

---

# Performax 4ET – SA

## Advanced 4-Axis Ethernet Stepper Motion Controller Standalone Version

### Manual



COPYRIGHT © 2008 ARCUS,  
ALL RIGHTS RESERVED

First edition, June 2008

ARCUS TECHNOLOGY copyrights this document. You may not reproduce or translate into any language in any form and means any part of this publication without the written permission from ARCUS.

ARCUS makes no representations or warranties regarding the content of this document. We reserve the right to revise this document any time without notice and obligation.

**Revision History:**

- 1.01 – 1<sup>st</sup> Release
- 1.04 – 2<sup>nd</sup> Release
- 1.05 – 3<sup>rd</sup> Release

**Firmware Compatibility:**

†V225BL

**Software Compatibility:**

V126

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

## Table of Contents

1. Introduction	7
Features	7
Applications	7
Model Numbers	8
Top Board Options	8
2. Electrical Specifications	9
Controller Power Requirement	9
Driver Power Requirement †	9
Temperature Ratings †	9
Pulse, Dir, Enable Outputs	9
Digital Inputs	9
Digital Outputs	9
Alarm Input	9
3. Dimensions	10
4. Connections	11
2-pin Connector (5.08mm)	11
DIO 10-Pin Connector (3.81mm)	11
DIO 8-Pin Connector (3.81mm)	12
Motion Inputs 14-Pin Connector (3.81mm)	13
Encoder Signals 8-Pin Connectors (3.81mm)	14
TBS Axis Signals 8-Pin Connectors (3.81mm)	14
† TB9 Axis Signals DB9 Connectors	15
†TB9 2-pin Connector (5.08mm)	15
PMX-4ET-SA Interface Circuit	16
Pulse, Direction, and Enable Outputs	17
Limit, Home, and Digital Inputs	18
Digital Outputs	18
5. Getting Started	19
Typical Setup	19
Windows GUI features	20
Selecting Communication & Program	20
Main Control Screen	21
DXF Converter	33
DXF Converter – Important Notes:	38
Graphical Programmer	41
6. Motion Control Feature Overview	51
Motion Profile	51
Pulse Speed	52
On-The-Fly Speed Change	52
Motor Status	53
Individual/Linear Interpolation Moves	54
Circular Interpolation Moves	54
Arc Interpolation Moves	55
Buffered Linear Interpolation Moves	56
On-The-Fly Target Position Change	57

Homing	57
MODE 0 : Home Input Only (High speed only)	57
MODE 1 : Limit Only	58
MODE 2 : Home and Z-index	58
MODE 3 : Z-index only	59
MODE 4 : Home Input Only (High speed and low speed)	59
Jogging	60
Stopping	60
Polarity	60
Motor Position	60
Limits and Alarm	61
Digital Inputs/Outputs and Enable Outputs	61
Sync Outputs	63
StepNLoop Closed Loop Control	64
Device IP Address	66
Standalone Programming	67
Timer Register	68
Communication Time-out Feature (Watchdog)	68
Boot-up Sequence	69
Hard Reset (Flash Memory)	69
Storing to Flash	71
7. Ethernet Communication Protocol	72
Socket Settings	72
ASCII Protocol	72
8. ASCII Language Specification	73
Error Codes	78
9. Standalone Language Specification	79
;	79
ABORT	79
ABORT[axis]	79
ABS	79
ACC	80
ACC[axis]	80
ARC	80
CIR	81
DELAY	81
DI	81
DI[1-8]	82
DO	82
DO[1-8]	82
E[axis]	83
ECLEAR[axis]	83
ELSE	83
ELSEIF	84
END	85
ENDIF	85

ENDSUB	85
ENDWHILE	86
EO	86
EO[1-4]	86
GOSUB	87
HLHOME[axis][+ or -]	87
HOME[axis][+ or -]	87
HSPD	88
HSPD[axis]	88
IF	88
INC	89
JOG[axis]	89
LHOME[axis][+ or -]	89
LSPD	90
LSPD[axis]	90
MST[axis]	91
P[axis]	91
PS[axis]	91
SCV[axis]	92
SL[axis]	92
SLS[axis]	92
SR[0,3]	93
SSPD[axis]	93
SSPDM[axis]	93
STOP	94
STOP[axis]	94
STORE	94
SUB	95
SYNCFG[axis]	95
SYNOFF[axis]	95
SYNON[axis]	96
SYNPOS[axis]	96
SYNSTAT[axis]	96
SYNTIME[axis]	96
TOC	96
TR	96
U	97
V[index]	97
WAIT[axis]	98
WHILE	98
X	99
Y	99
Z	100
ZHOME[axis][+ or -]	100
ZOME[axis][+ or -]	100
10. Example Standalone Programs	101

---

Standalone Example Program 1 – Single Thread _____	101
Standalone Example Program 2 – Single Thread _____	101
Standalone Example Program 3 – Single Thread _____	101
Standalone Example Program 4 – Single Thread _____	102
Standalone Example Program 5 – Single Thread _____	102
Standalone Example Program 6 – Single Thread _____	103
Standalone Example Program 7 – Multi Thread _____	104
Standalone Example Program 8 – Multi Thread _____	105
Appendix A: Speed Settings _____	106
Acceleration/Deceleration Range _____	106
Acceleration/Deceleration Range – Positional Move _____	107

# 1. Introduction

PMX-4ET-SA is an advanced 4 axis stepper stand-alone programmable motion controller.

Communication to the PMX-4ET-SA can be established over Ethernet. It is also possible to download a stand-alone program to the device and have it run independent of a host.

## **Features**

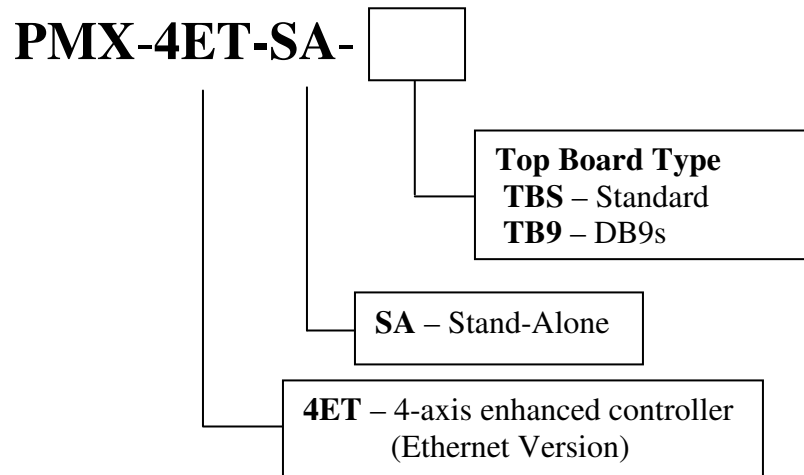
### **PMX-4ET-SA**

- Ethernet 10Mbps communication using Arcus ASCII command
- Standalone programmable
- Maximum pulse output rate of 6M PPS
- Trapezoidal or s-curve acceleration
- On-the-fly speed change
- XYZU linear coordinated motion
- XY circular coordinated motion
- XY arc coordinated motion
- Continuous linear coordinated buffered move for XYZ axes for smooth move control. Buffer size is 36.
- A/B/Z differential encoder inputs [Max frequency of 5 MHz]
  - StepNLoop closed loop control (position verification)
- Opto-isolated I/O
  - 8 x inputs [4 x high speed position capture latch input]
  - 8 x outputs [4 x synchronous output]
  - +Limit/-Limit/Home inputs per axis
- Homing routines:
  - Home input only (high speed)
  - Home input only (high speed + low speed)
  - Limit only
  - Z-index encoder channel only
  - Home input + Z index encoder channel

## **Applications**

- 3D CAD/CAM
- Engraving
- Laser cutting
- Dispensing

## Model Numbers



### ***Top Board Options***

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

#### **Standard Top Board (PMX-4ET-SA-TBS)**

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

#### **DB9 Top Board (PMX-4ET-SA-TB9)**

DB9 Top Board consists of DB9 headers for X/Y/Z/U pulse/dir/enable outputs. The DB9 headers on these top boards are pin-to-pin compatible with the Arcus series motor + driver (DMX-A2-DRV).

