PMX-2EX-SA

Advanced 2-Axis
Stepper Motion Controller

Firmware Compatibility:

†V132BLB

†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

PMX-2EX-SA is an advanced 2 axis stepper standalone programmable motion controller.

Communication to the PMX-2EX-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
    - XY linear coordinated motion
- A/B/Z differential encoder inputs [Max frequency of 5 MHz]
    - StepNLoop closed loop control (position verification)
- Opto-isolated I/O
    - 8 x inputs
    - 8 x outputs
    - +Limit/-Limit/Home inputs per axis
- Homing routines:
    - Home input only
    - Limit only
    - Z-index encoder channel only
    - Home input + Z index encoder channel
- 2 x 10-bit analog inputs
    - XYZU joystick control

2. Electrical Specifications

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

Table 2.0

[1]Based on component ratings

3. Dimensions

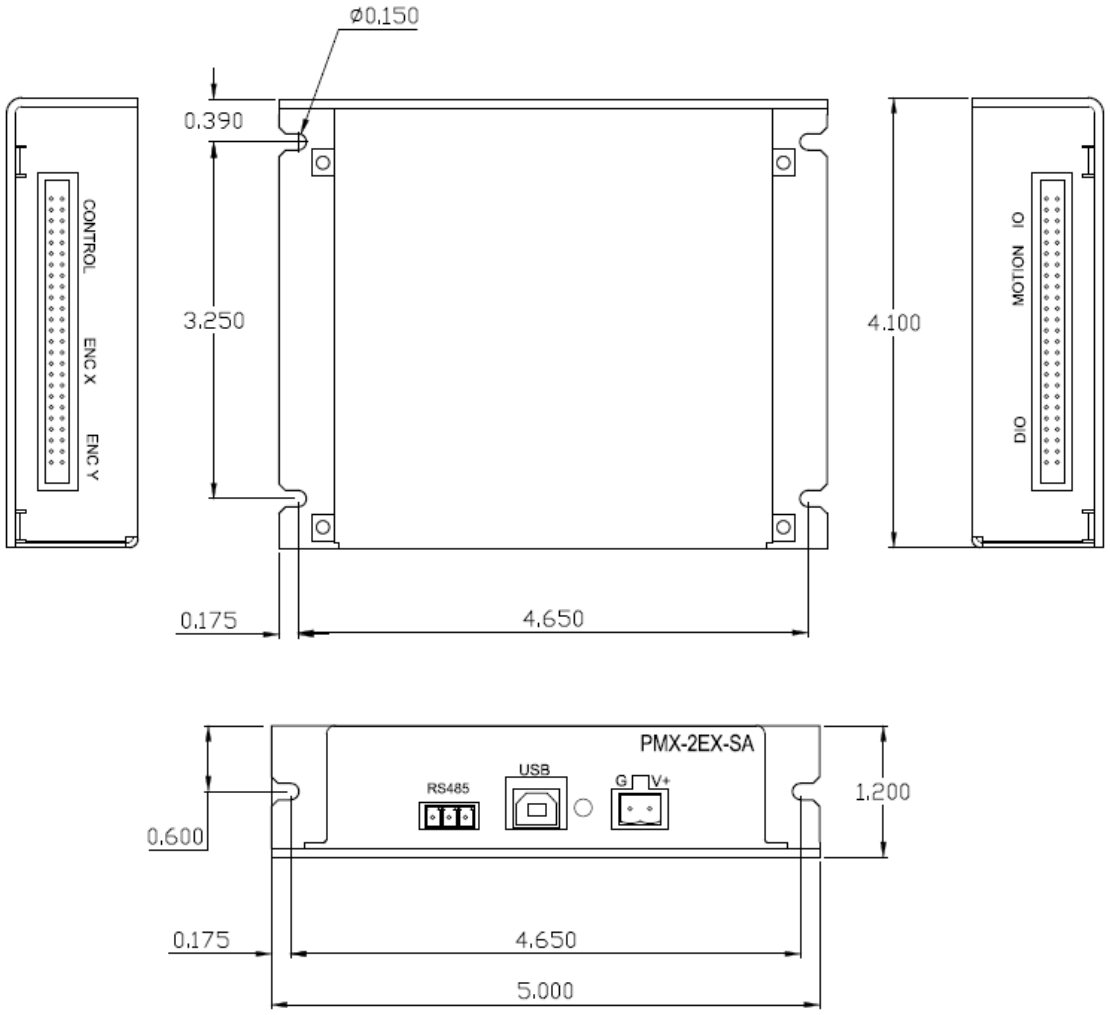

Figure 3.0

4. Connectivity

In order for PMX-2EX-SA to operate, it must be supplied with +12VDC to +24VDC. Power pins as well as communication port pin outs are shown below.

4.1. 2-Pin Power Connector (5.08mm)

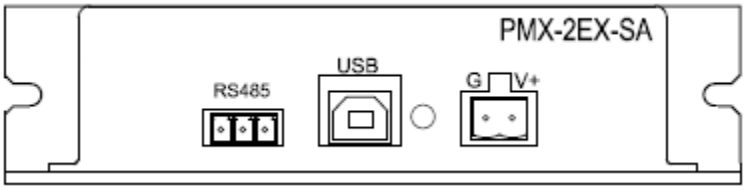

Figure 4.0

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

Table 4.0

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

[1] Other 5.08mm compatible connectors can be used.

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

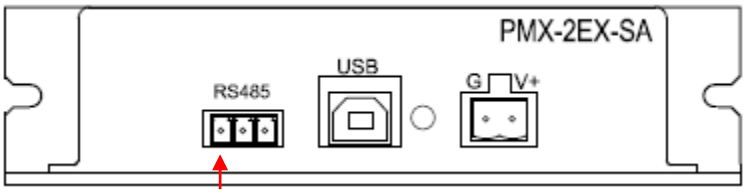

Figure 4.1

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

Table 4.1

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

[1] Other 3.81 compatible connectors can be used.

4.3. 50-Pin Control / Encoder IO

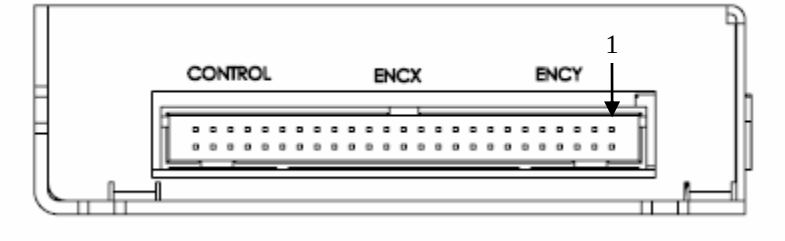

Figure 4.2

Pin #	In/Out	Name	Description
1	NC	NC	No Connection
2	NC	NC	No Connection
3	O	+5V	+5V
4	O	G	Ground
5	I	AY	Y-Axis A Channel Encoder Input
6	I	/AY	Y-Axis /A Channel Encoder Input
7	I	BY	Y-Axis B Channel Encoder Input
8	I	/BY	Y-Axis /B Channel Encoder Input
9	I	ZY	Y-Axis Z Index Encoder Input
10	I	/ZY	Y-Axis /Z Index Encoder Input
11	NC	NC	No Connection
12	NC	NC	No Connection
13	NC	NC	No Connection
14	NC	NC	No Connection
15	NC	NC	No Connection
16	NC	NC	No Connection
17	NC	NC	No Connection
18	NC	NC	No Connection
19	NC	NC	No Connection
20	NC	NC	No Connection
21	O	+5V	+5V
22	O	G	Ground
23	I	AX	X-Axis A Channel Encoder Input
24	I	/AX	X-Axis /A Channel Encoder Input
25	I	BX	X-Axis B Channel Encoder Input
26	I	/BX	X-Axis /B Channel Encoder Input
27	I	ZX	X-Axis Z Index Encoder Input
28	I	/ZX	X-Axis /Z Index Encoder Input
29	NC	NC	No Connection
30	NC	NC	No Connection
31	NC	NC	No Connection

32	NC	NC	No Connection
33	NC	NC	No Connection
34	NC	NC	No Connection
35	NC	NC	No Connection
36	NC	NC	No Connection
37	NC	NC	No Connection
38	NC	NC	No Connection
39	O	+5V	+5V
40	O	G	Ground
41	O	PulX	X-Axis Pulse
42	O	DirX	X-Axis Direction
43	O	EnaX	X-Axis Enable
44	I	AI1	Analog Input 1
45	O	PulY	Y-Axis Pulse
46	O	DirY	Y-Axis Direction
47	O	EnaY	Y-Axis Enable
48	I	AI2	Analog Input 2
49	NC	NC	No Connection
50	NC	NC	No Connection

Table 4.2

Mating Connector Description: 50 pin 0.1" connector
Mating Connector Manufacturer: CW Industries
Mating Connector Manufacturer Part: [1] C3AAG-5018M

[1] Other compatible IDC connectors can be used.

4.4. 50-Pin Motion IO / DIO

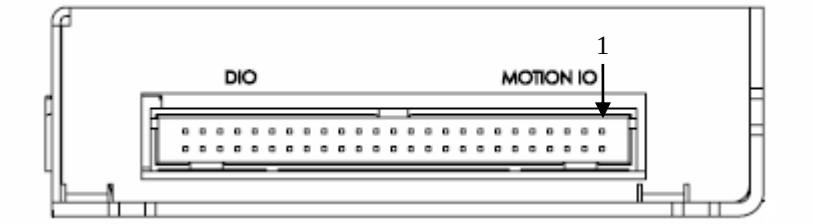

Figure 4.3

Pin #	In/Out	Name	Description
1	NC	NC	No Connection
2	NC	NC	No Connection
3	I	Opto	Opto-Supply Input +12 to +24VDC
4	NC	NC	No Connection
5	I	+LX	+Limit [X Axis]
6	I	-LZ	-Limit [X Axis]
7	I	HX	Home [X Axis]
8	NC	NC	No Connection
9	I	+LY	+Limit [Y Axis]
10	I	-LY	-Limit [Y Axis]
11	I	HY	Home [Y Axis]
12	NC	NC	No Connection
13	NC	NC	No Connection
14	NC	NC	No Connection
15	NC	NC	No Connection
16	NC	NC	No Connection
17	NC	NC	No Connection
18	NC	NC	No Connection
19	NC	NC	No Connection
20	NC	NC	No Connection
21	NC	NC	No Connection
22	NC	NC	No Connection
23	NC	NC	No Connection
24	NC	NC	No Connection
25	NC	NC	No Connection
26	NC	NC	No Connection
27	NC	NC	No Connection
28	NC	NC	No Connection
29	I	Opto-Supply	Opto-Supply Input +12 to +24VDC
30	I	Opto-Ground	Opto-Ground

31	I	DI1	Digital Input 1
32	I	DI2	Digital Input 2
33	I	DI3	Digital Input 3
34	I	DI4	Digital Input 4
35	I	DI5	Digital Input 5
36	I	DI6	Digital Input 6
37	I	DI7	Digital Input 7
38	I	DI8	Digital Input 8
39	O	DO1	Digital Output 1
40	O	DO2	Digital Output 2
41	O	DO3	Digital Output 3
42	O	DO4	Digital Output 4
43	O	DO5	Digital Output 5
44	O	DO6	Digital Output 6
45	O	DO7	Digital Output 7
46	O	DO8	Digital Output 8
47	NC	NC	No Connection
48	NC	NC	No Connection
49	NC	NC	No Connection
50	NC	NC	No Connection

Table 4.3

Mating Connector Description: 50 pin 0.1" connector
Mating Connector Manufacturer: CW Industries
Mating Connector Manufacturer Part: [1] C3AAG-5018M

[1] Other compatible IDC connectors can be used.

