielding on the 3-pin programming cable may be helpful to insure reliable communication between the PC and the ION unit. Contact your PMD representative for additional information and support.

2.2.4 Power Connections

The following table summarizes the power connection from the DC power supply to the N-Series ION DK board at the HV Power Connector (J1). The HV voltage is the voltage at which the motor is driven and must be in the range of 12V - 56V.

The HV Power Connector is a Würth Elektronik 3 Position Terminal Block Header P/N 691313710003.

Pin #	Signal Name	Description
		J1 — HV Power Connector
1	HV Aux	Positive motor voltage power used to drive N-Series ION's internal logic
2	HV	Positive motor voltage power used to drive the motor
3	GND	Motor voltage power ground

The HV Aux and HV pins are normally tied together but may be kept separate.

2.3 Step #3 — Applying Power

Once the motion hardware, motor coil, communication, and power connections have been made hardware installation is complete and the N-Series ION is ready for operation. When power is applied, the N-Series ION's green power LED should light. This LED along with an additional red LED indicate the status of the N-Series ION unit. The diagram below shows the location of these LEDs.

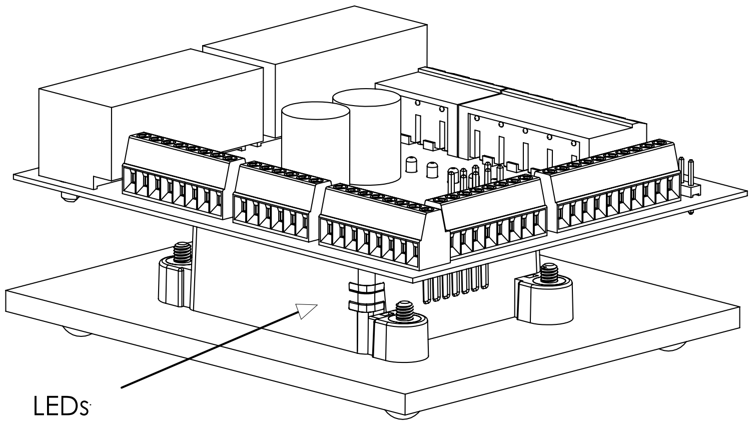


Figure 2-5: N-Series ION LEDs Location

A solid green LED indicates a normal startup. Other startup conditions may result in a blinking LED or a red LED. If this is the case refer to Section 6.7, "LEDs," which lists the N-Series ION LED output states. When ready, re-power and re-check the LED output.

After power up no motor output will be applied. Therefore the motors should remain stationary. If the motors move or jump, power down and check the motor and encoder connections. If anomalous behavior is still observed call PMD or your PMD representative 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 N-Series ION with your PC so that they are communicating properly, and the second is to initialize the axis and bring it under stable control capable of making controlled moves. While there are many additional capabilities that Pro-Motion and N-Series IONs provide, the first time verification will create a foundation for further successful exploration and development.

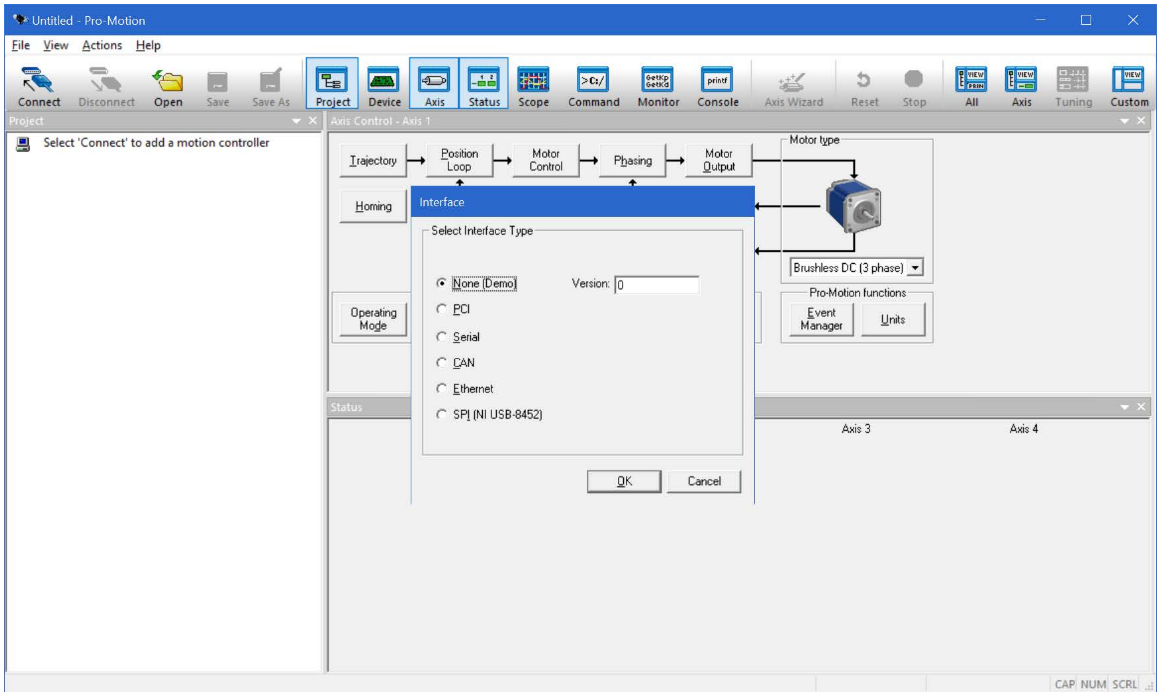
2.4.1 Establishing Communications

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

1. Make sure the N-Series ION DK is powered and connected to the PC via the 3-pin programming cable as shown in Figure 2-4.
2. Launch the Pro-Motion application.

   When Pro-Motion is launched you will be prompted with an Interface selection dialog box. 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 N-Series ION is connected to the PC. It provides various selectable communication options such as serial, CAN, Ethernet.



(● None (Demo)) Version: 0

Motor type: Brushless DC (3 phase)

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

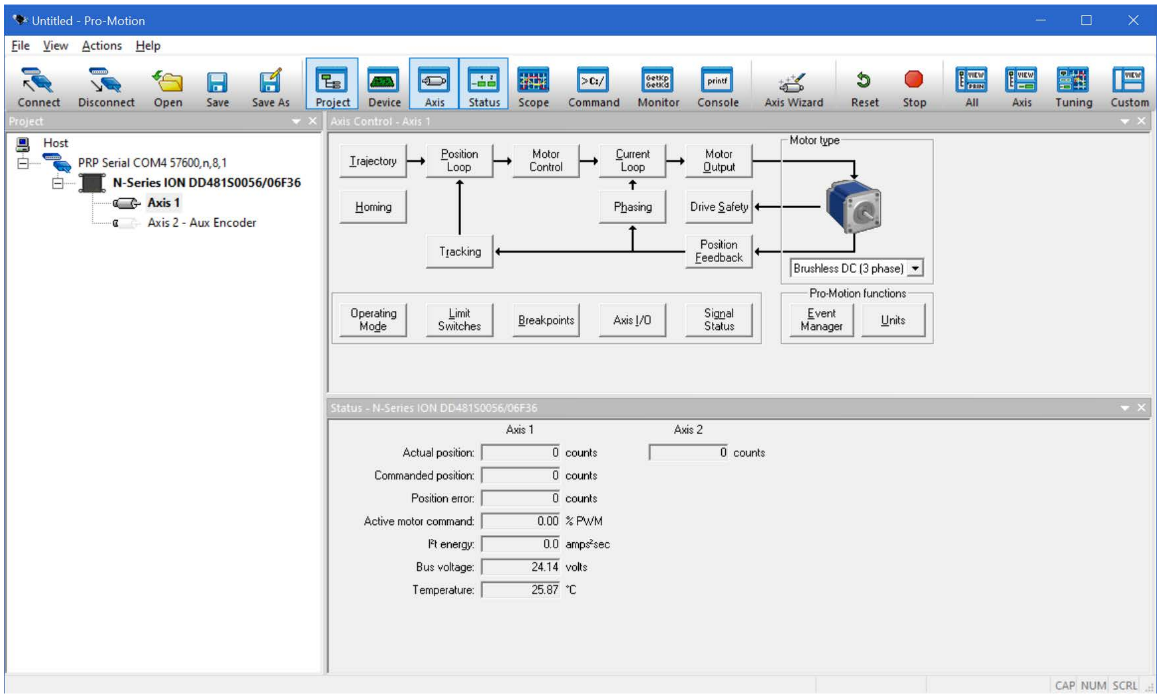
If you are not sure which of the listed COM ports is the correct one, you may use the following procedure:

- First, unplug the 3-pin programming 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 3-pin programming 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 3-pin programming cable.
- 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, a set of object graphics loads into the Project window to the left, as shown in the following figure.



Host
PRP Serial COM4 57600,n,8,1
N-Series ION DD481S0056/06F36
Axis 1
Axis 2 - Aux Encoder

Motor type: Brushless DC (3 phase)

Status - N-Series ION DD481S0056/06F36

	Axis 1		Axis 2	
Actual position:	0	counts	0	counts
Commanded position:	0	counts		
Position error:	0	counts		
Active motor command:	0.00	% PWM		
I²t energy:	0.0	amps²sec		
Bus voltage:	24.14	volts		
Temperature:	25.87	°C		

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.

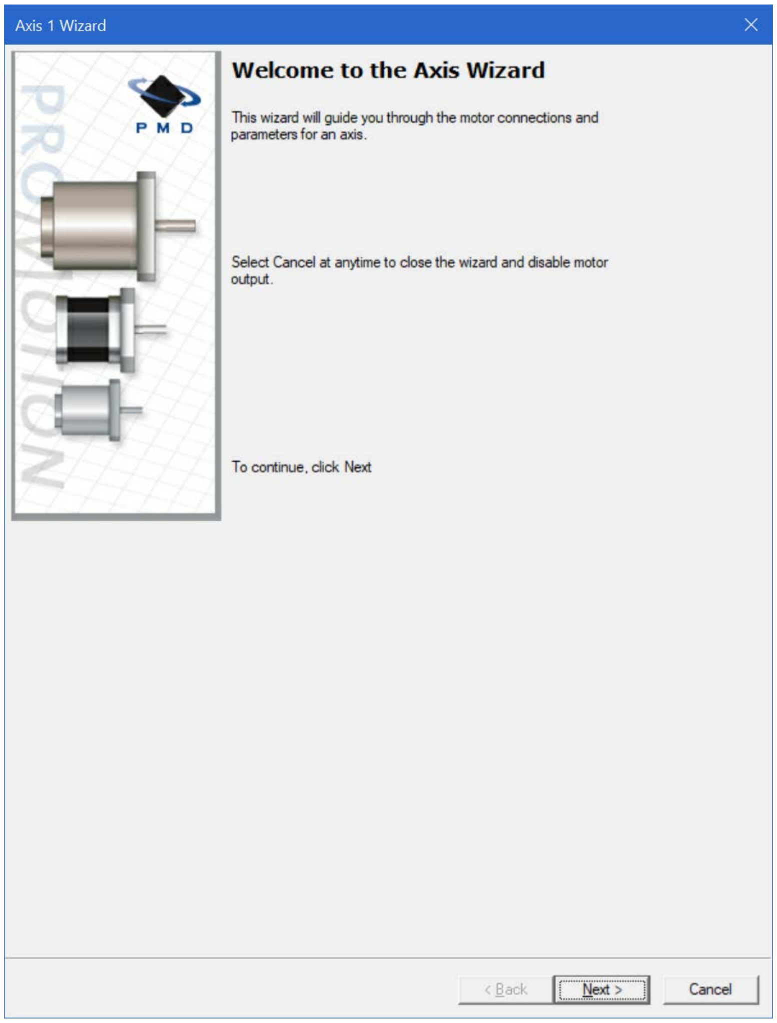
2.4.2 Running the Axis Wizard

The next step is to verify correct operation of the system thereby verifying correct connections to the motor setup. All of this can be conveniently accomplished using Pro-Motion’s Axis Wizard.

To operate the Axis Wizard:

1. Select axis 1 to initialize in the Project window to the left of the screen.
2. With this icon highlighted, 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.



Axis 1 Wizard

Welcome to the Axis Wizard

This wizard will guide you through the motor connections and parameters for an axis.

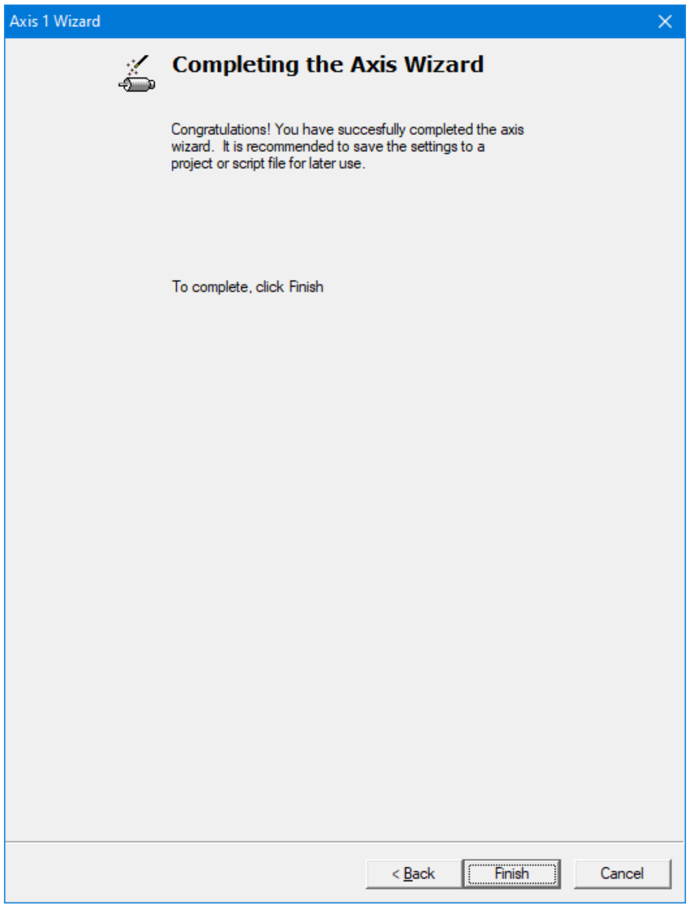
Select Cancel at anytime to close the wizard and disable motor output.

To continue, click Next

3 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.” Another section that you may find helpful is Section 3.11.2, “Application Note — Tuning the Position Loop.”

The last screen in the axis wizard 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 without the need to re-run the Axis Wizard.



Completing the Axis Wizard

Congratulations! You have succesfully completed the axis wizard. It is recommended to save the settings to a project or script file for later use.

To complete, click Finish

4 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.”

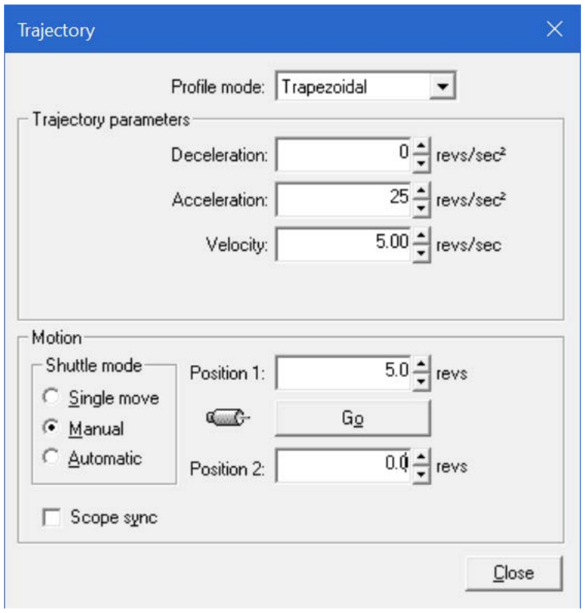
5 When you have specified a file name and saved your settings select Finish at the bottom of the screen. You will now be returned to the main Pro-Motion screen.

2.4.3 Exercising the Motor

The next step is to perform a simple move to verify the motor is reacting correctly to programmed commands.

To perform a simple move:

1 Click the Trajectory button in the Axis Control window. The Trajectory dialog box appears.



2 In the Profile mode list select Trapezoidal, and then select Manual in the lower left.

3 Enter motion profile settings for deceleration, acceleration, velocity, and two destination positions (Position 1, Position 2) 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 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 motion occurs in a stable and controlled fashion. When the motion completes click Go again to confirm that the motion in the opposite direction is also stable and controlled.

Congratulations! First-time system verification for this axis is now complete.

2.5 Next Steps

When ready, to continue exercising and optimizing the motion system and to begin developing the software for your control application refer to the subsequent chapters of this manual listed below.

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 ICs.

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.

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

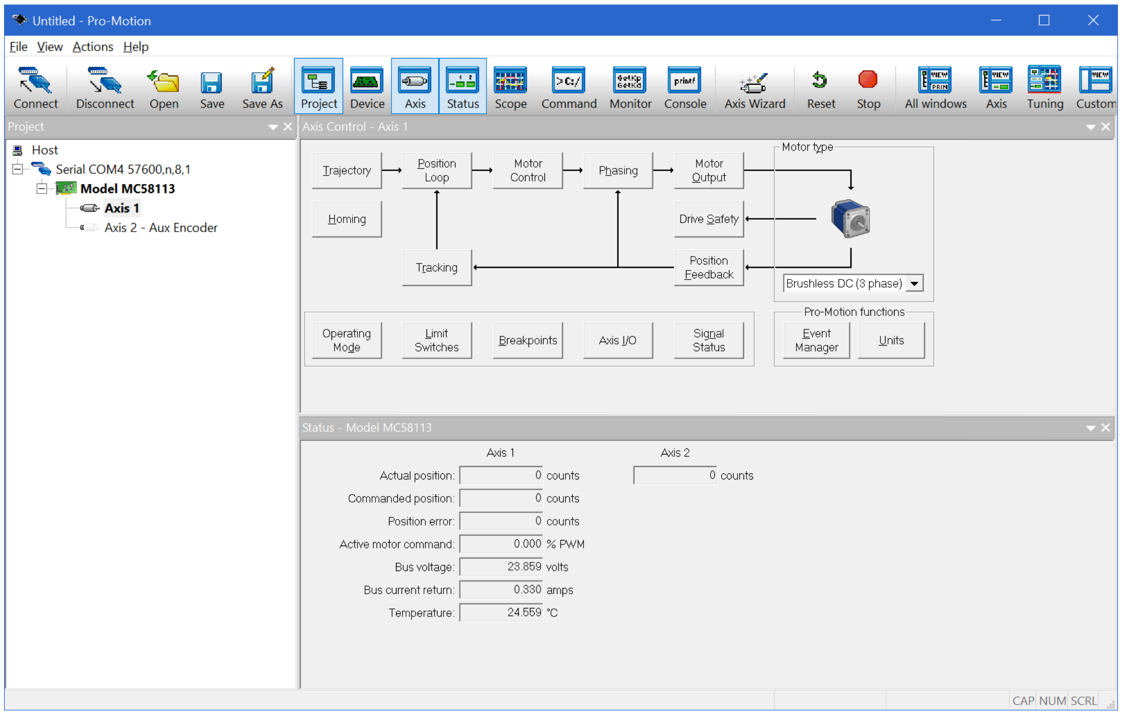
3. Going Further with Pro-Motion

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



	Axis 1		Axis 2	
Actual position:	0	counts	0	counts
Commanded position:	0	counts		
Position error:	0	counts		
Active motor command:	0.000	% PWM		
Bus voltage:	23.859	volts		
Bus current return:	0.330	amps		
Temperature:	24.559	°C		

The screen capture above shows an example of Pro-Motion as it appears after first launching and connecting to a single-axis PMD 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

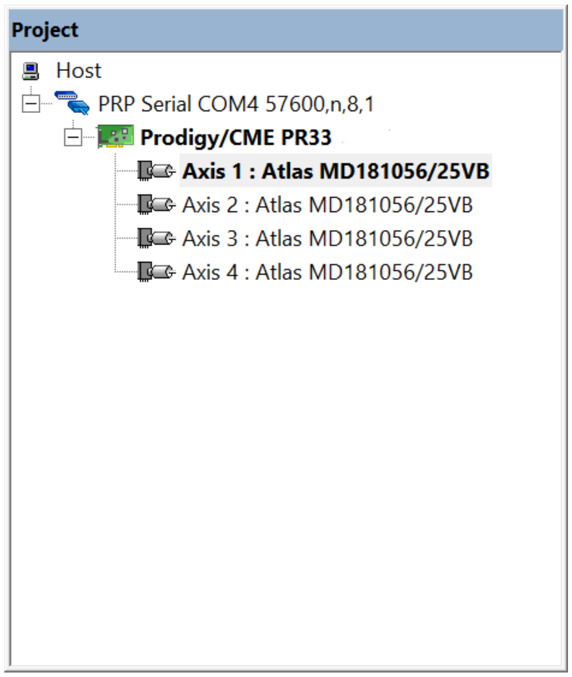
The Reset button will immediately reset the currently selected device in the Project window. The Stop button will immediately stop the motion of the axis selected in the Project window.

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



Project

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

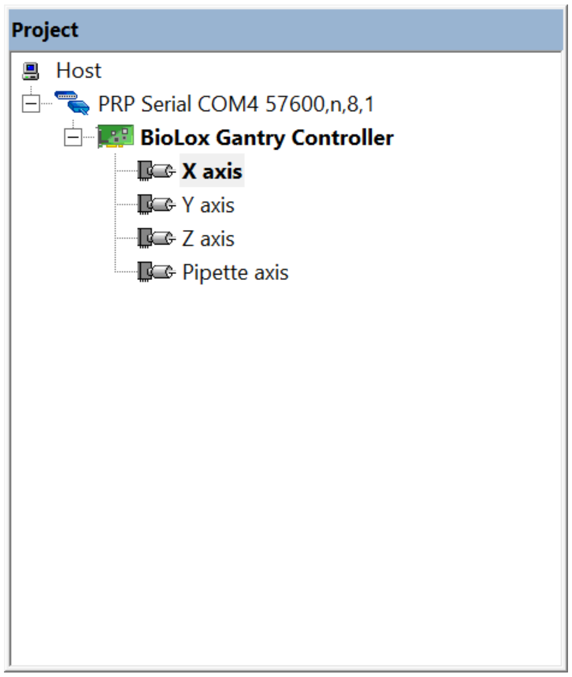
The Project window shows all PMD controllers presently connected to Pro-Motion. It presents these connections using a Windows Tree View scheme. The above screen capture shows an example Project window for a four axis controller.

In addition to displaying an icon and part number for each connected PMD controller the Project window 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. Single axis devices such as a DK58113 developer kit board or 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 labels from the previous example Project window have been changed.



Project

Host

PRP Serial COM4 57600,n,8,1

BioLox Gantry Controller

X axis

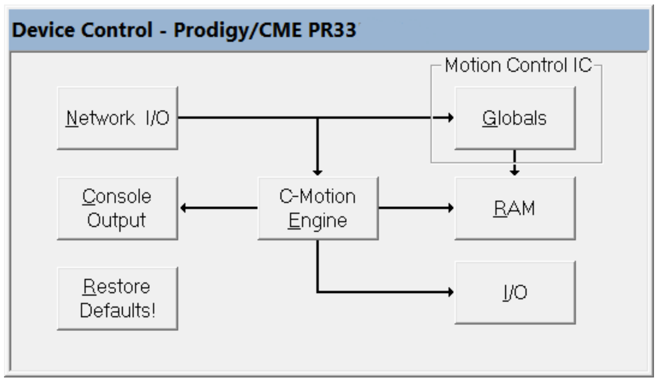
Y axis

Z axis

Pipette axis

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


Device Control - Prodigy/CME PR33

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.

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.

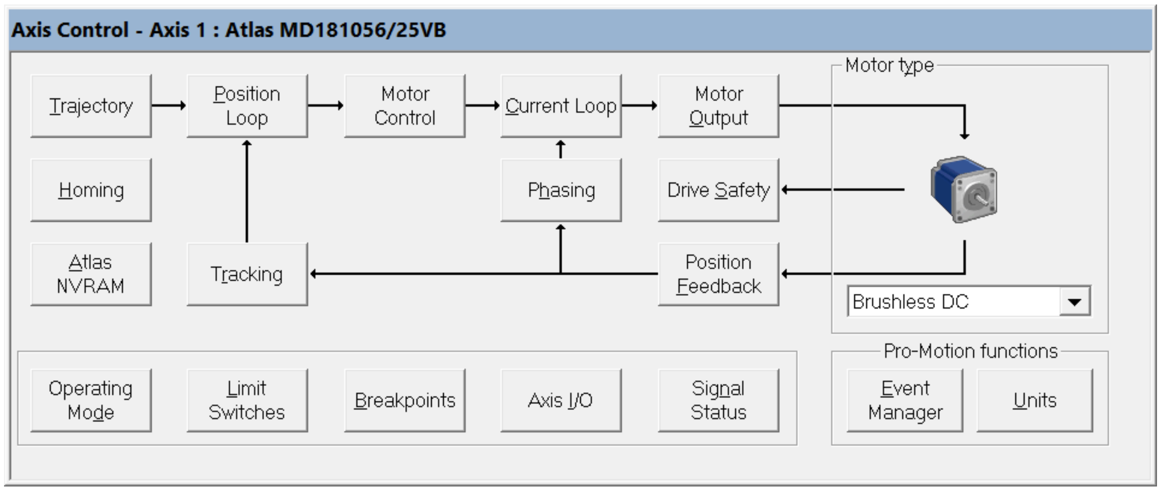
Restore Defaults – This function reverts the PMD controller's parameters to their default conditions. Any changes that may occur as a result of this operation occur in the device's NVRAM. Only after a reset or power cycle will NVRAM values become the active settings used by the PMD controller. Particular 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. For products that support multiple functions per pin such as N-Series ION drives this module also allows programming of those pin MUX settings.

The list of clickable function boxes above represents a superset of the boxes that may appear is the Device Control window for a particular PMD controller. The actual boxes displayed is based on the type of PMD controller that is selected. For example, motion IC developer kit boards such as DK58113 or DK58420 will display fewer selectable boxes while products such as ION/CME Drives and Prodigy/CME boards will display more of these boxes.

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.

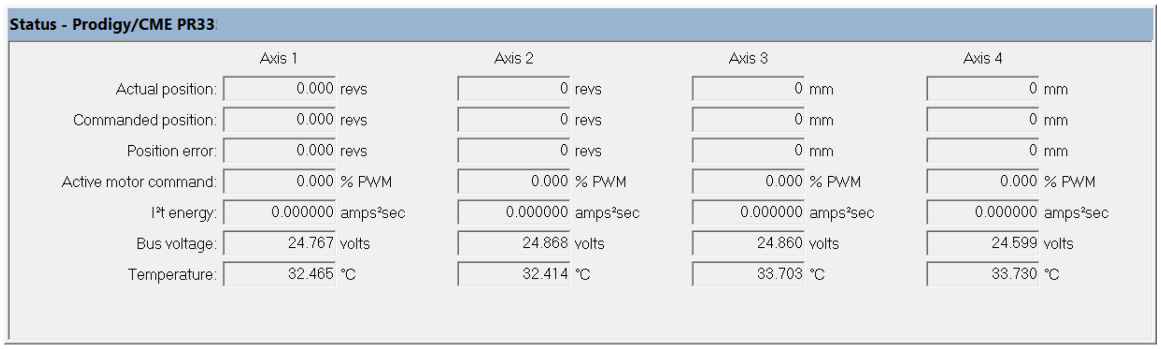
Some of the boxes at the bottom and left hand side of the Axis Control window are not connected via the control flow arrows. These provide access to motion control settings such as for limit switches, breakpoints, axis I/O, signal status, event management, units, homing, and Atlas NVRAM (for axes which 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 developer kit 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 an example of a Status window display, in this case for a four axis controller. 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. Single axis controllers such as ION drives or MC58113 IC-based systems will display two axes in the Status window, the primary axis (#1) and the auxiliary axis (#2).



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
I²t energy:	0.000000 amps²sec	0.000000 amps²sec	0.000000 amps²sec	0.000000 amps²sec
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 various 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…

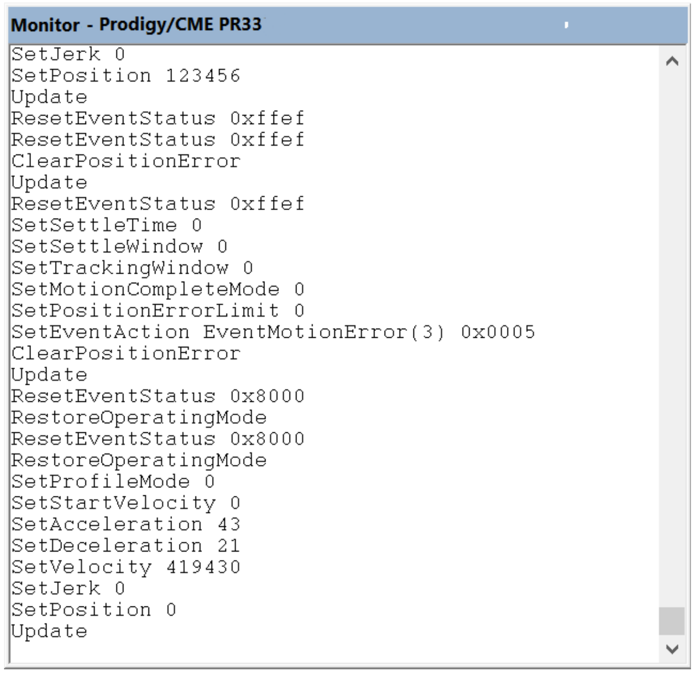
By default this background status polling is enabled, however there are situations where it may become disabled. The first is after some types of communication errors between Pro-Motion and the connected PMD controller. To avoid continually generating communication error warnings automatic polling may 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 difficult 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.

