Performax 2EX-SA

2-Axis
Stepper Motor Controller
Standalone Version

Manual

First edition, January 2006

Revision History:

1.0 – First revision
1.02 – StepNLoop update
1.03 – Added standalone capability
1.04 – Updated "STORE" example, changed connections section
1.07 – Update standalone section
1.08 – Software interface update
1.09 - Interpolated moves uses global speed settings at all times, removed extra on-the-fly speed explanation, remove extra STOP/ABORT explanation, add DI[bit], DO[bit], ID to ASCII table, pulse/encoder ratio max spec corrected, section 8 changed to "ASCII Language Specification", while in StepNLoop mode user not allowed to set pulse counter, simplified motion profile and on-the-fly speed change section, SNL not allowed with joystick is on, fixed SNL status in ASCII table, removed extra blank pgs, individual speed settings while using SSPD

Firmware Compatibility:

V112

Software Compatibility:

V105

Table of Contents

1. Introduction

Performax 2EX-SA is an advanced 2 axis standalone programmable motion controller with USB 2.0 communication.

Performax 2EX-SA Features

- 2 Axis Stepper Controller
- Maximum step output frequency of 6M PPS
- Plus Limit, Minus Limit, and Home opto-isolated inputs per axis
- 8 general-purpose opto-isolated inputs.
- 8 general-purpose opto-isolated outputs. 100mA sink capable.
- 2 analog inputs. 0-5VDC 10 bit resolution.
- Advanced joystick control for X and Y axis control.
- S-curve or trapezoidal acceleration profile control
- On-the-fly speed change
- Advanced StepNLoop closed loop control
- USB 2.0 communication.
- Standalone programmable

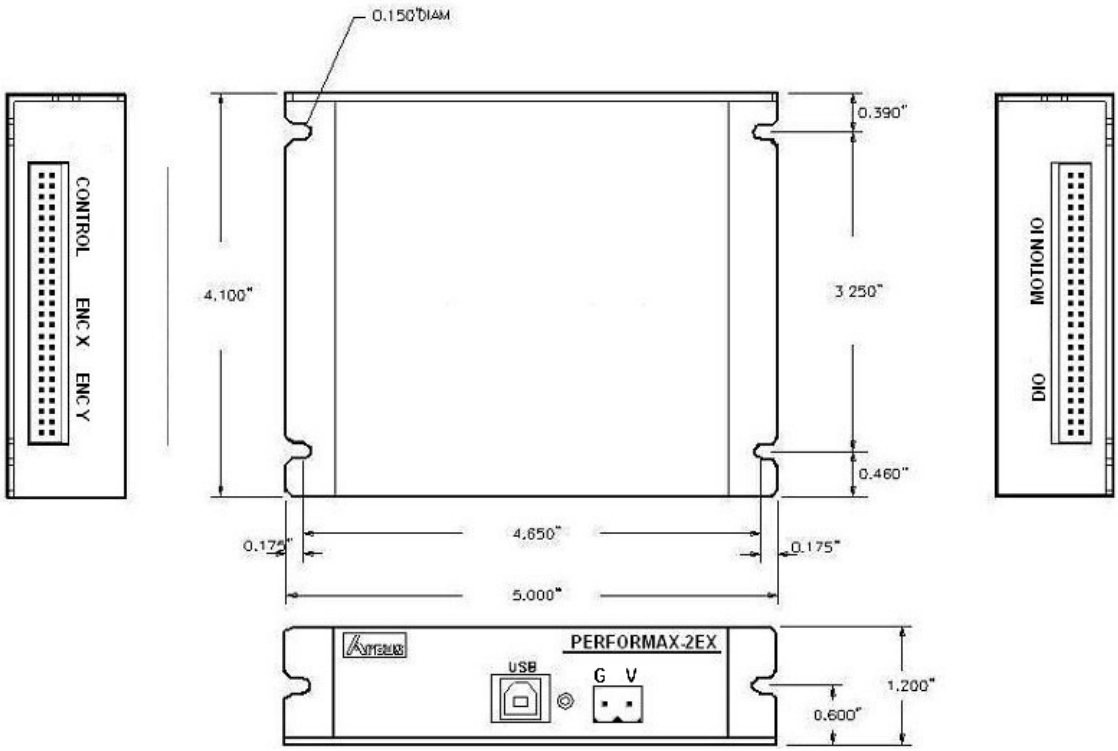
Performax 2EX-SA comes with a Windows DLL for easy interface with the program from common programming language such as VB, VC++, and LabVIEW. Sample program in VB is provided.

Contacting Support

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

Or, contact your local distributor for technical support.

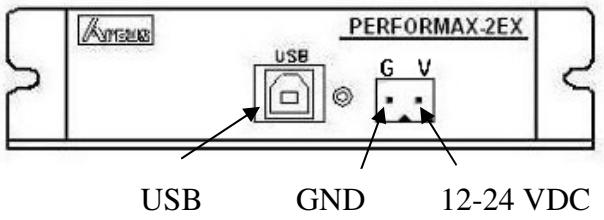
2. Dimensions



3. Pin Descriptions

USB and Input Power

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



Control/Encoder IO

Description	Pin	Pin	Description
NC	1	2	NC
+5V	3	4	GND
Encoder AY	5	6	/Encoder AY
Encoder BY	7	8	/Encoder BY
Encoder ZY	9	10	/Encoder ZY
NC	11	12	NC
NC	13	14	NC
NC	15	16	NC
NC	17	18	NC
NC	19	20	NC
+5V	21	22	GND
Encoder AX	23	24	/Encoder AX
Encoder BX	25	26	/Encoder BX
Encoder ZX	27	28	/Encoder ZX
NC	29	30	NC
NC	31	32	NC
NC	33	34	NC
NC	35	36	NC
NC	37	38	NC
+5V	39	40	GND
Pulse X	41	42	Dir X
Enable X	43	44	Analog Input 1
Pulse Y	45	46	Dir Y
Enable Y	47	48	Analog Input 2
NC	49	50	NC

50 pin Motion IO / DIO

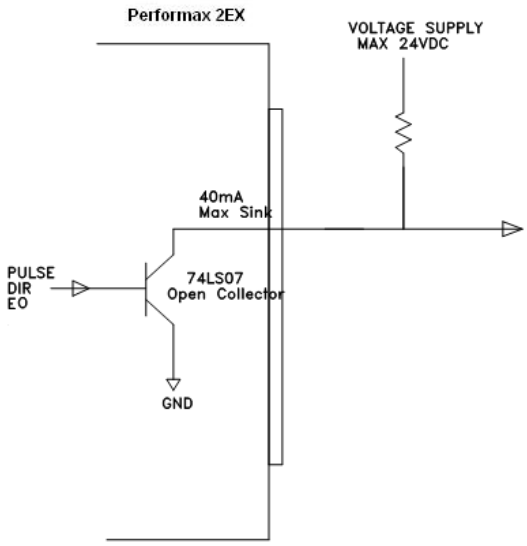
Description	Pin	Pin	Description
NC	1	2	NC
Opto Supply	3	4	NC
+LIM X	5	6	-LIM X
HOME X	7	8	NC
+LIM Y	9	10	-LIM Y
HOME Y	11	12	NC
NC	13	14	NC
NC	15	16	NC
NC	17	18	NC
NC	19	20	NC
NC	21	22	NC
NC	23	24	NC
NC	25	26	NC
NC	27	28	NC
Opto Supply	29	30	Opto Ground
DI1	31	32	DI2
DI3	33	34	DI4
DI5	35	36	DI6
DI7	37	38	DI8
DO1	39	40	DO2
DO3	41	42	DO4
DO5	43	44	DO6
DO7	45	46	DO8
NC	47	48	NC
NC	49	50	NC

Connector Information
For 50 pin connector use standard 0.1 inch spacing dual row 50 pin connector.

Manufacturer: CW Industries
Manufacturer: C3AAG-5018M
Digikey Part: C3AAG-5018M-ND
Description: IDC Cable

Pulse/Dir/Enable Outputs

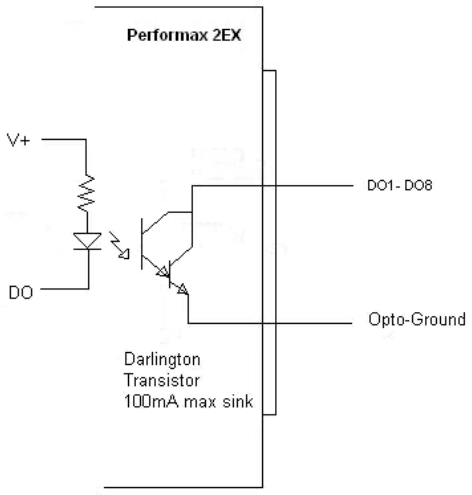
Pulse/Dir/Enable outputs are all open collector outputs capable of sinking up to 40mA at 24VDC as shown below.



Digital Outputs

Digital outputs are opto-isolated outputs using Darlington transistors that can sink up to 100mA current at maximum voltage of 24VDC.

Digital outputs are active high outputs.