4.5. PMX-2EX-SA Interface Circuit

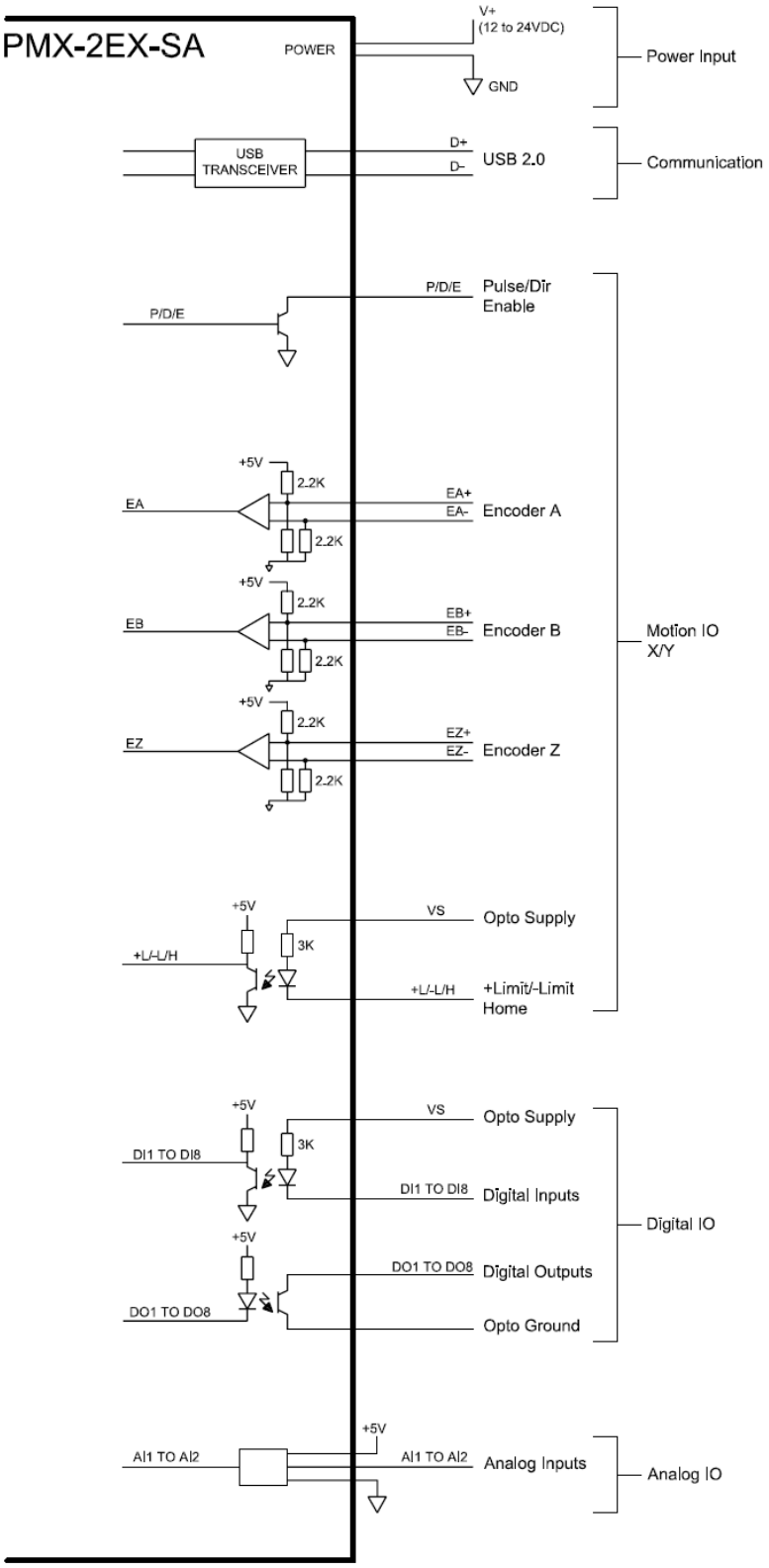

Figure 4.4

4.6. 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 4.11 shows the detailed schematic of these outputs. Each output is capable of sinking up to 40mA of current.

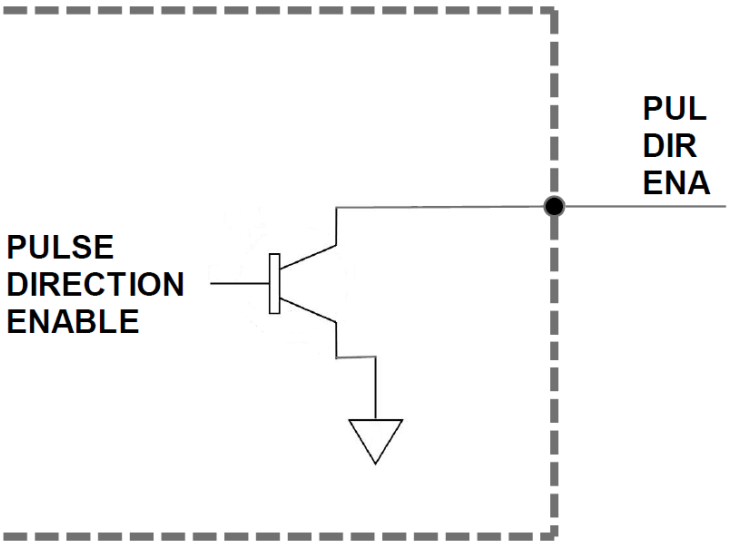

Figure 4.5

Figure 4.12 shows an example wiring diagram between the pulse, direction, and enable outputs on the PMX-2EX-SA and the corresponding input on a typical stepper driver.

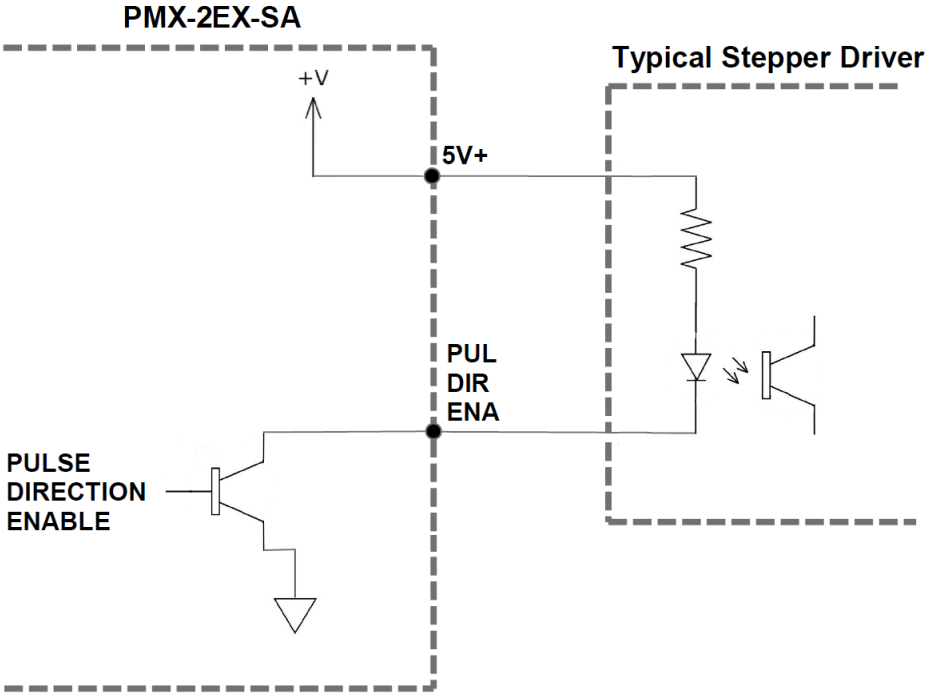

Figure 4.6

4.7. Limit, Home, and Digital Inputs

Figure 4.7 shows the detailed schematic of the opto-isolated limit, home, and general purpose digital inputs. All opto-isolated digital inputs are NPN type.

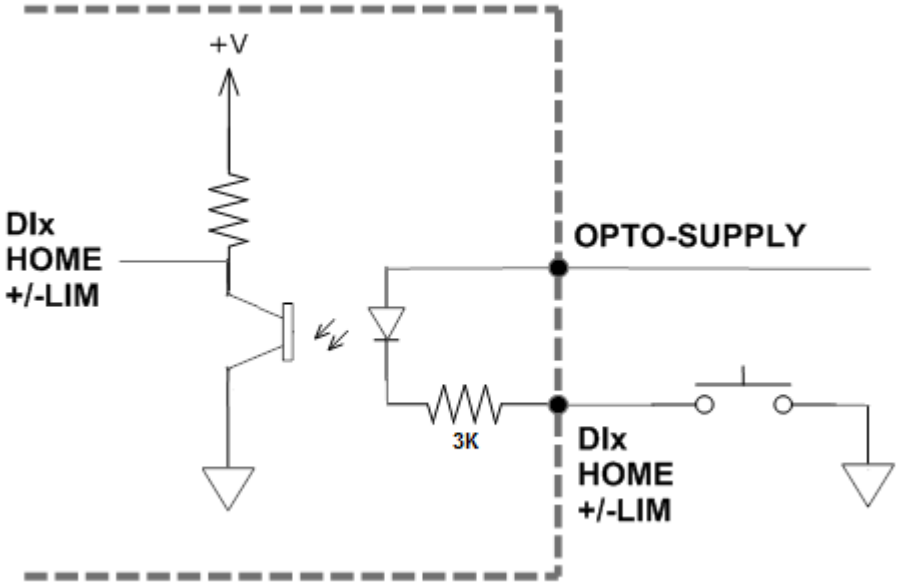

Figure 4.7

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.