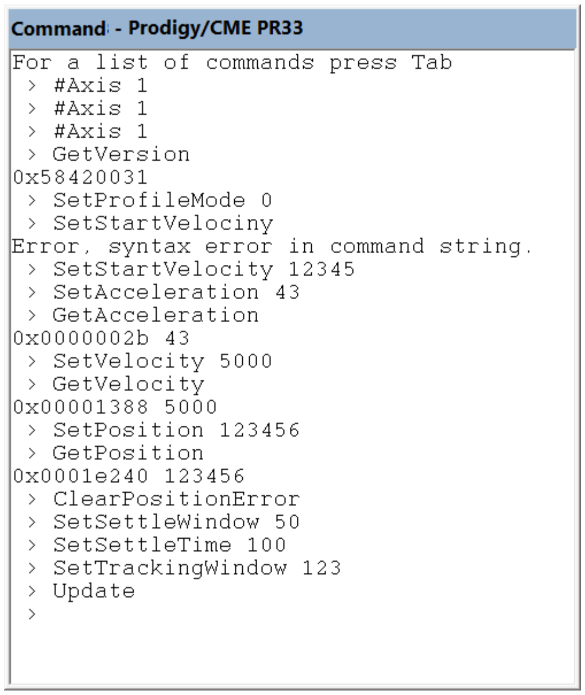
Here is some additional information about the Monitor window that you may find helpful:

- 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 devices see Chapter 5, *Software Development*.

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.

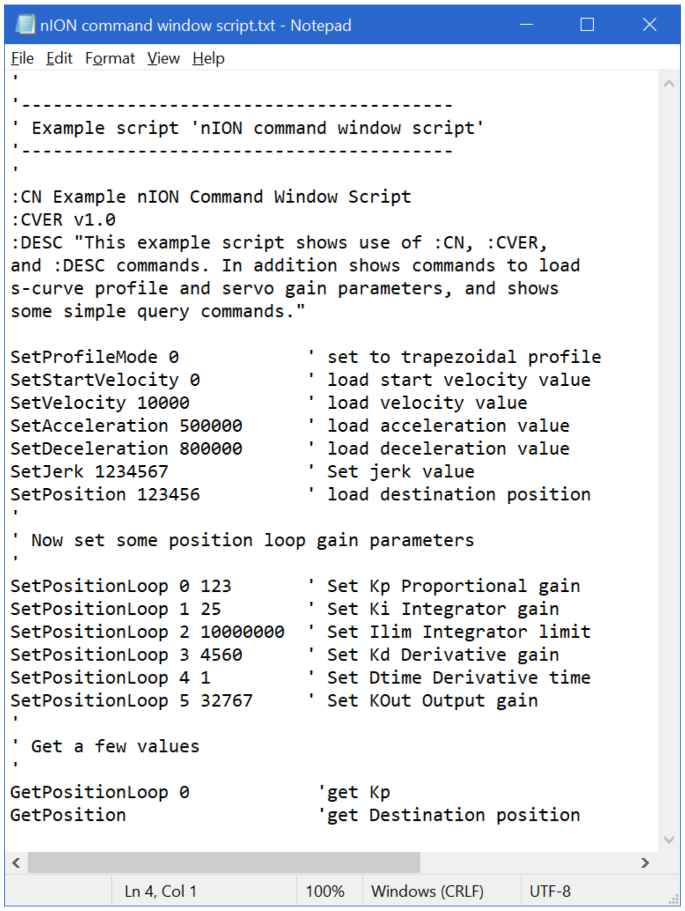
The *C-Motion Magellan Programming Reference* provides detailed syntax and argument definitions for each enterable command. Go to the alphabetized Instruction Reference section of this manual to lookup specific commands. The command syntax is indicated at the top with argument definitions provided just 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.

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.
- 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


nION command window script.txt - Notepad
File Edit Format View Help
Ln 4, Col 1
100%
Windows (CRLF)
UTF-8

```
'
'-----------------------------------------
' Example script 'nION command window script'
'-----------------------------------------
'
:CN Example nION Command Window Script
:CVER v1.0
:DESC "This example script shows use of :CN, :CVER,
and :DESC commands. In addition shows commands to load
s-curve profile and servo gain parameters, and shows
some simple query commands."

SetProfileMode 0            ' set to trapezoidal profile
SetStartVelocity 0          ' load start velocity value
SetVelocity 10000           ' load velocity value
SetAcceleration 500000      ' load acceleration value
SetDeceleration 800000      ' load deceleration value
SetJerk 1234567             ' Set jerk value
SetPosition 123456          ' load 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
GetPosition                  'get Destination position
```

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.

The screen capture above shows an example of a Command window script. 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.

:CN, :CVER, and :DESC entries – Command window scripts for N-Series ION drives support three special entries used to identify the content of a script. These are the :CN, :CVER and :DESC entries. Only one entry for each is allowed per script file and the entry must begin at the first character of a new line. Although by convention these entries are listed at the top of the script, this is not a requirement. They can be located anywhere in the script and appear in any order and still be recognized correctly. Use of these entries is optional.

The main advantage of using these special entries in a script file is that Pro-Motion can store them into the PMD controller's Magellan IC NVRAM, and similarly display them when read from the motion IC NVRAM. This is useful for identifying the content of a script when stored into the Magellan IC NVRAM. For more information on storing scripts into the Magellan IC NVRAM see Section 3.7.3, "Downloading Scripts to the Motion IC NVRAM."

Note that the actual content specified with the :CN, :CVER and :DESC entries does not affect how the script's commands are parsed or executed. When a Command window script containing one or more of these entries is executed, these entries are ignored. The content is solely useful to whomever is viewing them for understanding the content of the NVRAM. Here is additional information on each of these three entries:

:CN – The :CN entry is intended to register an ASCII script name. The name string follows the ":CN" entry after a space without double quotes and consists of all characters till the newline excepting trailing spaces.

:CVER – The :CVER entry is intended to record a version number for the script. This may be useful to track different versions of the script as it is being developed. The format of the :CVER field is the same as for :CN, namely an ASCII string without beginning and ending double quote characters that is read up to the end of the line.

:DESC – The :DESC entry is intended to register a narrative description of the content of the script. Unlike :CN and :CVER this entry uses double quotes to demarcate the beginning and end of the DESC content. In addition, newlines do not represent the end of the field content, only a second double quote character ends the content. This means the description can extend over multiple lines of ASCII text.

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.

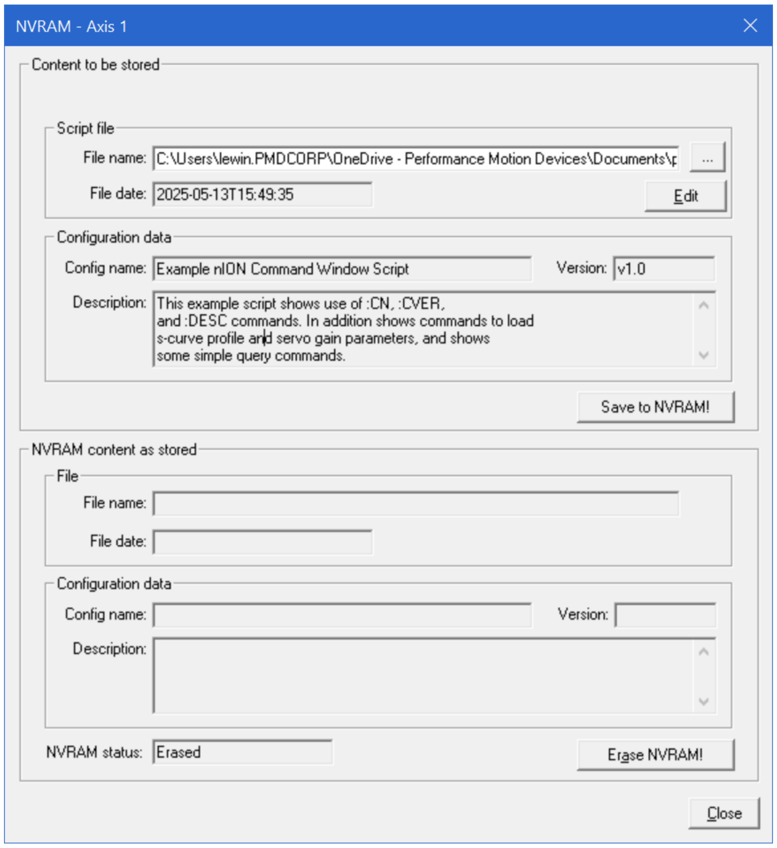
Only Magellan Motion Control IC configuration settings are exported to scripts via this export function. For PRP devices such as ION/CME drives or Prodigy/CME Machine Controller bards non Motion Processor resource configuration settings are not exported as part of this function. For more information on PRP devices see Chapter 5, *Software Development*.

3.7.3 Downloading Scripts to the Motion IC NVRAM

N-Series IONs provide the ability to store configuration information such as gain parameters, drive-related safety parameters, and other parameters into the internal NVRAM (non-volatile memory) of the motion IC within the N-Series ION. This initialization configuration content is specified using Pro-Motion Command window scripts.

Once loaded into the motion IC's NVRAM, the commands in the specified script are automatically executed and any parameter settings contained in the script become the active settings used by the IC at power-up or reset. The main advantage of storing initialization sequences in NVRAM is that the stored settings are available immediately after powerup of the N-Series ION, rather than having to be downloaded by a host PC or microcontroller.

Programming or re-programming of the motion IC’s NVRAM is accomplished by selecting the NVRAM box in Pro-Motion’s Device Control window. The screen capture below shows the dialog box that appears.



File name: C:\Users\lewin.PMDCORP\OneDrive - Performance Motion Devices\Documents\p
File date: 2025-05-13T15:49:35
Config name: Example nION Command Window Script
Version: v1.0
Description: This example script shows use of :CN, :CVER, and :DESC commands. In addition shows commands to load s-curve profile and servo gain parameters, and shows some simple query commands.

File name:
File date:
Config name:
Version:
Description:
NVRAM status: Erased

Specify the script file in the “File to download” field. If you click the small box to the right a Windows Open dialog box will appear. When ready click Download! to store the script file content into the IC’s NVRAM. Similar to scripts that are executed in the Command window and sent to the IC via a communication port, the script content is parsed by Pro-Motion, converted into hexadecimal-coded packets, and then sent to the motion IC for storage.

If the NVRAM already contains stored content it will be overwritten when a new script is downloaded. To erase the IC’s NVRAM memory select the Erase! Button. This will restore the motion IC to use its default parameter settings after powerup.

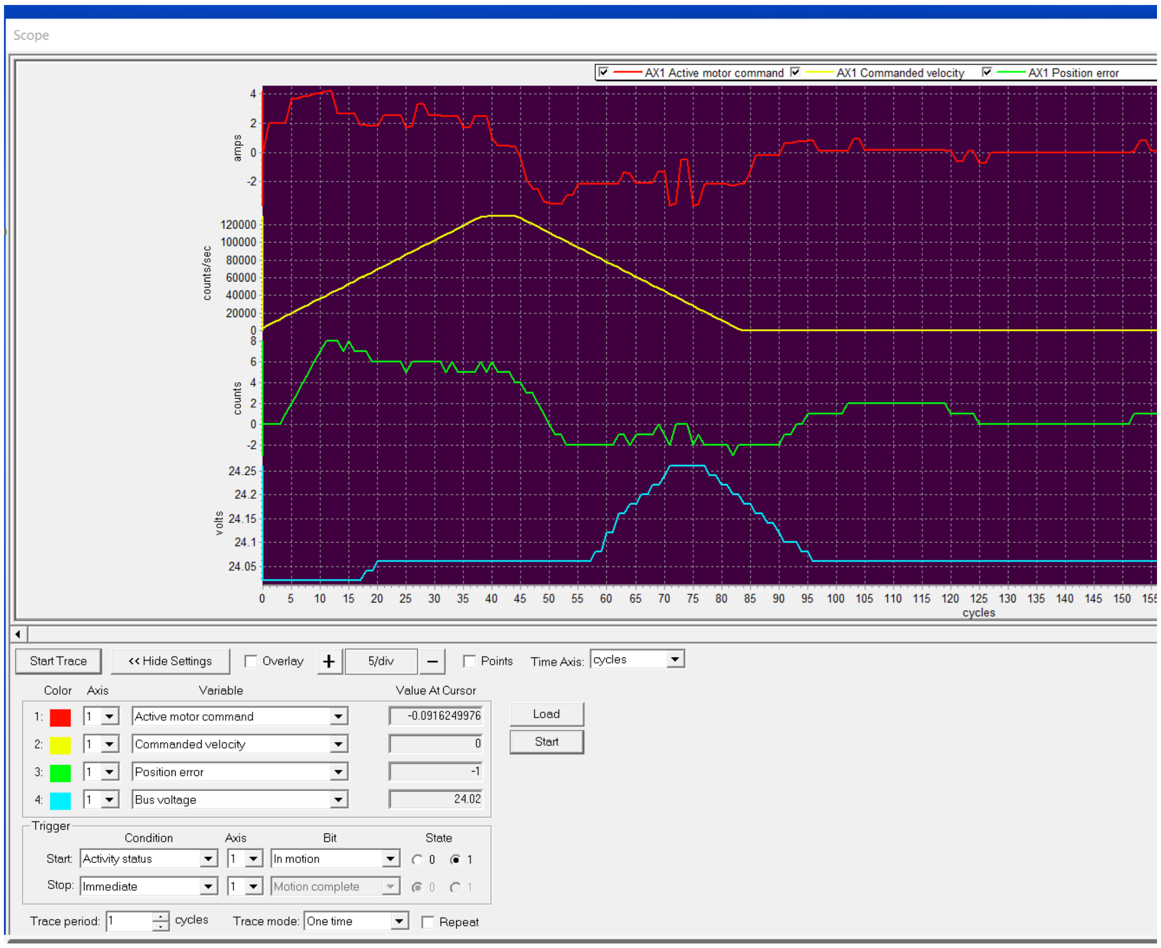
3.8 Scope Window

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 (depending on the PMD controller).

Once a trace operation is specified the motion 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.



Start Trace | << Hide Settings | Overlay | + | 5/div | − | Points | Time Axis: cycles

	Color	Axis	Variable	Value At Cursor
1:	🟥	1	Active motor command	-0.0916249976
2:	🟨	1	Commanded velocity	0
3:	🟩	1	Position error	-1
4:	🟦	1	Bus voltage	24.02

Load | Start

Trigger

	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	0 [✓ 1]
Stop:	Immediate	1	Motion complete	[✓ 0] 1

Trace period: 1 cycles Trace mode: One time Repeat

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 ?Sec per cycle. So specifying a period of 1 cycle captures data at ~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 then 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...

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, "Setting Up the PMD Controller for RS232 Communications."

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 format that can be read (and written) by Pro-Motion, but is 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.

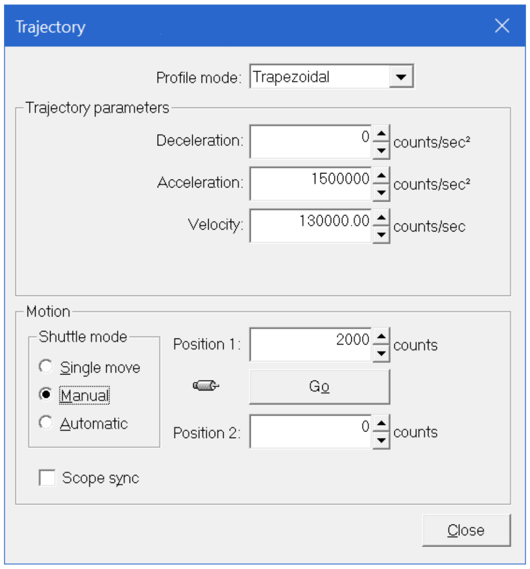
3.11 Pro-Motion Application Notes

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.

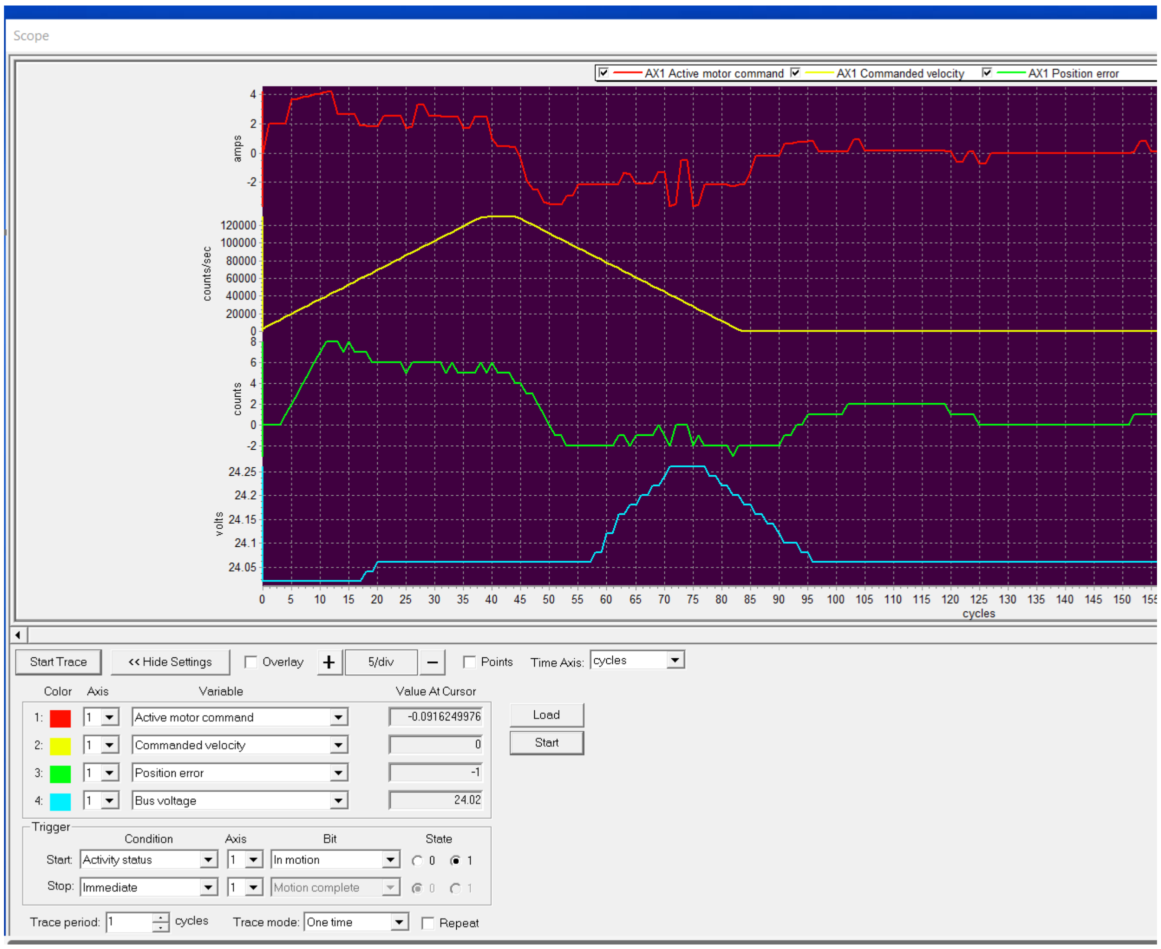
Based on the resultant graph display you may also want to change the shuttle move distance so that the entire trapezoidal move is displayed.

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:



Time Axis: cycles

Color	Axis	Variable	Value At Cursor
1: 🟥	1	Active motor command	-0.0916249976
2: 🟨	1	Commanded velocity	0
3: 🟩	1	Position error	-1
4: 🟦	1	Bus voltage	24.02

Trigger

	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	0 [✓ 1]
Stop:	Immediate	1	Motion complete	[✓ 0] 1

Trace period: 1 cycles Trace mode: One time Repeat

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

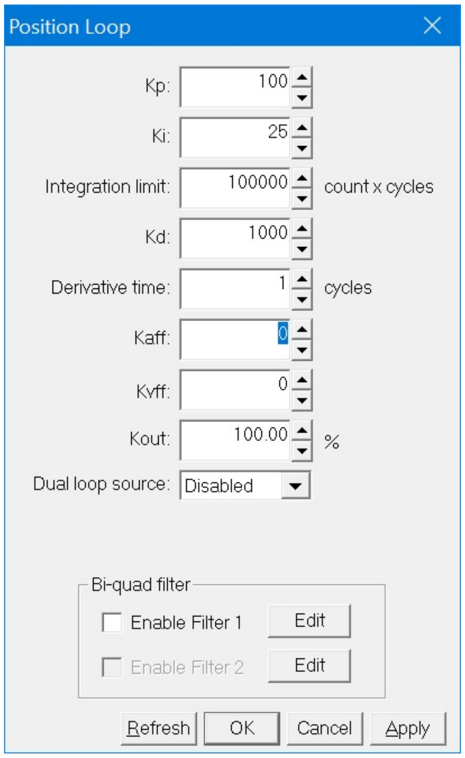
This application note provides an approach to developing PID position loop gain settings. Note that this section only applies to servo motors (DC Brush, Brushless DC) and motors using a closed loop stepper method.

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:



Position Loop

Kp: 100

Ki: 25

Integration limit: 100000 count x cycles

Kd: 1000

Derivative time: 1 cycles

Kaff: 0

Kvff: 0

Kout: 100.00 %

Dual loop source: Disabled

Bi-quad filter

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 Kvff (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 5 times greater than the Kp value you entered. Set the derivative time to 5 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. If the motor produces a high-

pitched chatter or whine you can lower the KD derivative gain, or alternatively you can increase the derivative time and proportionally lower the Kd value. For example if the derivative time was 5 with a Kd of 1,000, changing the derivative time to 10 means the Kd should change to 500.

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

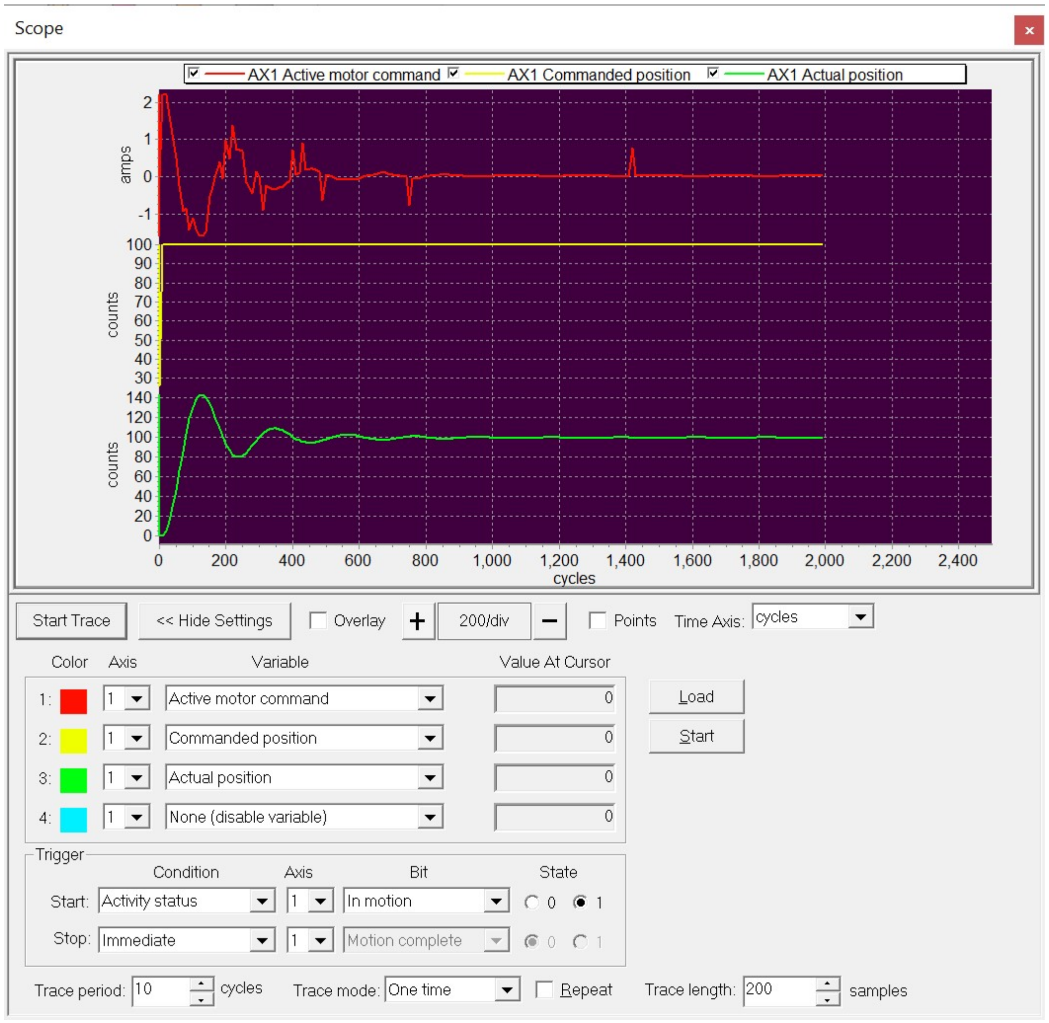
- View the scope display to determine whether the response is underdamped, critically damped, or overdamped (see below for instructions on how to do this)
- Adjust the gain settings accordingly
- Make another step move

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.



Overlay: ☐ 200/div Points: ☐ Time Axis: cycles

	Color	Axis	Variable	Value At Cursor
1:	🟥	1	Active motor command	0
2:	🟨	1	Commanded position	0
3:	🟩	1	Actual position	0
4:	🟦	1	None (disable variable)	0

Trigger

	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	0 [✓ 1]
Stop:	Immediate	1	Motion complete	[✓ 0] 1

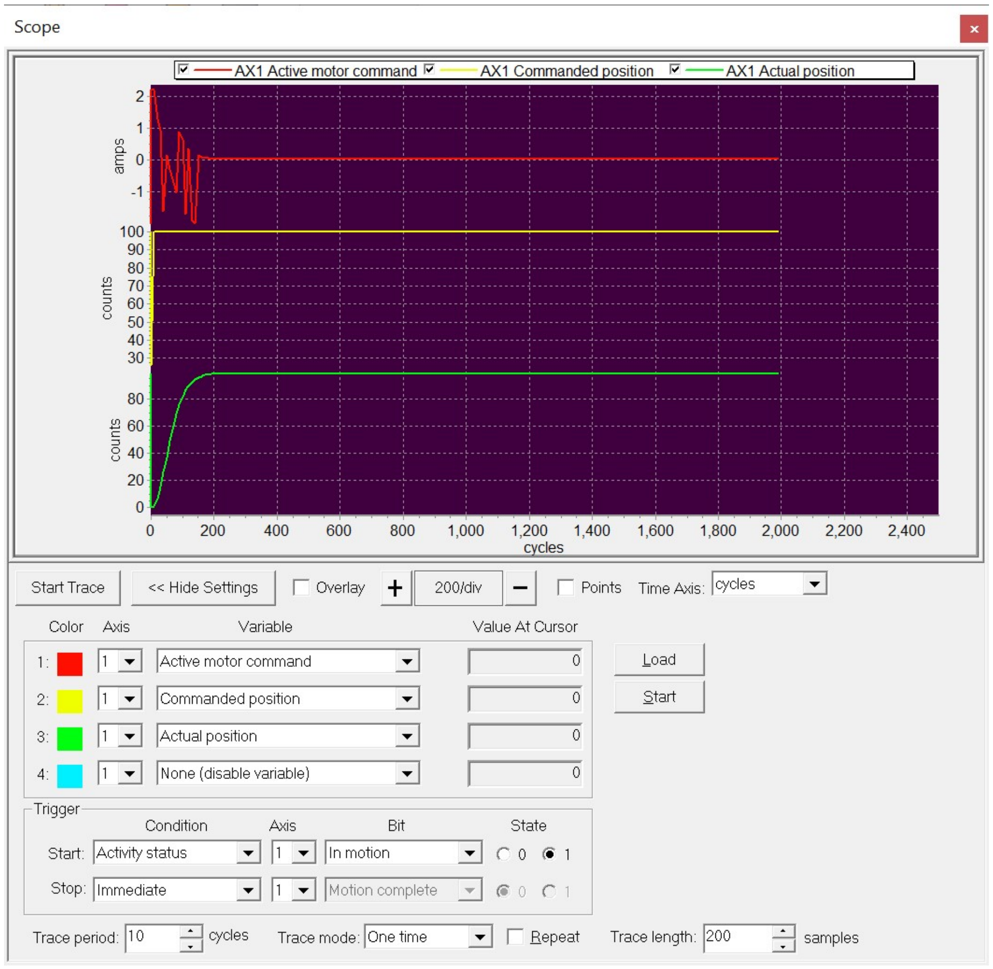
Trace period: 10 cycles Trace mode: One time ☐ Repeat Trace length: 200 samples

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.



