



PMX-4EX-SA

Advanced 4-Axis Stepper Motion Controller



COPYRIGHT © 2018 NIPPON PULSE AMERICA, INC.,
ALL RIGHTS RESERVED

NIPPON PULSE AMERICA, INC. copyrights this document. You may not reproduce or translate into any language in any form and means any part of this publication without the written permission from NIPPON PULSE AMERICA, INC.

NIPPON PULSE AMERICA, INC. makes no representations or warranties regarding the content of this document. We reserve the right to revise this document any time without notice and obligation.

Firmware Compatibility:

†V180BL

†If your module's firmware version number is less than the listed value, contact Nippon Pulse for the appropriate documentation.

Table of Contents

1. Introduction	6
1.1. FEATURES	6
1.2. MODEL NUMBERS	7
1.2.1. <i>Top Board Options</i>	7
2. Electrical Specifications	8
3. Dimensions	9
3.1. PMX-4EX-SA-TBS DIMENSIONS	9
3.2. PMX-4EX-SA-TB9 DIMENSIONS	10
4. Connectivity	11
4.1. 2-PIN POWER CONNECTOR (5.08MM)	11
4.2. 3-PIN RS-485 CONNECTOR (3.81MM)	11
4.3. 10-PIN AIO CONNECTOR (2.0MM)	12
4.4. 10-PIN DI CONNECTOR (3.81MM).....	12
4.5. 8-PIN DO CONNECTOR (3.81MM).....	13
4.6. 14-PIN MOTION INPUTS CONNECTOR (3.81MM).....	14
4.7. 8-PIN ENCODER CONNECTORS (3.81MM)	15
4.8. 8-PIN TBS AXIS CONNECTORS (3.81MM)	16
4.9. 9-PIN TB9 AXIS CONNECTORS (D-SUB 9)	17
4.10. 2-PIN TB9 POWER CONNECTORS (5.08MM).....	18
4.11. PMX-4EX-SA INTERFACE CIRCUIT	19
4.12. PULSE, DIRECTION, AND ENABLE OUTPUTS	20
4.13. LIMIT, HOME, AND DIGITAL INPUTS	21
4.14. DIGITAL OUTPUTS.....	21
4.15. ENCODER INPUT CONNECTION.....	22
5. Communication Interface	23
5.1. USB COMMUNICATION	23
5.1.1. <i>Typical USB Setup</i>	23
5.1.2. <i>USB Communication API</i>	23
5.1.3. <i>USB Communication Issues</i>	24
5.2. SERIAL COMMUNICATION.....	25
5.2.1. <i>Typical RS-485 Setup</i>	25
5.2.2. <i>Communication Port Settings</i>	26
5.2.3. <i>ASCII Protocol</i>	26
5.2.4. <i>RS-485 Communication Issues</i>	27
5.3. DEVICE NUMBER.....	28
5.4. WINDOWS GUI.....	28
6. General Operation Overview	29
6.1. MOTION PROFILE	29
6.2. PULSE SPEED	31
6.3. ON-THE-FLY SPEED CHANGE	31
6.4. MOTOR POSITION	32
6.5. MOTOR POWER.....	32
6.6. JOG MOVE.....	33
6.7. STOPPING.....	33
6.8. POSITIONAL MOVES.....	34
6.9. ON-THE-FLY TARGET POSITION CHANGE.....	34
6.10. CIRCULAR INTERPOLATION MOVES.....	35
6.11. ARC INTERPOLATION MOVES	35
6.12. BUFFERED INTERPOLATION MOVES	37
6.13. HOMING.....	38
6.13.1. <i>MODE 0: Home Input Only (High Speed Only)</i>	38

6.13.2. <i>MODE 1: Limit Only</i>	39
6.13.3. <i>MODE 2: Home Input and Z-index</i>	40
6.13.4. <i>MODE 3: Z-index Only</i>	41
6.13.5. <i>MODE 4 : Home Input Only (High Speed and Low Speed)</i>	41
6.14. LIMITS AND ALARM SWITCH FUNCTION	42
6.15. MOTOR STATUS	43
6.16. DIGITAL INPUTS/OUTPUTS.....	44
6.16.1. <i>Digital Inputs</i>	44
6.16.2. <i>Digital Outputs</i>	44
6.17. HIGH SPEED LATCH INPUTS	45
6.18. SYNC OUTPUTS	46
6.19. ANALOG INPUTS.....	47
6.20. JOYSTICK CONTROL	48
6.21. POLARITY	51
6.22. STEPLOOP CLOSED LOOP CONTROL.....	51
6.23. TIMER REGISTER	54
6.24. COMMUNICATION TIME-OUT WATCHDOG.....	54
6.25. STANDALONE PROGRAM SPECIFICATION.....	54
6.25.1. <i>Standalone Program Specification</i>	54
6.25.2. <i>Standalone Control</i>	54
6.25.3. <i>Standalone Status</i>	55
6.25.4. <i>Standalone Subroutines</i>	55
6.25.5. <i>Error Handling</i>	55
6.25.6. <i>Standalone Variables</i>	56
6.25.7. <i>Standalone Run on Boot-Up</i>	56
6.26. STORING TO FLASH	57
7. Software Overview	58
7.1. MAIN CONTROL SCREEN	60
7.1.1. <i>Status</i>	60
7.1.2. <i>Control</i>	62
7.1.3. <i>On-The-Fly-Speed Control</i>	64
7.1.4. <i>On-The-Fly-Position Control</i>	65
7.1.5. <i>Sync Outputs</i>	65
7.1.6. <i>Digital Input/Output</i>	65
7.1.7. <i>Analog Inputs</i>	66
7.1.8. <i>Program File Control</i>	66
7.1.9. <i>Standalone Program Editor</i>	67
7.1.10. <i>Standalone Program Control</i>	67
7.1.11. <i>Standalone Program Compile/Download/Upload</i>	68
7.1.12. <i>Setup</i>	68
7.1.13. <i>Terminal</i>	70
7.1.14. <i>Latches</i>	71
7.1.15. <i>Variable Status</i>	71
7.2. DXF CONVERTER	73
7.2.1. <i>DXF Viewer</i>	74
7.2.2. <i>Status</i>	74
7.2.3. <i>Control</i>	75
7.2.4. <i>DXF Action</i>	75
7.2.5. <i>Motion Conversion Program</i>	76
7.2.6. <i>DXF Converter – Important Notes</i>	76
8. ASCII Language Specification	79
8.1. ASCII COMMAND SET	79

8.2. ERROR CODES.....	82
9. Standalone Language Specification.....	83
9.1. STANDALONE COMMAND SET.....	83
9.2. EXAMPLE STANDALONE PROGRAMS	87
9.2.1. <i>Standalone Example Program 1 – Single Thread</i>	87
9.2.2. <i>Standalone Example Program 2 – Single Thread</i>	87
9.2.3. <i>Standalone Example Program 3 – Single Thread</i>	88
9.2.4. <i>Standalone Example Program 4 – Single Thread</i>	88
9.2.5. <i>Standalone Example Program 5 – Single Thread</i>	89
9.2.6. <i>Standalone Example Program 6 – Single Thread</i>	90
9.2.7. <i>Standalone Example Program 7 – Multi Thread</i>	91
9.2.8. <i>Standalone Example Program 8 – Multi Thread</i>	92
A: Speed Settings	93
A.1. ACCELERATION/DECELERATION RANGE.....	93
A.2. ACCELERATION/DECELERATION RANGE – POSITIONAL MOVE	94

1. Introduction

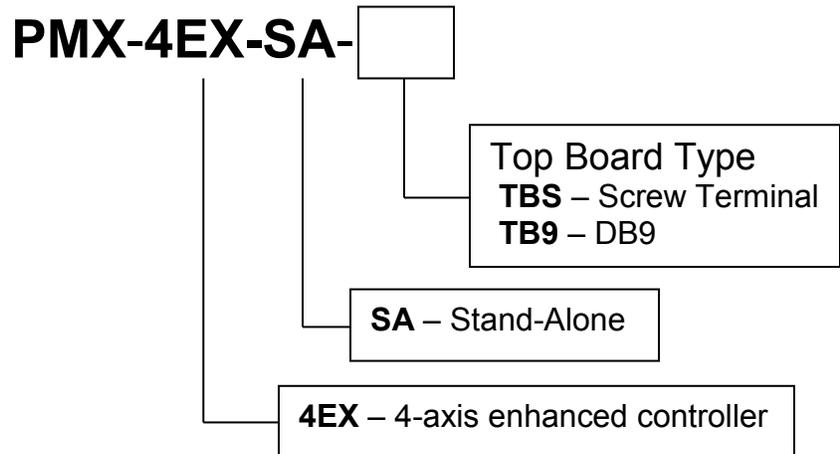
PMX-4EX-SA is an advanced 4 axis stepper standalone programmable motion controller.

Communication to the PMX-4EX-SA can be established over USB or RS-485. It is possible to download a standalone program to the device and have it run independent of a host.

1.1. Features

- USB 2.0 communication
- RS-485 ASCII communication
 - 9600, 19200, 38400, 57600, 115200 bps
- Standalone programmable using A-SCRIPT
- Pulse/Dir/Enable open collector outputs per axis
 - Maximum pulse output rate of 6M PPS
- Advanced Motion features
 - Trapezoidal or s-curve acceleration
 - On-the-fly speed change
 - XYZU linear coordinated motion
 - XY circular coordinated motion
 - XY arc coordinated motion
 - Continuous linear coordinated buffered move for XYZ axes for smooth contouring.
- A/B/Z differential encoder inputs [Max frequency of 5 MHz]
 - StepNLoop closed loop control (position verification)
- Opto-isolated I/O
 - 8 x inputs [4 x high speed position capture latch input]
 - 8 x outputs [4 x synchronous output]
 - +Limit/-Limit/Home inputs per axis
- Homing routines:
 - Home input only
 - Limit only
 - Z-index encoder channel only
 - Home input + Z index encoder channel
- 8 x 10-bit analog inputs
 - XYZU joystick control

1.2. Model Numbers



1.2.1. Top Board Options

The PMX-4EX-SA is available in two different top board configurations. The top board should be selected depending on your interfacing needs.

The standard top board option (TBS) consists of 3.81mm screw terminals for X/Y/Z/U pulse/dir/enable outputs and alarm inputs.

The DB9 top board consists of DB9 female headers for X/Y/Z/U pulse/dir/enable outputs.

Contacting Support

For technical support contact: info@nipponpulse.com.
Or, contact your local distributor for technical support.

2. Electrical Specifications

Parameter	Min	Max	Units
Main Power Input	+12	+24	V
	-	1.5	A
Opto-supply Power Input	+12	+24	V
Driver Power Input ¹	+12	+48	V
	-	3.0 ¹	A
Pulse/Direction/Enable Open Collector	-	+24	V
	-	40	mA
Alarm Input Forward Diode Current	-	40	mA
Digital Input Forward Diode Current	-	40	mA
Digital Output Source Current	-	90	mA
Operating Temperature ²	-20	+80	°C
Storage Temperature ²	-55	+150	°C

Table 2-1

¹Only applicable to the TB9 top board option. Current requirement is dependent on the driver settings

²Based on component ratings

3. Dimensions

3.1. PMX-4EX-SA-TBS Dimensions

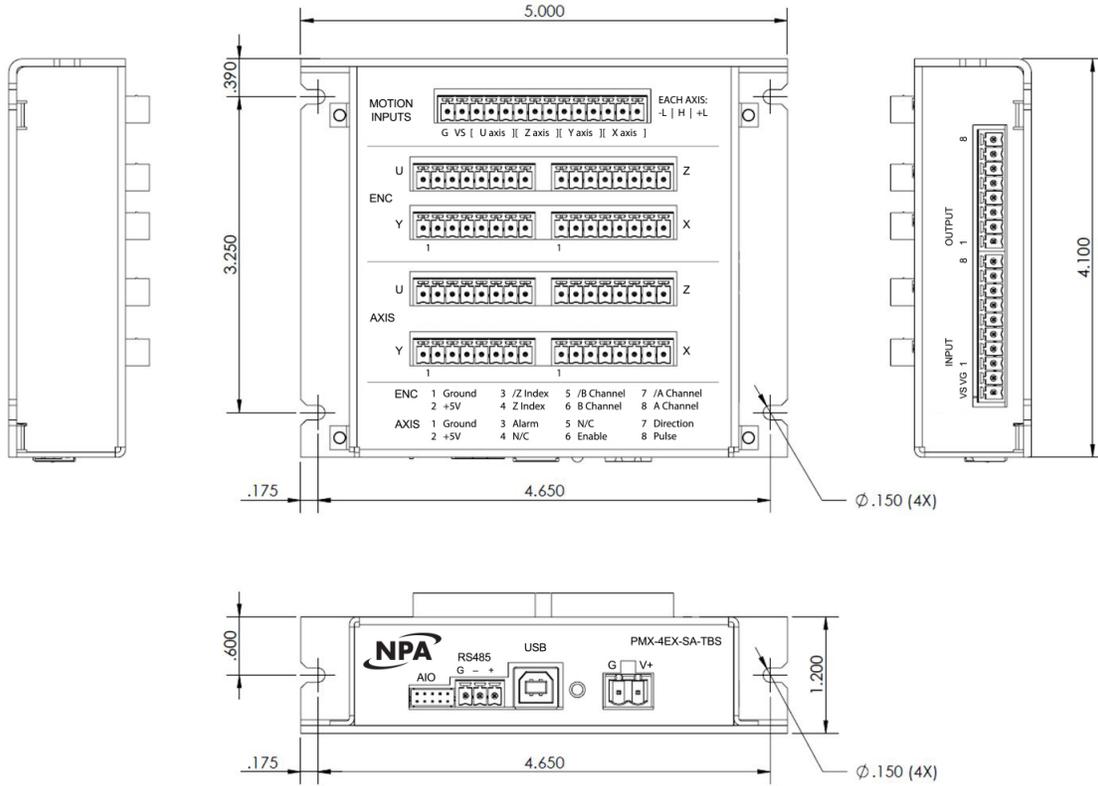


Figure 3-1

3.2. PMX-4EX-SA-TB9 Dimensions

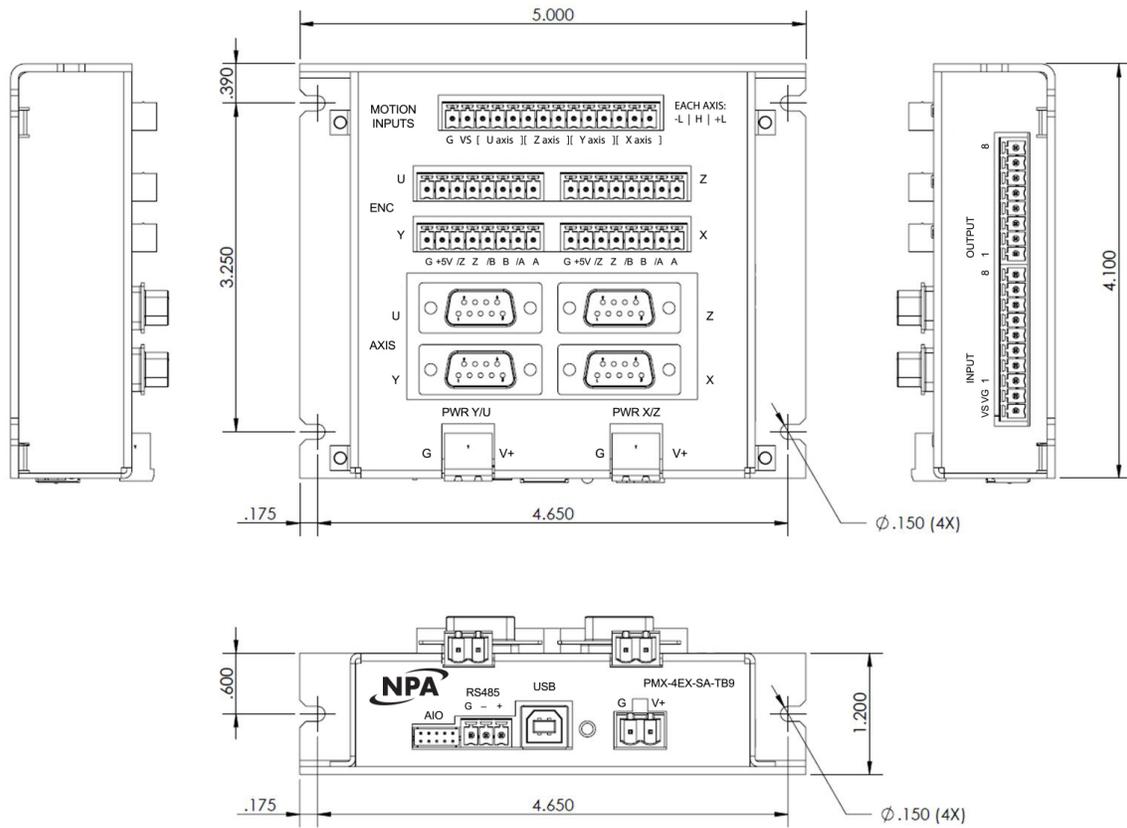


Figure 3-2

4. Connectivity

In order for PMX-4EX-SA to operate, it must be supplied with +12VDC to +24VDC. Power pins as well as communication port pin outs are shown below.

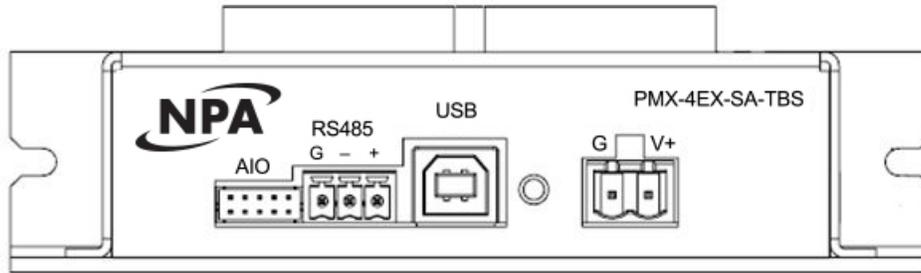


Figure 4-1

4.1. 2-Pin Power Connector (5.08mm)

Pin #	In/Out	Name	Description
1	I	G	Ground
2	I	V+	Power Input +12 to +24 VDC

Table 4-1

Mating Connector Description: 2 pin 0.2" (5.08mm) connector
 Mating Connector Manufacturer: On-Shore
 Mating Connector Manufacturer Part: 1EDZ950/2

1 Other 5.08mm compatible connectors can be used.

4.2. 3-Pin RS-485 Connector (3.81mm)

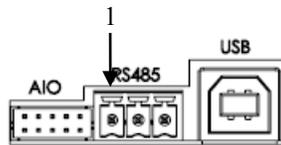


Figure 4-2

Pin #	In/Out	Name	Description
1	I	G	Ground
2	I/O	-	RS-485 minus signal
3	I/O	+	RS-485 plus signal

Table 4-2

Mating Connector Description: 3 pin 0.15" (3.81mm) connector
 Mating Connector Manufacturer: On-Shore
 Mating Connector Manufacturer Part: 1EDZ1550/3

1 Other 3.81 compatible connectors can be used.

4.3. 10-Pin AIO Connector (2.0mm)

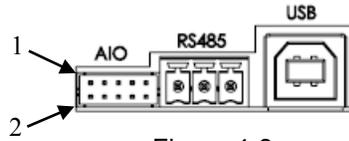


Figure 4-3

Pin #	In/Out	Name	Description
1	O	+5V	+5V
2	O	GND	Ground
3	I	AI7	Analog Input 7
4	I	AI8	Analog Input 8
5	I	AI5	Analog Input 5
6	I	AI6	Analog Input 6
7	I	AI3	Analog Input 3
8	I	AI4	Analog Input 4
9	I	AI1	Analog Input 1
10	I	AI2	Analog Input 2

Table 4-3

Mating Connector Description: Female 10 pin 2mm dual row
 Mating Connector Manufacturer: HIROSE
 Mating Connector Manufacturer Part: DF11-10DS-2C (10 pin female connector)
 DF11-2428SC (female socket pin)

4.4. 10-Pin DI Connector (3.81mm)

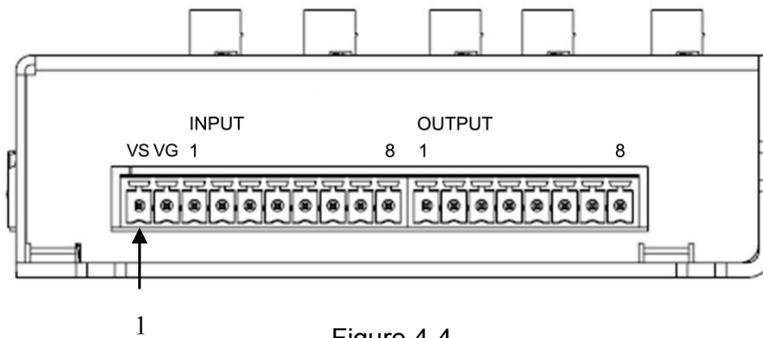


Figure 4-4

Pin #	In/Out	Name	Description
1	I	Vs	Opto-Supply +12V to +24 VDC (for digital IO)
2	I	Vg	Opto-Ground (for digital IO)
3	I	DI1	Digital Input 1
4	I	DI2	Digital Input 2
5	I	DI3	Digital Input 3
6	I	DI4	Digital Input 4
7	I	DI5	Digital Input 5

8	I	DI6	Digital Input 6
9	I	DI7	Digital Input 7
10	I	DI8	Digital Input 8

Table 4-4

Mating Connector Description: 10 pin 0.15" (3.81mm) connector
Mating Connector Manufacturer: On-Shore
Mating Connector Manufacturer Part: 1EDZ1550/10

1Other 3.81 compatible connectors can be used.

4.5. 8-Pin DO Connector (3.81mm)

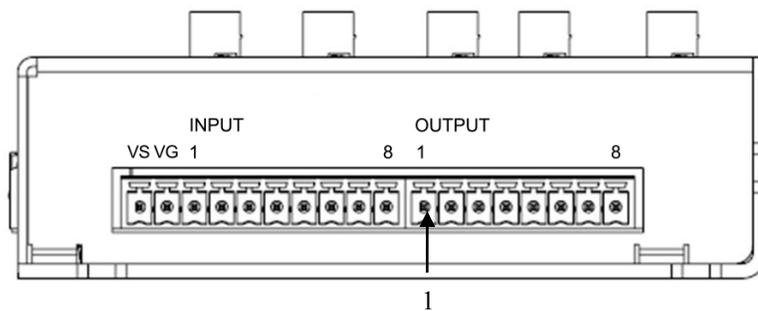


Figure 4-5

Pin #	In/Out	Name	Description
1	O	DO1	Digital Output 1
2	O	DO2	Digital Output 2
3	O	DO3	Digital Output 3
4	O	DO4	Digital Output 4
5	O	DO5	Digital Output 5
6	O	DO6	Digital Output 6
7	O	DO7	Digital Output 7
8	O	DO8	Digital Output 8

Table 4-5

Mating Connector Description: 8 pin 0.15" (3.81mm) connector
Mating Connector Manufacturer: On-Shore
Mating Connector Manufacturer Part: 1EDZ1550/8

1Other 3.81 compatible connectors can be used.