### **Contacting Support**

For technical support contact: [support@arcus-technology.com](mailto:support@arcus-technology.com).

Or, contact your local distributor for technical support.

## 2. Electrical Specifications

### **Controller Power Requirement**

Regulated Voltage: **+12 to +24 VDC**  
 Current (Max): **1.5A (Peak)**

### **Driver Power Requirement †**

Regulated Voltage: **+12 to +48 VDC**  
 Current (Max): **3.0A (Peak)**

† Applicable to TB9 top board option + DMX-A2-DRV driver combination only

### **Temperature Ratings †**

Operating Temperature: **-20°C to +80°C**  
 Storage Temperature: **-55°C to +150°C**

† Based on component ratings

### **Pulse, Dir, Enable Outputs**

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

### **Digital Inputs**

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

### **Digital Outputs**

Type: **Opto-isolated Darlington outputs**  
 Max voltage: **+12V to +24VDC**  
 Max source current: **90 mA**

### **Alarm Input**

Type: **TTL**  
 Voltage: **+0 to +5VDC**

### 3. Dimensions

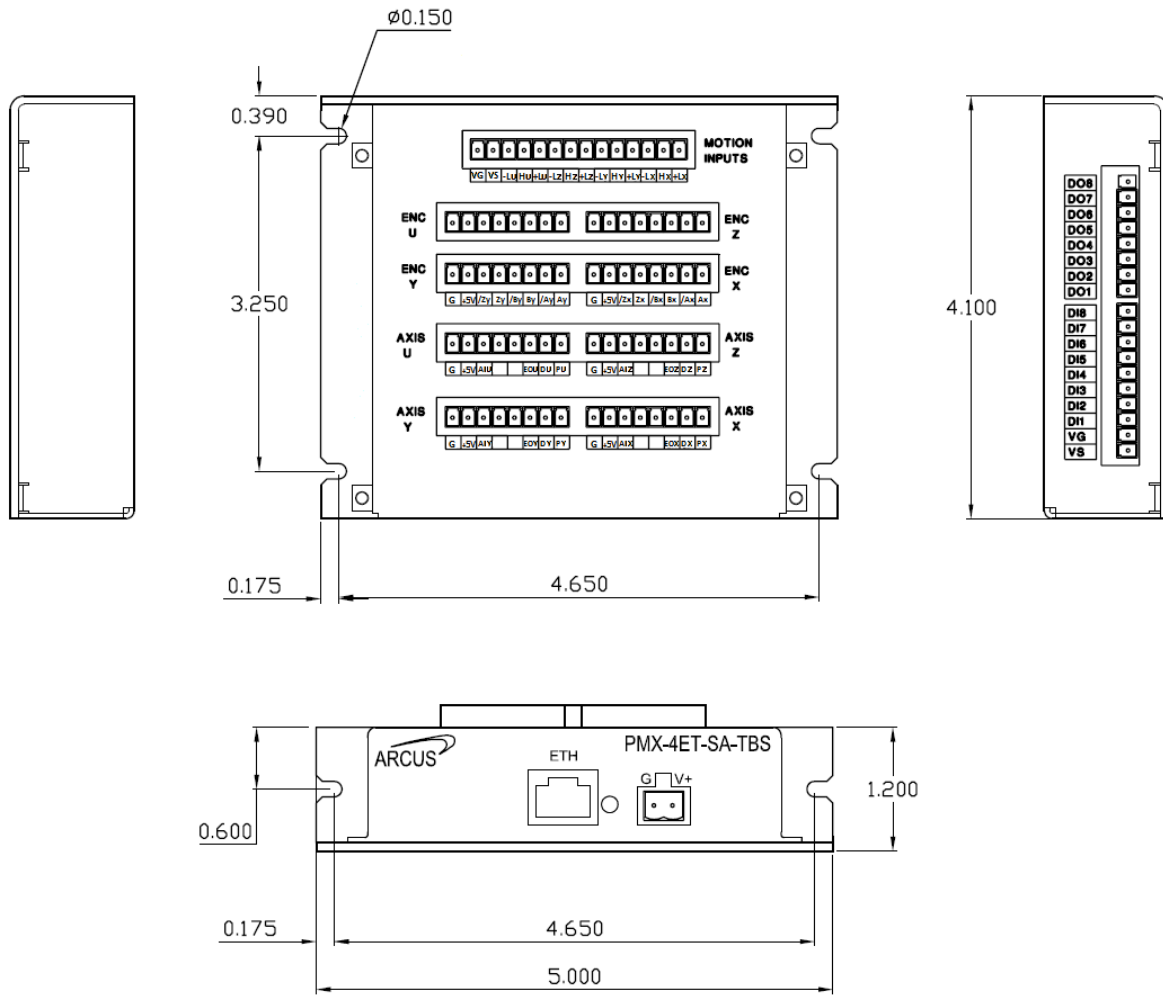


Figure 3.0

**Note:** The image above is of PMX-4ET-SA-TBS. The dimensions of PMX-4ET-SA-TB9 are the same, with the exception of the top board connectors.

## 4. Connections

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

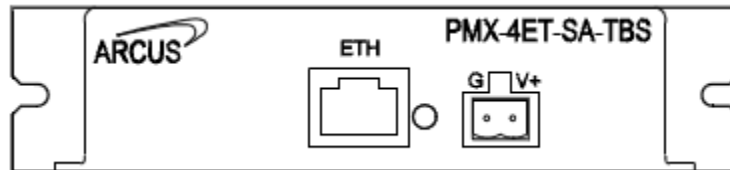


Figure 4.0

### 2-pin Connector (5.08mm)

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

Table 4.0

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

† Other 5.08mm compatible connectors can be used.

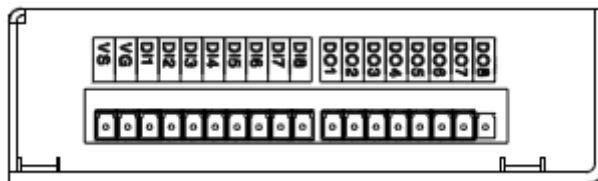


Figure 4.1

### DIO 10-Pin Connector (3.81mm)

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

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

Table 4.1

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

† Other 3.81 compatible connectors can be used.

***DIO 8-Pin Connector (3.81mm)***

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

Table 4.2

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

† Other 3.81 compatible connectors can be used.

### Motion Inputs 14-Pin Connector (3.81mm)

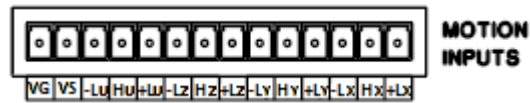


Figure 4.2

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

Table 4.3

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

† Other 3.81 compatible connectors can be used.

### Encoder Signals 8-Pin Connectors (3.81mm)



Figure 4.3

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

Table 4.4

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

† Other 3.81 compatible connectors can be used.

### TBS Axis Signals 8-Pin Connectors (3.81mm)

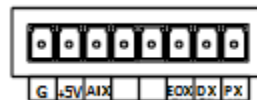


Figure 4.4

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

Table 4.5

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

† Other 3.81 compatible connectors can be used.

### † TB9 Axis Signals DB9 Connectors

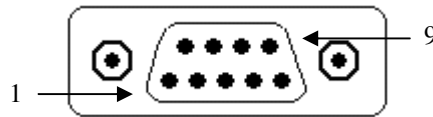


Figure 4.5

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

Table 4.6

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

### †TB9 2-pin Connector (5.08mm)

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

Table 4.7

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

† Other 5.08mm compatible connectors can be used. There are two separate driver power inputs. The right side is for axes X + Z. The left side is for axes Y + U.