4.8. Digital Outputs

Figure 4.8 shows an example wiring to the digital output. All opto-isolated digital outputs will be NPN type.

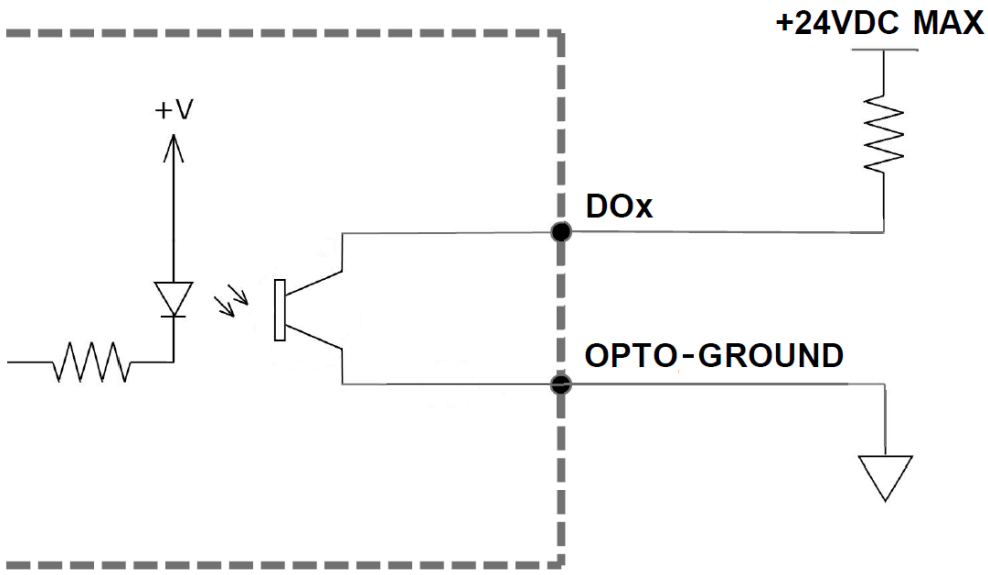

Figure 4.8

The opto-ground must be connected in order for the digital outputs to operate.

When activated, the opto-isolator for the digital output sinks the voltage on the digital output line to the opto-ground. The maximum sink current for digital outputs is 90mA. Take caution to select the appropriate external resistor so that the current does not exceed 90mA. Additionally, the pull up voltage should not exceed +24VDC.

When deactivated, the opto-isolator will break the connection between the digital output and the opto-ground. In this case, the voltage on the digital output signal will be the pull-up voltage.

4.9. 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.

4.10. Analog Inputs

Analog inputs are 0 to 5V range and 10 bit in resolution. Using two analog inputs, joystick control can be achieved.

The maximum source current for the analog inputs is 10mA.

5. Communication Interface

5.1. USB Communication

PMX-2EX-SA USB communication is USB 2.0 compliant.

In order to communicate with PMX-2EX-SA via USB, the proper software driver must be first installed. Before connecting the PMX-2EX-SA 2-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-2EX-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 5.0.

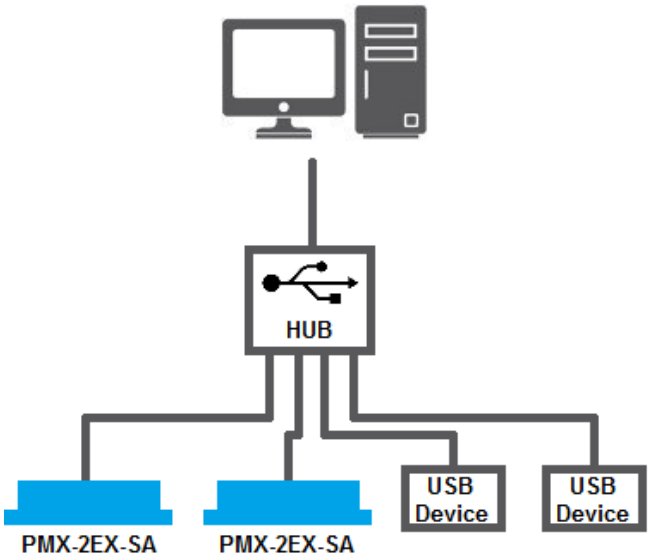

Figure 5.0

5.1.2. USB Communication API

Communication between the PC and PMX-2EX-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-2EX-SA.

Typical communication transaction time between PC and PMX-2EX-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 5.1.

Figure 5.1

5.2. Serial Communication

The PMX-2EX-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-2EX-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 5.2. Several techniques can be used to increase the robustness of an RS-485 network. Please see section 5.2.4 for details.

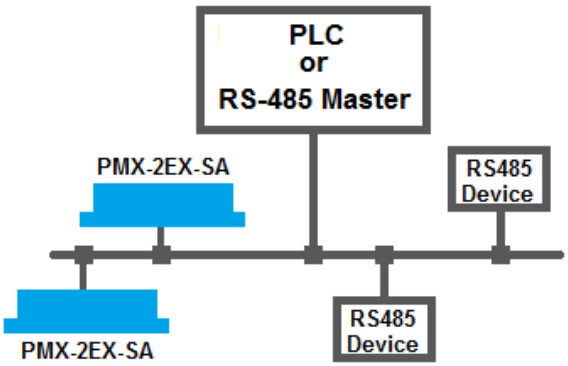

Figure 5.2

5.2.2. Communication Port Settings

The PMX-2EX-SA has the communication port settings shown in table 5.0.

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

Table 5.0

PMX-2EX-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 5.1

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-2EX-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-2EX-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: @00POLX[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 5.2 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-2EX-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-2EX-SA, FIRST STORE THE DESIRED NUMBER USING THE DN **COMMAND. NOTE THAT THIS VALUE MUST BE WITHIN THE RANGE [2EX00,2EX99].**

For example, to change the device number, the command **DN=2EX02** can be sent. The device name can also be changed through the *Setup* window of the standard PMX-2EX-SA software. See section 7 for details.

BY DEFAULT, ALL PMX-2EX-SA START WITH DEVICE NUMBER 2EX00**.**

To save a modified device number to the flash memory of the PMX-2EX-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-2EX-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. See section 7 for further details.

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-2EX-SA.

6.1. Motion Profile

By default, the PMX-2EX-SA uses trapezoidal velocity profile as shown in figure 6.0.

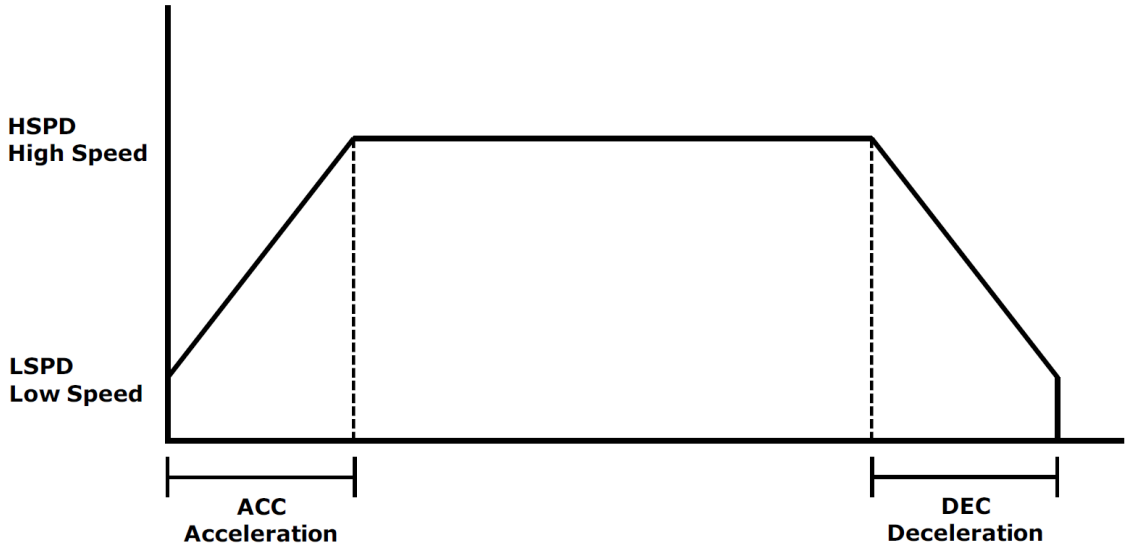

Figure 6.0

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 6.1 for details.

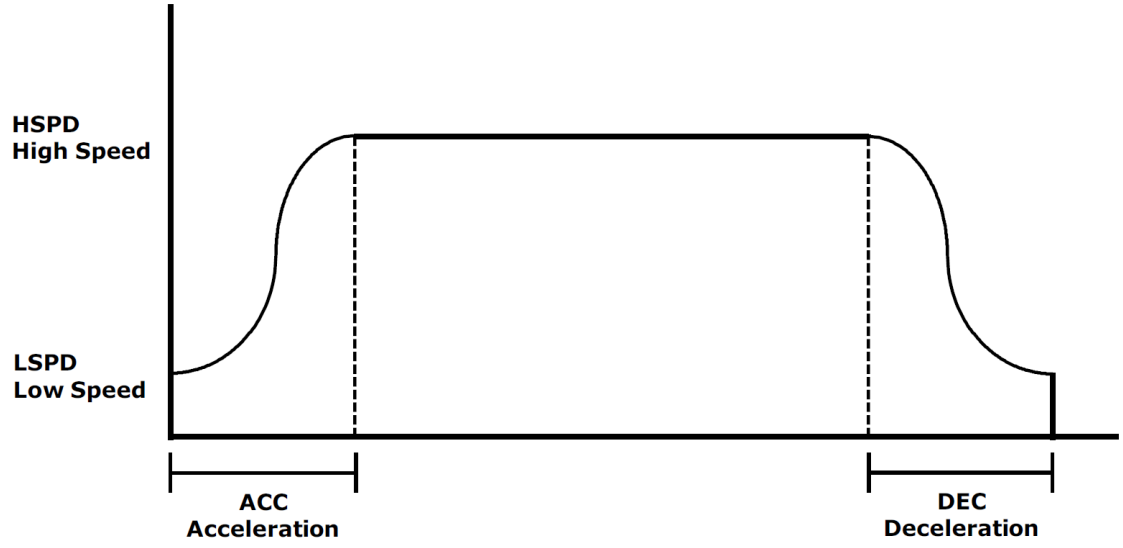

Figure 6.1

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 commands **HSPD[axis]** and **LSPD[axis]** to set/get the individual high speed and low speed settings. To set/get the global high speed and low speed values use the 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.0 and Figure A.0 in **Appendix A** for details.