4.6. 14-Pin Motion Inputs Connector (3.81mm)

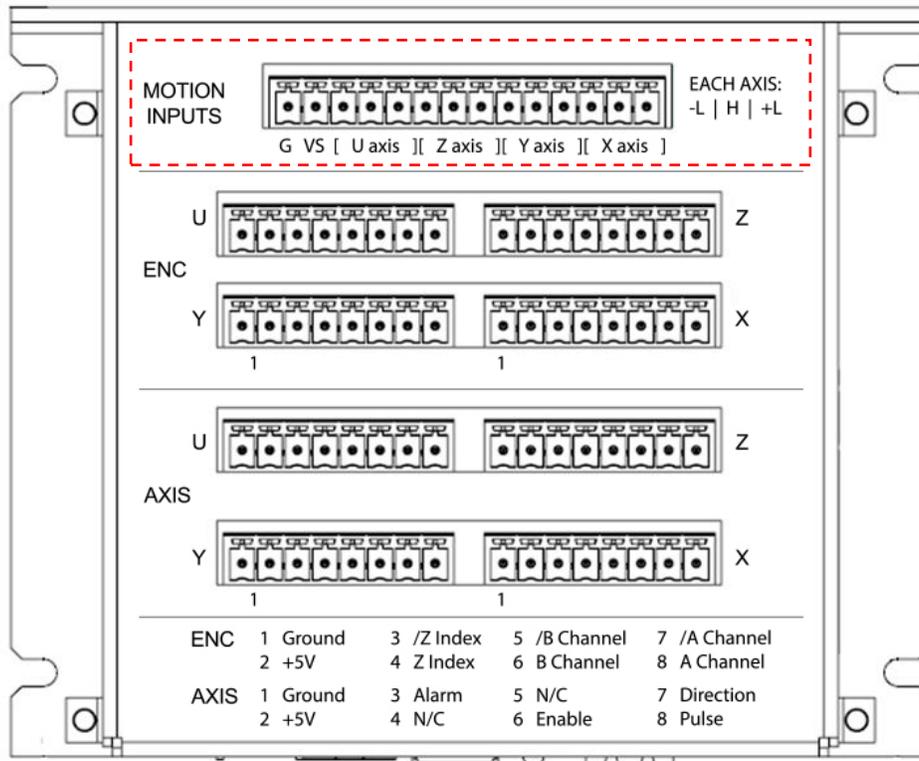


Figure 4-6

Pin #	In/Out	Name	Description
1	I	G	Ground
2	I	Vs	Opto-Supply Input +12 to +24 VDC (for limit and home inputs)
3	I	-L	-Limit [U Axis]
4	I	H	Home [U Axis]
5	I	+L	+Limit [U Axis]
6	I	-L	-Limit [Z Axis]
7	I	H	Home [Z Axis]
8	I	+L	+Limit [Z Axis]
9	I	-L	-Limit [Y Axis]
10	I	H	Home [Y Axis]
11	I	+L	+Limit [Y Axis]
12	I	-L	-Limit [X Axis]
13	I	H	Home [X Axis]
14	I	+L	+Limit [X Axis]

Table 4-6

Mating Connector Description: 14 pin 0.15" (3.81mm) connector
 Mating Connector Manufacturer: On-Shore
 Mating Connector Manufacturer Part: 1EDZ1550/14

1Other 3.81 compatible connectors can be used.

4.7. 8-Pin Encoder Connectors (3.81mm)

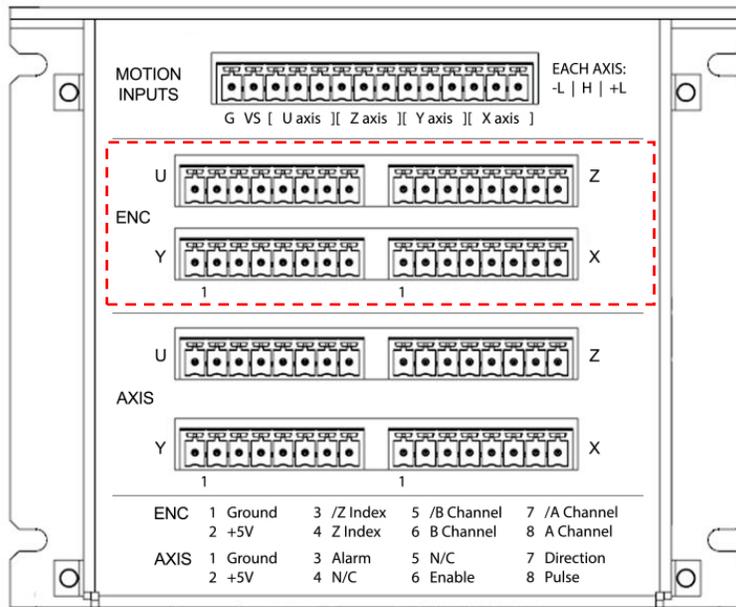


Figure 4-7

Pin #	In/Out	Name	Description
1	O	G	Ground
2	O	+5V	+5V
3	I	/Z	/Z Index Encoder Input
4	I	Z	Z Index Encoder Input
5	I	/B	/B Channel Encoder Input
6	I	B	B Channel Encoder Input
7	I	/A	/A Channel Encoder Input
8	I	A	A Channel Encoder Input

Table 4-7

Mating Connector Description: 8 pin 0.15" (3.81mm) connector
 Mating Connector Manufacturer: On-Shore
 Mating Connector Manufacturer Part: 1EDZ1550/8

1Other 3.81 compatible connectors can be used.

4.8. 8-Pin TBS Axis Connectors (3.81mm)

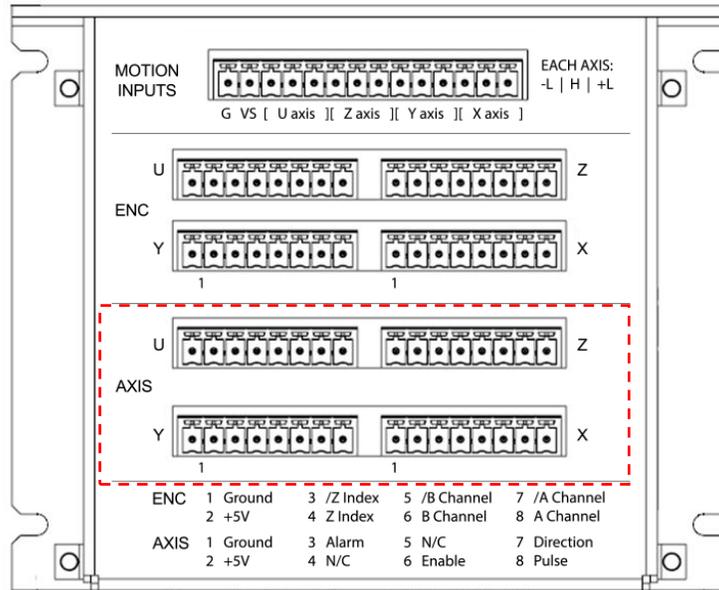


Figure 4-8

Pin #	In/Out	Name	Description
1	I	G	Ground
2	O	+5V	+5V
3	I	AI	Alarm
4	NC	NC	No Connection
5	NC	NC	No Connection
6	O	E	Enable
7	O	D	Direction
8	O	P	Pulse

Table 4-8

Mating Connector Description: 8 pin 0.15" (3.81mm) connector
Mating Connector Manufacturer: On-Shore
Mating Connector Manufacturer Part: 1EDZ1550/8

1Other 3.81 compatible connectors can be used.

4.9. 9-Pin TB9 Axis Connectors (D-SUB 9)

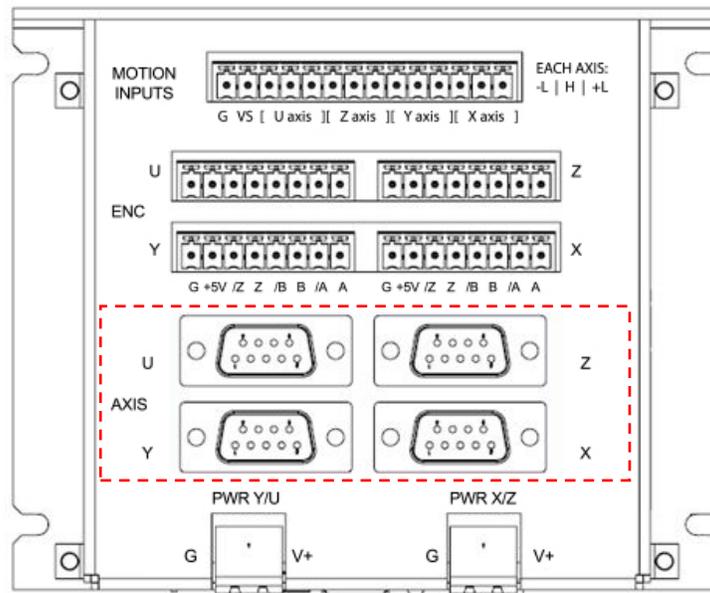


Figure 4-9

Pin #	In/Out	Name	Description
1	O	V+	Driver Power
2	O	P	Pulse
3	O	E	Enable
4	I	ALM	Alarm Input (5V TTL)
5	NC	NC	Reserved. Do not make connection.
6	O	G	Driver Ground
7	O	D	Direction
8	NC	NC	Reserved. Do not make connection.
9	O	5V	+5V

Table 4-9

The pins on the DB9 headers can be connected directly to a DMX-A2-DRV module (pin-to-pin compatible).

4.10. 2-Pin TB9 Power Connectors (5.08mm)

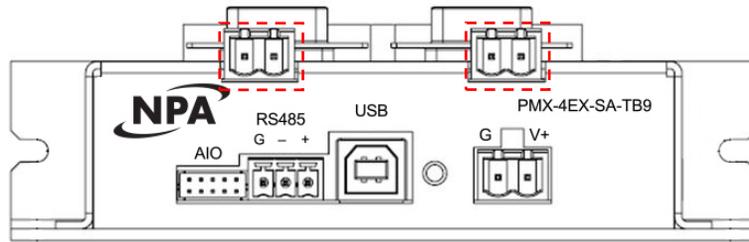


Figure 4-10

Pin #	In/Out	Name	Description
1	I	G	Driver Ground
2	I	V+	Driver Power

Table 4-10

Mating Connector Description: 2 pin 0.2" (5.08mm) connector
 Mating Connector Manufacturer: On-Shore
 Mating Connector Manufacturer Part: 1EDZ950/2

1 Other 5.08mm compatible connectors can be used.

There are two separate driver power inputs. The right side is for axes X + Z. The left side is for axes Y + U. The power requirement will be dependent on the driver specifications for each axis.

4.11. PMX-4EX-SA Interface Circuit

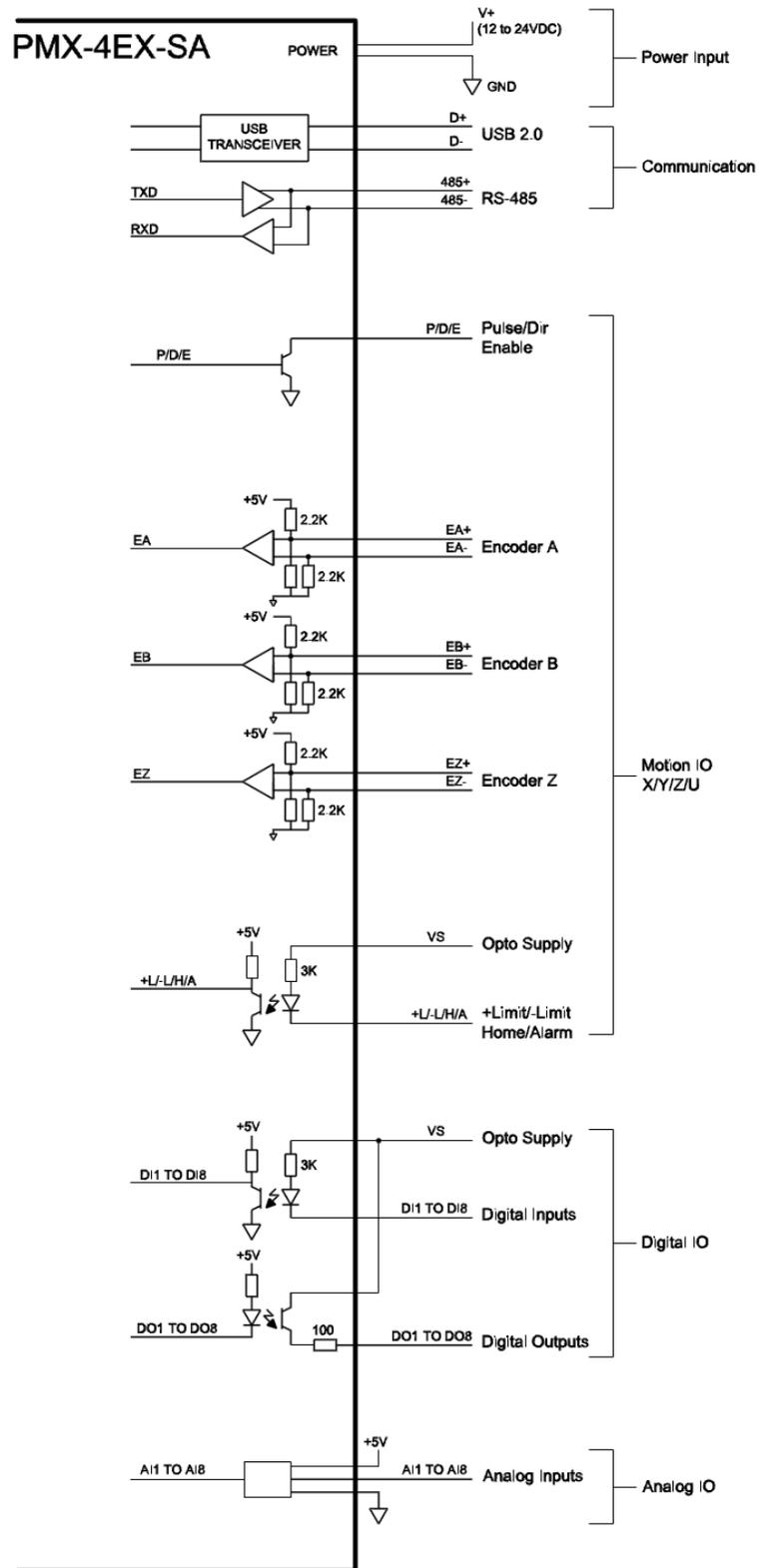


Figure 4-11

4.12. Pulse, Direction, and Enable Outputs

The Pulse, Direction, and Enable outputs for both the standard and DB9 top boards are all open collector outputs. Figure 4-12 shows the detailed schematic of these outputs. Each output is capable of sinking up to 40mA of current.

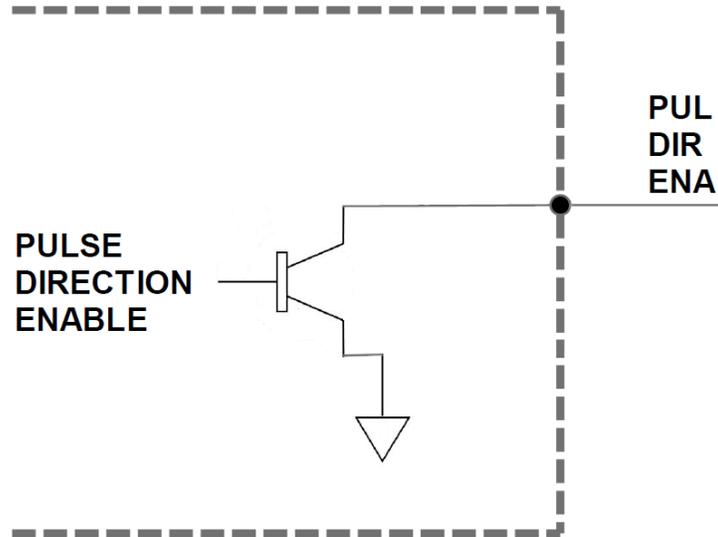


Figure 4-12

Figure 4-13 shows an example wiring diagram between the pulse, direction, and enable outputs on the PMX-4EX-SA and the corresponding input on a typical stepper driver.

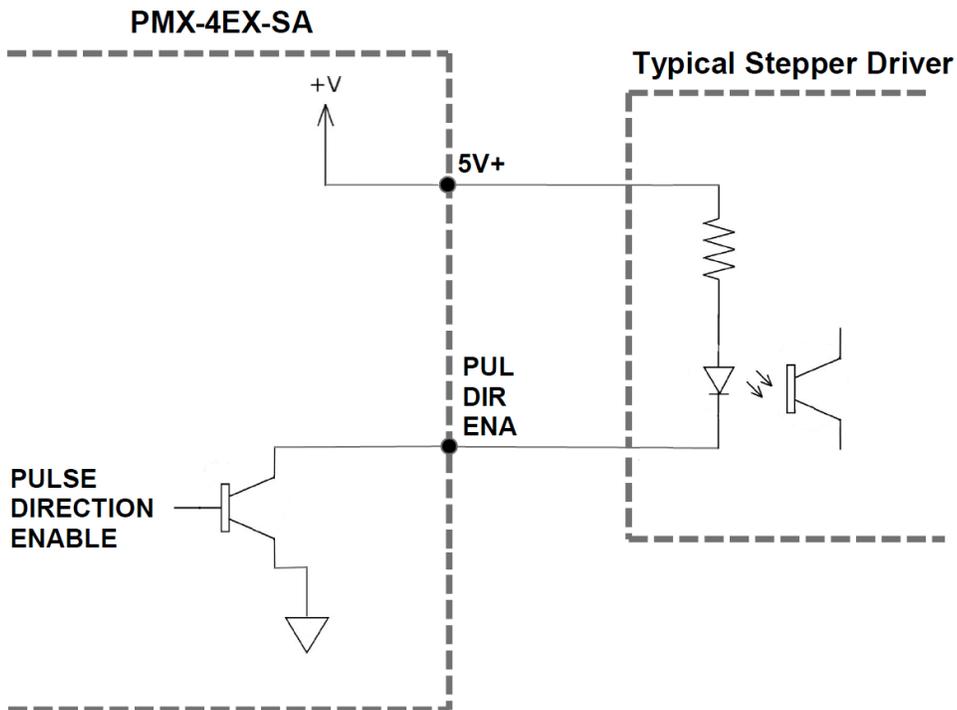


Figure 4-13

4.13. Limit, Home, and Digital Inputs

Figure 4-14 shows the detailed schematic of the opto-isolated limit, home, and general purpose digital inputs. All opto-isolated digital inputs are NPN type.

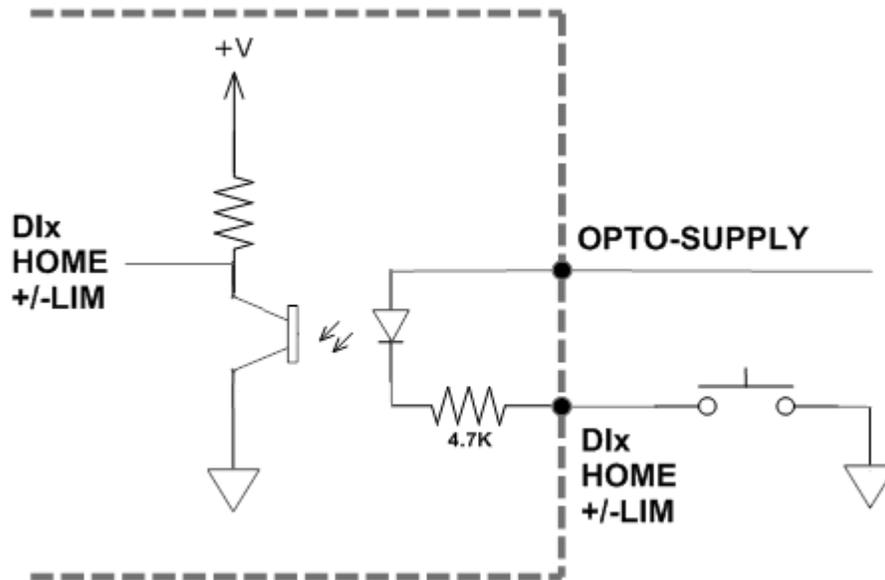


Figure 4-14

The opto-supply must be connected to +12 to +24VDC in order for the limit, home, and digital inputs to operate.

When the digital input is pulled to ground, current will flow from the opto-supply to ground, turning on the opto-isolator and activating the input.

To de-activate the input, the digital input should be left unconnected or pulled up to the opto-supply, preventing current from flowing through the opto-isolator.

4.14. Digital Outputs

Figure 4-15 shows an example wiring of the digital outputs. All opto-isolated digital outputs will be PNP type.

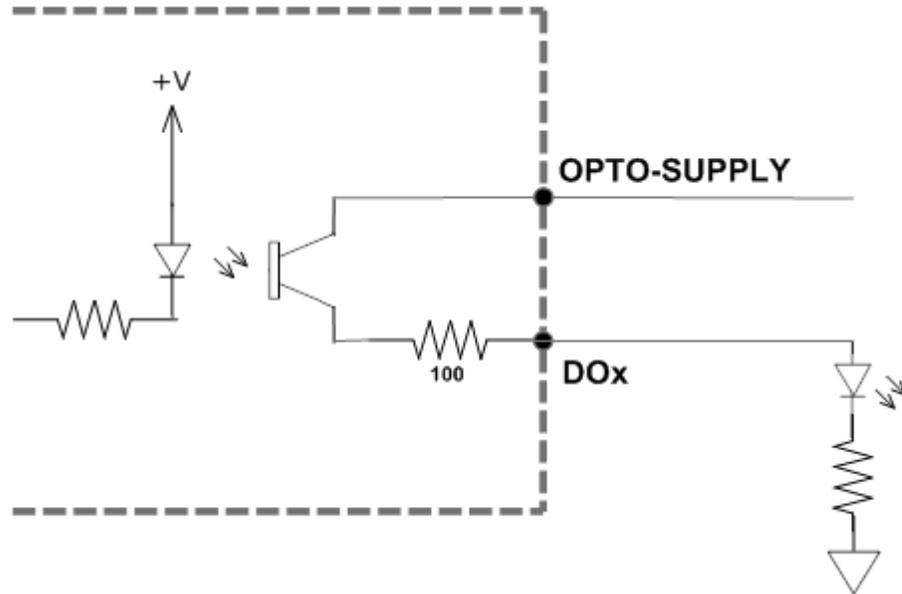


Figure 4-15

The opto-supply must be connected to +12 to +24VDC in order for the digital outputs to operate.

When activated, the opto-isolator for the digital output pulls the voltage on the digital output line to the opto-supply. The maximum sink current for digital outputs is 45mA. Take caution to select the appropriate external resistor so that the current does not exceed 45mA.

When deactivated, the opto-isolator will break the connection between the digital output and the opto-supply.

4.15. Encoder Input Connection

Both single-ended and differential quadrature encoder inputs are accepted.

When using single-ended encoders, use the /A, /B, and /Z inputs.

+5V supply and Ground signals are available to power the encoder. Make sure that the total current usage is less than 200mA for the +5V.

The maximum encoder frequency is 5MHz.

5. Communication Interface

5.1. USB Communication

PMX-4EX-SA USB communication is USB 2.0 compliant.

In order to communicate with PMX-4EX-SA via USB, the proper software driver must be first installed. Before connecting the PMX-4EX-SA 4-axis controller, or running any programs, please go to the Nippon Pulse web site, download the Drivers and Tools Setup, and run the installation.

All USB communication will be done using an ASCII command protocol.

5.1.1. Typical USB Setup

The PMX-4EX-SA can be connected to a PC directly via USB or through a USB hub. All USB cables should have a noise suppression choke to avoid communication loss or interruption. See a typical USB network setup in Figure 5-1.

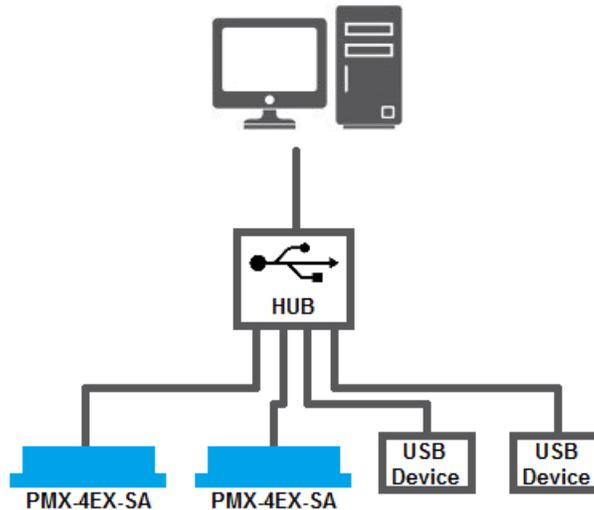


Figure 5-1

5.1.2. USB Communication API

Communication between the PC and PMX-4EX-SA is done using the Windows compatible DLL API function calls shown below. Windows programming languages such as Visual BASIC, Visual C++, LabView, or any other programming language that can use a DLL can be used to communicate with the PMX-4EX-SA.