Overlay: ☐ 200/div Points: ☐ Time Axis: cycles

	Color	Axis	Variable	Value At Cursor
1:	🟥	1	Active motor command	0
2:	🟨	1	Commanded position	0
3:	🟩	1	Actual position	0
4:	🟦	1	None (disable variable)	0

Trigger

	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	0 [✓ 1]
Stop:	Immediate	1	Motion complete	[✓ 0] 1

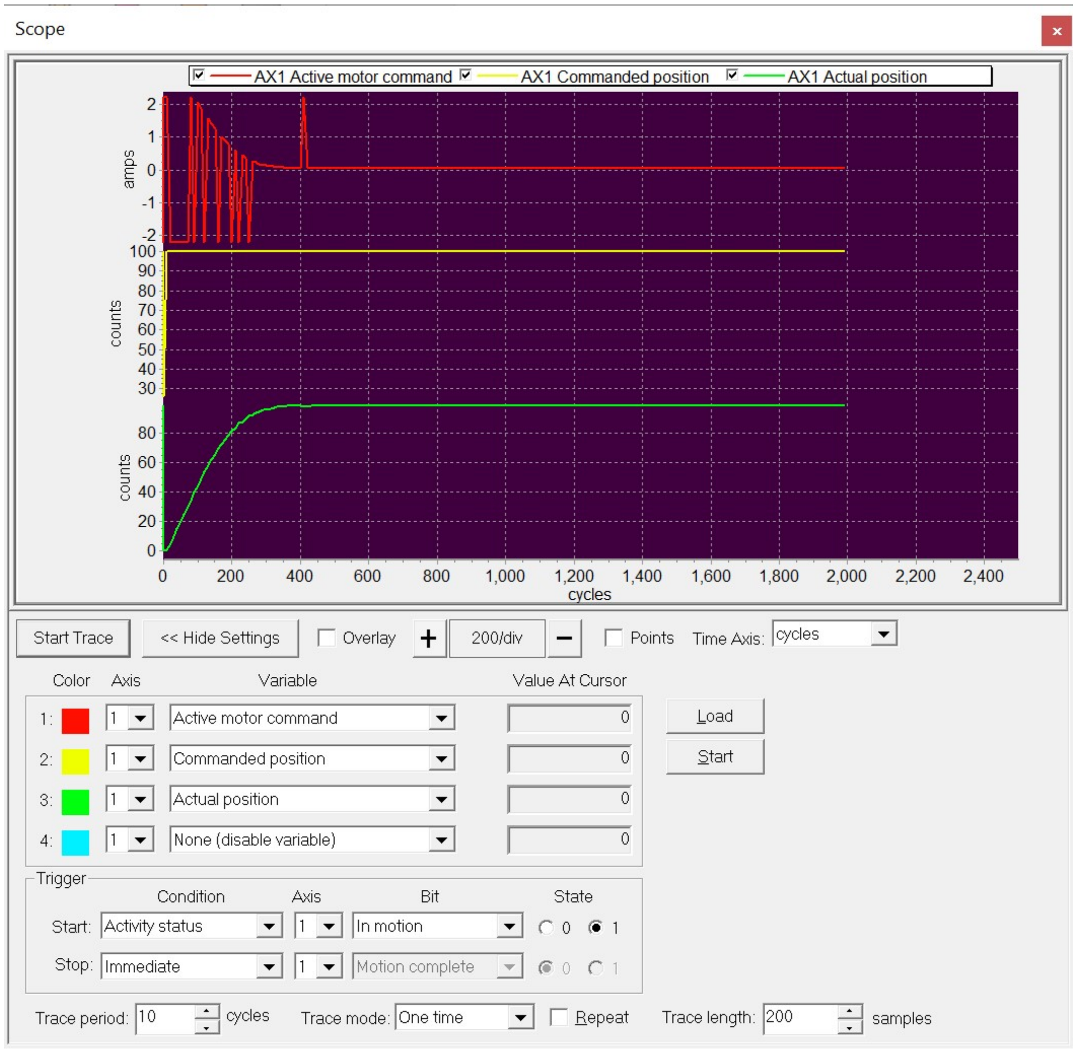
Trace period: 10 cycles Trace mode: One time ☐ Repeat Trace length: 200 samples

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.



Time Axis: cycles

Color	Axis	Variable	Value At Cursor
1: 🟥	1	Active motor command	0
2: 🟨	1	Commanded position	0
3: 🟩	1	Actual position	0
4: 🟦	1	None (disable variable)	0

Trigger

	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	0 [✓ 1]
Stop:	Immediate	1	Motion complete	[✓ 0] 1

Trace period: 10 cycles Trace mode: One time ☐ Repeat Trace length: 200 samples

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 Kd, trying to find the smallest value of Kd that still gives a critically damped response.

For purposes of illustration in the above three example graphs the Kp setting was 100 and the Kd 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 Kp to as high a value as possible while still finding associated Kd values that can create a critically damped response. Higher Kp values will result in more accurate tracking and faster responses to command position changes.

As you try higher and higher Kp values, at some value of Kp you will find that there is no value of Kd that can create a stable and critically damped response. At this point you should reduce the Kp setting by 35-50% and determine the associated critically damped Kd setting. Backing down from the 'borderline' Kp 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 Derivative Time

The Derivative Time setting is the period at which the PID loop derivative contribution is calculated in units of sample time cycles. Increasing the Derivative Time can be useful for reducing axis 'chatter' by effectively introducing a low pass filter on the PID loop's control response. This in turn allows the effective Kd damping contribution to be increased.

The impact of the Kd on the PID loop command output is a direct multiple of the derivative time. For example if the derivative time is set to 10 and the Kd value is 100 this would be an equivalent impact as a derivative time of 5 and a Kd of 200. Generally speaking larger motors with larger loads will use higher Derivative Time settings and vice versa for smaller motors with smaller loads.

3.11.2.8 Setting Ki and Ilimit

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

3.11.2.9 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.

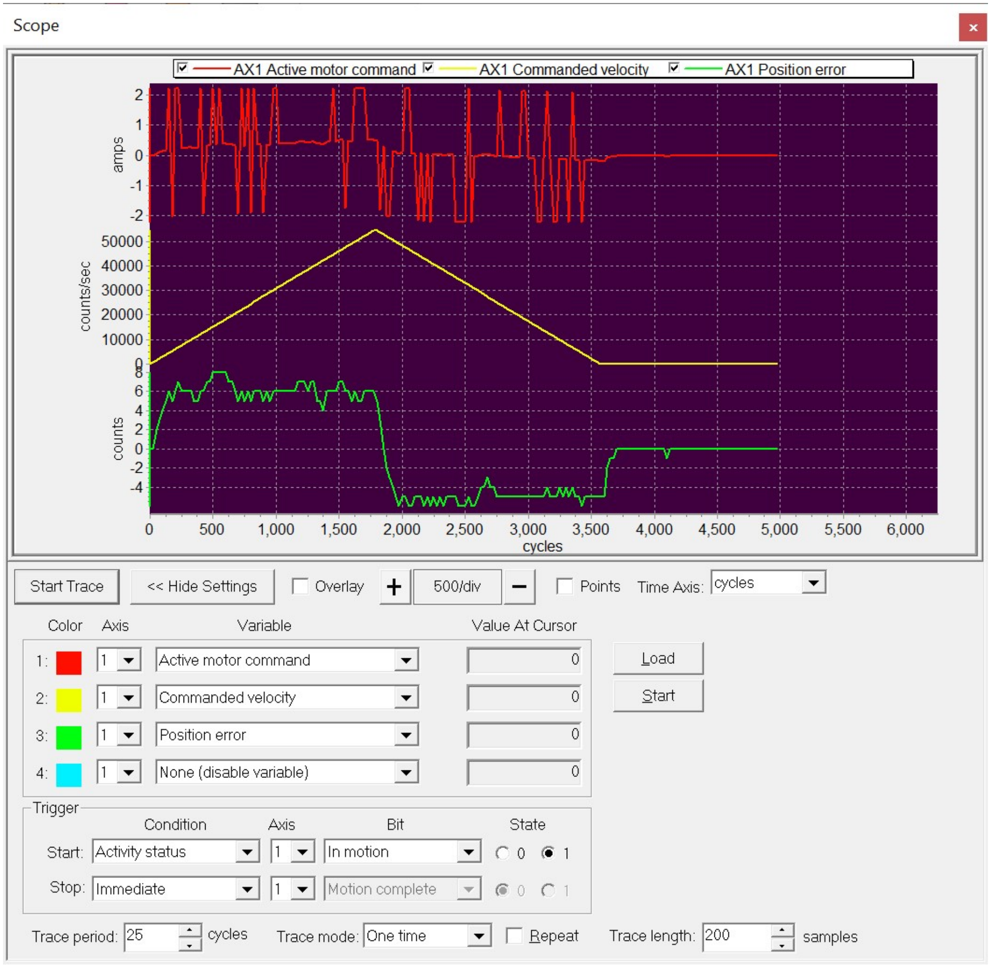
Bode plots can be used in different ways and can help characterize your mechanical system as well as determine important characteristics of the operating control loop such as the bandwidth and phase margin. The Pro-Motion Bode plot tool can be accessed from the *View/BodePlots* Pro-Motion menu selection.

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.



Time Axis: cycles

Color	Axis	Variable	Value At Cursor
1: 🟥	1	Active motor command	0
2: 🟨	1	Commanded velocity	0
3: 🟩	1	Position error	0
4: 🟦	1	None (disable variable)	0

Trigger

	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	0 [✓ 1]
Stop:	Immediate	1	Motion complete	[✓ 0] 1

Trace period: 25 cycles Trace mode: One time Repeat Trace length: 200 samples

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.

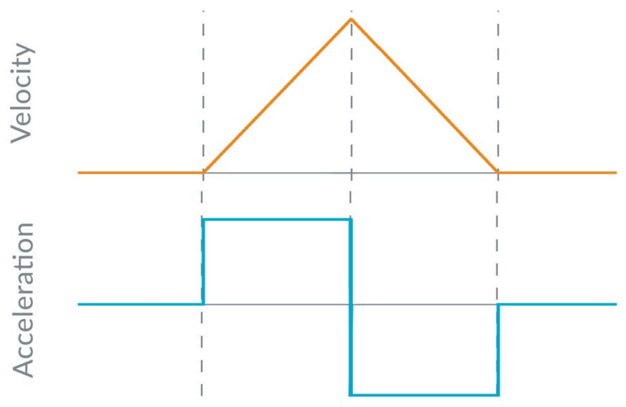


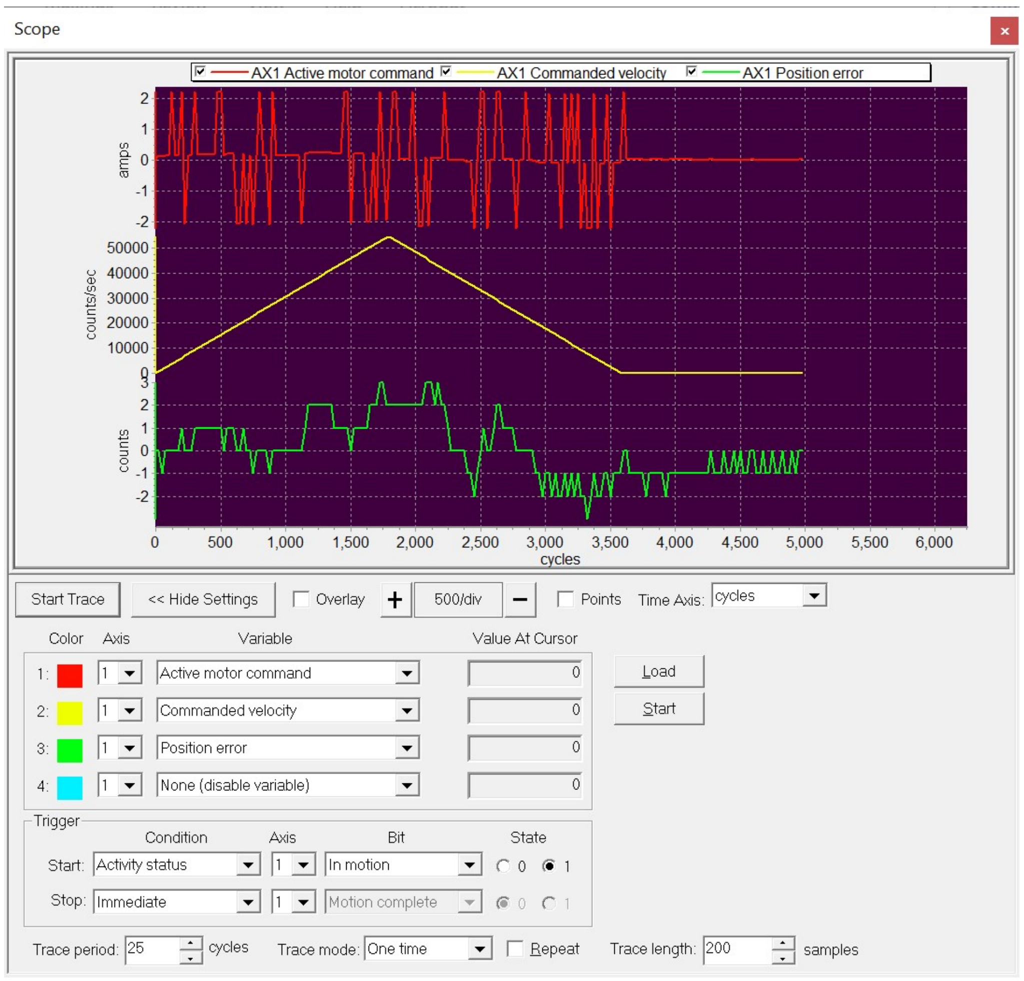
Figure 3-1: Velocity and Acceleration Versus Time for Point-To-Point Trapezoidal Trajectory

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.



Overlay [ ] | 500/div | Points [ ] | Time Axis: cycles

Color	Axis	Variable	Value At Cursor
1: 🟥	1	Active motor command	0
2: 🟨	1	Commanded velocity	0
3: 🟩	1	Position error	0
4: 🟦	1	None (disable variable)	0

Trigger

	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	0 [● 1]
Stop:	Immediate	1	Motion complete	[● 0] 1

Trace period: 25 cycles Trace mode: One time [ ] Repeat Trace length: 200 samples

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.

3.11.4 Application Note – Monitoring and Enhancing DC Bus Voltage Stability

An important aspect of designing a robust motion control system is ensuring stability of the power supply voltage. This can be challenging because motors, as they accelerate and decelerate, may demand rapid increases in current from the power supply, or may actually send current back to the power supply.

In this application note we will walk through how to characterize the stability of the voltage supply under various motor operating conditions, and then suggest techniques for improving voltage stability.

There are three HV supply voltage conditions to consider for a given combination of power supply and motor operation:

1 Motor and load trajectories that result in stable operating voltage of the DC supply.

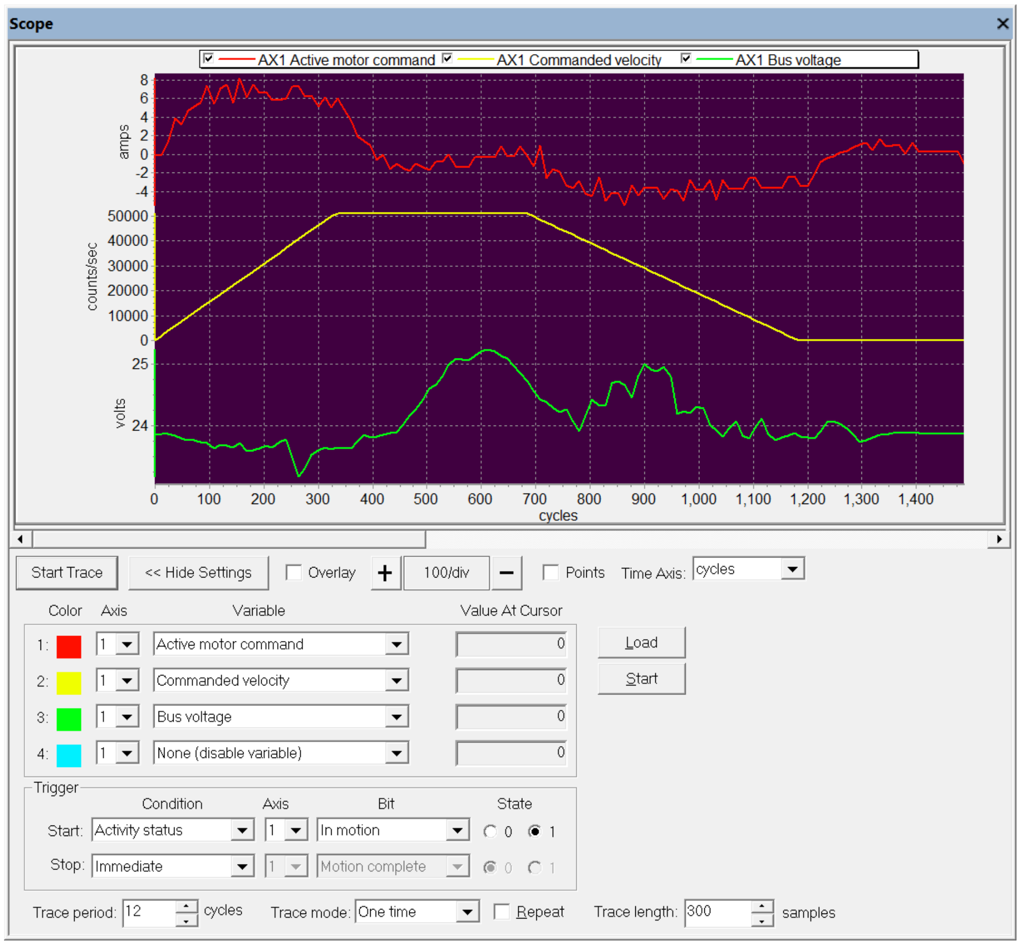
2 Motor and load trajectories that result in undesirable voltage drops because the DC supply's current output capacity is exceeded.

3 Motor and load trajectories that result in undesirable voltage increases because the DC supply cannot absorb any, or enough, current from the controller.

3.11.4.1 Stable DC Supply Voltage Operation

In this section we will use the trace capability of PMD controllers and Pro-Motion to show what stable operation of the DC supply looks like while the motor is moving. We will perform a simple point to point trapezoidal profile move while monitoring the voltage level of the DC supply.

All PMD controllers that provide amplification have the capability to measure the DC bus voltage directly and to trace this value at high speed along with other motion parameters. The resulting scope trace is shown below.



Time Axis: cycles

Color	Axis	Variable	Value At Cursor
1: 🟥	1	Active motor command	0
2: 🟨	1	Commanded velocity	0
3: 🟩	1	Bus voltage	0
4: 🟦	1	None (disable variable)	0

Trigger

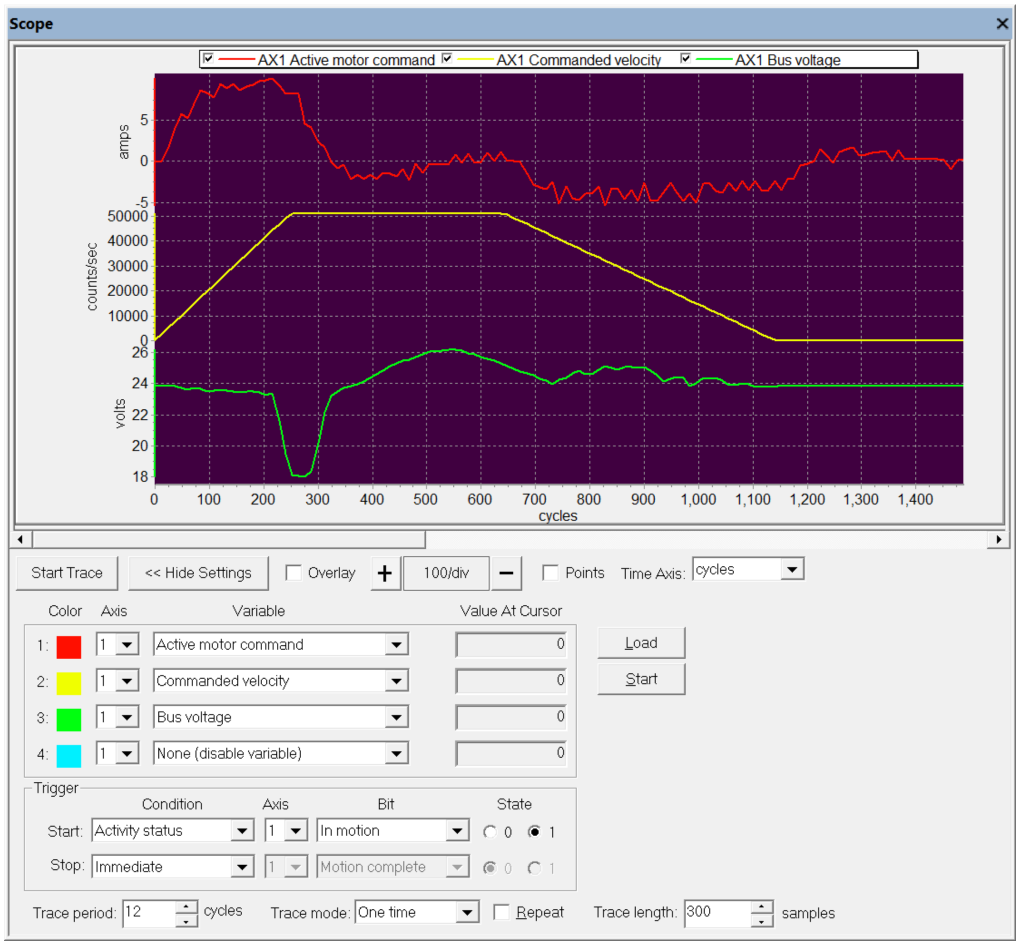
	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	0 [✓ 1]
Stop:	Immediate	1	Motion complete	[✓ 0] 1

Trace period: 12 cycles Trace mode: One time Repeat Trace length: 300 samples

In this stable voltage scenario the main feature of interest is that as the motor accelerates and decelerates, although there is a variation in the 24V supply voltage, its magnitude is small - in the above trace between 23.2 and 25.3 volts. This level of variation is acceptable in nearly all motion systems, resulting in little stress on the power supply, and in no meaningful impact on the controller's ability to maintain accurate position control through the move.

3.11.4.2 Undervoltage Condition During Motion

In the Pro-Motion scope trace below the same motor setup and load used with the scope trace above is driven at a higher acceleration, resulting in a significant drop of the supply voltage from 24V to 18V. The trajectory's coasting and deceleration phase allows the power supply to return to a stable voltage supply situation.



Overlay + 100/div − Points Time Axis: cycles

Color	Axis	Variable	Value At Cursor
1: 🟥	1	Active motor command	0
2: 🟨	1	Commanded velocity	0
3: 🟩	1	Bus voltage	0
4: 🟦	1	None (disable variable)	0

Trigger

	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	0 [✓ 1]
Stop:	Immediate	1	Motion complete	[✓ 0] 1

Trace period: 12 cycles Trace mode: One time Repeat Trace length: 300 samples

Why does the voltage decrease during trajectory acceleration? The reason that the supply voltage drops is that during acceleration the motion controller rapidly commands more current to the motor, which in turns means more current is 'requested' from the DC supply. The power supply cannot provide this dramatic increase in current and the result is a drop in voltage, an effect similar to an electrical grid brownout.

The current commanded by the motor controller is directly proportional to the amount of torque needed to accelerate the motor and load, with the torque being proportional to acceleration times the moment of rotational inrertia. For example if the acceleration doubles and the load stays the same the amount of torque (and therefore required current) doubles. The significance of this is that to establish the supply voltage stability for a given mechanism the worst case current demand should be investigated. For a single axis machine this would be the highest acceleration move with the heaviest load, and for multi-axis machines this would be the same, but including all axes that move simultaneously.

Why is a drop in voltage a problem? There are a number of reasons why a voltage drop may be a problem. The first is that the power supply may become overstressed. Depending on its design, a request for current that exceeds its output capacity, may overheat or damage the power supply.

Another problem is that servo stability may be impacted. In extreme cases of voltage drop the current control loop or the position control loop may become unstable. Finally, since motion controllers generally derive their internal logic power from the main supply voltage, if that supply voltage drops too far the controller may cease functioning. Modern motion controllers (including PMD's products) have a feature called undervoltage protection which automatically disables the motor drive and shuts down the trajectory generator. When an undervoltage event occurs, although the move is aborted, the more dangerous situation where the motion controller completely shuts down is avoided.

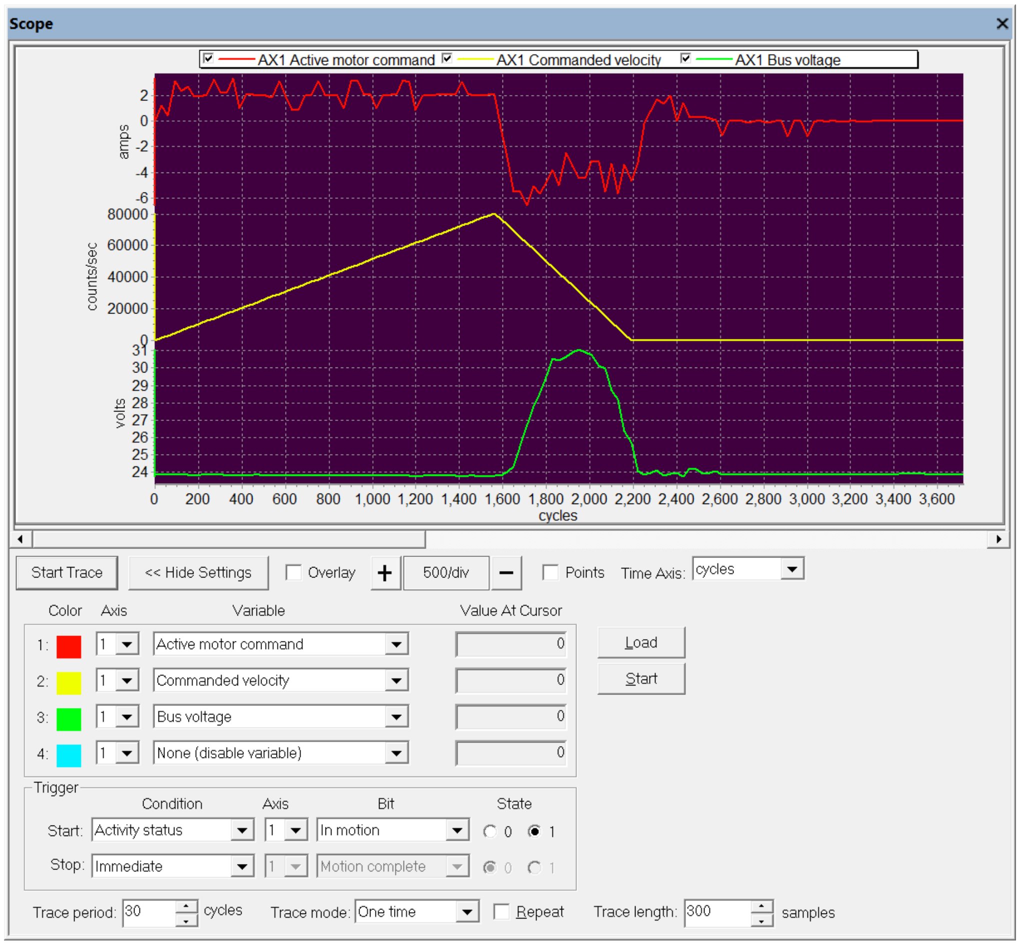
What are the remedies to voltage drop during motor acceleration? The simplest solution is to reduce the acceleration rate. This will directly reduce the amount of current the power supply needs to deliver. See Section 3.11.1, "Application Note — Determining the Optimum Trajectory Acceleration," for an application note on this exact subject.

If reducing the acceleration is not an option the next two options to consider are switching to a power supply with greater current output or adding capacitance to the DC supply line. Especially to provide short bursts of additional current during acceleration, adding capacitance is often the preferred option.

One more option for reducing supply voltage sag during motor acceleration is to switch to a motor that can output more torque for a given current. Note that this might also be accompanied by the need for an increase in the operating supply voltage.

3.11.4.3 Overvoltage Condition During Motion

The scope trace below shows a setup similar to the one above except with a higher deceleration trajectory resulting in a significant increase in DC supply voltage. Overvoltage conditions are potentially dangerous because they may damage the power supply, motion controller, or motor, or may lead to instability of the motion controller's servo control function.



Overlay + 500/div − Points Time Axis: cycles

Color	Axis	Variable	Value At Cursor
1: 🟥	1	Active motor command	0
2: 🟨	1	Commanded velocity	0
3: 🟩	1	Bus voltage	0
4: 🟦	1	None (disable variable)	0

Trigger

	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	○ 0 [● 1]
Stop:	Immediate	1	Motion complete	[● 0] ○ 1

Trace period: 30 cycles Trace mode: One time Repeat Trace length: 300 samples

Why does the voltage increase during trajectory deceleration? The reason that the voltage increases during deceleration is that the motor acts like a generator. During deceleration the mechanical energy stored in the rotation of the motor and attached load is converted to electrical energy which must be absorbed somewhere. This negative current output has the effect of increasing the supply voltage because most regulated power supplies have very little capacity to absorb energy from the device they are powering.

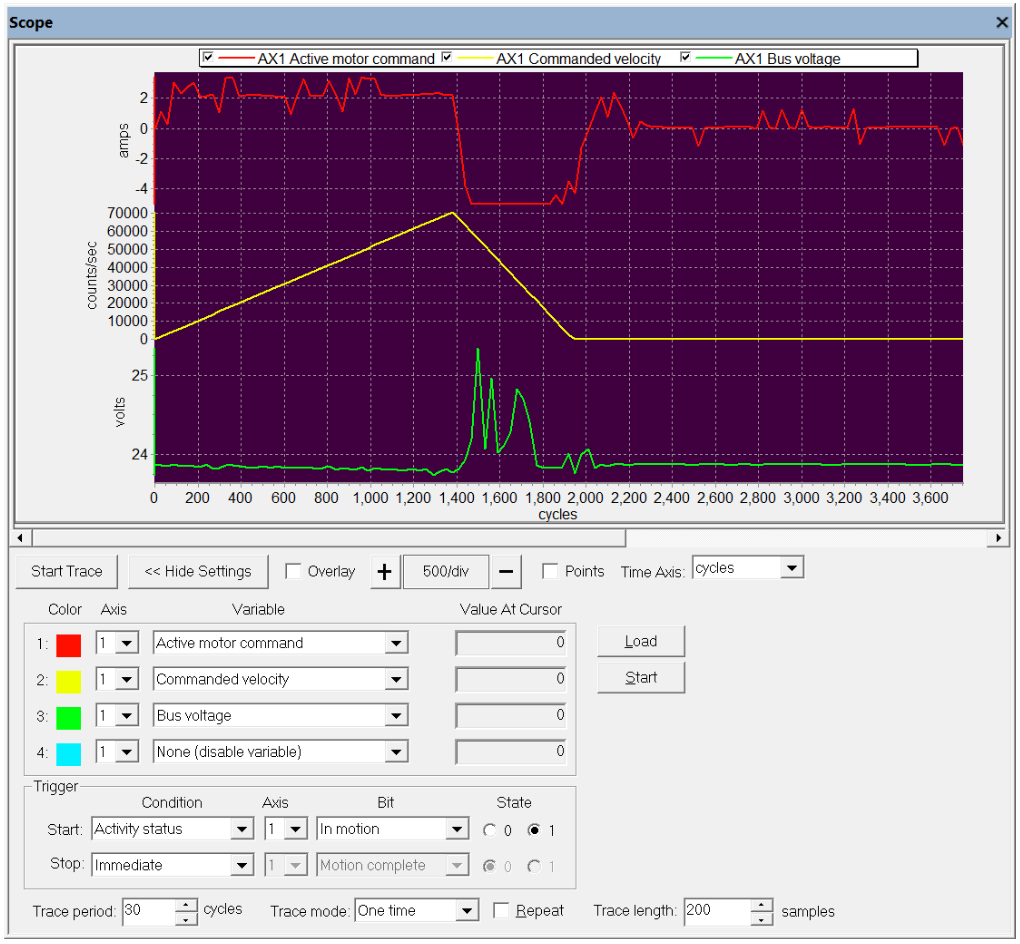
Does trajectory deceleration always generate negative current flow? No. The motor becoming a net generator occurs at higher deceleration rates for motors & loads with larger rotational inertia. An additional factor that affects this is the amount of friction in the system. Predicting when a mechanism will become a net generator is complicated so most engineers exercise the system and directly observe the supply voltage.