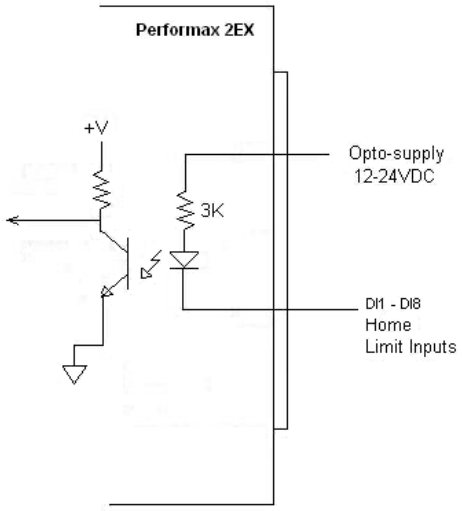


Digital Inputs

All inputs including limits, homes, and digital inputs DI1 to DI8 are opto-isolated and require a opto-supply input of 12 to 24VDC.

To trigger the input, sink the signal to the ground of the opto-supply.

The digital inputs are active high inputs.



Encoder Inputs

When connecting differential connector use Encoder A, /A, B, /B, Z index, /Z index channels.

When connecting single-ended encoders, use Encoder /A, /B, and /Z index channels.

Analog Inputs

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

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

4. Getting Started

Typical Software Setup Steps for First Time Users:

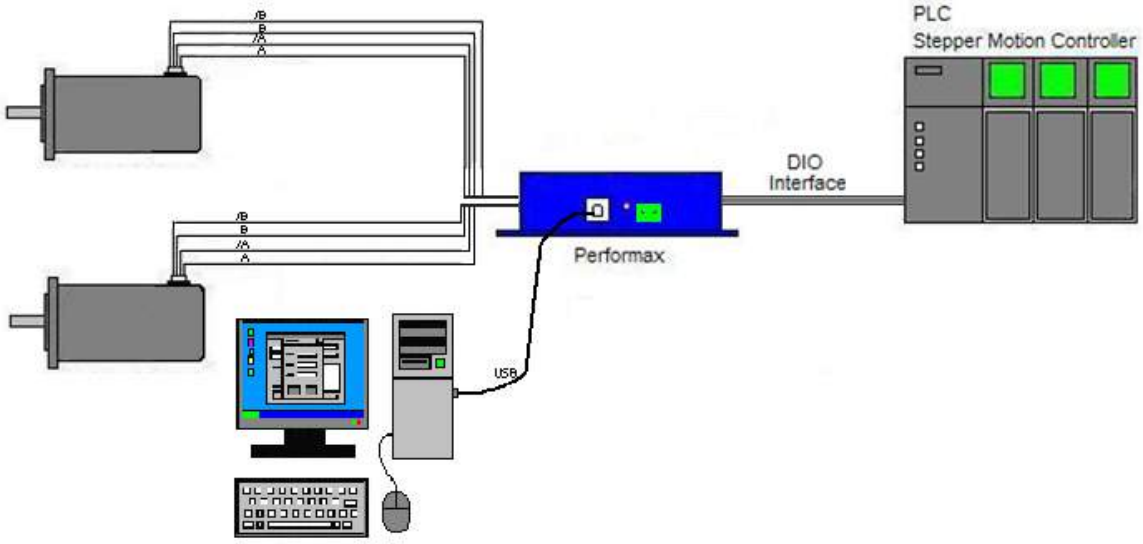
1) **Run Performax USB Driver Setup program.** Both setup program and the manual for Performax USB Driver setup can be downloaded from the web site.

2) **Connect the Performax USB device to the Windows PC**. Go through typical Windows USB device setup. See the Performax USB Driver Setup manual for details.

3) **Use test program to check the USB communication and controller features**. Sample program for each of the Performax devices can be downloaded from the web site.

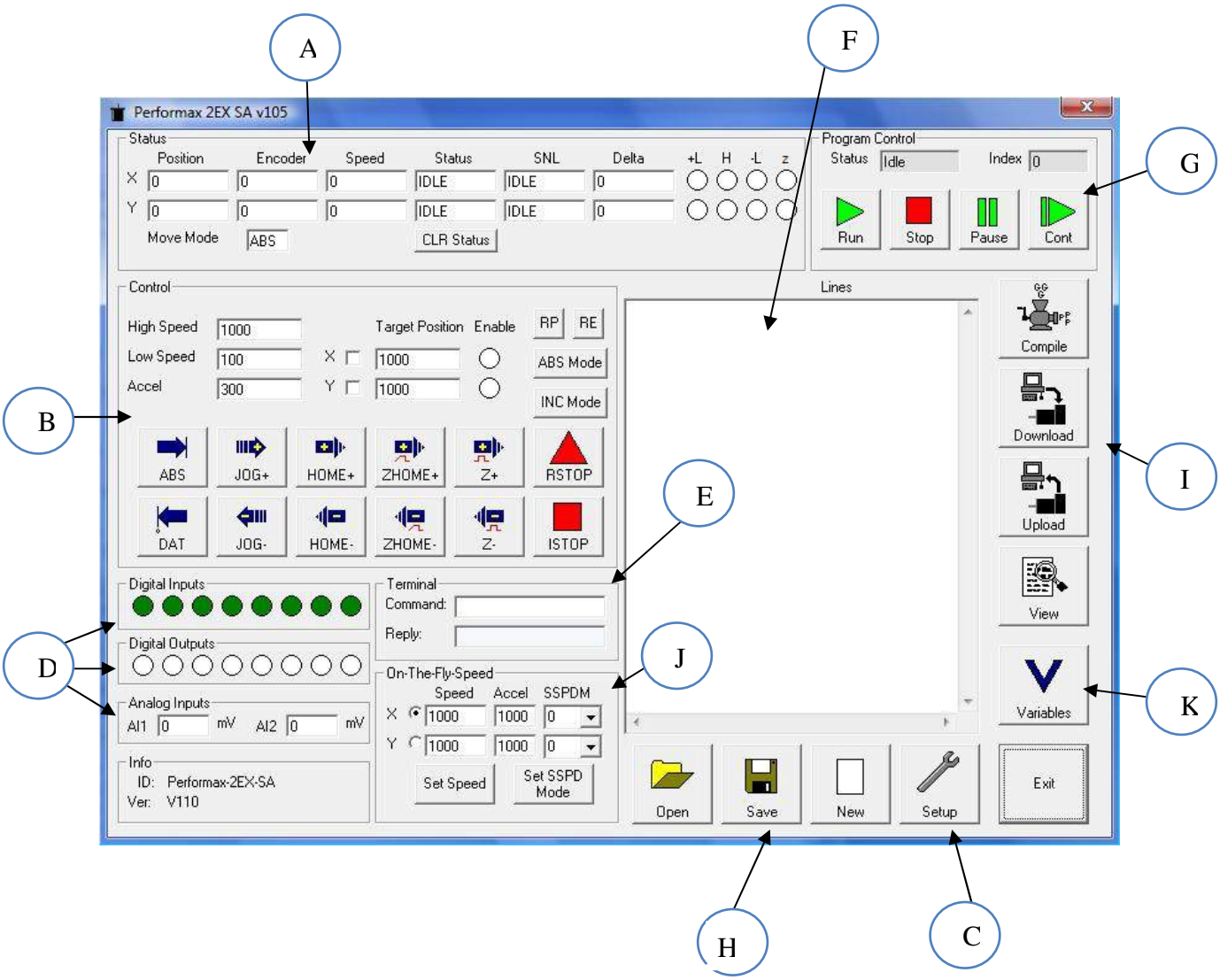
4) **Write your custom application program.** Sample application programs written in VB6, C++, or LabVIEW can be used as a starting point and these programs are available for download from the website.

Typical Setup

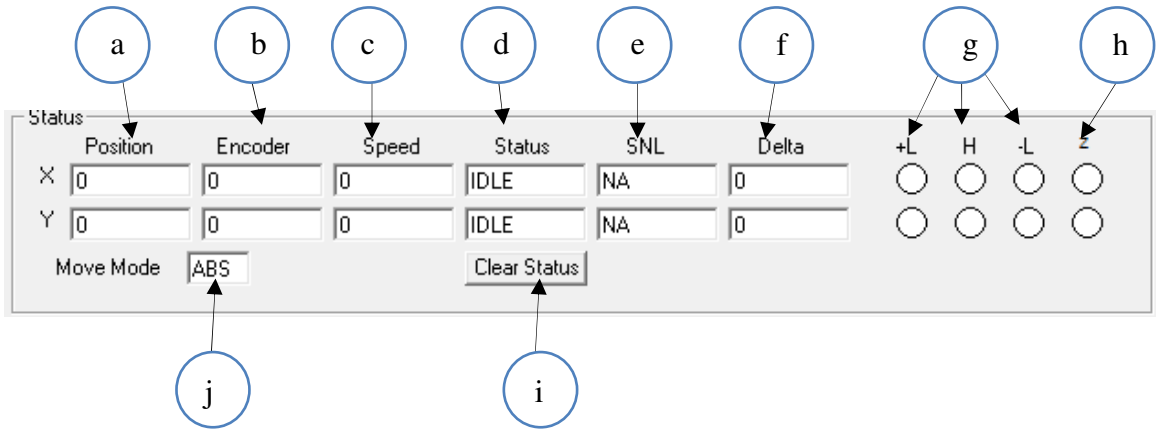


Important Note: In order to communicate with PMX-2EX-SA through USB, proper driver must be installed first. Before connecting the PMX-2EX-SA device or running any program, please go to the Arcus website and download the USB driver installation instruction and run the USB Driver Installation Program.

