Performax USB 4EX – SA

Advanced 4-Axis
Stepper Motion Controller
Standalone Version

Manual

First edition, February 2007

Revision History:

1.0 – First Revision
2.1 – New firmware features
2.2 – Radius update on interpolation moves, multi-thread limitation
2.3 – Added joystick control
2.4 – Added StepNLoop
2.5 – Added IACC option
2.6 – Home input bit status, analog input pin out

Firmware Compatibility:

V140 - V141

Software Compatibility:

V114

Table of Contents

1. Introduction

PMX-4EX-SA is an advanced 4 axis stepper stand-alone programmable motion controller with USB 2.0 and RS-485 communication.

PMX-4EX-SA has linear coordinated and buffered motion capability for smooth curved motion control applications such as

- 3D CAD/CAM
- Engraving
- Laser cutting

Performax 4EX SA Features

- USB 2.0 communication
- RS-485 ASCII communication with 9600, 19200, 38400, 57600, 115200 Baud rate
- Maximum pulse output rate of 6M PPS per axis
- Trapezoidal or s-curve acceleration
- On-the-fly speed change
- Continuous linear coordinated buffered move for XYZ axes for smooth move control with buffer size of 64
- XYZU linear coordinated motion
- XY circular coordinated motion
- XY arc coordinated motion
- Opto-isolated +Limit, -Limit, Home, and Alarm inputs per axis
- Pulse/Dir/Enable open collector outputs per axis
- Single-ended or differential quadrature encoder inputs per axis
- 8 opto-isolated digital inputs (NPN)
- 8 opto-isolated digital outputs (PNP)
  - Option to use 4 of the digital outputs for synchronous triggering
- 8 10-bit analog inputs
- Joystick control for XYZU axes
- Standalone programmable

PMX-4EX-SA comes with a Windows DLL for easy interface with the program from common programming language such as VB, VC++, and LabVIEW (USB). Sample program in VB is provided.

Contacting Support

For technical support contact: support@arcus-technology.com.

Or, contact your local distributor for technical support.

2. Quick Startup Guide

If you are a first time user and want to have the PMX-4EX-SA unit up and running quickly, follow the recommended steps:

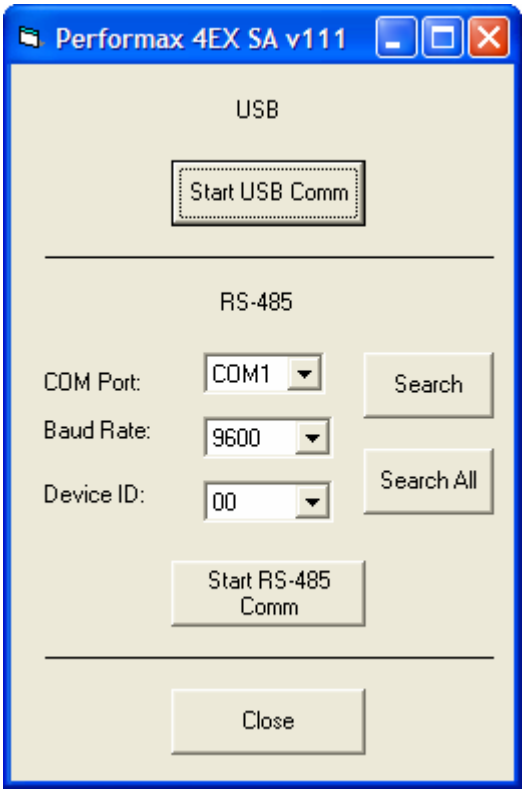
1) Prepare a Windows XP compatible PC with a USB communication port.

2) Run Performax USB Driver Setup program. Both setup program and the manual for Performax USB Driver setup can be downloaded from the web site
   www.arcus-technology.com/support

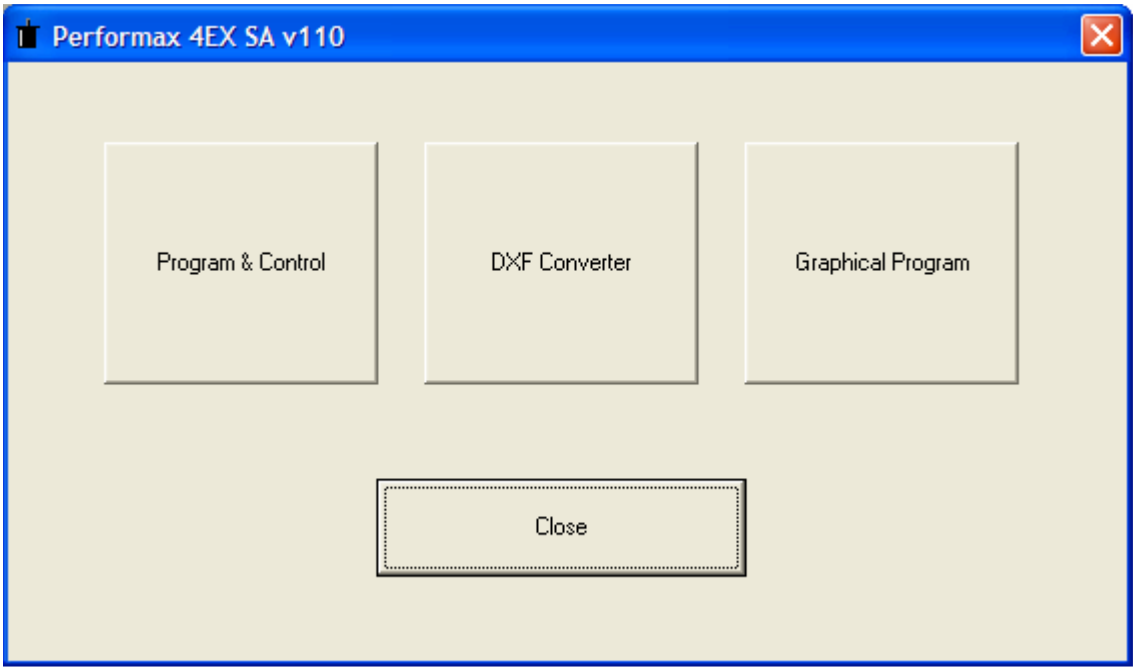
3) Supply power and ground to the PMX-4EX-SA module (12-24VDC). Connect PMX-4EX-SA and PC using USB cable

4) Download "4EX SA USB Test Program" from the web site:
   www.arcus-technology.com/support

5) Once program has started, you will see the following box:

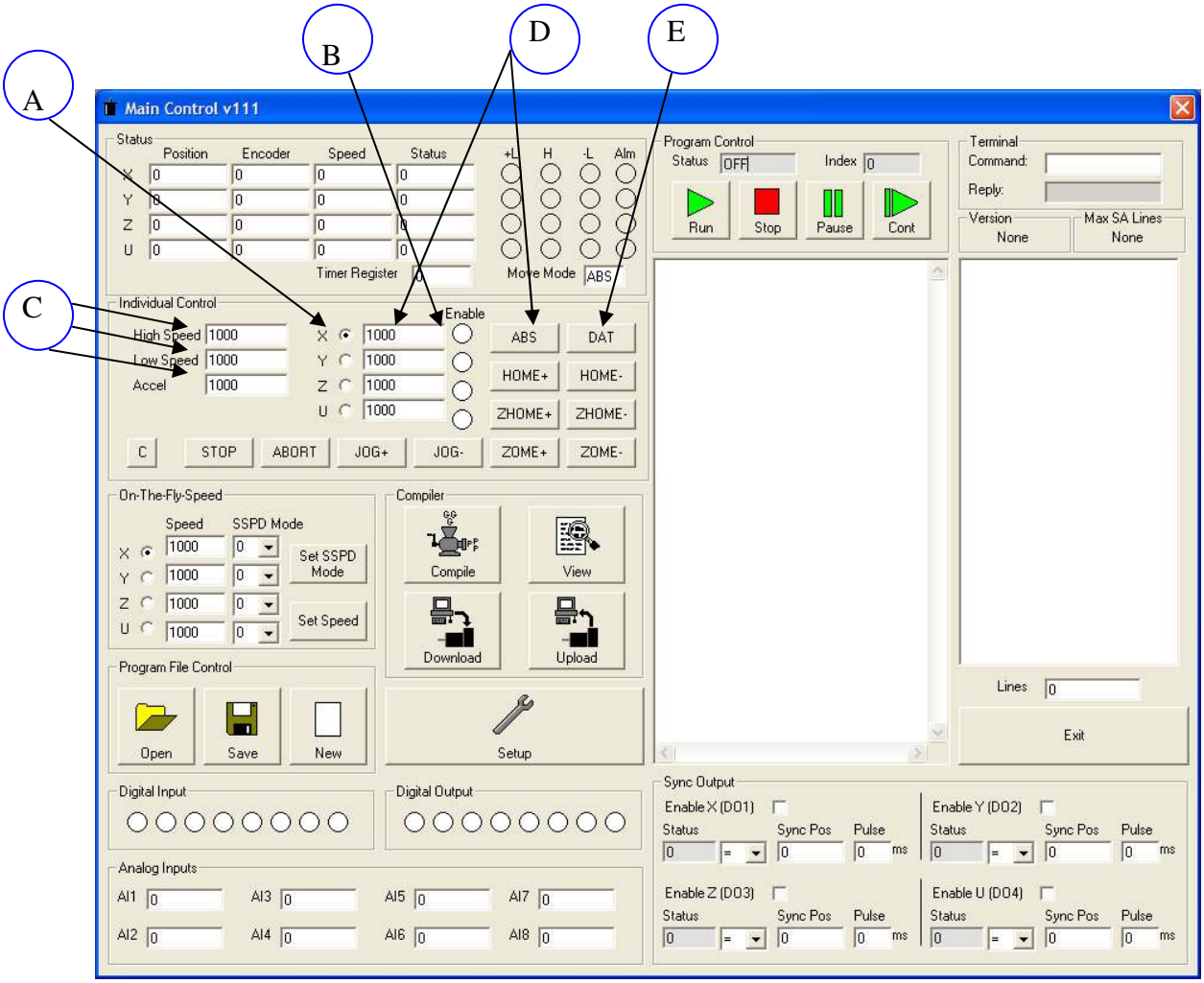


Click on "Start USB Comm" to control the PMX-4EX-SA via USB



Then click on “Program & Control” to open a GUI to test PMX-4EX-SA features

6) Once communication begins, you can start controlling the device.



A. To select the X-axis, check the X-axis radio button.
B. To toggle the enable output for the X-axis, click the X-axis enable button.
C. Change speed and acceleration values to see moves at different speed.
D. To move to position, enter the target position and perform move ABS.
E. To move back to zero position, push DATUM button.

For detailed information see Windows Program: USB.

3. Top Board Options

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

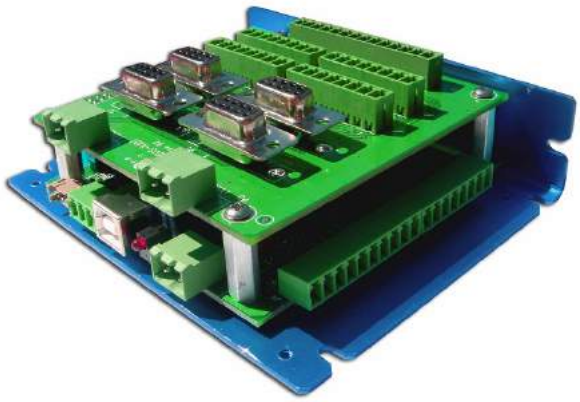
Standard Top Board (TB-S)

Standard Top Board consisting of 3.81 mm headers for X/Y/Z/U pulse/dir/enable outputs and alarm inputs.

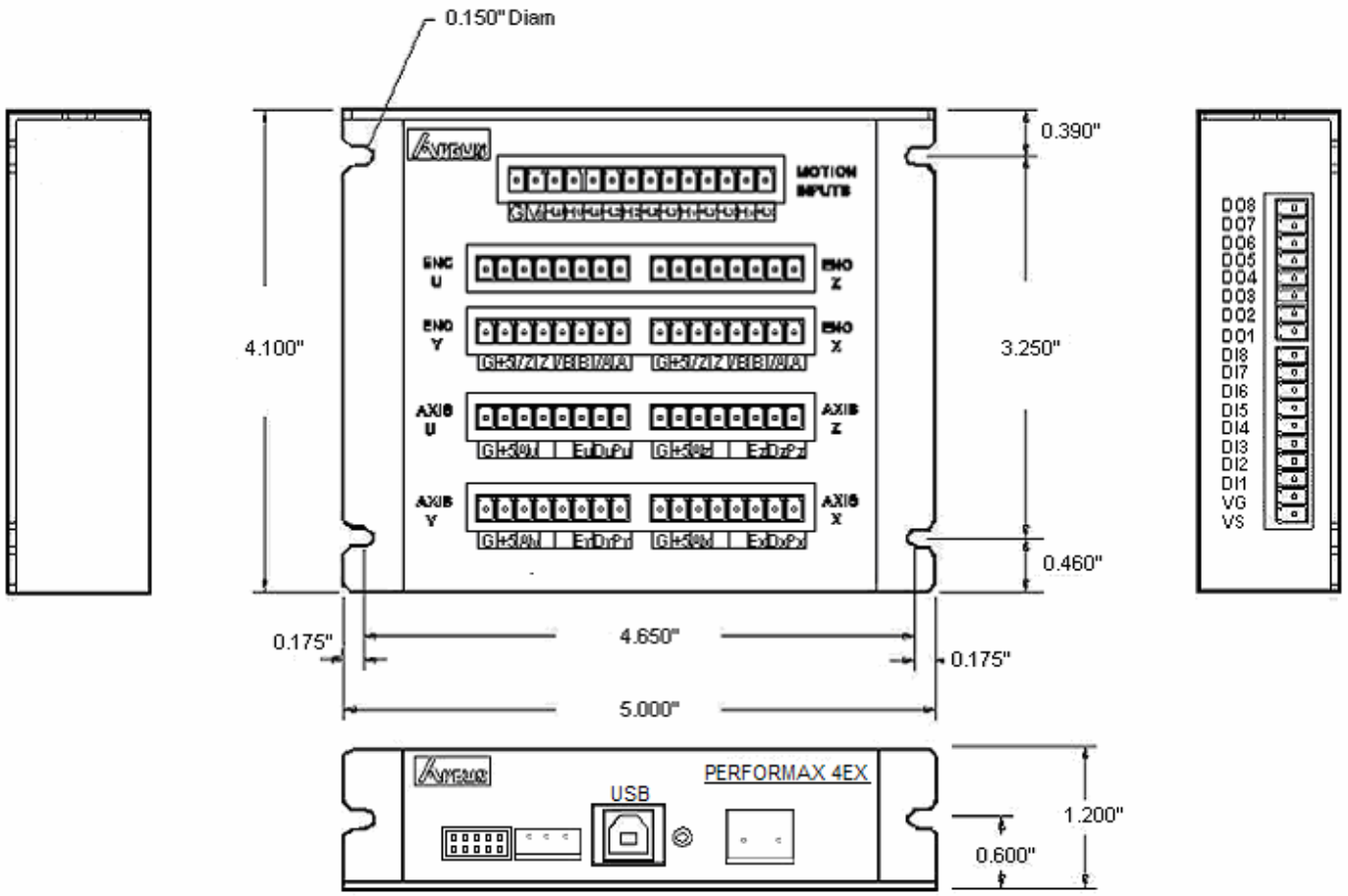
DB9 Top Board (TB-DB9)

DB9 Top Board consisting of DB9 headers for X/Y/Z/U pulse/dir/enable outputs and alarm inputs. The DB9 headers on these top boards are pin-to-pin compatible with the Arcus DriveMax DRV series motor + drivers.

Currently, the DB9 top board option does not come with top cover enclosure.



4. Dimensions



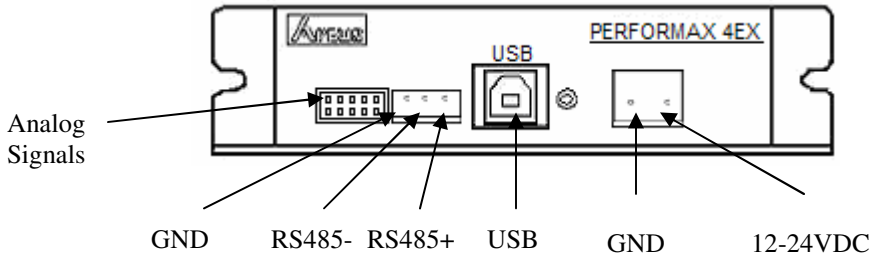
Note: Dimension of bottom plate is the same for both standard as well as DB9 top board

5. Connections and Pin Outs

A. Connecting input power, USB and RS-485

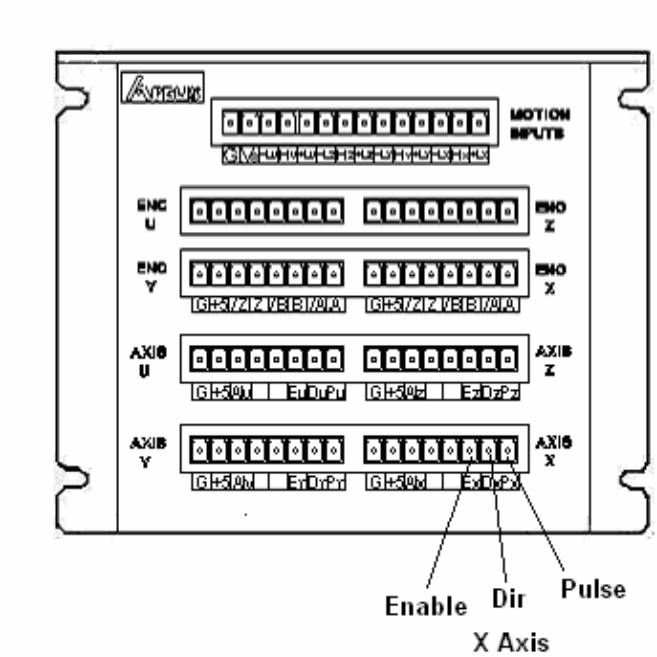
In order for PMX-4EX-SA to operate, it must be supplied with +12VDC to +24VDC. Power pin as well as communication port pin outs are shown below.

Note that only one method of communication can be used at the same time (i.e. user can not communicate via USB as well as RS-485 at the same time)



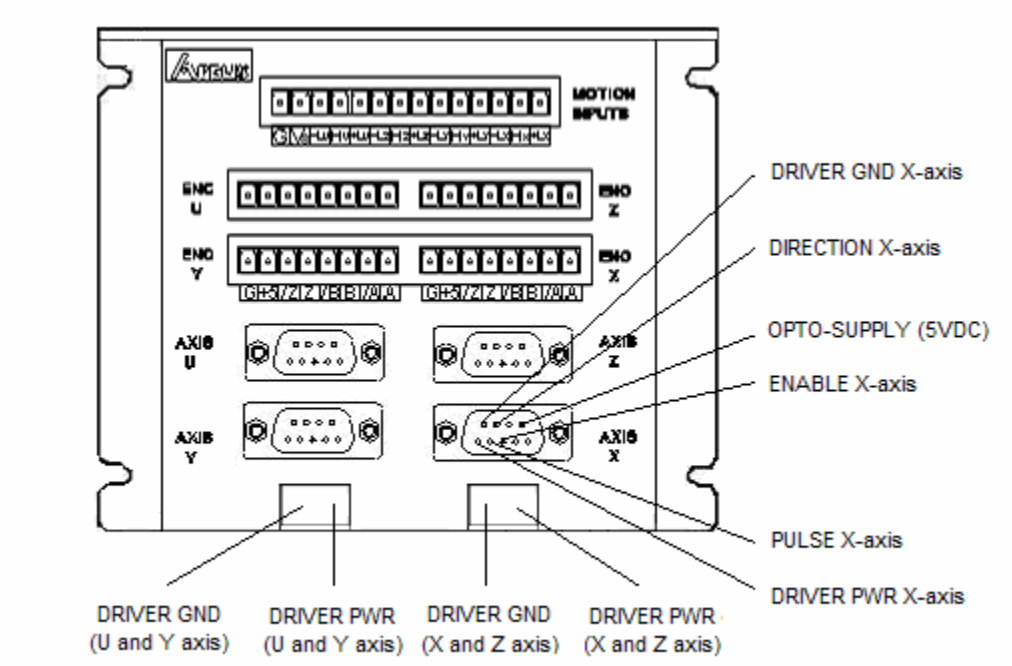
B1. Connecting to a stepper driver (Standard Top Board Option)

Each axis has pulse, direction, and enable outputs for stepper driver control. The following shows the connector location for X axis pulse/dir/enable outputs.



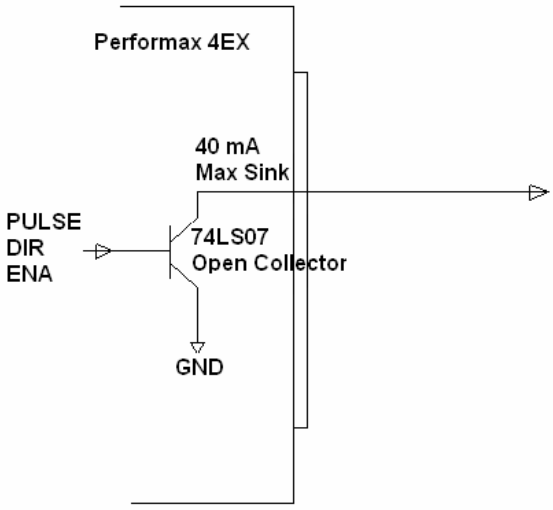
B2. Connecting to a stepper driver (DB9 Top Board)

Each axis has pulse, direction, and enable outputs for DriveMax-DRV control. Following shows the connector location for X axis pulse/dir/enable outputs.

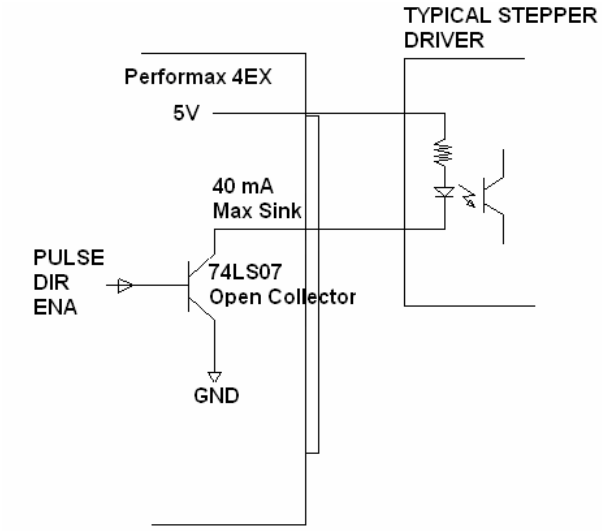


The pins on the DB9 headers can be connected directly to a DriveMax-DRV module (pin-to-pin compatible)

Pulse/Dir/Enable outputs for both the standard and DB9 top boards are all open collector outputs capable of sinking up to 40mA of current.