What are the remedies to voltage rise during motor acceleration? If an undesirable voltage rise is seen in the deceleration phase of the trajectory the simplest solution is to reduce the deceleration rate. This will reduce or eliminate the power supply needing to absorb current. Another option is to add capacitance to the HV DC supply. This is especially effective to absorb short bursts of current.

A third option is called shunt regulation. Shunt regulation functions by continually comparing the DC bus voltage to a user programmed threshold (usually set to a few volts above the supply voltage) and ‘shunting’ or diverting power directly from HV to GND when the threshold is exceeded. A resistor and diode in the shunt circuit path controls the rate of current flow and ensures that current flow is always in the correct direction. The power shunted through the resistor causes the resistor to heat up, therefore care should be taken to ensure that the resistor and any heat sinking elements it is mounted on can dissipate the worst case heat load.

See Section 6.8, “Shunt Regulation,” for more information on using this PMD controller’s shunt function.

In the trace below the same exact motion profile as above is executed with a shunt function in place with the voltage threshold set for 25.0V. As can be seen shunt does an excellent job of regulating the voltage, lowering the maximum voltage rise from 31V to 25.3V.



☐ Overlay + 500/div – ☐ Points Time Axis: cycles

Color	Axis	Variable	Value At Cursor
1: 🟥	1	Active motor command	0
2: 🟨	1	Commanded velocity	0
3: 🟩	1	Bus voltage	0
4: 🟦	1	None (disable variable)	0

Trigger

	Condition	Axis	Bit	State
Start:	Activity status	1	In motion	0 [✓ 1]
Stop:	Immediate	1	Motion complete	[✓ 0] 1

Trace period: 30 cycles Trace mode: One time ☐ Repeat Trace length: 200 samples

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 Section 6.9, “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, used with some Juno IC DKs and N-Series ION DKs, is susceptible to noise if high current moves are commanded because it uses a UART signal scheme which is not a differential signal. 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 more robust signaling scheme such as RS232, CAN, or Ethernet.

This page intentionally left blank.

4. Communication Port Connections

In This Chapter

- Serial Communications
- CAN Communications
- Ethernet Communications
- SPI 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, Ethernet, and SPI communications.

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 setup.

4.1 Serial Communications

N-Series IONs of serial host interface type support communications in both RS232 and RS485 format. A UART signal level serial port connection is also provided on all N-Series IONs regardless of host interface type.

In RS232 mode two serial ports are supported, Serial1 and Serial2. Serial1 serves as the host serial interface for the ION unit while Serial2 serves as an extra general purpose RS232 port. For detailed information on N-Series ION's serial interfaces refer to the *ION/CME N-Series Digital Drive User Manual*.

In RS485 mode a single serial port is supported at Serial1. RS485 may be operated in either half duplex or full duplex mode. The table below shows which RS485 signals are used in half duplex and full duplex mode.

Pin Name	RS485 Full Duplex Function	RS485 Half Duplex Function
RS485Xmt+	Positive (non-inverting) transmit output	Positive transmit/receive
RS485Xmt-	Negative (inverting) transmit output	Negative transmit/receive
RS485Rcv+	Positive (non-inverting) receive input	not used
RS485Rcv-	Negative (inverting) receive input	not used

By default pin #9 of the ION DK's Indexer Connector (J7) selects whether RS232 or RS485 communications mode is used. If left floating, the ION operates Serial1 and Serial2 in RS232 mode. If tied to ground Serial1 is operated in RS485 mode and Serial2 is not used. Note that a change in the status of this pin will not take effect until after a power on or reset of the ION unit. Whether or not this pin selects the RS232/RS485 mode can be user programmed. See Section 4.1.5, "Setting Up the PMD Controller for RS485 Communications" for details.

The UART serial port is referred to as Serial3 and is generally used for 3-pin programming cable communications with Pro-Motion. But it may be used in an application as a third serial connection if desired. This serial port supports point-to-point serial communications but does not support multi-drop communications.

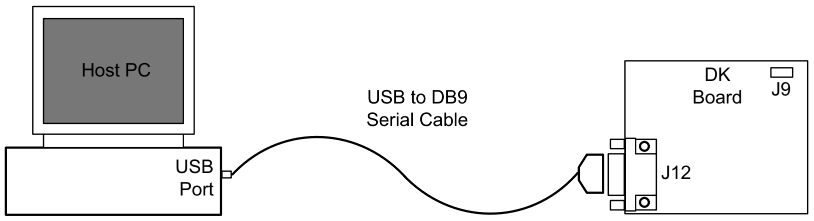
The Serial1, Serial2, and Serial3 ports can be operated at various communication settings as shown in the following table:

Parameter	Range	Serial1 Default	Serial2 Default	Serial3 Default
Baud rate	1,200 to 921,600	57,600	115,200	57,600
Parity	none, even, odd	none	none	none
# Data bits	5, 6, 7, 8	8	8	8
# Stop bits	1, 2	1	1	1

In the following sections we will detail how to connect from a PC to the N-Series ION's serial connections using two different connection schemes; RS232 and RS485.

4.1.1 RS232 Hardware Setup

Figure 4-1: Hardware Setup for Communications via RS232



For your 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 serial host N-Series ION DKs. As shown in Figure 4-1 the DK interconnect board directly accepts this cable at its J12 connector.

With a factory default N-Series ION unit Serial1 is automatically set for RS232 communications. However if this default Serial1 comm setting was changed Section 4.1.5, "Setting Up the PMD Controller for RS485 Communications" provides detailed information on how to set Serial1 for RS232 operation.

Although in this RS232 setup description we will only use Serial1, PMD has available in its accessories library a cable that allows access to both the Serial1 and Serial2 RS232 ports. To order this cable contact your local PMD representative. For convenience the two cables mentioned above are listed in the table below:

Vendor	P/N	Description	Included with Serial DK
PMD	Cable-USB-DB9	Male DB9 to USB serial converted cable.	Yes
PMD	Cable-4355-01.R	Male DB9 dual serial to two single-channel female DB9 cable.	No

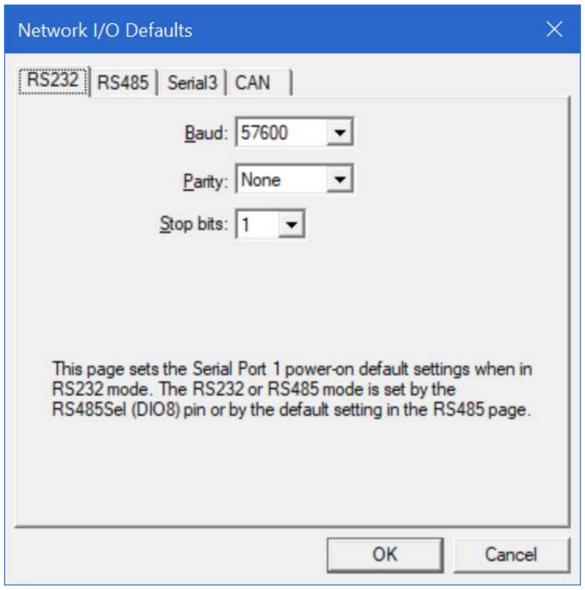
For detailed wiring information on selected DK cables refer to Section 6.5, "Selected Cable & Accessory Specifications."

4.1.2 Setting Up the PMD Controller for RS232 Communications

Here is the Pro-Motion sequence used to set up the PMD controller to connect via its Serial1 RS232 host interface.

1. With serial communications via the 3-pin programming cable functioning properly and with the RS232 Cable-USB-DB9 installed, click the Device toolbar button. The Device window appears.
2. Click Network I/O. The Network I/O Defaults dialog box appears.

3 Click the RS232 tab. This window appears with data entry fields for the serial port setting. This is shown below with default values visible.



Baud: 57600

Parity: None

Stop bits: 1

This page sets the Serial Port 1 power-on default settings when in RS232 mode. The RS232 or RS485 mode is set by the RS485Sel (DIO8) pin or by the default setting in the RS485 page.

4 Enter the desired RS232 settings in the corresponding data fields.

5 Click OK to store as the power on default in the PMD controller's NVRAM.

6 Disconnect 3-pin programming cable 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 is now ready for RS232 communications.

4.1.3 Setting Up Pro-Motion for RS232 Communications

To setup Pro-Motion to communicate by RS232:

1 Click the Connect toolbar button.

2 Select RS232 and then click OK.

3 Enter the same serial port settings as were programmed into the PMD controller earlier.

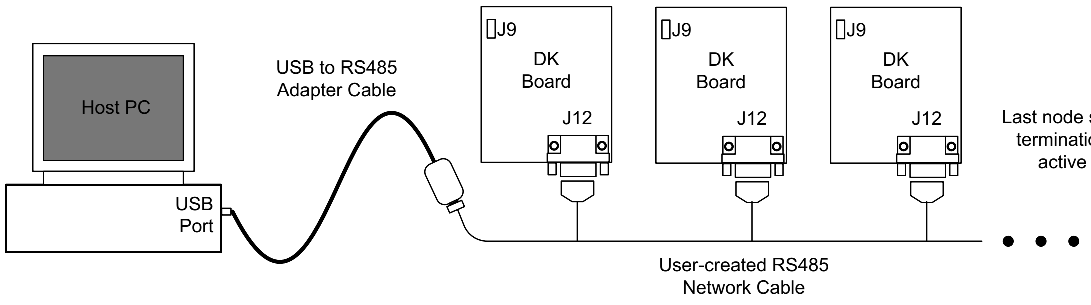
4 When complete, click OK.

If RS232 communication is successful, a set of graphical icons representing the PMD controller and axis will be loaded into the Project window. If communication is not successful after about 2 seconds a Communications Timeout Error dialog box appears. If this happens, recheck your connections, and retry to establish RS232 communications.

You can also re-connect via the 3-pin programming cable and use this connection to help diagnose why the RS232 connection is not working. Once RS232 is up and running most users disconnect and unplug the 3-pin programming connection, but doing so is optional. There may be situations where it is beneficial to have two separate connections between the PMD controller and Pro-Motion, particularly when developing application code.

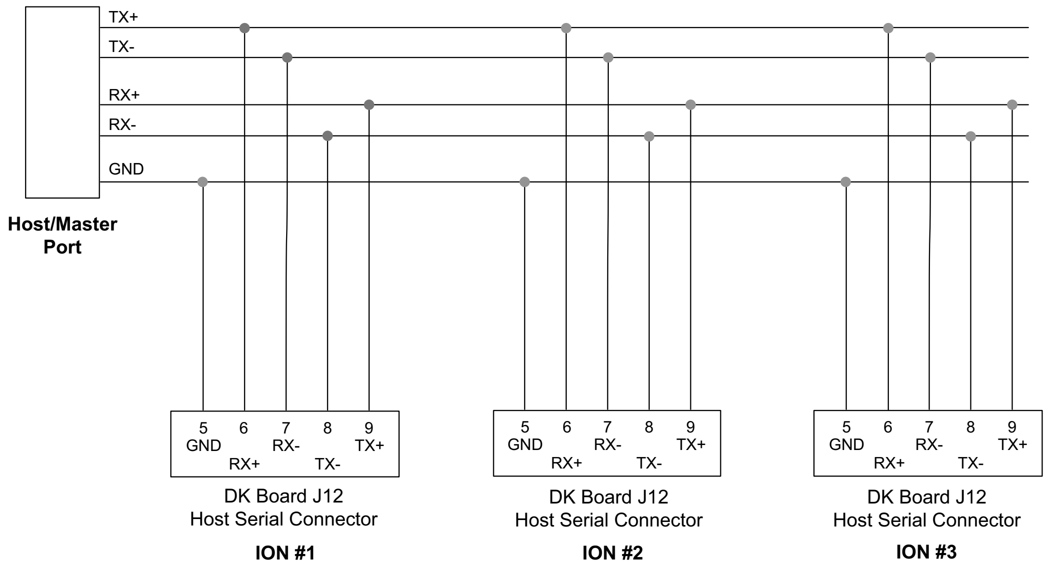
4.1.4 RS485 Hardware Setup

Figure 4-2: Hardware Setup for Communications via RS485


Host PC
USB Port
USB to RS485 Adapter Cable
J9
DK Board
J12
J9
DK Board
J12
J9
DK Board
J12
Last node sets termination active
User-created RS485 Network Cable

Communicating by RS485 is typically less of a 'plug and play' process than communicating by RS232 and requires the user to create their own cables. 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 N-Series IONs.

Figure 4-3: RS485 Signal Connection Diagram


TX+
TX-
RX+
RX-
GND
Host/Master Port
5 GND
6 RX+
7 RX-
8 TX-
9 TX+
DK Board J12 Host Serial Connector
ION #1
DK Board J12 Host Serial Connector
ION #2
DK Board J12 Host Serial Connector
ION #3

RS485 is a master slave system. When connecting to a PC running Pro-Motion the PC functions as the master, and each N-Series ION 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. 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.

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. N-Series IONs provide selectable internal termination of 120 ohm.

To prepare the PC to function as the RS485 master an RS422/485 USB adapter 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 adapter 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 unit to be installed in the RS485 network needs to be programmed with network settings. The table below provides a summary of the available settings:

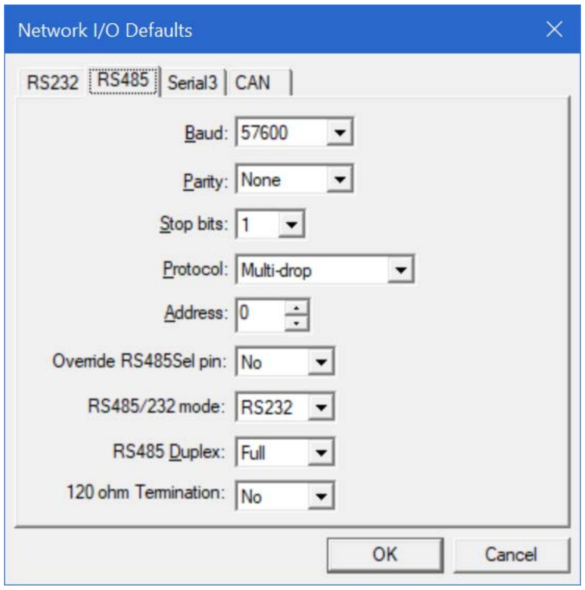
Parameter	Default Setting	Setting Options	Comment
Override RS485Sel pin	no	no yes	If this field is set to "no" the state of Pin 9 of J7, the Indexer Connector will be read to select RS232 or RS485 (floating = RS232, tied to GND = RS485). If this field is set to "yes" the pin value will be ignored and the RS232/RS485 setting will be read from the setting of the next field.
RS232 or RS485	RS232	RS232 RS485	See above. This setting is only used if the 'Override RS485Sel pin' is programmed to "yes".
Address	0	0-31	Each ION unit should be programmed with a different 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.
Termination active	no	no yes	The last ION in the network should have termination active. All other units should have termination inactive.

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

1 With Pro-Motion communications via the 3-pin programming cable functioning properly, click the Device toolbar icon. The Device Control window appears.

2 Click Network I/O. The Network I/O Defaults dialog box appears.

3 Click the RS485 tab. This window appears with data entry fields for the serial port setting such as the baud rate, protocol, address, etc. This is shown below with default values visible.



Baud: 57600
Parity: None
Stop bits: 1
Protocol: Multi-drop
Address: 0
Override RS485Sel pin: No
RS485/232 mode: RS232
RS485 Duplex: Full
120 ohm Termination: No

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 3-pin programming cable communications using the Disconnect toolbar button.

7 Power down the PMD controller.

8 With the PMD controller powered off connect the RS485 cable at the J12 DB9 connector according to Figure 4-3.

9 Repower the PMD controller.

The PMD controller is now ready for operating via RS485.

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 a RS485 cable in place according to Figure 4-3, 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 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 reconnect via the 3-pin programming cable to check the PMD controller's RS485 settings. Connecting via the 3-pin programming cable is described in Section 2.4.1, "Establishing 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 is 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. Each unit on the network must have a unique address and be programmed with the same baud rate.
- 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

N-Series IONs of CAN/SPI host interface type support CAN FD communications through their host CAN port, and all N-series IONs support CAN FD communication through their expansion CAN port. CAN FD supports CANbus 2.0 functionality as a subset and therefore if you plan to use CANbus 2.0 this is supported. For detailed information on N-Series ION's CAN interfaces refer to the *ION/CME N-Series Digital Drive User Manual*.

The CAN interfaces may be operated at various communication rates from 50,000 to 5,000,000 bps (bits per second). In addition, each CAN device is assigned two CAN identifiers (also called addresses); one for transmission of PRP messages, and one for reception of PRP messages. CAN FD supports two separate bit rates known as the nominal bit rate and the data bit rate. This information is summarized in the following table:

Parameter	Range	Default
Nominal bit rate	50,000 to 1,000,000 bps	1,000,000 (host CAN), 1,000,000 (expansion CAN)
Data bit rate	50,000 to 5,000,000 bps	1,000,000 (host CAN), 1,000,000 (expansion CAN)
Host send address	0 – 0x800	0x580
Host receive address	0 – 0x800	0x600

4.2.1 CAN Hardware Setup

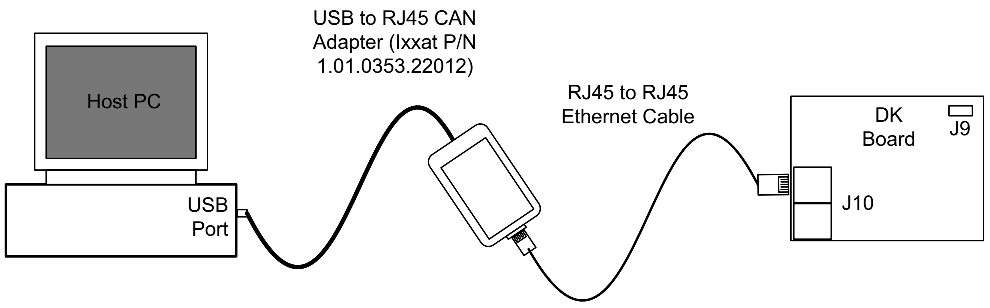

USB to RJ45 CAN Adapter (Ixxat P/N 1.01.0353.22012)
RJ45 to RJ45 Ethernet Cable
Host PC
USB Port
DK Board
J9
J10

Figure 4-4: Hardware Setup for Communicating via CAN

For your PC to communicate via CAN you will typically use a USB to CAN converter which provides an RJ45 interface. The CAN/SPI N-Series ION DK board directly accepts this cable at either J10-1 or J10-2. Either socket can be used because each signal is daisy chained from one connector to the other. This is convenient for creating a CAN based network with multiple ION units installed.

If the CAN connection scheme is a single node network such as shown in Figure 4-4 the N-Series ION DK’s JP4-2 jumper should be installed to provide 120 ohm termination at the DK’s host CAN port. If the network has multiple N-Series ION DKs JP4-2 should be installed on the first unit in the network chain and the last unit in the chain.

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

Vendor	P/N	Description	Included with CAN DK
Ixxat	1.01.0353.22012	USB to CAN FD connector, RJ45 interface	No
CableWholesale	10x8-56101	6' Cat 6 Ethernet connector cable	No

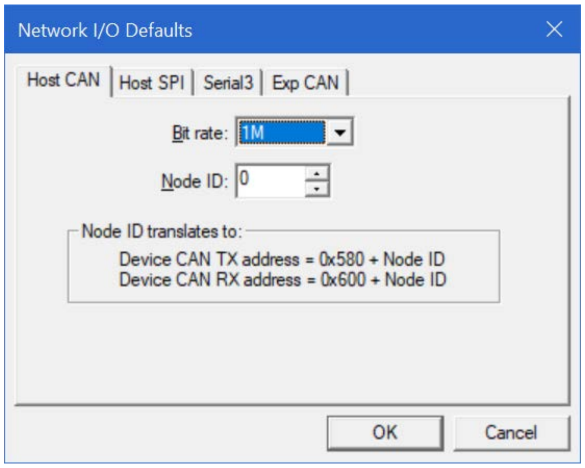
4.2.2 Setting Up the PMD Controller for CAN Communications

Here is the Pro-Motion sequence used to set up the N-Series ION to connect via its CAN host interface.

1 With communications via the 3-pin programming cable functioning properly, click the Device 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.



Bit rate: 1M

Node ID: 0

Node ID translates to:
Device CAN TX address = 0x580 + Node ID
Device CAN RX address = 0x600 + Node ID

4 Enter the desired bit 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 3-pin programming cable 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 N-Series ION is now ready for CAN communications.

4.2.3 Setting Up Pro-Motion For CAN Communications

With a CAN adapter and driver installed in the PC, and with CAN cable connections in place, 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 bit rate and Node ID as was programmed into the PMD controller previously.

4 When complete, click OK.

If CAN communication is successful, an additional set of graphical icons representing your N-Series ION and axis 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 after rechecking your connections communication is still not correctly established you may want to connect via the 3-pin programming connector and check your CAN settings. Connecting via the 3-pin programming cable is described in Section 2.4.1, "Establishing 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. Each unit on the network must have a unique Node ID and be programmed with the same bit 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

N-Series IONs that are of Ethernet host interface type provide 100 Base-T full duplex Ethernet communications. Two different Ethernet protocols are supported; TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP is used for primary Ethernet communications to the ION unit, while UDP is typically used for noncritical applications such as data logging, or for the Pro-Motion console window. See Section 5.4, "C-Motion Engine User Code Development" for more information on the C-Motion Engine Console window.

When used to receive PRP messages the physical node on the Ethernet network controller is assigned a 32-bit IP (Internet Protocol) address, along with a 32-bit netmask and a 32-bit gateway value. The Netmask is used to indicate which IP addresses are local, and the gateway value is used to route non-local addresses. To correctly receive communications from the host controller a 16-bit identifier known as a port must also be specified. To determine what the unused IP addresses are for your Ethernet network, and what values for netmask and gateway to use, you should contact your network administrator.

The table below shows the range and default settings for the Ethernet controller of the N-Series ION:

Parameter	Range	Default
IP Address	0.0.0.0 – 255.255.255.255	192.168.2.2
Netmask	0.0.0.0 – 255.255.255.255	255.255.255.0
Gateway	0.0.0.0 – 255.255.255.255	0.0.0.0
TCP Port	0 – 65,535	40100

Each physical hardware device on an Ethernet network is assigned one IP address, however, a given IP address can have multiple ports. This is useful because it allows user application code running on the C-Motion Engine to open up peripheral connections using port numbers other than the PRP communications port (which has a default value of 40100), thereby allowing PRP messages and application-specific data in any format to co-exist on the same Ethernet IP node.

4.3.1 Ethernet Hardware Setup

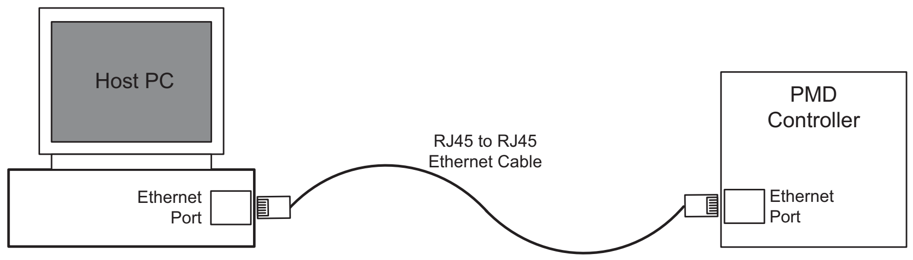


Figure 4-5: Hardware Setup for Communicating via Ethernet

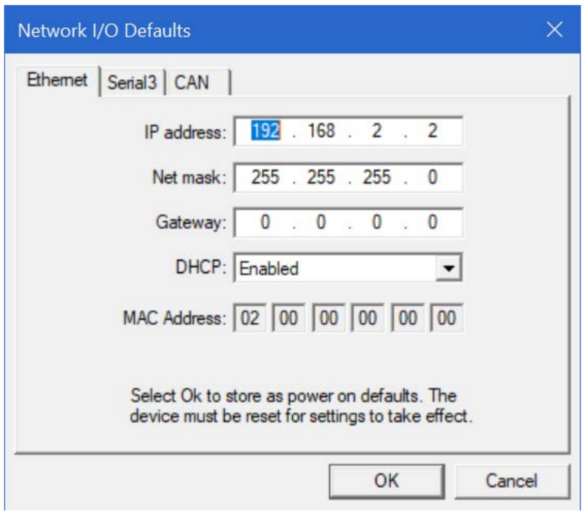
To connect to the N-Series ION via Ethernet you will use the included RJ45 to RJ45 cable which plugs into J13 on the Ethernet N-Series ION DK board. There are two common ways to set your PC up to communicate with a PMD controller via Ethernet. The first is to install an Ethernet NIC (Network Interface 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 to set up the N-Series ION to connect via its Ethernet host interface.

1. With communications via the 3-pin programming cable functioning properly, click the Device 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, Net Mask, and Gateway. This is shown below with default values visible.



IP address: 192 . 168 . 2 . 2
Net mask: 255 . 255 . 255 . 0
Gateway: 0 . 0 . 0 . 0
DHCP: Enabled
MAC Address: 02 00 00 00 00 00

Select Ok to store as power on defaults. The device must be reset for settings to take effect.

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 3-pin programming cable communications 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 is now 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 3-pin programming cable to check or change the Ethernet settings. Connecting via the 3-pin programming cable is described in Section 2.4.1, "Establishing 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 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 SPI Communications

N-Series IONs of CAN/SPI host interface type support SPI (Serial Peripheral Interface) communications through their host SPI port, and all N-series IONs support SPI communications through their expansion SPI port. For more information on the N-Series ION's SPI interface refer to the *ION/CME N-Series Digital Drive User Manual*.

4.4.1 SPI Hardware Setup

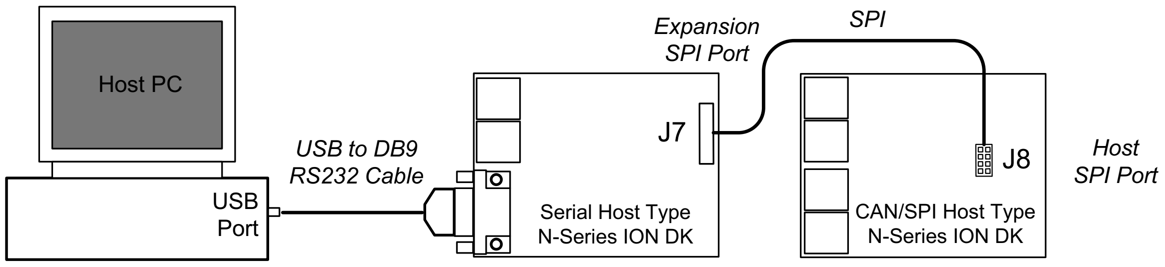


Figure 4-6: Hardware Setup for SPI Communications via N-Series ION Drive

Communicating by SPI between the PC and a PMD controller may be useful during code development, or as a communications hardware reference when developing a user-designed board that will use SPI to communicate to the PMD controller.

The figure above shows the recommended approach to achieving this which uses a PC-connected N-Series ION drive which in turn connects to an SPI host type N-Series ION via SPI. In this scheme the PC-connected N-Series ION acts as a communications bridge and is not used to control a motor. This PC to N-Series ION connection can be serial, CAN, or Ethernet, but for the purpose of the instructions below we will assume a serial host type N-Series ION is used. There are three available serial host N-Series ION developer kits, PMD part numbers DK481S0056/02, DK481S0056/06, and DK481S0056/18 which differ only in their power output level and therefore will all work equally well.

Communicating by SPI between the PC and a PMD controller is not recommended in the production application due to SPI's relative lack of noise immunity when connected via cables.

4.4.2 PC-Connected N-Series ION DK to CAN/SPI N-Series ION DK Connections

In the connection setup shown in Figure 4-6 the PC-connected N-series ION will use its expansion SPI port, functioning as an SPI master, to communicate with the CAN/SPI host type N-Series ION functioning as an SPI slave. Although only one CAN/SPI host type N-Series ION is shown, up to four separate Enable signals are provided by the N-Series ION expansion SPI port therefore up to four CAN/SPI host type N-Series IONs could be connected to via SPI in this same way.

The table below shows the SPI bus connections between the PC-connected N-Series ION DK and the CAN/SPI N-Series ION DK.

PC-Connected N-Series ION DK Signal Name	J7 Connector Pin #	CAN/SPI N-Series ION DK Signal Name	J8 Connector Pin #
ExpSPIXmt	2	HostSPIRcv	2
ExpSPIRcv	3	HostSPIXmt	1
ExpSPIClock	4	HostSPIClock	3
EXPSPICS1	5	HostSPIEnable	4
ExpSPIStatus	8	HostSPIStatus	6
GND	10	GND	7

All of the connections in the table above are made via terminal screws. The assembled connection wires will function best if they are as short as possible and encased in shielding.

4.4.3 Setting-Up Pro-Motion for Communications with the PC-Connected N-Series ION

Here is the sequence used to set up Pro-Motion for communications with the PC-connected N-Series ION.