5. Sample Program



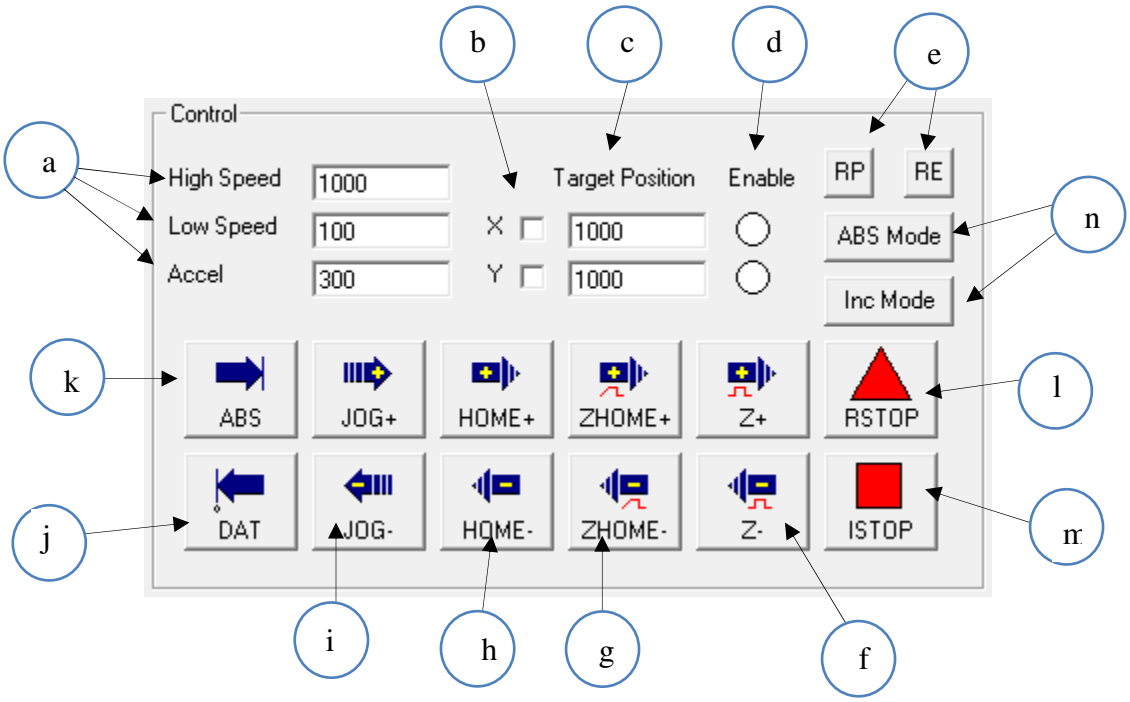
A. X Status, Y-Status:



a. Current pulse position (X/Y axis).
b. Current encoder position (X/Y axis).
c. Current speed (X/Y axis).
d. Motor status (X/Y axis).
    i. IDLE – Motor is not moving.
    ii. ACCEL – Motor is accelerating.
    iii. CONST – Motor is moving at constant speed.
    iv. DECEL – Motor is decelerating
e. StepNLoop value (X/Y axis).
    i. NA – StepNLoop is disabled.
    ii. IDLE – Motor is not moving.
    iii. MOVING – Motor is moving.
    iv. CORRECTING – Motor is attempting to correct its position.
    v. STOPPING – Motor is stopping using deceleration.
    vi. ABORTING – Motor is stopping without deceleration.
    vii. JOGGING – Motor is jogging.
    viii. HOMING – Motor is homing using the home switch.
    ix. Z-HOMING – Motor is homing using the Z-index
    x. ERR-RANGE – The error range has been exceeded.
    xi. ERR-ATMPT – The maximum number of attempts has been made to correct the position.
    xii. ERR-STALL – The motor has stalled.
    xiii. ERR-LIM – A limit switch has been hit.
f. StepNLoop delta value (X/Y axis).
g. –Limit, +Limit, Home input status (X/Y axis).
h. Z encoder index channel status (X/Y axis)
i. Clear status button and StepNLoop status(X/Y axis).
j. Move mode.

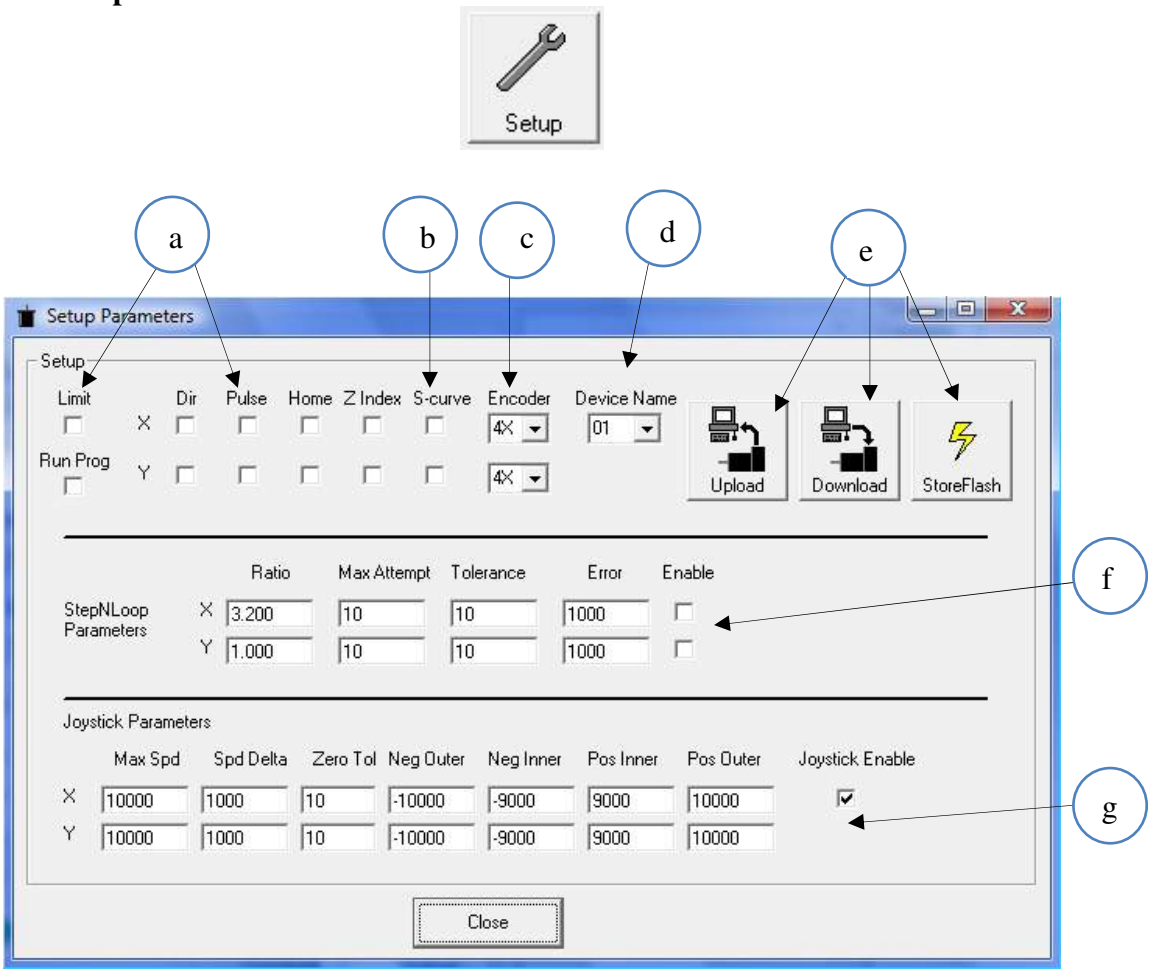
i. ABS mode: On individual and interpolated move commands, motor will move to target position.
ii. INC mode: On individual and interpolated move commands, motor will increase its position by the target position amount.

B. Control



a. Global high speed, low speed, and acceleration values are entered here (X/Y axis). To give each axis individual speed parameters, enter HSPD[axis], LSPD[axis], and ACC[axis] into the command line in the "Terminal" section.
b. Select X/Y axis. Selection of both axes will result in synchronous movement.
c. Target position entered here (X/Y axis).
d. Enable driver power for the indicated motor (X/Y axis).
e. Reset the position/encoder position.
f. ZOME+/ZOME- Only encoder index channel used for homing.
g. ZHOME+/ZHOME- Both encoder index and home sensor used for homing.
h. HOME+/HOME- Only home sensor used for homing.
i. JOG+/JOG- Jogs the motor in the positive or negative direction.
j. DAT – Moves the motor to position zero.
k. ABS – Moves the motor to the target position using the given speed parameters.
l. RSTOP – Stop the motor with deceleration.
m. ISTOP – Stop the motor immediately, without deceleration.
n. Sets the move mode to ABS Mode or INC Mode.