By default, moves made by a single axis use global speed settings defined by **HSPD**, **LSPD**, **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.

HSPD=10000
HSPDX=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	**HSPD**	**LSPD**	**ACC**	**DEC**	**HSPDn**	**LSPDn**	**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 6.0

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

Table 6.0

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-2EX-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. 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-2EX-SA also has a 32 bit signed encoder position counter for each axis. 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	**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 0-3.

Enable output values can also be referenced one bit at a time by the **EO[1-2]** 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

Table 6.1

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[1-2]**	**EOBOOT**
Standalone	**EO**	**EO[1-2]**	**-**

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. In ASCII mode, the **J[+/-]** command can be used to jog both motors synchronously. 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 8.1 for details on error responses.

ASCII	**J[axis][+/-]**	**J[+/-]**
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-2EX-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-2EX-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 the **X[target]** and **Y[target** 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

For linear interpolation axis control in ASCII mode use the **I[X Target]:[Y Target]** to perform coordinated movement to the specified target position. The command **X[target]Y[target]** can be used for Standalone mode.

For example, the command **[I1000:-1000]** in ASCII mode or **X1000Y-1000** in Standalone mode will move the X-axis to position 1000 and the Y-axis to position -1000 using linear interpolation.

ASCII	**X[target]**	**Y[target]**	**I[Xtarget]:[Ytarget]**
Standalone	**X[target]**	**Y[target]**	**X[target]Y[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. Homing

Home search routines involve moving the motor and using the home, limit, or Z-index inputs to determine the zero reference position. The PMX-2EX-SA has five different homing routines.

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 8.1 for details on error responses.

6.10.1. Home Input Only (High Speed Only)

Use the **H[axis][+/-]** command in ASCII mode or the **HOME[axis][+/-]** command in standalone mode to issue a homing command that uses the home input only. Figure 6.2 shows the homing routine.

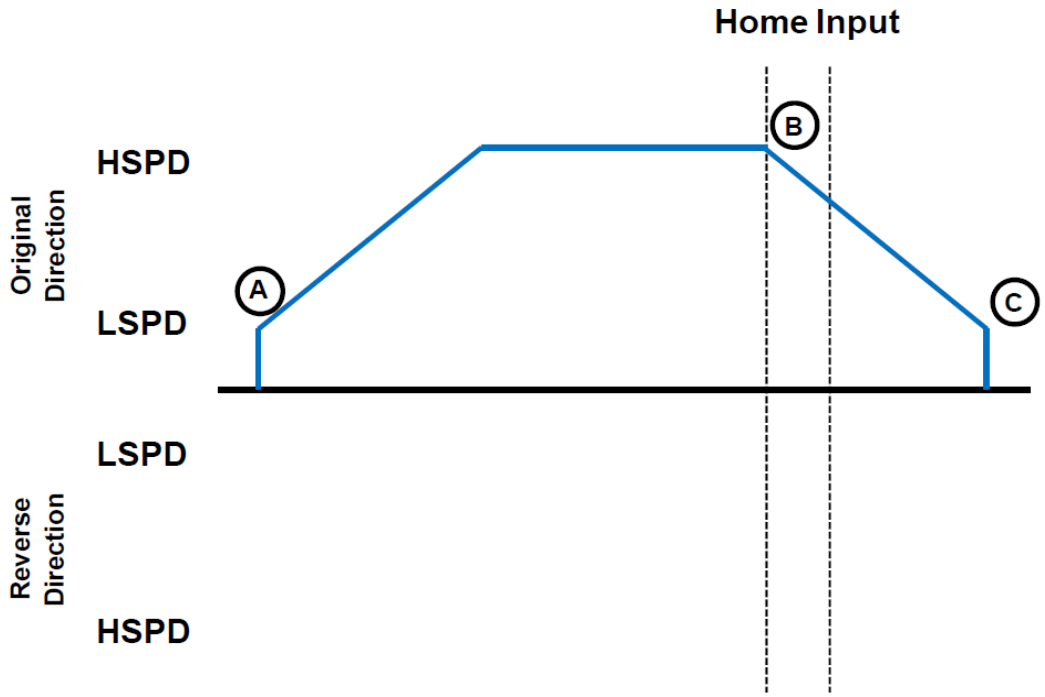

Figure 6.2

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][+/-]**
Standalone	**HOME[axis][+/-]**

6.10.2. Limit Only

Use the **L[axis][+/-]** command for ASCII mode or the **LHOME[axis]+/-** command for standalone mode. Figure 6.3 shows the homing routine.

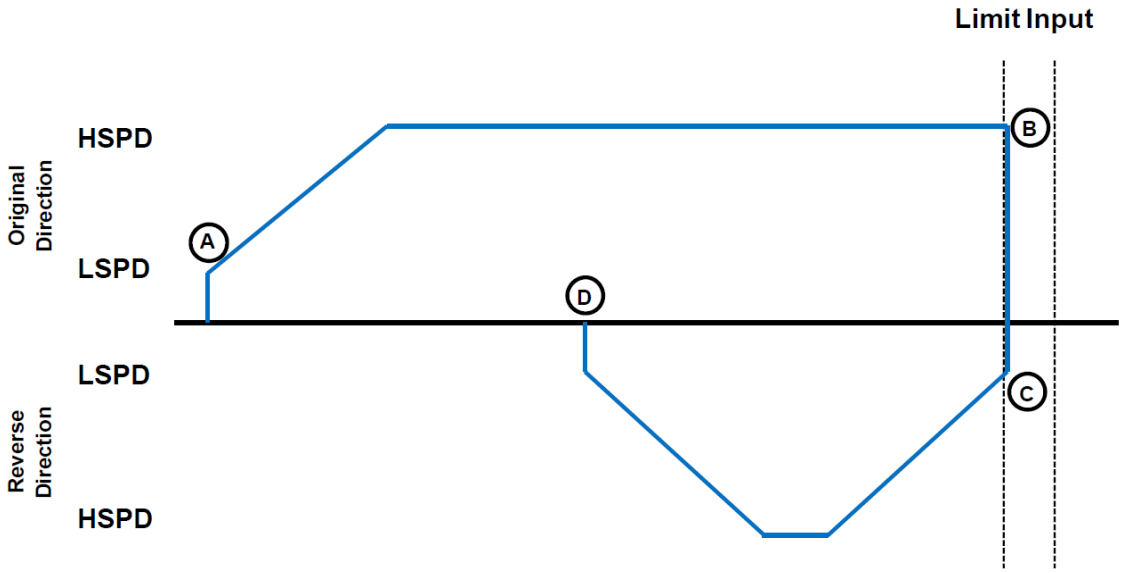

Figure 6.3

A. Issuing a limit home command starts the motor from low speed and accelerates to high speed.
B. The corresponding limit is triggered and the motor stops immediately.
C. The motor reverses direction by the amount defined by the limit correction amount (**LCA**) at high speed.
D. The zero position is reached.

ASCII	**L[axis][+/-]**	**LCA[axis]**	**LCA**
Standalone	**LHOME[axis][+/-]**	**-**	**-**

6.10.3. Home Input and Z-index

Use the **ZH[axis][+/-]** 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 6.4 shows the homing routine.

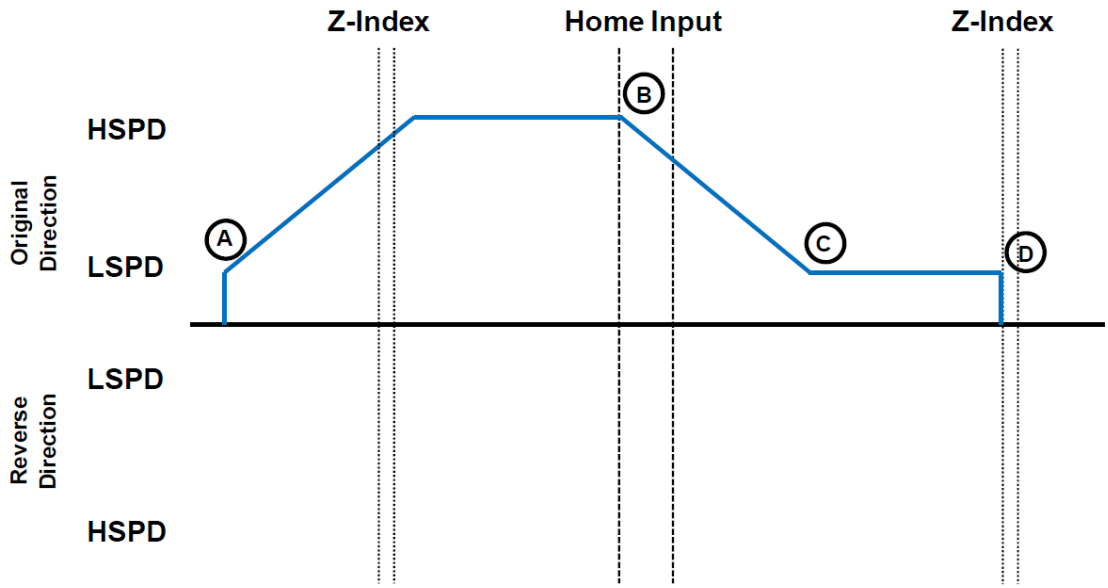

Figure 6.4

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	**ZH[axis][+/-]**
Standalone	**ZHOME[axis][+/-]**

6.10.4. Z-index Only

Use the **Z[axis][+/-]** 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 6.5 shows the homing routine.