1. With the RS232 USB to DB9 cable installed in the N-Series ION DK and power applied, click the Connect toolbar icon in Pro-Motion.
2. Click Serial, select the com port that the N-Series ION DK 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.
3. Click OK without changing any of these settings. If serial communication is correctly established, a set of object graphics for the N-Series ION you are configuring will load into the Project window.

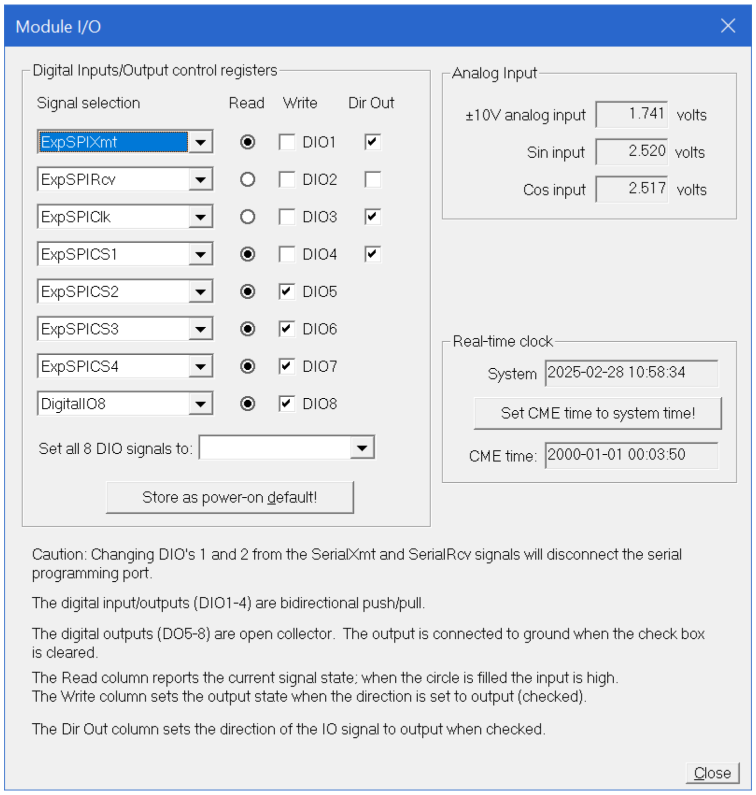
Note that the Serial1 RS232 USB to DB9 cable (P/N Cable-USB-DB9) is used here rather than the Serial3 3-pin programmer cable. This is because Srl3Xmt and Srl3Rcv, which are the signals used for Serial3 communications, also carry the expansion SPI port signals that will be needed to connect with the SPI host-type N-Series ION DK. Use of the Serial1 RS232 connection avoids this issue.

In the above instructions we have assumed the default serial connection parameters of the N-Series ION are adequate. To change the connection parameters such as the baud rate, you will need to change the serial settings in both the N-Series ION unit and Pro-Motion. Refer to the Section 4.1, "Serial Communications" for instructions on doing this.

4.4.4 Changing N-Series ION Pin Function Settings

Next, some of the PC-connected N-Series ION pin function settings will be changed to enable its expansion SPI port to communicate with the CAN/SPI N-Series ION DK. These settings are known as pin MUX settings.

1 Click the Device toolbar icon and select the I/O box from the Device Control window. A dialog will appear as shown below.



Module I/O

Digital Inputs/Output control registers

Signal selection	Read	Write		Dir Out
ExpSPIXmt	●	☐	DIO1	✓
ExpSPIRcv	○	☐	DIO2	☐
ExpSPIClk	○	☐	DIO3	✓
ExpSPICS1	●	☐	DIO4	✓
ExpSPICS2	●	✓	DIO5	
ExpSPICS3	●	✓	DIO6	
ExpSPICS4	●	✓	DIO7	
DigitalIO8	●	✓	DIO8	

Set all 8 DIO signals to:

Analog Input

±10V analog input 1.741 volts
Sin input 2.520 volts
Cos input 2.517 volts

Real-time clock

System 2025-02-28 10:58:34
CME time: 2000-01-01 00:03:50

Caution: Changing DIO's 1 and 2 from the SerialXmt and SerialRcv signals will disconnect the serial programming port.

The digital input/outputs (DIO1-4) are bidirectional push/pull.

The digital outputs (DO5-8) are open collector. The output is connected to ground when the check box is cleared.

The Read column reports the current signal state; when the circle is filled the input is high.
The Write column sets the output state when the direction is set to output (checked).

The Dir Out column sets the direction of the IO signal to output when checked.

2 Select "Set all 8 DIO signals to" and set to Expansion SPI.

3 Click on "Store as power-on default!" and then click on Close.

4 Disconnect Serial1 RS232 communications using the Disconnect toolbar icon.

5 Power down the PC-connected N-Series ION.

4.4.5 Connecting to the CAN/SPI N-Series ION

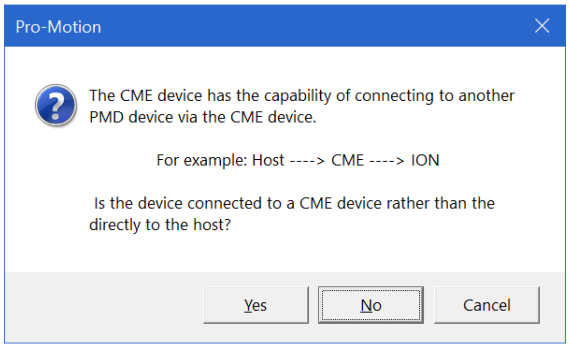
Next, the PC-connected N-Series ION's expansion SPI port is connected to the CAN/SPI N-Series ION. Note that in the procedure below no communication parameter settings need to be programmed into the CAN/SPI host type N-Series ION. This is because it operates as a slave, and user-selectable control settings such the SPI bus clock rate is programmed into the master, which in this scheme is the PC-connected N-Series ION.

Here are the instructions to connect to the CAN/SPI N-Series ION via the PC-connected N-Series ION's expansion SPI port:

1 With the USB to DB9 cable between the PC and the PC-connected N-Series ION developer kit installed and the two N-Series ION DKs connected via SPI as detailed in Section 4.4.2, "PC-Connected N-Series ION DK to CAN/SPI N-Series ION DK Connections," power on both N-Series ION DKs.

2 Connect to the PC-connected N-Series ION by executing steps 1 through 3 in Section 4.4.3, "Setting-Up Pro-Motion for Communications with the PC-Connected N-Series ION."

3 Click the Connect toolbar icon. A dialog box will appear as shown below indicating that since you are already connected (via serial) to the PC-connected N-Series ION you have the option of connecting via an expansion port. Click Yes.

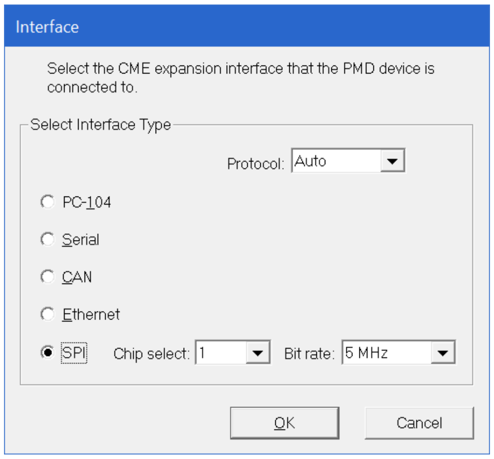

Pro-Motion

The CME device has the capability of connecting to another PMD device via the CME device.

For example: Host ----> CME ----> ION

Is the device connected to a CME device rather than the directly to the host?

4 A dialog box will appear as shown below. Select SPI, set the protocol to Magellan, set Chip Select to 1, and specify a bit rate of 5 MHz. Then click OK.


Interface

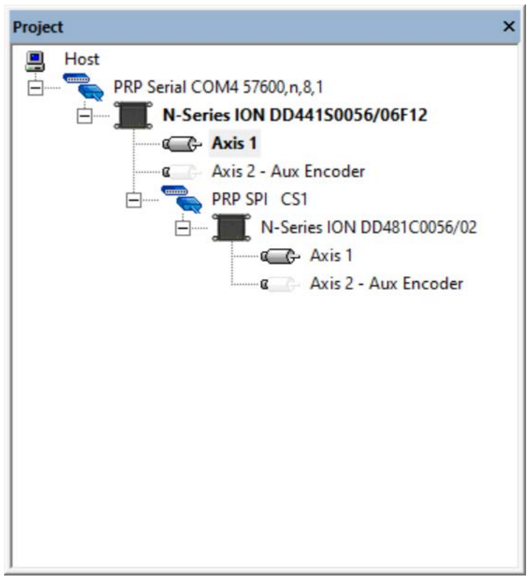
Select the CME expansion interface that the PMD device is connected to.

Select Interface Type

Protocol: Auto

○ PC-104
○ Serial
○ CAN
○ Ethernet
● SPI Chip select: 1 Bit rate: 5 MHz

If SPI communication is successfully established a set of graphical icons representing the CAN/SPI host type N-Series ION will be loaded into the Project window as shown below. 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 SPI communications.


Project

Host
PRP Serial COM4 57600,n,8,1
N-Series ION DD441S0056/06F12
Axis 1
Axis 2 - Aux Encoder
PRP SPI CS1
N-Series ION DD481C0056/02
Axis 1
Axis 2 - Aux Encoder

4.5 Communicating With Attached Devices

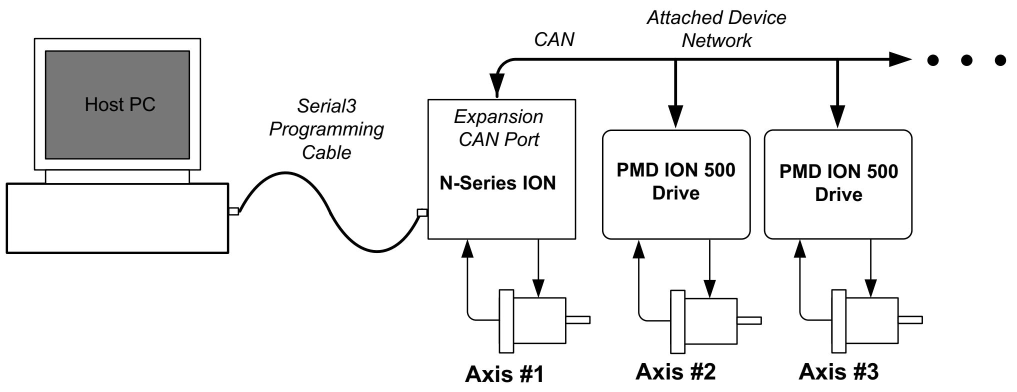


Figure 4-7: PMD Controller with Attached Device Network

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 controller which in turn is connected to the PC. In the case of N-Series ION drives two different attached device networks are supported, CAN FD and SPI.

In this section 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 drives with CANbus host interface. ION 500 drives are intelligent single axis drives for DC Brush, Brushless DC, or step motors. The choice of this particular product is arbitrary. Any of PMD's products that connect via CAN and support either a Magellan or PRP packet protocol including N-Series IONs, MC58113 ICs, MC58420 ICs, and others could also have been used.

The devices on the attached network can all be of the same PMD product type, they can be of different types, and they can even be non-PMD devices, although if this is the case they can not be connected via Pro-Motion because Pro-Motion expects attached devices to support either PMD's Magellan packet protocol or PMD's PRP packet protocol.

The choice of a CAN network versus an SPI attached device network is also arbitrary, although SPI is generally not recommended for communicating with cable-connected controllers. On a single PCB substrate however an N-Series ION acting as an attached device network master communicating to SPI devices also on the same PCB is a common and popular use of attached device networks.

4.5.1 Setting Up the CAN Attached Device PMD Controllers

Each ION 500 unit on the attached device network should be programmed with the network bit 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. The instructions for programming each CAN ION 500 unit to be connected on the PMD controller's attached device network is detailed in the *ION Digital Drive User Manual*.

4.5.2 Setting Up Pro-Motion for CAN Attached Network Communications

Setting up Pro-Motion for CAN 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 serial host-type N-Series ION connected via the 3-pin programming cable.

1 Connect the 3-pin programming cable to the N-Series ION DK.

2 Connect the ION 500 unit at its host CAN connector to the N-Series ION expansion CAN network connector (J11) 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 N-Series ION.

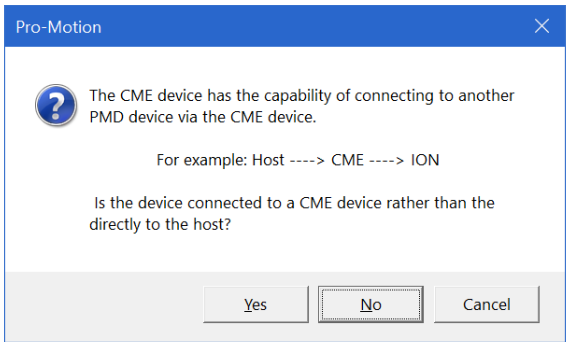
3 Power up the N-Series ION and the ION 500.

4 Click the Connect toolbar button.

5 Click Serial, select the com port that the N-Series ION 3-pin programming cable 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 serial communication is correctly established, a graphic object for the N-Series ION will load into the Project window.

7 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 N-Series ION this connection will be for an attached device network.

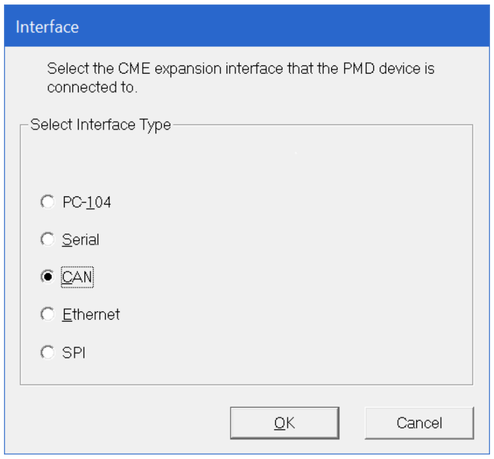


The CME device has the capability of connecting to another PMD device via the CME device.

For example: Host ----> CME ----> ION

Is the device connected to a CME device rather than the directly to the host?

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



Select the CME expansion interface that the PMD device is connected to.

Select Interface Type

PC-104
Serial
[✓ CAN]
Ethernet
SPI

9 Select CAN, and then click OK.

10 Enter the same bit rate and Node ID as was programmed into the ION 500 unit previously.

11 When complete, click OK.

If CAN communication is successful, a set of graphical icons representing the attached device network ION 500 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.

4.5.3 Setting Up Multiple ION Units On the Attached Device Network

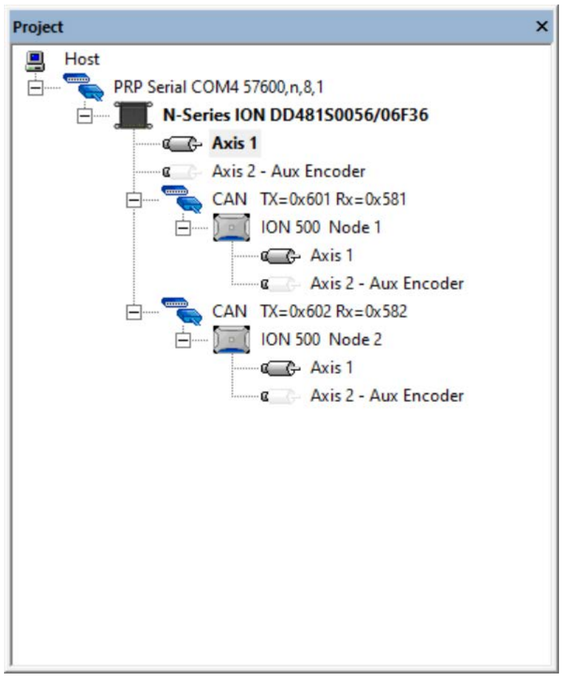
Section 4.5.1 and Section 4.5.2 show how to program a single ION 500 and configure Pro-Motion so that it communicates on the attached device network. This is useful to verify that the CAN network cabling and communication settings are correct.

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 4.5.1, "Setting Up the CAN Attached Device PMD Controllers." Each unit on the network must have a unique Node ID and be programmed with the same bit rate.
- Install all of the 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 500 on the network. You may use the included TRM-RJ45-02 device for this purpose. To order more TRM-RJ45-02 devices contact your local PMD representative.

- Follow the instructions in Section 4.5.2, "Setting Up Pro-Motion for CAN Attached Network Communications" step1 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.


Project
Host
PRP Serial COM4 57600,n,8,1
N-Series ION DD481S0056/06F36
Axis 1
Axis 2 - Aux Encoder
CAN TX=0x601 Rx=0x581
ION 500 Node 1
Axis 1
Axis 2 - Aux Encoder
CAN TX=0x602 Rx=0x582
ION 500 Node 2
Axis 1
Axis 2 - Aux Encoder

This motion setup consists of an N-Series ION which hosts two CAN ION 500 drives on an attached device network. The Pro-Motion user can address any of these axes 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 N-Series ION's attached device network.

The ability of PMD controllers to provide this seamless 'pass through' communications capability is a powerful feature of PMD products generally, and 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.

This page intentionally left blank.

5. Software Development

In This Chapter

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 Visual Basic.

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:

- Provided as source code, allowing easy compilation & porting onto various run-time environments including PCs, user-designed boards, or C-Motion Engines
- 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

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 may be 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	**User-designed board.** 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	**Host-connected user-designed board.** 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	**Host-connected ION drive or Prodigy board.** 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	**ION/CME drive or Prodigy/CME board.** 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 MC58113 Family ICs & DKs	Magellan	C-Motion Magellan, C-Motion PRP*
Magellan MC5x000 Family ICs & DKs	Magellan	C-Motion Magellan, C-Motion PRP*
Juno MC78113 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
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 N-Series ION drives is C-Motion PRP II. PRP stands for PMD Resource Access Protocol, which is both a packet protocol and an associated controller architecture. While the details of how PRP packets are formatted is not important for most C-Motion PRP users, a general understanding of PRP device 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.

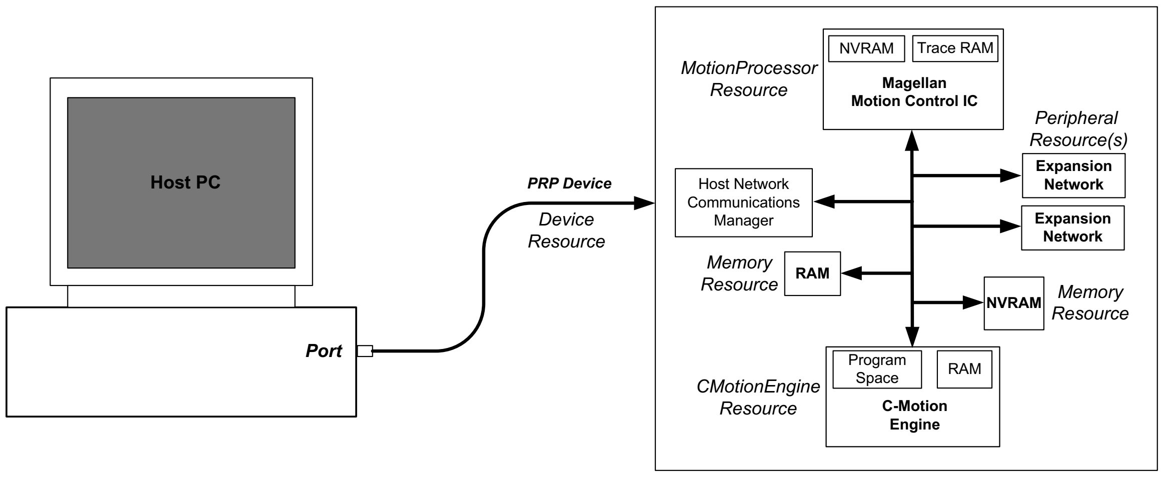
After a PRP communication is sent to a device a return communication is sent by the PRP device which consists of a response header and an optional return message body. The return message body may contain information associated with the requested PRP command, or it may contain error information if there 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 drives 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 devices on it, all ten connections between the PRP device and the attached CAN network devices would have a 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 library 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 to Magellan Motion Control ICs on a PRP device. Accessing the MotionProcessor resource is similar to the procedure detailed above for Peripheral resources, 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.

There are numerous callable MotionProcessor 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 application 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 drives. Running on the PC is also the most popular approach when coding in C# and other .NET languages.

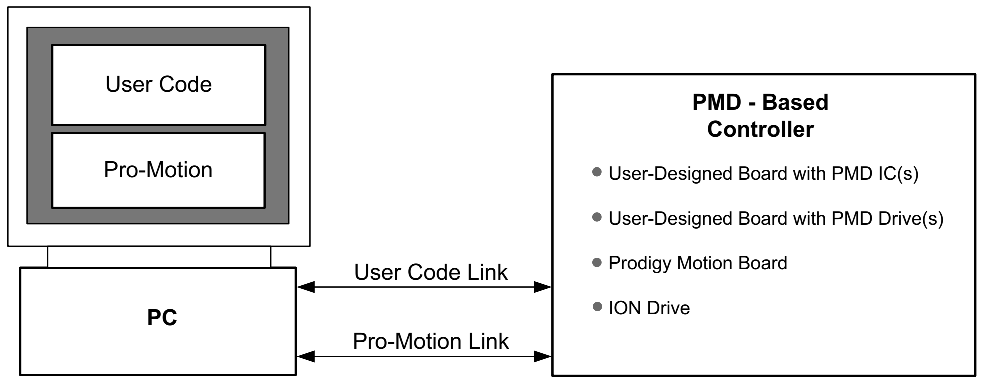


Figure 5-2: Typical Connection Scheme for PC-Based User Code Development

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 user application code sequences. 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 PC Link Options for N-Series IONs

The communication link combination options that are available for different types of PMD controllers vary. The table below shows link combination options for User Code and Pro-Motion links when developing PC-based code to control an N-Series ION drive.

There are three different host interface types for N-Series ION drives (serial, Ethernet, CAN/SPI) therefore the link options are given separately for each:

User Code Link	Pro-Motion Link
Serial Host Type N-Series ION	
Serial1 RS232 or RS485	Serial3 3-pin programming port
Serial1 RS232 or RS485	Expansion CAN FD*
Ethernet Host Type N-Series ION	
Ethernet TCP port 40100	Serial3 3-pin programming port
Ethernet TCP port 40100	Expansion CAN FD*
CAN/SPI Host Type N-Series ION	
Host CAN FD	Serial3 3-pin programming port
Host CAN FD**	Expansion CAN FD*

**Connecting Pro-Motion to the expansion CAN port rather than the host CAN port is done by plugging the CAN cable into the N-Series ION's expansion CAN connector (J11) rather than the host CAN connector.*

***Note that in this combination which uses two separate CAN networks, two separate USB to CAN converter cables are needed; one for the User Code link and one for the Pro-Motion link.*

See Chapter 4, *Communication Port Connections*, for information on setting up Pro-Motion to PMD controller connections using the above-listed link types.

5.3.2 Choosing the SDK For PC-Based User Code Development

In most setups where the host code will run on the PC 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 the 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'.

Conversely, a good 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.

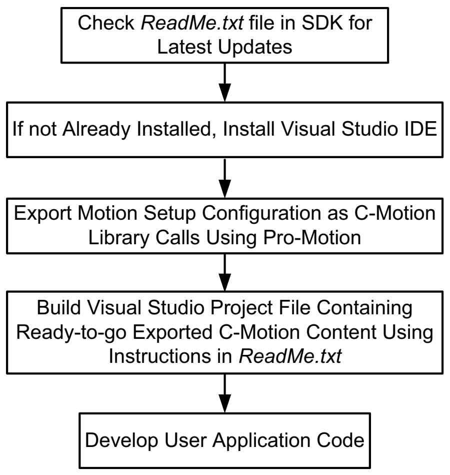

Check *ReadMe.txt* file in SDK for Latest Updates
If not Already Installed, Install Visual Studio IDE
Export Motion Setup Configuration as C-Motion Library Calls Using Pro-Motion
Build Visual Studio Project File Containing Ready-to-go Exported C-Motion Content Using Instructions in *ReadMe.txt*
Develop User Application Code

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 II refer to *C-Motion PRP II Programming Reference*. For detailed information on Magellan Motion Control IC commands refer to the *C-Motion Magellan Programming Reference*. For 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 section provides information on how to develop user code that runs on the C-Motion Engine.

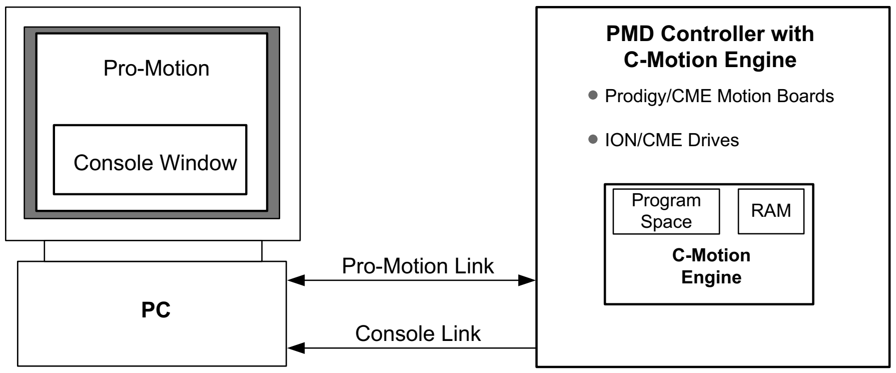


Figure 5-4: Typical Connection Links When User Code Runs on C-Motion Engine

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 between the PC and the PMD controller:

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. It directly transmits whatever ASCII messages are sent using printf commands by the C-Motion Engine user code.

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

5.4.1 Link Options for the N-Series ION

The communication link combination options that are available for different types of PMD controllers vary. The table below shows typical link combination options for for Pro-Motion and Console links when developing code that is running on the N-Series ION's C-Motion Engine

There are three different host interface types for N-Series ION drives (serial, Ethernet, CAN/SPI) therefore the link options are given separately for each:

Pro-Motion Link	Console Link
Serial Host Type N-Series ION	
Serial1 RS232	Serial2 RS232
Serial1 RS232	Internal PRP messaging*
Serial1 RS485	Internal PRP messaging*
Expansion CAN FD**	Serial2 RS232
Expansion CAN FD**	Internal PRP messaging*
Serial3 programming port	Internal PRP messaging*
Ethernet Host Type N-Series ION	
Ethernet TCP port 40100	Ethernet UDP
Expansion CAN FD**	Internal PRP messaging*
Serial3 programming port	Internal PRP messaging*

Pro-Motion Link	Console Link
CAN/SPI Host Type N-Series ION	
Host CAN FD	Internal PRP messaging*
Expansion CAN FD**	Internal PRP messaging*
Serial3 programming port	Internal PRP messaging*

**Internal PRP messaging means that console message traffic is carried via PRP packets and intermixed with regular Pro-Motion to PMD controller traffic carried on the Pro-Motion link.*

***Connecting Pro-Motion to the expansion CAN is achieved by plugging the CAN cable into the N-Series ION's expansion CAN connector (J11) rather than the host CAN connector.*

From the table above there are often several options for connecting between Pro-Motion and the PMD controller. While it is perhaps easiest to always connect Pro-Motion via the PMD controller's primary host interface using an alternative channel may be necessary if the application itself uses the primary host interface. For example if the PMD controller will receive and process RS232-based external commands from a factory-floor interface on the Serial1 port, a CAN connection can be used for the Pro-Motion link or vice-versa.

Most users will use the Pro-Motion Console window to display the stream of messages sent by the C-Motion Engine to the console link channel. However some users may prefer to direct the console data stream to a separate terminal or PC-based terminal emulator rather than to Pro-Motion. If this is the case using the Serial2 RS232 link may be the best choice since RS232 is readily accepted by a broad range of data capture devices. Refer to Section 6.5, "Selected Cable & Accessory Specifications," for information on a cable that provides connection to the Serial2 RS232 channel.

To program where the PMD controller directs console traffic output select the Console box in the Device Control window. To program where the Pro-Motion Console window receives its data right-click from within the Console window and select Connect.

5.4.2 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.

5.4.3 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.

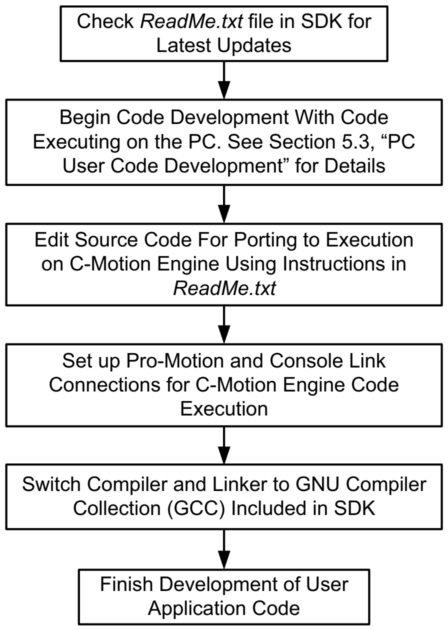

Check *ReadMe.txt* file in SDK for Latest Updates
Begin Code Development With Code Executing on the PC. See Section 5.3, “PC User Code Development” for Details
Edit Source Code For Porting to Execution on C-Motion Engine Using Instructions in *ReadMe.txt*
Set up Pro-Motion and Console Link Connections for C-Motion Engine Code Execution
Switch Compiler and Linker to GNU Compiler Collection (GCC) Included in SDK
Finish Development of User Application Code

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.

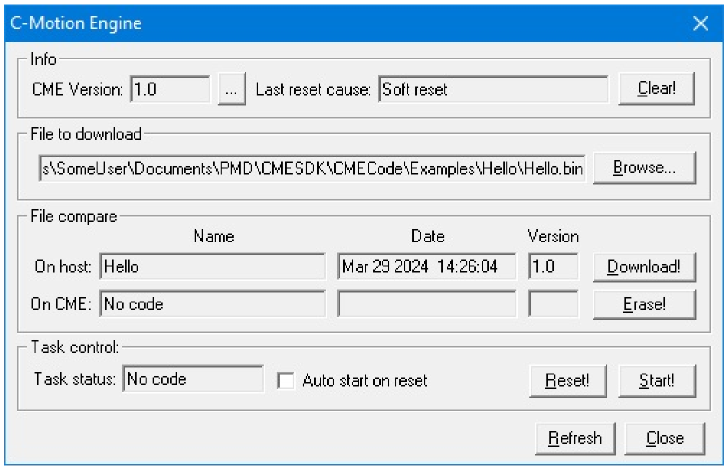
At some point however you will be ready to port the code to run on the C-Motion Engine. The timing of this switch may be influenced by several factors but a particularly important one is the fact that some features (particularly features provided by CMotion PRP II which is the SDK used with the N-Series 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 5.4.1, “Link Options for the N-Series ION.”