Example of Pulse/Dir/Enable connection to stepper driver with opto-isolated input is shown below.

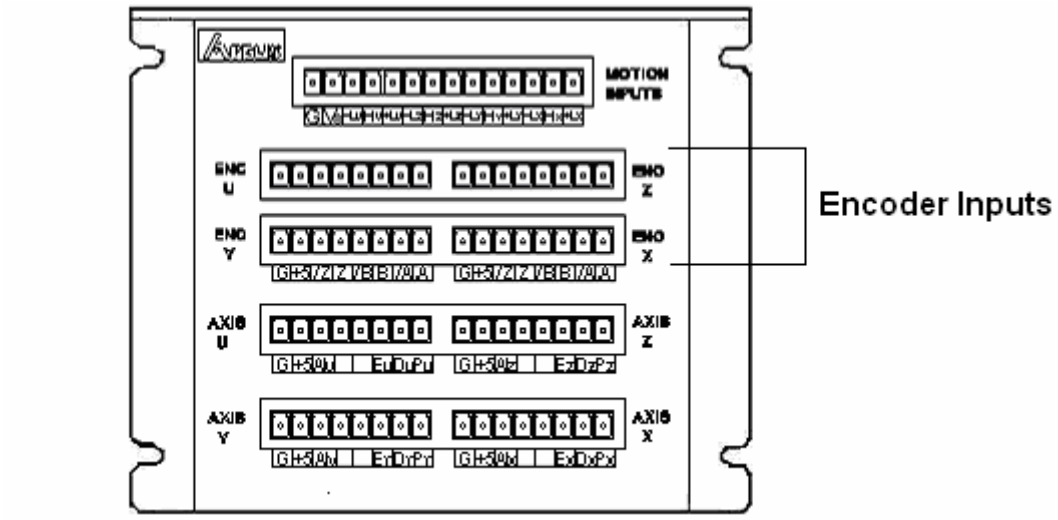


C. Connecting Encoders

PMX-4EX-SA supports both single-ended and differential quadrature encoder inputs. Inputs signals are 5V TTL.

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.



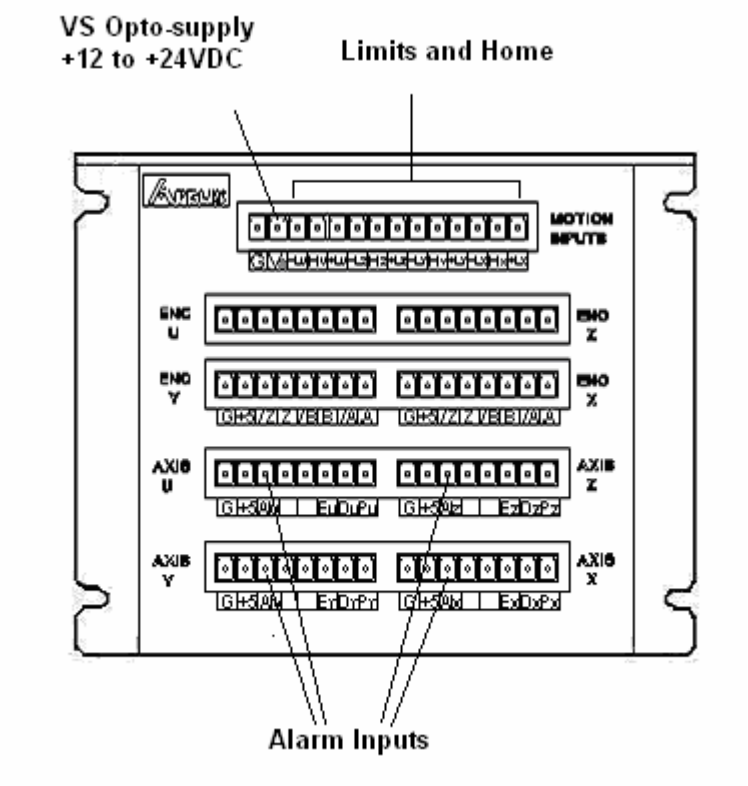
Note: Encoder pins are identical for both standard as well as DB9 top board

D. Connecting Limits/Home/Alarm

PMX-4EX-SA has opto-isolated +limit, -limit, home, and alarm inputs for each axis.

In order for these opto-isolated inputs to work properly, VS (opto-isolator voltage supply) must be supplied. Range of VS is from +12VDC to +24VDC.

To trigger the opto-isolated inputs, sink the limit or home input signal to the ground of the Vs. For wiring diagram, see "Connecting Digital Inputs and Outputs"

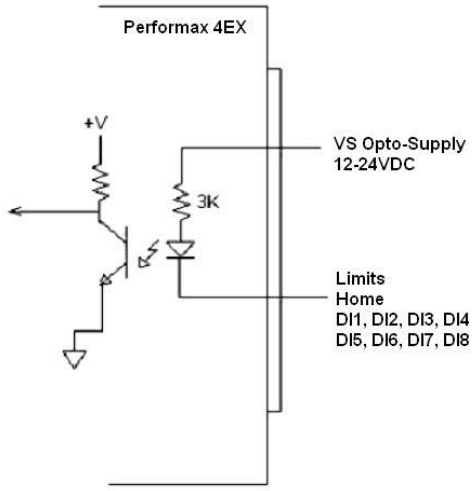


Note: Limit/Home input pins are identical for both standard as well as DB9 top board. Alarm input is not available on the DB9 top board.

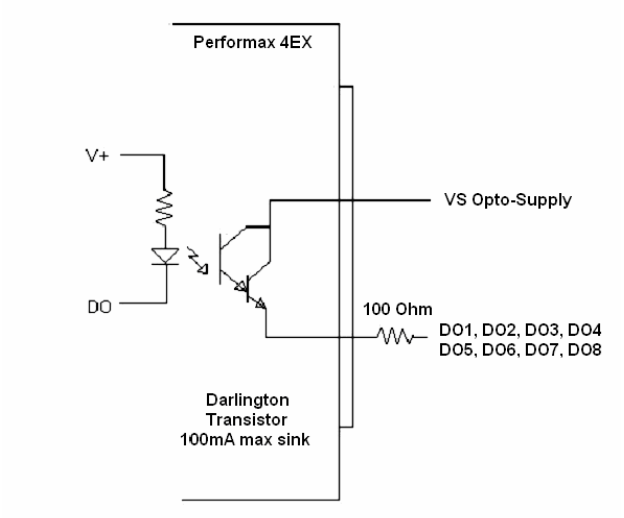
E. Connecting Digital Inputs and Outputs

PMX-4EX-SA has 8 opto-isolated digital inputs and 8 opto-isolated digital outputs.

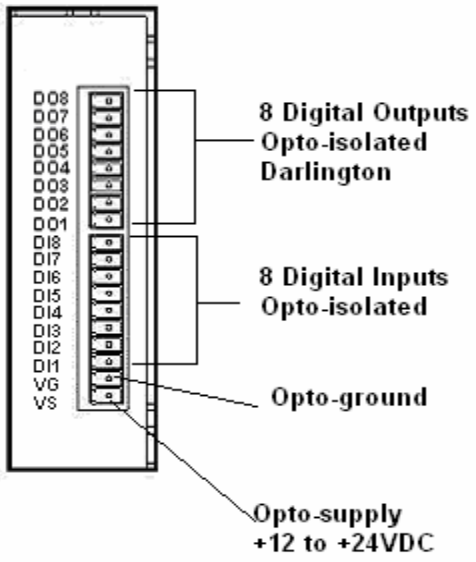
In order for these opto-isolated inputs and outputs to work properly, VS (opto-isolator voltage supply) located on the side connector and VG (opto-isolator voltage ground) also located on the side connector must be supplied. Range of VS is from +12VDC to +24VDC.



To trigger the opto-isolated digital inputs, sink the digital input signal to the ground of the VS.



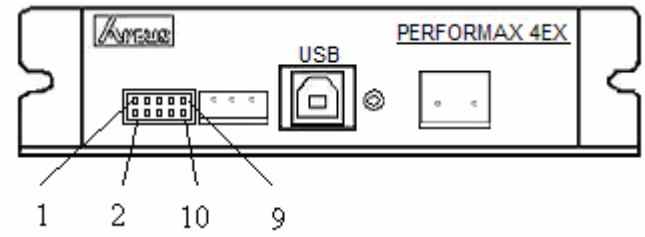
For the opto-isolated outputs, the digital output signal will source from the VS opto-supply when the signal is turned on.



Note: Digital inputs and outputs are found on the bottom board of the PMX-4EX-SA. Therefore, pin out is the same regardless of top board choice.

F. Connecting Analog Inputs

PMX-4EX-SA has 8 10-bit analog inputs. The inputs are voltage based, which accept from 0 to 5 VDC.



Description	Pin	Pin	Description
5V output	1	2	GND
AI7	3	4	AI8
AI5	5	6	AI6
AI3	7	8	AI4
AI1	9	10	AI2

Note: Analog inputs are found on the bottom board of the PMX-4EX-SA. Therefore, pin out is the same regardless of top board choice

6. Electrical Specifications

Power Requirement

Supply Power Requirement:	**+12 to +24 VDC**

USB 2.0 Communication Interface

USB Connector Type:	**B Type**
USB Communication Compliance:	**USB 2.0**
Recommended Max USB Cable Length:	**12 ft**

RS-485 Communication Interface

Baud Rate:	**9600, 19200, 38400, 57600, 115K**
Type:	**2-wire**
Protocol:	**RS-485 Arcus ASCII command support**

Important Note:
Factory default setting for the baud rate is 9600 bps, with Device Name = 4EX00

Pulse, Dir, Enable Outputs

Type:	**Open-collector output**
Maximum sink voltage:	**+24 VDC**
Maximum sink current:	**40 mA**

+Lim, -Lim, Home, Alarm and Digital Inputs

Type:	**Opto-isolated inputs**
Voltage range:	**+12V to +24VDC**
Max sink current:	**40 mA**

Digital Outputs

Type:	**Opto-isolated Darlington outputs**
Max voltage:	**+12V to +24VDC**
Max source current:	**100 mA**

Analog Inputs

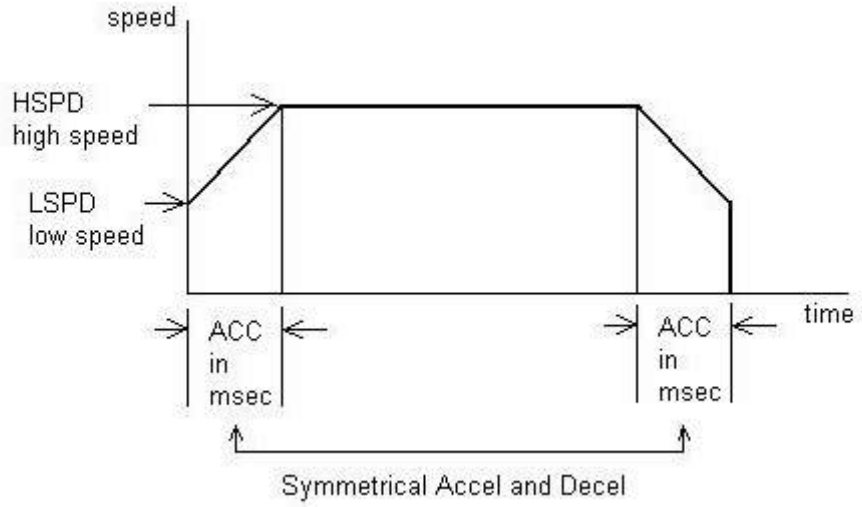
Type:	**10-bit, Voltage**
Max voltage:	**0V to +5VDC**
Max source current:	**10 mA**

7. Motion Control Feature Overview

PMX-4EX-SA is a 4-axis stepper motion controller able to generate up to 6M pulses per second for each of the axis.

Acceleration and Speed Settings

By default, 4EX incorporates trapezoidal velocity profile as shown below.



Acceleration and deceleration time is in milliseconds and are symmetrical. Use **ACC** command to set and get acceleration/deceleration value.

> *Example: To set the acceleration to 500 milliseconds, issue **ACC=500** command. To read the current acceleration setting, issue **ACC** command without the '=' character.*

High Speed and Low Speed are in pps (pulses/second). Use **HS** and **LS** to set and get global high speed and low speed settings.

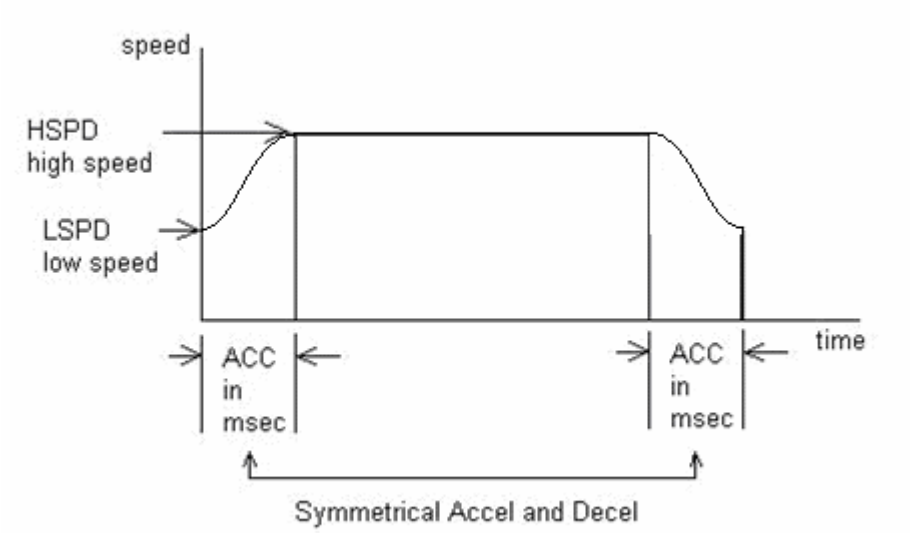
> *Example: To set the high-speed to 1500 pulses/second, issue **HS=1500** command. To read the current high-speed setting (not the actual speed), issue **HS** command without the '=' character and the reply will be the current high speed setting.*

To set different speed settings for each axis, use the **HS[axis], LS[axis]** and **ACC[axis]** commands. By default, all moves use the global speed settings, unless, ALL parameters (i.e. high speed, low speed, and acceleration) for a certain axis is configured.

Example: To set the high-speed of the X-axis to 1500 pulses/second, and the Y-axis to 2000 pulses/second, issue the following speed setting commands:

```
HSX=1500     ' set high speed for x-axis only
HSY=2000     ' set high speed for y-axis only
LSX=300      ' other parameters for the axis MUST be set as well for
LSY=300      ' the controller to use the individual speed settings instead
ACCX=100     ' of the global speed settings
ACCY=100
```

S-curve velocity profile is shown below.



Use **SCV[axis]** command to enable s-curve velocity profile instead of trapezoidal for a certain axis.

Note on speed settings:

The minimum value of **LS** setting depends on the **HS** setting. See chart below:

HS [pps]	Minimum LS [pps]
1-65 K	1
65K-130 K	2
130K-325 K	5
325K-650 K	10
650K-1.3 M	20
1.3M-3.2 M	50

3.2M-6 M	100

Note on acceleration:

The allowable acceleration values depend on the **LS** and **HS** settings. Please see chart below:

HS [pps]	Minimum ACC [ms]	Accel Delta [pps]
1-65 K	1	50
65K-130 K	1	100
130K-325 K	1	200
325K-650 K	1	800
650K-1.3 M	1	1500
1.3M-3.2 M	1	3800
3.2M-6 M	1	7500

Speed Delta: For every increment of **Accel Delta**, the maximum value of acceleration increases by 1000 ms (1.0 seconds).

Examples:

a) If **HSPD** = 100K, **LSPD** = 100:
   a. Get Speed delta: ((100,000 – 100) / 100) = 999
   b. Max acceleration allowable: 999 x 1,000 ms = **999,000** ms (999 sec)

b) If **HSPD** = 50,000K, **LSPD** = 49.5K:
   a. Get Speed delta: ((50,000 – 49,500) / 50) = 10
   b. Max acceleration allowable: 10 x 1000 ms = **10,000** ms (10 sec)

On-The-Fly Speed Change

On-the-fly speed change can be achieved with the **SSPD[axis]** command. **SSPD[axis]** command is only valid with trapezoidal acceleration.

During on-the-fly speed change operation, you must keep the initial and destination speeds within a certain window. See speed setting windows below:

SSPDM value	Lowest Speed [pps]	Highest Speed [pps]
0	SSPD not used	SSPD not used
1	1	65,000
2	2	130,000
3	5	325,000
4	10	650,000
5	20	1,300,000

6	50	3,200,000
7	100	6,000,000

To select a speed window, use the **SSPDM[axis]** command. At boot-up, the **SSPDM[axis]** value is equal to 0.

If you are to set your destination speed outside of your current window, the **SSPD[axis]** feature will not work correctly.

*Note: The lower the **SSPDM[axis]** value, the more accurate the pulse output speed will be. Therefore, it is recommended to choose the lowest **SSPDM[axis]** value as possible.*

To set acceleration of the on-the-fly speed change, use the **ACC** or **ACC[axis]** command. Set the acceleration before calling the **SSPD[axis]** command.

Note: The maximum acceleration value allowed depends on both the SSPDM value as well as the difference between the initial and destination speeds. See table below.

SSPDM value	Speed Delta Increment [pps]
0	SSPD not used
1	50
2	100
3	200
4	800
5	1500
6	3800
7	7500

Speed Delta: For every increment of speed delta, the maximum value of acceleration increases by 1000 ms (1.0 seconds).

Examples:

a) If **Destination Speed** = 300,000 pps, **Current Speed** = 250,000 pps:
   c. Get Speed delta: ((300,000 – 250,000) / 200) = 250
   d. Max acceleration allowable: 250 x 1,000 ms = **250,000** ms (250 sec)

b) If **Destination Speed** = 900,000 pps, **Current Speed** = 889,000 pps:
   e. Get Speed delta: ((900,000 – 889,000) / 1,500) = 7.3
   f. Max acceleration allowable: 7.3 x 1000 ms = **7300** ms (7.3 sec)

*Note: In order to begin normal operation after on-the-fly speed moves, it is required to first set **SSPDM** to 0.*

Individual Moves

For individual axis control use **X**, **Y**, **Z** and **U** command followed by the target position value. A single move command can consist of up to 4 target positions (one for each axis). If more than 1 axis is specified, the motion will be linearly interpolated.

Example:

1) ***"X1000":*** *Move X-axis to position 1000.*
2) ***"X1000 Y1000":*** *Move X-axis to position 1000, Y-axis to position 1000 using linear interpolation.*
3) ***"X1000 Y1000 Z100":*** *Move X-axis to position 1000, Y-axis to position 1000, Z-axis to position 100 using linear interpolation.*
4) ***"X1000 Y1000 Z100 U800":*** *Move X-axis to position 1000, Y-axis to position 1000, Z-axis to position 100, U-axis to position 800 using linear interpolation.*
5) ***"X1000 U800":*** *Move X-axis to position 1000, U-axis to position 800 using linear interpolation.*

Individual move commands use true S-curve acceleration and deceleration profile.

Circular Interpolation Moves

PMX-4EX-SA supports circular interpolation moves using the **CIRP** and **CIRN** commands. Circles are drawn using X,Y axes only.

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

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

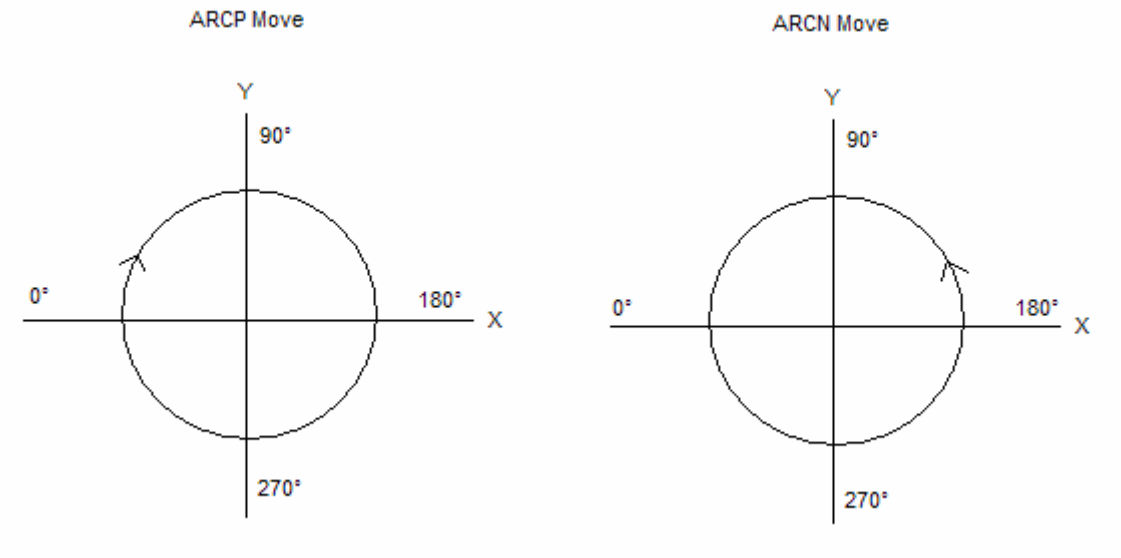
Arc Interpolation Moves

PMX-4EX-SA supports circular interpolation moves using the **ARCP** and **ARCN** commands. Arcs are drawn using X,Y axes only. Angle is in whole number in thousandth. For example, 45 degrees is 45,000.

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

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

The θ value should be calculated by making the 0° reference point to be the negative x-axis.