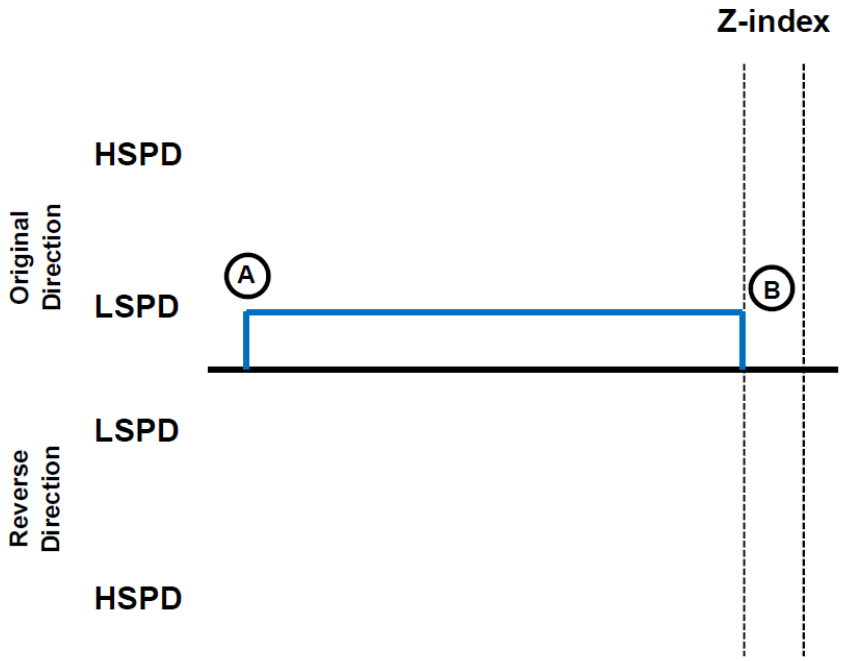

Figure 6.5

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	**Z[axis][+/-]**
Standalone	**ZOME[axis][+/-]**

6.10.5. Home Input Only (High Speed and Low Speed)

Use the **HL[axis][+/-]** command for ASCII mode and the **HLHOME[axis]+/-** for standalone mode. Figure 6.6 shows the homing routine.

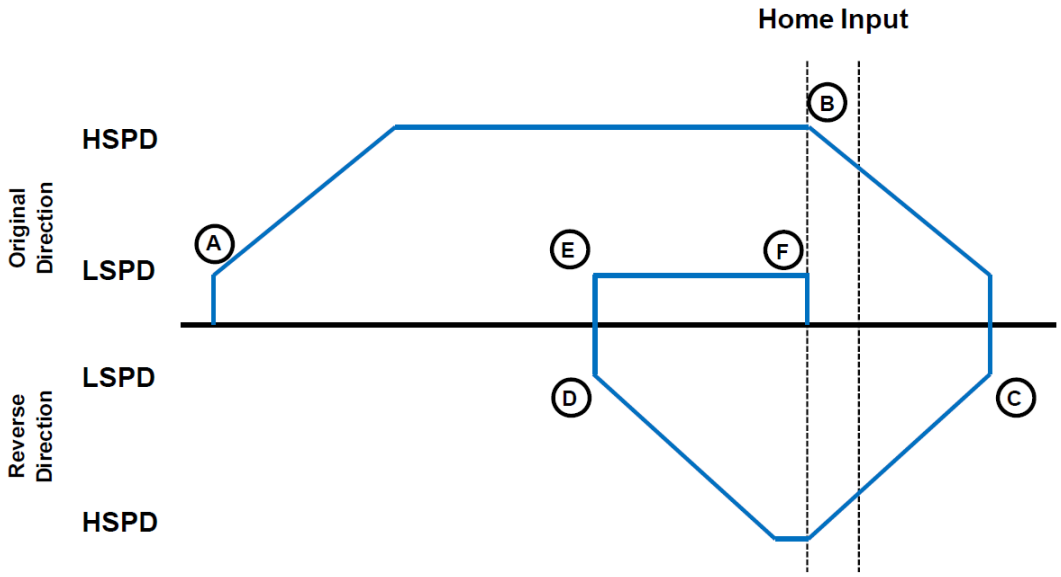

Figure 6.6

A. Starts the motor from low speed and accelerates to high speed.
B. As soon as the home input is triggered, the position counter is reset to zero and the motor decelerates to low speed.
C. Once low speed is reached, the motor reverses direction to search for the home switch.
D. Once the home switch is reached, it will continue past the home switch by the amount defined by the home correction amount (**HCA**) at high speed.
E. The motor is now past the home input by the amount defined by the home correction amount (**HCA**). The motor now moves back towards the home switch at low speed.
F. The home input is triggered again, the position counter is reset to zero and the motor stops immediately

ASCII	**HL[axis][+/-]**	**LCA[axis]**	**LCA**
Standalone	**HLOME[axis][+/-]**	**-**	**-**

6.11. Limit 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.

ONCE THE LIMIT 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 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]**	**IERR**
Standalone	**ECLEAR[axis]**	**-**

6.12. Motor Status

Motor status can be read anytime using **MSTX/MSTY** 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	Not used
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	Z Index Channel status
10	Joystick Control On status
11	TOC time-out status

Table 6.2

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

6.13. Digital Inputs/Outputs

PMX-2EX-SA module comes with 8 digital inputs and 8 digital outputs.

6.13.1. 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 6.3 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 6.3

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 6.16 for details.

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

6.13.2. 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 6.4 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 6.4

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 6.16 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]**	**-**

6.14. Analog Inputs

The PMX-2EX-SA has 2 x 10-bit analog inputs available. The **AI[1-2]** command can be used to read the current analog input value. The return value is in milli-volts and can range from 0 to 5000 mV.

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

6.15. Joystick Control

Joystick control is available on PMX-2EX-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

Table 6.5

To enable or disable joystick control for an axis, use the ASCII command **JO** or the standalone command **JOYENA=[value]**. The joystick enable parameter is a 2 bit value. For example, digital output value of 3 (11 in binary or 0x3 in hex) means joystick feature is enabled on all axes. If joystick control is enabled, StepNLoop is automatically disabled.

The maximum joystick speed for the X, Y, Z, and U axis is set using the ASCII commands **JV1** and **JV2**, 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 **JV3** and **JV4**, 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.

The ASCII commands **JV5** and **JV6**, or the standalone command **JOYTOL[axis]**, can be used to define the zero tolerance zone around the analog input value of 2500mV. 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 2500mV point to define to zero tolerance zone. See figure 6.7 for details.

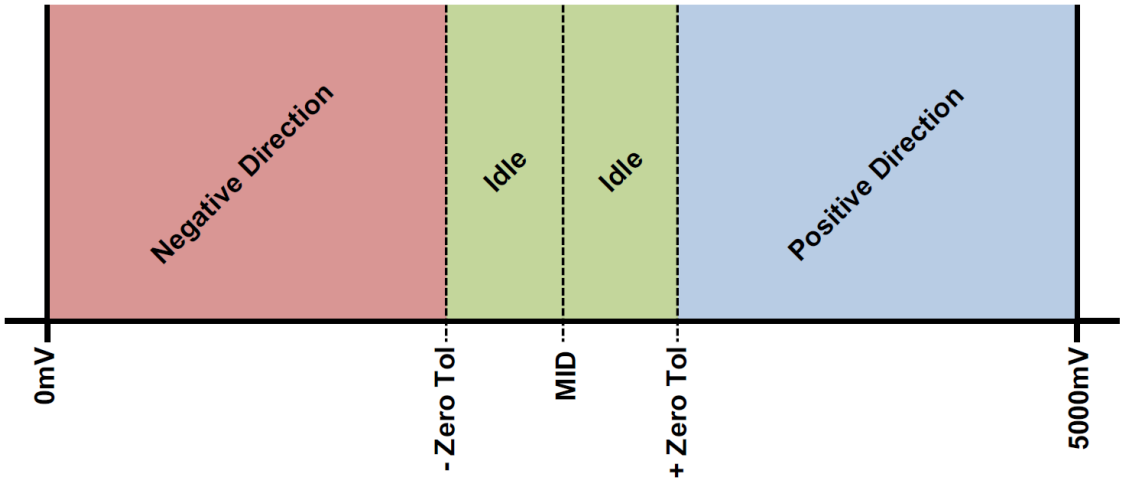

Figure 6.7

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 6.8 represents the relationship between the joystick speed and inner and outer limits.

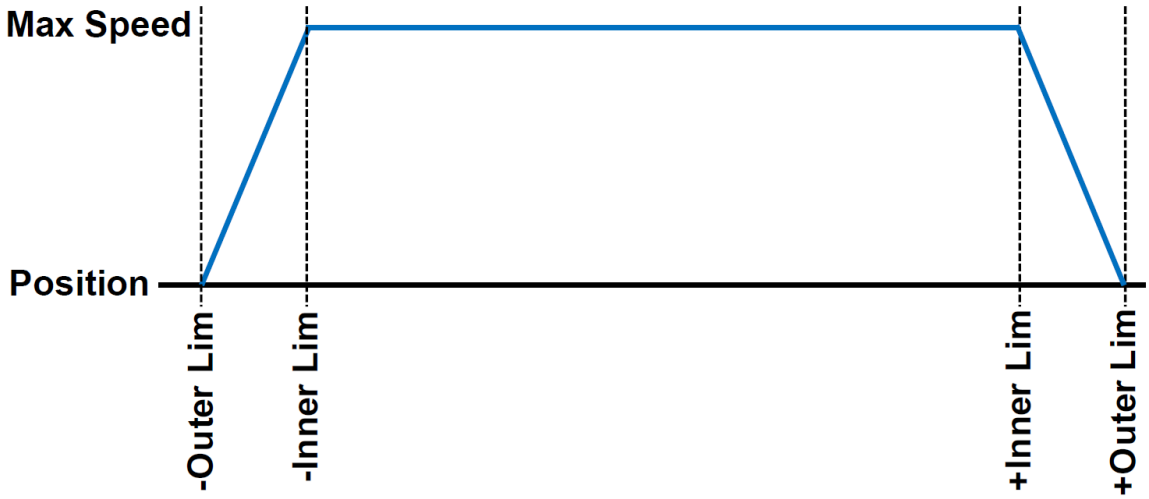

Figure 6.8

See table 6.6 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	X-axis maximum speed change
JV4	Y-axis maximum speed change
JV5	X-axis zero tolerance range for analog input
JV6	Y-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

Table 6.6

ASCII	**JO**	**JV[1-2]**	**JV[3-4]**	**JV[5-6]**
Standalone	**JOYENA=[n]**	**JOYHS[axis]=[n]**	**JOYDEL[axis]=[n]**	**JOYTOL[axis]=[n]**

ASCII	**JL[1]**	**JL[2**	**JL[3]**	**JL[4]**
Standalone	**JOYNOX=[n]**	**JOYNIX=[n]**	**JOYPIX=[n]**	**JOYPOX=[n]**

ASCII	**JL[5]**	**JL[6]**	**JL[7]**	**JL[8]**
Standalone	**JOYNOY=[n]**	**JOYNIY=[n]**	**JOYPIY=[n]**	**JOYPOX=[n]**

6.16. Polarity

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

Bit	Description	
0	Pulse	
1	Direction	
2	Not Used	
3	Not Used	
4	Not Used	
5	Home	
6	+/- Limit	
7	Z-Index	
8,9	Encoder decoding	
	00	1X
	01	2X
	10	4X
10	Digital Input	
11	Digital Output	
12	Enable Output	
13	Jump to Line 0 on error[1]	

Table 6.7

[1]Used for error handling within standalone operation. If this bit is on, the line that is executed after SUB31 is called will be line 0. Otherwise, it will be the line that caused the error.