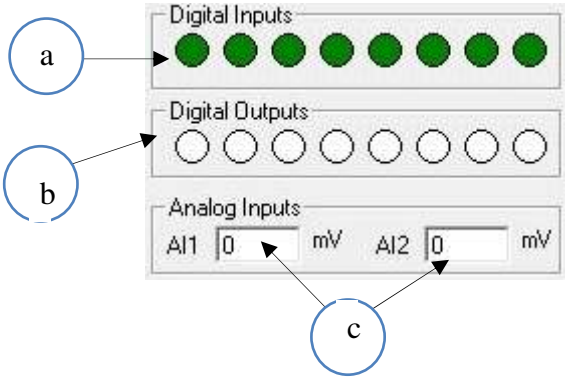
C. Setup



a. Setup polarity for direction, pulse, home, z-index, and limit (X/Y axis).
b. Enable/Disable s-curve (X/Yaxis).
c. Encoder multiplier – Increase the number of encoder counts by this factor (X/Y axis).
d. Device Name – Set the name of the device. Must be in the range of 2EX00 to 2EX99.
e. Upload parameters from flash, Download parameter to the device, or Store the setting to flash memory. The following parameters are stored to flash:
    i. Device Number
    ii. Polarity Settings
    iii. S-Curve Settings
    iv. Joystick Control Settings
    v. StepNLoop Settings
    vi. Non-volatile variables
    vii. Automatic program run on boot up

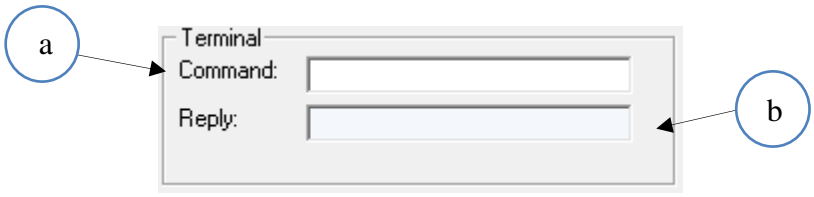
f. StepNLoop parameters (X/Y axis).
   i. Enable/disable StepNLoop
   ii. Ratio between pulses/rev and encoder counts/rev
   iii. Maximum number of attempts to correct the position
   iv. Allowable tolerance range in which the motor will not try to correct itself.
   v. Error range in which the motor will try to correct itself. The axis goes into error state if the error range is exceeded.
g. Joystick parameters (X/Y axis). See joystick section for further details.

D. Inputs/Outputs



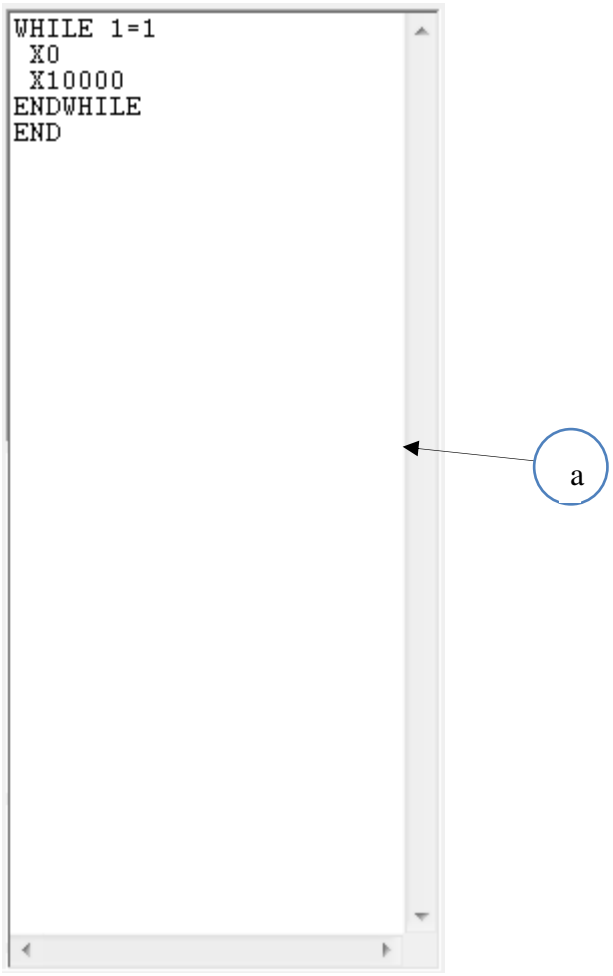
a. Digital input status for DI1-DI8.
b. Digital output status for DO1-DO8.
c. Status of analog inputs AI1 and AI2. Units are in milli-volts [0-5000 mV].

E. Terminal



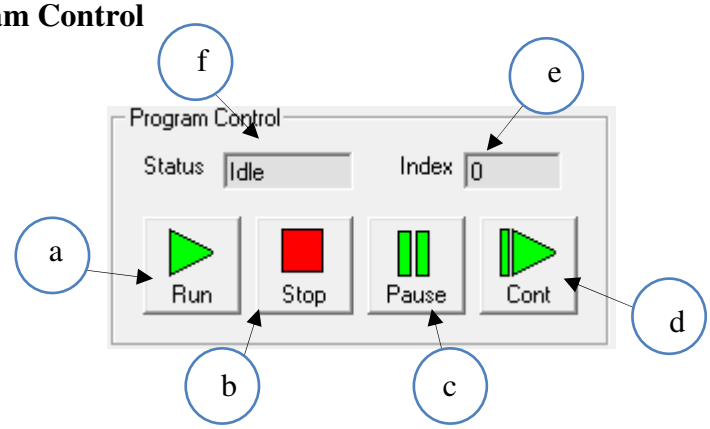
a. Send commands to the PMX-2EX-SA through this terminal
b. Replies from the PMX-2EX-SA will be shown here.

F. Text Programming Box



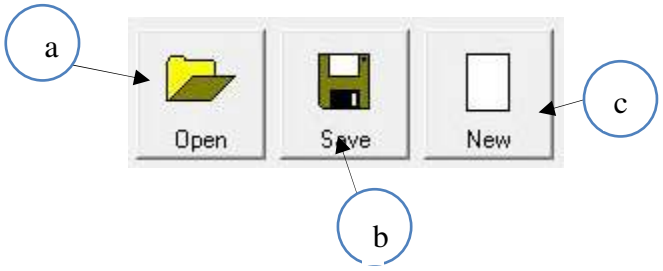
a. Text box for standalone program. See details on programming language.

G. Program Control

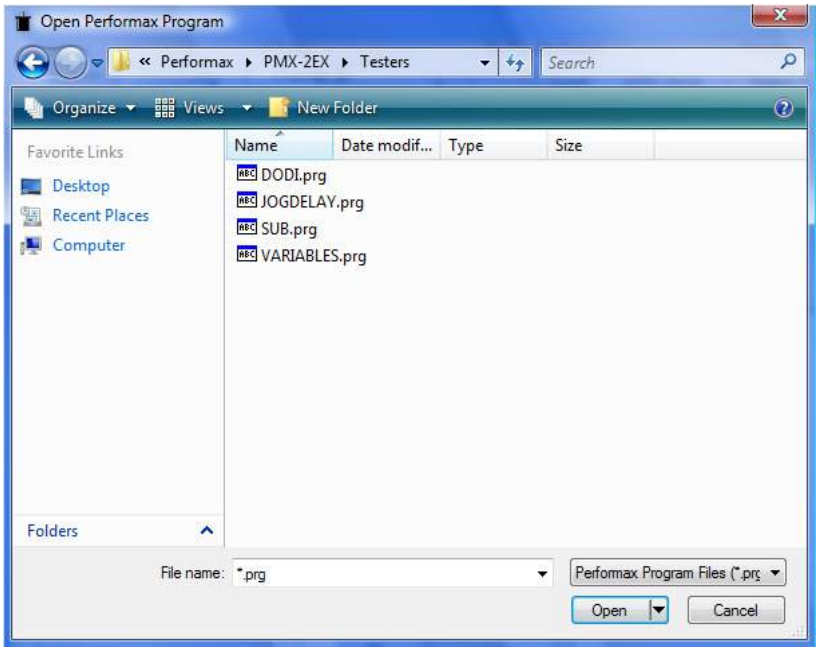


a. Run –Standalone program is run.
b. Stop – Program is stopped.
c. Pause – Program that is running can be stopped.
d. Cont – Program that is paused can be continued
e. Index - Current line of code that is being executed.
f. Status of standalone program:
    i. Idle – Program is not running.
    ii. Running – Program is running.
    iii. Paused – Program is paused.
    iv. Error – Program is in an error state.