Typical communication transaction time between PC and PMX-4EX-SA for sending a command from a PC and getting a reply from the controller using the **fnPerformaxComSendRecv()** API function is in single digit milliseconds. This value will vary with CPU speed of PC and the type of command.

For USB communication, following DLL API functions are provided.

BOOL fnPerformaxComGetNumDevices(OUT LPDWORD lpNumDevices);

- This function is used to get total number of all types of Performax and Performax USB modules connected to the PC.

BOOL fnPerformaxComGetProductString(IN DWORD dwNumDevices,
OUT LPVOID lpDeviceString,
IN DWORD dwOptions);

- This function is used to get the Performax or Performax product string. This function is used to find out Performax USB module product string and its associated index number. Index number starts from 0.

BOOL fnPerformaxComOpen(IN DWORD dwDeviceNum,
OUT HANDLE* pHandle);

- This function is used to open communication with the Performax USB module and to get communication handle. dwDeviceNum starts from 0.

BOOL fnPerformaxComClose(IN HANDLE pHandle);

- This function is used to close communication with the Performax USB module.

BOOL fnPerformaxComSetTimeouts(IN DWORD dwReadTimeout,
DWORD dwWriteTimeout);

- This function is used to set the communication read and write timeout. Values are in milliseconds. This must be set for the communication to work. Typical value of 1000 msec is recommended.

BOOL fnPerformaxComSendRecv(IN HANDLE pHandle,
IN LPVOID wBuffer,
IN DWORD dwNumBytesToWrite,
IN DWORD dwNumBytesToRead,
OUT LPVOID rBuffer);

- This function is used to send commands and receive replies. The number of bytes to read and write must be 64 characters.

BOOL fnPerformaxComFlush(IN HANDLE pHandle)

- Flushes the communication buffer on the PC as well as the USB controller. It is recommended to perform this operation right after the communication handle is opened.

5.1.3. USB Communication Issues

A common problem that users may have with USB communication is that after sending a command from the PC to the device, the response is not received by the PC until another command is sent. In this case, the data buffers between the

PC and the USB device are out of sync. Below are some suggestions to help alleviate this issue.

- 1) **Buffer Flushing:** If USB communication begins from an unstable state (i.e. your application has closed unexpectedly), it is recommended to first flush the USB buffers of the PC and the USB device. See the following function prototype below:

BOOL ***fnPerformaxComFlush***(IN HANDLE pHandle)

Note: *fnPerformaxComFlush* is only available in the most recent PerformaxCom.dll which is not registered by the standard USB driver installer. A sample of how to use this function along with this newest DLL is available for download on the website

- 2) **USB Cable:** Another source of USB communication issues may come from the USB cable. Confirm that the USB cable being used has a noise suppression choke. See Figure 5-2.



Figure 5-2

5.2. Serial Communication

The PMX-4EX-SA has the ability to communicate over an RS-485 interface using an ASCII protocol. An RS-485 serial port on the PC or PLC can be used to communicate with the PMX-4EX-SA. A USB to RS-485 converter can also be used.

5.2.1. Typical RS-485 Setup

A typical RS-485 network is shown in Figure 5-3. Several techniques can be used to increase the robustness of an RS-485 network.

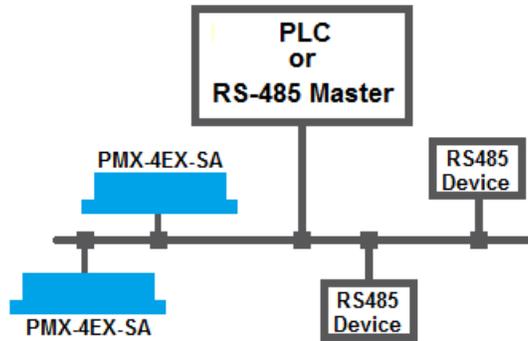


Figure 5-3

5.2.2. Communication Port Settings

The PMX-4EX-SA has the communication port settings shown in Table 5-1.

Parameter	Setting
Byte Size	8 bits
Parity	None
Flow Control	None
Stop Bit	1

Table 5-1

PMX-4EX-SA provides the user with the ability to set the desired baud rate of the serial communication. In order to make these changes, first set the desired baud rate by using the **DB** command.

Return Value	Description
1	9600
2	19200
3	38400
4	57600
5	115200

Table 5-2

To write the values to the device's flash memory, use the **STORE** command. After a complete power cycle, the new baud rate will be written to memory. Note that until a power cycle is completed, the settings will not take effect.

By default, the PMX-4EX-SA has a baud rate setting of 9600 bps.

5.2.3. ASCII Protocol

The following ASCII protocol should be used for sending commands and receiving replies from the PMX-4EX-SA. Details on valid ASCII command can be found in section 8.

Sending Command

ASCII command string in the format of
@[DeviceName][ASCII Command][CR]

[CR] character has ASCII code 13.

Receiving Reply

The response will be in the format of
[Response][CR]

[CR] character has ASCII code 13.

Examples:

For querying the polarity
Send: @00POX[CR]
Reply: 7[CR]

For reading the digital input status
Send: @00DI[CR]
Reply: 8[CR]

For performing a store to flash memory
Send: @00STORE[CR]
Reply: OK[CR]

5.2.4. RS-485 Communication Issues

RS-485 communication issues can arise due to noise on the RS-485 bus. The following techniques can be used to help reduce noise issues.

Daisy Chaining

For a multi-drop RS-485 network, be sure that the network uses daisy-chain wiring. Figure 5-3 shows an example of a daisy chain network.

Number of Nodes

The maximum number of nodes recommended is 32. Increasing beyond this number will require special attention

Twisted Pair Wiring

To reduce noise, it is recommended to use twisted pair wiring for the 485+ and 485- lines. This technique will help cancel out electromagnetic interference.

Termination

For an RS-485 network, it may be required that a 120 Ohm resistor is placed in between the 485+ and 485- signals, at the beginning and end of the bus. A terminal resistor will help eliminate electrical reflections on the RS-485 network.

Note that on short communication buses, or buses with a small number of nodes, termination resistors may not be needed. Inclusion of terminal resistors when they are not needed may mask the main signal entirely.

5.3. Device Number

If multiple PMX-4EX-SA devices are connected to the PC, each device should have a unique device number. This will allow the PC to differentiate between multiple controllers. This is applicable for both USB and RS-485 communication types. In order to make this change to a PMX-4EX-SA, first store the desired number using the **DN** command. Note that this value must be within the range [4EX00,4EX99].

For example, to change the device number, the command **DN=4EX02** can be sent. The device name can also be changed through the *Setup* window of the standard PMX-4EX-SA software.

By default, all PMX-4EX-SA start with device number **4EX00**.

To save a modified device number to the flash memory of the PMX-4EX-SA, use the **STORE** command. After a power cycle, the new device number will be used. Note that before a power cycle is completed, the settings will not take effect.

5.4. Windows GUI

PMX-4EX-SA comes with a Windows GUI program to test, program, compile, download, and debug the controller. The Windows GUI will perform all communication via USB.

6. General Operation Overview

Important Note: All the commands described in this section are defined as ASCII or standalone commands. ASCII commands are used when communicating over USB. Standalone commands are used when writing a standalone program onto the PMX-4EX-SA.

6.1. Motion Profile

By default, the PMX-4EX-SA uses trapezoidal velocity profile as shown in Figure 6-1.

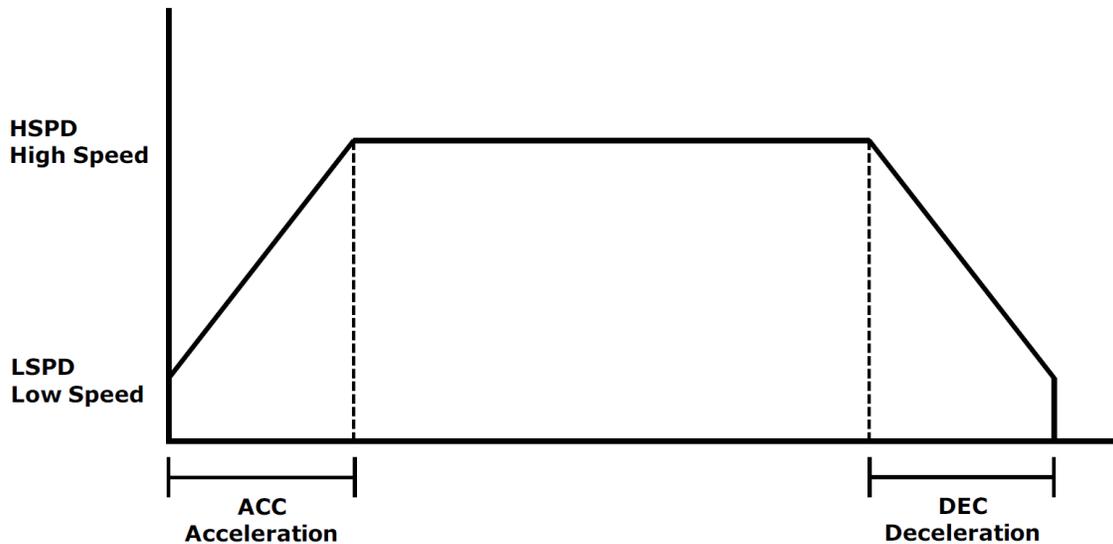


Figure 6-1

S-curve velocity profile can also be achieved by using the **SCV[axis]** command. Setting this command to 1 will enable s-curve for the indicated axis. See Figure 6-2 for details.

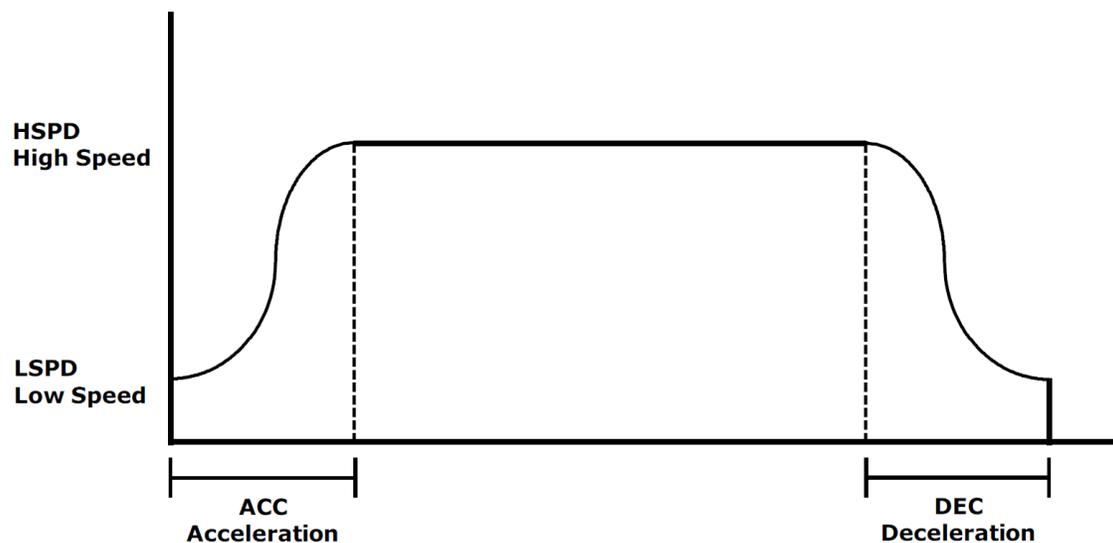


Figure 6-2

Once a typical move is issued, the axis will immediately start moving at the low speed setting and accelerate to the high speed. Once at high speed, the motor will move at a constant speed until it decelerates from high speed to low speed and immediately stops.

High speed and low speed are in pps (pulses/second). Use ASCII commands **HS[axis]** and **LS[axis]** to set/get individual high speed and low speed settings. In standalone mode, the **HSPD[axis]** and **LSPD[axis]** commands should be used. To set/get the global high speed and low speed values use the ASCII command **HS** and **LS** or the standalone commands **HSPD** and **LSPD**.

Acceleration and deceleration times are in milliseconds. Use the **ACC[axis]** command to set/get individual acceleration values. To set/get the global acceleration value, use the **ACC** command. Similarly, the **DEC** and **DEC[axis]** commands can be used to set/get the deceleration value.

The **EDEC** command can be used if the acceleration and deceleration are symmetrical. Set this command to 0 to use the **ACC** and **ACC[axis]** commands for both the acceleration and deceleration.

The minimum and maximum acceleration values depend on the high speed and low speed settings. Refer to Table A-1 and Figure A-1 in **Appendix A** for details.

By default, moves made by a single axis use global speed settings defined by **HS**, **LS**, **ACC**, and **DEC**. However, if a non-zero value is written to an individual speed setting, it will take priority over the global speed and will be used for single axis movements. Consider the example below.

The settings below will set both the global speed settings as well as the speed setting for the X axis.

```

HS=10000
HSX=2000
LS=300
ACCX=500
ACC=300
DEC=300

```

Once a move command is issued, the global speed settings for the low speed and deceleration while using the individual X-axis speed settings for the high speed and acceleration.

ASCII	HS	LS	ACC	DEC	HSn	LSn	ACCn	DECn	EDEC	SCV
Standalone	HSPD	LSPD	ACC	DEC	HSPDn	LSPDn	ACCn	DECn	-	-

6.2. Pulse Speed

The current pulse rate can be read using the ASCII command **PS** or the standalone command **PS[axis]**. For units, see Table 6-1.

Operation Mode	Speed Units
StepNLoop disabled	Pulse / sec
ALL interpolated moves	Pulse / sec
StepNLoop enabled and non-interpolated move	Encoder counts / sec

Table 6-1

Note that the ASCII command returns the current speed of all axes. The return value will have the following format.

[Speed X]:[Speed Y]:[Speed Z]:[Speed U]

The standalone command will return the speed of the indicated axis. See section 6.22 for more information regarding StepNLoop.

ASCII	PS
Standalone	PS[axis]

6.3. On-The-Fly Speed Change

An on-the-fly speed change can be achieved at any point while the motor is in motion. In order to perform an on-the-fly speed change, s-curve velocity profile must be disabled.

Before an on-the-fly speed change is performed, the correct speed window must be selected. To select a speed window, use the **SSPDM[axis]** command. Choosing the correct speed window will depend on the initial target speed and the final target speed. Both speeds will need to be within the same speed window.

The speed window must be set while the motor is idle. Refer to **Appendix A** for details on the speed windows.

Once the speed window has been set, an on-the-fly speed change can occur anytime the motor is in motion. The **SSPD[axis]=[speed]** command can be used to perform the actual speed change. For non on-the-fly speed change moves, set the speed window to 0.

ASCII	SSPDM[axis]	SSPD[axis]
Standalone	SSPDM[axis]	SSPD[axis]

6.4. Motor Position

The PMX-4EX-SA has a 32 bit signed step position counter for each axis. Range of the position counter is from $-2,147,483,648$ to $2,147,483,647$. Motor positions can be read using the ASCII command **PP**, which returns the pulse position of all 4 axes. The return value has the following format:

[Pulse X]:[Pulse Y]:[Pulse Z]:[Pulse U]

The pulse position of each axis can also be accessed individually. To manually set/get the pulse position of an individual axis, use the **P[axis]** command.

Similarly, the PMX-4EX-SA also has a 32 bit signed encoder position counter for each axis. Encoder positions can be read using ASCII command **PE**, which returns the encoder position of all 4 axes. Encoders are set to 4X reading. The return value has the following format:

[Encoder X]:[Encoder Y]:[Encoder Z]:[Encoder U]

To manually set/get the encoder position of an individual axis, use the **E[axis]** command.

When StepNLoop closed-loop control is enabled, the encoder counter commands return the encoder position and the pulse position commands return the real-time target position of the motor.

When StepNLoop closed-loop control is disabled, the encoder counter commands return the encoder position and the pulse position commands return the step position. See section 6.22 for details on the StepNLoop feature.

ASCII	PP	PE	P[axis]	E[axis]
Standalone	-	-	P[axis]	E[axis]

6.5. Motor Power

The **EO** command can be used to enable or disable the current to the motor. The effect of the enable output signals will depend on the characteristics of the motor drive. The enable output value must be within the range of 0-15.

Enable output values can also be referenced one bit at a time by the **EO[1-4]** commands. Note that the indexes are 1-based for the bit references (i.e. EO1 refers to bit 0, not bit 1).

Bit	Description	Bit-Wise Command
0	Enable Output 1 [X-axis]	EO1
1	Enable Output 2 [Y-axis]	EO2
2	Enable Output 3 [Z-axis]	EO3
3	Enable Output 4 [U-axis]	EO4

Table 6-2

The initial state of the enable outputs can be defined by setting the **EOBOOT** register to the desired initial enable output value. The value is stored to flash memory once the **STORE** command is issued.

ASCII	EO	EO[1-4]	EOBOOT
Standalone	EO	EO[1-4]	-

6.6. Jog Move

A jog move is used to continuously move the motor without stopping. Use the **J[axis]+/J[axis]-** command when operating in ASCII mode and the **JOG[axis]+/JOG[axis]-** in standalone mode. Once this move is started, the motor will only stop if a limit input is activated during the move or a stop command is issued.

If a motion command is sent while the controller is already moving, the command is not processed. Instead, an error response is returned. See Table 8-2 for details on error responses.

ASCII	J[axis][+/-]
Standalone	JOG[axis][+/-]

6.7. Stopping

When the motor is performing any type of move, motion can be stopped abruptly or with deceleration. It is recommended to use decelerated stops so that there is less impact on the system. The **ABORT[axis]** command will immediately stop an individual axis. Use the **ABORT** command to immediately stop ALL axes.

To employ deceleration on a stop, use the **STOP[axis]** command to stop an individual axis. Use the **STOP** command to stop ALL axes.

If an interpolation operation is in process when a **STOP[axis]** or **ABORT[axis]** command is entered, all axes involved in the interpolation operation will stop.

ASCII	ABORT	ABORT[axis]	STOP	STOP[axis]
Standalone	ABORT	ABORT[axis]	STOP	STOP[axis]

6.8. Positional Moves

The PMX-4EX-SA can perform positional moves for individual axis. Multiple axis can also be commanded to move to a specified position in order to perform linear coordinated motion.

The PMX-4EX-SA can perform positional moves in absolute or incremental mode. For absolute mode, the **ABS** command should be used and, for incremental mode, the **INC** command should be used. These commands should be sent before the move command is issued. The move mode will remain in absolute or incremental mode until it is changed.

In absolute mode, the axis will move by the specified target position. In incremental mode, the axis will increase or decrease its current position by the specified target position.

The motor status described in section 6.15 can be used to determine which move mode is active.

6.8.1. Individual Position Moves

For individual axis control use **X**, **Y**, **Z** and **U** commands followed by the target position value. For example, the **X1000** command will move the X-axis to position 1000 if performed in absolute mode.

6.8.2. Interpolated Position Moves

The PMX-4EX-SA can perform interpolated motion for up to 4 axis. A single move command will use the **X**, **Y**, **Z**, and **U** commands and can consist of up to 4 target positions (one for each axis). At least two axis must be referenced in order to perform linear interpolation.

For example, the **X1000Y1000Z100U800** will move all for axis to position (1000,1000,100,800). Additionally the **X1000Y-1000Z500** will move the X, Y, and Z axis to position (1000,-1000,500) while the U axis remains idle.

ASCII	X[target]	Y[target]	Z[target]	U[target]
Standalone	X[target]	Y[target]	Z[target]	U[target]

6.9. On-The-Fly Target Position Change

On-the-fly target position change can be achieved using the **T[axis][value]** command. While the motor is moving, **T[axis][value]** will change the final destination of the motor. If the motor has already passed the new target position, it will reverse direction once the target position change command is issued.

An on-the-fly target position change command will only be valid if the specified axis is performing in individual position move.

6.10. Circular Interpolation Moves

PMX-4EX-SA supports circular interpolation moves using the **CIRP** and **CIRN** commands. This will perform a complete 360° circle using the X and Y axis. The Z and U axis are not supported for circular interpolation.

CIRP[X]:[Y] – Draw circle in CW direction where [X][Y] signifies X,Y position of the circle center.

CIRN[X]:[Y] – Draw circle in CCW direction where [X][Y] signifies X,Y position of the circle center.

A circular interpolated moves will use the current XY position as well as the specified circle center to automatically calculate the circle radius. The maximum allow radius is 134,216,773 pulses for circular interpolated moves. All circular moves are interpreted as absolute moves.

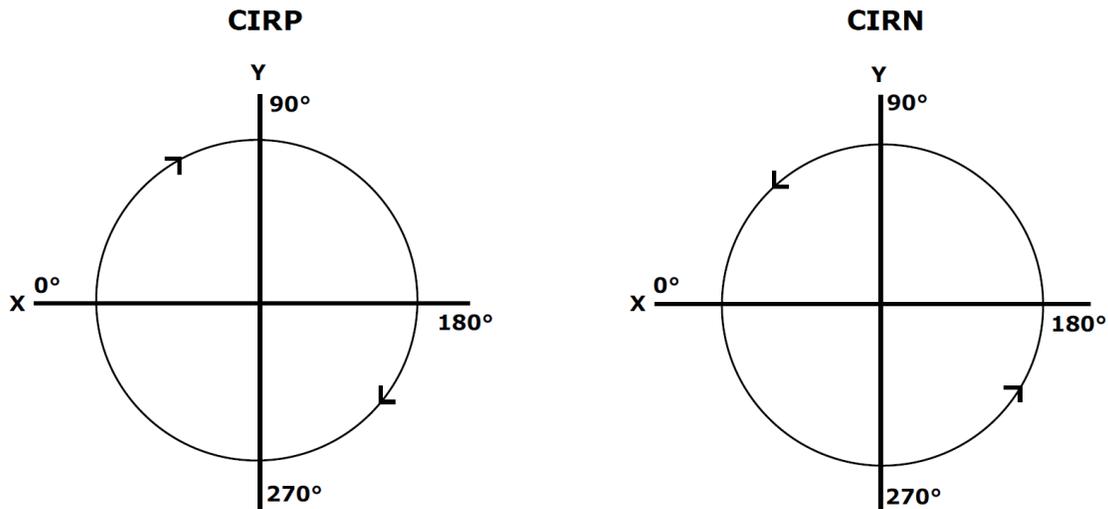


Figure 6-3

ASCII	CIRP [X]:[Y]	CIRN [X]:[Y]
Standalone	CIRP [X]:[Y]	CIRN [X]:[Y]

6.11. Arc Interpolation Moves

PMX-4EX-SA supports arc interpolation moves using the **ARCP** and **ARCN** commands. Arcs are drawn using X,Y axes only. The Z and U axis are not support for arc interpolation.

ARCP[X]:[Y]:[θ] – Draw arc in CW direction where [X][Y] signifies X,Y position of the arc center and θ signifies the absolute arc angle

ARCN[X]:[Y]:[θ] – Draw arc in CCW direction where [X][Y] signifies X,Y position of the arc center and θ signifies the absolute arc angle.

An arc interpolated move will use the current XY position as well specified arc center to automatically calculate the radius. The maximum allow radius is 134,216,773 pulses on arc interpolated moves. All arc moves are interpreted as absolute moves.

The absolute angle (θ) is in units of a milli-degree. For example, to move to the absolute angle of 45° , the absolute angle parameter would be 45000. The absolute angle locations can be found in Figure 6-4.