## PMX-4ET-SA Interface Circuit

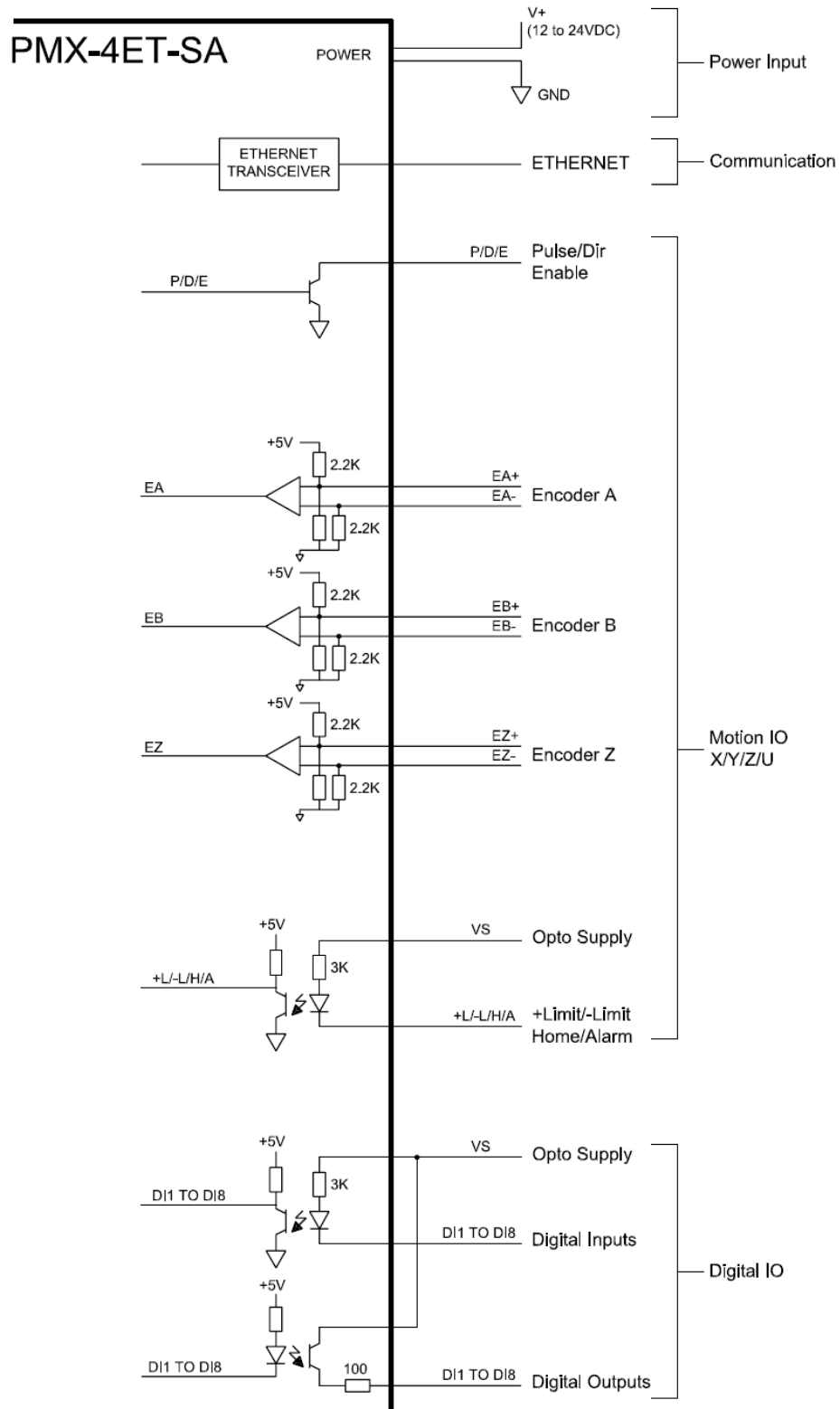


Figure 4.6

## Pulse, Direction, and Enable Outputs

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.

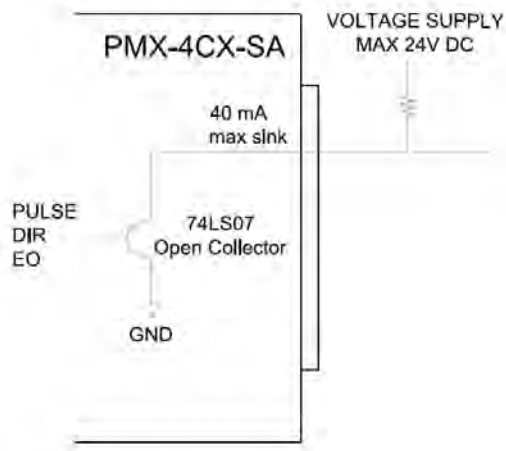


Figure 4.7

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

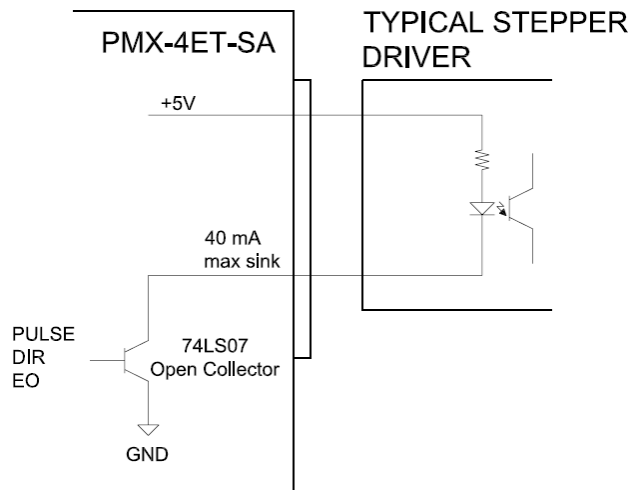


Figure 4.8

### Limit, Home, and Digital Inputs

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.

Limit and home inputs require a separate opto-supply which can be found on the 3.8mm, 14-pin connector. If desired, these opto-supplies can be tied together.