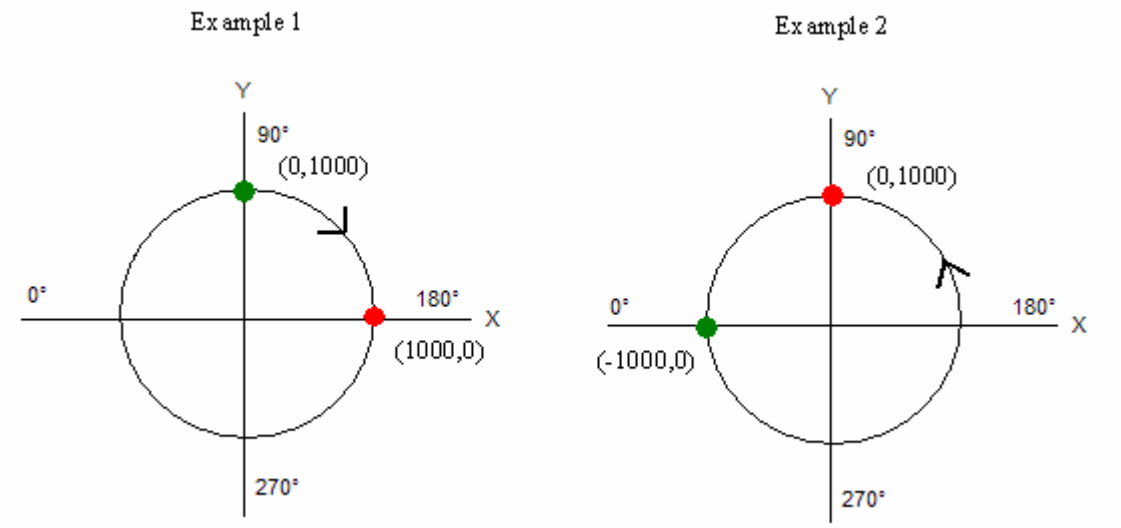


Example 1:

Arc start position:	(0,1000)
Arc end position:	(1000,0) in CW direction
Move command:	ARCP0:0:180000

Example 2:

Arc start position:	(-1000,0)
Arc end position:	(0,1000) in CCW direction
Move command:	ARCN0:0:450000



Note: PMX-4EX-SA does not allow a radius larger than 46399 pulses on arc or circular moves

Buffered Linear Interpolation Moves

PMX-4EX-SA supports buffered linear coordinated motions for X, Y, and Z-axes using **I** command. Each move has its own constant speed setting.

Example: To move to location X, Y, Z to 1000, 2000, 3000 position with speed of 250, use following command ***I1000:2000:3000:250***

Manual Acceleration Control

To control the acceleration or deceleration manually, gradually increase or decrease the speed value for each interpolated move. Use to use manual acceleration, disable automatic buffered move acceleration using the **IACC** command.

Automatic Acceleration Control

To control the acceleration or deceleration automatically, enable automatic buffered move acceleration with the **IACC** command. In this case, the speed acceleration profile will be automatically generated between sequential buffered moves.

The acceleration value used can be updated using the **ACC** command.

Linear Interpolations is buffer move size is 36 points. Buffered move mode is turned on with **BO** command turned off with **BF** command. With the buffered mode on, as soon as the I command is issued the motion will start.

Buffered moves apply only to X, Y and Z axes.

Homing

Use **H** command for homing the motor. Use following format for the command:

H[axis selection X,Y,Z,U][direction + or -][homing mode 0,1,2,3]

Four homing modes are available.

0 - Using home switch
1 - Using limit switch
2 - Using home switch and encoder index channel
3 – Using encoder index channel only

Examples:

1) *To home X axis in positive direction using the home sensor only (homing mode 0)*
   HX+0

2) *To home Y axis in negative direction using the limit senor only (homing mode 1)*
   HY-1

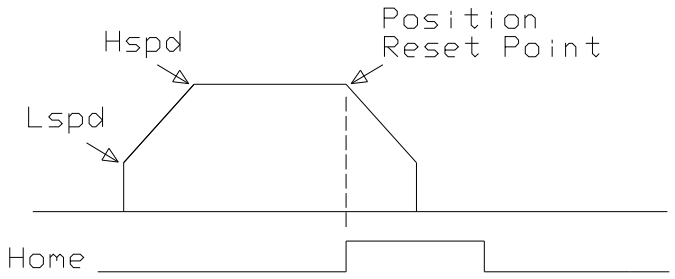
3) *To home Z axis in positive direction using the home and encoder index channel (homing mode 2)*

HZ+2

4) *To home Z axis in positive direction using encoder index channel only (homing mode 3)*

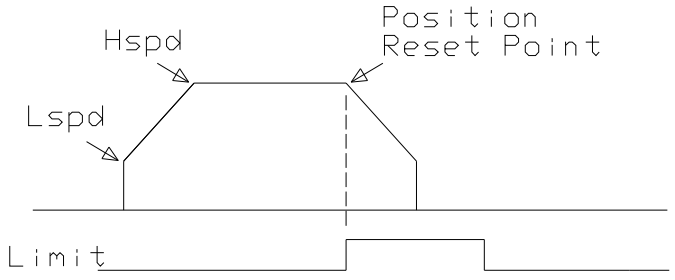
HZ+3

Homing Mode 0



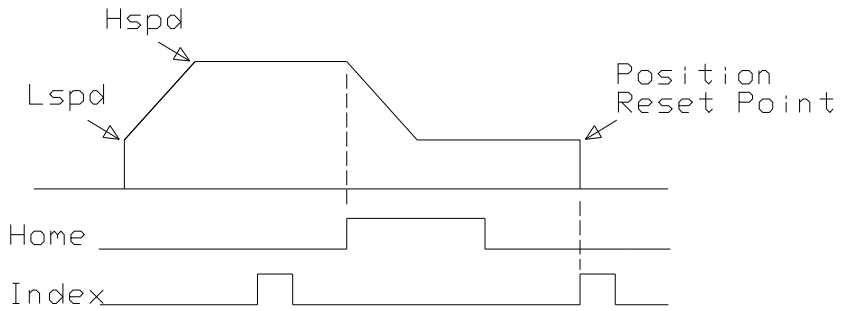
In homing mode 0, the axis ramps from low speed (Lspd) to high speed (Hspd) and maintains the high speed until the home sensor is triggered. At the home sensor trigger, pulse and encoder position counters reset to zero and the deceleration is done to ensure smooth ramp down to low speed. At the end of the home routine, actual position may not be exactly zero due to ramp down at the home sensor trigger.

Homing Mode 1



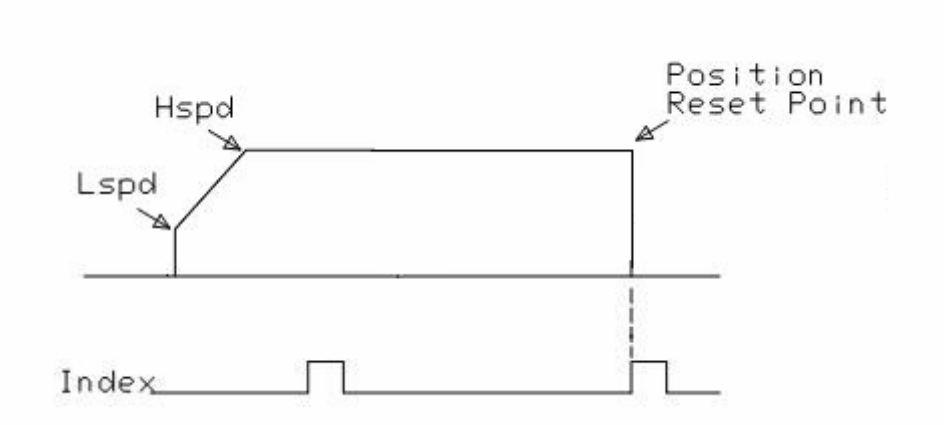
In homing mode 1, the axis ramps up from low speed to high speed and maintain the high speed until the limit sensor is triggered. At the limit sensor trigger, pulse and encoder position counters reset to zero and the deceleration is done to ensure smooth ramp down to low speed. At the end of the home routine, actual position may not be exactly zero due to ramp down at the sensor trigger.

Homing Mode 2



In homing mode 2, the axis ramps from low speed (Lspd) to high speed (Hspd) and maintain the high speed until the home sensor is triggered. At the home sensor trigger, deceleration is done to ensure smooth ramp down to low speed. Low speed is maintained until the index channel of the encoder is triggered at which point the motion stops and pulse and encoder position counters are reset to zero.

Homing Mode 3



In homing mode 3, the axis ramps from low speed (Lspd) to high speed (Hspd) and maintains the high speed until the index channel of the encoder is triggered at which point the motion stops and pulse and encoder position counters are reset to zero.

Jogging

Use **J** command for jogging the motor. Use format for the command:

J[axis selection X,Y,Z,U][direction + or -]

Jogging uses the previous high speed, low speed, and acceleration setting.

Stopping

When motor is moving, **ABORT[axis selection X,Y,Z,U]** command will immediately stop all the motor. Use **ABORT** command to immediately stop ALL axes.

To decelerate stop, use **STOP[axis selection X,Y,Z,U]** command. Use **STOP** command to decelerate stop ALL axes.

Note: If any interpolation operation is in process while a **STOP[axis selection X,Y,Z,U** or **ABORT[axis selection X,Y,Z,U** command is entered, all axes will stop.

Polarity

Using **POX, POY, POZ, and POU** command to get and set polarity of following signals:

Bit 0 - Home
Bit 1 - Alarm
Bit 2 – Limit (X axis limit input setting controls limit switch polarity for all axes)

Motor Position Reading and Setting

Motor positions can be set and read using the **PP** command which returns the pulse position of all 4 axes. Encoder positions can be set and read using **PE** command which returns the encoder position of all 4 axes. Encoders are set to 4X reading.

To manually set the pulse position use following format:

P[axis selection X,Y,Z,U]=[position value]

To manually set the encoder position use following format:

E[axis selection X,Y,Z,U]=[position value]

Pulse Speed Reading

Current pulse rate or speeds can be read using the **PS** command.

Motor Status Reading

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

Bit	Description
0	Accelerating
1	Decelerating
2	Constant Speed
3	Alarm input status
4	+ Limit input status
5	- Limit input status
6	Home input status
7	+ Limit Error
8	- Limit Error
9	Alarm Error

Limits and Alarm

If during motion, limit in the move direction is triggered, the motor will stop immediately and the limit error status bit will be on. If alarm input is triggered move in any direction will immediately stop the motor and the alarm error status bit will be on.

If the motor is not moving, alarm or limit trigger will not affect the status.

Once the motor status is in limit or alarm error, the error must be cleared to issue another move command. Error can be cleared using **CLR[axis]** command.

During buffered move module, if limit or alarm error is triggered, the motors will stop and buffered move will be disabled.

Enable Outputs

4 bits of enable outputs are available to enable or disable the driver if the stepper driver has such input. Enable outputs are open collector outputs similar to pulse/dir outputs. Enable output can also be used for general-purpose output. Use **EO** command to read or set the enable outputs. Enable output value is a 4 bit value. For example, enable output value of 15 (1111 in binary or F in hex) means all bits are turned on. To access individual bits, use **EO[1-4].**

Digital Outputs

8 bits of digital outputs are available on PMX-4EX-SA. Use **DO** command to read and set the digital output value. Digital outputs are Darlington opto-isolated outputs and when the output is turned on, the signal sources VS. Digital output value is an 8 bit value. For example, digital output value of 255 (11111111 in binary or FF in hex) means all bits are turned on. To access individual bits, use **DO[1-8].**

Sync Outputs

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

Axis	Synchronization Output
X	DO1
Y	DO2
Z	DO3
U	DO4

Note: While feature is enabled for an axis, the corresponding digital output can not be controlled by user.

Use **SYN[axis]O** to enable the synchronization output feature for an axis.

Use **SYN[axis]F** to disable the synchronization output feature for an axis.

Use **SYN[axis]P** to read and set the synchronization position value for an axis. (28-bit signed number)

Use **SYN[axis]C** to set the synchronization condition.

1 – Turn the output on when the encoder position is EQUAL to sync position. If the synchronization output is done during motion, the sync output pulse will turn on only when the encoder position and sync position are equal.
2 - Turns output on when the encoder position is GREATER than the sync position.
3 – Turns output on when the encoder position is LESS than sync position.

Use **SYN[axis]T** to set the pulse width output time (ms). This parameter is only used if the synchronization condition is set to 1. Note the maximum pulse width is 10 ms. If this parameter is set to 0, the output pulse will depend on how long the encoder value is equal to the sync position.

Use **SYN[axis]S** to read the synchronization output status for an axis

0 – Sync output feature is off
1 – Waiting for sync condition
2 – Sync condition occurred

Timer Register

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

Note: This timer is a uses a lower priority interrupt. Therefore, it is most accurate when a PC is not polling the PMX-4EX-SA with USB commands. In this case, the USB commands take precedence of over the timer register. If a timer is desired while polling for USB commands, use the **DELAY** stand-alone command instead.

Digital Inputs

8 bits of digital inputs are available on PMX-4EX-SA. Use **DI** command to read the digital input value. Digital inputs are opto-isolated inputs and when the input is sunk to the ground, the digital input is triggered. Digital input value is an 8 bit value. For example, digital input value of 255 (11111111 in binary or FF in hex) means all bits are turned on. To access individual bits, use **DI[1-8].**

Analog Inputs

8 x 10-bit analog inputs are available on PMX-4EX-SA. Use **AI[1-8]** command to read the analog input value. Range is from 0-5000 mV.

Joystick Control

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

Axis	Analog Input
X	AI1
Y	AI2
Z	AI3
U	AI4

During joystick operation, analog input of 0 mV to 2500 mV represents negative joystick direction and analog input of 2500 mV to 5000 mV represents positive joystick direction. 2500 mV represents the zero joystick position.

Maximum joystick speed is set using the **JV1**, **JV2**, **JV3** and **JV4** variables.

Maximum speed change (delta) is set using the **JV5**, **JV6**, **JV7** and **JV8** variables.

To set tolerance of the zero joystick position, use **JV9**, **JV10**, **JV11** and **JV12** variables. For example, if **JV9** is set to 100, then the zero range for X-axis joystick control will be from 2400 mV to 2600 mV.

Joystick control also has soft limit controls. Limits are broken into: negative outer limit, negative inner limit, positive inner limit and positive outer limit.

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. Same goes for the negative direction and negative limits.

Summary of joystick control parameters

Parameter	Description
JV1	X-axis maximum joystick speed at 5000 mV and 0 mV
JV2	Y-axis maximum joystick speed at 5000 mV and 0 mV
JV3	Z-axis maximum joystick speed at 5000 mV and 0 mV
JV4	U-axis maximum joystick speed at 5000 mV and 0 mV
JV5	X-axis maximum speed change
JV6	Y-axis maximum speed change
JV7	Z-axis maximum speed change
JV8	U-axis maximum speed change
JV9	X-axis zero tolerance range for analog input
JV10	Y-axis zero tolerance range for analog input
JV11	Z-axis zero tolerance range for analog input
JV12	U-axis zero tolerance range for analog input
JL1	X-axis negative outer limit
JL2	X-axis negative inner limit
JL3	X-axis positive inner limit
JL4	X-axis positive outer limit
JL5	Y-axis negative outer limit
JL6	Y-axis negative inner limit
JL7	Y-axis positive inner limit
JL8	Y-axis positive outer limit
JL9	Z-axis negative outer limit
JL10	Z-axis negative inner limit
JL11	Z-axis positive inner limit
JL12	Z-axis positive outer limit
JL13	U-axis negative outer limit
JL14	U-axis negative inner limit
JL15	U-axis positive inner limit
JL16	U-axis positive outer limit

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

StepNLoop Closed-Loop Control

PMX-4EX-SA module has closed loop position control algorithm called StepNLoop control for accurate positioning when using encoder feedback.

StepNLoop control does following operations:

1) Position Delta monitoring: Delta position is the difference between the actual and the target position. When the Delta goes over the allowed Correction Range, the motor is stopped and the StepNLoop Status goes into the "stall" error state. Delta monitoring is done for all moves including homing and jogging. View the Delta value by using the **DX[axis]** command.

2) Position Correction at the end of the move: Correction of the motor position is done at the end of any targeted move.

Following are configuration required for StepNLoop control:

StepNLoop Parameters	**Description**
Tolerance Range	When the actual encoder position is within desired encoder position by this tolerance range, no position correction is done. Use **SLT[axis]** command to set the tolerance range.
Error Range	When the actual encoder position is within desired encoder position by this error range, position correction is done when idle. If the actual encoder position is outside of this error range, the motor status goes to error state. Use **SLE[axis]** command to set the correction range.
Correction Attempt Number	This is the maximum number of correction tries that the controller will attempt. If the correction cannot be done within this number of tries, the motor status goes to error state. Use **SLA[axis]** command to set the maximum correction attempt number.

To enable and disable the StepNLoop feature use **SL[axis]** command.

To read the StepNLoop status, use **SLS[axis]** command.

Following are the StepNLoop status values:

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 **CLR[axis]** command.
9	Correction Attempt Error. To clear this error, use **CLR[axis]** command.
10	Stall Error. **DX[axis]** value has exceeded the **SLE[axis]** value. To clear this error, use To clear this error, use **CLR[axis]** command.
11	Limit Error. To clear this error, use **CLR[axis]** command.
12	NA

Notes:

Once StepNLoop is enabled, position move commands are in terms of encoder position. For example, X1000 means to move the x-axis to encoder position 1000. ***This is applies to individual as well as interpolated moves.***

Once StepNLoop is enabled, the speed is in encoder speed.
For example HSPDX=1000 when StepNLoop is enabled means that the target high speed of the x-axis is 1000 encoder counts / second. ***This applies only to individual axis moves.***

Linear Interpolation w/ StepNLoop: If StepNLoop is used during a linear interpolation move, StepNLoop must be enabled for all axes being moved. Also note that unlike the individual axis moves, the speed during a linear interpolation is calculated as pulse/sec, NOT encoder counts/sec.

Arc/Circular Interpolation w/ StepNLoop: If StepNLoop is used during an arc/circular interpolation move, StepNLoop must be enabled for both X and Y axes. Also note that unlike the individual axis and linear interpolation moves, the StepNLoop ratio of X and Y MUST be the same. The speed during an arc/circular interpolation move is calculated as pulse/sec, NOT encoder counts/sec.

StepNLoop correction is done only when the pulse rate is idle. For example, when the motor is moving, correction is not done. Once the pulse rate is idle, StepNLoop correction is done.

Device Number and Baud Rate

Performax 4EX comes with following default factory communication setting:

Baud Rate: **9600**
Device Name: **4EX00**

PMX-4EX-SA module provides the user with the ability to set the device number for RS-485 multi-drop applications. In order to make these changes, first set the desired device number using the **DN** command. Please note that this value must be within the range [4EX00, 4EX99].

PMX-4EX-SA module provides the user with the ability to change the baud rate for RS-485 communication. In order to make these changes, first set the desired baud rate using the **DB** command. Please note the following baud rate codes:

Device Baud Value	Baud Rate (bps)
1	9600
2	19200
3	38400
4	57600
5	115200

To write the values to the device number and baud rate permanently to flash memory, use the **STORE** command. After a complete power cycle, the new device ID will be used. Note that before a power cycle is done, the settings will not take effect.

Calling subroutines from USB

Once a subroutine is written into the flash of the PMX-4EX-SA, they can be called via USB communication using the **GS** command. The subroutines are referenced by their subroutine number [0-31]. If a subroutine number is not defined, the PMX-4EX-SA will return with an error.

Standalone Program Specification

Memory size: 1785 assembly lines ~ 10.5 KB

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

Storing to Flash

The following items are stored to flash:

- Device Number
- Baud rate
- Polarity settings
- S-curve settings
- Joystick settings
- Buffered interpolated move automatic acceleration
- Automatic program run on power up

Note: When standalone program is downloaded, the program is immediately written on the flash memory.

8. USB Communication Protocol

Performax USB communication is USB 2.0 compliant.

Communication between the PC and Performax is done using Windows compatible DLL API function calls as shown below. Windows programming language such as Visual BASIC, Visual C++, LABView, or any other programming language that can use DLL can be used to communicate with the Performax module.

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

Important Note: PerformaxCom.dll only supports single-threaded programming. Calling PerformaxCom.dll functions from different threads will lead to unexpected behavior even if the functions are not being used by different threads simultaneously.

USB Communication API Functions

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);

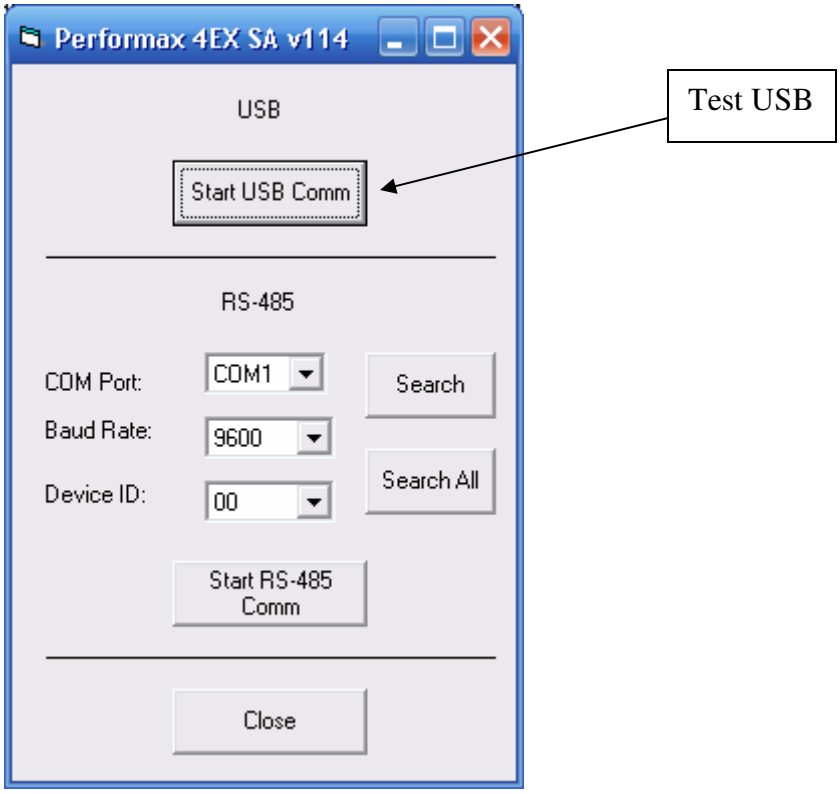
- This function is used to send command and get reply. Number of bytes to read and write must be 64 characters.

9. PMX-4EX-SA Program (USB)

PMX-4EX-SA comes with user friendly Windows Program to quickly communicate, test, program, and debug the PMX-4EX-SA unit.

Before running the program, make sure to run the Performax USB Driver Setup program. Both setup program and the manual for Performax USB Driver setup can be downloaded from the web site

www.arcus-technology.com/support



10. RS-485 Communication Protocol

If RS-485 communication is required, first you need to communicate using RS-232 and use the Windows program to change the communication method to RS-485, download the setup, and store to flash. Once communication method is changed, you need to reboot the module for the new parameter to take effect and communicate through RS-485.

When communicating on RS-485, it is recommended to add 120 Ohm terminating resistor between 485+ and 485- signal on the last module.

Communication Protocol

Communication protocol and commands are the same for both RS-485.

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]

[Null] character has ASCII code 13.