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.

This page intentionally left blank.

6. Reference

In This Chapter

6.1 Connector Reference

Figure 6-1 shows the location of various components for the three different N-Series ION DK interconnect boards which differ in the host interface type; serial, CAN/SPI, and Ethernet.

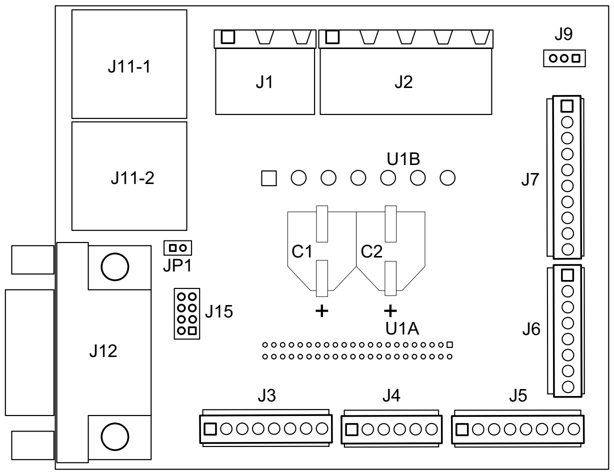


Figure 6-1: Serial Host Interface DK Interconnect Board

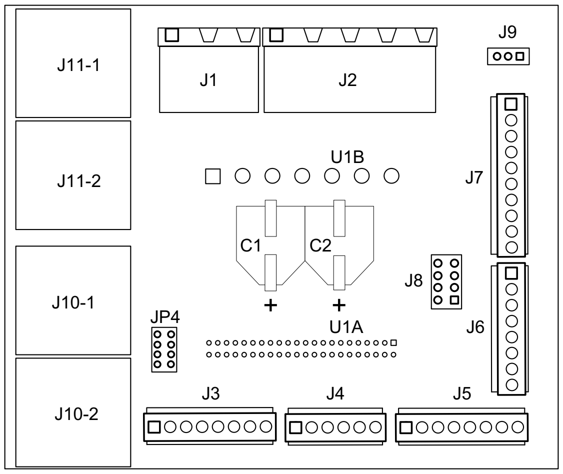


Figure 6-2: CAN/SPI Host Interface DK Interconnect Board

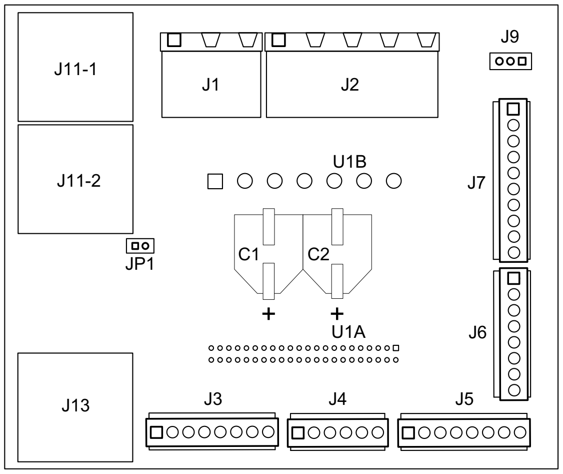


Figure 6-3: Ethernet Host Interface DK Interconnect Board

The following table identifies these components:

Label	Description
J1	HV Power Connector
J2	Motor Drive Connector
J3	Feedback Connector
J4	Hall Signals Connector
J5	Auxiliary Connector
J6	Motion Signals Connector
J7	Indexer Connector
J8	Host SPI Connector (CAN/SPI DK version only)
J9	Programming Connector
J10	Host CAN Connector (CAN/SPI DK version only)
J11	Expansion CAN Connector
J12	Host Serial Connector (Serial DK version only)
J13	Host Ethernet Connector (Ethernet DK version only)
J15	Host Serial Header Connector (Serial DK version only)
JP1	Serial & Ethernet board jumper
JP4	CAN/SPI board jumpers

6.1.1 Feedback Connector (J3)

The following table details the Feedback Connector (J3), which is an 8-pin terminal screw style connector.

Pin #	Signal Name	Description
		J3 - Feedback Connector
1	+5V	+5V power output which may be used to power the motor's encoder circuitry
2	GND	This is the preferred ground connection for the quadrature and Index signal inputs
3	QuadA1+/Cos+*	Differential A+ or Cos+ encoder input for primary axis. *Optional for step motors.*
4	QuadA1-/Cos-*	Differential A- or Cos- encoder input for primary axis. *Optional for step motors.*
5	QuadB1+/Sin+*	Differential B+ or Sin+ encoder input for primary axis. *Optional for step motors.*
6	QuadB1-/Sin-*	Differential B- or Sin- encoder input for primary axis. *Optional for step motors.*
7	Index1+/BiSS-Clock+	Differential Index+ or BiSSClock+ encoder connection for primary axis. *Optional for step motors.*
8	Index1-/BiSSClock-	Differential Index- or BiSSClock- encoder connection for primary axis. *Optional for step motors.*

** Quadrature encoder type is the default setting. For information on how to change encoder types refer to the ION/CME N-Series Digital Drive User Manual.*

6.1.1.1 Single-ended Encoder Connections

Encoder inputs may be connected differentially, with two wires per signal (as shown in the table above), or with just one wire per signal. If single-ended encoders are used, connect encoder signals to the positive encoder input only. The negative input may remain unconnected.

6.1.2 Motion Signals Connector (J6)

The following table details the Motion Signals Connector (J6). This connector is an 8-pin terminal screw style connector.

Pin #	Signal Name	Description
		J6 - Motion Signal Connector
1	PosLim	Positive position limit input (optional)
2	Neglim	Negative position limit input (optional)
3	Home	Home signal input (optional)
4	Enable	Enable input signal
5	FaultOut	FaultOut signal output
6	Reset	Reset signal input (optional)
7	Brake	Brake signal input (optional)
8	GND	Digital ground

6.1.2.1 Enabling the Board

N-Series IONs require an active *Enable* signal to operate. To accomplish this the Motion Signals Connector (J6) is used. Connect terminal #4 of J6 (indicated on the board as En) to terminal #8 of the same terminal screw connector (indicated on the board as GND) using a short wire.

6.1.3 Hall Signals Connector (J4)

The following table details the Hall Signals Connector (J4). This connector is a 6-pin terminal screw style connector.

Pin #	Signal Name	Description
		J4 -Hall Signal Connector
1	HallA	Hall Sensor A Input

2	HallB	Hall Sensor B Input
3	HallC	Hall Sensor C Input
4	GND	Digital ground
5	AnalogIn +	Positive differential signal of the general purpose analog input
6	AnalogIn-	Negative differential signal of the general purpose analog input

6.1.4 Programming Connector (J9)

The following table details the Programming Connector (J9). This connector is a 3-pin 2 mm header.

Pin #	Signal Name	Description
		J9 - Programming Connector
1	Srl3Xmt	Serial programming port transmit signal
2	Srl3Rcv	Serial programming port receive signal
3	GND	Ground

6.1.5 Power Connector (J1)

The following table details the Power Connector (J1).

The Power Connector is a Würth Elektronik 3 Position Terminal Block Header P/N 691313710003.

Pin #	Signal Name	Description
		J1 - Power Connector
1	HV Aux	Positive motor voltage power used to drive N-Series ION's internal logic
2	HV	Positive motor voltage power used to drive the motor
3	GND	Motor voltage power ground

The HV Aux and HV pins are normally tied together but may be kept separate.

6.1.6 Motor Drive Connector (J2)

The following table details the motor drive connector (J2). There are four motor drive connections and a shield connection. Not every motor type uses all four drive connections.

The J2 Motor Drive Connector is a Würth Elektronik 5 Position Terminal Block Header P/N 691313710005.

Pin #	Signal Name	Description
		J2 - Motor Drive Connector
1	Motor A	A motor drive lead. Used with all motor types.
2	Motor B	B motor drive lead. Used with all motor types
3	Motor C	C motor drive lead. Used with all motor types except DC Brush
4	Motor D/Shunt	D motor drive lead used with step motors only, or shunt output used with DC Brush or Brushless DC motors.
5	Case/shield	Connection to motor case/shield. A shield connection is strongly recommended for most motor setups. This pin is connected to the N-Series ION's Power Connector GND signal.

The table below shows which leads should be connected for each supported motor type:

Motor Type	DK Board Pin # and Name	Motor Coil Connections
Brushless DC	1, Motor A	A winding connection
	2, Motor B	B winding connection
	3, Motor C	C winding connection
	5, Case/shield	(optional) motor shield connection
DC Brush	1, Motor A	+ winding connection
	2, Motor B	- winding connection
	5, Case/shield	(optional) motor shield connection
Step motor	1, Motor A	phase A+ winding connection
	2, Motor B	phase A- winding connection
	3, Motor C	phase B+ winding connection
	4, Motor D	phase B- winding connection
	5, Case/shield	(optional) shield connection

Shield connections to the motor are strongly recommended. Not connecting the shield signal may result in increased EMI (electromagnetic interference), reduced immunity to ESD (electrostatic discharge), or electrical noise resulting in motor operation failure.

6.1.7 Auxiliary Connector (J5)

The following table details the Auxiliary Connector (J5). This connector is an 8-pin terminal screw style connector.

Pin #	Signal Name	Description
		J5 - Auxiliary Connector
1	+5V	+5V power output which may be used to power the encoder
2	GND	Ground
3	QuadA2+	Auxiliary encoder differential A+ quadrature input
4	QuadA2-	Auxiliary encoder differential A- quadrature input
5	QuadB2+	Auxiliary encoder differential B+ quadrature input
6	QuadB2-	Auxiliary encoder differential B- quadrature input
7	Index2+/BiSS-Data+*	Differential Index+ or BiSSData+ encoder connection.
8	Index2-/BiSSData-*	Differential Index- or BiSSData- encoder connection.

** Quadrature encoder type is the default setting for both the primary and auxiliary axis. For information on how to change encoder types refer to the ION/CME N-Series Digital Drive User Manual.*

6.1.8 Indexer Connector (J7)

The following table details the Indexer Connector (J7). This connector is an 10-pin terminal screw style connector.

Pin #	Signal Name	Description
		J7 - Indexer Connector
1	GND	Ground
2	DigitalIO1/Srl3Xmt/ExpSPIXmt	Digital I/O bit 1, Srl3Xmt, or ExpSPIXmt signal depending on pin MUX setting. Srl3Xmt is the default functionality.
3	DigitalIO2/Srl3Rcv/ExpSPIRcv	Digital I/O bit 2, Srl3Rcv, or ExpSPIRcv signal depending on pin MUX setting.

Pin #	Signal Name	Description
4	DigitalIO3/AxisIn/ ExpSPIClock	Digital I/O bit 3, AxisIn, or ExpSPIClock signal depending on pin MUX setting. AxisIn is the default functionality.
5	DigitalIO4/SynchIn/ ExpSPICS1	Digital I/O bit 4, SynchIn, or ExpSPICS1 signal depending on pin MUX setting. SynchIn is the default functionality.
6	DigitalIO5/AxisOut/ ExpSPICS2	Digital I/O bit 5, AxisOut, or ExpSPICS2 signal depending on pin MUX setting. AxisOut is the default functionality.
7	DigitalIO6/HostInter-rupt/ExpSPICS3	Digital I/O bit 6, HostInterrupt, or ExpSPICS3 signal depending on pin MUX setting. HostInterrupt is the default functionality.
8	DigitalIO7/Syn-chOut/ExpSPIStatus/ ExpSPICS4	Digital I/O bit 7, SynchOut, ExpSPIStatus, or ExpSPICS4 sig-nal depending on pin MUX setting. SynchOut is the default functionality.
9	DigitalIO8	Digital I/O bit 8
10	GND	Ground

6.1.9 Host SPI Connector (J8)

The following table details the Host SPI Connector (J8), which is provided with CAN/SPI host interface N-Series ION DKs. This connector is an 8-pin 4x2 100 mil header.

Pin #	Signal Name	Sense Interpretation	Description
		J8 - Host SPI Connector	
1	HostSPIXmt	N/A	Host SPI transmit output
2	HostSPIRcv	N/A	Host SPI receive input
3	HostSPIClock	N/A	Host SPI clock input
4	HostSPISelect	Active Low	Host SPI Enable input
5	HostInterrupt	Active Low	Magellan HostInterrupt signal out-put
6	HostSPIStatus	Active Low	Magellan Host SPI Status signal output
7	GND	N/A	Ground
8	Reset	Active Low	Master N-Series ION unit reset

6.1.10 Host CAN Connector (J10-1, J10-2)

The following table details the Host CAN Connectors (J10-1 and J10-2), which are provided with CAN host interface N-Series ION DKs. These connectors consist of two 8-pin RJ45s wired such that each J10-1 connector pin connects to the matching J10-2 connector pin (daisy chain configuration).

J10-1 Pin #	J10-2 Pin #	Signal Name	Description
		J10 - Host CAN Connector	
1	1	HostCAN+	Host CAN+
2	2	HostCAN-	Host CAN-
3	3	GND	Ground
4	4	SynchIn/SynchOut	Depending on DK board jumper setting this pin may carry a synch signal. See Section 6.2, “Jumper Settings,” for details.
5	5	N.C.	
6	6	N.C.	
7	7	GND	Ground
8	8	N.C.	

6.1.11 Expansion CAN Connector (J11-1, J11-2)

The following table details the Expansion CAN Connectors (J11-1 and J11-2), which consist of two 8-pin RJ45 connectors wired such that each J11-1 pin connects to the matching J11-2 connector pin (daisy chain configuration).

J11-1 Pin #	J11-2 Pin #	Signal Name	Description
		J11 - Expansion CAN Connector	
1	1	ExpCAN+	Expansion CAN+
2	2	ExpCAN-	Expansion CAN-
3	3	GND	Ground
4	4	N.C.	
5	5	N.C.	
6	6	N.C.	
7	7	GND	Ground
8	8	N.C.	

6.1.12 Host Serial Connector (J12)

The following table details the Host Serial Connector (J12), which is provided with serial host interface N-Series ION DKs. This connector supports two RS232 ports or a single RS485 port operating in either RS485 full duplex, or RS485 half duplex mode. This connector is a female DB-9.

Pin #	Signal Name	RS232	RS485 Full Duplex	RS485 Half Duplex
		J12 - Host Serial Connector		
1	No connect	no connect	no connect	no connect
2	Srl1Xmt	Serial 1 transmit output	no connect	no connect
3	Srl1Rcv	Serial 1 receive input	no connect	no connect
4	No connect	no connect	no connect	no connect
5	GND	Ground	Ground	Ground
6	RS485Rcv$^+$	no connect	Positive (non-inverting) receive input	no connect
7	Srl2Rcv/ RS485Rcv$^-$	Serial 2 receive input	Negative (inverting) receive input	no connect
8	Srl2Xmt/ RS485Xmt$^-$	Serial 2 transmit output	Negative (inverting) transmit output	Negative transmit/receive
9	RS485Xmt$^+$	no connect	Positive (non-inverting) transmit output	Positive transmit/receive

6.1.13 Host Serial Header Connector (J15)

The following table details the Host Serial Header Connector (J15), which is provided with serial host interface N-Series ION DKs. This connector provides a subset of the signals provided with the Host Serial Connector (J12). This connector is an 8-pin 100 mil header.

Pin #	Signal Name	Description
	J15 - Host Serial Header Connector	
1	Srl2Xmt	Serial2 Transmit
2	Srl2Rcv	Serial2 Receive
3	Srl1Xmt	Serial1 Transmit
4	Srl1Rcv	Serial1 Receive
5	DigitalIO6	DigitalIO6 signal
6	RS485Sel	RS232/RS485 select pin
7	GND	Ground

Pin #	Signal Name	Description
8	GND	Ground

6.1.14 Host Ethernet Connector (J13)

The following table details the Host Ethernet Connector (J13), which is provided with Ethernet host interface N-Series ION DKs. This connector is a female RJ-45.

Pin #	Signal Name	Description
	J13 - Host Ethernet Connector	
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 Jumper Settings

Here is the jumper setting for the Serial and Ethernet DK boards, which are shown in Figure 6-1 and Figure 6-3 respectively:

Jumper	Label	Factory Default	Description
JP1	N/A	Installed	When installed connects a 120 ohm termination resistor at the Expansion CAN bus. The terminating resistor should be used if the DK board is located at the end of the CAN network bus.

Here are the jumper settings for the SPI/CAN DK board, which is shown in Figure 6-2:

Jumper	Label	Factory Default	Description
JP4-1	E-Term	Installed	When installed connects a 120 ohm termination resistor at the Expansion CAN bus. The terminating resistor should be used if the DK board is located at the end of the CAN network bus.
JP4-2	H-Term	Installed	When installed connects a 120 ohm termination resistor at the Host CAN bus. This terminating resistor should be used if the DK board is located at the end of the CAN network bus.
JP4-3	SynchIn	Not installed	When installed connects pin 4 of the J10-2 Host CAN connector to the N-Series ION's SynchIn signal which is pin 34 of the Signal Connector. This connection allows daisy-chain ID assignment to be used described in the *ION/CME N-Series Digital Drive User Manual*.
JP4-4	SynchOut	Not installed	When installed connects pin 4 of the J10-1 Host CAN connector to the N-Series ION's SynchOut signal which is pin 37 of the Signal Connector. This connection allows daisy-chain ID assignment to be used described in the *ION/CME N-Series Digital Drive User Manual.*

6.3 N-Series ION Unit Part Numbers and Configurations

There are 36 different ION/CME N-Series Digital Drives in all, consisting of the combinations of four motor types (step motor, Brushless DC, DC Brush, Multi-Motor), three host interfaces (Serial, CAN/SPI, Ethernet), and three power levels (low, medium, high). Note that multi-motor units allow the motor type, either Brushless DC, DC Brush, or step motor, to be user programmed.

The following table shows the available N-Series ION part numbers:

P/N	Host Interface	Power Level	Voltage	Motor Type
Step Motor				
DD441S0056/02	Serial	Low (75W)	12-56V	Step Motor
DD441S0056/06	Serial	Medium (350W)	12-56V	Step Motor
DD441S0056/18	Serial	High (1,000W)	12-56V	Step Motor
DD441C0056/02	CAN/SPI	Low (75W)	12-56V	Step Motor
DD441C0056/06	CAN/SPI	Medium (350W)	12-56V	Step Motor
DD441C0056/18	CAN/SPI	High (1,000W)	12-56V	Step Motor
DD441D0056/02	Ethernet	Low (75W)	12-56V	Step Motor
DD441D0056/06	Ethernet	Medium (350W)	12-56V	Step Motor
DD441D0056/18	Ethernet	High (1,000W)	12-56V	Step Motor
Brushless DC				
DD431S0056/02	Serial	Low (75W)	12-56V	Brushless DC
DD431S0056/06	Serial	Medium (350W)	12-56V	Brushless DC
DD431S0056/18	Serial	High (1,000W)	12-56V	Brushless DC
DD431C0056/02	CAN/SPI	Low (75W)	12-56V	Brushless DC
DD431C0056/06	CAN/SPI	Medium (350W)	12-56V	Brushless DC
DD431C0056/18	CAN/SPI	High (1,000W)	12-56V	Brushless DC
DD431D0056/02	Ethernet	Low (75W)	12-56V	Brushless DC
DD431D0056/06	Ethernet	Medium (350W)	12-56V	Brushless DC
DD431D0056/18	Ethernet	High (1,000W)	12-56V	Brushless DC
DC Brush				
DD411S0056/02	Serial	Low (75W)	12-56V	DC Brush
DD411S0056/06	Serial	Medium (350W)	12-56V	DC Brush
DD411S0056/18	Serial	High (1,000W)	12-56V	DC Brush
DD411C0056/02	CAN/SPI	Low (75W)	12-56V	DC Brush
DD411C0056/06	CAN/SPI	Medium (350W)	12-56V	DC Brush
DD411C0056/18	CAN/SPI	High (1,000W)	12-56V	DC Brush
DD411D0056/02	Ethernet	Low (75W)	12-56V	DC Brush
DD411D0056/06	Ethernet	Medium (350W)	12-56V	DC Brush
DD411D0056/18	Ethernet	High (1,000W)	12-56V	DC Brush
Multi Motor				
DD481S0056/02	Serial	Low (75W)	12-56V	Multi Motor
DD481S0056/06	Serial	Medium (350W)	12-56V	Multi Motor
DD481S0056/18	Serial	High (1,000W)	12-56V	Multi Motor
DD481C0056/02	CAN/SPI	Low (75W)	12-56V	Multi Motor
DD481C0056/06	CAN/SPI	Medium (350W)	12-56V	Multi Motor
DD481C0056/18	CAN/SPI	High (1,000W)	12-56V	Multi Motor
DD481D0056/02	Ethernet	Low (75W)	12-56V	Multi Motor
DD481D0056/06	Ethernet	Medium (350W)	12-56V	Multi Motor
DD481D0056/18	Ethernet	High (1,000W)	12-56V	Multi Motor

6.4 Developer Kit Hardware Contents

6.4.1 Serial Host Interface DKs

The table below lists the hardware components and accessories provided with serial host interface N-Series ION DKs, P/Ns DK481S0056/02, DK481S0056/06, DK481S0056/18, and DK4X1S:

Name	PMD P/N	Description
N-Series ION Stack*		
N-Series ION unit**	DD481S0056/02 (low power) DD481S0056/06 (medium power) DD481S0056/18 (high power)	Either low, medium, or high power multi-motor serial interface N-Series ION unit
Serial DK PCB	PCB-1047-41	Populated four-layer interconnect PCB for serial host interface N-Series ION DKs
Base plate	NION-DKH-01	Black anodized aluminum N-Series ION DK base plate
Mounting screws	N/A	Four M2.5 x 0.45 mm thread, 8 mm long, button head screws
Thermal pad	NION-DKH-02	10 mil (.25 mm) thick thermal pad cut to dimensions 33.5 mm by 33.5 mm
Other Cables & Components		
3-pin programming cable	Cable-USB-3P	USB to 3-pin programming cable for communicating with N-Series ION
DB9 serial cable	Cable-USB-DB9	Male DB9 to USB serial converter cable
Allen key***	N/A	1.5 mm Allen key

** With the pre-assembled DKs the ION unit, interconnect DK PCB, base plate, thermal pad, and mounting screws are all pre-assembled together and the ION unit is soldered into the PCB. With the non pre-assembled DKs the N-Series ION unit is not included and the components are provided without being assembled together. For instructions on assembling a complete N-Series ION DK stack refer to Section 6.10, "Component Developer Kit Assembly."*

*** ION unit not included with non pre-assembled DKs*

**** Allen key not included with pre-assembled DKs*

6.4.2 CAN/SPI Host Interface DKs

The table below lists the hardware components and accessories provided with CAN/SPI host interface N-Series ION DKs, P/Ns DK481C0056/02, DK481C0056/06, DK481C0056/18, and DK4X1C:

Name	PMD P/N	Description
N-Series ION Stack*		
N-Series ION unit**	DD481C0056/02 (low power) DD481C0056/06 (medium power) DD481C0056/18 (high power)	Either low, medium, or high power multi-motor CAN/SPI interface N-Series ION unit
CAN/SPI DK PCB	PCB-1047-21	Populated four-layer interconnect PCB for CAN/SPI host interface N-Series ION DKs
Base plate	NION-DKH-01	Black anodized aluminum N-Series ION DK base plate
Mounting screws	N/A	Four M2.5 x 0.45 mm thread, 8 mm long, button head screws
Thermal pad	NION-DKH-02	10 mil (.25 mm) thick thermal pad cut to dimensions 33.5 mm by 33.5 mm

Name	PMD P/N	Description
Other Cables & Components		
3-Pin programming cable	Cable-USB-3P	USB to 3-pin programming cable for communicating with N-Series ION
CAN terminator	TRM-RJ45-02	RJ45 CAN terminator
Allen key***	N/A	1.5 mm Allen key

** With the pre-assembled DKs the ION unit, interconnect DK PCB, base plate, thermal pad, and mounting screws are all pre-assembled together and the ION unit is soldered into the PCB. With the non pre-assembled DKs the N-Series ION unit is not included and the components are provided without being assembled together. For instructions on assembling a complete N-Series ION DK stack refer to Section 6.10, "Component Developer Kit Assembly."*

*** ION unit not included with non pre-assembled DKs*

**** Allen key not included with pre-assembled DKs*

6.4.3 Ethernet Host Interface DKs

The table below lists the hardware components and accessories provided with Ethernet host interface N-Series ION DKs, P/Ns: DK481E0056/02, DK481E0056/06, DK481E0056/18, DK4X1E:

Name	PMD P/N	Description
N-Series ION Stack*		
N-Series ION Unit**	DD481D0056/02 (low power) DD481D0056/06 (medium power) DD481D0056/18 (high power)	Either low, medium, or high power multi-motor Ethernet interface N-Series ION unit
Ethernet DK PCB	PCB-1047-01	Populated four-layer interconnect PCB for Ethernet host interface N-Series ION DKs
Base plate	NION-DKH-01	Black anodized aluminum N-Series ION DK base plate
Mounting screws	N/A	Four M2.5 x 0.45 mm thread, 8 mm long, button head screws
Thermal pad	NION-DKH-02	10 mil (.25 mm) thick thermal pad cut to dimensions 33.5 mm by 33.5 mm
Other Cables & Components		
3-Pin Programming Cable	Cable-USB-3P	USB to 3-pin programming cable for communicating with N-Series ION
Ethernet cable	N/A	Standard 6' Cat 6 Ethernet connector cable
Allen key***	N/A	1.5 mm Allen key

** With the pre-assembled DKs the ION unit, interconnect DK PCB, base plate, thermal pad, and mounting screws are all pre-assembled together and the ION unit is soldered into the PCB. With the non pre-assembled DKs the N-Series ION unit is not included and the components are provided without being assembled together. For instructions on assembling a complete N-Series ION DK stack refer to Section 6.10, "Component Developer Kit Assembly."*

*** ION unit not included with non pre-assembled DKs*

**** Allen key not included with pre-assembled DKs*

6.4.4 Additional DK Accessories

The table below lists additional components that may be useful when working with N-Series ION DKs.

Name	Vendor & P/N	Description
USB to CAN FD connector	Ixxat, P/N: 1.01.0353.22012	USB to CAN FD converter, RJ45 interface. See Section 4.2.1, "CAN Hardware Setup," for more information on the use of this cable.
Dual serial channel cable	PMD, P/N: Cable-4355-01.R	Male DB9 dual serial to two single-channel female DB9 cable. See Section 4.1.1, "RS232 Hardware Setup," for more information on the use of this cable.
USB to RS422/RS485 connector	Advantech, BB-USOPTL4	USB to RS485 converter, terminal screw interface. See Section 4.1.4, "RS485 Hardware Setup," for more information on the use of this cable.

6.5 Selected Cable & Accessory Specifications

PMD Part #: Cable-4355-01.R

Description: Male DB9 dual serial to two single-channel female DB9 cable

Length: 5 ft (1.5 m)

Notes: Bifurcated shielded cable with male DB9 (P1 in wiring table) splitting to female DB9 carrying Serial1 channel (Srl1 in table) and female DB9 carrying Serial2 channel (Srl2 in table).

P1 DB9 Pin#	P1 DB9 Signal Name	Srl1 DB9 Pin#	Srl2 DB9 Pin#
1	N.C.*	N.C.	N.C.
2	Srl1Xmt	2	N.C.
3	Srl1Rcv	3	N.C.
4	N.C.	N.C.	N.C.
5	GND	5	5
6	N.C.	N.C.	N.C.
7	Srl2Rcv	N.C.	3
8	Srl2Xmt	N.C.	2
9	N.C.	N.C.	N.C.
Shield	Shield	Shield	Shield

**N.C. = No Connection*

PMD Part #: Cable-6001-01

Description: Female 40-pin dual 100 mil header to female 8-pin dual 100 mil header cable

Length: 10 ft (3.0 m)

Cable: UL 1569 22 AWG

Notes: Connects National Instruments NI-8452 converter module to PMD 8-pin SPI connector interface.

40-Pin Header Pin #*	40-Pin Header Signal Name	8-pin SPI Header Pin #	Cable Color
2	GND	7	Blue
4	SPI_SCLK	3	Red
8	SPI_MISO	1	Black
12	SPI_MOSI	2	Brown
13	DIO(0)	6	Green
16	CS(0)	4	Orange
17	DIO(1)	5	Yellow

**all pins not detailed are No Connect*

PMD Part #: TRM-RJ45-02

Description: Male 120 ohm RJ45 CAN terminator (at pins 1 & 2)

Notes: UL94V-0 compliant polycarbonate RJ45 housing with phosphor bronze contacts and thermoplastic white cap. Meets requirements of TIA-1096 and CE2011/65/EU.

8-Pin RJ45*	Wiring
1	Pin 1 connected to pin 2 by 120 ohm resistor
2	

** all pins not detailed are No Connect*

6.6 Conversion Factors, Defaults and Limits

To correctly control various N-Series ION features it may be helpful to know certain drive-specific scale factors. The following tables summarize these values.

6.6.1 Conversion Factors, Low Power Units

The following table provides electrical conversion factors for low power level N-Series ION units (P/Ns: DD4X1X0056/02).

These factors convert various integer Magellan IC command arguments (referred to as having units of counts) to physical quantities such as amperage, volts, etc... For more information on the Magellan Motion Control IC refer to the *Magellan Motion Control IC User Guide*. For more information on C-Motion commands refer to the *C-Motion PRP II Programming Reference*.