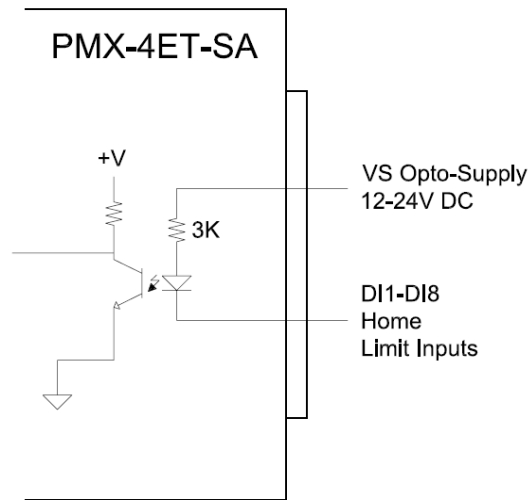


Figure 4.9

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

### Digital Outputs

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

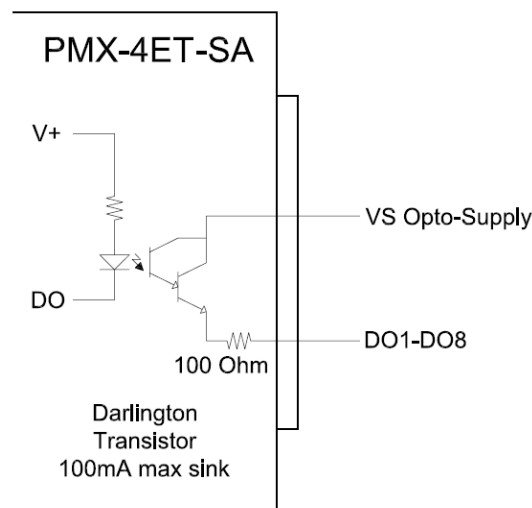


Figure 4.10

# 5. Getting Started

## Typical Setup

### Point-to-point

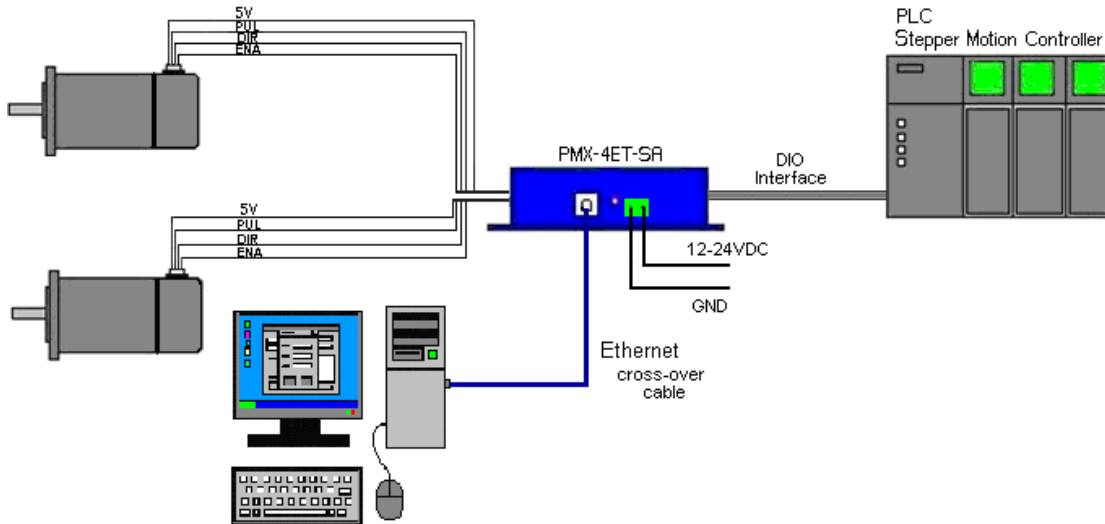


Figure 5.0

### Network-based

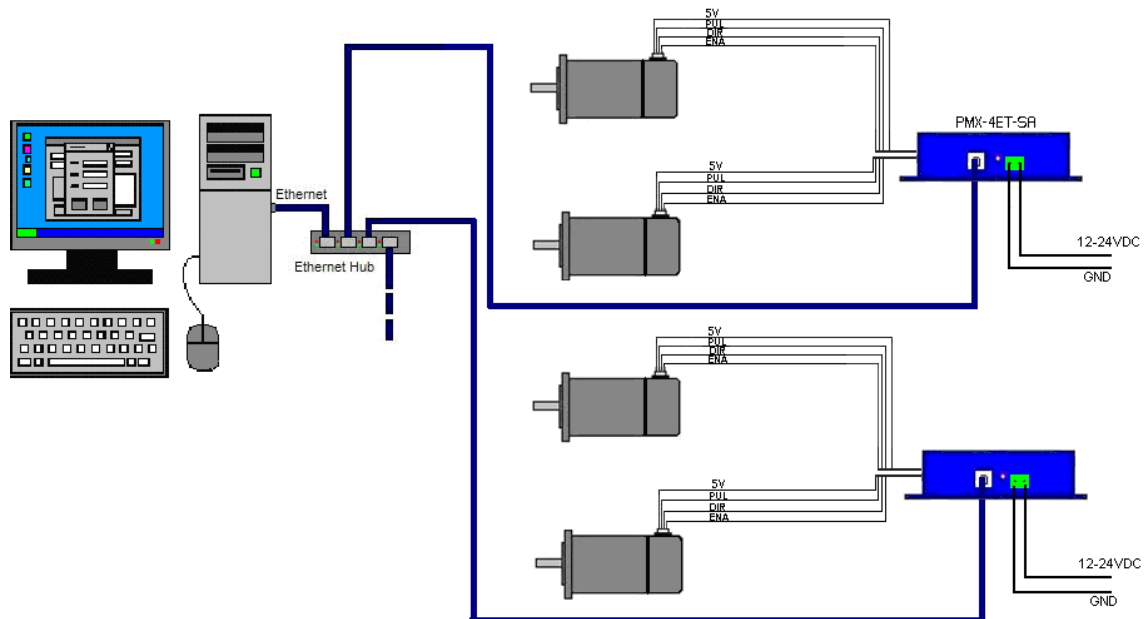


Figure 5.1

## Windows GUI features

PMX-4ET-SA comes with a Windows GUI program to test, program, compile, download, and debug the controller. The following control types are available.

Program	Description
Program & Control	Allows the user to test all the features interactively. Also provides the interface for stand-alone programming
DXF Converter	Allows the user to load a DXF file and convert it to motion commands
Graphical Converter	Allows the user to create a stand-alone program graphically, instead of writing text-based code

Table 5.0

## Selecting Communication & Program

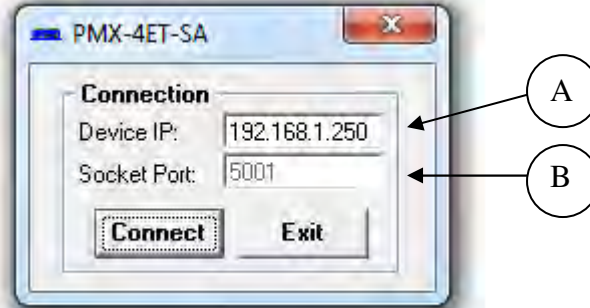


Figure 5.2

- A. Device IP of the PMX-4ET-SA. The device IP later can be changed.
- B. Port number of the socket that must be opened to being Ethernet TCP/IP communication. This socket number cannot be changed. Default is 5001.

This screen allows the user to choose between the three different program types. To use a particular program, click on the corresponding button.