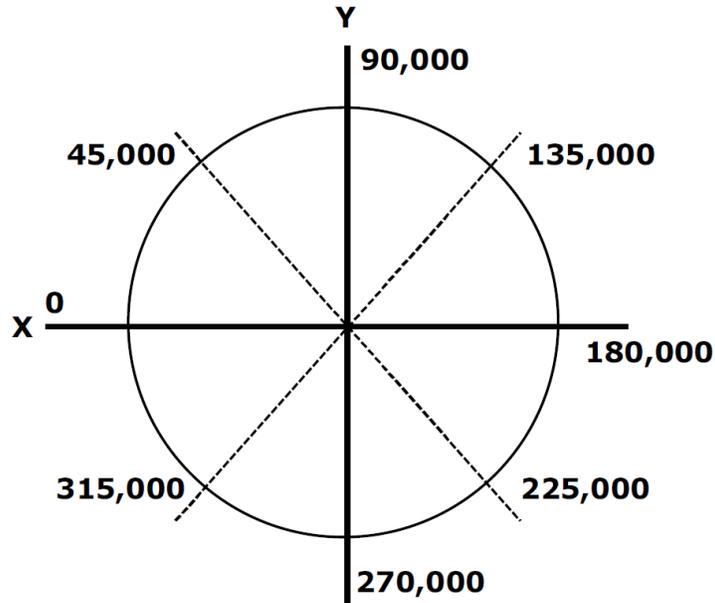


Figure 6-4

Figure 6-5 shows the difference between issuing the **ARCP0:0:90000** and the **ARCNO:0:90000** command if both cases have the same starting point at the 0° .

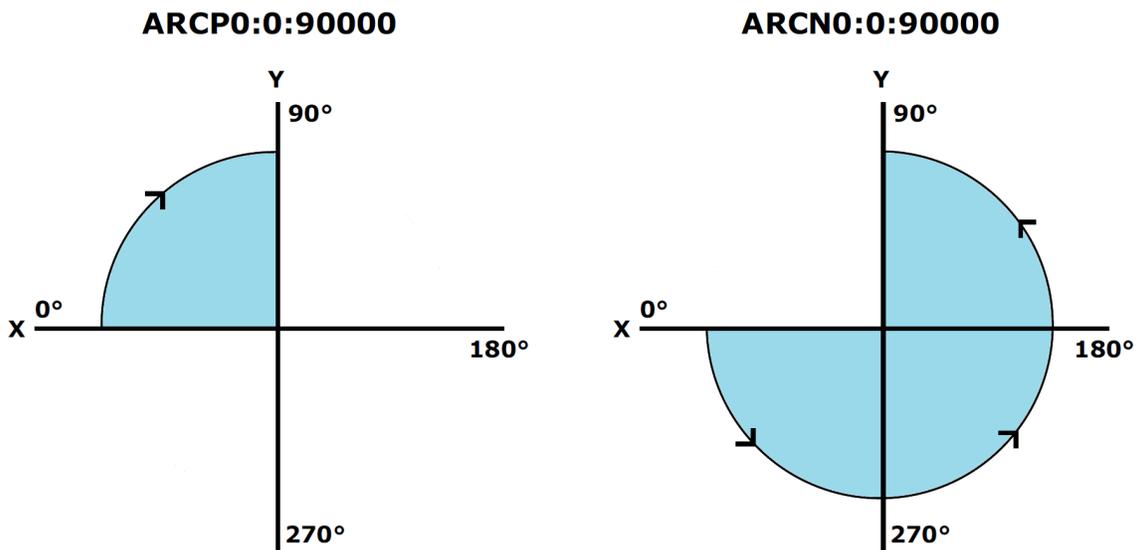


Figure 6-5

ASCII	ARCP[X]:[Y]:[θ]	ARC�[X]:[Y]:[θ]
Standalone	ARCP[X]:[Y]:[θ]	ARC�[X]:[Y]:[θ]

6.12. Buffered Interpolation Moves

PMX-4EX-SA supports buffered linear coordinated motion for X, Y, and Z-axes using the **I** command. This allows for the contouring of unique patterns without any delay between multiple moves. Each move has its own constant speed setting.

The buffer for the linear interpolation buffer mode contains 36 separate position points. To turn on and off buffer move, use the **BO** and **BF** ASCII command respectively. For standalone mode, the **BUFON** and **BUFOFF** commands should be used. When enabled, motion will start as soon as the first buffered move command is issued. The buffered move operation cannot be used while StepNLoop is enabled.

The motor status of the PMX-4EX-SA contains the status of the buffer operation. See section 6.15 for more information on this.

In ASCII mode, a buffered move command will have the syntax **I[X pos]:[Y pos]:[Z pos]:[speed]**. For example, to enter the position (1000,2000,3000) at a speed of 5000 into the buffer, the **I1000:2000:3000:5000** command should be used. If buffered mode is not enabled, **I** commands will not be processed by the controller.

In standalone mode, the buffered move command will have the syntax **X[pos]Y[pos]Z[pos]** and the speed for the move command will use the current global high speed setting. If buffered mode is not enabled, these standalone commands will be processed as regular linear interpolation commands.

Circular and arc interpolation moves can also be added to the buffer. The **ARC** and **CIR** commands, described in section 6.10 and 6.11, can be used when buffer mode is enabled. The syntax for these commands will stay the same in ASCII and standalone mode.

The buffered interpolation feature includes an automatic acceleration option. The **IACC** command is used to control this feature. When **IACC=0**, acceleration and deceleration must be done manually. To control the acceleration or deceleration manually, gradually increase or decrease the speed value for each interpolated move.

When **IACC=1**, acceleration and deceleration are processed automatically. In this case, the speed acceleration profile will be automatically generated between sequential buffered moves. The acceleration/deceleration value used for automatic acceleration control is found in the global acceleration value (**ACC**).

ASCII	BO	BF	I[X]:[Y]:[Z]:[S]	IACC
Standalone	BUFON	BUFOFF	X[pos]Y[pos]Z[pos]	-

6.13. Homing

Home search routines involve moving the motor and using the home, limit, or Z-index inputs to determine the zero reference position.

The homing routines that involve a decelerated stop will result in a final position that is non-zero. The zero reference position will be preserved as the position is marked when the home trigger is detected. If a motion command is sent while the controller is already moving, the command is not processed. Instead, an error response is returned. See Table 8-2 for details on error responses.

The PMX-4EX-SA has five different homing routines. In ASCII mode, use the **H** command to perform a home move. The syntax for the home command in ASCII mode can be found below.

H[axis][direction + or -][homing mode 0,1,2,3,4]

MODE 0 – Home Input Only (High speed only)

MODE 1 – Limit Input Only

MODE 2 – Home Input and Z-Index

MODE 3 – Z-Index Only

MODE 4 – Home Input Only (High speed and low speed)

6.13.1. MODE 0: Home Input Only (High Speed Only)

Use the **H[axis][+/-]0** command in ASCII mode or the **HOME[axis][+/-]** command in standalone mode to issue a homing command that uses the home input only. Figure 6-6 shows the homing routine.

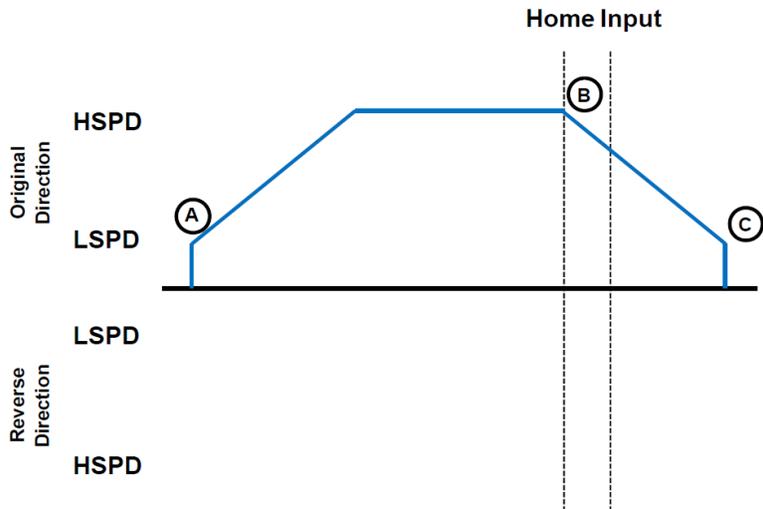


Figure 6-6

- A. Issuing the command starts the motor from low speed and accelerates to high speed in search of the home input.
- B. As soon as the home input is triggered, the position counter is reset to zero and the motor begins to decelerate to low speed. As the motor decelerates, the position counter keeps counting with reference to the zero position.
- C. Once low speed is reached the motor stops. Although the position is non-zero, the zero position is maintained.

ASCII	H[axis][+/-]0
Standalone	HOME[axis][+/-]

6.13.2. MODE 1: Limit Only

Use the **H[axis][+/-]1** command for ASCII mode or the **LHOME[axis] +/-** command for standalone mode. Figure 6-7 shows the homing routine.

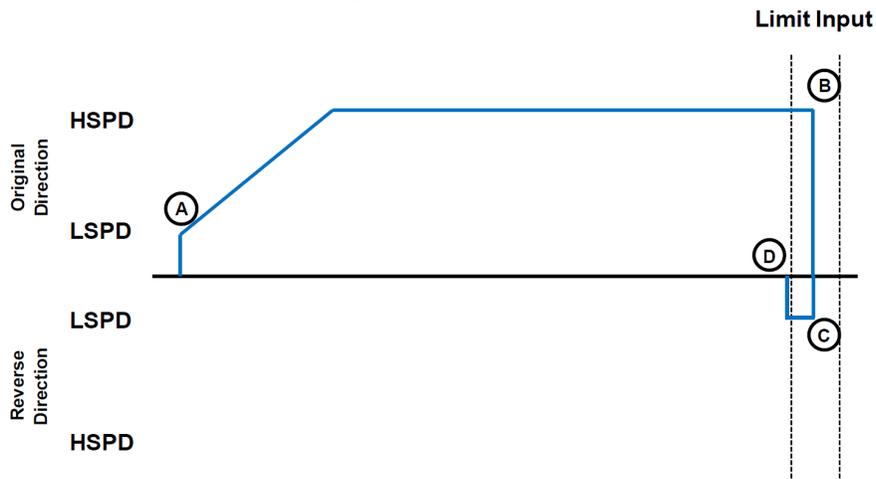


Figure 6-7

- A. Issuing the command starts the motor from low speed and accelerates to high speed in search of the specified limit input.
- B. As soon as the relevant limit input is triggered, the motor immediately stops motion. Depending on the high speed, it is possible that an immediate stop results in a small overshoot of the limit input.
- C. The motor will start moving in the opposite direction at the low speed setting until the limit input is cleared.
- D. Once the specified limit input is cleared, the motor will immediately stop and mark the zero reference position.

ASCII	H[axis][+/-]1
Standalone	LHOME[axis][+/-]

6.13.3. MODE 2: Home Input and Z-index

Use the **H[axis][+/-]2** command for ASCII mode or the **ZHOME[axis][+/-]** command for standalone mode. In order to use this homing routine, an encoder must be used for the specified axis. Figure 6-8 shows the homing routine.

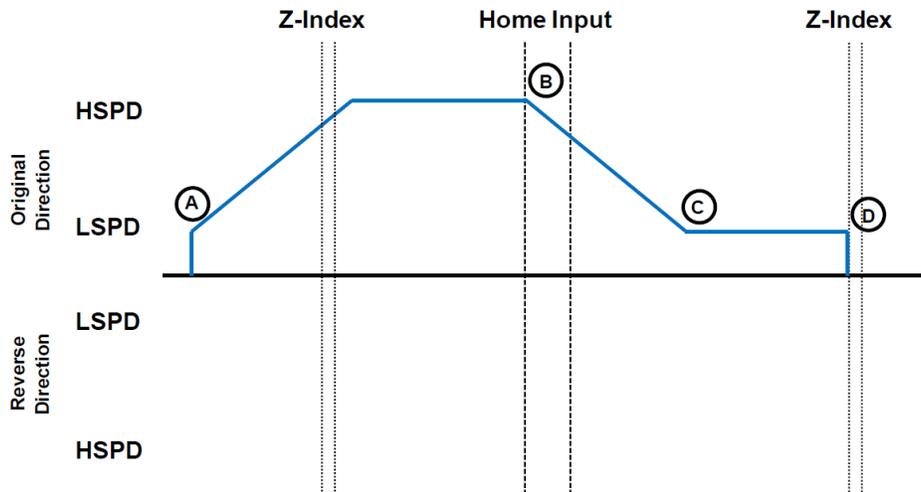


Figure 6-8

- A. Issuing the command starts the motor from low speed and accelerates to high speed in search of the home input.
- B. As soon as the home input is triggered, the motor decelerates to low speed
- C. After the home input is triggered, the motor begins to search for the z-index pulse.
- D. Once the z-index pulse is found, the motor immediately stops and the position is set to zero.

ASCII	H[axis][+/-]2
Standalone	ZHOME[axis][+/-]

6.13.4. MODE 3: Z-index Only

Use the **H[axis][+/-]3** command for ASCII mode or the **ZOME[axis]+/-** command for standalone mode. In order to use this homing routine, an encoder must be used on the specified axis. Figure 6-9 shows the homing routine.

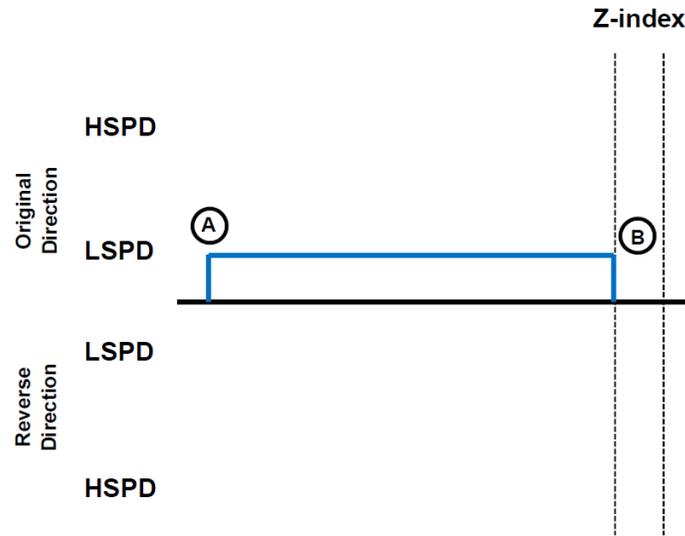


Figure 6-9

- A. Issuing the command starts the motor at low speed. The motor will not use the high speed setting when performing this homing routine.
- B. Once the z-index pulse is found, the motor stops and the position is set to zero.

ASCII	H[axis][+/-]3
Standalone	ZOME[axis][+/-]

6.13.5. MODE 4: Home Input Only (High Speed and Low Speed)

Use the **H[axis][+/-]4** command for ASCII mode and the **HLHOME[axis]+/-** for standalone mode. Figure 6-10 shows the homing routine.

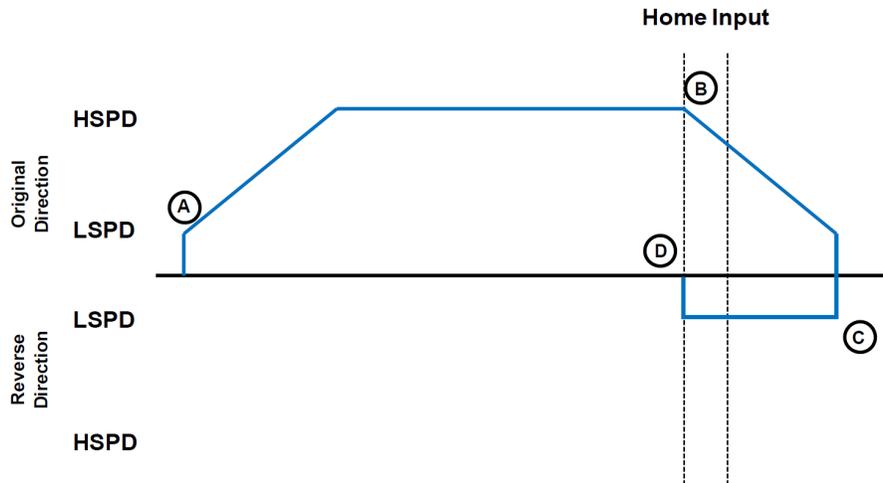


Figure 6-10

- A. Starts the motor from low speed and accelerates to high speed.
- B. As soon as the home input is triggered, the motor decelerates to low speed.
- C. Once low speed is reached, the motor reverses direction until the home input is cleared.
- D. Once the home switch is cleared, the motor will immediately stop and reset the position and encoder counter to zero.

ASCII	H[axis][+/-]4
Standalone	HLHOME[axis][+/-]

6.14. Limits and Alarm Switch Function

Triggering the limit switch while the axis is moving will stop the motion immediately. For example, if the positive limit switch is triggered while moving in positive direction, the motor will immediately stop and the motor status bit for positive limit error is set. The same will apply for the negative limit while moving in the negative direction. Each axis will have its own designated positive and negative limit inputs.

If the alarm input for an axis is triggered during movement in either direction, the motor will immediately stop and the motor status bit for alarm error is set. Each axis will have its own designated alarm input.

Once the limit or alarm error is set, use the **CLR[axis]** command to clear the error in ASCII mode or the **ECLEAR[axis]** command in standalone mode.

The limit and alarm error states can be ignored by setting **IERR=1**. In this case, the motor will still stop when the appropriate switch is triggered, however, it will not enter an error state.

ASCII	CLR[axis]
Standalone	ECLEAR[axis]

6.15. Motor Status

Motor status can be read anytime using **MST** command. Value of the motor status is replied as an integer with following bit assignment:

Bit	Description
0	Motor in acceleration
1	Motor in deceleration
2	Motor running at constant speed
3	Alarm input status
4	Plus limit input switch status
5	Minus limit input switch status
6	Home input switch status
7	Plus limit error. This bit is latched when plus limit is hit during motion. This error must be cleared using the CLR command before issuing any subsequent move commands.
8	Minus limit error. This bit is latched when minus limit is hit during motion. This error must be cleared using the CLR command before issuing any subsequent move commands.
9	Alarm error. This bit is latched when alarm is triggered during motion. This error must be cleared using the CLR command before issuing any subsequent move commands.
10	Reserved
11	TOC time-out status

Table 6-3

This command returns the motor status for all axes, as well as other information. The **MST** return value has the following format:

[Motor Status X]:
[Motor Status Y]:
[Motor Status Z]:
[Motor Status U]:
[Buffer enabled]:
[Buffer start]:
[Buffer end]:
[Available Buffer]:
[Move mode]

- Motor Status [X/Y/Z/U] – Provide motor status of the axis define by Table 6-3.
- Buffer enabled – The enable status of the buffered interpolation feature. (0: off, 1: on)
- Buffer start – The index of the current command in the 36 position buffer
- Buffer end – The index of the last command in the 36 position buffer
- Available Buffer – The amount of empty spaces in the 36 position buffer
- Move mode – The current move mode for positional moves (0: ABS, 1: INC)

6.16. Digital Inputs/Outputs

PMX-4EX-SA module comes with 8 digital inputs and 8 digital outputs.

6.16.1. Digital Inputs

The digital input status of all 8 available inputs can be read with the **DI** command. Digital input values can also be referenced one bit at a time by using the **DI[1-8]** commands. Note that the indexes are 1-based for the bit references. For example, DI1 refers to bit 0, not bit 1. See Table 6-4 for details.

Bit	Description	Bit-Wise Command
0	Digital Input 1	DI1
1	Digital Input 2	DI2
2	Digital Input 3	DI3
3	Digital Input 4	DI4
4	Digital Input 5	DI5
5	Digital Input 6	DI6
6	Digital Input 7	DI7
7	Digital Input 8	DI8

Table 6-4

If a digital input is on, the corresponding bit of the **DI** command is 1. Otherwise, the bit status is 0. The voltage level required to activate a digital input is determined by the polarity setting. See section 6.21 for details.

ASCII	DI	DI[1-8]
Standalone	DI	DI[1-8]

6.16.2. Digital Outputs

The **DO** command can be used to set the output voltage of all available digital outputs. The **DO** value must be within the range of 0-255.

Digital output values can also be referenced one bit at a time with the **DO[1-8]** commands. Note that the indexes are 1-based for the bit references. For example, DO1 refers to bit 0, not bit 1. See Table 6-5 for details.

Bit	Description	Bit-Wise Command
0	Digital Output 1	DO1
1	Digital Output 2	DO2
2	Digital Output 3	DO3
3	Digital Output 4	DO4
4	Digital Output 5	DO5
5	Digital Output 6	DO6
6	Digital Output 7	DO7
7	Digital Output 8	DO8

Table 6-5

If a digital output is turned on, the corresponding bit of the **DO** command is 1. Otherwise, the bit status is 0. The voltage level of the digital output when it is on or off is determined by the polarity setting. See section 6.21 for details.

The initial state of the digital outputs can be defined by setting the **DOBOOT** register to the desired initial digital output value. The **DOBOOT** value must be within the range of 0-255. The value is stored to flash memory once the **STORE** command is issued.

ASCII	DO	DO[1-8]	DOBOOT
Standalone	DO	DO[1-8]	-

6.17. High Speed Latch Inputs

The PMX-4EX-SA module provides the following high speed position latch inputs. This feature will allow the controller to perform a high speed position capture of both pulse and encoder counters based on a digital input trigger.

Each axis has a designated digital input to perform a high speed position latch. See corresponding latch input for each axis below in Table 6-6.

Axis	Latch Input
X	DI4
Y	DI6
Z	DI5
U	DI8

Table 6-6

Use the **LT[axis]** command to enable and disable the latch feature. To read the latch status, use the ASCII command **LT[axis]S** or the standalone command

LTS[axis]. Table 6-7 describes the possible return values of the latch status command.

Return Value	Description
0	Latch off
1	Latch on and waiting for latch trigger
2	Latch Triggered

Table 6-7

Once the latch is triggered, the triggered pulse position can be retrieved using **LT[axis]P** command in ASCII mode or the **LTP[axis]** command in standalone mode. Similarly, the triggered encoder position can be retrieved using the **LT[axis]E** command in ASCII mode or the **LTE[axis]** command in standalone mode.

When StepNLoop mode is enabled, the position value should be ignored.

ASCII	LT[axis]	LT[axis]S	LT[axis]P	LT[axis]E
Standalone	LT[axis]	LTS[axis]	LTP[axis]	LTE[axis]

6.18. Sync Outputs

PMX-4EX-SA has a designated synchronization digital output for each axis. The synchronization digital output is triggered when the encoder position value meets the set condition. See the synchronization digital output for each axis below:

Axis	Synchronization Output
X	DO1
Y	DO2
Z	DO3
U	DO4

Table 6-8

While the synchronization feature is enabled for an axis, the corresponding digital output cannot be manually controlled by user.

Use **SYN[axis]O** in ASCII mode or the **SYNON[axis]** in standalone mode to enable the synchronization output feature for an axis.

Use **SYN[axis]F** in ASCII mode or the **SYNOFF[axis]** in standalone mode to disable the synchronization output feature for an axis.

To set the synchronization feature, use the ASCII command **SYN[axis]P** or the standalone command **SYNPOS[axis]** to read and set the synchronization position value for an axis. This will designate the encoder position that will be used to trigger the synchronization output.

The synchronization condition, defined by the ASCII command **SYN[axis]C** or the standalone command **SYNCFG[axis]**, is used to determine how the previously defined synchronization position is used. Table 6-9 shows the condition values that are available.

Configuration Value	Description
1	Trigger when the encoder position is EQUAL to sync position.
2	Trigger when the encoder position is LESS than the sync position
3	Trigger when the encoder position is GREATER than sync position

Table 6-9

The pulse width of the synchronization output can be set using the ASCII command **SYN[axis]T** or the standalone command **SYNTIME[axis]**. The pulse width time is defined by milliseconds with a maximum length of 10ms. This parameter is only used if the synchronization condition is set to 1. If this parameter is set to 0, the output pulse will depend on how long the encoder value is equal to the sync position.

The status of the synchronization output feature can be read using the ASCII command **SYN[axis]S** or the standalone command **SYNSTAT[axis]**. Table 6-10 describes the possible return values of the synchronization status.

Return Value	Description
0	Sync output is off
1	Waiting for the sync condition to occur
2	Sync condition has occurred

Table 6-10

ASCII	SYN[axis]O	SYN[axis]F	SYN[axis]P	SYN[axis]C
Standalone	SYNON[axis]	SYNOFF[axis]	SYNPOS[axis]	SYNCFG[axis]

ASCII	SYN[axis]T	SYN[axis]S
Standalone	SYNTIME[axis]	SYNSTAT[axis]

6.19. Analog Inputs

The PMX-4EX-SA has 8 x 10-bit analog inputs available. The **AI[1-8]** command can be used to read the current analog input value. The return value is in millivolts and can range from 0 to 5000 mV.