Examples:

For querying the x-axis polarity

Send: @00POX[CR]

Reply: 7[CR]

For jogging the x-motor in positive direction

Send: @00JX+[CR]

Reply: OK[CR]

For aborting any motion in progress

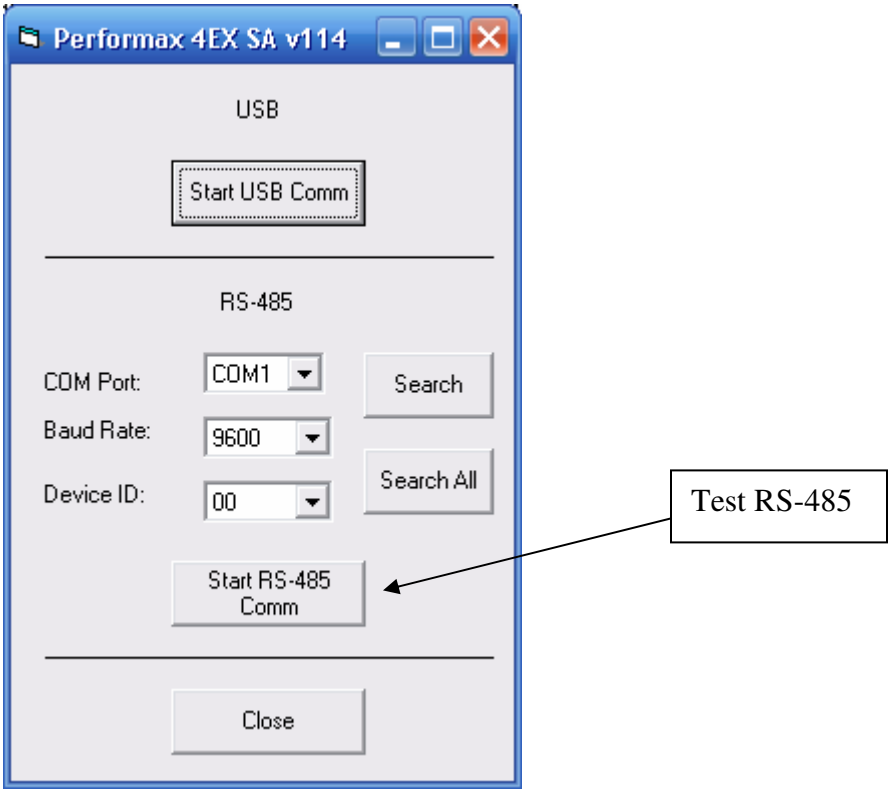
Send: @00ABORT[CR]

Reply: OK[CR]

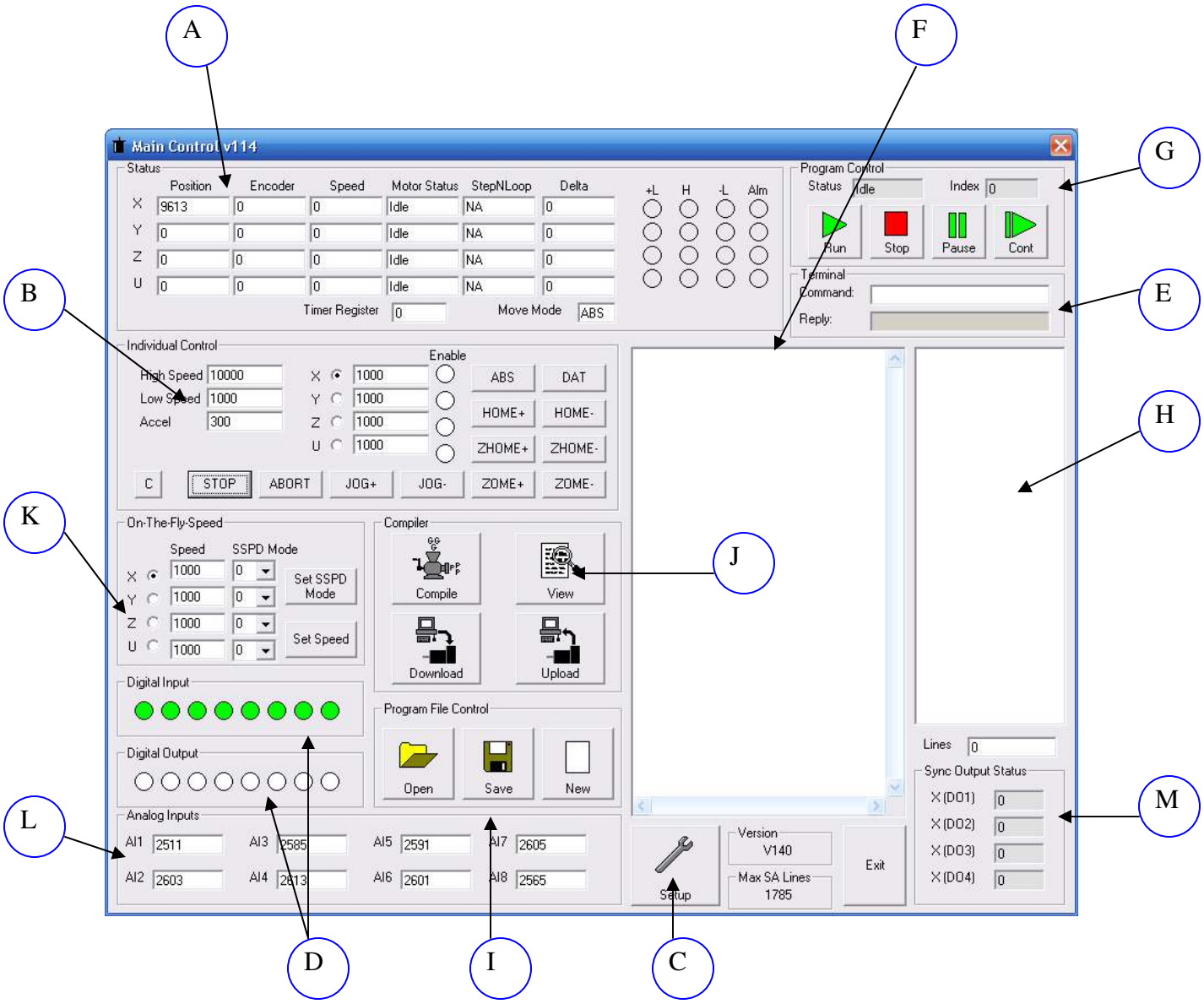
11. PMX-4EX-SA Program (RS-485)

PMX-4EX-SA comes with user friendly Windows Program to quickly communicate, test, program, and debug the PMX-4EX-SA unit over RS-485.

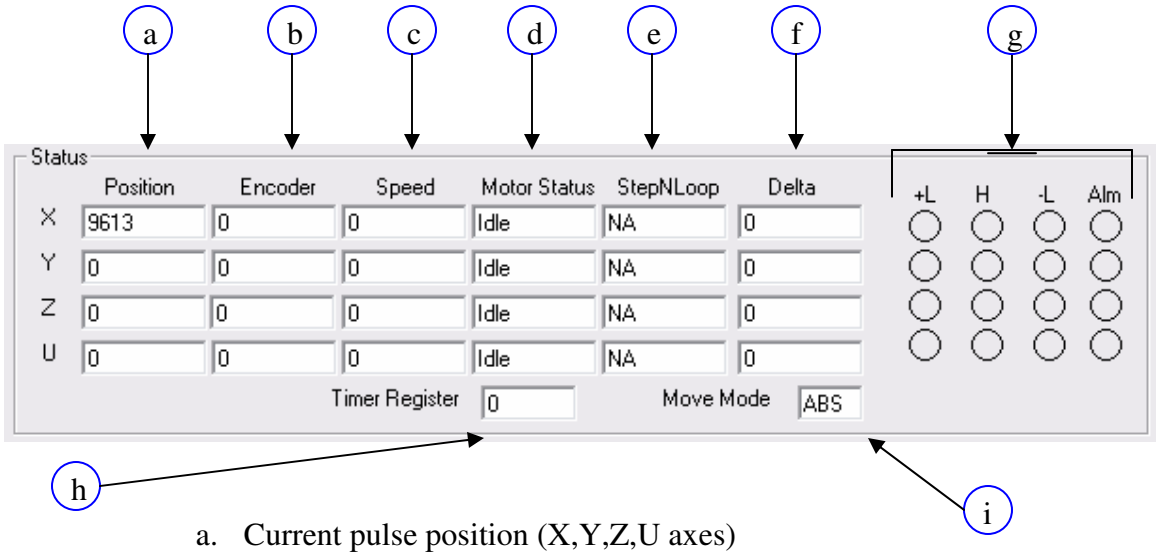
Before starting RS-485 communication, be sure to connect RS-485 signals to PMX-4EX-SA. See "Connections and Pin outs".



12. Program and Control Software

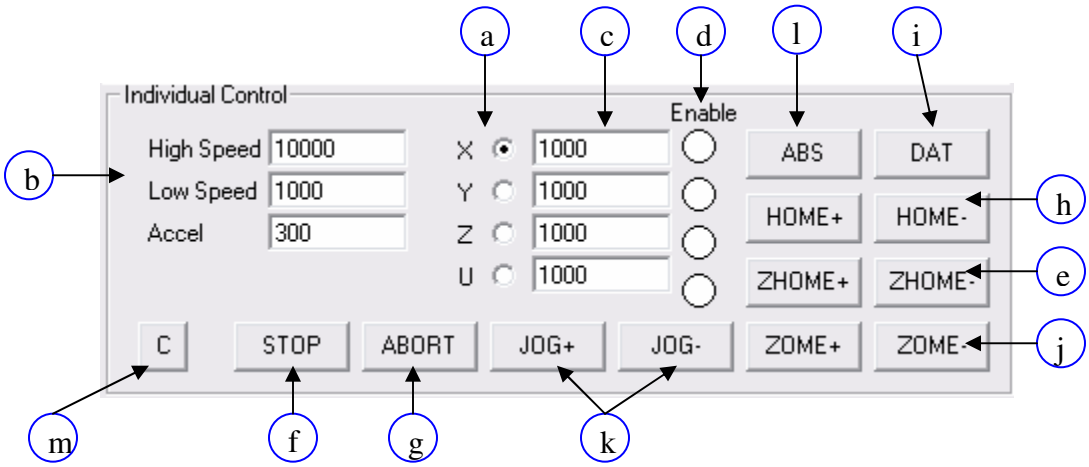


A. Status box:



a. Current pulse position (X,Y,Z,U axes)
b. Current encoder position (X,Y,Z,U axes)
c. Current speed (X,Y,Z,U axes)
d. Motor status (X,Y,Z,U axes)
    i. Idle – motor is not moving.
    ii. Accel – motor is accelerating
    iii. Const – motor is running in constant speed
    iv. Decel – motor is decelerating
    v. +LimError – plus limit error
    vi. –LimError – minus limit error
e. StepNLoop status
f. StepNLoop delta status
g. –Limit, + Limit, Home and Alarm input status (X,Y,Z,U axes)
h. Timer register status (counts down)
i. Move mode status: ABS – absolute move, INC – incremental move

B. Individual Control box:



a. Select X/Y/Z/U axis to control.

b. Global High speed, low speed, and acceleration is entered here (X,Y,Z,U axes). To give each axis individual speed parameters, enter HS[axis], LS[axis] and ACC[axis] commands via the command line.

c. Target position is entered here (X,Y,Z,U axes)

d. Enable – motor power is turned on or off by clicking on these circles (X,Y,Z,U axes)

e. ZHOME+/ZHOME- Home sensor and encoder index channel is used to home.

f. ABORT – the motion is immediately stopped without deceleration.

g. STOP – the motion is stopped with deceleration.

h. HOME+/HOME- homing is done using only the home sensor. When the home sensor is triggered during homing, the position counter is reset to zero and the motor decelerates to low speed and stops. After homing, the position is not necessarily zero due to deceleration after the trigger of the home switch.

i. DAT – moves the motor to the zero target position.

j. ZOME+/ZOME- Only encoder index channel is used to home.

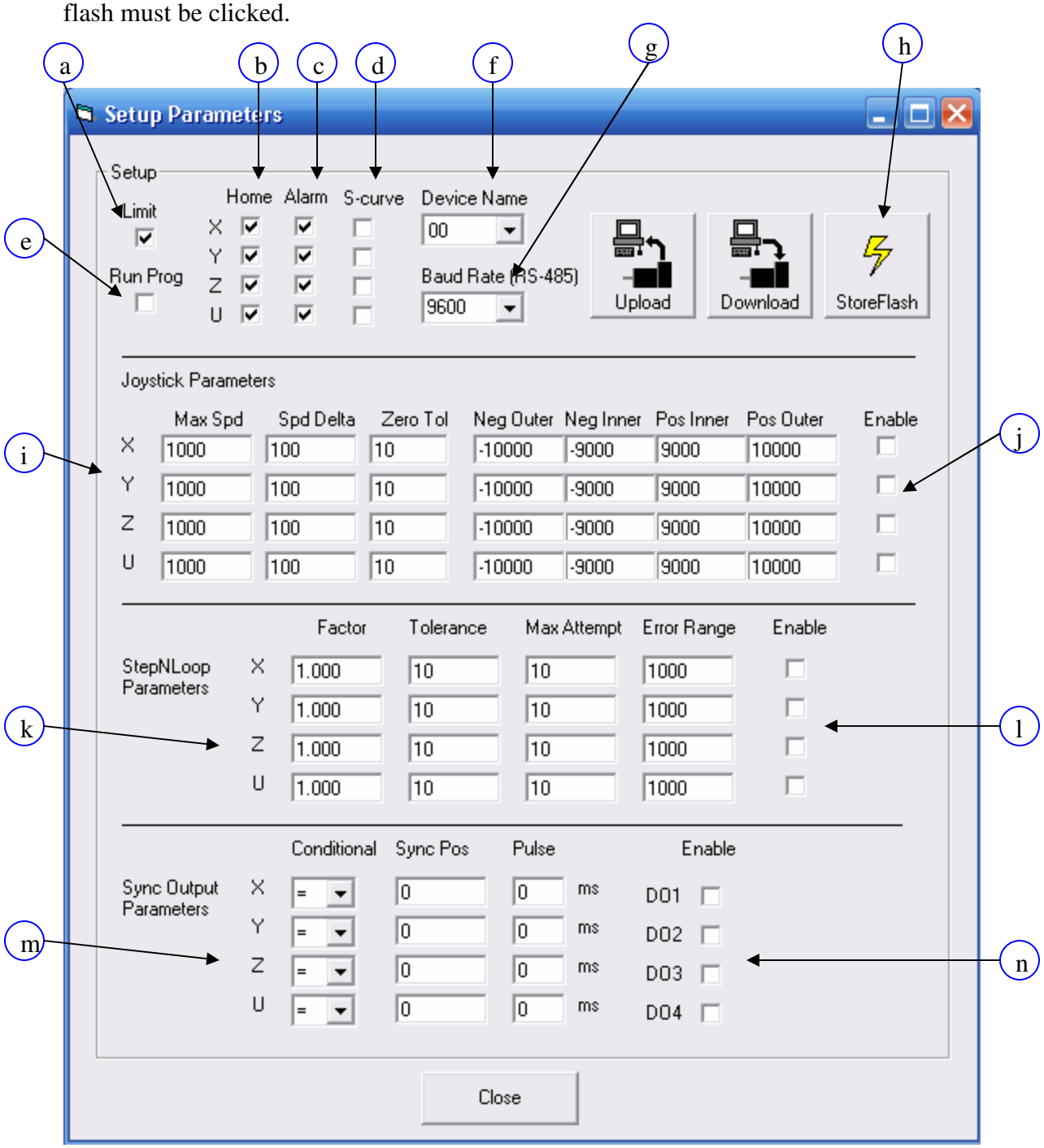
k. JOG+/JOG- - jogs the motor in positive and negative direction.

l. ABS – moves the motor to the target absolute position using the high speed and the low speed and the acceleration values.

m. CLEAR – clear motor error

C. Setup box:

PMX-4EX-SA configuration values are automatically loaded when the program is started. In order for any configuration to be permanent, store to flash must be clicked.

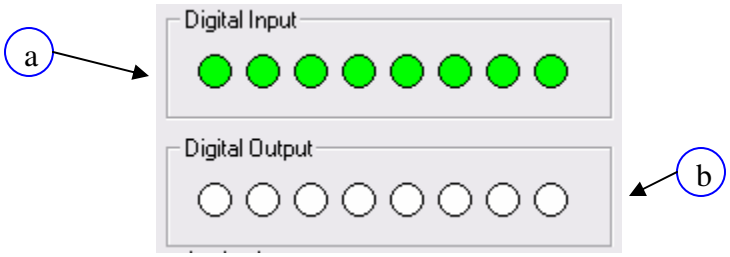


a. Limit polarity: global for all X,Y,Z and U axes
b. Home polarity (X,Y,Z,U axes)

c. Alarm polarity (X,Y,Z,U axes)
d. S-curve enable/disable (X,Y,Z,U axes)
e. Run Prog – Click and perform a store to flash to have a standalone program run on boot up
f. Device Name – set device name for device. Must be in the range [4EX00-4EX99]
g. Baud Rate – set baud rate of the device (9600, 12900, 38400, 57600, 115200 bps)
h. StoreFlash – Store the settings to flash memory. The following parameters are stored to flash
    i. Device Number
    ii. Baud Rate
    iii. Polarity Settings
    iv. S-curve settings
    v. Joystick control settings
    vi. Automatic program run on power up
i. Joystick parameters
j. Joystick enable
k. StepNLoop parameters
l. Enable StepNLoop (X,Y,Z,U axes)
m. Sync output parameters
n. Enable sync output

Note: Standalone program is directly stored to flash memory when it is downloaded to the PMX-4EX-SA

D. Digital Input/ Digital Output boxes:

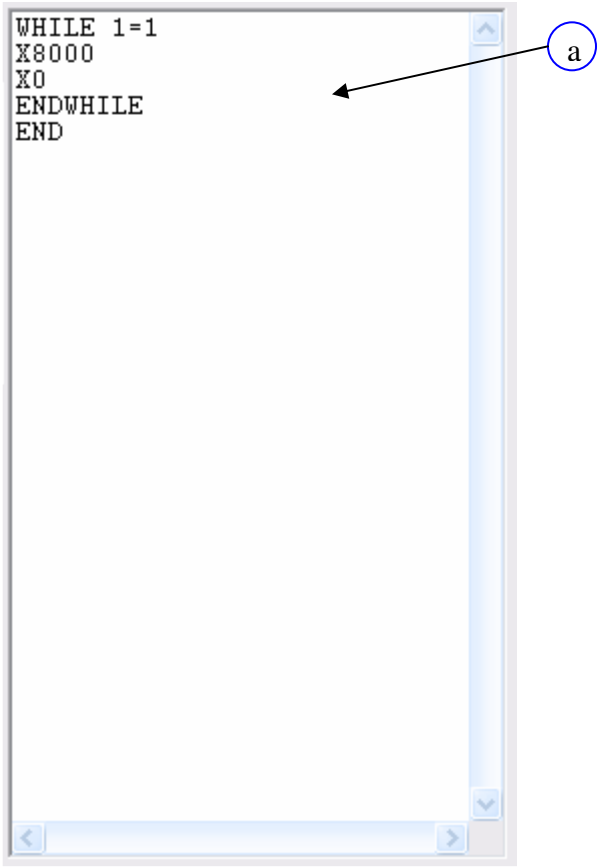


a. Digital input status DI1-DI8
b. Digital output status DO1-DO8. To turn on-off digital output, click on the corresponding circle.

E. Terminal box:

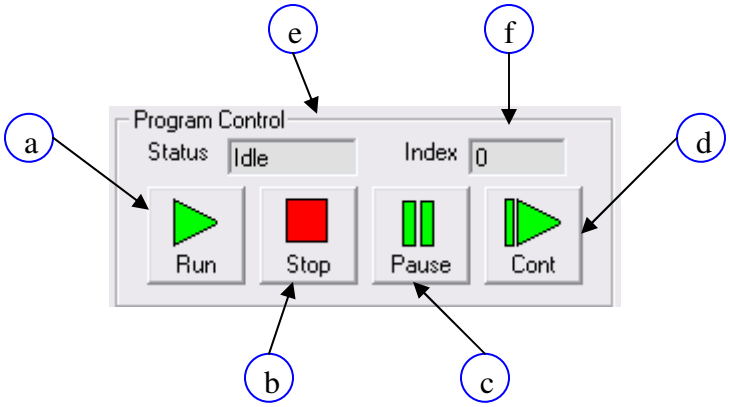
a. ASCII text command to be sent. To send, press "ENTER" on the keyboard.
b. Reply from the PMX-4EX-SA. Reply will appear immediately after the ASCII command is sent

F. Text Programming box:



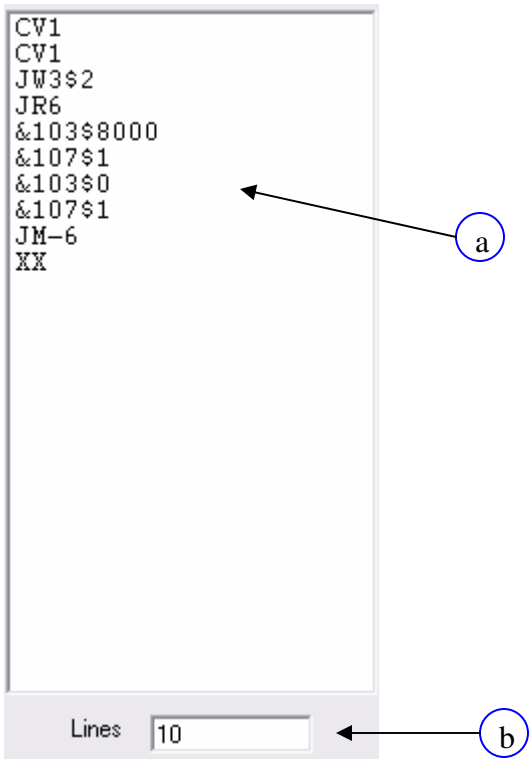
a. Text box for standalone program. For details on programming language, see section 13 of manual.