Figure 5.3

## Main Control Screen

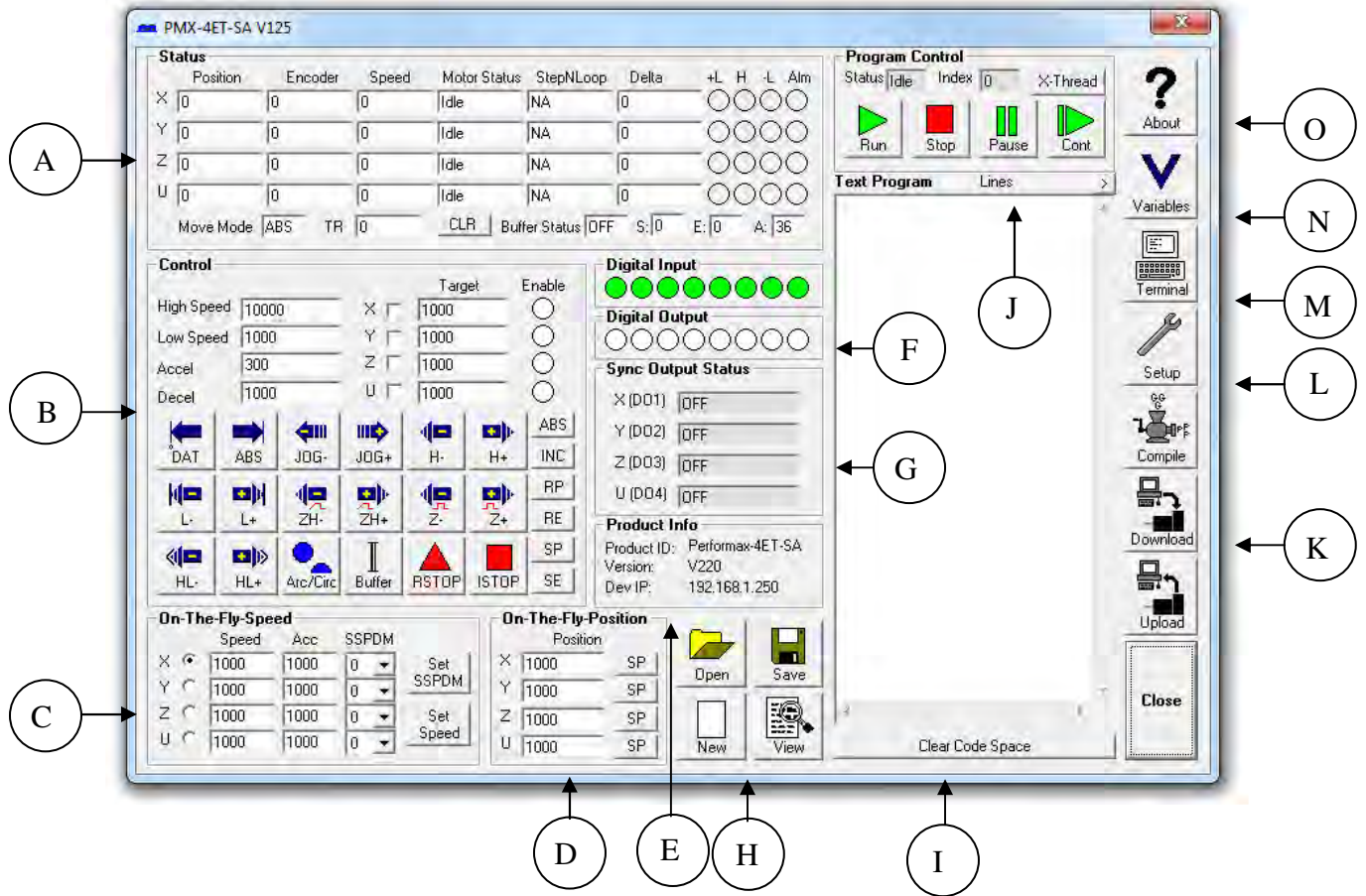


Figure 5.4

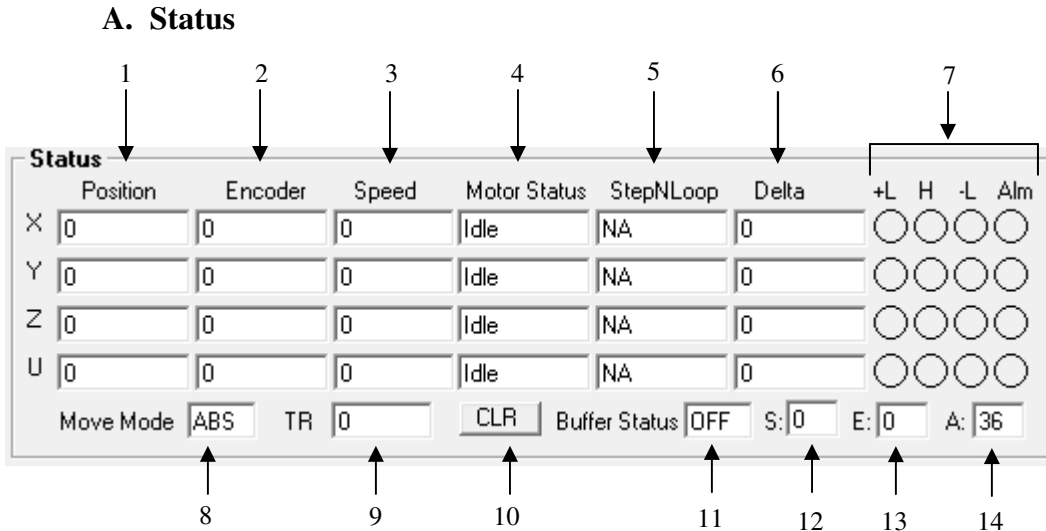


Figure 5.5

1. **Current pulse position** (X,Y,Z,U axes). If StepNLoop is enabled, this shows the real-time target position.
2. **Current encoder position** (X,Y,Z,U axes)
3. **Current speed** (X,Y,Z,U axes) pulse/sec. If StepNLoop is enabled, the speed is in encoder counts/sec, unless an interpolation move is in process.
4. **Motor status** (X,Y,Z,U axes)
  - 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
5. **StepNLoop status** - valid only when StepNLoop is enabled and displays current StepNLoop status by displaying one of the following:
  - NA – StepLNoop is disabled
  - IDLE – motor is not moving
  - MOVING – target move is in progress
  - JOGGING – jog move is in progress
  - HOMING – homing is in progress
  - Z-HOMING – homing using Z-index channel in progress
  - ERR-STALL – StepNLoop has stalled.
  - ERR-LIM – plus/minus limit error
6. **StepNLoop delta status**
7. **-Limit, + Limit, Home and Alarm input status** (X,Y,Z,U axes)
8. **Move mode status**
  - i. ABS – absolute move
  - ii. INC – incremental move
9. **Timer register status** (counts down)

10. **Clears** any limit or StepNLoop error
11. **Buffer move enable status**
12. **Buffer start:** This is the current index of the buffer. Note that the buffer is a 36 position ring buffer. (Used for buffer move mode only)
13. **Buffer end:** This is the current end of the buffer. Note that the buffer is a 36 position ring buffer. (Used for buffer move mode only)
14. **Provides the available empty positions of the buffer** (Used for buffer move mode only)

## B. Control

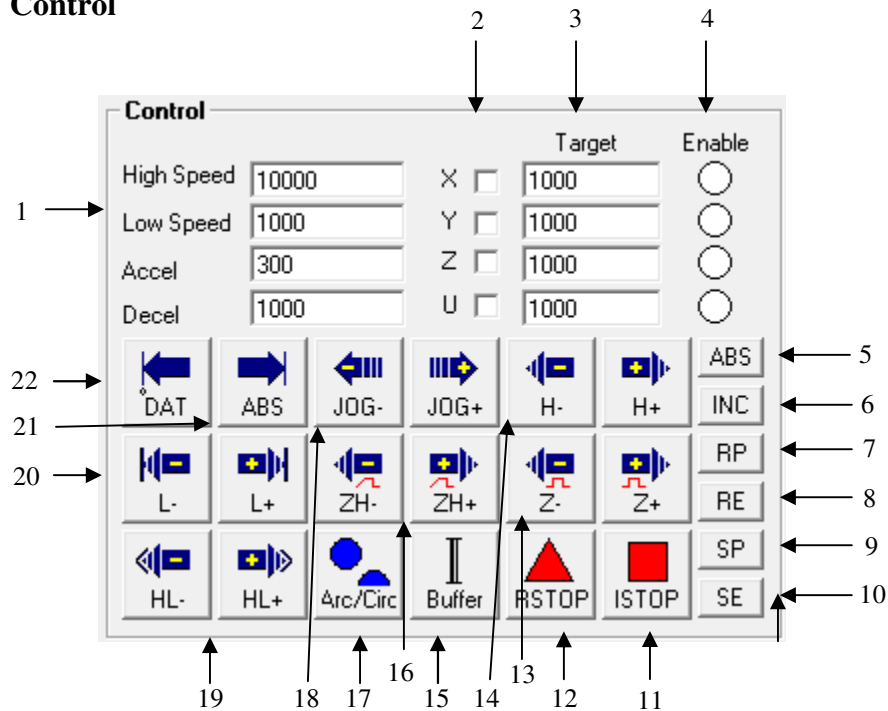


Figure 5.6

1. **Global High speed, low speed, and acceleration.** To give each axis individual speed parameters, enter HS[axis], LS[axis] and ACC[axis] commands via the command line.
2. **Select X/Y/Z/U axis to control.**
3. **Target Position** (X,Y,Z,U axes)
4. **Enable** – motor power is turned on or off by clicking on these circles (X,Y,Z,U axes)
5. **Set absolute move mode**
6. **Set incremental move mode**
7. **RP** - Reset pulse counter for the specified axis. Not allowed if StepNLoop is enabled.
8. **RE** - Reset encoder counter for the specified axis.
9. **SP** - Set pulse counter for the specified axis.
10. **SE** - Set encoder counter for the specified axis.
11. **ISTOP** – the motion is immediately stopped without deceleration.

12. **RSTOP** – the motion is stopped with deceleration.
13. **Z+/Z-**: Home the axis using only encoder index channel.
14. **H+/H-**: Home the axis at high speed using only the home sensor.
15. **Buffer move Tool** – Clicking on this button will provide the user with an interface to load buffer move commands to the PMX-4ET-SA.