Unit	Example C-Motion Commands	Scaling	Example usage
Amps	GetCurrentLoopValue	.231 mA/count*	A command request to read the Actual-Current parameter returns a value 12,345. This corresponds to a current of 12,345 counts * 0.231 mA/count = 2.851A
Volts	SetDriveFaultParameter GetBusVoltage	10.0 mV/count	To set an overvoltage threshold of 50V, the command value should be 50 V *1,000 mV/V / 10.0 mV/count = 5,000
Temperature	SetDriveFaultParameter GetTemperature	.00391 °C/count	To set an overtemperature threshold of 65 °C, the command value should be 65 °C / .0039 °C/count = 16,624
I^2t continuous current	SetCurrent	.231 mA/count	To set an I^2t continuous current of 1.0A, the command value should be 1.0A * 1,000 mA/A / 0.231 mA/count = 4,329
I^2t energy	SetCurrent	.00588 A^2Sec/count	To set an I^2t energy of 5.0 A^2Sec, the command value should be 5.0 A^2Sec / 0.00588 A^2Sec/count = 850

** Some C-Motion commands specifying a current may have a scaling of two times this value. Refer to the Magellan Motion Control IC User Guide for more information.*

6.6.2 Defaults & Limits, Low Power Units

The following table provides default values, low limits and high limits for various specifiable drive-related parameters for low power level N-Series ION units (P/Ns: DD4X1X0056/02).

Setting	Default Setting	Low Limit	High Limit
Overtemperature limit	75.0 °C	0 °C	75.0 °C
Overtemperature hysteresis	5.0 °C	0 °C	25.0 °C
Overvoltage limit	60.0 V	10.0 V	60.0 V
Undervoltage limit	10.0 V	10.0 V	56.0 V
I^2t continuous current limit, Brushless DC motor	1.5 A	0.0 A	1.5 A
I^2t continuous current limit, DC Brush motor	1.5 A	0.0 A	1.5 A
I^2t continuous current limit, step motor	1.5 A	0.0 A	1.5 A
I^2t energy limit, Brushless DC Motor	2.95 A^2sec	0.0 A^2sec	2.95 A^2sec
I^2t energy limit, DC Brush Motor	3.63 A^2sec	0.0 A^2sec	3.63 A^2sec
I^2t energy limit, step motor	2.95 A^2sec	0.0 A^2sec	2.95 A^2sec

6.6.3 Conversion Factors, Medium Power Units

The following table provides electrical conversion factors for medium power level N-Series ION units (P/Ns: DD4X1X0056/06).

These factors convert various integer Magellan IC command arguments (referred to as having units of counts) to physical quantities such as amperage, volts, etc... For more information on the Magellan motion control IC refer to the *Magellan Motion Control IC User Guide*. For more information on C-Motion commands refer to the *C-Motion / PRP II Programming Reference*.

Unit	Example C-Motion Commands	Scaling	Example usage
Amps	GetCurrentLoopValue	.733 mA/count*	A command request to read the Actual-Current parameter returns a value 12,345. This corresponds to a current of of 12,345 counts * 0.733 mA/count = 9.049
Volts	SetDriveFaultParameter GetBusVoltage	10.0 mV/count	To set an overvoltage threshold of 50V, the command value should be 50 V *1,000 mV/V / 10.0 mV/count = 5,000
Temperature	SetDriveFaultParameter GetTemperature	.0039 °C/count	To set an overtemperature threshold of 65 °C, the command value should be 65 °C / .0039 °C/count = 16,667
I^2t continuous current	SetCurrent	.733 mA/count	To set an I^2t continuous current of 4.0A, the command value should be 4.0A * 1,000 mA/A / 0.733 mA/count = 5,457
I^2t energy	SetCurrent	.0590 A^2Sec/count	To set an I^2t energy of 20.0 A^2Sec, the command value should be 20.0 A^2Sec / 0.0590 A^2Sec/count = 339

6.6.4 Defaults & Limits, Medium Power Units

The following table provides default values, medium limits and high limits for various specifiable drive-related parameters for low power level N-Series ION units (P/Ns: DD4X1X0056/06).

Setting	Default Setting	Low Limit	High Limit
Overtemperature limit	75.0 °C	0 °C	75.0 °C
Overtemperature hysteresis	5.0 °C	0 °C	25.0 °C
Overvoltage limit	60.0 V	10.0 V	60.0 V
Undervoltage limit	10.0 V	10.0 V	56.0 V
I^2t continuous current limit, Brushless DC motor	5.5 A	0.0 A	5.5 A
I^2t continuous current limit, DC Brush motor	7.1 A	0.0 A	7.1 A
I^2t continuous current limit, step motor	5.0 A	0.0 A	5.0 A
I^2t energy limit, Brushless DC Motor	25.1 A^2sec	0.0 A^2sec	25.1 A^2sec
I^2t energy limit, DC Brush Motor	28.1 A^2sec	0.0 A^2sec	28.1 A^2sec
I^2t energy limit, step motor	28.2 A^2sec	0.0 A^2sec	28.2 A^2sec

6.6.5 Conversion Factors, High Power Units

The following table provides electrical conversion factors for high power level N-Series ION units (P/Ns: DD4X1X0056/18).

These factors convert various integer Magellan IC command arguments (referred to as having units of counts) to physical quantities such as amperage, volts, etc... For more information on the Magellan Motion Control IC refer to the *Magellan Motion Control IC User Guide*. For more information on C-Motion commands refer to the *C-Motion PRP II Programming Reference*.

Unit	Example C-Motion Commands	Scaling	Example usage
Amps	GetCurrentLoopValue	2.198 mA/count	A command request to read the ActualCurrent parameter returns a value 12,345. This corresponds to a current of of 12,345 counts * 2.198 mA/count = 27.13A
Volts	SetDriveFaultParameter GetBusVoltage	10.0 mV/count	To set an overvoltage threshold of 50V, the command value should be 50 V *1,000 mV/V / 10.0 mV/count = 5,000
Temperature	SetDriveFaultParameter GetTemperature	.0039 °C/count	To set an overtemperature threshold of 65 °C, the command value should be 65 °C / .0039 °C/count = 16,667
I^2t continuous current	SetCurrent	2.198 mA/count	To set an I^2t continuous current of 15.0A, the command value should be 15.0A * 1,000 mA/A / 2.198 mA/count = 6,824
I^2t energy	SetCurrent	.531 A^2Sec/count	To set an I^2t energy of 100.0 A^2Sec, the command value should be 100 A^2Sec / .531 A^2Sec/count = 188

6.6.6 Defaults & Limits, High Power Units

The following table provides default values, high limits and high limits for various specifiable drive-related parameters for low power level N-Series ION units (P/Ns: DD4X1X0056/18).

Setting	Default Setting	Low Limit	High Limit
Overtemperature limit	75.0 °C	0 °C	75.0 °C
Overtemperature hysteresis	5.0 °C	0 °C	25.0 °C
Overvoltage limit	60.0 V	10.0 V	60.0 V
Undervoltage limit	10.0 V	10.0 V	56.0 V
I^2t continuous current limit, Brushless DC motor	14.8 A	0.0 A	14.8 A
I^2t continuous current limit, DC Brush motor	19.0 A	0.0 A	19.0 A
I^2t continuous current limit, step motor	13.4 A	0.0 A	13.4 A
I^2t energy limit, Brushless DC Motor	257.4 A^2sec	0.0 A^2sec	257.4 A^2sec
I^2t energy limit, DC Brush Motor	280.5 A^2sec	0.0 A^2sec	280.5 A^2sec
I^2t energy limit, step motor	281.0 A^2sec	0.0 A^2sec	281.0 A^2sec

For the N-Series ION, default values and limits for the I^2t continuous current limit and I^2t energy limit are designed to be safe for operation in the drive's highest output mounting option, namely, using a cold plate for heat sinking. See Section 6.12, "Mechanical Mounting," for information on N-Series ION mounting options.

If the N-Series ION drive is being operated at a lower voltage it may be possible to specify values for I^2t continuous current limit and I^2t energy limit that are higher than the default, but lower than or equal to the limit, since the continuous output current rating of the N-Series ION drive is higher for lower input voltages. See *ION/CME N-Series Digital Drive User Manual* for drive output specifications.

For other mounting configurations, or for use with motors that have lower current and energy limits, it may be useful to set these parameters to values lower than the default values.

It is the responsibility of the user to set the I^2t continuous current and I^2t energy limit parameters to values that are safe for the specific N-Series ION mounting configuration and motor setup being used.

6.7 LEDs

N-Series IONs have two LEDs for indicating the status of the unit, one green and one red. The green LED is called the power LED and is solid on when the unit is powered and the *Enable* signal is active (low). If the unit is powered but the *Enable* signal is inactive (high) this LED will blink. If the unit is not powered this LED is off.

The red LED is called the status LED and is normally off indicating no errors. There are three additional states of this LED consisting of fast blink (three times per second), slow blink (once per second), and solid on. The table below summarizes the above:

Green (Power) LED	Red (Status) LED	Condition
off	off	Unit not powered
solid on	off	Unit Enabled (signal level low)
blinking slow	off	Unit Disabled (Enable signal level high)
solid on	blinking fast	Unit experienced an overvoltage or undervoltage condition
solid on	blinking slow	Unit experienced an overtemperature or I^2t overcurrent condition
solid on	solid on	Unit experienced a hard fault condition

For more information on overvoltage, undervoltage, overtemperature, I^2t overcurrent, and overcurrent conditions including suggested recovery procedures refer to the *ION/CME N-Series Digital Drive User Manual*.

Any activation of the red LED indicates a potentially serious fault in the drive or in the system being controlled. It is the responsibility of the user to determine the correct and safe recovery procedure for such a condition.

6.8 Shunt Regulation

The N-Series ION provides a connection for a shunt resistor and diode that may be used to regulate overvoltage conditions on the DC bus. Such conditions can occur during deceleration of a motor with a large inertia. Shunt control is an option for use with DC Brush and Brushless DC motors. It is not used with step motors.

Shunt regulation functions by comparing the DC bus voltage to a user programmable threshold. If the DC bus voltage exceeds the comparison threshold the Shunt pin outputs a PWM waveform at a user programmable duty cycle, variously connecting the shunt pin to GND in its active state and HV in its inactive state.

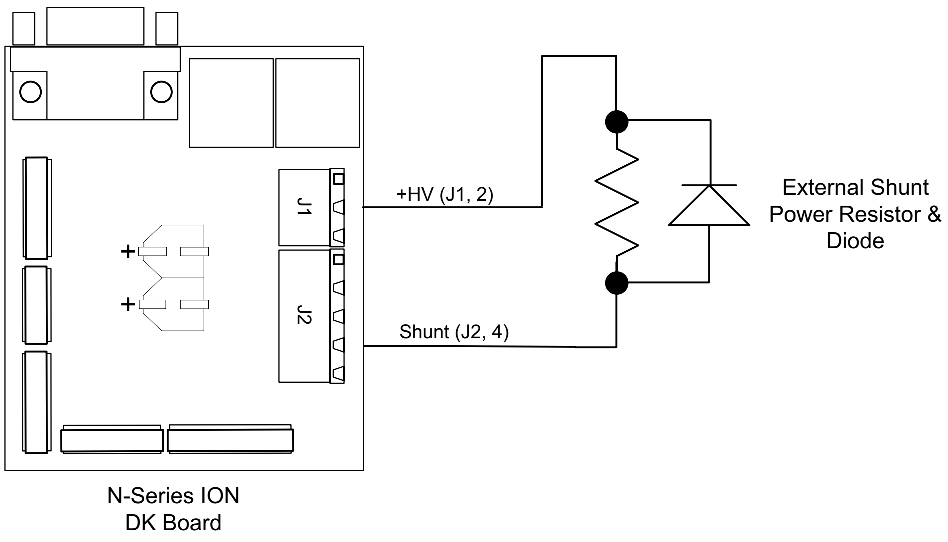
Whether or not shunt should be used in a given application depends on a number of factors. For a complete application note on DC Bus voltage regulation including use of shunt regulation see Section 3.11.4, "Application Note – Monitoring and Enhancing DC Bus Voltage Stability."

Figure 6-4 shows how the shunt resistor and diode should be wired to the N-Series ION DK board. The connection to the shunt output is made via pin #4 of the Motor Drive Connector (J2) and the connection to HV is made via pin# 2 of the Power Connector (J1). Both the resistor and the diode should have ratings at least 50% above the DC supply voltage to be used. General guidelines for the maximum current that should be output through the Shunt pin are 20A for high power N-Series IONs, 10A for medium power units, and 5A for low power.

To calculate the average current flow through the resistor Ohm's law is used. For example if a shunt resistor with a resistance of 10 ohms is installed with a programmed comparison value of 51 volts and a PWM duty cycle of 75%, when the +HV voltage exceeds 51.0 volts HV will be connected to GND via the shunt resistor resulting in an effective average current flow of (51.0V * 0.75)/10 ohms = 3.825 amps.

To program the Shunt output function select the Drive Safety box in Pro-Motion's Axis Control window.

Figure 6-4: Shunt Resistor Wiring Diagram



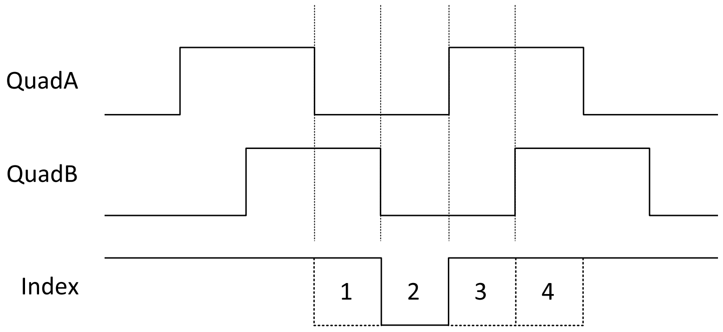
6.9 Index Capture Qualification

N-Series ION drives 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 6-5: 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.

6.10 Component Developer Kit Assembly

6.10.1 Overview

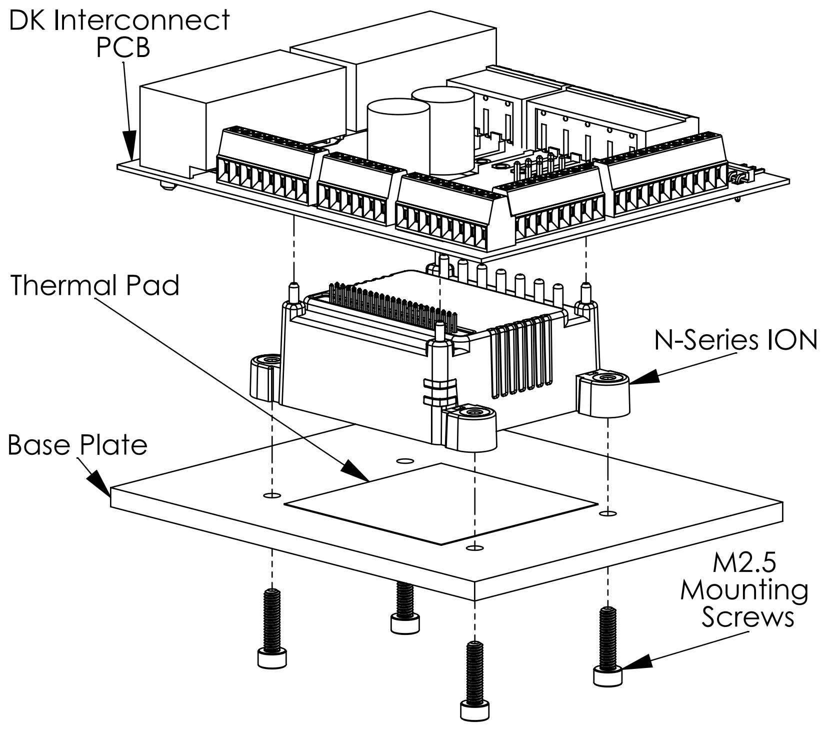


Figure 6-6: N-Series ION DK Showing Component Stack

When using non pre-assembled N-Series ION DKs assembly of the N-Series ION with the DK components is required. Along with the N-Series ION unit, which is not included with non pre-assembled DKs and must be ordered separately, the component stack elements are listed in the table below from top to bottom.

DK Stack Component Name	Description
DK Interconnect PCB	One of three possible PCBs that is soldered to the N-Series ION and provides convenient connectors to access the ION functions. The ION unit interface type (serial, CAN/SPI, or Ethernet) must be matched with the associated DK interconnect PCB type. For more information on this and exact P/Ns refer to Section 6.4, "Developer Kit Hardware Contents."

DK Stack Component Name	Description
N-Series ION unit	One of 36 different N-Series ION units. As noted above the ION unit type must match the PCB interconnect PCB unit type
Thermal pad	33.5 mm x 33.5 mm x .25 mm thermal pad that insures correct thermal flow between the ION unit and the included base plate or application mounting surface
Mounting screws	Four M2.5 mounting screws
Base plate	Black anodized aluminum mounting plate which functions both as a heat sink and as a platform support for bench-top exercising of the DK. The bottom of the plate has four rubber feet for stability.

The next few sections will provide detailed instructions for assembling a complete DK stack from the components listed above. For additional information on mechanical mounting of the N-Series ION unit refer to Section 6.12, "Mechanical Mounting." For additional information on DK P/Ns and contents refer to Section 6.4, "Developer Kit Hardware Contents."

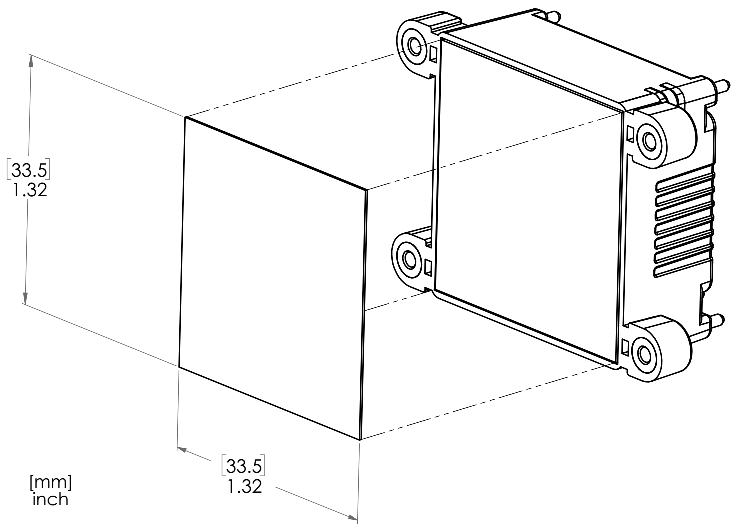
The N-Series ION contains electronic components and is therefore subject to damage from ESD (Electro Static Discharge). Therefore, during the assembly procedures described below and at all other times while handling the ION unit the operator should apply measures to reduce the potential for ESD damage which include use of grounding straps, anti-static mats, environmental controls, and other measures.

6.10.2 Assembly Sequence

Attaching the thermal pad to the N-Series ION

The first step of the assembly procedure is to insure that there is good thermal contact between the N-Series ION and the base plate heat sink. For this purpose, a thermal pad will be attached to the ION's metallic heat sink interface.

Figure 6-7: Mounting Thermal Pad to N-Series ION



To accomplish this, locate the thermal pad in the developer kit box. Next, carefully remove the thin plastic protective sheets on either side of the thermal pad and mount onto the ION unit, carefully aligning the pad with the ION unit's metallic backing and applying finger pressure to adhere the pads to the metal.

After adherence to the ION's metallic backing the thermal pad should appear flat and without bubbles or bulges. Once pressed in place the pad should stay in place but if required the pad can be removed and remounted.

Mounting the N-Series ION to the base plate

The next step of the assembly process is to mount the N-Series ION with thermal pad attached to the rectangular base plate. Four M2.5 screws and the Allen key are used for this purpose as shown in Figure 6-6. Make sure to orient the base plate so that the rubber feet are away from the ION unit.

When tightening the screws use only modest torquing force, and alternate tightening in an X pattern so that the applied force is increased slowly and equally amongst the four screws. Should you hear or see a snap of the ION unit tab while tightening a screw this means the ION unit's protective fuse tab has engaged because too much torque was applied. If this occurs the N-Series ION unit can not be used. It must be de-installed and discarded, and a new unit installed.

A fuse break event serves as a signal to the operator to revise their procedure so that the tab mounting torque limits are not exceeded during future attachment procedures. For detailed mounting torque specifications refer to Section 6.12.2.2, "Installation & Torque Limits."

Soldering the N-Series ION to the DK PCB

The final step of the DK stack assembly process is to solder the DK PCB to the N-Series ION. To accomplish this place the assembled base plate and N-Series ION onto a level surface with the base plate resting on its rubber pads.

With the PCB oriented so that the connector-side faces up, locate the PCB in the correct mounting position taking note of the location of the hole patterns in the PCB which mate with the N-Series ION. When ready gently push the PCB down onto the ION unit until the bottom of the PCB sits flush with the ION unit's PCB interface surface.

Altogether you will solder three different groups of pins; the four alignment posts located at the corners of the ION unit, the 44 signal connector pins, and the 7 power connector pins. Carefully solder each of these pins into the PCB taking care to insure proper solder flow so that good electrical and mechanical contact is made.

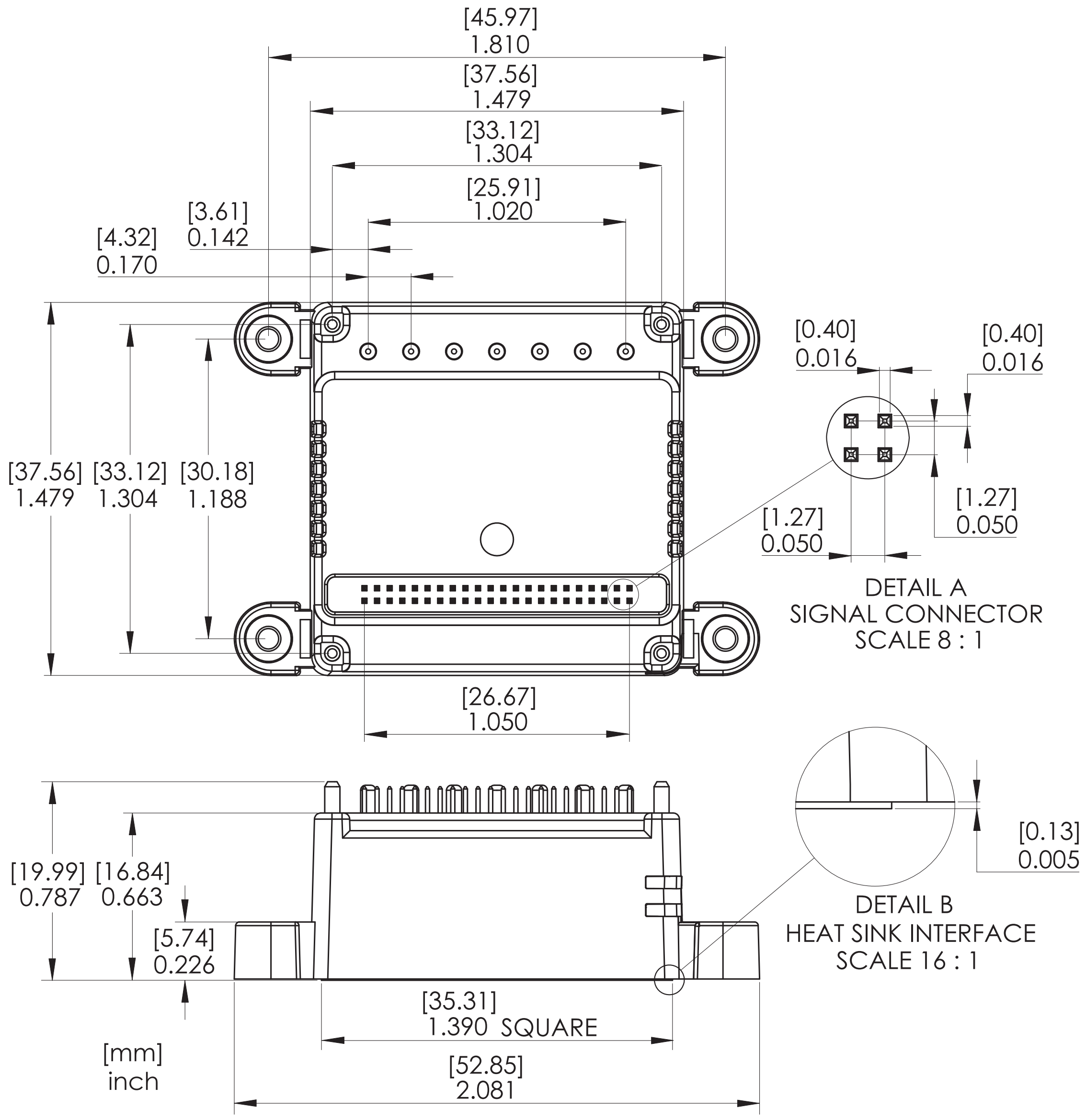
Congratulations! Once soldering has been completed assembly of your N-Series ION developer kit is complete and the DK is ready for operation.

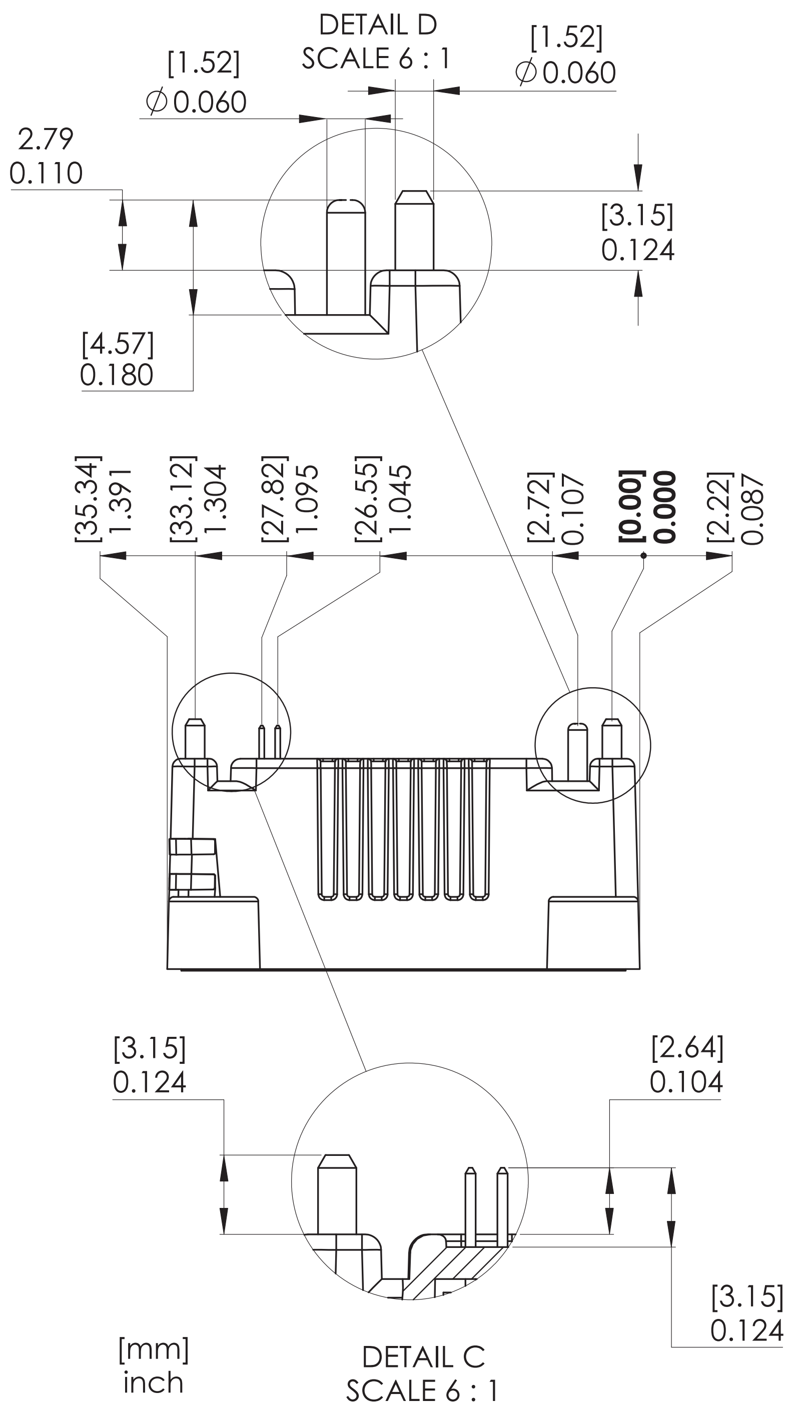
Take special care to insure proper solder flow occurs for the seven power connector pins. These larger pins may require more time to heat or a higher soldering iron temperature setting. To aid confirmation of a proper solder joint the area around the ION unit's power pins is recessed, allowing the technician visual access of the power pins from the underside of the PCB.