G. Program Control box



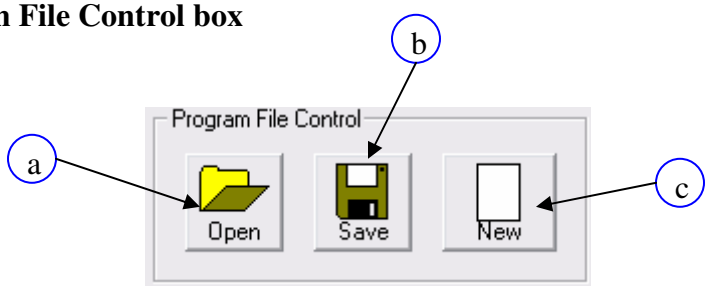
a. Run – program is run
b. Stop – program is stopped
c. Pause – program that is running can be paused
d. Cont – program that is paused can be continued
e. Status of standalone program
    i. Idle – Program stopped
    ii. Running – Program executing
    iii. Paused – Program paused
    iv. Error – Program in error state
f. Index – Current line of code that is being executed

H. Compiled Code box

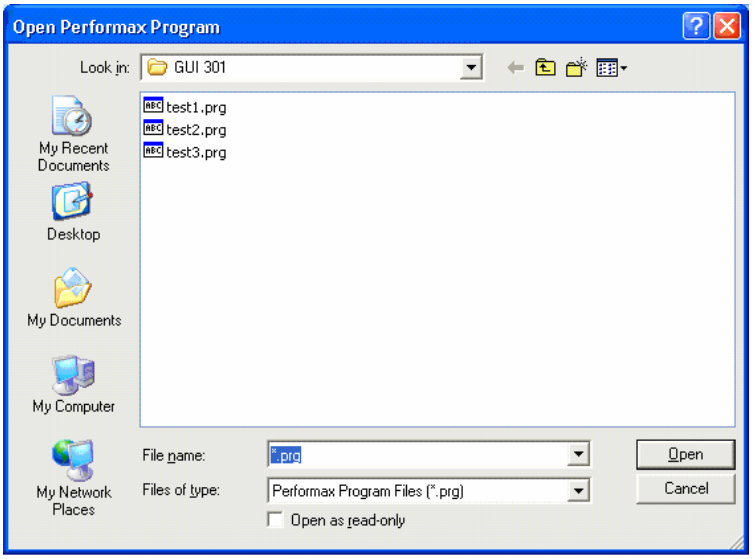


a. View assembly level code of the compiled code. This box is populated after the compile button is clicked.
b. Number of lines of compiled code. Note that maximum number of compiled code that PMX-4EX-SA supports is 1785.

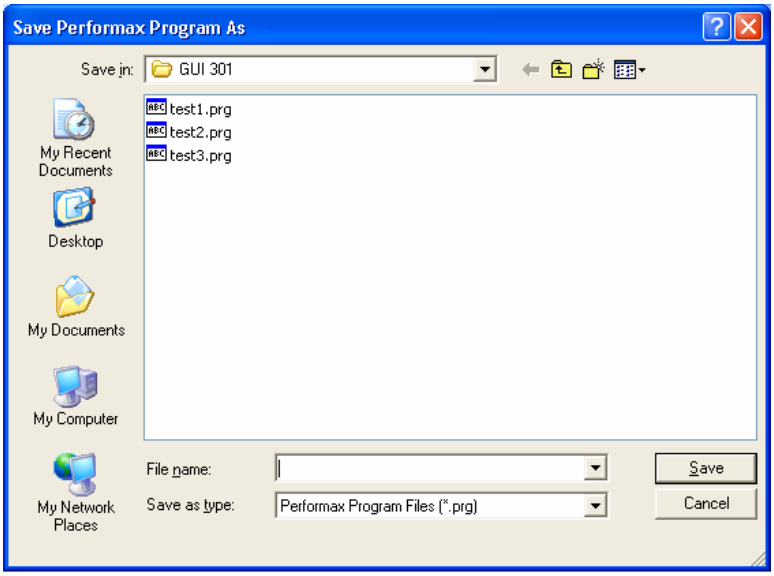
I. Program File Control box



a. Open - standalone program is loaded to the editor box. When this button is pressed, typical Windows file open dialog box will open:

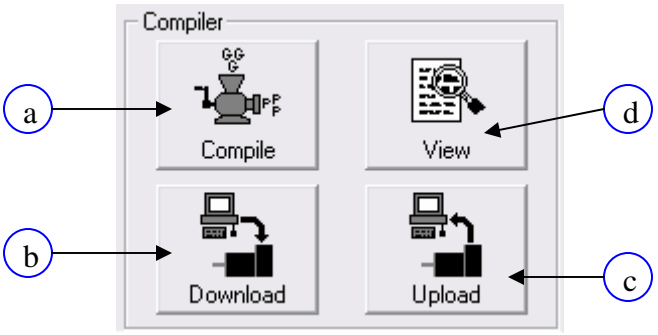


b. Save – standalone program in the text edit is saved to a file. When this button is pressed, typical Windows file save dialog box will open:



c. New – when this button is pressed, the text editor is cleared.

J. Compiler box



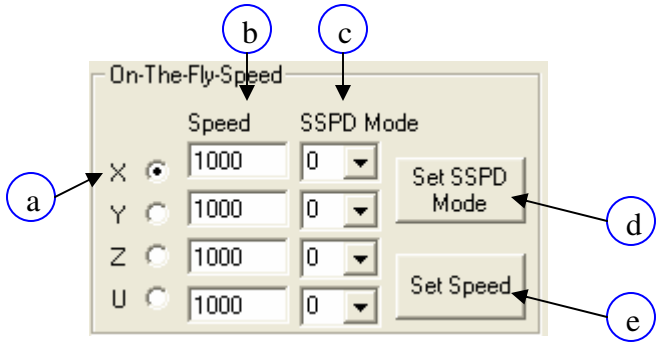
a. Compile code in text programming box into assembly level code that the PMX-4EX-SA understands.

b. After code has compiled, download the compiled code the PMX-4EX-SA. Note that text based code must first be compiled before downloaded.

c. Upload standalone code that is currently on your PMX-4EX-SA to the text programming box. This automatically translates assembly level language from the PMX-4EX-SA to readable text-based code.

d. View compiled code for easy cutting and pasting

K. On-The-Fly Speed box



a. Select X/Y/Z/U axis to control

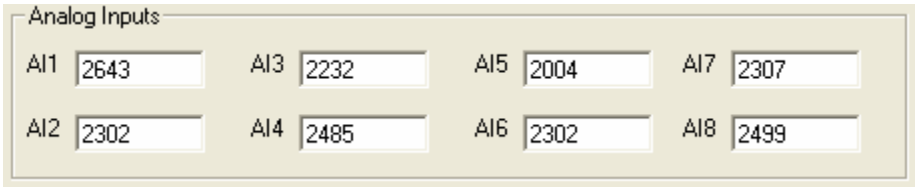
b. Select destination speed of the axis

c. Select SSPD mode for the axis. See On-The-Fly Speed section for details

d. Set SSPD mode for the axis.

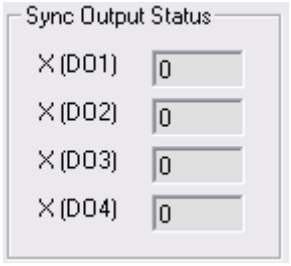
e. Set on-the-fly speed change. Acceleration will be taken for the "Accel" field of the Control box

L. Analog Inputs



Status of analog inputs AI1-AI8. Units are in milli-volts [0-5000 mV]

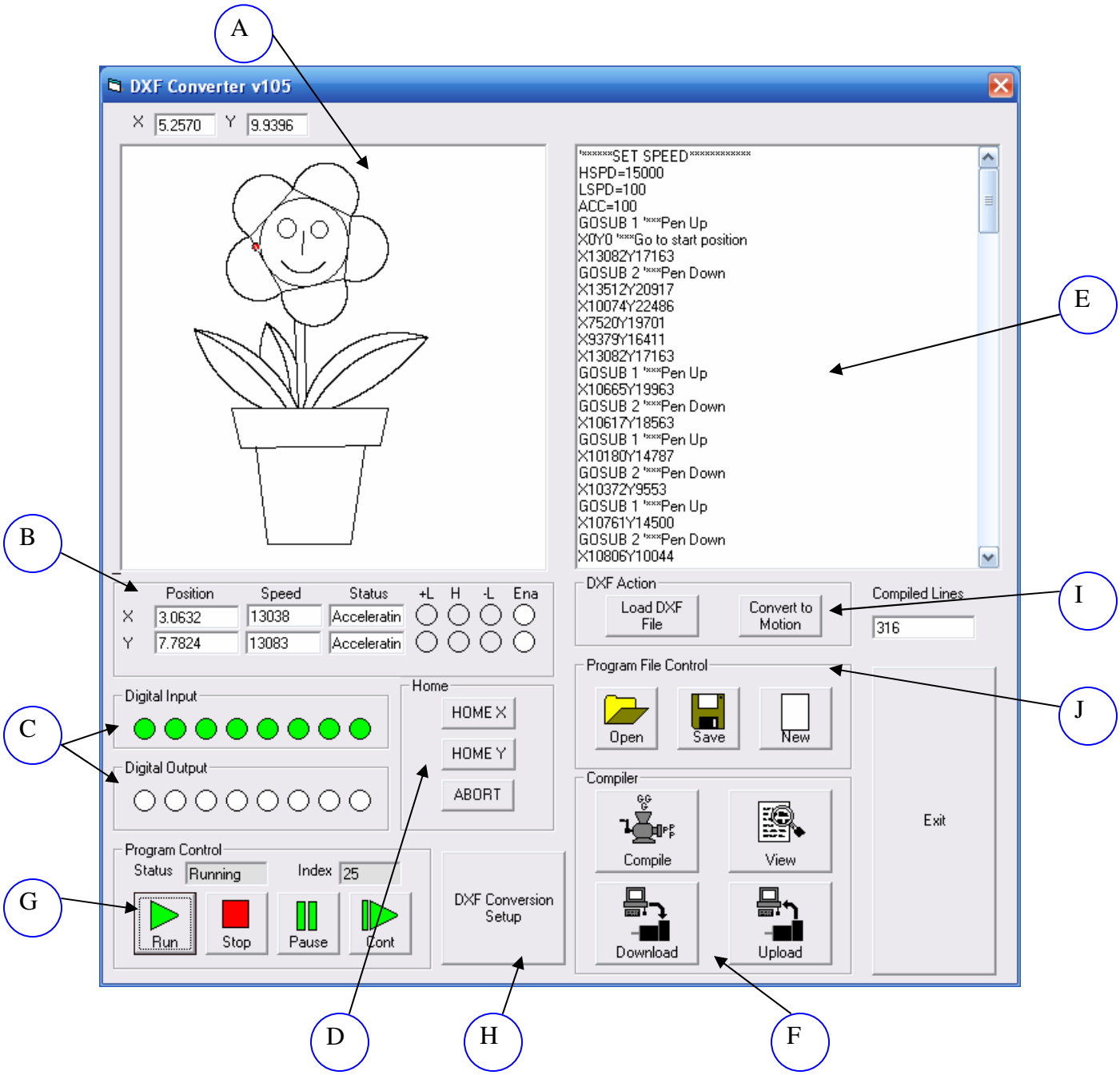
M. Sync Output



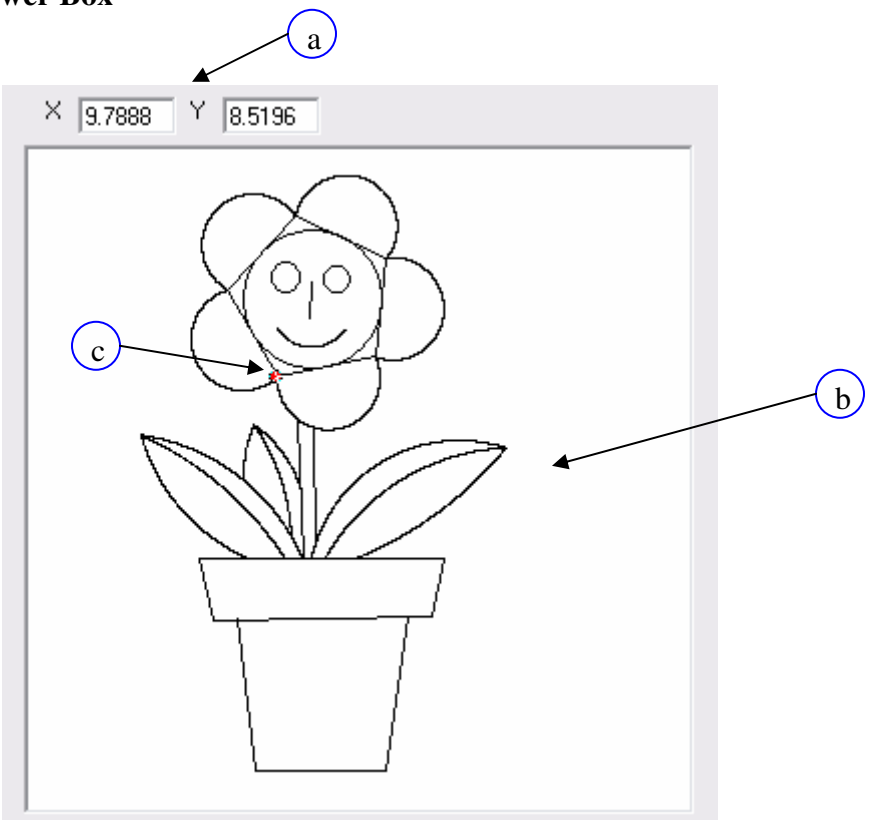
Sync output status for each axis.

- 0 = Sync output off
- 1 = Sync output waiting
- 2 = Sync output triggered

13. DXF Converter Software

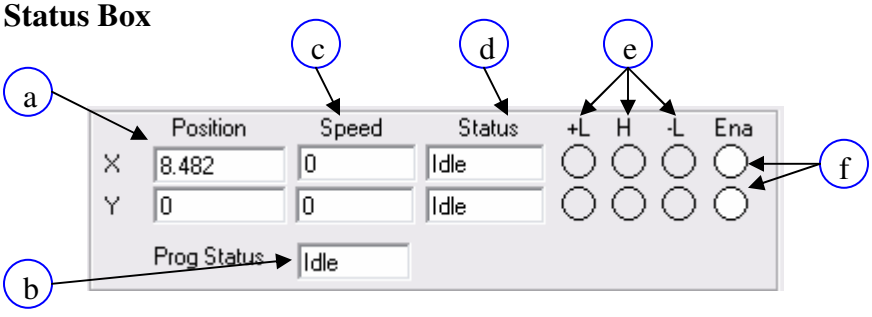


A. DXF Viewer Box



a. Displays the (X,Y) position of the mouse cursor within the DXF viewer box
b. Preview of the DXF file. For DXF preview to appear, click on "Load DXF File"
c. Z-axis cursor – Whenever the Z-axis is enabled, the cursor turns the color red. Otherwise the cursor is the color white

B. Status Box



a. Status of standalone program
    a. Idle – Program stopped
    b. Running – Program executing

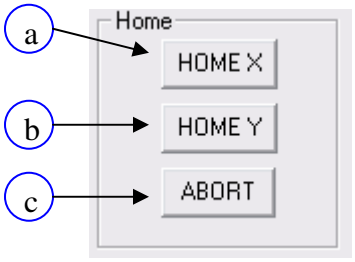
   c. Paused – Program paused
   d. Error – Program in error state

b. Current pulse position (X,Y axes)
c. Current speed (X,Y axes)
d. Motor status (X,Y axes)
   i. Idle – motor is not moving.
   ii. Accel – motor is accelerating
   iii. Const – motor is running in constant speed
   iv. Decel – motor is decelerating
   v. +LimError – plus limit error
   vi. –LimError – minus limit error
e. +Limit, -Limit, Home status
f. Enable status (X,Y axes). To enable/disable the axis, click on the corresponding circle

C. Digital Input/Output Box

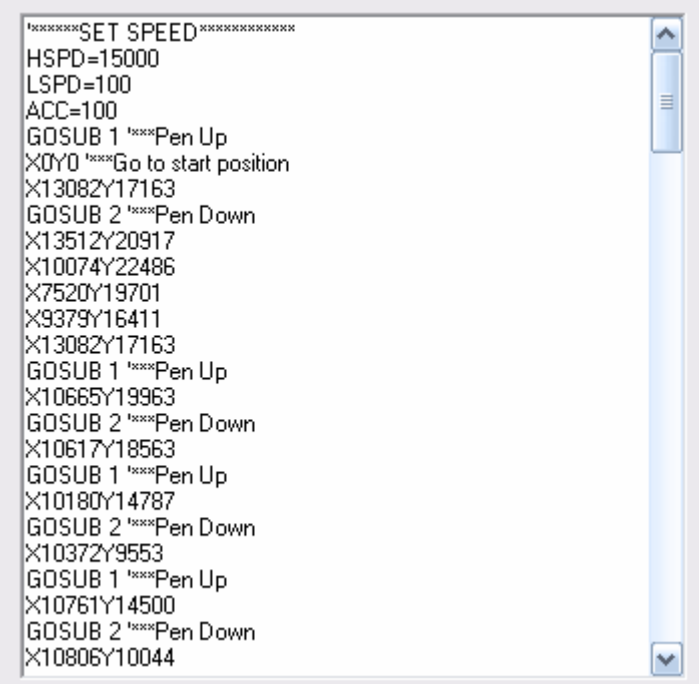
*See **Section D** of "Program and Control Software"*

D. Homing Box



a. Home x-axis to the negative direction
b. Home y-axis to the negative direction
c. Abort all movement

E. Motion Conversion Program Box



View DXF file code once it is converted to Arcus Technology text-based language. To populate this box, first select a DXF file by clicking on "Load DXF File", secondly click "Convert to Motion"

F. Compiler box

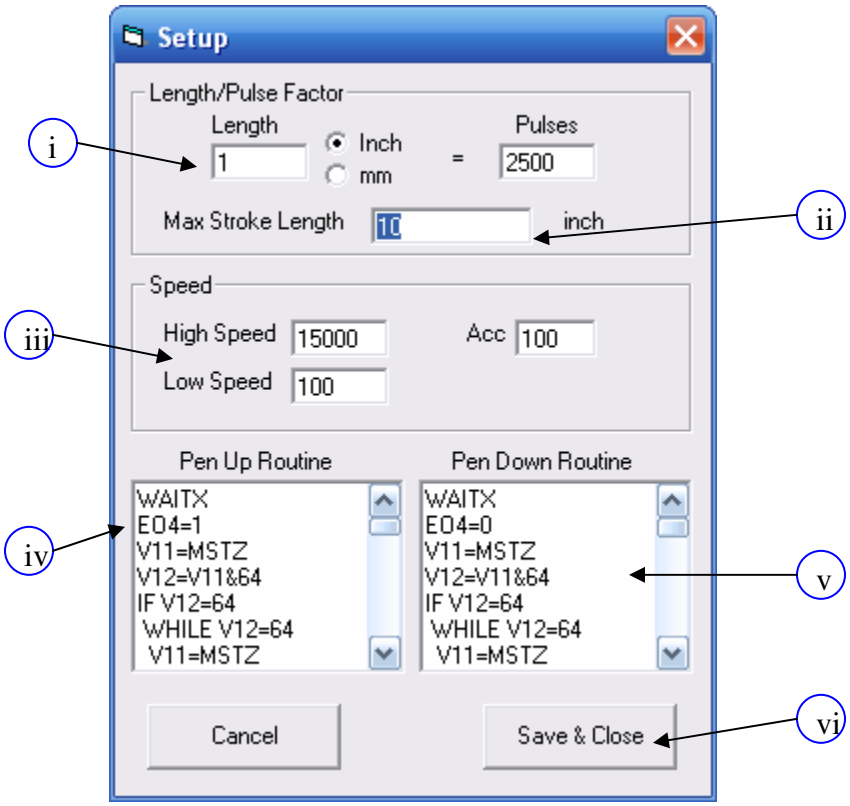
*See **Section J** of "Program and Control Software"*

G. Program Control box

*See **Section G** of "Program and Control Software"*

H. DXF Conversion Setup button

Setup parameters for DXF X,Y motion profile. After clicking on this button, the following screen will appear:



i. Length/Pulse Factor – Select relationship between number of pulses and length of movement in terms of inch or millimeter.

ii. Max Stroke Length – The largest allowable stroke length. This will affect the scaling of the DXF viewer box

iii. High Speed, Low Speed and Acceleration settings

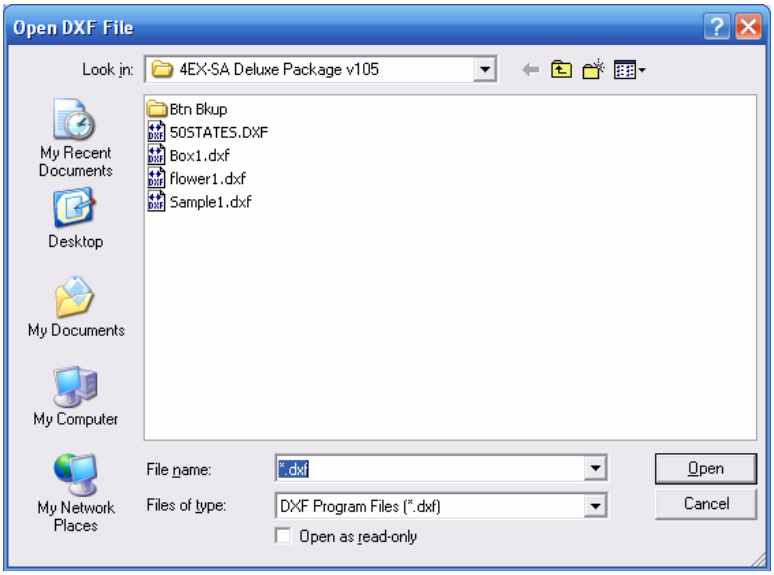
iv. Pen Up Routine – The routine when the XY axis is **not** in position

v. Pen Down Routine – The routine when the XY axis is in position

vi. Save parameters and exit setup

I. DXF Action Box

a. Once clicking on this button, the following screen will appear:



Select the desired DXF file and click "Open". At this point, the select DXF file will be previewed in the DXF Viewer box.

b. Convert the loaded DXF file into PMX-4EX-SA compatible motion commands. The result will be loaded into the Motion Conversion Program box

J. Program File Control box

*See **Section I** of "Program and Control Software"*

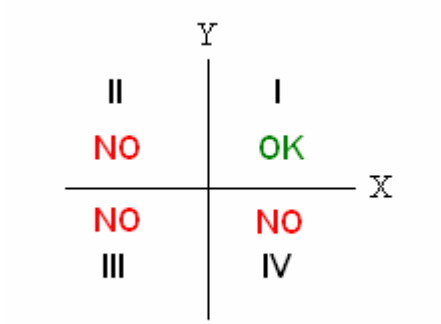
Important Notes for DXF Converter:

Creating a compatible DXF file:

Margins: Many times a DXF file may have extra text or margins describing the project. These should be removed. The only elements in the DXF file should be the picture that is desired to be drawn.

Radius Size: PMX-4EX-SA does not allow a radius larger than 46399 pulses on arc or circular moves. To keep your radius moves smaller than 46399, decrease the **Length/Pulse Factor**.