Voltage supplied to the analog inputs should stay within the 0V to 5V range.

6.20. Joystick Control

Joystick control is available on PMX-4EX-SA. When this mode is enabled, the pulse speed and direction output can be controlled by a corresponding analog input. See the axis to analog input relationship in the table below:

Axis	Analog Input
X	AI1
Y	AI2
Z	AI3
U	AI4

Table 6-11

To enable or disable joystick control for an axis, use the ASCII command **JE** or the standalone command **JOYENA**. The joystick enable parameter is a 4 bit value. For example, digital output value of 15 (1111 in binary or 0xF in hex) means joystick feature is enabled on all axes. If joystick control is enabled, StepNLoop is automatically disabled.

The **JMAX[axis]** command can be used to define the maximum analog input value for the specified axis. This parameter has units of millivolts and a range of 0 to 5000. This value will define the analog input value in which the maximum joystick speed occurs in the positive direction. This command is only available in ASCII mode.

Similarly, the **JMIN[axis]** command can be used to define the minimum analog input value for the specified axis. This parameter has units of milli-volts and a range of 0 to 5000. This value will define the analog input value in which the maximum joystick speed occurs in the negative direction. This command is only available in ASCII mode.

During joystick operation, analog input of **JMIN[axis]** mV to *MID* mV represents negative joystick direction and analog input of *MID* mV to **JMAX[axis]** mV represents positive joystick direction.

MID mV represents the zero joystick position. There is no command corresponding to *MID*. This is an internally calculated value using the following formula:

$$\mathbf{MID = ((JMAX[axis] - JMIN[axis]) / 2) + JMIN[axis]}$$

The ASCII commands **JV9**, **JV10**, **JV11** and **JV12** can be used to define the zero tolerance zone around the *MID* value. No movement will occur while the analog input value is within the zero tolerance zone. This parameter has units of millivolts and a range of 0 to 5000. The zero tolerance parameter will be on the positive and negative side of the *MID* point to define to zero tolerance zone. See Figure 6-11 for details.

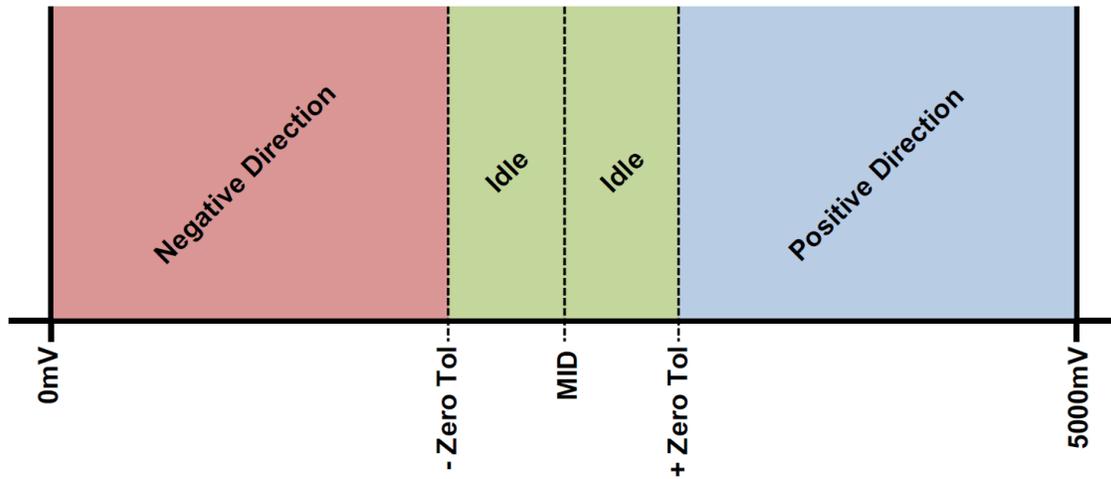


Figure 6-11

The maximum joystick speed for the X, Y, Z, and U axis is set using the ASCII commands **JV1**, **JV2**, **JV3** and **JV4**, respectively. For standalone mode, the **JOYHS[axis]** command should be used. This parameter will define the speed of the axis at the maximum and minimum analog input values.

The maximum allowable speed change (delta) for the X, Y, Z, and U axis is set using the ASCII commands **JV5**, **JV6**, **JV7** and **JV8**, respectively. For standalone mode, the **JOYDEL[axis]** command should be used in standalone mode. This parameter will control the acceleration/deceleration of the axis while joystick mode is enabled.

Joystick control also has soft limit controls. The soft limits are defined by a negative outer limit, negative inner limit, positive inner limit and positive outer limit. The soft limits are in units of pulses.

When moving in positive direction, as soon as the positive inner limit is crossed, the speed is reduced. If the position reaches the positive outer limit, the joystick speed is set to zero and movement in the positive direction is prohibited. The same behavior is applicable to the negative direction and negative limits.

Figure 6-12 represents the relationship between the joystick speed and inner and outer limits.

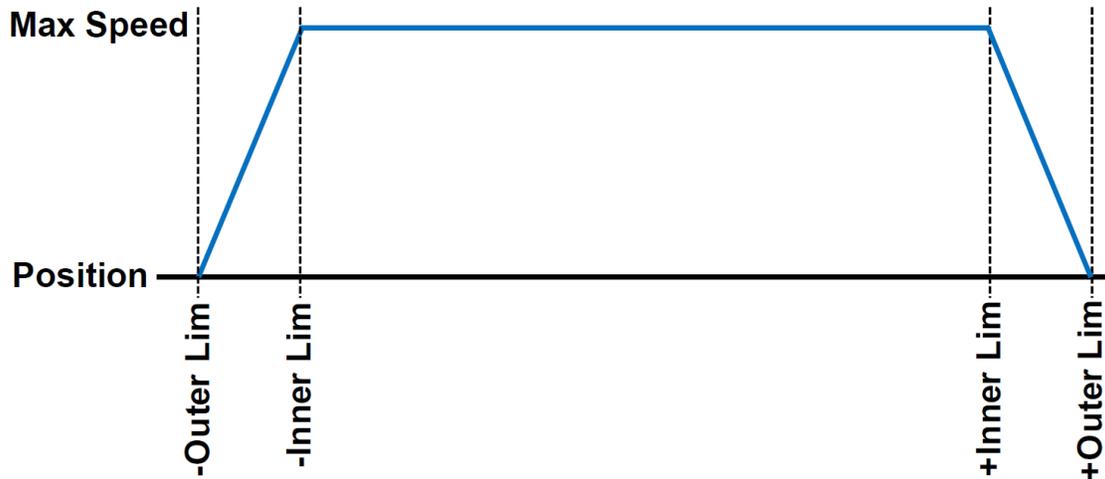


Figure 6-12

See Table 6-12 for the command assignment of the negative and positive soft limits for joystick control as well as a summary of the additional joystick commands.

Command	Description
JV1	X-axis maximum joystick speed at 5000 mV and 0 mV
JV2	Y-axis maximum joystick speed at 5000 mV and 0 mV
JV3	Z-axis maximum joystick speed at 5000 mV and 0 mV
JV4	U-axis maximum joystick speed at 5000 mV and 0 mV
JV5	X-axis maximum speed change
JV6	Y-axis maximum speed change
JV7	Z-axis maximum speed change
JV8	U-axis maximum speed change
JV9	X-axis zero tolerance range for analog input
JV10	Y-axis zero tolerance range for analog input
JV11	Z-axis zero tolerance range for analog input
JV12	U-axis zero tolerance range for analog input
JL1	X-axis negative outer limit
JL2	X-axis negative inner limit
JL3	X-axis positive inner limit
JL4	X-axis positive outer limit
JL5	Y-axis negative outer limit
JL6	Y-axis negative inner limit
JL7	Y-axis positive inner limit
JL8	Y-axis positive outer limit
JL9	Z-axis negative outer limit
JL10	Z-axis negative inner limit
JL11	Z-axis positive inner limit
JL12	Z-axis positive outer limit
JL13	U-axis negative outer limit
JL14	U-axis negative inner limit
JL15	U-axis positive inner limit

JL16	U-axis positive outer limit
JMAXX	X-axis maximum mV
JMAXY	Y-axis maximum mV
JMAXZ	Z-axis maximum mV
JMAXU	U-axis maximum mV
JMINX	X-axis minimum mV
JMINY	Y-axis minimum mV
JMINZ	Z-axis minimum mV
JMINU	U-axis minimum mV

Table 6-12

ASCII	JE	JMAX[axis]	JMIN[axis]	JV[1-4]	JC[5-8]	JL[1-16]
Standalone	JOYENA	-	-	JOYHS[axis]	JOYDEL[axis]	-

6.21. Polarity

Using the **PO[axis]** command, the polarity of the following signals can be configured.

Bit	Description
0	Home
1	Alarm
2	Limit (X-axis limit input setting controls limit switch polarity for all axes)
3	Pulse Direction
4	Encoder Direction

Table 6-13

The polarity can also be set for the digital inputs and outputs. The **DIP** command can be used to toggle the polarity of the digital inputs and the **DOP** can be used to toggle the polarity of the digital outputs.

Similarly the **EOP** command can be used to set the polarity of the enable outputs.

ASCII	PO[axis]	DIP	DOP	EOP
Standalone	-	-	-	-

6.22. StepNLoop Closed Loop Control

PMX-4EX-SA features a closed-loop position verification algorithm called StepNLoop (SNL). The algorithm requires the use of an incremental encoder.

SNL performs the following operations:

- 1) Position Verification: At the end of any targeted move, SNL will perform a correction if the current error is greater than the tolerance value.
- 2) Delta Monitoring: The delta value is the difference between the actual and the target position. When delta exceeds the error range value, the motor is stopped and the SNL Status goes into an error state. Delta monitoring is performed during moves – including homing and jogging. To read the delta value, use the **DX[axis]** command.

See Table 6-14 for a list of the SNL control parameters.

SNL Parameter	Description	Command
StepNLoop Ratio	Ratio between motor pulses and encoder counts. This ratio will depend on the motor type, microstepping, encoder resolution and decoding multiplier. Value must be in the range [0.001 , 999.999].	SLR[axis]
Tolerance	Maximum error between target and actual position that is considered “In Position”. In this case, no correction is performed. Units are in encoder counts.	SLT[axis]
Error Range	Maximum error between target and actual position that is not considered a serious error. If the error exceeds this value, the motor will stop immediately and go into an error state.	SLE[axis]
Correction Attempt	Maximum number of correction tries that the controller will attempt before stopping and going into an error state.	SLA[axis]

Table 6-14

A convenient way to find the StepNLoop ratio is to set EX=0, PX=0 and move the motor +1000 pulses. The ratio can be calculated by dividing 1000 by the resulting EX value. Note that the value must be positive. If it is not, then the direction polarity must be adjusted. This test should be performed while StepNLoop is disabled.

To enable/disable the SNL feature use the **SL[axis]** command. To read the SNL status, use **SLS[axis]** command. See Table 6-15 for a list of the **SLS[axis]** return values.

Return Value	Description
0	Idle
1	Moving
2	Correcting
3	Stopping
4	Aborting
5	Jogging

6	Homing
7	Z-Homing
8	Correction range error. To clear this error, use CLRS or CLR command.
9	Correction attempt error. To clear this error, use CLRS or CLR command.
10	Stall Error. DX value has exceeded the correction range value. To clear this error, use CLRS or CLR command.
11	Limit Error
12	N/A (i.e. SNL is not enabled)
13	Limit homing

Table 6-15

See Table 6-16 for SNL behavior within different scenarios.

Condition	SNL behavior (motor is moving)	SNL behavior (motor is idle)
$\delta \leq \text{SLT}$	Continue to monitor the DX[axis]	In Position. No correction is performed.
$\delta > \text{SLT}$ AND $\delta < \text{SLE}$	Continue to monitor the DX[axis]	Out of Position. A correction is performed.
$\delta > \text{SLT}$ AND $\delta > \text{SLE}$	Stall Error. Motor stops and signals an error.	Error Range Error. Motor stops and signals an error.
Correction Attempt > SLA	NA	Max Attempt Error. Motor stops and signals an error.

Table 6-16

Key

- [δ]: Error between the target position and actual position
- SLT: Tolerance range
- SLE: Error range
- SLA: Max correction attempt

Once SNL is enabled, position move commands are in term of encoder position. For example, X1000 means to move the motor to encoder position 1000. This applies to individual as well as interpolated moves.

Additionally, once SNL is enabled, the speed is in encoder speed. For example HSPD=1000 when SNL is enabled means that the target high speed is 1000 encoder counts per second. This only applies to individual axis moves.

For linear interpolated move the speed during a linear interpolation move is calculated as pulse/sec, NOT encoder counts/sec. The same will apply to the X and Y axis while performing an arc/circular interpolated move.

ASCII	DX[axis]	SL[axis]	SLS[axis]	SLR[axis]	STL[axis]	SLE[axis]	SLZ[axis]
Standalone	-	SL[axis]	SLS[axis]	-	-	-	-

6.23 Timer Register

PMX-4EX-SA comes with a timer register. Once the timer register is set, it begins to count down to 0. Read and write to the timer register using the **TR** command. The units are in milliseconds.

ASCII	TR
Standalone	TR

6.24. Communication Time-out Watchdog

PMX-4EX-SA allows for the user to trigger an alarm if a master has not communicated with the device for a set period of time. When an alarm is triggered, bit 11 of the **MSTX** parameter is turned on. The time-out value is set by the **TOC** command. Units are in milliseconds. This feature is typically used in stand-alone mode. Refer to section 9 for an example.

In order to disable this feature set **TOC=0**.

ASCII	TOC
Standalone	TOC

6.25. Standalone Program Specification

Standalone programming allows the controller to execute a user defined program that is stored in the internal memory of the PMX-4EX-SA. The standalone program can be run independently of USB and serial communication or while communication is active.

Standalone programs can be written to the PMX-4EX-SA using the Windows GUI described in section 7. Once a standalone program is written by the user, it is then compiled and downloaded to the PMX-4EX-SA. Each line of written standalone code creates 1-4 assembly lines of code after compilation

The PMX-4EXSA can store and operate up to four separate standalone programs simultaneously.

6.25.1. Standalone Program Specification

Memory size: 1,275 assembly lines.

Note: Each line of pre-compiled code equates to 1-4 lines of assembly lines.

6.25.2. Standalone Control

The PMX-4EX-SA supports the simultaneous execution of four standalone programs. All programs can be controlled by the **SR[0-3]** command, where Program 0 uses command **SR0**, Program 1 uses command **SR1**, and so on. For

examples of multi-threading, please refer to section 9. The following assignments can be used for the **SR[0-3]** command.

Value	Description
0	Stop standalone program
1	Start standalone program
2	Pause standalone program
3	Continue standalone program

Table 6-17

6.25.3. Standalone Status

The **SASTAT[0-3]** command can be used to determine the current status of the specified standalone program. Table 6-18 details the return values of this command.

Value	Description
0	Idle
1	Running
2	Paused
3	N/A
4	Errored

Table 6-18

The **SPC[0-3]** command can also be used to find the current assembled line that the specified standalone program is executing. Note that the return value of the **SPC[0-3]** command is referencing the assembly language line of code and does not directly transfer to the pre-compiled user generated code. The return value can range from [0-1274].

6.25.4. Standalone Subroutines

The PMX-4EX-SA has the capabilities of using up to 32 separate subroutines. Subroutines are typically used to perform functions that are repeated throughout the operation of the standalone program. Note that subroutines can be shared by both standalone programs. Refer to section 9 for further details on how to define subroutines.

Once a subroutine is written into the flash, they can be called via USB communication using the **GS** command. Standalone programs can also jump to subroutine using the **GOSUB** command. The subroutines are referenced by their subroutine number [SUB 0 - SUB 31]. If a subroutine number is not defined, the controller will return with an error.

6.25.5. Error Handling

Subroutine 31 is designated for error handling. If an error occurs during standalone execution (i.e. limit error, StepNLoop error), the standalone program

will automatically jump to SUB 31. If SUB 31 is not defined, the program will cease execution and go into error state.

If SUB 31 is defined by the user, the code within SUB 31 will be executed. Typically the code within subroutine 31 will contain the standalone command **ECLEARX** in order to clear the current error. Section 9 contains examples of using subroutine 31 to perform error handling.

The return jump from subroutine 31 will be determined by the ASCII command **SAP**. Write a "0" to this setting to have the standalone program jump back to the last performed line. Write a "1" to this setting to have the standalone program jump back to the first line of the program.

6.25.6. Standalone Variables

The PMX-4EX-SA has 100 32-bit signed standalone variables available for general purpose use. They can be used to perform basic calculations and support integer operations. The **V[0-99]** command can be used to access the specified variables. The syntax for all available operations can be found below. Note that these operations can only be performed in standalone programming.

Operator	Description	Example
+	Integer Addition	V1=V2+V3
-	Integer Subtraction	V1=V2-V3
*	Integer Multiplication	V1=V2*V3
/	Integer Division (round down)	V1=V2/V3
%	Modulus	V1=V2%5
>>	Bit Shift Right	V1=V2>>2
<<	Bit Shift Left	V1=V2<<2
&	Bitwise AND	V1=V2&7
	Bitwise OR	V1=V2 8
~	Bitwise NOT	V1=~V2

Table 6-19

Variables V50 through V99 can be stored to flash memory using the **STORE** command. Variables V0-V49 will be initialized to zero on power up.

6.25.7. Standalone Run on Boot-Up

Standalone can be configured to run on boot-up using the **SLOAD** command. Once this command has been issued, the **STORE** command will be needed to save the setting to flash memory. It will take effect on the following power cycle. See description in Table 6-20 for the bit assignment of the **SLOAD** setting.

Bit	Description
0	Standalone Program 0
1	Standalone Program 1
2	Standalone Program 2
3	Standalone Program 3

Table 6-20

Standalone programs can also be configured to run on boot-up using the Windows GUI. See section 7 for details.

ASCII	SR[0-3]	SASTAT[0-3]	SPC[0-3]	GS[0-31]	V[0-99]	SLOAD
Standalone	SR[0-3]	-	-	GOSUB[0-31]	V[0-99]	-

6.26. Storing to Flash

The following items can be stored to flash by issuing the **STORE** command.

ASCII Command	Description
DN	Device name
DB	Serial communication baud rate
DIP	Digital input polarity
DOBOOT	DO configuration at boot-up
DOP	Digital output polarity
EDEC	Unique deceleration enable
EOBOOT	EO configuration at boot-up
EOP	Enable output polarity
IACC	Automatic I-move acceleration/deceleration enable
IERR	Ignore limit error enable
JO, JF, JV[1-12], JL[1-16], JMAX[axis], JMIN[axis]	Joystick settings
POL[axis]	Polarity settings
SAP	Jump return line select (Stand-alone error handling)
SCV[axis]	S-curve enable
SL[axis], SLR[axis], SLE[axis], SLT[axis], SLA[axis]	StepNLoop parameters
SLOAD	Standalone program run on boot-up parameter
TOC	Time-out counter reset value
V50-V99	Note that on boot-up, V0-V49 are reset to value 0

Table 6-21

When a standalone program is downloaded, the program is immediately written to flash memory.

7. Software Overview

The PMX-4EX-SA has a Windows compatible software that allows for USB or RS485 communication. Standalone programming, along with all other available features of the PMX-4EX-SA, will be accessible through the software. It can be downloaded from the Nippon Pulse website.

To communicate over a USB connection, make sure that the PMX-4EX-SA is connected to one of the available ports on the PC. To communicate over RS485, make sure that the PMX-4EX-SA is connected to the COM port.

Startup the PMX-4EX-SA GUI program and you will see the following screen in Figure 7-1. This will allow the user to search for all the motors that are currently connected to the USB network or RS485 bus.

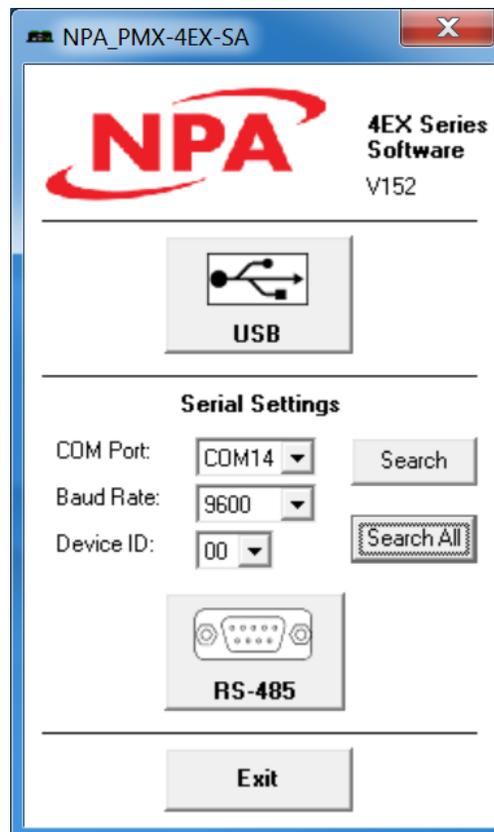


Figure 7-1

USB communication can be established by clicking the USB button. All PMX-4EX-SA connected to the PC will automatically be detected and displayed.

For RS-485 communication, the first PMX-4EX-SA connected to the PC can be found using the Search button. If there are multiple PMX-4EX-SA connected to the PC over the RS-485 bus, the Search All button can be used to find them.

If the search fails, or a connection cannot be opened, check the following items:

- Check power supply to PMX-4EX-SA. Allowable power is range is from 12VDC to 24VDC.
- Check communication wiring. The 485+ from PMX-4EX-SA is connected to 485+ of the master and 485- from PMX-4EX-SA is connected to 485- of the master.
- Confirm that the device name is set correctly. Default factory device name setting is "00". If this name has been changed and stored to flash, enter the new name.

Once the correct serial settings have been determined, the RS-485 button can be used to open the software.

After a successful connection has been established, the screen below allows the user to choose between two different program types. To use a particular program, click on the corresponding button.

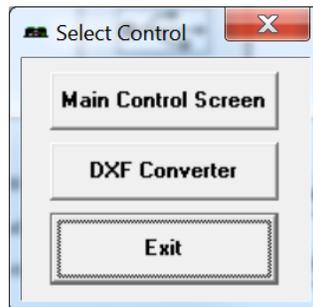


Figure 7-2

7.1. Main Control Screen

The Main Control Screen provides accessibility to all the available functions on the PMX-4EX-SA. All features can be tested and verified.