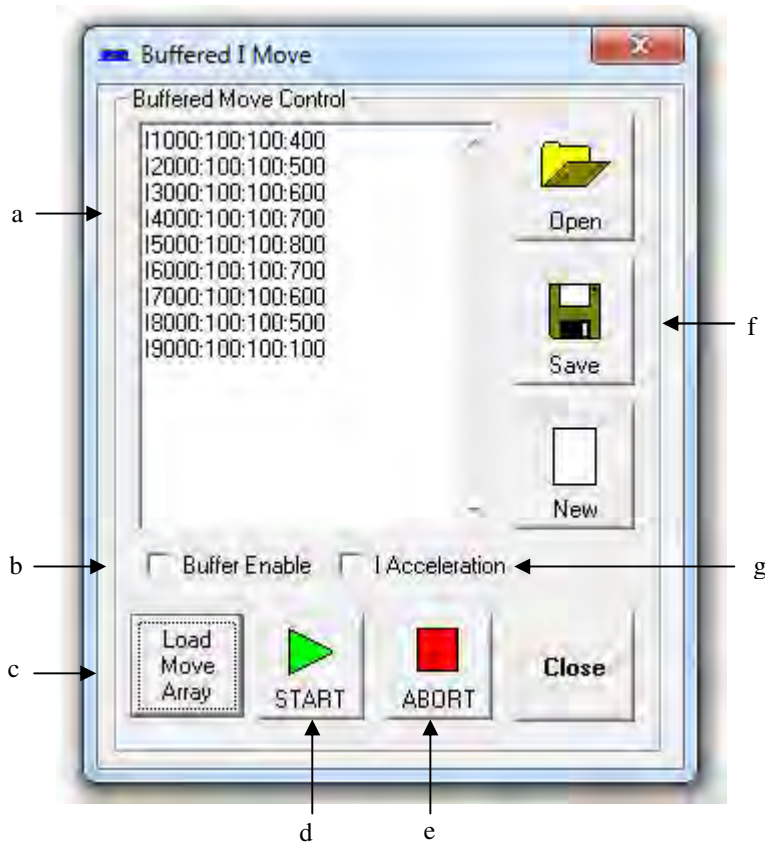


Figure 5.7

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

- f. **Open/Save/New** – Allows users to save/open or create new buffer array lists
  - g. **I Accel** – Enable/Disable buffered I move acceleration.
16. **ZH+/ZH-**: Home sensor and encoder index channel is used to home.
17. **Arc/Circle Tool** – Clicking on this button will provide the user with an interface to perform Arc/Circle XY moves.

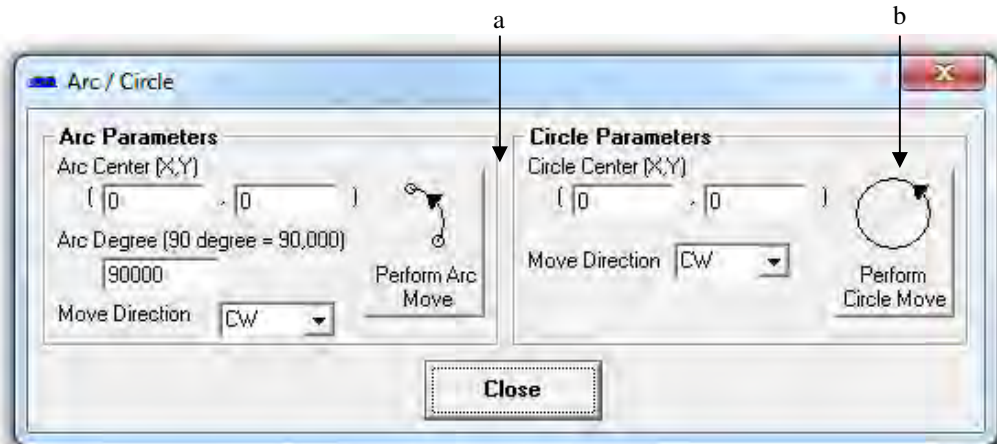


Figure 5.8

- a. **Perform Arc Move** – Once the arc center/degree/move direction parameters are set, clicking on this button will begin the arc move.
- b. **Perform Circle Move** – Once the circle center/move direction parameters are set, clicking on this button will begin the circle move.

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

- 18. **JOG+/JOG-**: jogs the motor in positive and negative direction.
- 19. **HL+/HL-**: Home the axis at high speed and low speed using only the home sensor.
- 20. **L+/L-**: Home the axis using the limit sensor.
- 21. **ABS**: Perform absolute move. If more than one axis is selected, an interpolated move will result.
- 22. **DAT**: Return to 0 position. If more than one axis is selected, an interpolated move will result.

### C. On-The-Fly-Speed Control

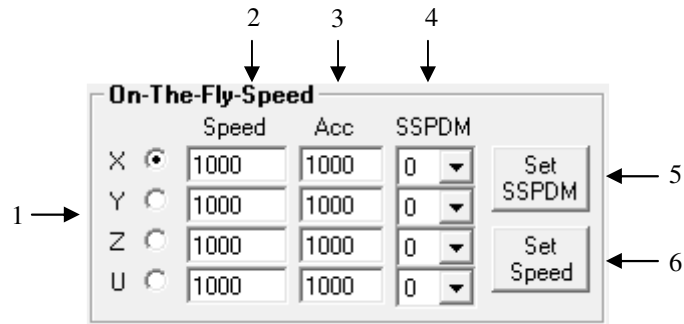


Figure 5.9

1. **Select X/Y/Z/U axis.**
2. **Select destination speed** of the axis.
3. **Select the acceleration** used during an on-the-fly speed change.
4. **Select the SSPD mode** for the axis. See On-The-Fly Speed section for details.
5. **Set the SSPD mode** the axis.
6. **Set on-the-fly speed change.** Acceleration will be taken from the “Acc” field. Make sure that the SSPDM mode has been set before issuing the on-the-fly speed operation.

### D. On-The-Fly-Position Control

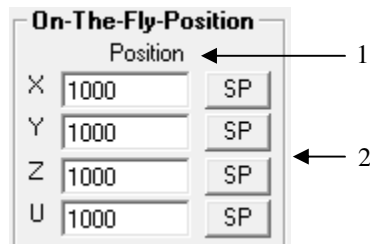


Figure 5.10

1. Set the new target position for the specified axis.
2. **SP** - Perform and on-the-fly position change.

### E. Product Information

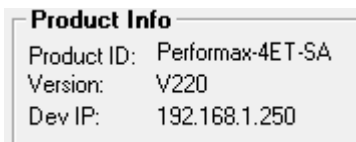


Figure 5.11

## F. Digital Input/Output

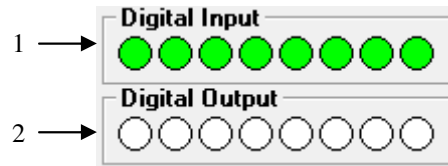


Figure 5.12

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

## G. Sync Outputs

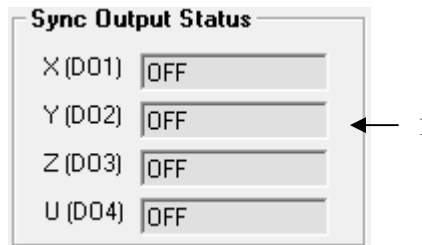


Figure 5.13

1. **Sync output status** for DO1-DO4.
  - i. OFF
  - ii. WAITING
  - iii. TRIGGERED

## H. Program File Control

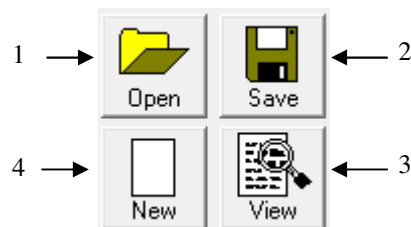


Figure 5.14

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

## I. Standalone Program Editor

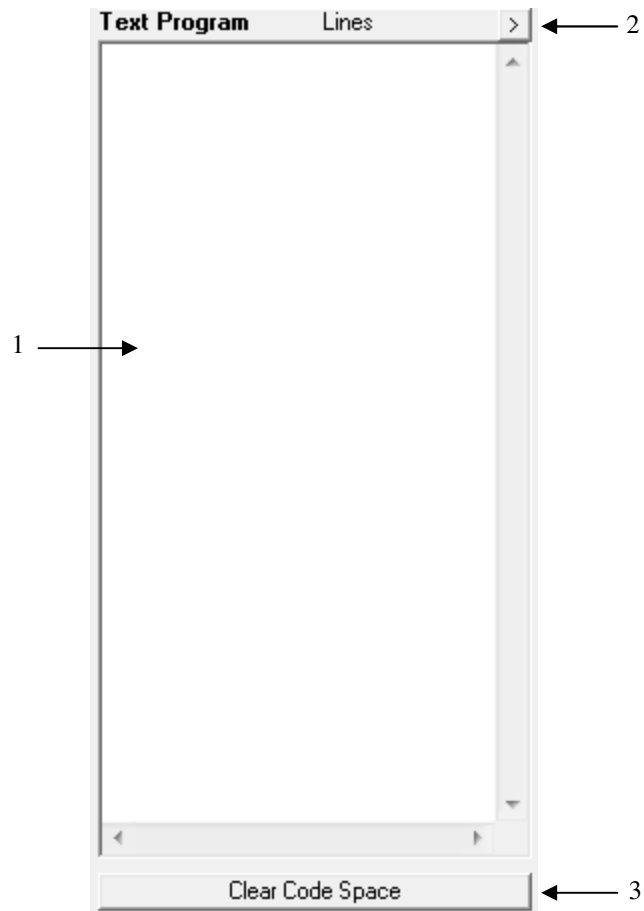


Figure 5.15

1. **Text Program** – Text box for writing and editing a standalone program.
2. Opens a larger Program Editor window for easier programming.
3. **Clear Code Space** – Clear the code space on the PMX-4ET-SA.