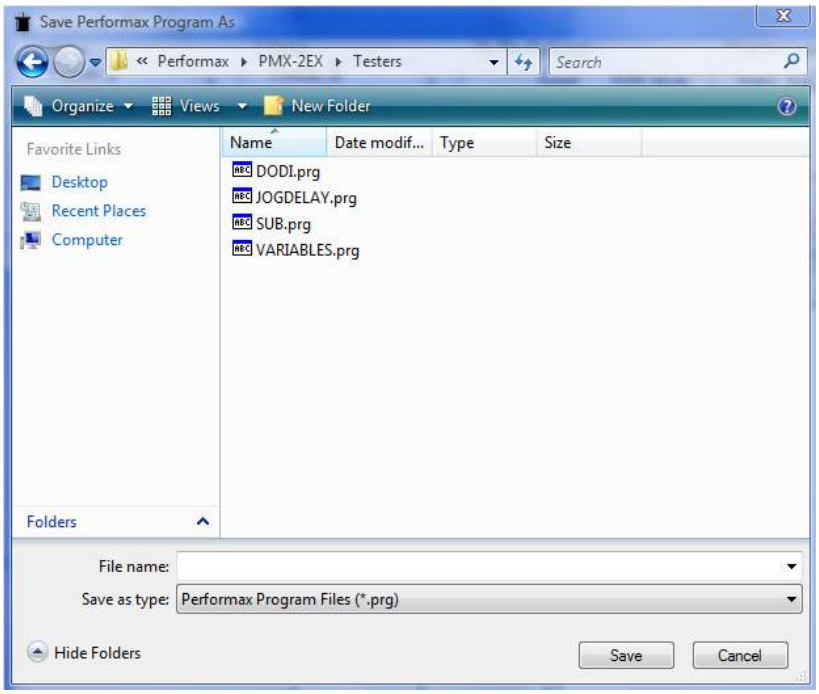
H. Program File Control



a. Open – Standalone program is loaded to the text programming box. When this button is pressed, a typical Windows file open dialog box will open:

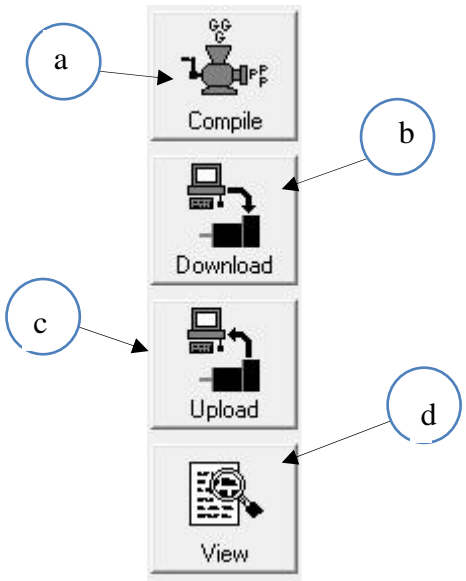


b. Save – Standalone program in the text programming box is saved to a file. When this button is pressed, a typical Windows file save dialog box will open:



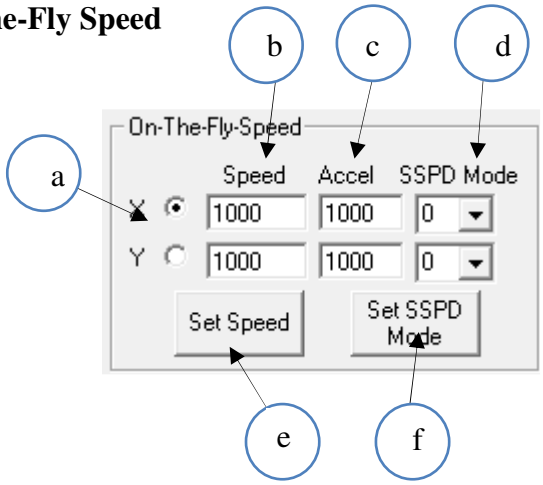
c. When this button is pressed, the text programming box is cleared.

I. Compiler



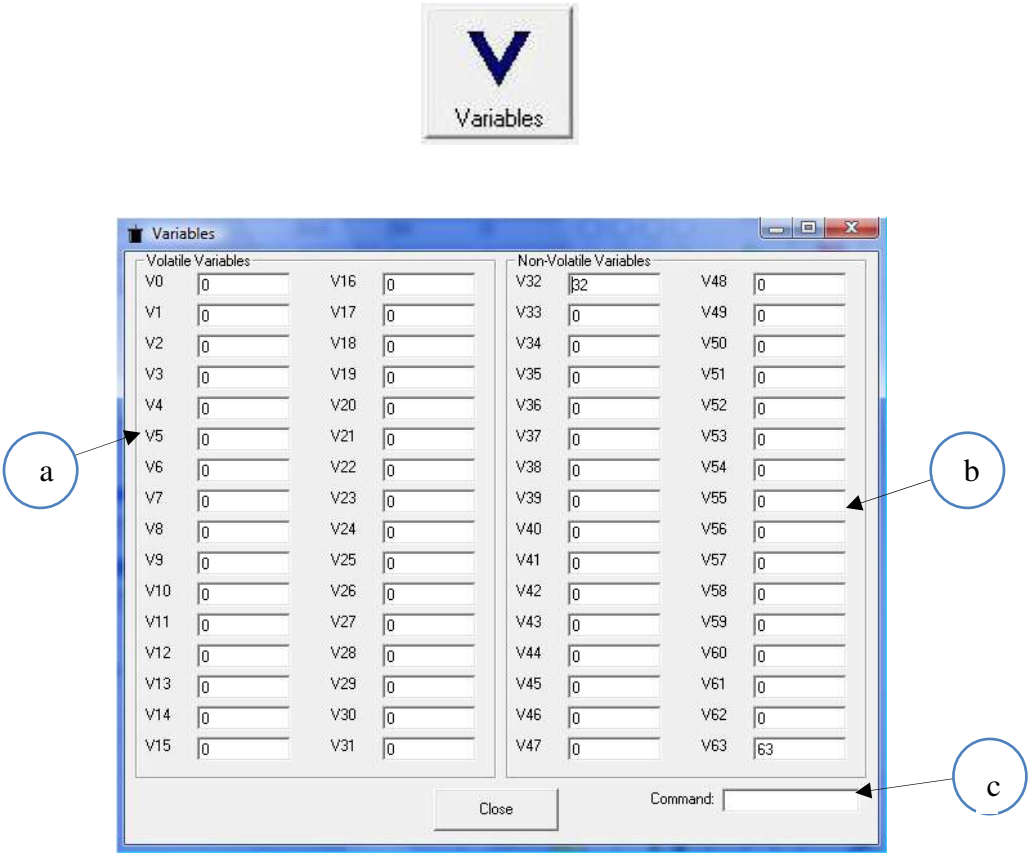
a. Compile code in text programming box into assembly level code that the PMX-2EX-SA can understand.
b. Download the compiled code into memory. Note that the text based code must be compiled before download.
c. Upload standalone code that is currently on your PMX-2EX-SA. This automatically translates assembly level language to readable text-based code.
d. View compiled code for easy cutting and pasting.

J. On-The-Fly Speed



a. Select X/Y axis.
b. Select destination speed of the axis.
c. Select the acceleration used during an on-the-fly speed change. The acceleration will return back to its normal value after the speed change.
d. Select the SSPD mode for the axis. See On-The-Fly Speed section for details.
e. Set the SSPD mode for the axis.
f. Set on-the-fly speed change. Acceleration will be taken from the "Accel' field.

K. Variables



a. Current values of variables that cannot be stored to flash.
b. Current values of variables that can be stored to flash.
c. Send commands to the PMX-2EX-SA through this terminal.

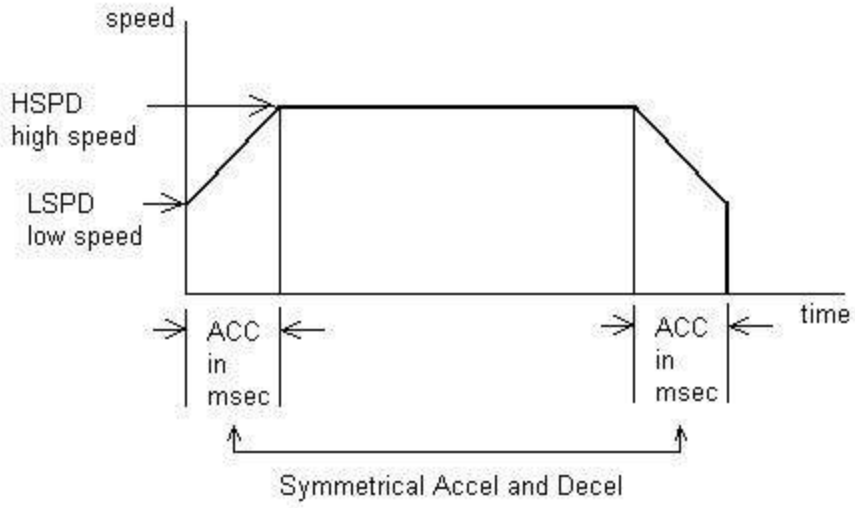
6. Motion Control Overview

All the commands described in this section are all interactive commands and do not transfer over directly to standalone commands. Please refer to the "Standalone Language Specification" section for details.

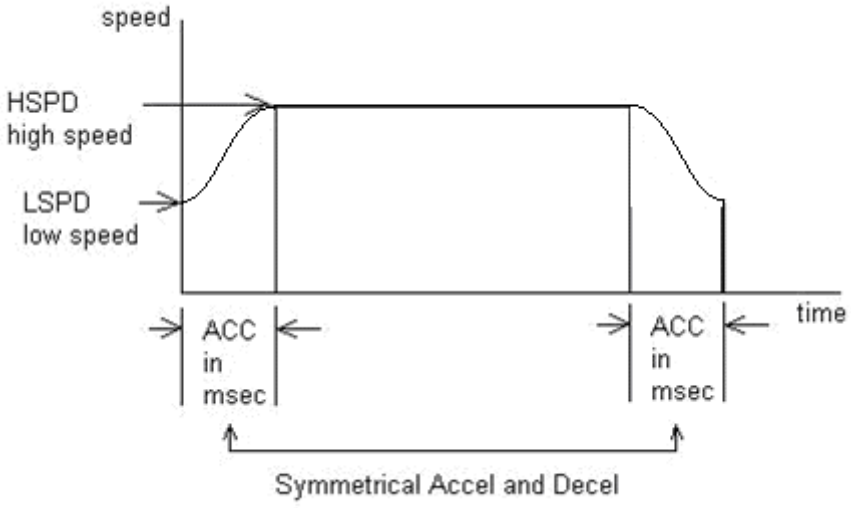
Motion Profile

Performax 2EX-SA can generate up to 6M pulses per second pulse rate.