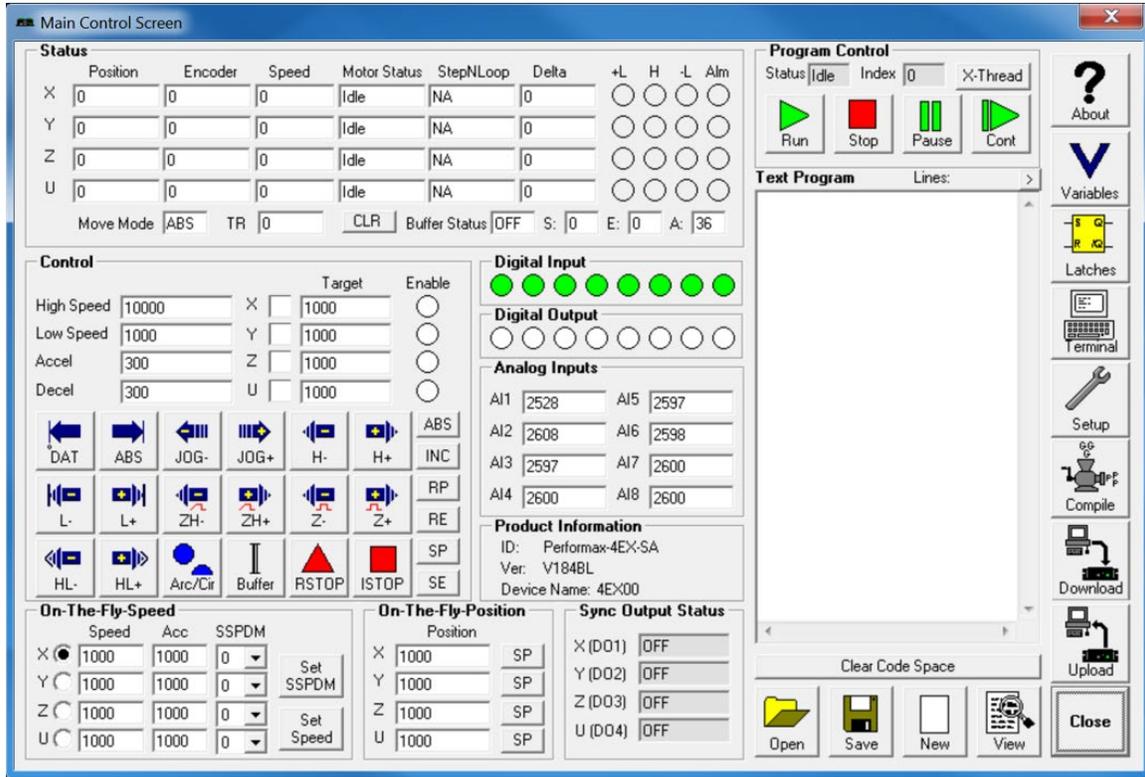


Figure 7-3

7.1.1. Status

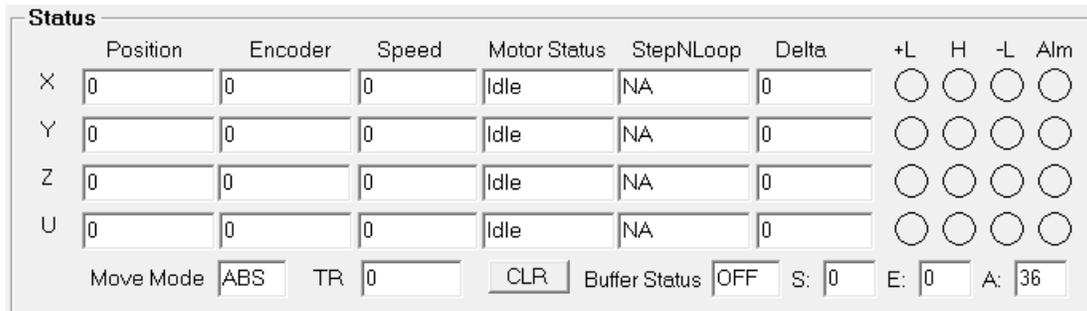


Figure 7-4

-
1. **Position** (X,Y,Z,U) - displays the current pulse position counter. If StepNLoop is enabled, this shows the real-time target position.
 2. **Encoder** (X,Y,Z,U) - displays the current encoder position counter.
 3. **Speed** (X,Y,Z,U) - displays the current pulse speed output rate. If StepNLoop is enabled, the speed is in encoder counts/sec, unless an interpolation move is in process.
 4. **Motor Status** (X,Y,Z,U axes) - the current motor status of the axis.
The following status can be shown.
 - Idle – motor is not moving.
 - Accel – motor is accelerating
 - Const – motor is running in constant speed
 - Decel – motor is decelerating
 - +LimError – plus limit error
 - -LimError – minus limit error
 5. **StepNLoop** (X,Y,Z,U) - valid only when StepNLoop is enabled and displays the current StepNLoop status
 - NA – StepNLoop is disabled
 - IDLE – motor is not moving
 - MOVING – target move is in progress
 - JOGGING – jog move is in progress
 - HOMING – homing is in progress
 - Z-HOMING – homing using Z-index channel in progress
 - ERR-STALL – StepNLoop has stalled.
 - ERR-LIM – plus/minus limit error
 6. **Delta** (X,Y,Z,U) - valid only for StepNLoop. Displays the difference between the target positions and the actual position.
 7. **Limit/Home/Alarm Input Status** (X,Y,Z,U axes) - Indicators for the limit, home, and alarm inputs.
 8. **Move Mode** - displays the current move mode for positional moves.
 - ABS – absolute move
 - INC – incremental move
 9. **TR** - displays the timer register status (counts down)
 10. **CLR** - Clears any limit, alarm, or StepNLoop error
 11. **Buffer Status** - displays whether buffer mode is ON or OFF
 12. **S** - valid only for buffer mode. Displays the current index of the buffer.
Note that the buffer is a 36 position ring buffer.
 13. **E** - valid only for buffer mode. This is the current end of the buffer.
Note that the buffer is a 36 position ring buffer.
 14. **A** - valid only for buffer mode. Provides the available empty positions of the buffer.

7.1.2. Control

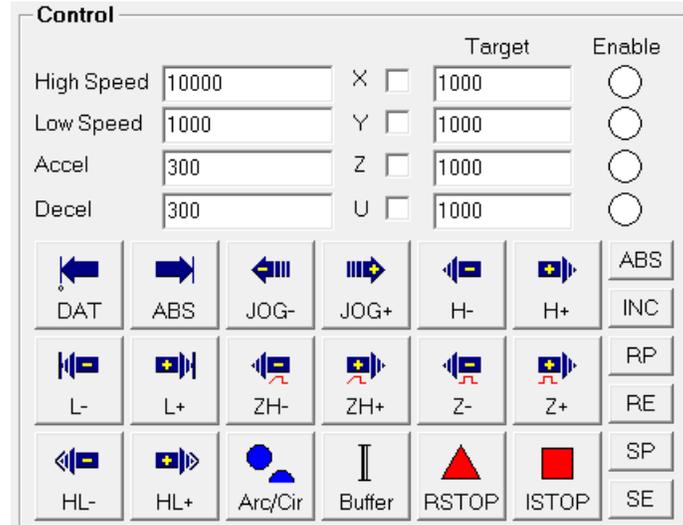


Figure 7-5

1. **High/Low Speed, Accel, and Decel** - use these to set the speed of a move command. To give each axis individual speed parameters, enter HS[axis], LS[axis], and ACC[axis] commands via the command terminal.
2. **X/Y/Z/U Check Boxes** - A move performed by one of the move buttons will apply to the selected axis.
3. **Target (X,Y,Z,U)** - positional moves will use this value as the target position.
4. **Enable (X,Y,Z,U)** - motor power is turned on or off by clicking on these circles.
5. **ABS** - Set absolute move mode
6. **INC** - Set incremental move mode
7. **RP** - Reset pulse counter for the specified axis. Not allowed if StepNLoop is enabled.
8. **RE** - Reset encoder counter for the specified axis.
9. **SP** - Set pulse counter for the specified axis. This will use the value in the Target position field.
10. **SE** - Set encoder counter for the specified axis. This will use the value in the Target position field.
11. **ISTOP** - the motion is immediately stopped without deceleration.
12. **RSTOP** - the motion is stopped with deceleration.
13. **Z+/Z-** - Home the axis using only the encoder index channel.
14. **H+/H-** - Home the axis at high speed using only the home input.
15. **ZH+/ZH-** - Home the axis using the home input and encoder index channel.
16. **HL+/HL-** - Home the axis at high speed and low speed using only the home sensor.

17. **L+/L-** - Home the axis using only the limit sensor.
18. **JOG+/JOG-** - Move the axis indefinitely in the positive or negative direction.
19. **ABS** - Perform absolute move. If more than one axis is selected, an interpolated move will result.
20. **DAT** - Return to 0 position. If more than one axis is selected, an interpolated move will result.
21. **Buffer move Tool** – Clicking on this button will provide the user with an interface to load buffer move commands to the PMX-4EX-SA.

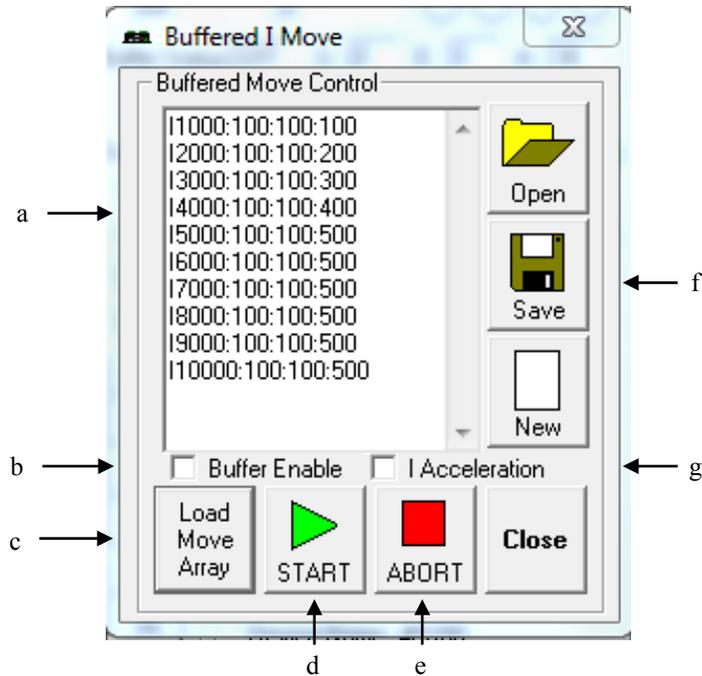


Figure 7-6

- a. **Buffer Array List** – Enter the desired list of buffer commands here. Once the list is loaded, if the number of commands is greater than 36 (max buffer size), the program will automatically send the remaining commands to the PMX-4EX-SA as spaces clears up in the buffer.
- b. **Buffer enable** – Enable/Disable buffer move mode
- c. **Load Move Array** – Once the buffer array list is created, click here to load the array list to the program.
- d. **Start** – Once the array has been loaded, click here to begin sending the buffered commands to the PMX-4EX-SA. Note that after the “START” button is clicked, the buffer commands will not begin to be sent to the PMX-4EX-SA until the Buffer I Move window is closed.
- e. **Abort** – Stop sending buffer commands to the PMX-4EX-SA. Also disables buffer move mode.

- f. **Open/Save/New** – Allows users to save/open or create new buffer array lists
 - g. **I Accel** – Enable/Disable buffered I move acceleration.
22. **Arc/Circle Tool** – Clicking on this button will provide the user with an interface to perform Arc/Circle XY moves. This will bring up the window shown in Figure 7-7.

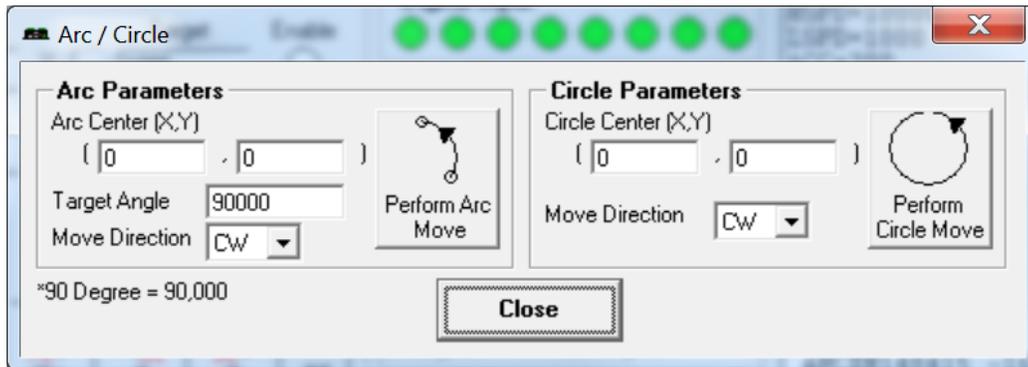


Figure 7-7

Note that after an arc or circle move is started, the position/speed values of the main control window will not begin to update until the above window is closed.

7.1.3. On-The-Fly-Speed Control

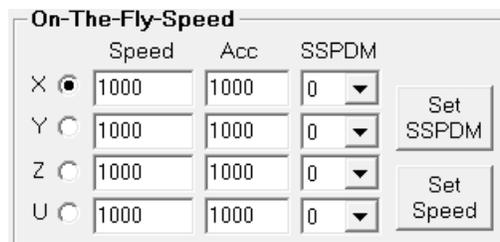


Figure 7-8

1. **Select X/Y/Z/U axis.**
2. **Speed** - configure the destination speed of the axis.
3. **Acc** - set the acceleration used during an on-the-fly speed change.
4. **SSPDM** - select the SSPD mode for the axis. See section 6.3 for details.
5. **Set SSPDM** - set the SSPDM displayed in the dropdown menu.
6. **Set Speed** - perform an on-the-speed change. Make sure that the SSPDM mode has been set before issuing the on-the-fly speed operation.

7.1.4. On-The-Fly-Position Control



Figure 7-9

1. **Position** - set the new target position for the specified axis.
2. **SP** - Perform an on-the-fly position change. Only valid during single axis position moves.

7.1.5. Sync Outputs

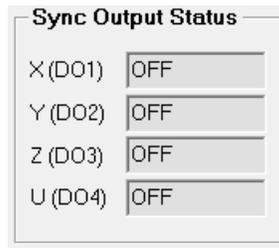


Figure 7-10

1. **Sync output status (X,Y,Z,U)** - displays the current sync output status.
 - OFF
 - WAITING
 - TRIGGERED

7.1.6. Digital Input/Output

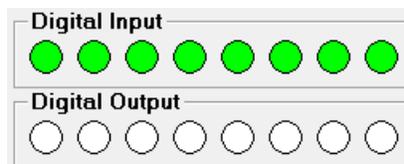


Figure 7-11

1. **Digital Input** - displays the current status of DI1-DI8
2. **Digital Outputs** - displays the current status of DO1-DO8. To turn on/off a digital output, click on the corresponding circle.

7.1.7. Analog Inputs

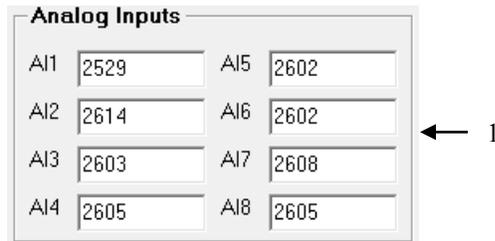


Figure 7-12

1. **Analog Inputs** - displays the analog input status of AI1-AI8. Units are in mV.

7.1.8. Program File Control



Figure 7-13

1. **Open** – Open standalone program
2. **Save** – Save standalone program
3. **New** – Clear the standalone program editor
4. **View** – View the compiled program

7.1.9. Standalone Program Editor



Figure 7-14

1. **Text Program** – Text box for writing and editing a standalone program.
2. **Clear Code Space** – Clear the code space on the PMX-4EX-SA.

7.1.10. Standalone Program Control

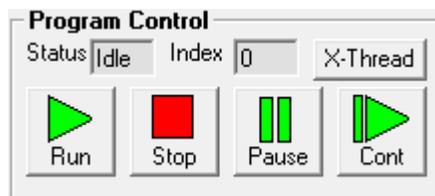


Figure 7-15

1. **Run** – run the standalone program (PRG 0)
2. **Stop** – stop the standalone program (PRG 0)
3. **Pause** – pause the standalone program (PRG 0)
4. **Cont** – continue the paused standalone program (PRG 0)

-
5. **XThread** – open the Standalone Program Control. Will allow for control of all 4 standalone programs.
 6. **Index** – the current line of low-level code that is being executed.
 7. **Status** – the current status of the standalone program (PRG 0)
 - Idle – Program is not running.
 - Running – Program is running.
 - Paused – Program is paused.
 - Error – Program is in an error state.

7.1.11. Standalone Program Compile/Download/Upload



Figure 7-16

1. **Compile** – Compile the standalone program
2. **Download** – Download the compiled program
3. **Upload** – Upload the standalone program from the controller

7.1.12. Setup



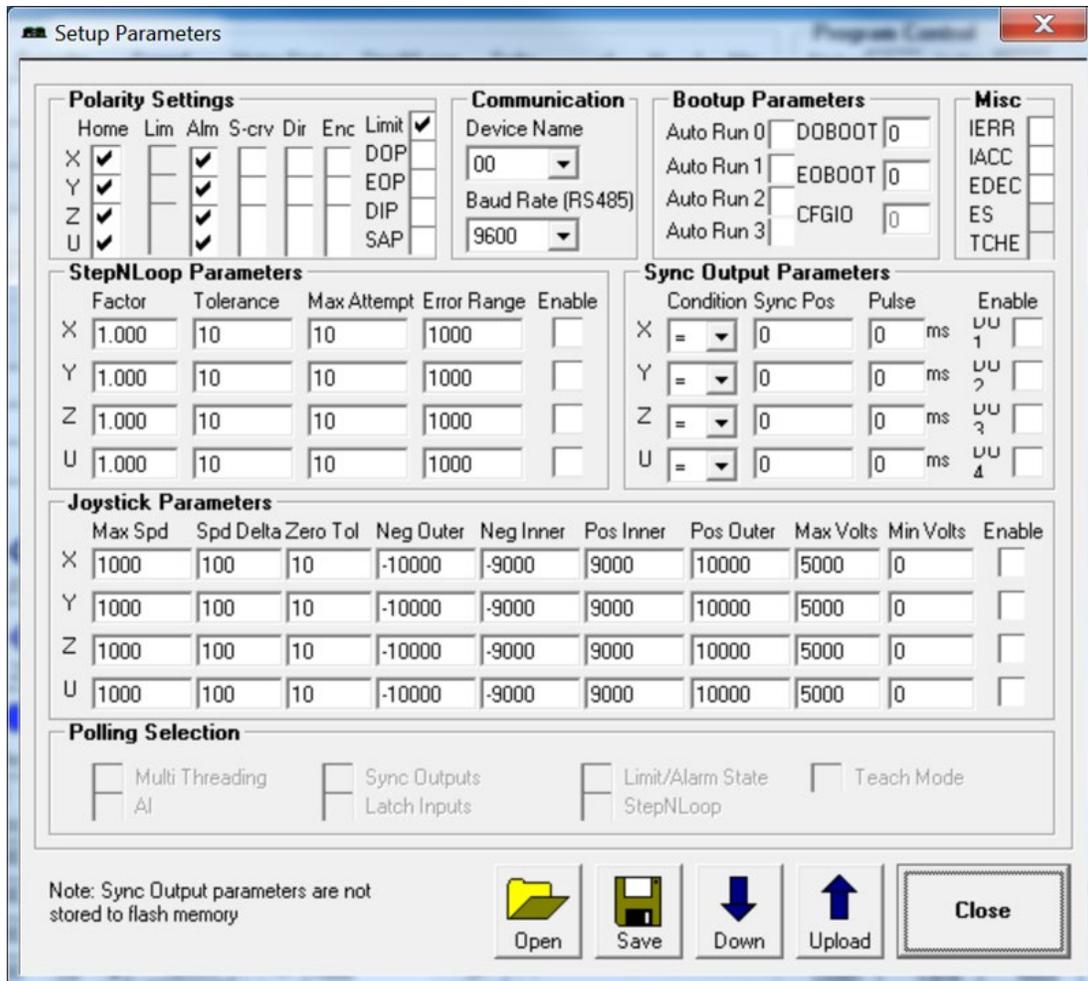


Figure 7-17

1. **Polarity Settings**
 - a. **Home/Alarm/Dir/Enc** (X,Y,Z,U) - set the home, alarm, pulse direction, and encoder direction polarity.
 - b. **S-crv** (X,Y,Z,U) - set s-curve enable/disable for each axes.
 - c. **Limit** - set the limit polarity. Note that the limit polarity is fixed either high/low for all axes.
 - d. **DOP** - set the digital output polarity
 - e. **DIP** - set the digital input polarity
 - f. **EOP** - set the enable output polarity
 - g. **SA Err** - set the return jump line for standalone error handling
2. **StepNLoop Parameters** - set the StepNLoop parameters for all axis. See section 6.22 for details.
3. **Joystick Parameters** - set the joystick parameters for all axis. See section 6.20 for details.
4. **Communication**
 - a. **Device Name** - set device name: [4EX00-4EX99]

-
- b. **Baud Rate (RS485)** - set baud rate (used for RS-485 communication)
 - 5. **Bootup Parameters**
 - a. **Auto Run N** - start the selected standalone program on bootup.
 - b. **DOBOOT/EOBOOT** - set the digital and enable output configuration status on boot-up.
 - 6. **Miscellaneous Settings**
 - a. **IERR** - enable/disable the ignore limit/alarm error feature
 - b. **IACC** - enable/disable I move acceleration (used with buffered I commands)
 - c. **EDEC** - enable/disable unique deceleration.
 - 7. **Sync Output Parameters** - set the sync output parameters for all axis. See section 6.18 for details.
 - 8. **Open/Save** parameters to file.
 - 9. **Upload/Download** parameters to and from flash memory.

7.1.13. Terminal

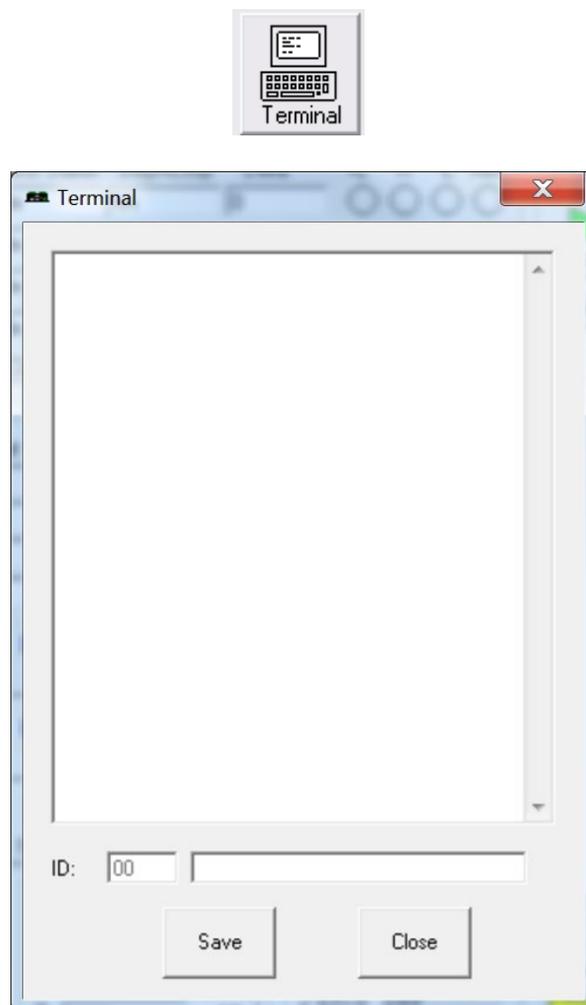


Figure 7-18

1. **Response Box** – Displays sent command as well as corresponding response
2. **Device Name** – Device name of the PMX-4EX-SA. In USB communication mode, the field is fixed. In RS-485 communication mode, this field can be modified so that the user can communicate with a specific device.
3. **Command line** – ASCII command line
4. **Save** – Save the current contents of the Response Box to file

7.1.14. Latches

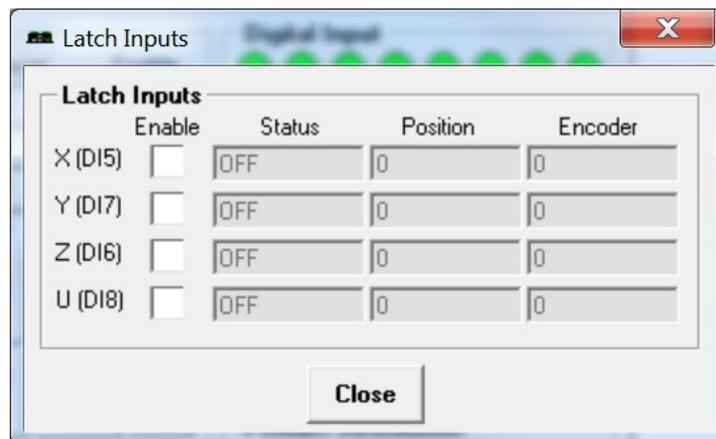
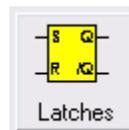


Figure 7-19

1. **Enable** – Enable latch support for X/Y/Z/U axis
2. **Status** – Display the current latch status
 - ON: Latch has been triggered
 - OFF: Latch is disabled
 - WAITING: Latch is enabled and is waiting to be triggered.
3. **Position** – Latched position value. Note that this is reset to 0 each time that the latch is enabled.
4. **Encoder** – Latched encoder value. Note that this is reset to 0 each time that the latch is enabled.