## J. Standalone Program Control

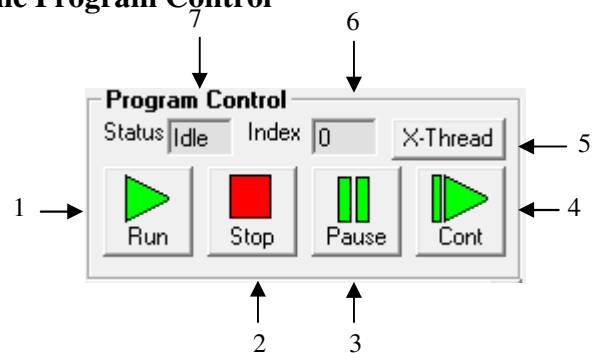


Figure 5.16

1. **Run** – Standalone program is run.
2. **Stop** – Program is stopped.
3. **Pause** – Program that is running can be stopped.
4. **Cont** – Program that is paused can be continued
5. **XThread** - Open the Standalone Program Control for all standalone programs.
6. **Index** – Current line of low-level code that is being executed.
7. **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.

### K. Standalone Program Compile/Download/Upload

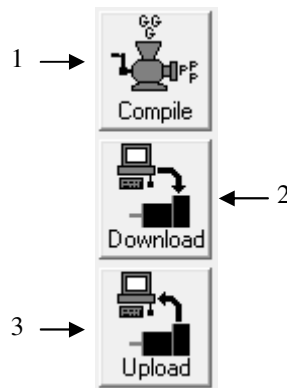


Figure 5.17

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

## L. Setup

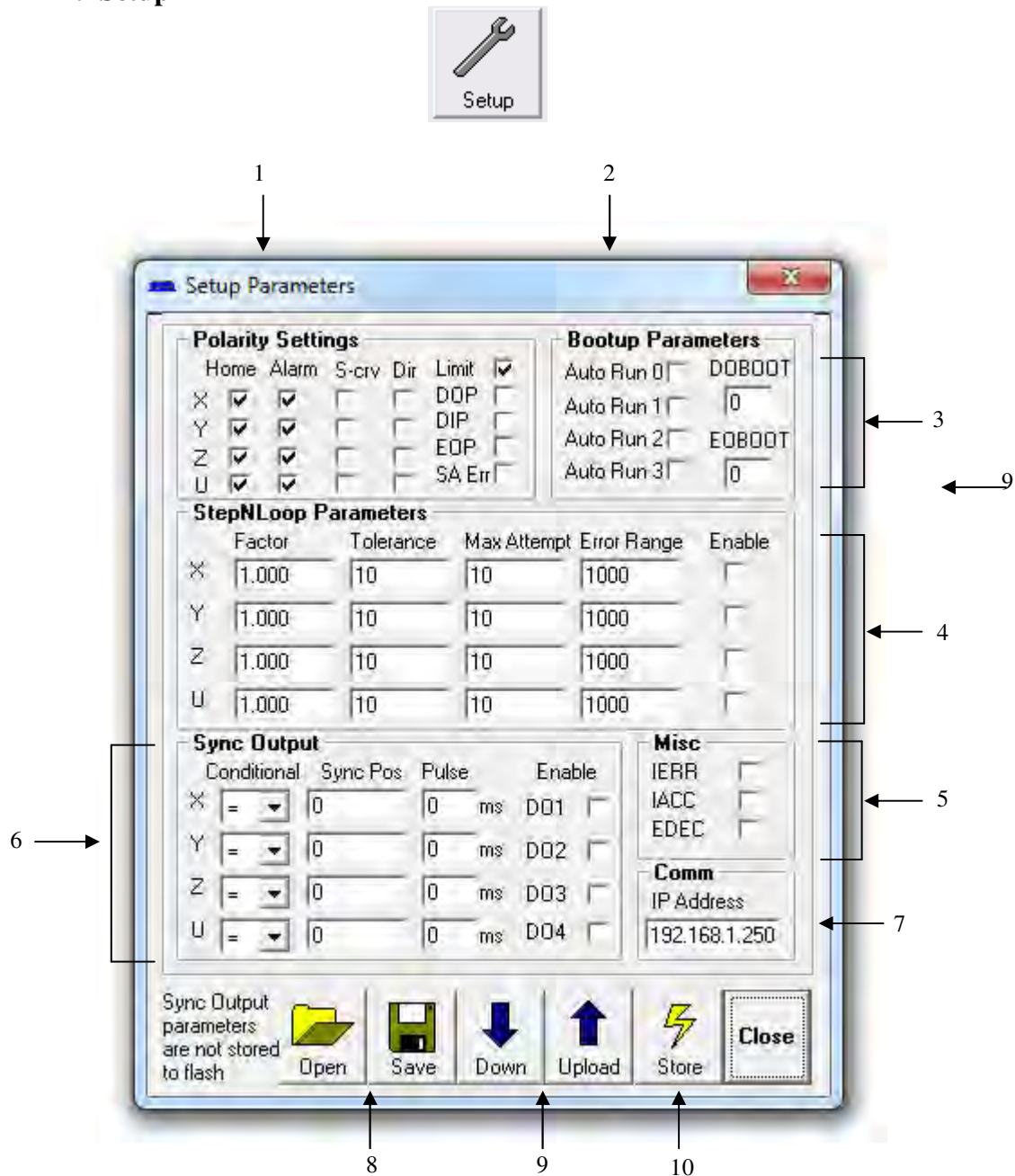


Figure 5.18

### 1. Polarity/S-curve:

- Set home/alarm/dir polarity for X/Y/Z/U axes. Note that limit polarity is fixed either high/low for ALL axes.
- Set s-curve enable/disable for each axes.
- DOP** - Set the digital output polarity
- DIP** - Set the digital input polarity
- EOP** - Set the enable output polarity

- f. **SA Err** - Set the return jump line for standalone error handling
2. **Auto Run** – Click and perform a store to flash to have the specified standalone program run on boot up
3. **DOBOOT/EOBOOT** - Set the digital and enable output configuration status on controller boot-up
4. **Set StepNLoop** parameters
5. **Miscellaneous Settings:**
  - a. **IERR** - Enable/disable the ignore limit/alarm error feature
  - b. **IACC** - Enable/disable I move acceleration (used with buffered I commands)
  - c. **EDEC** - Enable/disable unique deceleration.
6. **Set Sync output** parameters (Note that sync output parameters are not stored to flash memory)
7. **IP Address:**
  - a. Set IP address of the device
8. **Open/Save** parameters to file.
9. **Upload/Download** parameters to and from RAM
10. **Store** parameters to flash memory

## M. Terminal

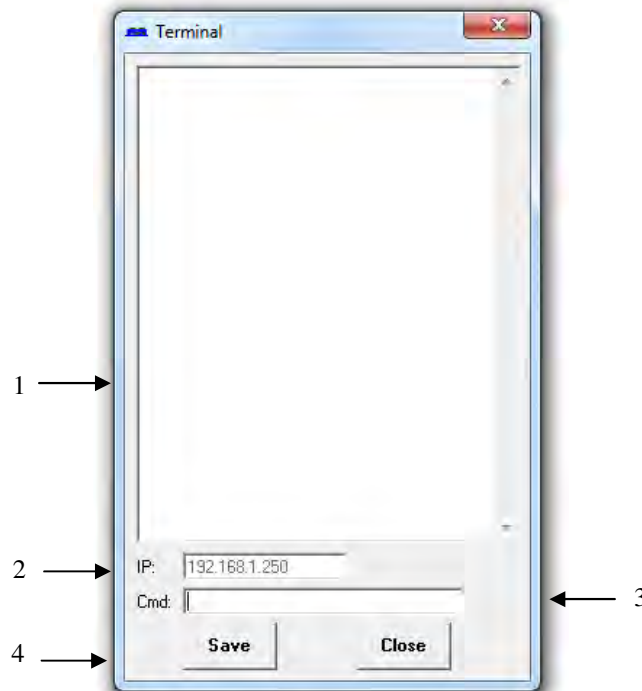
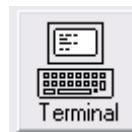