ASCII	**POL[axis]**
Standalone	**-**

6.17. StepNLoop Closed Loop Control

PMX-2EX-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 6.8 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, micro-stepping, 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 6.8

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 6.9 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 6.9

See Table 6.10 for SNL behavior within different scenarios.

Condition	SNL behavior (motor is moving)	SNL behavior (motor is idle)
δ <= SLT	Continue to monitor the **DX[axis]**	In Position. No correction is performed.
δ > SLT AND δ < SLE	Continue to monitor the **DX[axis]**	Out of Position. A correction is performed.
δ > SLT AND δ > 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 6.10

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[sxis]**	**SLR[axis]**	**SLT[axis]**	**SLE[axis]**
Standalone	**-**	**SL[axis]**	**SLS[axis]**	**-**	**-**	**-**

6.18. Communication Time-out Watchdog

PMX-2EX-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 the section 9 for an example.

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

ASCII	**TOC**
Standalone	**TOC**

6.19. Standalone Program Specification

Standalone programming allows the controller to execute a user defined program that is stored in the internal memory of the PMX-2EX-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-2EX-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-2EX-SA. Each line of written standalone code creates 1-4 assembly lines of code after compilation

The PMX-2EX-SA can store and operate up to two separate standalone programs simultaneously.

6.19.1. Standalone Program Specification

Memory size:1,275 assembly lines.
Note: Each line of pre-compiled code equates to 1-4 lines of assembly lines.

6.19.2. Standalone Control

The PMX-2EX-SA supports the simultaneous execution of two standalone programs. All programs can be controlled by the **SR[0-1]** command, where Program 0 uses command **SR0** and Program 1 uses command **SR1**. For examples of multi-threading, please refer to section 9. The following assignments can be used for the **SR[0-1]** command.

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

Table 6.11

6.19.3. Standalone Status

The **SASTAT[0-1]** command can be used to determine the current status of the specified standalone program. Table 6.12 details the return values of this command.

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

Table 6.12

The **SPC[0-1]** 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[0-1]** 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 [0-1273].

6.19.4. Standalone Subroutines

The PMX-2EX-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.

6.19.5. 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.

6.19.6. Standalone Variables

The PMX-2EX-SA has 64 32-bit signed standalone variables available for general purpose use. They can be used to perform basic calculations and support integer operations. The **V[0-63]** 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 6.13

Variables V32 through V63 can be stored to flash memory using the **STORE** command. Variables V0-V31 will be initialized to zero on power up.

6.19.7. 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 6.14 for the bit assignment of the **SLOAD** setting.

Bit	Description
0	Standalone Program 0
1	Standalone Program 1

Table 6.14

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

ASCII	**SR[0-1]**	**SASTAT[0-1]**	**SPC[0-1]**	**GS[0-31]**	**V[0-63]**	**SLOAD**
Standalone	**SR[0-1]**	**-**	**-**	**GOSUB[0-31]**	**V[0-63]**	**-**

6.20. Storing to Flash

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

ASCII Command	Description
DOBOOT	DO configuration at boot-up
DB	Serial communication baud rate
DN	Device name
EDEC	Unique deceleration enable
EOBOOT	EO configuration at boot-up
HCA, HCA[axis]	Home correction amount
IERR	Ignore limit error enable
JO, JF, JV[1-6], JL[1-8]	Joystick settings
LCA, LCA[axis]	Limit correction amount
POL[axis]	Polarity settings
RZ	Return to zero parameter
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
V32-V63	Note that on boot-up, V0-V31 are reset to value 0

Table 6.15

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

7. Software Overview

The PMX-2EX-SA has a Windows compatible software that allows for USB or RS485 communication. Standalone programming, along with all other available features of the PMX-2EX-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-2EX-SA is connected to one of the available ports on the PC. To communicate over RS485, make sure that the PMX-2EX-SA is connected to the COM port.

Startup the PMX-2EX-SA GUI program and you will see the following screen in figure 7.0. This will allow the user to search for all the motors that are currently connected to the USB network or RS485 bus.

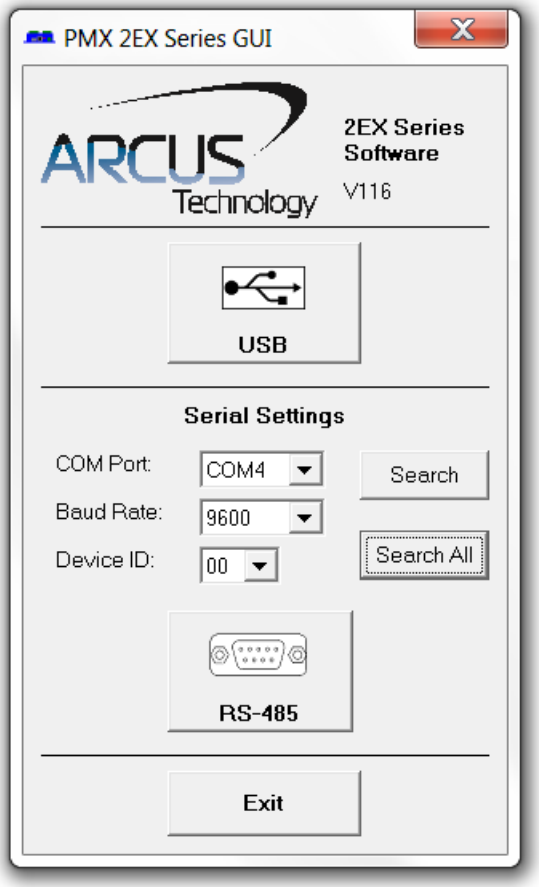

Figure 7.0

USB communication can be established by clicking the USB button. All PMX-2EX-SA connected to the PC will automatically be detected and displayed.

For RS-485 communication, the first PMX-2EX-SA connected to the PC can be found using the Search button. If there are multiple PMX-2EX-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-2EX-SA. Allowable power is range is from 12VDC to 24VDC.
- Check communication wiring. The 485+ from PMX-2EX-SA is connected to 485+ of the master and 485- from PMX-2EX-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.

7.1. Main Control Screen

The Main Control Screen provides accessibility to all the available functions on the PMX-2EX-SA. All features can be tested and verified.

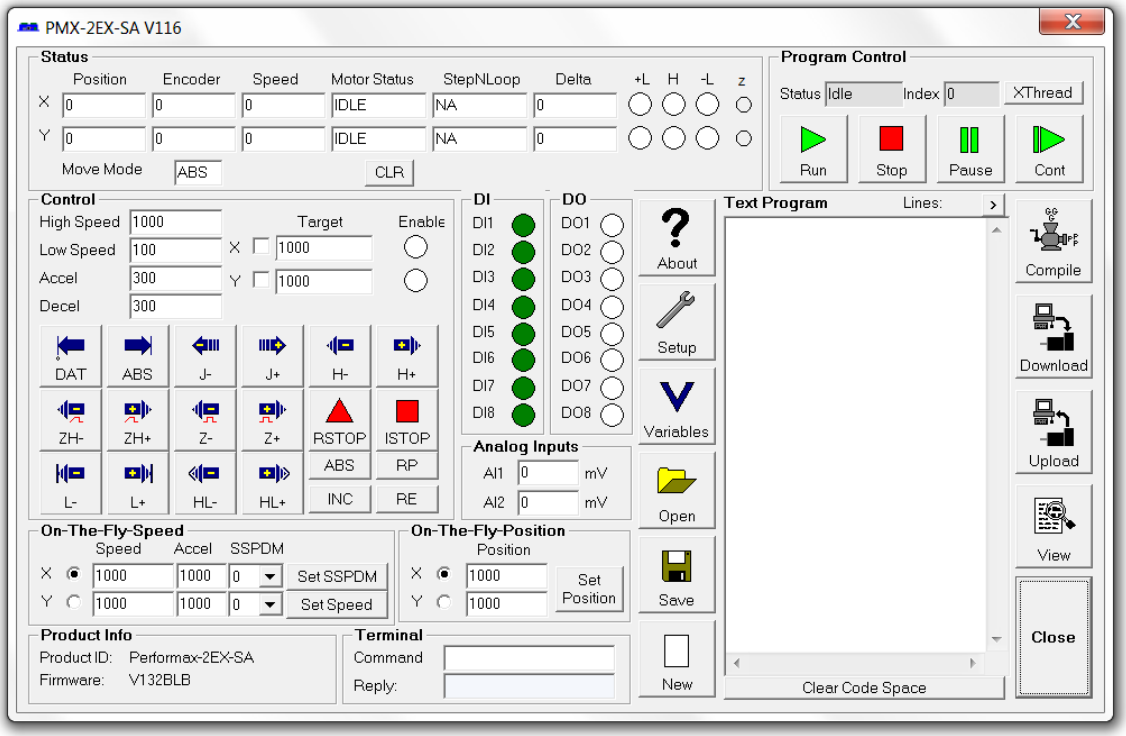

Figure 7.2

7.1.1. Status

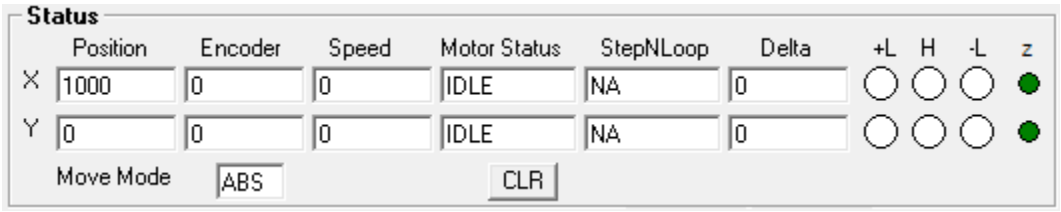

Figure 7.3

1. **Position** (X,Y) - displays the current pulse position counter. If StepNLoop is enabled, this shows the real-time target position.
2. **Encoder** (X,Y) - displays the current encoder position counter.
3. **Speed** (X,Y) - 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 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) - valid only when StepNLoop is enabled and displays the current StepNLoop status
    - NA – StepLNoop 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) - valid only for StepNLoop. Displays the difference between the target positions and the actual position.
7. **Limit/Home Input Status** (X,Y 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. **CLR** - Clears any limit, alarm, or StepNLoop error