7.1.15. Variable Status



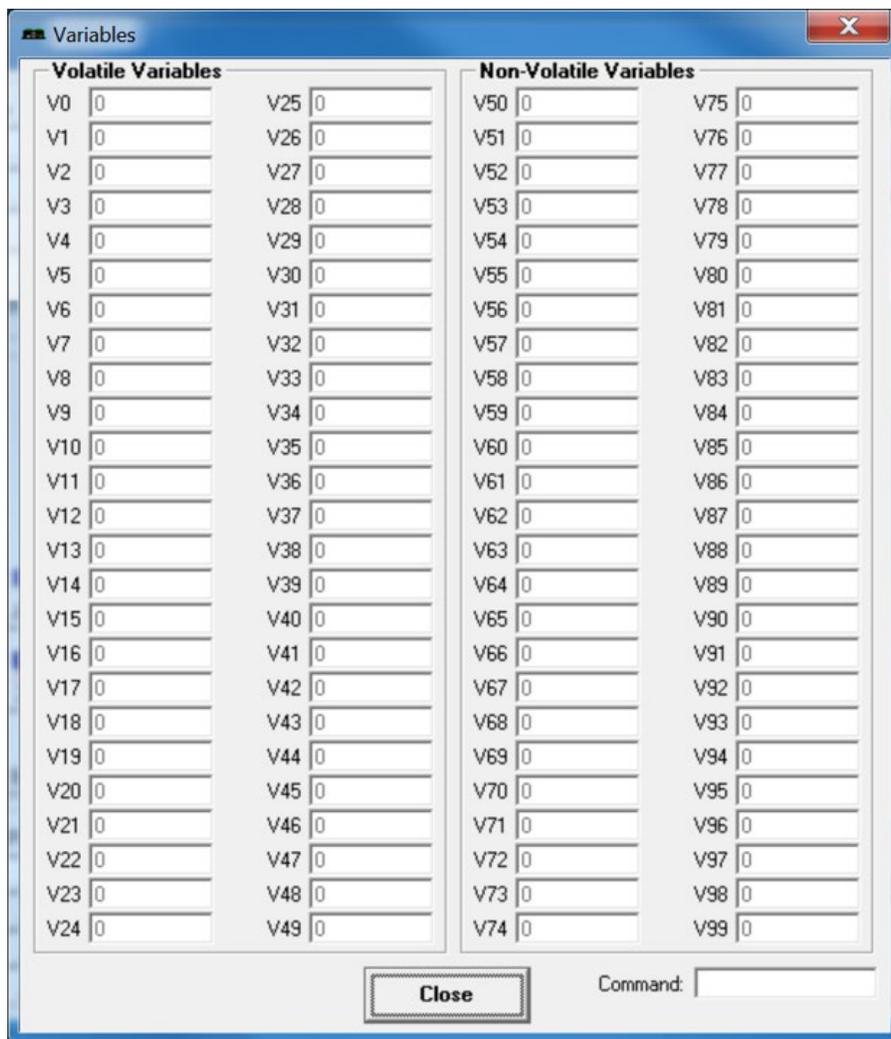


Figure 7-20

1. **Volatile Variables** – status of volatile variable V0-V49
2. **Non-volatile Variables** – status of non-volatile variable V50-V99
3. **Command line** – set variables using V[0-99]=[value] syntax

7.2. DXF Converter

The DXF Converter allows a 2D DXF file to be converted to a standard A-Script standalone program. The program can be downloaded to the PMX-4EX-SA to allow for easy development.

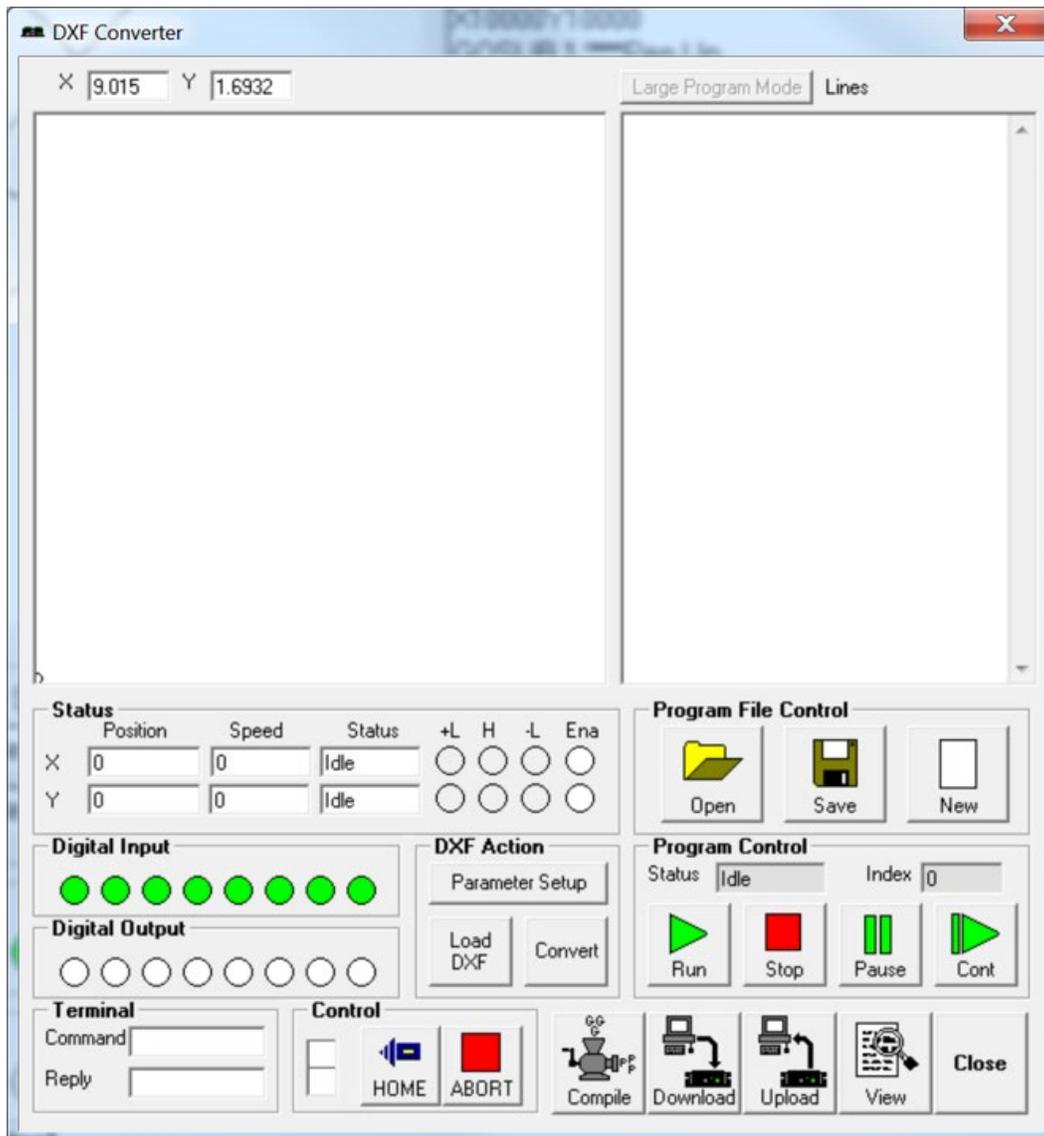


Figure 7-21

7.2.1. DXF Viewer

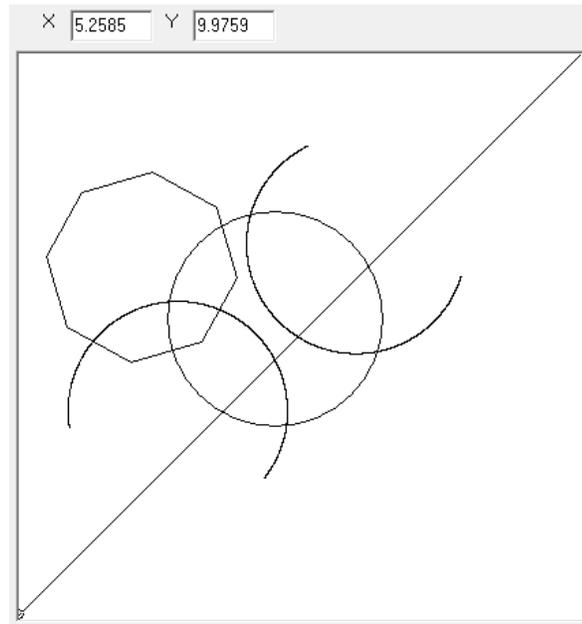


Figure 7-22

The DXF Viewer will display the last loaded DXF file. The "Load DXF File" button can be used to load a 2D file.

7.2.2. Status

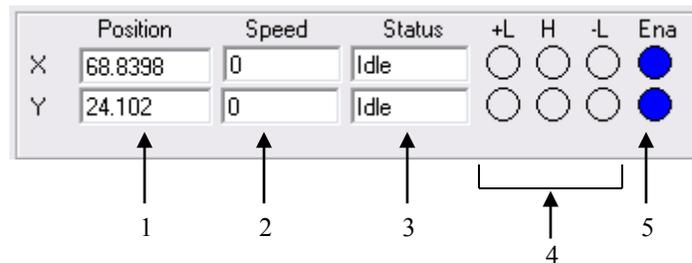


Figure 7-23

1. **Current inch/mm position** (X,Y axes).
2. **Current speed** (X,Y axes).
3. **Motor status** (X,Y axes)
 - Idle – motor is not moving.
 - Accel – motor is accelerating
 - Const – motor is running in constant speed
 - Decel – motor is decelerating
 - +LimError – plus limit error
 - -LimError – minus limit error
4. **+Limit, -Limit, Home status**

-
5. **Enable** status (X,Y axes). To enable/disable the axis, click on the corresponding circle

7.2.3. Control



Figure 7-24

1. **Home X** - homes the X axis in the negative direction using only the home input.
2. **Home Y** - homes the Y axis in the negative direction using only the home input.
3. **Abort** - immediately stops all movement

7.2.4. DXF Action



Figure 7-25

1. **Parameter Setup** – allows the user to setup the scaling of the DXF conversion. Once the button is clicked, the following screen appears:

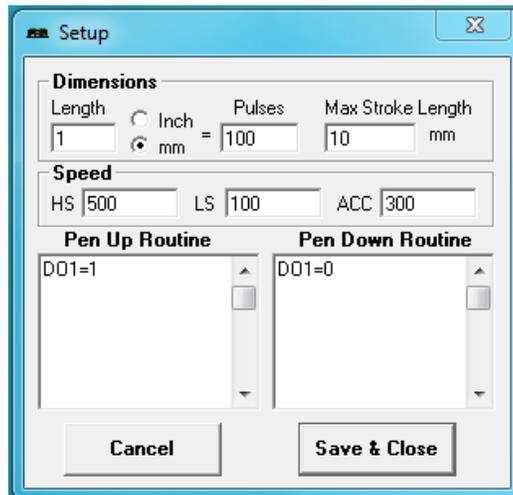


Figure 7-26

- a. **Length/Pulse** – Select relationship between number of pulses and length of movement in terms of inch or millimeter.
 - b. **Max Stroke Length** – The largest allowable stroke length. This will affect the scaling of the DXF viewer box
 - c. **HS/LS/ACC** - High Speed, Low Speed and Acceleration settings
 - d. **Pen Up Routine** – The routine when the XY axis is not in position
 - e. **Pen Down Routine** – The routine when the XY axis is in position
 - f. **Save & Close** parameters and exit setup
2. **Load DXF File** – browse your PC for a 2D DXF file to load into the DXF Viewer
 3. **Convert File** – Convert the loaded DXF file into PMX-4EX-SA compatible motion commands. The result will be loaded into the Motion Conversion Program box. The conversion scaling and speed will depend on the parameters set in the Parameter setup box.

7.2.5. Motion Conversion Program

```
*****SET SPEED*****
HSPD=10000
LSPD=100
ACC=100
GOSUB 1 ****Pen Up
X0Y0 ****Go to start position
X26164Y34325
GOSUB 2 ****Pen Down
X27023Y41834
X20148Y44971
X15039Y39402
X18758Y32822
X26164Y34325
GOSUB 1 ****Pen Up
X21330Y39926
GOSUB 2 ****Pen Down
X21233Y37126
GOSUB 1 ****Pen Up
X20360Y29573
GOSUB 2 ****Pen Down
X20745Y19105
GOSUB 1 ****Pen Up
X21522Y28999
GOSUB 2 ****Pen Down
X21612Y20088
```

Figure 7-27

View DXF file code once it is converted to A-Script standalone programming language. To populate this box, first select a DXF file by clicking on “Load DXF File,” secondly click “Convert File.”

This text box can be edited and compiled to customize your motion program

7.2.6. DXF Converter – Important Notes

Creating a compatible DXF file:

- **Margins:** Many times a DXF file may have extra text or margins describing the project. These should be removed. The only elements in the DXF file should be the picture that is desired to be drawn.
- **Radius Size:** PMX-4EX-SA does not allow a radius larger than 134216773 pulses on arc or circular moves. To keep your radius moves smaller than 134,216,773, decrease the **Length/Pulse Factor**.
- **Picture positioning:** A DXF file cannot contain any or part of an image that is not in quadrant I (i.e. all x,y positions of the DXF need to be positive). See figure below:

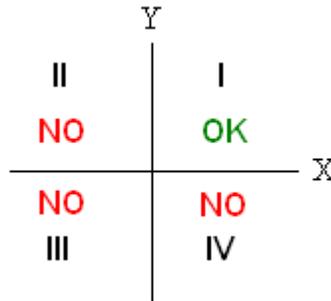


Figure 7-28

- **Export Type:** When exporting to a DXF type, the DXF must be "AutoCad R12".

Scaling the DXF Viewer Box:

When loading a DXF file, the picture may seem too small. See below for an example.

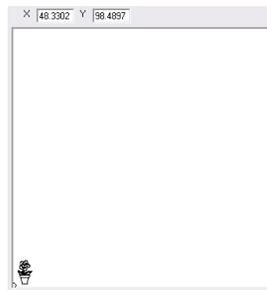


Figure 7-29

In this case, the window is zoomed out too much. To zoom in, increase the **Max Stroke Length** parameter.

In the case where you do not see any picture or the picture is cut off, the window is zoomed in too much. See below for an example.

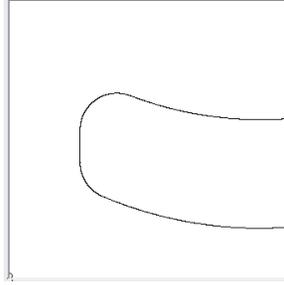


Figure 7-30

To zoom out, decrease the **Max Stroke Length** parameter.

When creating a DXF file, the scaling is maintained when you load it into the DXF converter.

8. ASCII Language Specification

Important Note: All the commands described in this section are interactive ASCII commands and are not analogous to standalone commands. Refer to section 9 for details regarding standalone commands.

PMX-4EX-SA ASCII protocol is case sensitive. All commands should be in upper case letters. The [axis] value in the relevant commands below can be "X", "Y", "Z", or "U".

An invalid command is returned with a "?" Always check for the proper reply when a command is sent.

For USB and RS-485 communication, the commands detailed in Table 8-1 are valid.

8.1. ASCII Command Set

Command	Description	Return
ABORT	Immediately stops all axis if in motion. Abort turns off the buffered move.	OK
ABORT[axis]	Immediately stops the individual axis if in motion. Abort turns off the buffered move.	OK
ABS	Set the move mode to absolute mode.	OK
ACC	Return the current global acceleration in milliseconds.	milliseconds
ACC=[value]	Set global acceleration in milliseconds.	OK
ACC[axis]	Return current individual acceleration in milliseconds.	milliseconds
ACC[axis]=[value]	Set individual acceleration in milliseconds.	OK
AI[1-8]	Return the analog input status in milli-volts.	[0-5000]
ARCP[X]:[Y]:[θ]	XY Arc interpolation move (CW direction).	OK
ARCN[X]:[Y]:[θ]	XY Arc interpolation move (CCW direction).	OK
BF	Disable buffered mode.	OK
BO	Enable buffer mode.	OK
CIRP[X]:[Y]	XY Circular interpolation move (CW direction).	OK
CIRN[X]:[Y]	XY Circular interpolation move (CCW direction).	OK
CLR[axis]	Clear the axis limit, alarm, or StepNLoop error status bit.	OK
DB	Return the current baud rate of the controller.	Refer to Table 6-2
DB=[value]	Set baud rate of the controller.	OK
DEC	Return the current global deceleration in milliseconds.	milliseconds
DEC=[Value]	Set the global deceleration value in milliseconds.	OK
DEC[axis]	Return current individual deceleration in milliseconds.	milliseconds
DEC[axis]=[value]	Set the individual deceleration value in milliseconds.	OK
DI	Return the status of the digital inputs.	Refer to Table 6-4
DI[1-8]	Return the bit status of general purpose digital input.	[0, 1]
DIP	Return the digital input polarity.	[0, 1]
DIP=[0, 1]	Set the digital input polarity.	OK
DO	Return the status of the digital outputs.	Refer to Table 6-5
DO=[value]	Set the digital outputs. Refer to Table 6-5.	OK
DO[1-8]	Return status of individual digital output.	[0, 1]
DO[1-8]=[value]	Set the individual digital output. Refer to Table 6-5.	OK
DOBOOT	Return the DO configuration at boot-up.	[0-255]

DOBOOT=[value]	Set the DO configuration at boot-up.	OK
DOP	Return the digital output polarity.	[0,1]
DOP=[0,1]	Set the digital output polarity.	OK
DN	Return the device name.	[4EX00-4EX99]
DN=[value]	Set the device name.	OK
DX[axis]	Return the StepNLoop delta value of axis.	28-bit number
EDEC	Return the enable unique deceleration status.	[0,1]
EDEC=[0,1]	Set the enabled unique deceleration status.	OK
EO	Returns the enable output status.	Refer to Table 6-2
EO=[value]	Set the enable outputs.	OK
EO[1-4]	Return the individual enable output status.	[0,1]
EO[1-4]=[value]	Set the individual enable output.	OK
EOBOOT	Return the EO configuration at boot-up.	Refer to Table 6-2
EOBOOT=[value]	Set the EI configuration at boot-up.	OK
EOP	Return the enable output polarity.	[0,1]
EOP=[0,1]	Set the enable output polarity.	OK
E[axis]	Return the encoder value of the axis.	28-bit number
E[axis]=[value]	Set the encoder value of the axis.	OK
GS[0-31]	Call a subroutine that has been previously stored to flash memory.	OK
HS	Return the global high speed setting.	[1-6,000,000]PPS
HS=[value]	Set the global high speed.	OK
HS[axis]	Return the individual high speed setting.	[1-6,000,000]PPS
HS[axis]=[value]	Sets individual high speed.	OK
H[axis][+/-][mode]	Homes the motor in plus [+] or minus [-] direction using the specified homing mode.	OK
I[X axis]: [Y axis]: [Z axis]: [speed]	XYZ interpolated move. Target move values are separated by ':' character. Last value is the constant speed that will be used in the move. Buffer mode must be enabled.	OK
IACC	Return the automatic acceleration setting during buffered interpolated move status.	[0,1]
IACC=[0,1]	Set the automatic acceleration setting during buffer interpolated move status.	OK
IERR	Return the ignore limit/alarm error status.	[0,1]
IERR=[0,1]	Set the ignore limit/alarm error status.	OK
INC	Set the incremental move mode.	OK
JE	Return the joystick enable status.	[0-15]
JE=[0-15]	Set the joystick enable status.	OK
J[axis][+/-]	Jogs the axis in plus [+] or minus [-] direction.	OK
JV[1-12]	Return the joystick speed, delta and tolerance settings.	28-bit number
JV[1-12]=[value]	Set the joystick speed, delta and tolerance settings.	OK
JL[1-16]	Return the joystick soft limit settings.	32-bit number
JL[1-16]=[value]	Set the joystick soft limit settings.	OK
JMAX[axis]	Return the joystick control maximum voltage level.	[0-5000 mV]
JMAX[axis]=[value]	Set the joystick control maximum voltage level.	OK
JMIN[axis]	Return the joystick control minimum voltage level.	[0-5000 mV]
JMIN[axis]=[value]	Set the joystick control minimum voltage level.	OK
LS	Return the global low speed setting.	[1-6,000,000]
LS=[value]	Set the global low speed.	OK
LS[axis]	Return the individual low speed setting.	[1-6,000,000]
LS[axis]=[value]	Set the individual low speed.	OK
LT[axis]=[0,1]	Set the enable position latch feature.	OK

LT[axis]E	Returns latched encoder position	28-bit number
LT[axis]P	Return the latched pulse position.	28-bit number
LT[axis]S	Returns latch status.	Refer to Table 6-7
MST	Returns all motor status, buffer move status, and move mode status for all axis.	Refer to Table 6-3
PE	Returns current encoder counter values of all axis.	[X Enc Position]: [Y Enc Position]: [Z Enc Position]: [U Enc Position]
PO[axis]	Return the current polarity setting.	Refer to Table 6-13
PO[axis]=[value]	Set the polarity setting. Refer to Table 6-13.	OK
PP	Returns current pulse counter values of all axis.	[X Pulse Position]: [Y Pulse Position]: [Z Pulse Position]: [U Pulse Position]
PS	Returns current pulse speed values of all axis.	[X Speed]: [Y Speed]: [Z Speed]: [U Speed]
P[axis]	Return the position value of the individual axis.	28-bit number
P[axis]=[value]	Set the position value of the individual axis.	OK
SA[0-1274]	Return the standalone line.	
SA[0-1274]=[value]	Set the standalone line.	OK
SAP	Return the standalone error handling return setting.	[0,1]
SAP=[0,1]	Set the standalone error handling return setting.	OK
SASTAT	Return the standalone program status. Refer to Table 6-18.	[0-4]
SCV[axis]	Returns the s-curve acceleration/deceleration setting.	[0,1]
SCVX=[0,1]	Returns the s-curve acceleration/deceleration setting.	OK
SLA[axis]	Return the StepNLoop maximum attempt value.	32-bit number
SLA[axis]=[value]	Set the StepNLoop maximum attempt value.	OK
SLE[axis]	Return the StepNLoop error range value.	32-bit number
SLE[axis]=[value]	Set the StepNLoop error range value.	OK
SLR[axis]	Return the StepNLoop ratio of axis (PPR / CPR).	[0.001-999.999]
SLR[axis]=[value]	Set the StepNLoop ratio of axis (PPR / CPR).	OK
SLS[axis]	Return the StepNLoop status. Refer to Table 6-15.	[0-12]
SLT[axis]	Return the StepNLoop tolerance.	32-bit number
SLT[axis]=[value]	Set the StepNLoop tolerance of axis.	OK
SL[axis]	Return the StepNLoop enable of axis.	[0,1]
SL[axis]=[value]	Set the StepNLoop enable of axis.	OK
SLOAD	Returns the standalone program run on boot parameter. Refer to Table 6-20.	[0-15]
SLOAD=[value]	Set the standalone program run on boot parameter. Refer to Table 6-20.	OK
SR[0-3]=[value]	Control the standalone program. Refer to Table 6-17.	OK
SPC[0-3]	Returns the program counter for the specified standalone program.	[0-1274]
SSPD[axis]=[value]	Perform on-the-fly speed change to the specified speed.	OK
SSPDM[axis]	Return the on-the-fly speed change mode. Refer to Table A-1.	[0-7]
SSPDM[axis]=[value]	Set on-the-fly speed change mode. Refer to Table A-1.	OK
STOP	Performs ramp down to stop for all axis if in motion.	OK
STOP[axis]	Performs ramp down to stop for individual axis.	OK

STORE	Store settings to flash. Refer to Table 6-21.	OK
SYN[axis]C	Return the sync output configuration. Refer to Table 6-9.	[1-3]
SYN[axis]C=[1-3]	Set the sync output configuration. Refer to Table 6-9.	OK
SYN[axis]F	Turn sync output feature OFF.	OK
SYN[axis]O	Turn sync output feature ON.	OK
SYN[axis]P	Return the sync trigger position.	28-bit number
SYN[axis]P=[value]	Set the sync trigger position.	28-bit number
SYN[axis]T	Return the sync output pulse width time (ms). Only applicable if sync output configuration is set to 1.	[0-10]
SYN[axis]T=[value]	Set the sync output pulse width time (ms). Only applicable if sync output configuration is set to 1.	OK
T[axis][value]	Perform on-the-fly target position change.	OK
TOC	Returns the communication time-out parameter in milliseconds.	milliseconds
TOC=[value]	Set the communication time-out parameter in milliseconds.	OK
TR	Return the timer register value in milliseconds.	milliseconds
TR=[value]	Set the timer register value in milliseconds.	OK
V[0-99]	Return the standalone variable value.	32-bit number
V[0-99]=[value]	Set the standalone variable value.	OK
VER	Return the controller firmware version	V[#]
X[target X] Y[target Y] Z[target Z] U[target U]	Perform an individual/interpolated move	OK

Table 8-1

8.2. Error Codes

If an ASCII command cannot be processed by the PMX-4EX-SA, the controller will reply with an error code. See below for possible error responses:

Error Code	Description
?[Command]	The ASCII command is not understood by the PMX-4EX-SA
?ALARM	A move command is sent while the alarm is on.
?BUFFER FULL	An attempt to add a move to a full buffer has been made.
?ERRORED	Command has been issued while the controller is in error state.
?In Incremental mode	A circle or arc interpolation move has been issue while the PMX-4EX-SA is in incremental mode.
?Index out of Range	The index for the command sent to the controller is not valid.
?LIMIT	A move command is sent while the axis has a limit error.
?PULSING	A move or position change command is sent while the PMX-4EX-SA is outputting pulses.
?SSPD Error	An on-the-fly speed change was issued without initialized the SSPDM parameter.
?Sub not Initialized	Call to a subroutine using the GS command is not valid because the specified subroutine has not been defined.
?Timer Running	An attempt to set the timer register while it is running has been made.