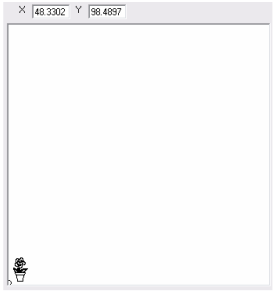
Picture positioning: A DXF file can not contain an any or part of an image that is not in quadrant I (i.e. all x,y positions of the DXF need to be positive). See figure below:



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

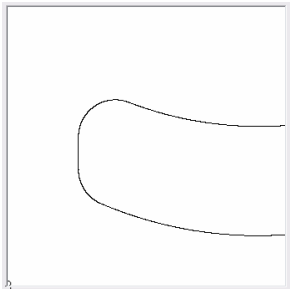
Scaling the DXF Viewer Box:

Sometimes when loading a DXF file, the picture may seem too small. See below:



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

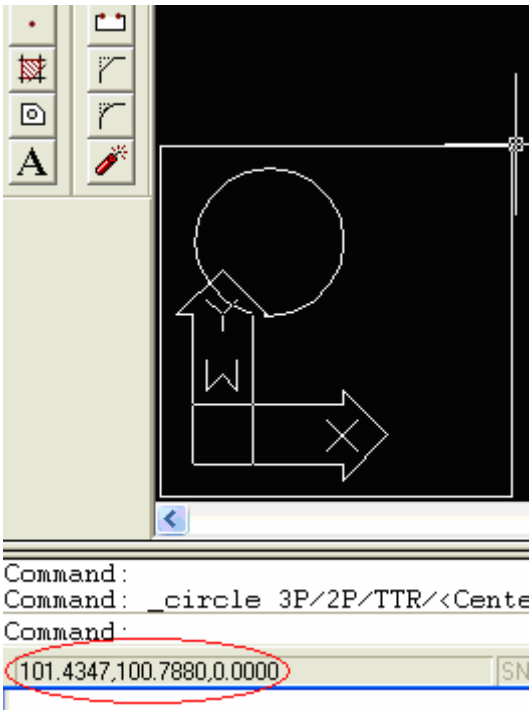
In the case where you do not see any picture or the picture is cut off, the window is zoomed in too much. See below:



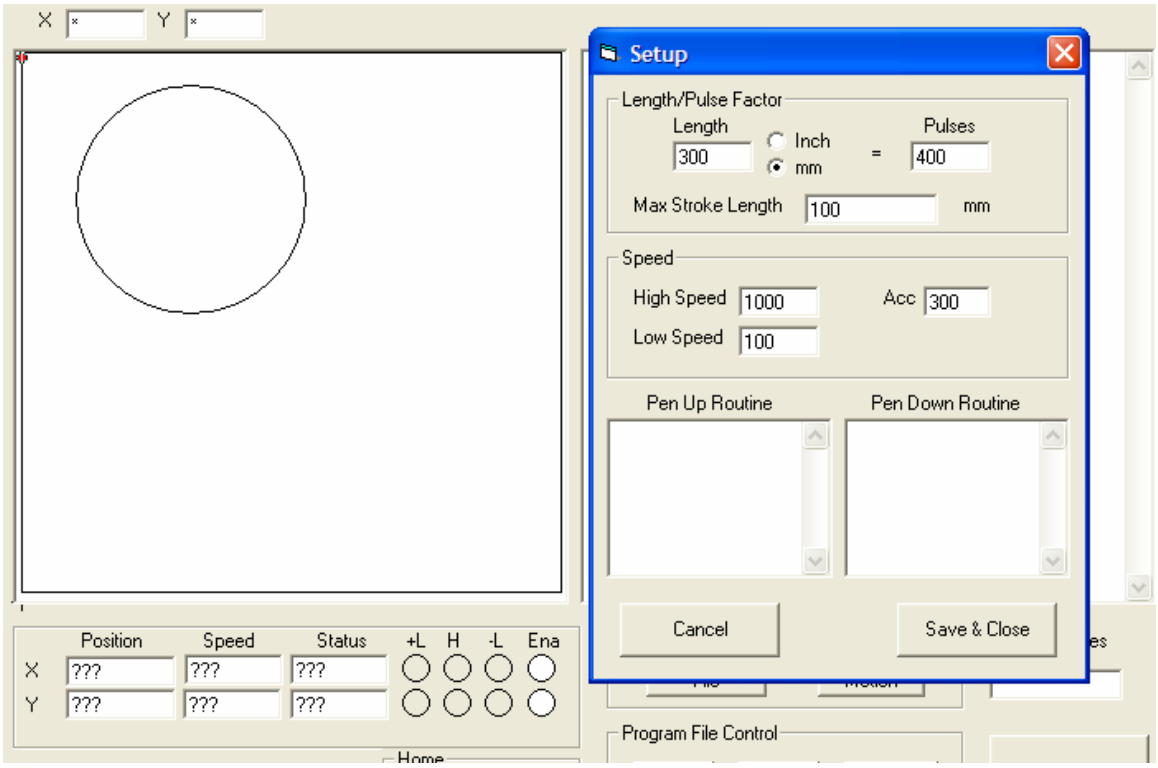
To zoom out, decrease the **Max Stroke Length** parameter.

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

For example, you can see in below in AutoCad drawing that the length and width of the picture is about 100 x 100 (circled in red). In this case, the units are mm.



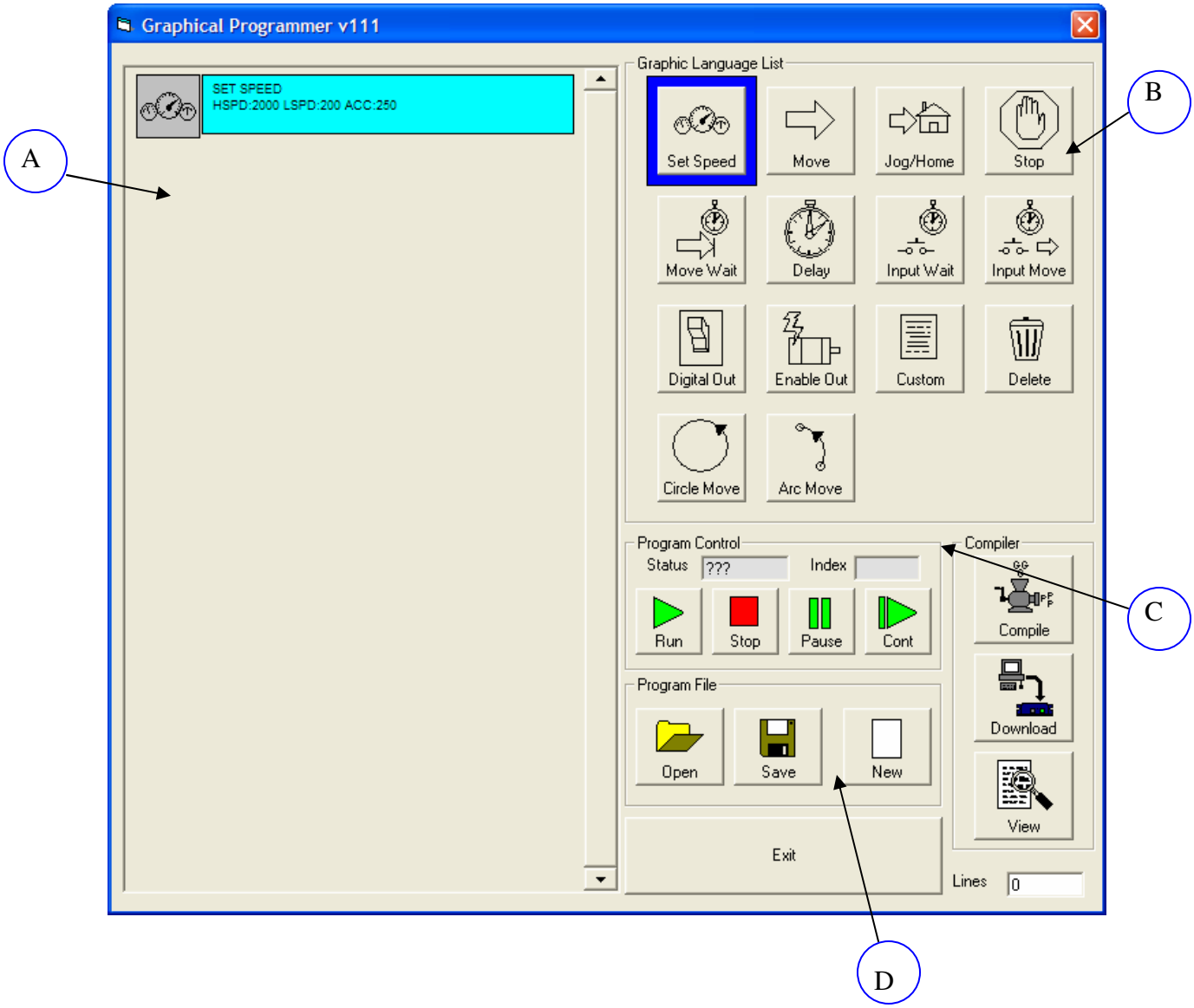
When loading the DXF, the **Max Stroke Length** should be set to 100 in order to properly show the picture. See below:



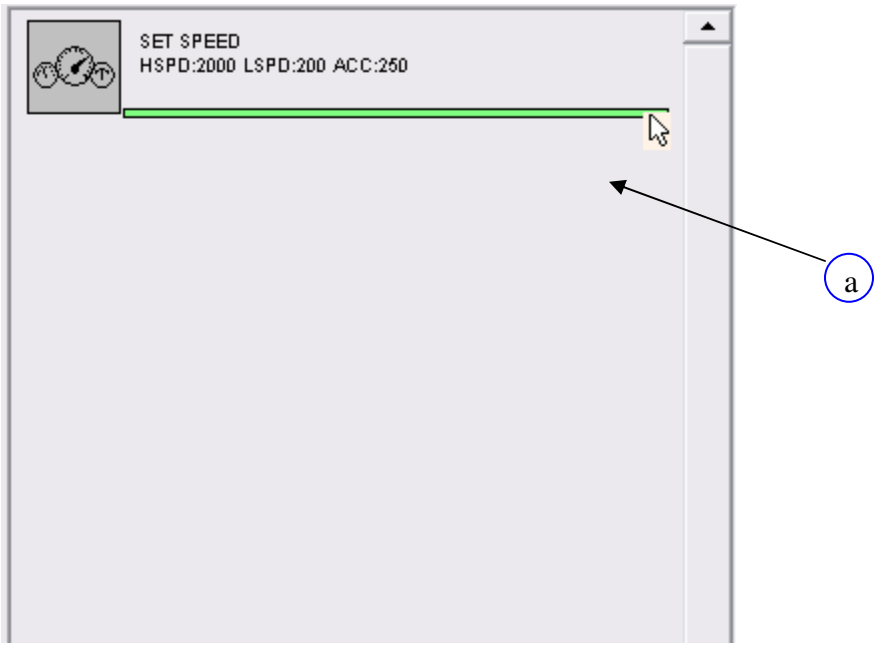
Scaling your XY table:

The scaling of your XY table will depend on the Length/Pulse Factor.

14. Graphical Programming Software



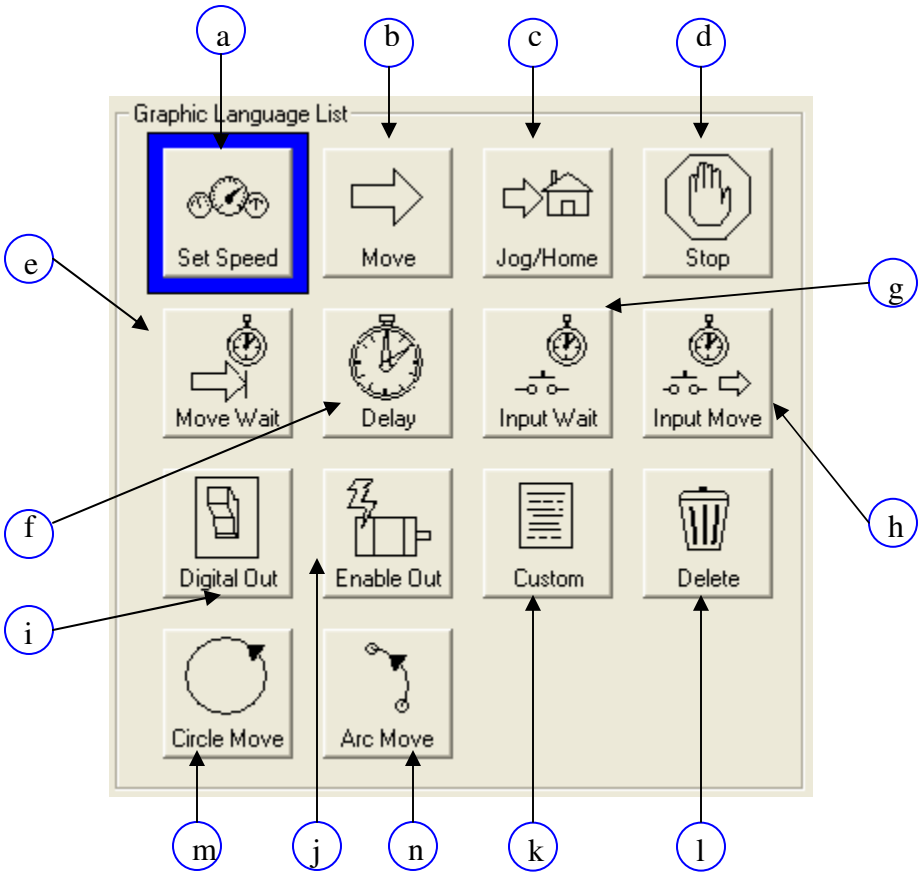
A. While Sequence Box



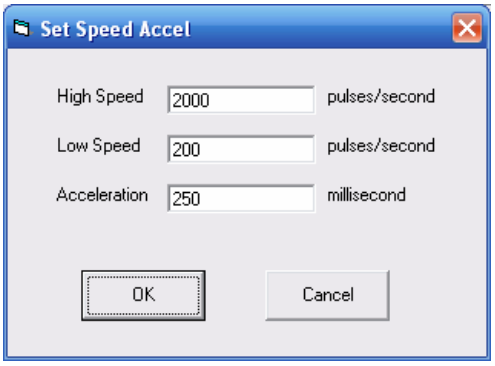
a. The following box contains the sequence that is executed in a continuous while loop. To enter things into the while loop, first click on an item in the Graphic Language List box and then move the cursor directly under the last item of the sequence (see above).

   Once this is done, a green line will appear. At this point, click on the green line to expose the graphic language items settable parameters.

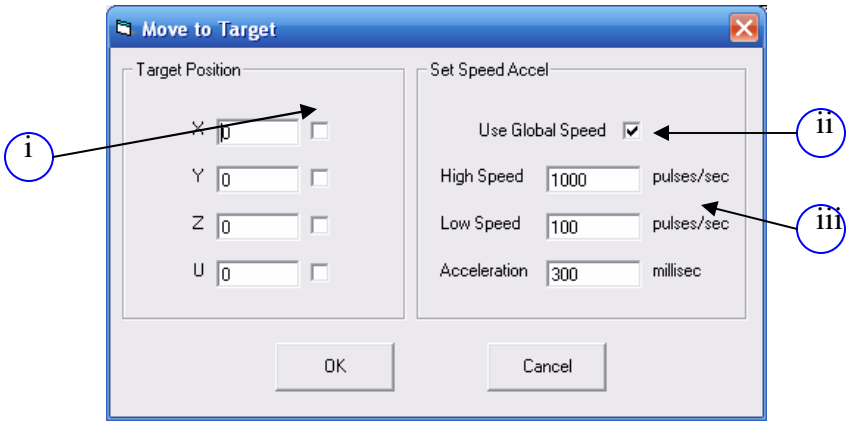
B. Graphic Language List Box



a. Set Speed Object: Set global high speed, low speed and acceleration settings. These speed settings will be used for all moves unless otherwise specified.

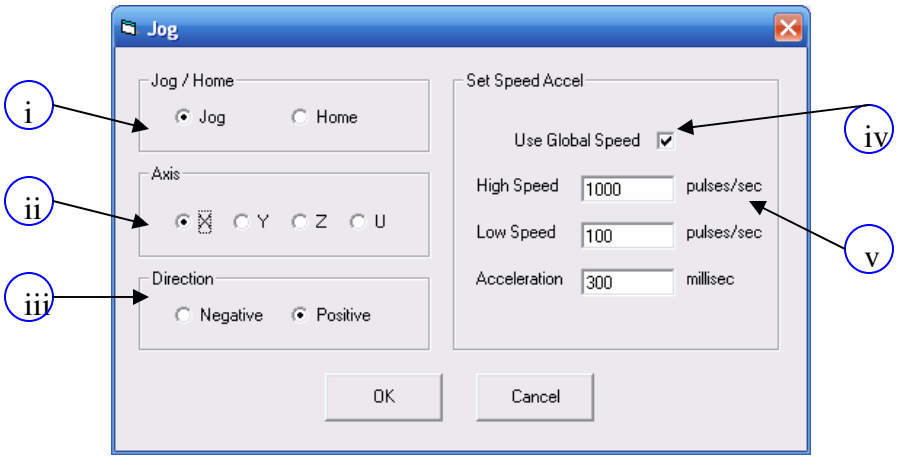


b. Move Object: Perform absolute move commands on selected axes



i. Select 1-4 axes to move. If more than one axis is selected, the move will be linear interpolated.
ii. Check to use global speeds specified in the Set Speed Object
iii. If "Use Global Speed" is not checked, the move will use the following local speed settings

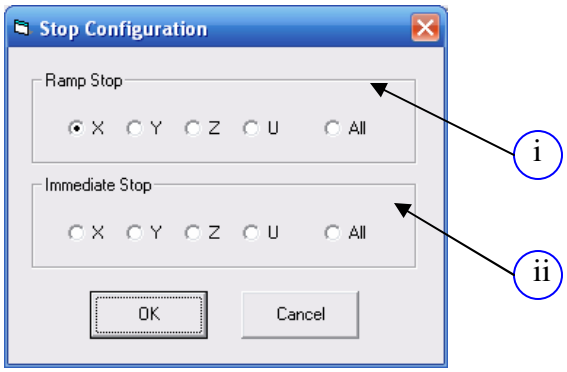
c. Jog/Home Object: Perform a plus/minus jog or home move for a single axis



i. Select this setting to be a jog or home move
ii. Select the axis to jog
iii. Select the direction of the move (i.e. "+" or "-")
iv. Check to use global speeds specified in the Set Speed Object

v. If “Use Global Speed” is not checked, the move will use the following local speed settings

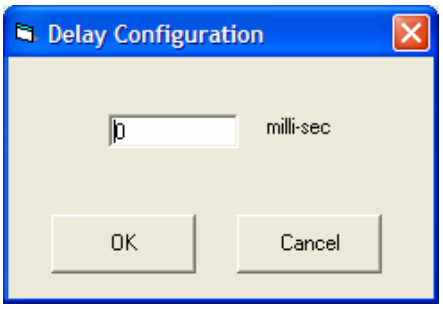
d. Stop Object: Perform a ramp stop or immediate stop on 1 or all axes



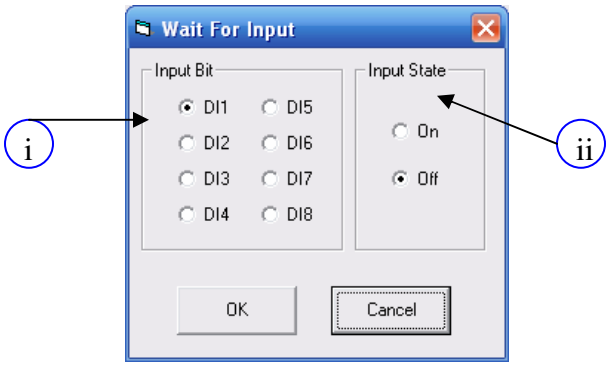
i. If this move is a ramp stop, select 1 or all axes
ii. If this moves is an immediate stop, select 1 or all axes

e. Move Wait Object: Wait until motion is done on a single axis until continuing to execute

f. Delay Object: Wait a set amount of time before continuing to execute. Units in milli-seconds.