6.11 Physical Dimensions

6.11.1 N-Series ION Unit

Figure 6-8: ION/CME N-Series Digital Drive Physical Dimensions




DETAIL D
SCALE 6 : 1
[1.52]
∅0.060
[1.52]
∅0.060
2.79
0.110
[3.15]
0.124
[4.57]
0.180
[35.34]
1.391
[33.12]
1.304
[27.82]
1.095
[26.55]
1.045
[2.72]
0.107
[0.00]
0.000
[2.22]
0.087
[3.15]
0.124
[2.64]
0.104
[3.15]
0.124
[mm]
inch
DETAIL C
SCALE 6 : 1

6.11.2 Developer Kit

Figure 6-9: DK Base Plate Dimensions

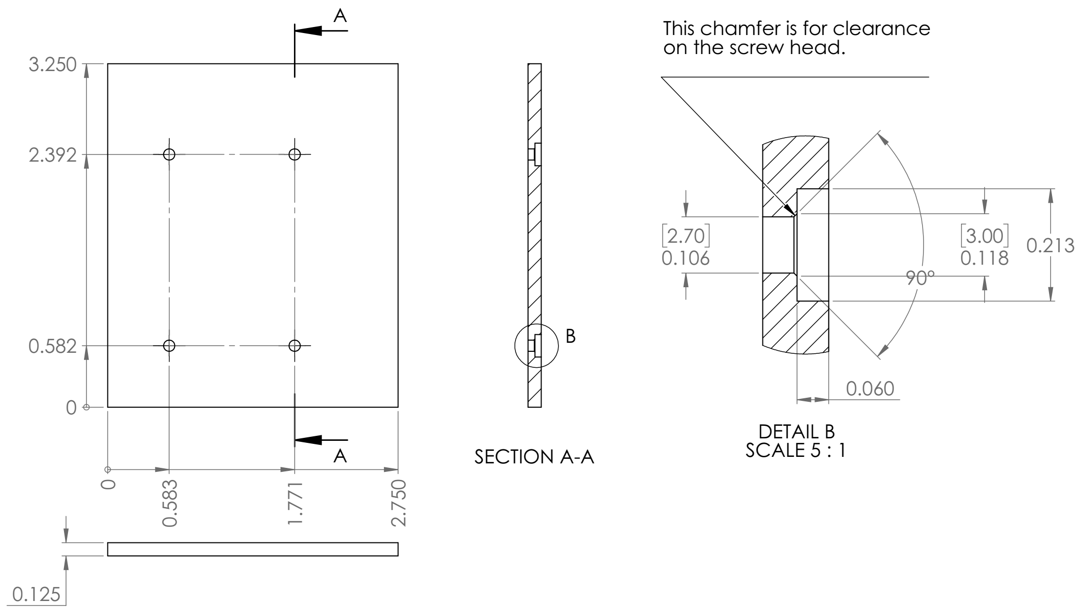
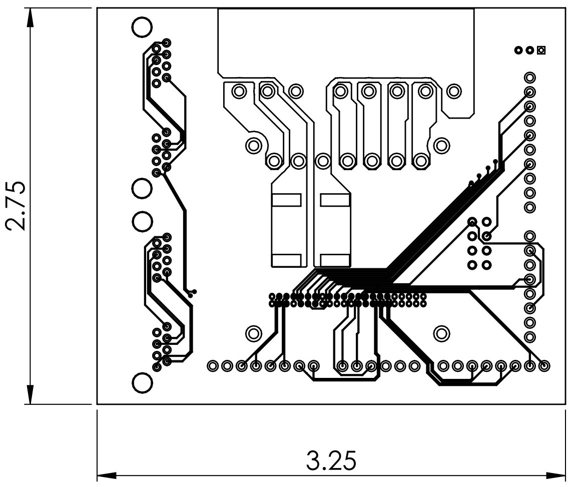


Figure 6-10: DK Interconnect Board Dimensions



6.12 Mechanical Mounting

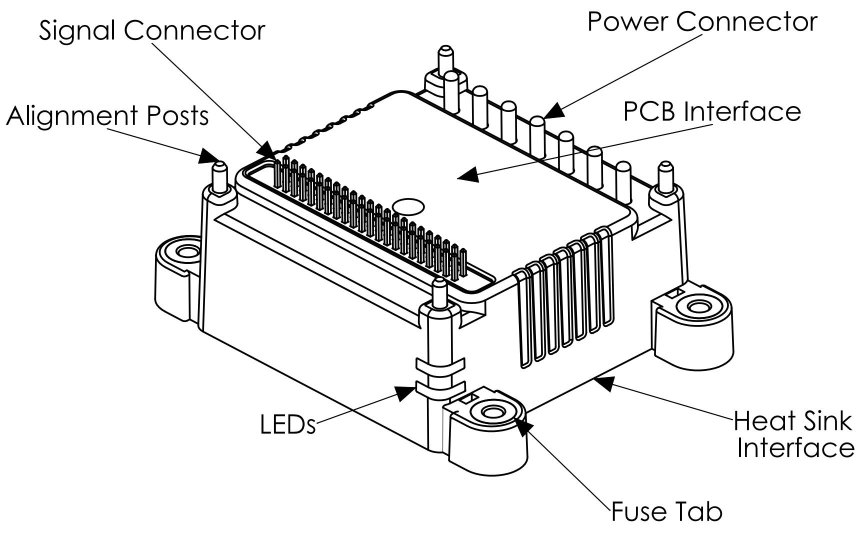


Figure 6-11: Mechanical Elements of N-Series ION Drive

The N-Series Drives are designed to be used in a wide variety of operating conditions. As shown in Figure 6-11 they have a robust mechanical design that allows rigid mechanical attachment at the two primary interface points; the interface between the N-Series Drive and the user PCB, and the interface between the N-Series Drive and the heat sink or cold plate also called a supporting plate. The following sections describe how mechanical attachment at both these interfaces occurs.

6.12.1 Attachment to the User PCB

Attachment at the PCB is accomplished by soldering the PCB to the N-Series Drive's four mechanical alignment posts. While these posts do not serve an electrical function, they provide a rigid attachment of the PCB to the N-Series ION, and thereby reduce strain that may occur in the solder connections between the user PCB and the Power and Signal Connectors of the N-Series ION. Use of the four PCB alignment posts are recommended for all applications.

6.12.2 Attachment to the Heat Sink

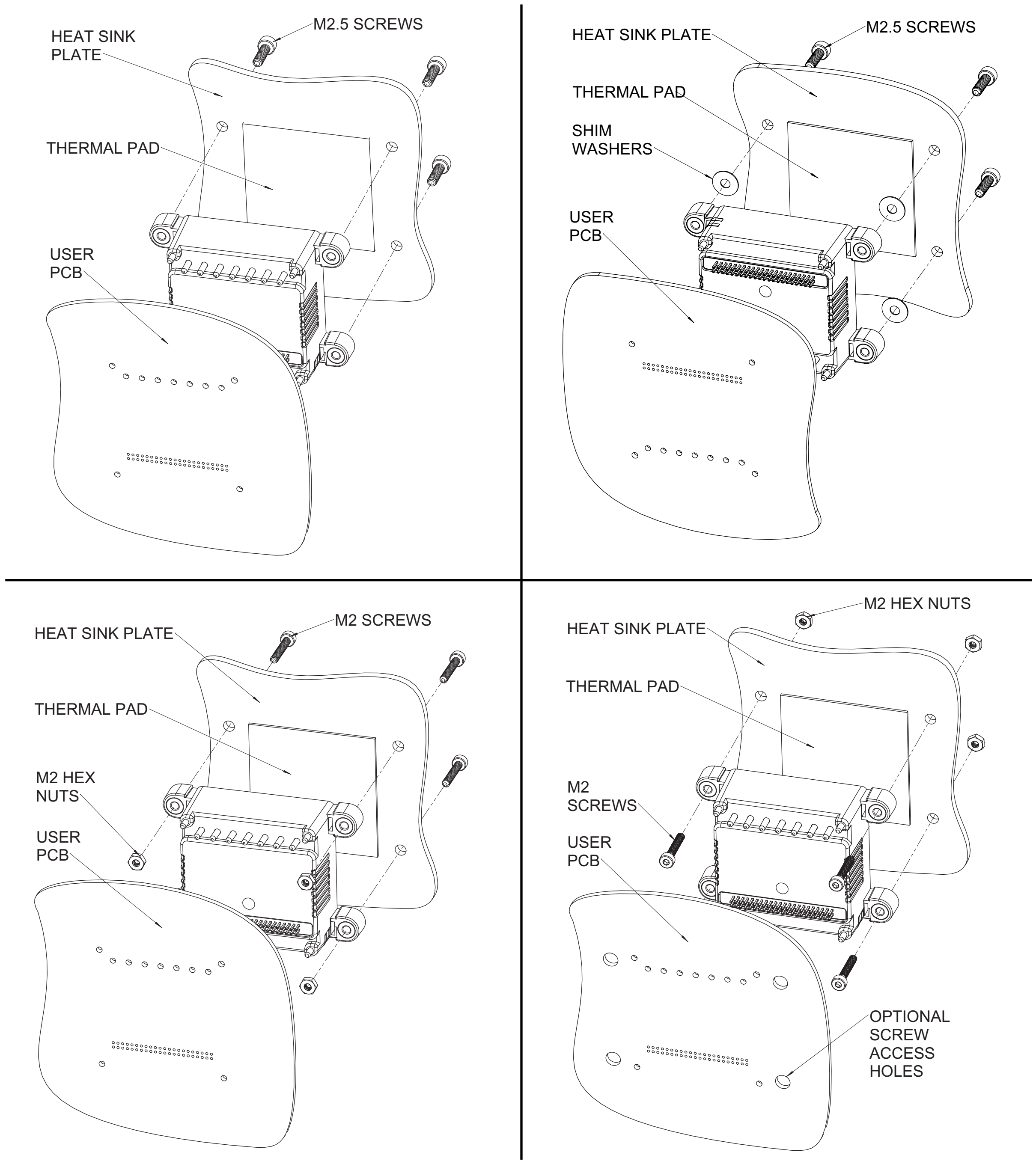


Figure 6-12: N-Series ION Mounting Options

6.12.2.1 Installation Overview

Attachment to the heat sink or supporting plate is accomplished via screws at the N-Series ION's four mounting tabs. As shown in Figure 6-12 there are four typical approaches to mechanically attaching to the heat sink.

The first is mounting from the heat sink side. In this approach M2.5 screws are installed directly into the ION's tab which is threaded for a M2.5 screw. A variation of this option may be used when greater mounting rigidity is desired, typically because the application environment has higher g forces. This mounting method locates shim washers under the tabs to limit tab deflection and to increase the amount of torque that can be applied to tighten down the tabs.

When using this method the thickness of the shim washer should be determined by the type and thickness of thermal transfer material used to create a vibration resistant mount, while maintaining optimal thermal transfer. Note that in this approach, the N-Series ION's fuse tab mechanism, which normally acts to limit the maximum force that can be applied via the tabs, is effectively bypassed. Therefore it is up to the user to ensure that the force applied to the ION unit's heat sink interface, the limits for which are provided in the table below, are not exceeded. For more information on PMD's patented fuse tab design see Section 6.12.2.3, "N-Series Drive Fuse Tab Design."

To achieve the shim function using washers, the recommended dimensions are .280" (7.0 mm) outer diameter and .105" (2.5 mm) inner diameter. Representative suppliers for washers such as this include Bokers, Superior Washer, and Phoenix Specialty.

When using shim washers or other approaches that raise the mounting interface under the N-Series ION tabs, the ION unit's force-limiting fuse tab design is effectively disabled. It is therefore up to the user to ensure that the mounting force limit on the ION unit's heat sink interface plate is not exceeded.

Two other mounting options use a nut and a screw. Most commonly M2 screws are used which go through the tab's M2.5 thread without engaging the thread, with the nut used to capture the M2 screw. The two possible orientations of the nut and screw represent these two different mounting options. Note that if the preferred mounting orientation is to have the screw heads facing the user PCB, it may be desirable to have holes in the PCB to give access to the ION screw hardware. However if the ION is mechanically mounted first, and then soldered onto the user PCB, these access holes may not be necessary.

6.12.2.2 Installation & Torque Limits

Before mating the N-Series Drive to the heat sink, in a typical installation a thermal transfer material will first be installed to enhance thermal contact. Refer to Section 6.12.2.4, "Thermal Transfer Materials," for more information on selection of these materials.

Once this material is in place the tabs can be tightened under careful torque limit control. The table below shows the recommended maximum torque for both M2 and M2.5 screws along with equivalent force values if an alternate attachment method such as a clamp is used, or if shim washers are used as described above. These torque values were determined using 18-8 stainless steel screws.

Type	Minimum	Recommended	Maximum
M2 screw torque	0.082 N-m (0.722 in-lb)	0.094 N-m (0.836 in-lb)	0.113 N-m (1.00 in-lb)
M2.5 screw torque	0.113 N-m (1.00 in-lb)	0.130 N-m (1.15 in-lb)	0.147 N-m (1.30 in-lb)
Force	25 kgs (55 lbs)	29 kgs (68 lbs)	33 kgs (71 lbs)

Exceeding the N-Series ION's mechanical attachment specifications indicated above may cause a failure of the drive unit at the time of installation, or at a later time in the field.

6.12.2.3 N-Series Drive Fuse Tab Design

N-Series IONs utilize a patented fuse-tab design to reduce the chance that overtightening of the tab attachment hardware will cause damage to the N-Series ION unit.

This is achieved by the tab deliberately breaking (snapping) under conditions of over tightening. Such a protective fuse break event (snap) of the tab should be audible to the operator during tightening. While an N-Series Drive unit that has undergone a fuse break event during tightening is no longer usable, it serves as a signal to the operator to check and revise their torque control procedure so that the tab mounting torque limits are not exceeded during future attachment procedures.

An ION/CME N-Series Digital Drive which has undergone a fuse tab event (snap) during installation must be de-installed and discarded. It is not possible to repair or to otherwise use an N-Series Drive unit which has undergone a fuse snap event.

The N-Series ION's fuse-tab functions when there is an air gap between the bottom of the tab and the surface that it is mounted to. If there is no air gap, or if the air gap is too small then the fuse/snap function may not occur even though the recommended mounting torque limit may have been exceeded. In all cases it is the responsibility of the user to determine that the recommended mounting torque limits or force indicated in Section 6.12.2.2, "Installation & Torque Limits," are not exceeded.

6.12.2.4 Thermal Transfer Materials

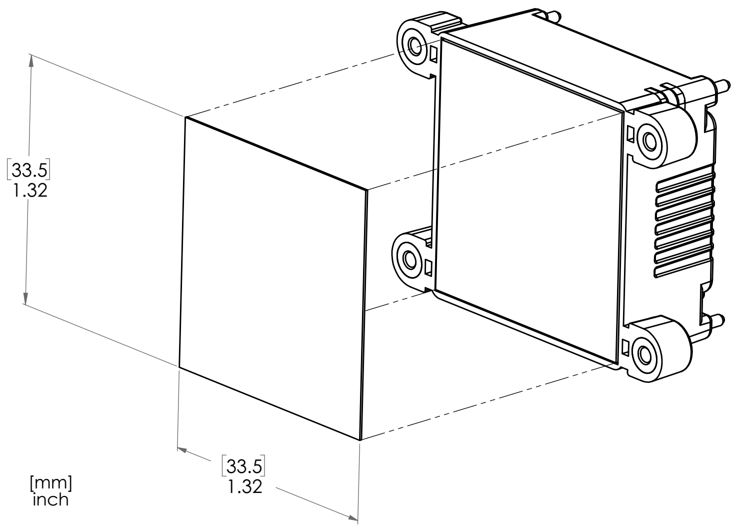


Figure 6-13: Recommended N-Series ION Thermal Transfer Material Dimensions

Thermal transfer materials in the form of thermal tape, pads, paste, or epoxy may be used to improve thermal transfer between the N-Series Drive's metal plate and an attached heat sink or supporting plate. These materials improve thermal conductivity by filling in air gaps that form when two metallic surfaces are mated.

Figure 6-13 shows a typical application of a thermal transfer material between the drive unit and a heat-removing metal surface. The following guidelines may be helpful in selecting and sizing the thermal transfer material best-suited to your application.

The capacity of thermal transfer materials to transfer heat (known as the bulk conductivity) is much lower than that of metals such as aluminum or copper. Therefore, in general, the thinner the transfer material the better. Thickness of the material is only precisely controllable for thermal pads and thermal tapes, with thermal pads providing the thinnest available interfaces beginning at 5 mils (.127 mm) or even less. For use with N-Series IONs thermal transfer materials that are thicker than 40 mils (1.0 mm) are not recommended regardless of the material used.

When using thermal paste or thermal epoxy glue the thickness should be carefully controlled via a silk screen or other wet film application process. The N-Series unit itself should not be used to squeeze non-uniformly applied paste or epoxy flat during installation. Doing so may result in damage to the unit.

Whether using tape, pads, paste, or epoxy, as shown in Figure 6-13, the thermal transfer material that is used as the interface should not extend to the area under the N-Series ION's tabs because this may reduce the amount of compression that occurs in the thermal transfer area. The following table provides dimensions for the applied thermal transfer material for N-Series IONs:

Parameter	Maximum Pad Dimensions
Value	33.5 mm x 33.5 mm (1.32" x 1.32”)

6.12.3 Additional Mounting Information

There are a number of additional precautions and procedures that should be followed to maintain the electrical and mechanical integrity of the N-Series ION unit during installation.

Soldering N-Series ION units in place. Applications that involve N-Series ION units mated to a supporting plate should take special care to insure that the solder joints are not stressed by the supporting plate once installed. The recommended method to achieve this is to mechanically mate the unit to the supporting plate before soldering into the PCB. If, for whatever reason, this is not possible, then special care should be taken to insure that the N-Series ION is aligned with the supporting plate after soldering and before mechanical attachment so that upon mechanical attachment no stress is placed on the ION unit, the solder contacts, or the PCB.

Mounting surface flat and clean. Thermal performance as well as safe operation of the N-Series ION requires that the heat sink or supporting plate surface that it is mounted to be flat and clean, free of dust, grease, or foreign objects. The recommended maximum deviation of the mating surface flatness is 3 mils (.076 mm).

Progressive tightening. N-Series ION units that are mated to a heat sink or supporting plate should be attached by progressively tightening each of the unit's four tabs. This means that one screw may be tightened, followed by the others, than back to the first etc. until the desired torque at each screw has been achieved. Following this procedure is particularly important when installing N-Series ION units over paste or epoxy, where the subsurface layer will undergo compression and movement before settling to a final installed position.

It is the responsibility of the user to ensure that all N-Series ION units have been installed within the prescribed mechanical stress limits and following the above described procedures. Failure to observe any of the above recommended procedures and limits may result in incorrect operation or failure of the unit during operation.

This page intentionally left blank.

7. DK Interconnect Board Schematics

In This Chapter

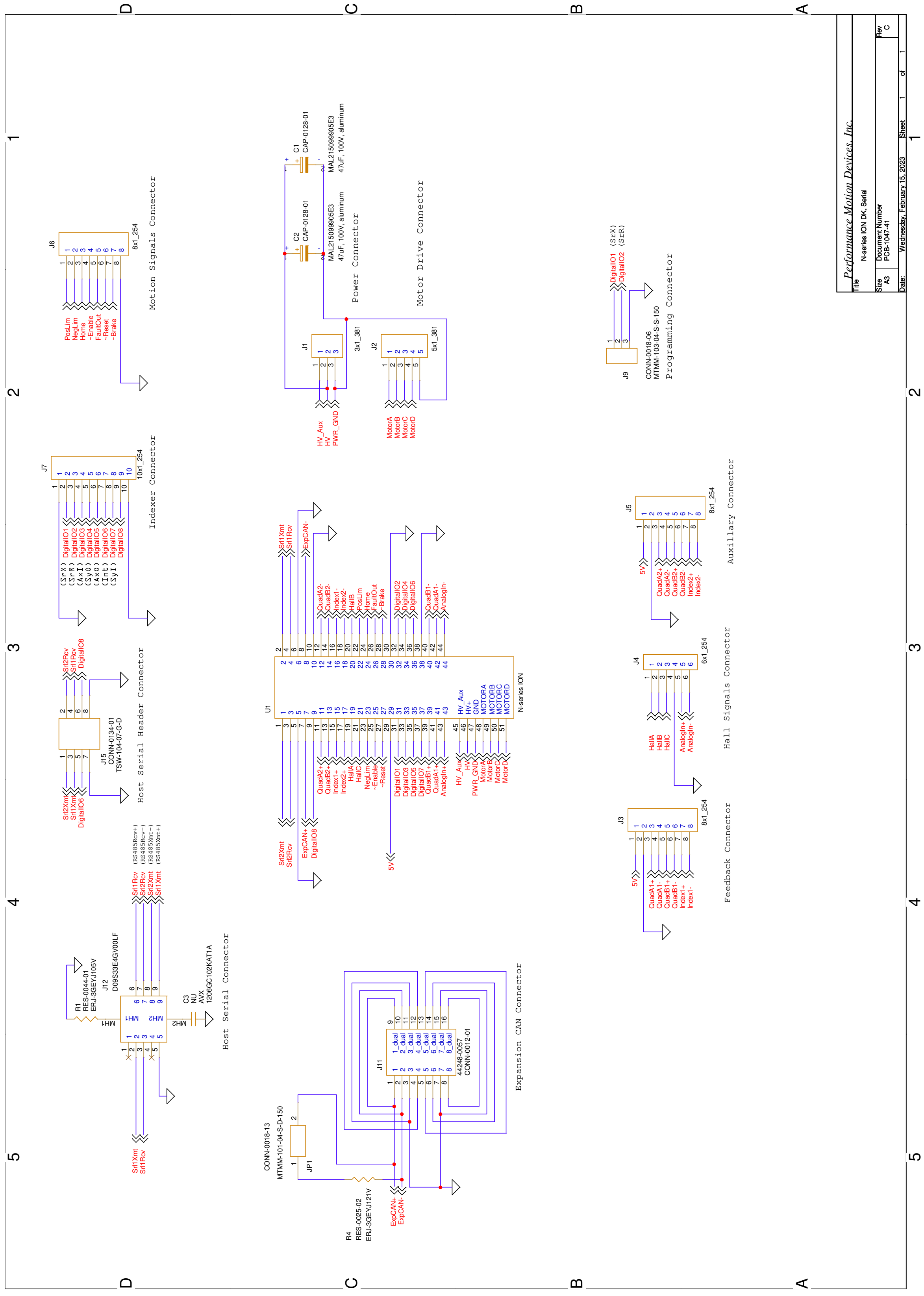
7.1 DK Interconnect Board Schematics

The schematics on the next three pages are the designs for the ION/CME N-Series Digital Drive Developer Kit's printed circuit board, differing by the host communication variant.

As the N-Series ION is designed to limit the requirement for external components, the interconnect boards simply provide connectors to all pins on the N-series ION, similar to a breakout board. Aside from host communication pins, which use common connectors, all signals are accessible through screw terminals.

7.1.1 Serial Host Interface Type

Figure 7-1: Serial Host Interface DK Interconnect Board Schematic



7.1.2 CAN/SPI Host Interface Type

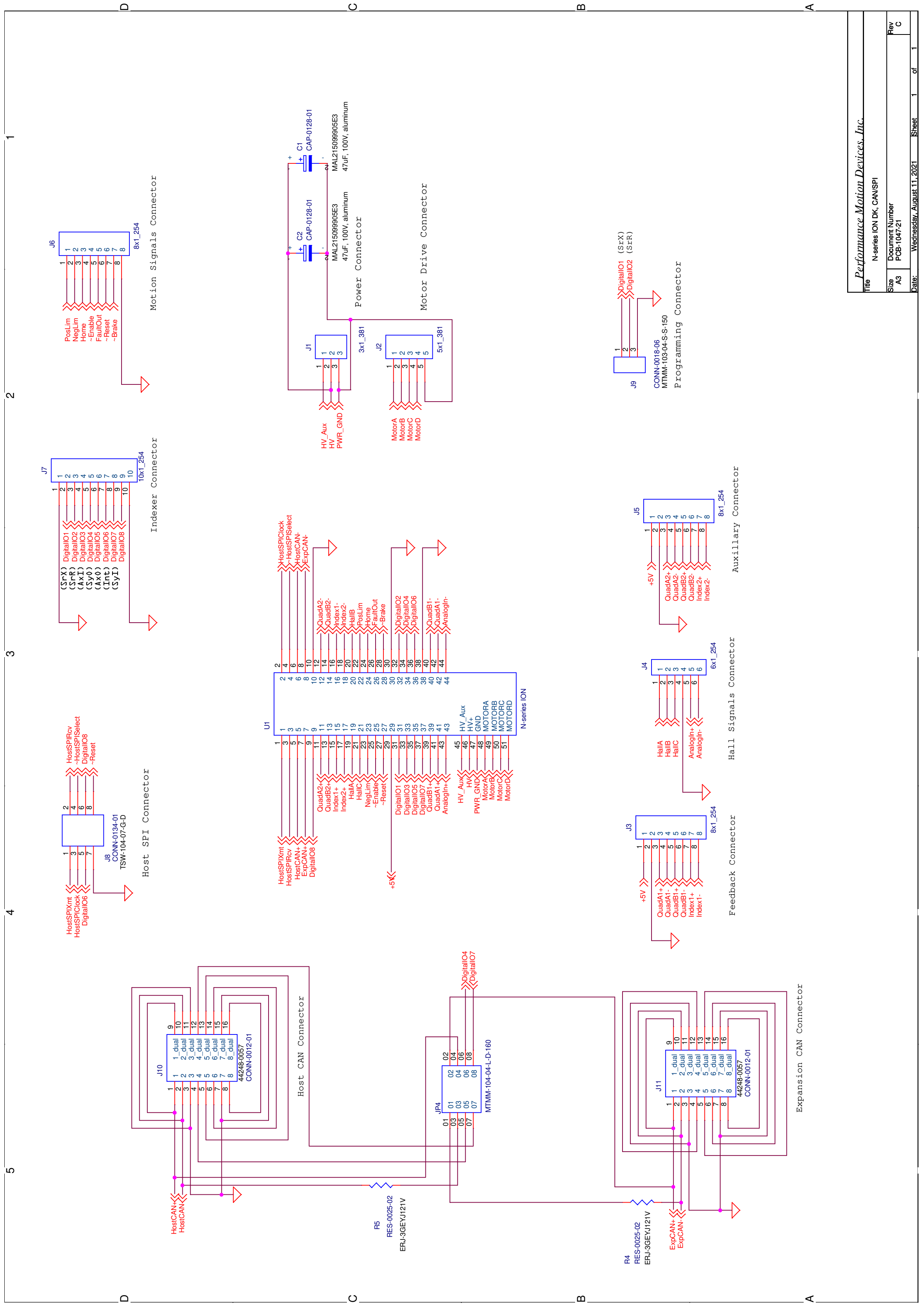
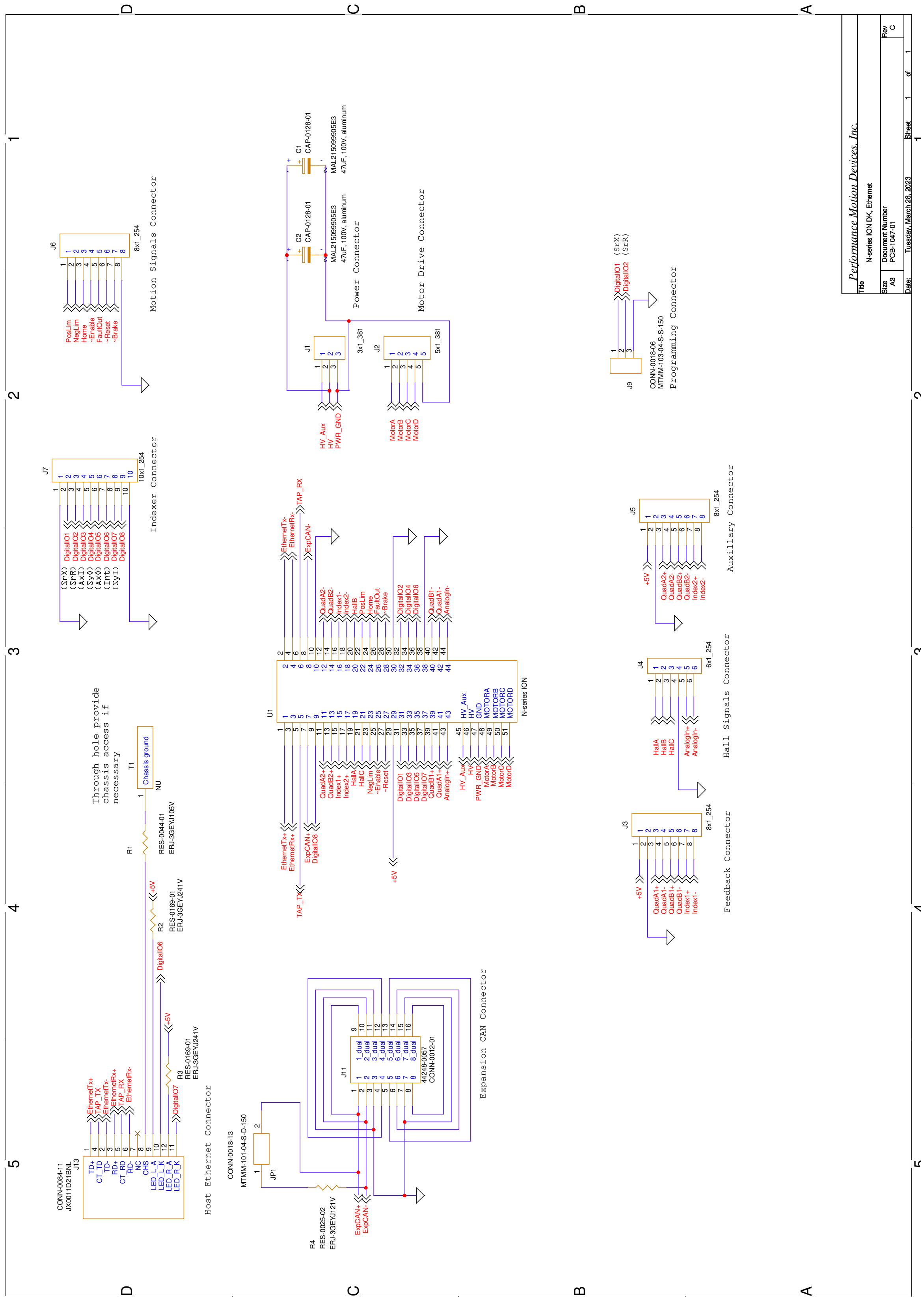


Figure 7-2: CAN/SPI Host Interface DK Interconnect Board Schematic

7.1.3 Ethernet Host Interface Type

Figure 7-3: Ethernet Host Interface DK Interconnect Board Schematic


Performance Motion Devices, Inc.
N-series ION DK, Ethernet
PCB-1047-01
Tuesday, March 28, 2023
Motion Signals Connector
Indexer Connector
Power Connector
Motor Drive Connector
Programming Connector
Auxillary Signals Connector
Hall Signals Connector
Feedback Connector
Expansion CAN Connector
Host Ethernet Connector
N-series ION
Through hole provide chassis access if necessary