Table 8-2

9. Standalone Language Specification

Important Note: All the commands described in this section are standalone language commands and are not analogous to ASCII commands.

9.1. Standalone Command Set

Command	R/W	Description	Example
;	-	Comment notation. Comments out any text following ; in the same line.	;This is a comment
ABORT	W	Immediately stop all motion for all axis.	ABORT
ABORT[axis]	W	Immediately stop all motion for a single axis	ABORTX ABORTZ
ABS	W	Set the move mode to absolute mode.	ABS X1000 ;move to position 1000
ACC	R/W	Set/get the global acceleration setting. Unit is in milliseconds.	ACC=500 ACC=V1
ACC[axis]	R/W	Set/get the individual acceleration setting. Unit is in milliseconds.	ACCX=500 ACCY=V1
AI[1-8]	R	Get the analog input value.	IF AI2>2500 DO=1 ENDIF V2=AI1
ARCP[X]:[Y]:[θ]	W	Perform ARC move in CW direction using the X and Y axis.	;move CW to angle 90 degrees ARCP1000:0:90000
ARCN[X]:[Y]:[θ]	W	Perform ARC move in CCW direction using the X and Y axis. [X] and [Y] represent the arc center.	;move CCW to angle 90 degrees ARCN1000:0:90000
BUFOFF	W	Turn the buffer move mode setting off.	BUFOFF
BUFON	W	Turn the buffer move mode setting on.	BUFON
CIRP[X]:[Y]	W	Perform circle move in CW direction using the X and Y axis. [X] and [Y] represent the circle center.	;Perform CW circle around (0,0) CIRP0:0
CIRN[X]:[Y]	W	Perform circle move in CCW direction using the X and Y axis. [X] and [Y] represent the circle center.	; Perform CCW circle around (0,0) CIRN0:0
DEC	R/W	Set/get the global deceleration setting. Unit is in milliseconds.	DEC=500 DEC=V1
DEC[axis]	R/W	Set/get the individual deceleration setting. Unit is in milliseconds.	DECX=500 DECY=V1
DELAY	W	Set a delay in milliseconds. Assigned value is a 32-bit unsigned integer or a variable.	DELAY=1000 ;1 second DELAY=V1 ;assign to variable
DI	R	Return status of digital inputs. See Table 6-4 for bitwise assignment.	IF DI=0 DO=1 ;Turn on DO1 ENDIF V2=DI
DI[1-8]	R	Get individual bit status of digital inputs. Will return [0,1]. See Table 6-4 for bitwise assignment.	IF DI1=0 DO=1 ;Turn on DO1 ENDIF V3=DI1
DO	R/W	Set/get digital output status. See Table 6-5 for bitwise assignment.	DO=2 ;Turn on DO2

DO[1-8]	R/W	Set/get individual bit status of digital outputs. Range for the bit assigned digital outputs is [0,1].	DO2=1 ;Turn on DO2
ECLEAR[axis]	W	Clear any motor status errors.	ECLEARX
EO	R/W	Set/get the enable output status. Refer to Table 6-2.	EO=3 ;Enable the X and Y motor
EO[1-4]	R/W	Set/get individual bit status of the enable outputs.	EO3=1 ;Enable the Z motor
E[axis]	R/W	Set/get the current encoder position.	EX=1000 ;Set to X enc to 1000 V1=EU ;Read current U encoder
GOSUB [0-31]	-	Call a subroutine that has been previously stored to flash memory.	GOSUB 0 END
HLHOME[axis][+/-]	W	Home the motor using the home input at low and high speeds in the specified direction. See section 6.13.5 for details.	HLHOMEX+ ;positive X home WAITX ;wait for X home move
HOME[axis][+/-]	W	Home the motor using the home input at high speed in specified direction. See section 6.13.1 for details.	HOMEX- ;negative X home WAITX ;wait for X home move
HSPD	R/W	Set/get the global high speed setting. Unit is in pulses/second.	HSPD=1000 HSPD=V1
HSPD[axis]	R/W	Set/get the individual high speed setting. Unit is in pulses/second.	HSPDY=1000 HSPDZ=V1
IF ELSEIF ELSE ENDIF	-	Perform a standard IF/ELSEIF/ELSE conditional. Any command with read ability can be used in a conditional. ENDIF should be used to close off an IF statement. Conditions [=, >, <, >=, <=, !=] are available	IF DI1=0 DO=1 ;Turn on DO1 ELSEIF DI2=0 DO=2; Turn on DO2 ELSE DO=0; Turn off DO ENDIF
INC	W	Set the move mode to incremental mode.	INC X1000 ;increment by 1000
JOG[axis][+/-]	W	Move the motor indefinitely in the specified direction.	JOGX+
JOYENA	W	Set the joystick enable setting. See section 6.20 for details.	JOYENA=3 ;enable joystick X,Y
JOYHS[axis]	W	Set the high speed setting for joystick control. See section 6.20 for details.	JOYHSX=2000 JOYHSZ=5000
JOYDEL[axis]	W	Set the speed change delta for joystick control. See section 6.20 for details.	JOYDELZ=100 JOYDELU=200
LHOME[axis][+/-]	W	Home the motor using the limit inputs in the specified directions. See section 6.13.2 for details.	LHOMEX+ ;positive home WAITX
LSPD	R/W	Set/get the global low speed setting. Unit is in pulses/second.	LSPD=100 LSPD=V3
LSPD[axis]	R/W	Set/get the individual low speed setting. Unit is in pulses/second.	LSPDX=100 LSPDY=V1
MST[axis]	R	Get the current motor status of the motor of an axis. See Table 6-3 for motor status assignment.	

PRG [0-3] END	-	Used to define the beginning and end of a main program. Four standalone programs are available	PRG 0 ;main program END
PSX	R	Get the current motor speed.	V2=PSX ;Set V2 to speed
P[axis]	R/W	Set/get the current motor position.	PX=1000 ;Set to X pos to 1000 V1=PY ;Read current Y position
SCV[axis]	R/W	Set/get the s-curve enable setting.	SCVX=1 ;enable X s-curve V1=SCVY ;read Y s-curve
SLS[axis]	R	Get the current StepNLoop status. See Table 6-15 for details.	V3=SLSX ;Set to status
SL[axis]	W	Enable/disable StepNLoop closed loop mode.	SLX=1 ;Enable StepNLoop SLX=0 ;Disable StepNLoop
SR[0-3]	W	Set the standalone control for the specified program. See Table 6-17.	SR0=0 ;Turn off program 0
SSPDM[axis]	W	Set the SSPD mode. Must be done before move command. See Table A-1 for details.	SSPDMX=1 ;Set SSPD mode SSPDMX=V2 JOGX+ ;Jog the motor
SSPD[axis]	W	Perform an on-the-fly speed change. SSPDM[mode] must be set first.	JOGX+ ;Jog the motor DELAY=1000 ;Wait 1 second SSPDX=1000 ;Change speed SSPDX=V1
STOP	W	Stop all motion using a decelerated stop.	
STOP[axis]	W	Stop motion using a decelerated stop for an individual axis.	STOPX STOPY
STORE	W	Store settings to flash.	STORE
SUB [0-31] ENDSUB	-	Defines the beginning of a subroutine. ENDSUB should be used to define the end of the subroutine.	SUB 1 DO=4 ENDSUB
SYNCFG[axis]	W	Set the sync output configuration. See Table 6-9 for details.	SYNCFGX=2 ;Equal to setting
SYNOFF[axis]	W	Disable the sync output configuration.	SYNOFFX ;Turn off sync
SYNON[axis]	W	Enable the sync output configuration.	SYNONX ;Turn on sync
SYNPOS[axis]	W	Set the sync output reference position.	SYNPOSX=1000 ;Set position SYNPOSX=V1 ;Set position
SYNSTAT[axis]	R	Get the current sync output status. See Table 6-10 for details.	V1=SYNSTATX ;Get status
SYNTIME[axis]	W	Set the sync output pulse width time in milliseconds. Maximum of 10 ms.	SYNTIMEX=5 ;set to 5 ms
TOC	W	Sets the communication time-out parameter. Units is in milliseconds.	TOC=1000 ;1 second time-out
TR	R/W	Set/get the timer register.	TR=1000 IF TR<500 DO=1 ENDIF
U[position]	W	If in absolute mode, move the U motor to [position]. If in incremental mode, move the motor to [current position] + [position].	U1000

V[0-99]	R/W	Set/get standalone variables. The following operations are available: [+] Addition [-] Subtraction [*] Multiplication [/] Division [%] Modulus [>>] Bit shift right [<<] Bit shift left [&] Bitwise AND [] Bitwise OR [~] Bitwise NOT	V1=12345 ;Set V1 to 12345 V2=V1+1;Set V2 to V1 + 1 V3=DI ;Set V3 to DI V4=DO ;Set V4 to DO V5=~EO ;Set V5 to NOT EO
WAIT[axis]	W	Wait for current motion to complete before processing the next line.	X1000 ;move to position 1000 WAITX ;wait for move
WHILE ENDWHILE	-	Perform a standard WHILE loop within the standalone program. ENDWHILE should be used to close off a WHILE loop. Conditions [=, >, <, >=, <=, !=] are available.	WHILE 1=1 ;Forever loop DO=1 ;Turn on DO1 DO=0 ;Turn off DO1 ENDWHILE
X[position]	W	If in absolute mode, move the X motor to [position]. If in incremental mode, move the motor to [current position] + [position].	X1000
X[pos]Y[pos] Z[pos]U[pos]	W	Perform linear interpolated move. This command requires 2 or more axis. If only 1 axis is used, a standard positional move will be performed. If buffer mode is enabled, the move command will be added to the buffer automatically.	X1000Y2000 X1000Y2000U4000 X1000Z3000
Y[position]	W	If in absolute mode, move the Y motor to [position]. If in incremental mode, move the motor to [current position] + [position].	Y1000
ZHOME[axis][+/-]	W	Home the motor using the home input and Z-index. See section 6.13.3 for details.	ZHOMEX+ ;positive X home WAITX
ZOME[axis][+/-]	W	Home the motor using the Z-index only. See section 6.13.4 for details.	ZOMEX- ;negative X home WAITX
Z[position]	W	If in absolute mode, move the Z motor to [position]. If in incremental mode, move the motor to [current position] + [position].	Z1000

Table 9-1

9.2. Example Standalone Programs

9.2.1. Standalone Example Program 1 – Single Thread

Task: Set the high speed and low speed and move the motor to 1000 and back to 0.

```
HSPD=20000      ;* Set the high speed to 20000 pulses/sec
LSPD=1000       ;* Set the low speed to 1000 pulses/sec
ACC=300         ;* Set the acceleration to 300 msec
EO=1           ;* Enable the motor power
X1000          ;* Move to 1000
WAITX          ;* Wait for X-axis move to complete
X0             ;* Move to zero
WAITX          ;* Wait for X-axis move to complete
END            ;* End of the program
```

9.2.2. Standalone Example Program 2 – Single Thread

Task: Move the motor back and forth indefinitely between position 1000 and 0.

```
HSPD=20000      ;* Set the high speed to 20000 pulses/sec
LSPD=1000       ;* Set the low speed to 1000 pulses/sec
ACC=300         ;* Set the acceleration to 300 msec
EO=1           ;* Enable the motor power
WHILE 1=1       ;* Forever loop
    X0          ;* Move to zero
    WAITX      ;* Wait for X-axis move to complete
    X1000      ;* Move to 1000
    WAITX      ;* Wait for X-axis move to complete
ENDWHILE       ;* Go back to WHILE statement
END
```

9.2.3. Standalone Example Program 3 – Single Thread

Task: Move the motor back and forth 10 times between position 1000 and 0.

```
HSPD=20000      ;* Set the high speed to 20000 pulses/sec
LSPD=1000       ;* Set the low speed to 1000 pulses/sec
ACC=300         ;* Set the acceleration to 300 msec
EO=1            ;* Enable the motor power
V1=0            ;* Set variable 1 to value 0
WHILE V1<10     ;* Loop while variable 1 is less than 10
  X0            ;* Move to zero
  WAITX        ;* Wait for X-axis move to complete
  X1000        ;* Move to 1000
  WAITX        ;* Wait for X-axis move to complete
  V1=V1+1      ;* Increment variable 1
ENDWHILE        ;* Go back to WHILE statement
END
```

9.2.4. Standalone Example Program 4 – Single Thread

Task: Move the motor back and forth between position 1000 and 0 only if the digital input 1 is turned on.

```
HSPD=20000      ;* Set the high speed to 20000 pulses/sec
LSPD=1000       ;* Set the low speed to 1000 pulses/sec
ACC=300         ;* Set the acceleration to 300 msec
EO=1            ;* Enable the motor power
WHILE 1=1       ;* Forever loop
  IF DI1=1      ;* If digital input 1 is on, execute the statements
    X0          ;* Move to zero
    WAITX      ;* Wait for X-axis move to complete
    X1000      ;* Move to 1000
    WAITX      ;* Wait for X-axis move to complete
  ENDIF
ENDWHILE        ;* Go back to WHILE statement
END
```

9.2.5. Standalone Example Program 5 – Single Thread

Task: Using a subroutine, increment the motor by 1000 whenever the DI1 rising edge is detected.

```
HSPD=20000      ;* Set the high speed to 20000 pulses/sec
LSPD=1000       ;* Set the low speed to 1000 pulses/sec
ACC=300         ;* Set the acceleration to 300 msec
EO=1           ;* Enable the motor power
V1=0           ;* Set variable 1 to zero
WHILE 1=1      ;* Forever loop
    IF DI1=1    ;* If digital input 1 is on, execute the statements
        GOSUB 1 ;* Move to zero
    ENDIF
ENDWHILE       ;* Go back to WHILE statement
END

SUB 1
    XV1         ;* Move to V1 target position
    WAITX       ;* Wait for X-axis move to complete
    V1=V1+1000 ;* Increment V1 by 1000
    WHILE DI1=1 ;* Wait until the DI1 is turned off so that
    ENDWHILE    ;* multiple increment are not continuously done
ENDSUB
```

9.2.6. Standalone Example Program 6 – Single Thread

Task: If digital input 1 is on, move to position 1000. If digital input 2 is on, move to position 2000. If digital input 3 is on, move to 3000. If digital input 5 is on, home the motor in negative direction. Use digital output 1 to indicate that the motor is moving or not moving.

```
HSPD=20000      ;* Set the high speed to 20000 pulses/sec
LSPD=1000       ;* Set the low speed to 1000 pulses/sec
ACC=300         ;* Set the acceleration to 300 msec
EO=1            ;* Enable the motor power
WHILE 1=1       ;* Forever loop
  IF DI1=1      ;* If digital input 1 is on
    X1000       ;* Move to 1000
    WAITX      ;* Wait for X-axis move to complete
  ELSEIF DI2=1 ;* If digital input 2 is on
    X2000       ;* Move to 2000
    WAITX      ;* Wait for X-axis move to complete
  ELSEIF DI3=1 ;* If digital input 3 is on
    X3000       ;* Move to 3000
    WAITX      ;* Wait for X-axis move to complete
  ELSEIF DI5=1 ;* If digital input 5 is on
    HOMEX-     ;* Home the motor in negative direction
    WAITX      ;* Wait for X-axis home move to complete
  ENDIF
  V1=MSTX      ;* Store the motor status to variable 1
  V2=V1&7     ;* Get first 3 bits
  IF V2!=0     ;* If one of first 3 bits is high (X axis moving)
    DO1=1      ;* Turn on digital output 1
  ELSE         ;* Else if first 3 bits are low (X axis idle)
    DO1=0      ;* Turn off digital output 1
  ENDIF
ENDWHILE      ;* Go back to WHILE statement
END
```

9.2.7. Standalone Example Program 7 – Multi Thread

Task: Program 0 will continuously move the motor between positions 0 and 1000. Simultaneously, program 1 will control the status of program 0 using digital inputs.

```
PRG 0                                ;* Start of Program 0
HSPD=20000                           ;* Set high speed to 20000 pulses/sec
LSPD=500                             ;* Set low speed to 500 pulses/sec
ACC=500                              ;* Set acceleration to 500 msec
WHILE 1=1                             ;* Forever loop
    X0                                ;* Move to position 0
    WAITX                             ;* Wait for the move to complete
    X1000                             ;* Move to position 1000
    WAITX                             ;* Wait for the move to complete
ENDWHILE                              ;* Go back to WHILE statement
END                                    ;* End Program 0

PRG 1                                ;* Start of Program 1
WHILE 1=1                             ;* Forever loop
    IF DI1=1                          ;* If digital input 1 is triggered
        ABORTX                        ;* Stop movement
        SR0=0                         ;* Stop Program 1
    ELSE                               ;* If digital input 1 is not triggered
        SR0=1                         ;* Run Program 1
    ENDIF
ENDWHILE                              ;* Go back to WHILE statement
END                                    ;* End Program 1
```

9.2.8. Standalone Example Program 8 – Multi Thread

Task: Program 0 will continuously move the motor between positions 0 and 1000. Simultaneously, program 1 will monitor the communication time-out parameter and triggers digital output 1 if a time-out occurs. Program 1 will also stop all motion, disable program 0 and then re-enable it after a delay of 3 seconds when the error occurs.

```
PRG 0                                ;* Start of Program 0
HSPD=1000                            ;* Set high speed to 1000 pulses/sec
LSPD=500                             ;* Set low speed to 500 pulses/sec
ACC=500                              ;* Set acceleration to 500 msec
TOC=5000                             ;* Set time-out alarm to 5 seconds
EO=1                                  ;* Enable motor
WHILE 1=1                             ;* Forever loop
    X0                                ;* Move to position 0
    WAITX                             ;* Wait for the move to complete
    X1000                             ;* Move to position 1000
    WAITX                             ;* Wait for the move to complete
ENDWHILE                             ;* Go back to WHILE statement
END                                  ;* End Program 0

PRG 1                                ;* Start of Program 1
WHILE 1=1                             ;* Forever loop
    V1=MSTX&2048                     ;* Get bit time-out counter alarm variable
    IF V1 = 2048                     ;* If time-out counter alarm is on
        SR0=0                        ;* Stop program 0
        ABORTX                       ;* Abort the motor
        DO=0                          ;* Set DO=0
        DELAY=3000                   ;* Delay 3 seconds
        SR0=1                        ;* Turn program 0 back on
        DO=1                          ;* Set DO=1
    ENDIF
ENDWHILE                             ;* Go back to WHILE statement
END                                  ;* End Program 1
```

A: Speed Settings

HSPD value [PPS] ₁	Speed Window [SSPDM]	Min. LSPD value	Min. ACC [ms]	δ	Max ACC setting [ms]
1 - 65K	0,1	1	2	50	((HSPD – LSPD) / δ) × 1000
65K - 130K	2	2	1	100	
130K - 325K	3	5	1	200	
325K - 650 K	4	10	1	800	
650K - 1.3M	5	20	1	1500	
1.3M - 3.2M	6	50	1	3800	
3.2M - 6M	7	100	1	7500	

Table A-1

₁If StepNLoop is enabled, the [HSPD range] values needs to be transposed from PPS (pulse/sec) to EPS (encoder counts/sec) using the following formula:

$$\text{EPS} = \text{PPS} / \text{Step-N-Loop Ratio}$$

A.1. Acceleration/Deceleration Range

The allowable acceleration/deceleration values depend on the **LS** and **HS** settings.

The minimum acceleration/deceleration setting for a given high speed and low speed is shown in Table A-1.

The maximum acceleration/deceleration setting for a given high speed and low speed can be calculated using the formula:

Note: The ACC parameter will be automatically adjusted if the value exceeds the allowable range.

$$\text{Max ACC} = ((\text{HS} - \text{LS}) / \delta) \times 1000 \text{ [ms]}$$

Figure A-1

Examples:

- a) If **HSPD** = 20,000 pps, **LSPD** = 10,000 pps:
 - a. Min acceleration allowable: **1 ms**
 - b. Max acceleration allowable:
 $((20,000 - 10000) / 50) \times 1,000 \text{ ms} = \mathbf{200,000 \text{ ms}}$ (200 sec)

- b) If **HSPD** = 900,000 pps, **LSPD** = 9,000 pps:
 - a. Min acceleration allowable: **1 ms**
 - b. Max acceleration allowable:
 $((900,000 - 9,000) / 1500) \times 1000 \text{ ms} = \mathbf{594,000 \text{ ms}}$ (594 sec)

A.2. Acceleration/Deceleration Range – Positional Move

When dealing with positional moves, the controller automatically calculates the appropriate acceleration and deceleration based on the following rules.

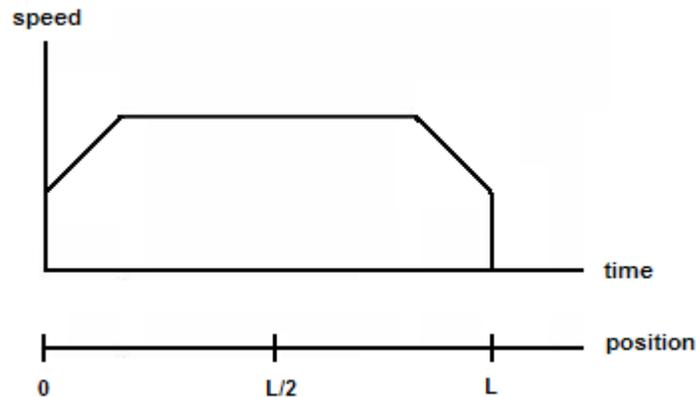


Figure A-2

- 1) ACC vs. DEC 1: If the theoretical position where the controller begins deceleration is less than $L/2$, the acceleration value is used for both ramp up and ramp down. This is regardless of the EDEC setting.
- 2) ACC vs. DEC 2: If the theoretical position where the controller begins constant speed is greater than $L/2$, the acceleration value is used for both ramp up and ramp down. This is regardless of the EDEC setting.
- 3) Triangle Profile: If either (1) or (2) occur, the velocity profile becomes triangle. Maximum speed is reached at $L/2$.



Contact Information

Nippon Pulse America, Inc.

4 Corporate Drive
Radford, VA 24141
540-633-1677

www.nipponpulse.com

The information in this document is believed to be accurate at the time of publication but is subject to change without notice.