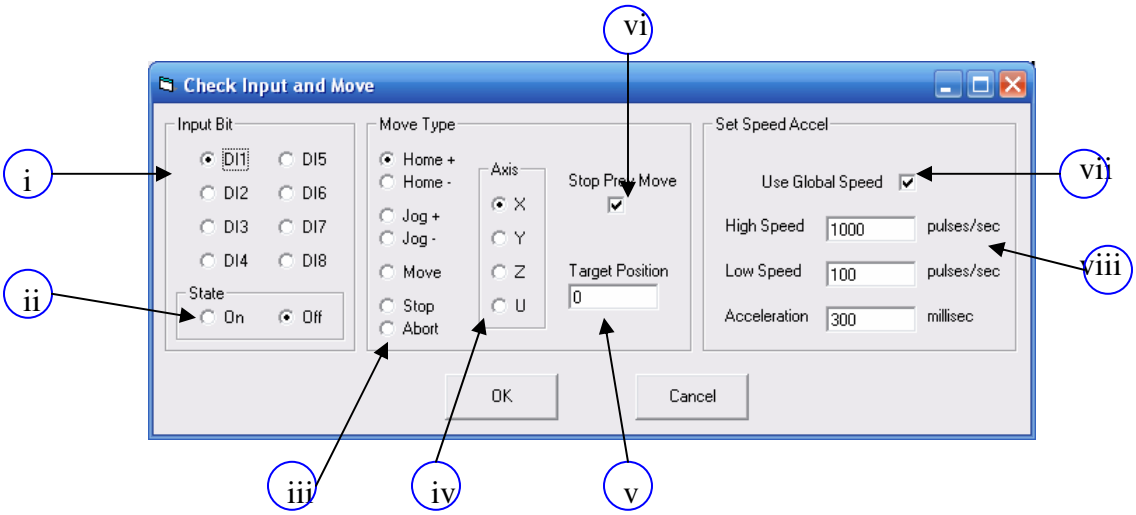


g. Input Wait Object: Wait until a single input is on/off before continuing to execute



i. Select the digital input
ii. Make the condition on or off

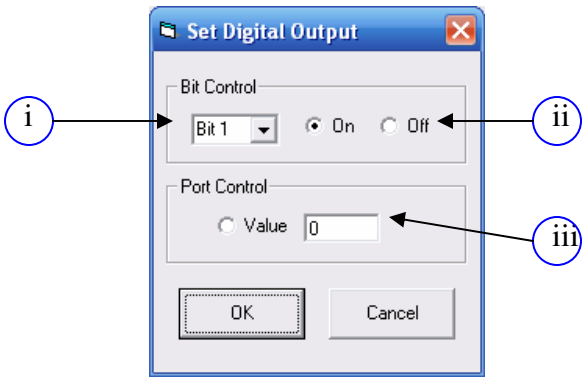
h. Input Move Object: Perform a move depending on a single digital input status



i. Select the digital input
ii. Make the condition on or off
iii. Select move type
iv. Select axis
v. Click to first stop the previous move before processing this setting
vi. Enter target position if “Move” type is selected
vii. Check to use global speeds specified in the Set Speed Object

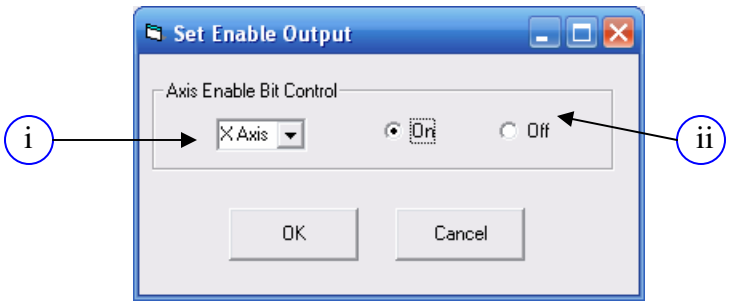
viii. If “Use Global Speed” is not checked, the move will use the following local speed settings

i. Digital Out Object: Set digital output status



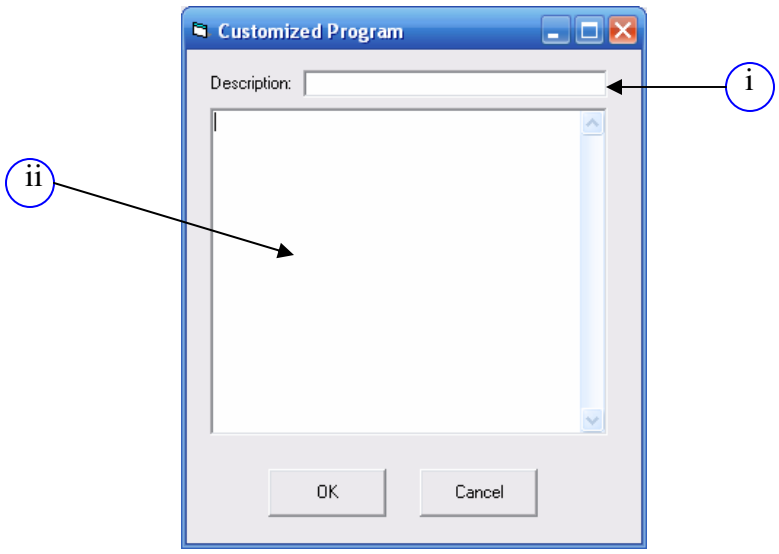
i. Select output bit
ii. Select on/off
iii. To set entire 8-bit output status, click the “Value” radio button and enter the 8-bit number in the field

j. Enable Out Object: Set enable output status for a single axis



i. Select output axis
ii. Select on/off state

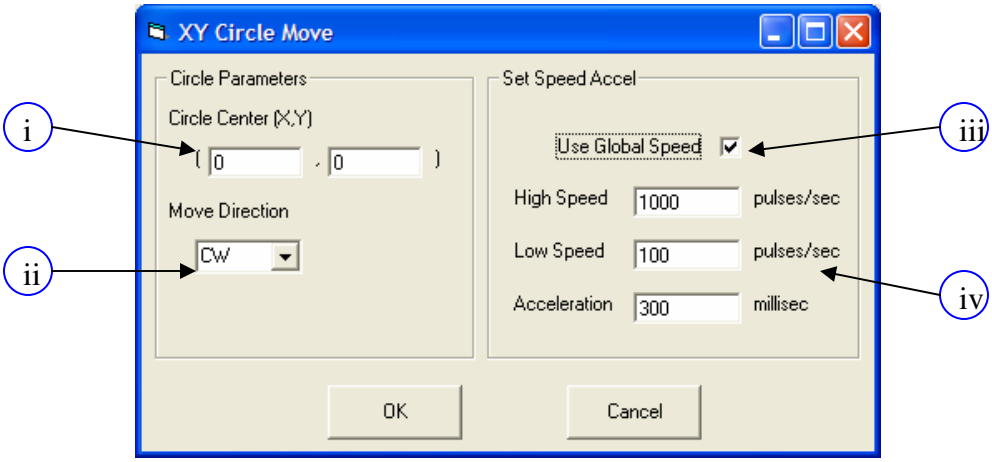
k. Custom: Write a custom program to insert into the sequence.



i. Description of program which will be displaying in the While Sequence
ii. Custom program text box. For details on programming language, see section 13 of manual.

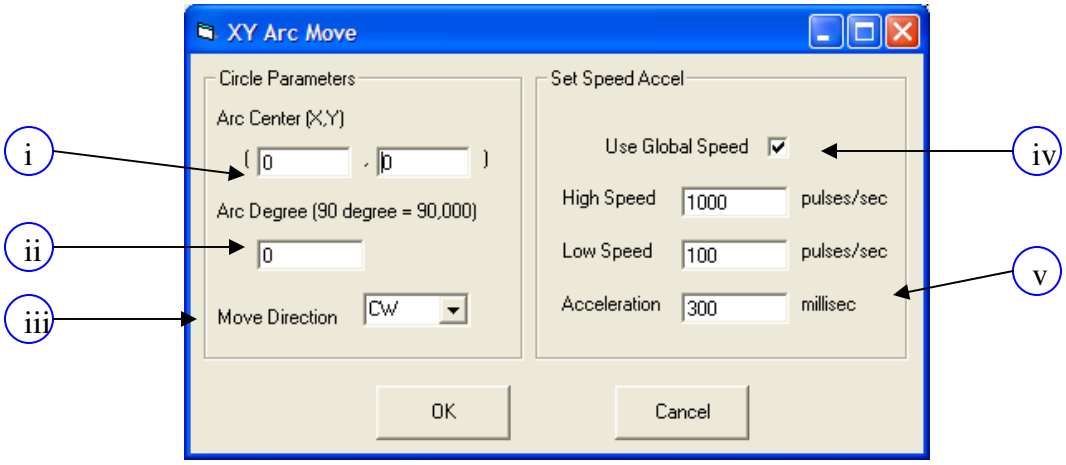
l. Delete: After clicking on this button. Clicking on any object in the While Sequence box will delete it.

m. Circle Move: Create a circle interpolation move



i. Center of the circle
ii. Draw circle in CW or CCW direction
iii. Check to use global speeds specified in the Set Speed Object
iv. If "Use Global Speed" is not checked, the move will use the following local speed settings

n. Arc Move: Create an arc interpolation move



i. Center of the circle
ii. Degree of arc drawn
iii. Draw arc in CW or CCW direction
iv. Check to use global speeds specified in the Set Speed Object
v. If "Use Global Speed" is not checked, the move will use the following local speed settings

C. Program Control Box

*See **Section G and J** of "Program and Control Software"*

D. Program File Box

*See **Section I** of "Program and Control Software"*

15. ASCII Language Specification

Invalid command is returned with ?(Error Message). Always check for proper reply when command is sent. Like the commands, all responses are in ASCII form.

Command	Description	Return
ABORT	Immediately stops all the motor if in motion. Abort turns off the buffered move.	OK
ABORTX ABORTY ABORTZ ABORTU	Immediately stops individual motor if in motion. Abort turns off the buffered move.	OK
ABS	Turns on absolute move mode	OK
ACC	Returns current global acceleration value in milliseconds.	
ACC=[Value]	Sets global acceleration value in milliseconds.	OK
ACCX ACCY ACCZ ACCU	Returns current individual acceleration value in milliseconds.	
ACCX=[value] ACCY=[value] ACCZ=[value] ACCU=[value]	Sets individual acceleration value in milliseconds.	OK
AI[1-8]	Get analog input status. Units in mV	[0-5000]
ARCP[X]:[Y]:[θ]	XY Arc interpolation move (CW direction)	OK
ARCN[X]:[Y]:[θ]	XY Arc interpolation move (CCW direction)	OK
BF	Disable buffered move	OK
BO	Enable buffer move on	OK
CIRP[X]:[Y]	XY Circular interpolation move (CW direction)	OK
CIRN[X]:[Y]	XY Circular interpolation move (CCW direction)	OK
CLRX CLRY CLRZ CLRU	Clears motor limit or alarm status bit.	OK
DB	Return baud rate	[1,2,3,4,5]
DB=[value]	Set baud rate 1 – 9600 bps 2 – 19200 bps 3 – 38400 bps 4 – 57600 bps 5 – 115200 bps	OK
DI	Returns 8 bits of general purpose digital input.	[0-255]
DI[1-8]	Returns bit status of general purpose digital input.	[0,1]
DO	Returns 8 bits of general purpose digital output value.	[0-255]
DO=[value]	Sets 8 bits of general purpose digital output.	OK
DO[1-8]	Returns bit of general purpose digital output value.	[0,1]
DO[1-8]=[value]	Sets bit of general purpose digital output.	OK
DN	Return device name	[4EX00-4EX99]
DN=[value]	Set device name. value must be in the range [4EX00, 4EX99]	OK
DXX DXY DXZ	Get StepNLoop delta value of axis	

DXU		
EO	Returns 4 bits of enable output value.	[0-15]
EO=[value]	Sets 4 bits of enable outputs.	OK
EO[1-4]	Returns bit of enable output value.	[0,1]
EO[1-4]=[value]	Set bit of enable outputs.	OK
EX=[value] EY=[value] EZ=[value] EU=[value]	Set encoder value of axis	OK
GS[SubNumber]	Call a defined subroutine	OK
HS	Returns global high speed setting	[1-6,000,000]
HS=[value]	Sets global high speed	OK
HSX HSY HSZ HSU	Returns individual high speed setting	[1-6,000,000]
HSX=[value] HSY=[value] HSZ=[value] HSU=[value]	Sets individual high speed	OK
HX[+/-][mode] HY[+/-][mode] HZ[+/-][mode] HU[+/-][mode]	Homes the motor in plus [+] or minus [-] direction using different homing mode.	OK
I[X axis]: [Y axis]: [Z axis]: [speed]	XYZ interpolated move. Target move values are separated by ':' character. Last value is the constant speed that will be used in the move.	OK
IACC	Get automatic acceleration during buffer interpolated move status	[0-1]
IACC=[0 or 1]	Set automatic acceleration during buffer interpolated move status	OK
INC	Turns on incremental move mode	OK
JE	Get joystick enable status	[0-15]
JE=[value]	Set joystick enable status	OK
JX[+/-] JY[+/-] JZ[+/-] JU[+/-]	Jogs the motor in plus [+] or minus [-] direction.	OK
JV[1-12]	Get joystick speed, delta and tolerance settings	
JV[1-12]=[value]	Set joystick speed, delta and tolerance settings	OK
JL[1-16]	Get joystick soft limit settings	
JL[1-16=[value]	Set joystick soft limit settings	OK
LS	Returns global low speed setting	[1-6,000,000]
LS=[value]	Sets global low speed	OK
LSX LSY LSZ LSU	Returns individual low speed setting	[1-6,000,000]
LSX=[value] LSY=[value] LSZ=[value]	Sets individual low speed	OK

LSU=[value]		
MST	Returns all motor status and buffer move status Motor Status Bit 0 – accelerating Bit 1 – decelerating Bit 2 – constant speed Bit 3 – alarm input Bit 4 - + limit input Bit 5 - -limit input Bit 6 – home input Bit 7 - + limit error Bit 8 - - limit error Bit 9 – alarm error	[X motor status]: [Y motor status]: [Z motor status]: [U motor status]: [Buffer enabled]: [Buffer start]: [Buffer end]: [Available Buffer]: [MoveMode]:
PE	Returns current encoder counter values of all 4 axes	[X Enc Position]: [Y Enc Position]: [Z Enc Position]: [U Enc Position]
POX POY POZ POU	Returns polarity setup Bit 0 – home input polarity Bit 1 – alarm polarity Bit 2 – limit polarity (X axis control polarity of all limits)	[0-7]
POX=[value] POY=[value] POZ=[value] POU=[value]	Sets polarity Bit 0 – home input polarity Bit 1 – alarm polarity Bit 2 – limit polarity (X axis control polarity of all limits)	OK
PP	Returns current pulse counter values of all 4 axes	[X Pulse Position]: [Y Pulse Position]: [Z Pulse Position]: [U Pulse Position]
PS	Returns current pulse speed values of all 4 axes	[X Speed]: [Y Speed]: [Z Speed]: [U Speed]
PX=[value] PY=[value] PZ=[value] PU=[value]	Set position value of axis	OK
SCVX SCVY SCVZ SCVU	Returns the s-curve control	[0,1]
SCVX=[0 or 1] SCVY=[0 or 1] SCVZ=[0 or 1] SCVU=[0 or 1]	Enable or disable s-curve. If disabled, trapezoidal acceleration/ deceleration will be used.	OK
SLAX SLAY SLAZ SLAU	Get StepNLoop maximum attempt value of axis	
SLAX=[value] SLAY=[value] SLAZ=[value] SLAU=[value]	Set StepNLoop maximum attempt value of axis	OK
SLEX	Get StepNLoop error range value of axis	

SLEY SLEZ SLEU		
SLEX=[value] SLEY=[value] SLEZ=[value] SLEU=[value]	Set StepNLoop error range value of axis	OK
SLRX SLRY SLRZ SLRU	Get StepNLoop ratio of axis (ppr / cpr)	[0.001-999.999]
SLRX=[value] SLRY=[value] SLRZ=[value] SLRU=[value]	Set StepNLoop ratio of axis (ppr / cpr)	OK
SLSX SLSY SLSZ SLSU	Get StepNLoop status of axis	[0-12]
SLTX SLTY SLTZ SLTU	Get StepNLoop tolerance of axis	
SLTX=[value] SLTY=[value] SLTZ=[value] SLTU=[value]	Set StepNLoop tolerance of axis	OK
SLOAD	Returns RunOnBoot parameter	[0,1]
SLOAD=[0 or 1]	0 – Do NOT run standalone program on boot up 1 – Run standalone program on boot up	OK
SR=[Value]	Control standalone program: 0 – Stop standalone program 1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program	OK
SPC	Get program counter for standalone program	[0-3229]
SASTAT	Get standalone program status 0 – Stopped 1 – Running 2 – Paused 4 – In Error	[0-4]
SA[LineNumber]	Get standalone line	Single line of compiled code
SA[LineNumber]=[Value]	Set standalone line	OK
SSPDX=[value] SSPDY=[value] SSPDZ=[value] SSPDU=[value]	PMX on-the-fly speed change. In order to use this command on a certain axis, S-curve control must be disabled for the corresponding axis. Use SCV[axis] command to enable and disable s-curve acceleration/ deceleration control.	OK
SSPDMX SSPDMY SSPDMZ SSPDMU	Get on-the-fly speed change mode for each axis	[0-7]

SSPDMX=[value] SSPDMY=[value] SSPDMZ=[value] SSPDMU=[value]	Set on-the-fly speed change mode for each axis.	OK
STOP	Performs ramp down to low speed and stop if the motor is moving. (All axes)	OK
STOPX STOPY STOPZ STOPU	Performs ramp down to low speed and stop if the motor is moving. (Individual axis)	OK
STORE	Store parameters to flash	OK
SYNXC SYNYC SYNZC SYNUC	Read sync output configuration for each axis 1 – trigger when encoder equals position 2 – trigger when encoder is greater than position 3 – trigger when encoder is less than position	[1-3]
SYNXC= SYNYC= SYNZC= SYNUC=	Set sync output configuration for each axis 1 – trigger when encoder equals position 2 – trigger when encoder is greater than position 3 – trigger when encoder is less than position	OK
SYNXF SYNYF SYNZF SYNUF	Turn of sync output for each axis	OK
SYNXO SYNYO SYNZO SYNUO	Turn on sync output for each axis	OK
SYNXP SYNYP SYNZP SYNUP	Get trigger position for each axis	28 bit signed number
SYNXP= SYNYP= SYNZP= SYNUP=	Set trigger position for each axis	28 bit signed number
SYNXT SYNYT SYNZT SYNUT	Get pulse width time (ms). Only applicable if sync output configuration is set to 1.	[0-10]
SYNXT= SYNYT= SYNZT= SYNUT=	Set pulse width time (ms). Only applicable if sync output configuration is set to 1.	OK
TR	Get timer register value	[0-1,000,000]
TR=	Set timer register value (ms)	OK
V[VarNumber]	Get standalone variable value VarNumber: [0-63]	Variable number
V[VarNumber]=[Value]	Write standalone variable value VarNumber:[0-63]	OK
VER	Returns controller software version	V[#]
X[target X] Y[target Y] Z[target Z] U[target U]	Individual move command	OK

16. Standalone Language Specification

Version 1.21

;

Description:

Comment notation. In programming, comment must be in its own line.

Syntax:

; [Comment Text]

Examples:

```
; ***This is a comment
JOGX+           ;***Jogs X axis to positive direction
DELAY=1000      ;***Wait 1 second
ABORT           ;***Stop immediately all axes including X axis
```

ABORT

Description:

Motion: Immediately stops all axes if in motion without deceleration.

Syntax:

ABORT

Examples:

```
JOGX+           ;***Jogs X axis to positive direction
DELAY=1000      ;***Wait 1 second
ABORT           ;***Stop immediately all axes including X axis
```

ABORT[axis]

Description:

Motion: Immediately stops individual axis without deceleration.

Syntax:

ABORT[axis]

Examples:

```
JOGX+          ;***Jogs X axis to positive direction
JOGY+          ;***Jogs Y axis to positive direction
JOGZ+          ;***Jogs Z axis to positive direction
```

ABS

Description:

Motion: Changes all move commands to absolute mode.

Syntax:

ABS

Examples:

```
ABS            ;***Change to absolute mode
PX=0           ;***Change X position to 0
X1000          ;***Move X axis to position 1000
X2000          ;***Move X axis to position 2000
ABORT          ;***Stop immediately all axes including X axis
```

ACC

Description:

Read: Get acceleration value
Write: Set acceleration value.

Value is in milliseconds.
Range is from 1 to 10,000.

Syntax:

Read: [variable] = ACC
Write: ACC = [value]
ACC = [variable]

Conditional: IF ACC=[variable]
ENDIF

IF ACC=[value]
ENDIF

Examples:

```
ACC=300     ;***Sets the acceleration to 300 milliseconds
V3=500      ;***Sets the variable 3 to 500
ACC=V3      ;***Sets the acceleration to variable 3 value of 500
```

ACC[axis]

Description:

Read: Get individual acceleration value
Write: Set individual acceleration value.

Value is in milliseconds.
Range is from 1 to 10,000.

Syntax:

Read: [variable] = ACC[axis]
Write: ACC[axis] = [value]
ACC[axis] = [variable]

Conditional: IF ACC[axis]=[variable]
ENDIF

IF ACC[axis]=[value]
ENDIF

Examples:

```
ACCX=300    ;***Sets the X acceleration to 300 milliseconds
V3=500      ;***Sets the variable 3 to 500
ACCX=V3     ;***Sets the X acceleration to variable 3 value of 500
```

ARC

Description:

Motion: Perform arc move using X and Y axis.

Specify clockwise or counter-clockwise, center location, and the angle.
Angle is in whole number in thousandth. For example, 45 degrees is 45,000.

Syntax:
ARC[P for clockwise, N for counter-clockwise][Center X]:[Center Y]:[Angle]

Examples:

```
ARCP0:100:30000     ;***Using  X0, Y100 perform arc move to
                    ;***30 degrees from center (CW)
ARCN0:100:30000     ;***Using  X0, Y100 perform arc move to
                    ;***30 degrees from center (CCW)
```

CIR

Description:
Motion: Perform circle move using X and Y axis.

Specify clockwise or counter-clockwise and the center location.

Syntax:
CIR[P for clockwise, N for counter-clockwise][Center X]:[Center Y]

Examples:

```
CIRP1000:1000 ;***Using  X 1000 and Y 1000 perform circular move  (CW)

CIRN0:2000      ;***Using  X 0 and Y 2000 perform circular move (CCW)
```

DELAY

Description:
Set a delay (1 ms units)

Syntax:
Delay=[Number] (1 ms units)

Examples:

```
JOGX+               ;***Jogs X axis to positive direction
DELAY=10000         ;***Wait 10 second
ABORT               ;***Stop with deceleration all axes including X axis
EX=0                ;***Sets the current X encoder position to 0
```

```
EY=0            ;***Sets the current Y encoder position to 0
EZ=0            ;***Sets the current Z encoder position to 0
EU=0            ;***Sets the current U encoder position to 0
```

DI

Description:

Read: Gets the digital input value

Performax 4EX has 8 digital inputs

Syntax:

Read: [variable] = DI

Conditional: IF DI=[variable]
ENDIF

IF DI=[value]
ENDIF

Examples:

```
IF DI=255
    DO=1        ;***If no digital inputs are triggered, set DO=1
ENDIF
```

DI[1-8]

Description:

Read: Gets the digital input value

Performax 4EX has 8 digital inputs

Syntax:

Read: [variable] = DI[1-8]

Conditional: IF DI[1-8]=[variable]
ENDIF

IF DI[1-8]=[0 or 1]
ENDIF

Examples:

```
IF DI1=1
    DO=1        ;***If digital input 1 is triggered, set DO=1
ENDIF
```

DO

Description:

Read: Gets the digital output value
Write: Sets the digital output value

Performax 4EX has 8 digital outputs

Syntax:

Read: [variable] = DO
Write: DO = [value]
DO = [variable]

Conditional: IF DO=[variable]
ENDIF

IF DO=[value]
ENDIF

Examples:

```
DO=7        ;***Turn first 3 bits on and rest off
```

DO[1-8]

Description:

Read: Gets the individual digital output value
Write: Sets the individual digital output value

Performax 4EX has 8 digital outputs

Syntax:

Read: [variable] = DO[1-8]
Write: DO[1-8] = [0 or 1]
DO[1-8] = [variable]

Conditional: IF DO[1-8]=[variable]
ENDIF

IF DO[1-8]=[0 or 1]
ENDIF

Examples:

```
DO7=1        ;***Turn DO7 on
DO6=1        ;***Turn DO6 on
```

E[axis]

Description:

Read: Gets the current encoder position
Write: Sets the current encoder position

Syntax:

Read: [variable] = E[axis]
Write: E[axis] = [0 or 1]
E[axis] = [variable]

Conditional: IF E[axis]=[variable]
ENDIF

IF E[axis]=[value]
ENDIF

Examples:

```
JOGX+            ;***Jogs X axis to positive direction
DELAY=1000       ;***Wait 1 second
ABORT            ;***Stop with deceleration all axes including X axis
EX=0             ;***Sets the current X encoder position to 0
EY=0             ;***Sets the current Y encoder position to 0
EZ=0             ;***Sets the current Z encoder position to 0
EU=0             ;***Sets the current U encoder position to 0
```

ECLEAR[axis]

Description:

Write: Clears error status

Syntax:

Write: ECLEAR[axis]

Examples:

```
ECLEARX          ;***Clears error of axis X
ECLEARY          ;***Clears error of axis Y
ECLEARZ          ;***Clears error of axis Z
ECLEARU          ;***Clears error of axis U
```

ELSE

Description:

Perform ELSE condition check as a part of IF statement

Syntax:

ELSE

Examples:

```
IF V1=1
    X1000        ;***If V1 is 1, then move to 1000
ELSE
    X-1000       ;***If V1 is not 1, then move to -1000
ENDIF
```

ELSEIF

Description:

Perform ELSEIF condition check as a part of the IF statement

Syntax:

ELSEIF [Argument 1] [Comparison] [Argument 2]

[Argument] can be any of the following:

- Numerical value
- Pulse or Encoder Position
- Digital Output
- Digital Input
- Enable Output
- Motor Status

[Comparison] can be any of the following

=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!=	Not Equal to

Examples:

```
IF V1=1
        X1000
ELSEIF V1=2
        X2000
ELSEIF V1=3
        X3000
ELSE
        X0
ENDIF
```

END

Description:

Indicate end of program.
Program status changes to idle when END is reached.