By default, the Performax 2EX-SA uses trapezoidal velocity profile.



The Performax 2EX-SA can also use an s-curve velocity profile by issuing the **SCVX/SCVY** command. S-curve velocity profile is shown below:



Acceleration and deceleration time is in milliseconds and are symmetrical. Use the **ACCX/ACCY** command to set and get the acceleration/deceleration value. For example to set the acceleration of the x-axis to 300 msec, use the **ACCX=300** command. To get the current acceleration setting for the x-axis, use the **ACCX** command and the reply will be the current acceleration setting. To set and retrieve the global acceleration, use the **ACC** command.

High Speed and Low Speed are in pps(pulses/second). Use **HSPDX/HSPDY** and **LSPD/LSPDX/LSPDY** to set and get high speed and low speed settings. For example, to set the high speed of the x-axis to 15000 use **HSPDX=15000** command. To read the current high speed for the x-axis, send **HSPDX** command and the reply will contain the current high speed. To set and retrieve the global high speed and low speed values use the **HSPD** and **LSPD** 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 are configured.

Interpolated XY moves use the global speed settings at all times.

Note on the acceleration setting:
The allowable acceleration values depend on the **LSPD** and **HSPD** settings. Please see the chart below for details:

HSPD[pps]	Minimum ACC [ms]	Accel Delta [pps]
1-16K	1	300
16K-32K	1	600
32K-80K	1	1,500
80K-160K	1	3,000
160K-325K	1	6,000
325K-815K	1	15,000
815K-1.6M	1	30,000
1.6M-3.2M	1	60,000
3.2M-6M	1	120,000

Note: While in StepNLoop mode, the PPS (pulse/sec) speed numbers must be transposed to encoder counts by using the following formula:

Encoder counter per second = PPS / Step-N-Loop ratio

Examples:

a) If **HSPD** = 300,000 pps, **LSPD** = 250,000 pps:
   a. Get Speed delta: ((300,000 – 250,000) / 6,000) = 8.33
   b. Max acceleration allowable: 6.85 x 1,000 ms = 8,330 ms (8.33 sec)

b) If **HSPD** = 900,000 pps, **LSPD** = 889,000 pps:
   a. Get Speed delta: ((900,000 – 889,000) / 30,000) = 0.033
   b. Max acceleration allowable: 0.026 x 1000 ms = 33 ms (0.033 sec)

SPEED WINDOWS

SSPDM value	Lowest Speed [pps]	Highest Speed [pps]
0	SSPD not used	SSPD not used
1	1	16 K
2	2	32 K
3	5	80 K
4	10	160 K
5	20	325 K
6	50	815 K
7	100	1.6 M
8	200	3.2 M
9	373	6 M

Note: While in StepNLoop mode, the PPS (pulse/sec) speed numbers must be transposed to encoder counts by using the following formula:

Encoder counter per second = PPS / Step-N-Loop ratio

SSPDM value only applies to on-the-fly speed operations. During normal operation, SPPDM should be set to 0.

On-The-Fly Speed Change

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

While using the on-the-fly speed change function, always use individual speed settings (i.e. HSPD[axis], LSPD[axis], ACC[axis] commands). Not doing so would cause unstable behavior.

1) During on-the-fly speed change operation, you must keep the initial and destination speeds within a certain window. See *"SPEED WINDOWS"* chart in previous section.

   To select a speed window, use the **SSPDM[axis]** command.

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

Note that 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.

2) To set acceleration of the on-the-fly speed change, use the **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	300
2	775
3	1,900
4	3,700
5	7,300
6	18,000
7	38,400
8	68,000
9	135,000

Note: While in StepNLoop mode, the PPS (pulse/sec) speed numbers must be transposed to encoder counts by using the following formula:

Encoder counter per second = PPS / Step-N-Loop ratio

Speed Delta [destination speed – initial speed]: 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) / 6,000) = 8.33
   d. Max acceleration allowable: 6.85 x 1,000 ms = 8,330 ms (8.33 sec)

b) If **Destination Speed** = 900,000 pps, **Current Speed** = 889,000 pps:
   e. Get Speed delta: ((900,000 – 889,000) / 30,000) = 0.033
   f. Max acceleration allowable: 0.026 x 1000 ms = 33 ms (0.033 sec)

3) To set speed on-the-fly, use **SSPD[axis]** command. Be sure to stay within the selected **SSPDM[axis]** speed window.

In order for the SSPD command to work, the controller must already be in motion by first calling either a jog or absolute move command.

4) To begin normal operation again (i.e. moves not using on-the-fly speed change), send the following command to the controller **"SSPDM[axis]=0"**.

Target Motion

Use **X** and **Y** command to move the axis to the target position.

Target motion has two modes, incremental mode, done by issuing the **INC** command, and absolute mode, done by issuing the **ABS** command. Under incremental mode, the command **X1000** will move the motor by 1000 from the current position. Under absolute mode, the command **X1000** will move the motor to position 1000.

Homing using Home Switch

When the homing command is sent using **HX+/HX-/HY+/HY-** command, the motor ramps up from low speed to high speed and as soon as the home input is triggered, the motor ramps down keeping the home input triggered position as zero. When ramp down is done, the position is not zero since the zero position is when the home input switch is triggered. To have both motors home synchronously use the **H+/H-** command.

Homing using only the Z-index Encoder Channel

Z index channel occurs one pulse per revolution of the motor. Homing can be done using only the Z index channel. When homing with only the Z index channel, only the low speed is used. Use the **ZX+/ZX-/ZY+/ZY-** commands for this type of homing. To have both motors home synchronously use the **Z+/Z-** command.

Homing using Home Switch and Encoder Z Index Channel

Homing can also be done using the Z index channel of encoder and the home switch. Index channel home command is issued using **ZHX+/ZHX-/ZHY+/ZHY-** command. When the command is issued, the motor ramps to high speed until the home switch is triggered which at that time motor decelerates to low speed. At low speed the motor continues to move until the Z index channel is triggered and at that time the position counter is set to zero. To have both motors home synchronously use the **ZH+/ZH-** command.

Jogging

Jogging is available for continuous speed operation. Use **JX+/JX-/JY+/JY-** command. To have both motors jog synchronously us the **J+/J-** command.

Stopping Motor

When the motor is moving, jogging, or homing, **ABORTX/ABORTY** command will immediately stop the indicated motor. **ABORT** command without the axis letter will stop all axes in motion without deceleration. **STOPX/STOPY** command will decelerate

the individual axis to low speed and then stop. **STOP** command without the axis letter will stop all the axes in motion with deceleration.

Motor Position

Motor position can be set and read using the **PX/PY** command. For example to set the X axis position to 2000, use the **PX=2000** command. To read the X axis current position use the **PX** command and the reply will contain the current position counter.

The encoder position can be read and set using the **EX/EY** command.

Pulse Speed

Current actual pulse rate or speed can be read using the **PSX/PSY** command.

Motor Status

Motor status can be read anytime using **MSTX/MSTY** command. Following are bit representation of motor status:

Bit	Description
0	Motor in acceleration
1	Motor in deceleration
2	Motor running at constant speed
3	Not Used
4	Plus limit input switch status
5	Minus limit input switch status
6	Home input switch status
7	Plus limit error. This bit is latched when plus limit is hit during motion. This error must be cleared (using **CLR/CLRX/CLRY** command) before issuing any subsequent move commands.
8	Minus limit error. This bit is latched when minus limit is hit during motion. This error must be cleared (using **CLR/CLRX/CLRY** command) before issuing any subsequent move commands.
9	Z Index Channel status
10	Joystick Control On status

Polarity

The polarity settings of the PMX-2EX-SA can also be read or set at anytime using the **POLX/POLY** commands. The following is the bit representation of the polarity:

Bit	Description
0	Pulse
1	Home
2	Not Used
3	Not Used
4	Not Used
5	Home
6	+/- Limit
7	Z-Index
8	Encoder 2X
9	Encoder 4X

Note: Settings bits 8 and 9 to zero will default the encoder to 1X.
Note: X axis limit input setting controls the limit polarity for both axes.

Digital Outputs

Digital output command **DO** can be used to read and set the 8 bit digital outputs.

To read all the digital outputs, use the **DO** command and the reply will contain DO values. To set all the digital outputs, use the **DO=[value]** command.

For example to set all the digital outputs on, use the **DO=255** command. Number 255 corresponds to FF in hex and 11111111 in binary.

Individual digital outputs can be read using the **DO[value]** command and written using the **DO[value]=[value]** command.

Digital Inputs

Digital input command **DI** can be used to read the 8 bit digital inputs. Each individual digital input can be read using the **DI[value]** command.