7.1.2. Control

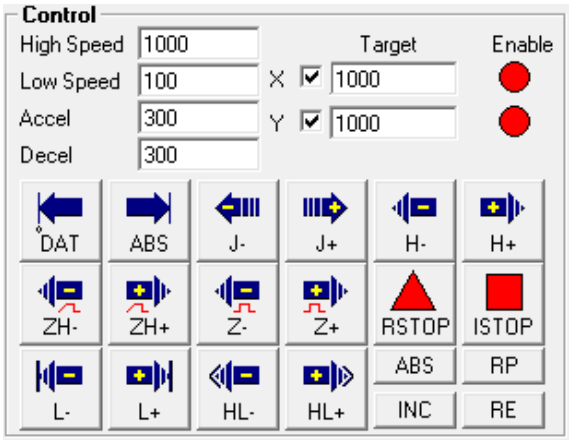

Figure 7.4

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 HSPD[axis], LSPD[axis], and ACC[axis] commands via the command terminal.
2. **X/Y Check Boxes** - A move performed by one of the move buttons will apply to the selected axis.
3. **Target** (X,Y) - positional moves will use this value as the target position.
4. **Enable** (X,Y) – 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. **ISTOP** – the motion is immediately stopped without deceleration.
10. **RSTOP** – the motion is stopped with deceleration.
11. **Z+/Z-** - Home the axis using only the encoder index channel.
12. **H+/H-** - Home the axis at high speed using only the home input.
13. **ZH+/ZH-** - Home the axis using the home input and encoder index channel.
14. **HL+/HL-** -Home the axis at high speed and low speed using only the home sensor.
15. **L+/L-** - Home the axis using only the limit sensor.
16. **JOG+/JOG-** - Move the axis indefinitely in the positive or negative direction.
17. **ABS** - Perform absolute move. If more than one axis is selected, an interpolated move will result.
18. **DAT** - Return to 0 position. If more than one axis is selected, an interpolated move will result.

7.1.3. On-The-Fly-Speed Control

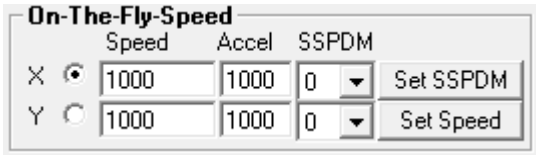

Figure 7.7

1. **Select X/Y** axis.
2. **Speed** - configure the destination speed of the axis.
3. **Accel** - 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

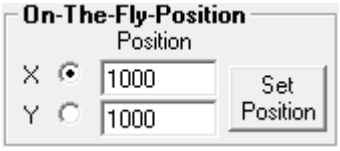

Figure 7.8

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

7.1.5. Digital Input/Output

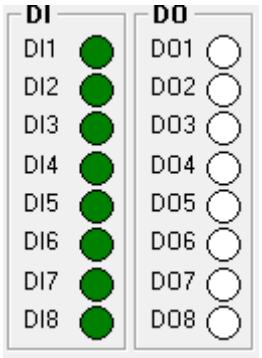

Figure 7.10

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.

7.1.6. Analog Inputs

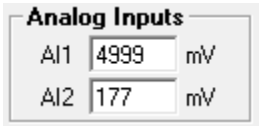

Figure 7.11

1. **Analog Inputs** - displays the analog input status of AI1-AI2. Units are in mV.

7.1.7. Program File Control

Figure 7.12

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

7.1.8. Standalone Program Editor

Figure 7.13

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

7.1.9. Standalone Program Control

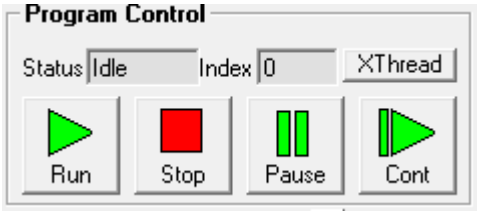

Figure 7.14

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 both 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.

7.1.10. Standalone Program Compile/Download/Upload

Figure 7.15

1. **Compile** – Compile the standalone program
2. **Download** – Download the compiled program
3. **Upload** – Upload the standalone program from the controller
4. **View** - View compiled code

7.1.11. Setup

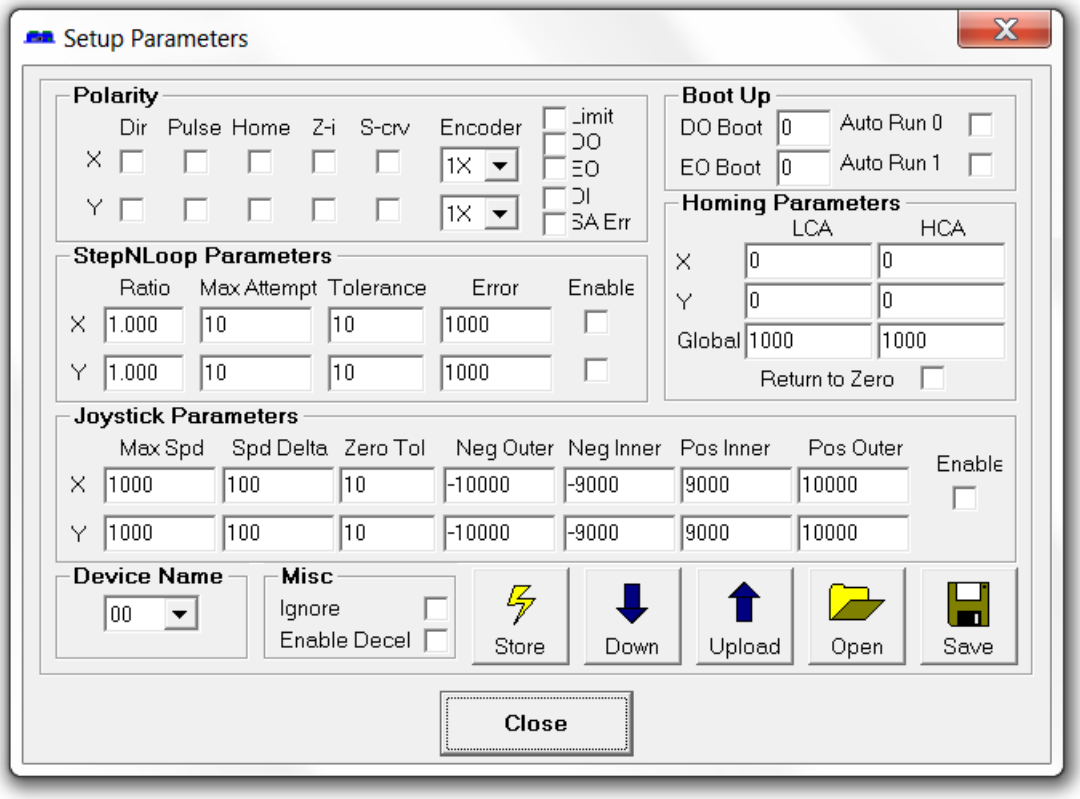

Figure 7.16

1. **Polarity Settings**
    a. **Dir/Pulse/Home/Z-i**(X,Y) - set the home, pulse direction, and Z-index polarity.
    b. **S-crv** (X,Y) - set s-curve enable/disable for each axes.
    c. Set the encoder multiplier to 1X/2X/4X for X/Y axis
    d. **Limit** - Set the limit input polarity
    e. **DO** - Set the digital output polarity
    f. **EO** - Set the enable output polarity
    g. **DI** - Set the digital input polarity
    h. **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: [2EX00-2EX99]
    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. **EDEC** - enable/disable unique deceleration.
7. **Open/Save** parameters to file.
8. **Upload/Download** parameters to and from flash memory.

7.1.12. Terminal

Figure 7.17

1. **Response Box –** Displays sent command as well as corresponding response
2. **Command line –** ASCII command line

7.1.13. Variable Status

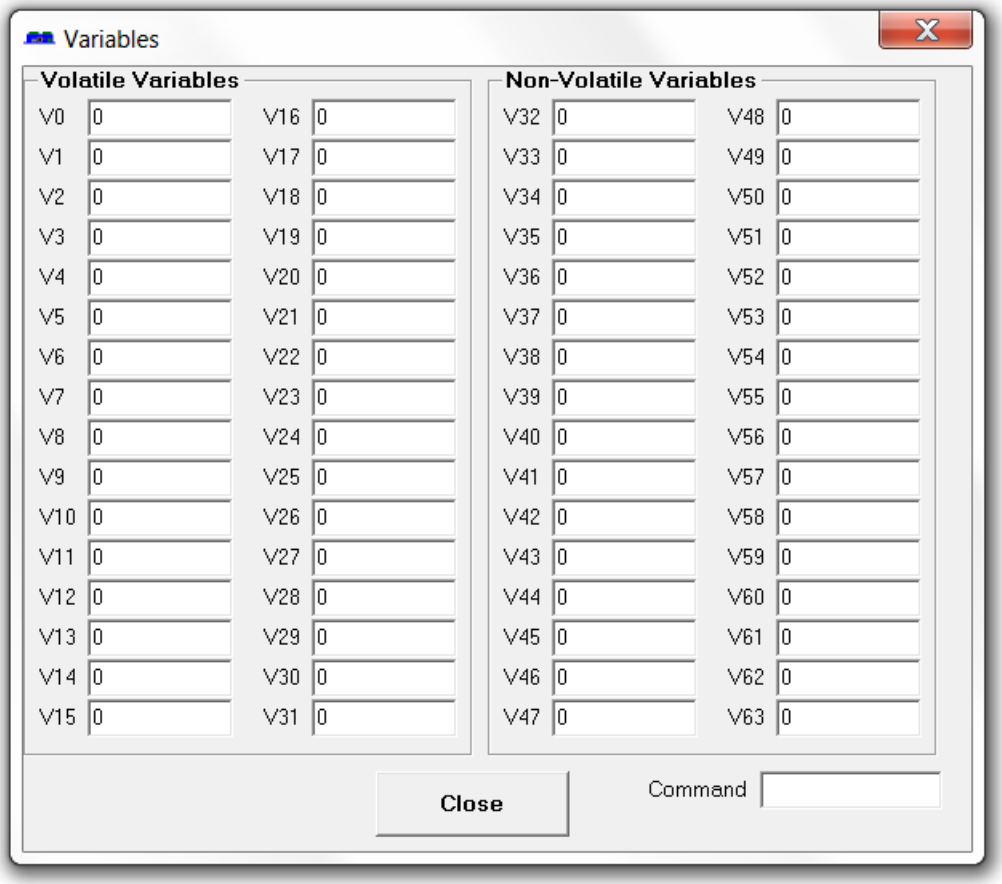

Figure 7.19

1. **Volatile Variables** – status of volatile variable V0-V31
2. **Non-volatile Variables** – status of non-volatile variable V32-V63
3. **Command line** – set variables using V[0-63]=[value] syntax