Note: Subroutine definitions should be written AFTER the END statement

Syntax:

END

Examples:

```
X0
X1000
END
```

ENDIF

Description:

Indicates end of IF operation

Syntax:

ENDIF

Examples:

```
IF V1=1
        X1000
ENDIF
```

ENDSUB

Description:

Indicates end of subroutine
When ENDSUB is reached, the program returns to the previously called subroutine.

Syntax:

ENDSUB

Examples:

```
GOSUB 1
END

SUB 1
    X0
    X1000
ENDSUB
```

ENDWHILE

Description:

Indicate end of WHILE loop

Syntax:

ENDWHILE

Examples:

```
WHILE V1=1          ;***While V1 is 1 continue to loop
    X0
    X1000
ENDWHILE            ;***End of while loop so go back to WHILE
```

EO

Description:

Read: Gets the enable output value
Write: Sets the enable output value

Performax 4EX has 4 enable outputs.

Syntax:

Read: [variable] = EO
Write: EO = [value]
EO = [variable]

Conditional: IF EO=[variable]
ENDIF

IF EO=[value]
ENDIF

Examples:

```
EO=3                ;***Turn first 2 bits of enable outputs

IF V1=1
    EO=V2           ;***Enable output according to variable 2
ENDIF
```

EO[1-4]

Description:

Read: Gets the individual enable output value
Write: Sets the individual enable output value

Performax 4EX has 4 enable outputs.

Syntax:

Read: [variable] = EO[1-4]
Write: EO[1-4] = [0 or 1]
EO[1-4] = [variable]

Conditional: IF EO=[variable]
ENDIF

IF EO=[value]
ENDIF

Examples:

```
EO1=31              ;***Turn enable output 1 on

IF V1=1
    EO2=V2          ;***Enable output 2 according to variable 2
ENDIF
```

GOSUB

Description:

Perform go to subroutine operation
Subroutine range is from 1 to 32.

Note: Subroutine definitions should be written AFTER the END statement

Syntax:

GOSUB [subroutine number]

[Subroutine Number] range is 1 to 32

Examples:

```
GOSUB 1
END

SUB 1
    X0
    X1000
ENDSUB
```

HOME[axis][+ or -]

Description:

Command: Perform homing using current high speed, low speed, and acceleration.

Syntax:

HOME[Axis][+ or -]

Examples:

```
HOMEX+      ;***Homes X axis in positive direction

HOMEZ-      ;***Homes Z axis in negative direction
```

HSPD

Description:

Read: Gets high speed. Value is in pulses/second
Write: Sets high speed. Value is in pulses/second.

Range is from 1 to 6,000,000.

Syntax:

Read: [variable] = HSPD
Write: HSPD = [value]
HSPD = [variable]

Conditional: IF HSPD=[variable]
ENDIF

IF HSPD=[value]
ENDIF

Examples:

```
HSPD=10000 ;***Sets the high speed to 10,000 pulses/sec

V1=2500    ;***Sets the variable 1 to 2,500
HSPD=V1    ;***Sets the high speed to variable 1 value of 2500
```

HSPD[axis]

Description:

Read: Gets individual high speed. Value is in pulses/second
Write: Sets individual high speed. Value is in pulses/second.

Range is from 1 to 6,000,000.

Syntax:

Read: [variable] = HSPD[axis]
Write: HSPD[axis] = [value]
HSPD[axis] = [variable]

Conditional: IF HSPD[axis]=[variable]
ENDIF

IF HSPD[axis]=[value]
ENDIF

Examples:

```
HSPDY=10000        ;***Sets the Y high speed to 10,000 pulses/sec
V1=2500            ;***Sets the variable 1 to 2,500
HSPDY=V1           ;***Sets the Y high speed to variable 1 value of 2500
```

IF

Description:

Perform IF condition check

Syntax:

IF [Argument 1] [Comparison] [Argument 2]

[Argument] can be any of the following:

- Numerical value
- Pulse or Encoder Position
- Digital Output
- Digital Input
- Enable Output
- Motor Status

[Comparison] can be any of the following

=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!=	Not Equal to

Examples:

```
IF V1=1
    X1000
ENDIF
```

INC

Description:

Command: Changes all move commands to incremental mode.

Syntax:

INC

Examples:

```
ABS             ;***Change to absolute mode
PX=0            ;***Change X position to 0
X1000           ;***Move X axis to position 1000 (0+1000)
X2000           ;***Move X axis to position 3000 (1000+2000)
ABORT           ;***Stop immediately all axes including X axis
```

JOG[axis]

Description:

Command: Perform jogging using current high speed, low speed, and acceleration.

Syntax:

JOG[Axis][+ or -]

Examples:

```
JOGX+       ;***Jogs X axis in positive direction

JOGY-       ;***Jogs Y axis in negative direction
```

JOYENA

Description:
Write: Enable joystick feature for axis
Range is from 0 or 15

Syntax:
Write: JOYENA=[value]

Examples:
JOYENA=1 ;***Enable joystick feature on X axis only
JOYENA=3 ;***Enable joystick feature on X and Y axis

JOYHS[axis]

Description:
Write: Set high speed setting for joystick control

Syntax:
Write: JOYHS[axis] = [value]
JOYHS[axis] = [variable]

Examples:
JOYHSX=10000 ;***High speed of X axis is set to 10,000 pps
JOYHSU=20000 ;***High speed of U axis is set to 20,000 pps

JOYDEL[axis]

Description:
Write: Set maximum delta value of change in speed for joystick control

Syntax:
Write: JOYDEL[axis] = [value]
JOYDELaxis] = [variable]

Examples:
JOYDELX=100 ;***Speed delta of X axis is set to 100 pps
JOYDELU=200 ;***Speed delta of Y axis is set to 200 pps

JOYNO[axis]

Description:

Write: Set negative outer limit for joystick control

Syntax:

Write: JOYNO[axis] = [value]
JOYNO[axis] = [variable]

Examples:

```
JOYNOX=-10000     ;*** negative outer limit of x-axis set to -10000
JOYNIX=-9000      ;*** negative inner limit of x-axis set to -9000
JOYPIX=9000       ;*** positive inner limit of x-axis set to 9000
JOYPOX=10000      ;*** positive outer limit of x-axis set to 10000
```

JOYNI[axis]

Description:

Write: Set negative inner limit for joystick control

Syntax:

Write: JOYNI[axis] = [value]
JOYNI[axis] = [variable]

Examples:

```
JOYNOX=-10000     ;*** negative outer limit of x-axis set to -10000
JOYNIX=-9000      ;*** negative inner limit of x-axis set to -9000
JOYPIX=9000       ;*** positive inner limit of x-axis set to 9000
JOYPOX=10000      ;*** positive outer limit of x-axis set to 10000
```

JOYPI[axis]

Description:

Write: Set positive inner limit for joystick control

Syntax:

Write: JOYPI[axis] = [value]
JOYPI[axis] = [variable]

Examples:

```
JOYNOX=-10000     ;*** negative outer limit of x-axis set to -10000
JOYNIX=-9000      ;*** negative inner limit of x-axis set to -9000
JOYPIX=9000       ;*** positive inner limit of x-axis set to 9000
JOYPOX=10000      ;*** positive outer limit of x-axis set to 10000
```

JOYPO[axis]

Description:

Write: Set positive outer limit for joystick control

Syntax:

Write: JOYPO[axis] = [value]
JOYPO[axis] = [variable]

Examples:

```
JOYNOX=-10000     ;*** negative outer limit of x-axis set to -10000
JOYNIX=-9000      ;*** negative inner limit of x-axis set to -9000
JOYPIX=9000       ;*** positive inner limit of x-axis set to 9000
JOYPOX=10000      ;*** positive outer limit of x-axis set to 10000
```

LSPD

Description:

Read: Get low speed. Value is in pulses/second.
Write: Set low speed. Value is in pulses/second.

Range is from 1 to 6,000,000.

Syntax:

Read: [variable]=LSPD
Write: LSPD=[long value]
LSPD=[variable]

Conditional:

```
IF LSPD=[variable]
ENDIF

IF LSPD=[value]
ENDIF
```

Examples:

```
LSPD=1000    ;***Sets the start low speed to 1,000 pulses/sec

V1=500       ;***Sets the variable 1 to 500
LSPD=V1      ;***Sets the start low speed to variable 1 value of 500
```

LSPD[axis]

Description:

Read: Get individual low speed. Value is in pulses/second.
Write: Set individual low speed. Value is in pulses/second.

Range is from 1 to 6,000,000.

Syntax:

Read: [variable]=LSPD[axis]
Write: LSPD[axis]=[long value]
LSPD[axis]=[variable]

Conditional: IF LSPD[axis]=[variable]
ENDIF

IF LSPD[axis]=[value]
ENDIF

Examples:

```
LSPDZ=1000 ;***Sets the Z low speed to 1,000 pulses/sec

V1=500     ;***Sets the variable 1 to 500
LSPDZ=V1   ;***Sets the Z low speed to variable 1 value of 500
```

MST

Description:

Command: Get motor status of axis

Syntax:

MST[Axis]

Examples:

```
IF MSTX=0
    DIO=6
ELSEIF MSTY=0
    DIO=3
ELSEIF MSTZ=0
    DIO=2
ELSEIF MSTU=0
    DIO=1
ENDIF
```

P[axis]

Description:

Read: Gets the current pulse position
Write: Sets the current pulse position

Syntax:

Read: Variable = P[axis]
Write: P[axis] = [value]
P[axis] = [variable]

Conditional: IF P[axis]=[variable]
ENDIF

IF P[axis]=[value]
ENDIF

Examples:

```
JOGX+           ;***Jogs X axis to positive direction
DELAY=1000      ;***Wait 1 second
ABORT           ;***Stop with deceleration all axes including X axis
PX=0            ;***Sets the current pulse position to 0
```

PS[axis]

Description:

Read: Get the current pulse position of an axis

Syntax:

Read: Variable = PS[Axis]

Conditional: IF PS[axis]=[variable]
ENDIF

IF PS[axis]=[value]
ENDIF

Examples:

```
JOGX+           ;***Jogs X axis to positive direction
DELAY=1000      ;***Wait 1 second
ABORT           ;***Stop with deceleration all axes including X axis
V1=PSX          ;***Sets variable 1 to pulse X
JOGY+           ;***Jogs Y axis to positive direction
V2=PSY          ;***Sets variable 2 to pulse Y
```

SCV[axis]

Description:

Read: Get individual s-curve enable. Value is 0 or 1.
Write: Set individual s-curve enable.

Range is from 0 or 1

Syntax:

Read: [variable]=SCV[axis]
Write: SCV[axis]=[0 or 1]
SCV[axis]=[variable]

Note: If s-curve is enabled for an axis, on-the-fly speed feature can not be used for the corresponding axis.

Examples:

```
SCVX=1      ;***Sets X axis to use s-curve acceleration: on-the-fly speed ; ;
            ; change is NOT allowed for this axis.
SCVY=0      ;***Sets Y axis to use s-curve acceleration: on-the-fly speed ; ;
            ; change is allowed for this axis.
SCVZ=1      ;***Sets Z axis to use s-curve acceleration: on-the-fly speed ; ;
            ; change is NOT allowed for this axis.
SCVU=0      ;***Sets U axis to use s-curve acceleration: on-the-fly speed ; ;
            ; change is allowed for this axis.
```

SSPD[axis]

Description:

Write: Set on-the-fly speed change for an individual axis.
Range is from 1 to 6,000,000 PPS

Syntax:

Write: SSPD[axis]=[value]
SSPD[axis]=[variable]

Note: If s-curve is enabled for an axis, on-the-fly speed feature can not be used for the corresponding axis.

Examples:

```
SCVX=0          ;***Disable s-curve acceleration for X-axis
HSPDX=1000      ;***X-axis high speed
LSPDX=100       ;***Set X-axis low speed
ACCX=100        ;***Set X-axis acceleration
JOGX+           ;***Jogs X axis to positive direction
DELAY=1000      ;***Wait 1 second
SSPDX=3000      ;***Change speed on X-axis on-the-fly to 3000 PPS
```

SSPDM[axis]

Description:

Write: Set individual on-the-fly speed change mode
Range is from 0 to 7

Syntax:

Write: SSPDM[axis]=[0-7]
SSPDM[axis]=[variable]

Examples:

```
SCVX=0          ;***Disable s-curve acceleration for X-axis
HSPDX=1000      ;***X-axis high speed
LSPDX=100       ;***Set X-axis low speed
ACCX=100        ;***Set X-axis acceleration
JOGX+           ;***Jogs X axis to positive direction
DELAY=1000      ;***Wait 1 second
SSPDMX=1        ;***Set on-the-fly speed change mode to 1
ACCX=20000      ;***Set acceleration to 20 seconds
SSPDX=190000    ;***Change speed on X-axis on-the-fly to 190000 PPS
```

STOP

Description:

Command: Stop all axes if in motion with deceleration.
Previous acceleration value is used for deceleration.

Syntax:

STOP

Examples:

```
JOGX+           ;***Jogs X axis to positive direction
DELAY=1000      ;***Wait 1 second
STOP            ;***Stop with deceleration all axes including X axis
```

STOP[axis]

Description:

Stop individual axis if in motion with deceleration.
Previous acceleration value is used for deceleration.

Syntax:

STOP[axis]

Examples:

```
JOGX+           ;***Jogs X axis to positive direction
DELAY=1000      ;***Wait 1 second
JOGY+           ;***Jogs Y axis to positive direction
DELAY=1000      ;***Wait 1 second
STOPX           ;***Stop with deceleration X axis only
```

SYN[axis]C

Description:

Write: Set sync output configuration for axis

Syntax:

Write: SYN[axis]C=[value]
SYN[axis]C=[variable]

Examples:

```
SYNXC=1          ;*** Set sync output configuration to 1 for x-axis
SYNXP=3000       ;*** Set sync output position to 3000 for x-axis
SYNXT=10         ;*** Set sync output pulse time to 10 ms for x-axis
SYNXO            ;*** Turn on sync output for x-axis

V1=1             ;*** Wait until sync output is triggered for x-axis
WHILE V1 != 2
    V1=SYNXS
ENDWHILE

SYNXF            ;*** Disable sync output for x-axis
```

SYN[axis]F

Description:

Write: Disable sync output for axis

Syntax:

Write: SYN[axis]F

Examples:

```
SYNXC=1          ;*** Set sync output configuration to 1 for x-axis
SYNXP=3000       ;*** Set sync output position to 3000 for x-axis
SYNXT=10         ;*** Set sync output pulse time to 10 ms for x-axis
SYNXO            ;*** Turn on sync output for x-axis

V1=1             ;*** Wait until sync output is triggered for x-axis
WHILE V1 != 2
    V1=SYNXS
ENDWHILE

SYNXF            ;*** Disable sync output for x-axis
```

SYN[axis]O

Description:

Write: Enable sync output for axis

Syntax:

Write: SYN[axis]O

Examples:

```
SYNXC=1          ;*** Set sync output configuration to 1 for x-axis
SYNXP=3000       ;*** Set sync output position to 3000 for x-axis
SYNXT=10         ;*** Set sync output pulse time to 10 ms for x-axis
SYNXO            ;*** Turn on sync output for x-axis

V1=1             ;*** Wait until sync output is triggered for x-axis
WHILE V1 != 2
        V1=SYNXS
ENDWHILE

SYNXF            ;*** Disable sync output for x-axis
```

SYN[axis]P

Description:

Write: Set sync output position for axis. 28-bit signed number

Syntax:

Write: SYN[axis]P=[value]
Write: SYN[axis]P=[variable]

Examples:

```
SYNXC=1          ;*** Set sync output configuration to 1 for x-axis
SYNXP=3000       ;*** Set sync output position to 3000 for x-axis
SYNXT=10         ;*** Set sync output pulse time to 10 ms for x-axis
SYNXO            ;*** Turn on sync output for x-axis

V1=1             ;*** Wait until sync output is triggered for x-axis
WHILE V1 != 2
        V1=SYNXS
ENDWHILE

SYNXF            ;*** Disable sync output for x-axis
```

SYN[axis]S

Description:

Read: Get status for sync output of axis

Syntax:

Read: [variable] = SYN[axis]S

Examples:

```
SYNXC=1          ;*** Set sync output configuration to 1 for x-axis
SYNXP=3000       ;*** Set sync output position to 3000 for x-axis
SYNXT=10         ;*** Set sync output pulse time to 10 ms for x-axis
SYNXO            ;*** Turn on sync output for x-axis

V1=1             ;*** Wait until sync output is triggered for x-axis
WHILE V1 != 2
        V1=SYNXS
ENDWHILE

SYNXF            ;*** Disable sync output for x-axis
```

SYN[axis]T

Description:

Write: Set pulse output width time for sync output of axis

Syntax:

Write: SYN[axis]T=[value]

Examples:

```
SYNXC=1          ;*** Set sync output configuration to 1 for x-axis
SYNXP=3000       ;*** Set sync output position to 3000 for x-axis
SYNXT=10         ;*** Set sync output pulse time to 10 ms for x-axis
SYNXO            ;*** Turn on sync output for x-axis

V1=1             ;*** Wait until sync output is triggered for x-axis
WHILE V1 != 2
        V1=SYNXS
ENDWHILE

SYNXF            ;*** Disable sync output for x-axis
```

SUB

Description:

Indicates start of subroutine

Syntax:

SUB [subroutine number]

[Subroutine Number] range is 0 to 31

Examples:

```
GOSUB 1
END
SUB 1
        X0
        X1000
ENDSUB
```

TR

Description:

Read: Get count status of timer register
Write: Set timer register

Once TR is set, it begins to count down to 0. Units ms.

Syntax:

Read: [variable]=TR
Write: TR=[value]

Conditional: IF TR=[variable]
ENDIF

Examples:

```
TR=1000
WHILE 1=1
        IF TR>8000
                X0
        ELSEIF TR>5000
                X3000
        ELSE
                X8000
        ENDIF
ENDWHILE
```

U

Description:

Command: Perform U axis move to target location

With other Axis moves in the same line, linear interpolation move is done.

Syntax:

U[value]

U[variable]

Examples:

```
U10000          ;***Move U Axis to position 10000
V10 = 1200      ;***Set variable 10 value to 1200
UV10            ;***Move U Axis to variable 10 value
```

V

Description:

Assign to variable.
Performax 4EX has 64 variables [V0-V63]

Syntax:

V[Variable Number] = [Argument]
V[Variable Number] = [Argument1][Operation][Argument2]

Special case for BIT NOT:
V[Variable Number] = ~[Argument]

[Argument] can be any of the following:

- Numerical value
- Pulse or Encoder Position
- Digital Output
- Digital Input
- Enable Output
- Motor Status

[Operation] can be any of the following

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
>>	Bit Shift Right
<<	Bit Shift Left
&	Bit AND
|	Bit OR
~	Bit NOT

Examples:

```
V1=12345        ;***Set Variable 1 to 123

V2=V1+1         ;***Set Variable 2 to V1 plus 1

V3=DI           ;***Set Variable 3 to digital input value

V4=DO           ;***Sets Variable 4 to digital output value

V5=~EO          ;***Sets Variable 5 to bit NOT of enable output value
```

WAIT

Description:

Tell program to wait until move on the certain axis is finished before executing next line.

Syntax:

WAIT[axis]
X[variable]

Examples:

```
X10000      ;***Move X Axis to position 10000
WAITX       ;***Wait until X Axis move is done
DO=5        ;***Set digital output
Y3000       ;***Move Y Axis to 3000
WAITY       ;***Wait until Y Axis move is done
```

WHILE

Description:

Perform WHILE loop

Syntax:

WHILE [Argument 1] [Comparison] [Argument 2]

[Argument] can be any of the following:

- Numerical value
- Pulse or Encoder Position
- Digital Output
- Digital Input
- Enable Output
- Motor Status

[Comparison] can be any of the following

=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!=	Not Equal to

Examples:

```
WHILE V1=1          ;***While V1 is 1 continue to loop
    X0
    X1000
ENDWHILE
```

X

Description:

Command: Perform X axis move to target location
With other Axis moves in the same line, linear interpolation move is done.

Syntax:

X[value]
X[variable]

Examples:

```
X10000          ;***Move X Axis to position 10000

X2000Y3000      ;***Move X to 2000 and Y to 3000 in linear interpolation move

V10 = 1200      ;***Set variable 10 value to 1200
XV10            ;***Move X Axis to variable 10 value
```

Y

Description:

Command: Perform Y axis move to target location
With other Axis moves in the same line, linear interpolation move is done.

Syntax:

Y[value]
Y[variable]

Examples:

```
Y10000          ;***Move Y Axis to position 10000

Y2000Z3000      ;***Move Y to 2000 and Z to 3000 in linear interpolation move

V10 = 1200      ;***Set variable 10 value to 1200
YV10            ;***Move Y Axis to variable 10 value
```

Z

Description:

Command: Perform Z axis move to target location
With other Axis moves in the same line, linear interpolation move is done.

Syntax:

Z[value]
Z[variable]

Examples:

```
Z10000                  ;***Move X Axis to position 10000

Y1000Z2000U3000   ;***Move Y to 1000, Z to 2000, U to 3000

V10 = 1200    ;***Set variable 10 value to 1200
ZV10          ;***Move Z Axis to variable 10 value
```

ZHOME[axis][+ or -]

Description:

Command: Perform Z-homing using current high speed, low speed, and acceleration.

Syntax:

ZHOME[Axis][+ or -]

Examples:

```
ZHOMEX+    ;***Z Homes X axis in positive direction

ZHOMEZ-    ;***Z Homes Z axis in negative direction
```

ZOME[axis][+ or -]

Description:

Command: Perform Zoming using current high speed, low speed, and acceleration.

Syntax:

ZOME[Axis][+ or -]

Examples:

```
ZOMEX+      ;***Homes X axis in positive direction

ZOMEZ-      ;***Homes Z axis in negative direction
```