Limit Inputs

If the positive limit switch is triggered while moving in the positive direction, the motor will immediately stop. Likewise for the negative limit while moving in the negative direction.

The limit switch is an opto-isolated input. Supply the opto-supply voltage 12 to 24VDC. To trigger the limit input switch, connect the input signal to ground of the opto-supply.

To read the limit switch input status, use the **MSTX/MSTY** command.

StepNLoop Closed Loop Control

Performax 2EX-SA module has closed loop position control algorithm called StepNLoop control for accurate positioning of the motor using the integrated encoder.

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 **DXX/DXY** 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:

SNL Parameter	Description
Pulse/Encoder Ratio	Number of pulse counts/number of encoder counts per revolution of motor. Use **SLRX/SLRY** command to set the ratio. Value must be in the range [0.001 , 9999.999].
Tolerance Range	When the actual encoder position is within desired encoder position by this tolerance range, no position correction is done. Use **SLTX/SLTY** command to set the tolerance range.
Correction Range	When the actual encoder position is within desired encoder position by this correction range, position correction is done when idle. If the actual encoder position is outside of correction range, the motor status goes to error state. Use **SLEX/SLEY** 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 **SLAX/SLAY** command to set the maximum correction attempt number.

To enable and disable the StepNLoop feature use **SLX/SLY** command.

To read the StepNLoop status, use **SLSX/SLSY** command to read the status.

Following are the StepNLoop status values:

Value	Description
0	Idle
1	Moving

2	Correcting
3	Stopping
4	Aborting
5	Jogging
6	Homing
7	ZHoming
8	Correction Range Error. To clear this error, use **CLRS** command
9	Correction Attempt Error. To clear this error, use **CLRS** command.
10	Stall Error. **DXX/DXY** value has exceeded the **SLEX/SLEY** value. To clear this error, use **CLRS** command.
11	Limit Error
12	N/A

Notes:
Once StepNLoop is enabled, the position move commands are in term of encoder position.

For example, X1000 means to move the motor to encoder position 1000.

Once StepNLoop is enabled, the speed is in encoder speed.

For example HSPD=1000 when StepNLoop is enabled means that the target high speed is 1000 encoder counts per second.

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.

While StepNLoop is enabled, the user can set the encoder value by using the **EX/EY** commands, however setting the pulse counter using the **PX/PY** commands is prohibited. Once the encoder position is set, the desired encoder position becomes the new target position.

Joystick Control

Using two analog inputs, XY joystick control can be done. Analog input 1 corresponds to the X axis and analog input 2 corresponds to the Y axis control. Analog input of 0V to 2.5V represents negative joystick direction and analog input of 2.5 to 5V represents positive joystick direction. 2.5V represents the zero joystick position. To set tolerance of the zero joystick position, use **JV5** and **JV6** variables. For example, if **JV5** is set to 100, then the zero range for X axis joystick control will be from 2.4V to 2.6V.

Maximum joystick speed is set using **JV1** and **JV2** variables.

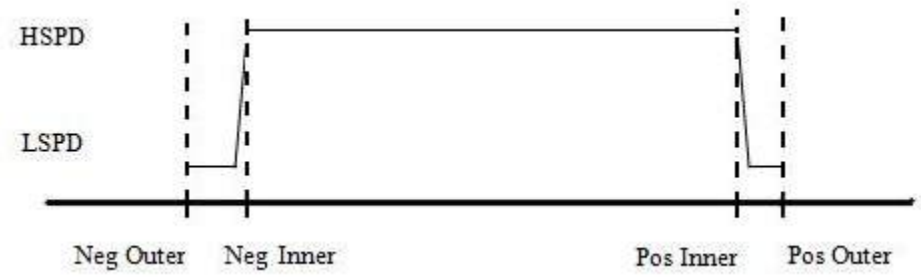
To enable joystick control use **JO** command. The disable joystick control use **JF** command or **ABORT** command.

Summary of joystick control parameters

JV1	X axis Maximum Joystick Speed at 5V and 0V.
JV2	Y axis Maximum Joystick Speed at 5V and 0V.
JV3	X axis Maximum speed change
JV4	Y axis Maximum speed change
JV5	X axis zero tolerance range for analog input
JV6	Y axis zero tolerance range for analog input

Joystick control also has soft limit control. Limits are broken down into positive inner and outer limits and negative inner and outer limits. When moving in positive direction, as soon as positive inner limit is crossed, the speed is reduced. If position crosses over the outer limit, the joystick speed is set to zero. Same goes for negative direction and negative limits.

The behavior of the limits of the joystick control is explained by the following:



Summary of joystick soft limit parameters

JL1	X axis Negative Outer Soft Limit
JL2	X axis Negative Inner Soft Limit
JL3	X axis Positive Inner Soft Limit
JL4	X axis Positive Outer Soft Limit
JL5	Y axis Negative Outer Soft Limit
JL6	Y axis Negative Inner Soft Limit
JL7	Y axis Positive Inner Soft Limit
JL8	Y axis Positive Outer Soft Limit

Note: If joystick control is enabled, StepNLoop is automatically disabled.

Standalone Program Specification

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.

Stand-alone execution while in Step-N-Loop:
While a stand-alone program is running in closed-loop operation, before executing an absolute move command, the controller first verifies that it is NOT correcting or moving to a previous absolute position.

Error Handling:
If an error occurs during standalone execution (i.e. limit error, stall error, max attempt error, etc.), the program automatically jumps to SUB 31. If SUB 31 is not defined, the program will cease execution and go to error state. If SUB 31 is defined by the user, the code within SUB 31 is first executed, and then standalone execution continues.

Calling subroutines over communication:
Once a subroutine is written into the flash, 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-2EX-SA will return with an error.

Device Number

Performax 2EX-SA module provides the user with the ability to set the device number of a specific device. In order to make these changes, first store the desired number using the **DN** command. Please note that this value must be within the range [2EX00-2EX99].

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

Storing to Flash

The following items are stored to flash:

- Device Name
- Polarity settings
- StepNLoop parameters
- S-curve settings
- Joystick parameters
- Automatic program run on boot
- Second half of general purpose variables (V32-V63)

Note: Standalone program is automatically stored to flash when it is downloaded.

7. Communication

Performax communication is USB 2.0 compatible.

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 Performax.

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 10 milliseconds. This value will vary slightly with CPU speed of PC and with command type.

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 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 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. Index number 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.

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.

USB Communication Issues

A common problem that users may have with USB communication is that after sending a command from the PC to the device, the response is not received by the PC until another command is sent. In this case, the data buffers between the PC and the USB device are out of sync. Below are some suggestions to help alleviate this issue.

1) **Multi-threading:** Be sure that your application does not employ multi-thread processing. See "important note" in the beginning of this section.

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

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

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

3) **USB Cable:** Another source of USB communication issues may come from the USB cable. Confirm that the USB cable being used has a noise suppression choke. See photo below:

8. ASCII Language Specification

All the commands described in this section are all interactive commands and do not transfer over directly to standalone commands. Please refer to the "Standalone Language Specification" section for details.

Performax language is case sensitive. All command should be in capital letters.

Invalid command is returned with ?(Error Message). Always check for proper reply when command is sent.

Command	Description	Return	Min	Max
ABORT	Aborts all axis moves	OK		
ABORTX ABORTY	Immediately stops the indicated motor if in motion	OK		
ACC	Returns the global acceleration value.	Acceleration in milliseconds	0	5000
ACCX ACCY	Returns acceleration setting for the X-axis and Y-axis	Acceleration in millisecond	0	5000
ACC=[Value]	Sets the global acceleration setting.	OK	0	5000
ACCX=[Value] ACCY=[Value]	Sets acceleration setting for the X-axis and Y-axis	OK	0	5000
AI1 AI2	Returns Analog Input in millivolt	0 to 5V in millivolt	0	5000
DI	Returns Digital Input Value	8 bit value	0	255
DI[1-8]	Return individual input bit status	0,1	0	1
DO	Returns Digital Output Value	8 bit value	0	255
DO[1-8]	Returns the individual output bit status.			
DO=[Value]	Sets Digital Output Value	OK	0	255
DO[1-8]=[Value]	Sets the individual output bit status.			
DN	Return Device Number	2EXXX		
DN=[Value]	Set Device Number	OK	2EX00	2EX99
DXX DXY	Returns delta value for X-axis and Y-axis			

EO1 EO2	Returns Driver Enable Status	0 or 1	0	1
EO1=[0 or 1] EO2=[0 or 1]	Sets Driver Enable	OK	0	1
EX EY	Returns Current Encoder Position	28 bit signed position	-134M	+134M
EX=[Value] EY=[Value]	Sets Current Encoder Position	OK	-134M	+134M
GS[SubNumber]	Call a defined subroutine	OK		
HSPD	Returns the global high speed setting.	High Speed	0	6M
HSPDX HSPDY	Returns high speed setting for the X-axis and Y-axis	High Speed	0	6M
HSPD=[Value]	Set the global high speed setting	OK	0	6M
HSPDX=[Value] HSPDY=[Value]	Sets high speed setting for the X-axis and Y-axis	OK	0	6M
H+	Homes both X/Y Motor Positive	OK		
H-	Homes both X/Y Motor Negative	OK		
HX+ HY+	Homes X/Y Motor Positive	OK		
HX- HY-	Homes X/Y Motor Negative	OK		
I[X Target]: [Y Target]	Performax Interpolated Motion	OK		
ID	Returns Controller ID	Performax-2EX-SA		
INC	Turns on incremental move mode.	OK		
JF	Turns off Joystick Control	OK		
JL[1 to 8]	Return Joystick Control Limits	JL1 – Neg Outer X JL2 – Neg Inner X JL3 – Pos Inner X JL4 – Pos Outer X JL5 – Neg Outer Y JL6 – Neg Inner Y JL7 – Pos Inner Y JL8 – Pos Outer Y		
JL[1 to 8]= [Value]	Sets Joystick Control Limits	OK		