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-2EX-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 8.0 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[1-2]	Return the analog input status in milli-volts.	[0-5000]
CLR[axis]	Clear the axis limit, alarm, or stepnloop error status bit.	OK
DB	Return the current baud rate or the controller.	Refer to Table 5.1
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 6.3
DI[1-8]	Return the bit status of general purpose digital input.	[0,1]
DO	Return the status of the digital outputs.	Refer to Table 6.4
DO=[value]	Set the digital outputs. Refer to Table 6.4.	OK
DO[1-8]	Return status of individual digital output.	[0,1]
DO[1-8]=[value]	Set the individual digital output. Refer to Table 6.4.	OK
DOBOOT	Return the DO configuration at boot-up.	[0-255]
DOBOOT=[value]	Set the DO configuration at boot-up.	OK
DN	Return the device name.	[2EX00-2EX99]
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 6.1
EO=[value]	Set the enable outputs.	OK
EO[1-2]	Return the individual enable output status.	[0,1]

EO[1-2]=[value]	Set the individual enable output.	OK
EOBOOT	Return the EO configuration at boot-up.	Refer to Table 6.1
EOBOOT=[value]	Set the EI configuration at boot-up.	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
H[axis][+/-]	Homes the motor in plus [+] or minus [-] direction. Refer to section 6.7.1.	OK
HCA	Return the home correction amount.	32-bit number
HCA=[value]	Set the home correction amount	OK
HCA[axis]	Return the home correction amount for the specified axis.	32-bit number
HCA[axis]=[value]	Set the home correction amount for the specified axis.	OK
HL[axis][+/-]	Homes the motor at high and low speed in the plus [+] or minus [-] direction. Refer to section 6.7.5.	OK
HSPD	Return the global high speed parameter.	PPS
HSPD=[value]	Set the global high speed parameter.	OK
HSPD[axis]	Return the individual high speed parameter for the specified axis.	PPS
HSPD[axis]=[value]	Set the individual high speed parameter for the specified axis.	OK
I[X axis]: [Y axis]	XY interpolated move. Target move values are separated by ':' character.	OK
ID	Returned the controller ID	Performax-2EX-SA
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
J[axis][+/-]	Jogs the axis in plus [+] or minus [-] direction.	OK
JF	Turn OFF joystick mode	OK
JL[1-8]	Return the joystick soft limit settings.	32-bit number
JL[1-8]=[value]	Set the joystick soft limit settings.	OK
JO	Turn ON joystick mode	OK
JV[1-6]	Return the joystick speed, delta and tolerance settings.	28-bit number
JV[1-6]=[value]	Set the joystick speed, delta and tolerance settings.	OK
L[axis][+/-]	Homes the motor at using the limit inputs in the plus [+] or minus [-] direction. Refer to section 6.7.2.	OK
LCA	Return the limit correction amount.	32-bit number
LCA=[value]	Set the limit correction amount	OK
LCA[axis]	Return the limit correction amount for the specified axis.	32-bit number
LCA[axis]=[value]	Set the limit correction amount for the specified axis.	OK
LSPD	Return the global low speed parameter.	PPS
LSPD=[value]	Set the global low speed parameter.	OK
LSPD[axis]	Return the individual low speed parameter for the specified axis.	PPS
LSPD[axis]=[value]	Set the individual low speed parameter for the specified axis.	OK
MM	Return the move mode that the controller is currently in.	0 - ABS 1 - INC
MST	Returns all motor status, buffer move status, and move mode status for all axis.	Refer to Table 6.2

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
POL[axis]	Return the current polarity setting.	Refer to Table 6.7
POL[axis]=[value]	Set the polarity setting. Refer to Table 6.7.	OK
PS[axis]	Return the current pulse speed of the axis.	28-bit number
RZ	Return the return to zero enable status used during homing operations	[0,1]
RZ=[value]	Set the return to zero enable status used during homing operation.	OK
SA[0-1273]	Return the standalone line.	
SA[0-1273]=[value]	Set the standalone line.	OK
SASTAT	Return the standalone program status. Refer to Table 6.17.	[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 6.9.	[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 6.14.	[0-3]
SLOAD=[value]	Set the standalone program run on boot parameter. Refer to Table 6.14.	OK
SPC[0-1]	Returns the program counter for the specified standalone program.	[0-1273]
SR[0-1]=[value]	Control the standalone program. Refer to Table 6.11.	OK
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.0.	[0-7]
SSPDM[axis]=[value]	Set on-the-fly speed change mode. Refer to Table A.0.	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 6.15.	OK
T[axis][value]	Perform on-the-fly target position change.	OK
T[axis][value]	Perform and on-the-fly position change operation	OK
TOC	Returns the communication time-out parameter in milliseconds.	milliseconds
TOC=[value]	Set the communication time-out parameter in milliseconds.	OK
V[0-63]	Return the standalone variable value.	32-bit number
V[0-63]=[value]	Set the standalone variable value.	OK
VER	Return the controller firmware version	V[#]
X[target X] Y[target Y]	Perform an individual/interpolated move	OK

Z[axis][+/-]	Homes the motor in the plus [+] or minus [-] direction using the Z index encoder channel only. Refer to section 6.7.4.	OK
ZH[axis][+/-]	Homes the motor using the home and Z-index input in the plus [+] or minus [-] direction. Refer to section 6.7.3.	OK

Table 8.0

8.2. Error Codes

If an ASCII command cannot be processed by the PMX-2EX-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-2EX-SA
?ABS/INC is not in operation	T[] command is invalid because a target position move is not in operation
?Clear SNL Error	A move command has been issued while the axis is in StepNLoop error
?ICommandOn	On-the-fly speed change attempted during an interpolated move
?Index out of Range	The index for the command sent to the controller is not valid.
?Invalid Answer	Invalid parameter input
?Low speed out of range	Low speed parameter is out of range
?Moving	A move or position change command is sent while the PMX-2EX-SA is outputting pulses.
?S-curve on	Cannot perform SSPD move because s-curve is enabled
?Speed out of range	SSPD move parameter is out of the range of the SSPDM speed window.
?SSPD Mode not Initialized	An attempt to perform an on-the-fly speed change without setting the SSPDM register has been made.
?Sub not Initialized	Call to a subroutine using the **GS** command is not valid because the specified subroutine has not been defined.

Table 8.1

9. Standalone Language Specification

Important Note: All the commands described in this section are standalone language commands and are not analogous to ASCII commands. Refer to section 9 for details regarding 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[1-2]	R	Get the analog input value.	IF AI2>2500 DO=1 ENDIF V2=AI1
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 6.3 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 6.3 for bitwise assignment.	IF DI1=0 DO=1 ;Turn on DO1 ENDIF V3=DI1
DO	R/W	Set/get digital output status. See Table 6.4 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 6.1	EO=3 ;Enable the X and Y motor
EO[1-2]	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	HLHOMEX+ ;positive X home WAITX ;wait for X home move

		direction. See section 6.13.5 for details.	
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 ;set to increment mode X1000 ;increment by 1000
JOG[axis][+/-]	W	Move the motor indefinitely in the specified direction.	JOGX+ ;jog in + direction
JOYENA=[value]	W	Set the joystick enable setting. See section 6.15 for details.	JOYENA=3 ;enable joystick X,Y
JOYHS[axis]	W	Set the high speed setting for joystick control. See section 6.15 for details.	JOYHSX=2000 ;max speed 2000
JOYDEL[axis]	W	Set the speed change delta for joystick control. See section 6.15 for details.	JOYDELX=100 ;delta 100
JOYTOL[axis]	W	Set the speed change delta for joystick control. See section 6.15 for details.	JOYTOLX=50 ;idle zone +/-50mV
JOYPO[axis]	W	Set the positive outer joystick limit. See section 6.15 for details.	JOYPOY=500000 ;set outer limit
JOYPI[axis]	W	Set the positive inner joystick limit. See section 6.15 for details.	JOYPIY=490000 ;set inner limit
JOYNO[axis]	W	Set the negative outer joystick limit. See section 6.15 for details.	JOYNOY=-500000 ;set outer limit
JOYNI[axis]	W	Set the negative inner joystick limit. See section 6.15 for details.	JOYNIY=-490000 ;set inner limit
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 ;set LSPD to 100 LSPD=V3 ;set LSPD to V3 value
LSPD[axis]	R/W	Set/get the individual low speed setting. Unit is in pulses/second.	LSPDX=100 ;set LSPDX to 100 LSPDY=V1 ;set LSPDX to V1
MST[axis]	R	Get the current motor status of the motor of an axis. See Table 6.2 for motor status assignment.	V1=MSTX IF MSTX=4 DO=1 ENDIF
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 6.14 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 6.16.	SR0=0 ;turn off program 0
SSPDM[axis]	W	Set the SSPD mode. Must be done before move command. See Table A.0 for details.	SSPDMX=1 ;set SSPD mode SSPDMX=V2 JOGX+ ;jog in + direction
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
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 6.8 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 6.9 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	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

		[/] Division [%] Modulus [>>] Bit shift right [<<] Bit shift left [&] Bitwise AND [|] Bitwise OR [~] Bitwise NOT	
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 9.0

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.0

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

EPS = PPS / 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.0.

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.

Max ACC = ((HS – LS) / δ) × 1000 [ms]

Figure A.0

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) x 1,000 ms = **200,000 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) x 1000 ms = **594,000** 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.

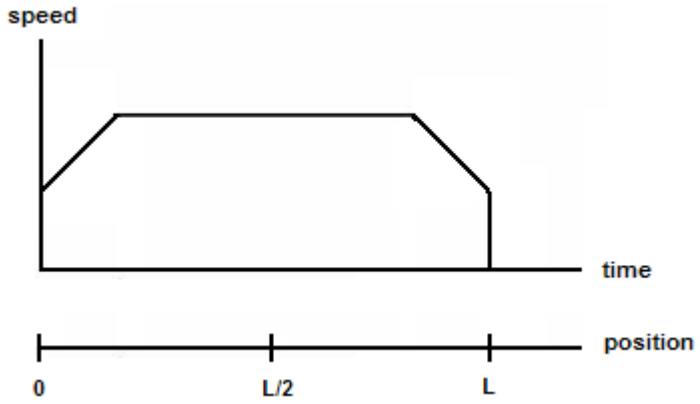

Figure A.1

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