JO	Turns on Joystick Control	OK		
JS	Get the Joystick status	0 – Joystick off 1 – Joystick on		
JV[1 to 6]	Returns Joy Stick Control Parameters	JV1 – Max X Speed JV2 – Max Y Speed JV3 – Max Delta X JV4 – Max Delta Y JV5 – Zero Tol X JV6 – Zero Tol Y		
JV[1 to 6]= [Value]	Sets Joystick Control Parameters	OK		
J+	Jogs both X/Y Motor Positive	OK		
J-	Jogs both X/Y Motor Negative	OK		
JX+ JY+	Jogs Motor Positive	OK		
JX- JY-	Jogs Motor Negative	OK		
LSPD	Returns the global low speed setting	Low Speed	0	6M
LSPDX LSPDY	Returns low speed setting for the X-axis and Y-axis	Low Speed	0	6M
LSPD=[Value]	Sets the global low speed setting.	OK	0	6M
LSPDX=[Value] LSPDY=[Value]	Sets low speed setting for the X-axis and Y-axis	OK	0	6M
MM	Returns the move mode that the controller is currently in.	0 – ABS mode 1 – INC mode		
MSTX MSTY	Returns Motor Status	bit 0 – Accelerating bit 1 – Decelerating bit 2 – Constant bit 3 – Not Used bit 4 – Positive Limit On bit 5 – Negative Limit On bit 6 – Home On bit 7 – Positive limit error bit 8 – Negative		

		limit error bit 9 – Z Index On bit 10 – Joystick On		
POLX POLY	Returns Polarity	bit 0 – Pulse bit 1 – Dir bit 5 – Home bit 7 – Z index bit 8 – Encoder 2X bit 9 – Encoder 4X (with bit 8 and 9 zero means Encoder 1X)	0	931
POLX=[Value] POLY=[Value]	Sets Polarity	OK	0	931
PSX PSY	Returns Current Pulse Speed		0	300000
PX PY	Returns Current Pulse Position	28 bit signed position	-134M	+134M
PX=[Value] PY=[Value]	Sets Current Pulse Position	OK	-134M	+134M
SCVX SCVY	Get s-curve on/off status		0	1
SCVX=[0 or 1] SCVY=[0 or 1]	Set s-curve on/off status	0 – off 1 – on		
SLAX SLAY	Returns maximum number of StepNLoop control attempt	32-bit		
SLAX=[value] SLAY=[value]	Sets maximum number of StepNLoop control attempt	OK		
SLEX SLEY	Returns StepNLoop correction value.	32-bit		
SLEX=[value] SLEY=[value]	Sets StepNLoop correction value.	OK		
SLRX SLRY	Returns StepNLoop ratio value	32-bit		
SLRX=[factor] SLRY=[factor]	Sets StepNLoop ratio value.	OK		
SLSX SLSY	Returns current status of StepNLoop control	0 – Idle 1 – Moving 2 – Correcting 3 – Stopping 4 – Aborting 5 – Jogging 6 – Homing		

		7 – Z Homing 8 – Correction Range Error. 9 – Correction Attempt Error. 10 – Stall Error 11 – Limit Error 12 – N/A		
SLTX SLTY	Returns StepNLoop tolerance value	32-bit		
SLTX=[value] SLTY=[value]	Sets StepNLoop tolerance value.	OK		
SLX=[0 or 1] SLY=[0 or 1]	Enable or disable StepNLoop Control	OK		
SLX SLY	Returns StepNLoop enable status	0 – StepNLoop Off 1 – StepNLoop On		
SSPDX[Value] SSPDY[Value]	Set speed of X/Y axis on-the-fly to [Value]	OK		
STOP	Stops both X/Y motor with deceleration	OK		
STOPX STOPY	Stops the motor with Deceleration	OK		
STORE	Store device settings and variables (V32-V63) to flash	OK		
V[0-63]	Returns value of indicated general purpose variable register	Variable value in 32 bits	-4.29G	4.29G
V[0-63]=[Value]	Sets value to general purpose variable register	OK	-4.29G	4.29G
VER	Returns Version	Firmware Version		
X[Value]	Individual move command. If in ABS mode, move to position *value*. If in INC mode, increase position by *value.*	OK		
Y[Value]	Individual move command. If in ABS mode, move to position *value*. If in INC mode, increase position by *value.*	OK		
ZH+	Index Homing both X/Y in Positive	OK		

ZH-	Index Homing both X/Y in Negative	OK		
ZHX+ ZHY+	Index Homing in Positive	OK		
ZHX- ZHY-	Index Homing in Negative	OK		

9. 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
ABORTX             ;***Stop the X-axis immediately.
```

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
```

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
```

DI

Description:

Read: Gets the digital input value

Performax 2ED has 8 digital inputs

Syntax:

Read: [variable] = DI

Conditional: IF DI=[variable]
ENDIF

IF DI=[value]
ENDIF

Examples:

```
IF DI=255
    DO=1        ;***If all digital inputs are triggered, set DO=1
ENDIF
```

DI[1-8]

Description:

Read: Gets the digital input value

Performax 2ED 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 2ED 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 2ED 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
```

ECLEAR[axis]

Description:

Write: Clears error status

Syntax:

Write: ECLEAR[axis]

Examples:

```
ECLEARX         ;***Clears error of axis X
ECLEARY         ;***Clears error of axis Y
```

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
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
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 2ED has 2 enable outputs.

Syntax:

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

Conditional:
```
IF EO=[variable]
ENDIF

IF EO=[value]
ENDIF
```

Examples:

```
EO=3                ;***Turn all 2 bits of enable outputs
IF V1=1
    EO=V2           ;***Enable output according to variable 2
ENDIF
```

EO[1-2]

Description:

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

Performax 2ED has 4 enable outputs.

Syntax:

Read: [variable] = EO[1-2]
Write: EO[1-2] = [0 or 1]
EO[1-2] = [variable]

Conditional:
```
IF EO=[variable]
ENDIF

IF EO=[value]
ENDIF
```

Examples:
```
EO1=1          ;***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
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
HOMEY-      ;***Homes Y 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 250
```

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

Syntax:

Write: JOYENA=[0,1]

Examples:

```
JOYENA=1    ;***Enable joystick feature on X axis only
```

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
JOYHSY=20000      ;***High speed of Y 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
JOYDELY=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
```

JOYTOL[axis]

Description:

Write: Set zero tolerance value for joystick control

Syntax:

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

Examples:

```
JOYTOLX=10        ;*** zero tolerance value of x-axis set to 10
```

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 300,000.

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 300,000.

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:

```
LSPDX=1000 ;***Sets the X low speed to 1,000 pulses/sec
V1=500     ;***Sets the variable 1 to 500
LSPDX=V1   ;***Sets the X low speed to variable 1 value of 500
```

MST[axis]

Description:

Command: Get motor status of axis

Syntax:

MST[Axis]

Examples:

```
IF MSTX=0
        DIO=6
ELSEIF MSTY=0
        DIO=3
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.
```

SL[axis]

Description:

Write: Set individual StepNLoop enable.

Range is from 0 or 1

Syntax:

Write: SL[axis]=[0 or 1]

Examples:

```
SLX=1        ;***Enables StepNLoop control for the X axis.
SLY=0        ;***Disables StepNLoop control for the Y axis.
```

SLS[axis]

Description:

Command: Get the StepNLoop status of axis

Syntax:

```
SLS[Axis]
V[Value] = SLS[Axis]
```

Examples:

```
IF SLSX=0
        DIO=6
ELSEIF SLSY=0
        DIO=3
ENDIF
```

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 9

Syntax:

Write: SSPDM[axis]=[0-9]
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
```

STORE

Description:

Store device settings and the second half of variables (V32-V63) to flash.

Syntax:

STORE

Example:

```
V32=100
V33=200
STORE       ;***Values of V1 and V2 will now be preserved after power cycle
```

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
```

V

Description:

Assign to variable.
Performax 2ED 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
```

Note: On the STORE command, the second half of general purpose variable registers (V32-V63) are stored to flash. Their values will be preserved after power cycle.

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
X2000Y3000    ;***Move X to 2000 and Y to 3000 in linear interpolation move
V10 = 1200    ;***Set variable 10 value to 1200
YV10          ;***Move Y 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
ZHOMEY-    ;***Z Homes Yaxis 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
ZOMEY-      ;***Homes Y axis in negative direction
```

Contact Information

Arcus Technology, Inc.

3061 Independence Dr. Suite H,
Livermore, CA 94551
925-373-8800

www.arcus-technology.com