Figure 5.19

1. **Response Box** – Displays sent command as well as corresponding response
2. **Device IP** – Device IP of the PMX-4ET-SA. This IP can be changed to place many different units on an Ethernet network.
3. **Command line** – ASCII command line
4. **Save** – Save the current contents of the Response Box to file

## N. Variable Status

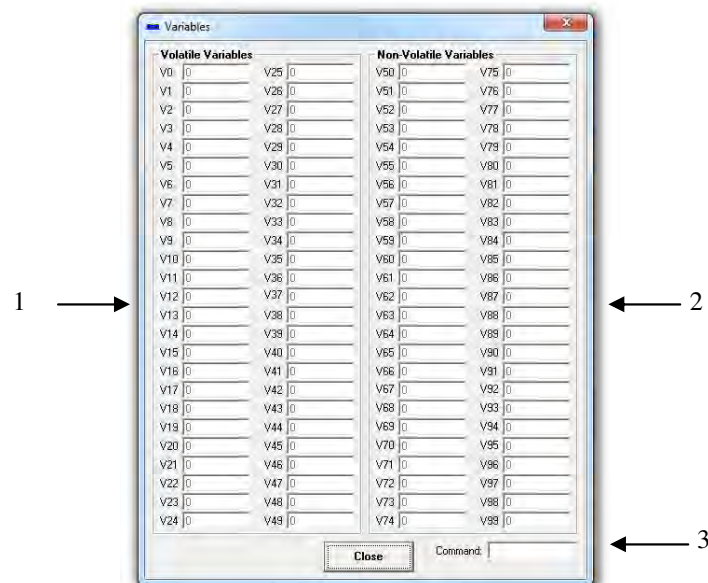


Figure 5.20

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

## O. About



Figure 5.21

Displays the current Software and Firmware versions.

## DXF Converter

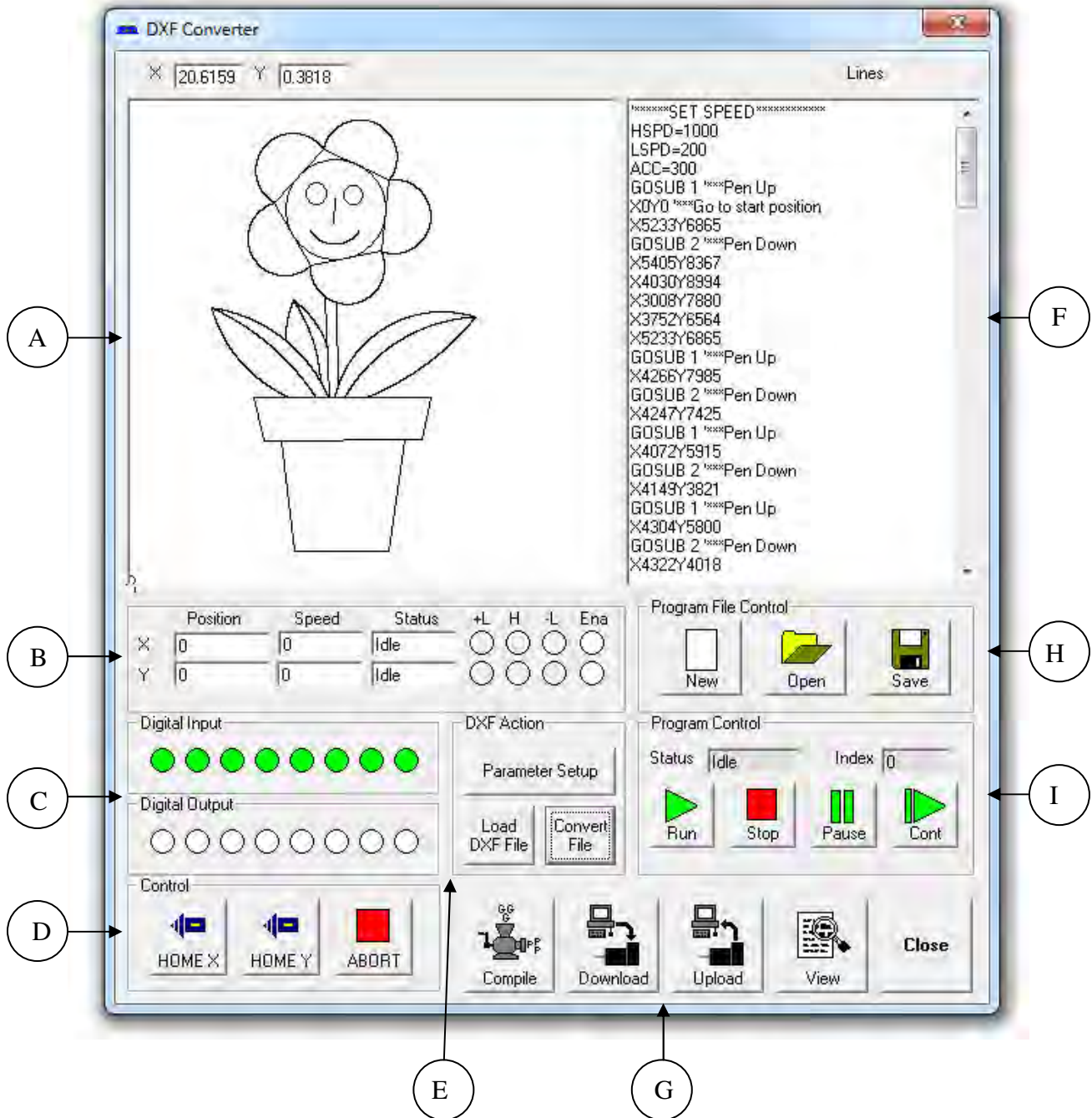


Figure 5.22

### A. DXF Viewer

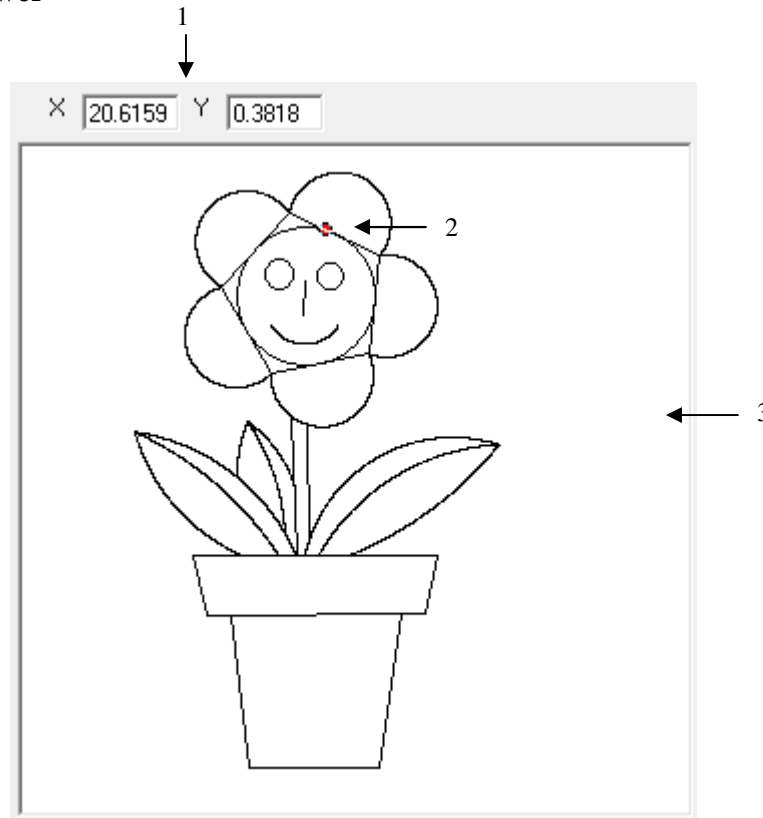


Figure 5.23

1. **Displays the (X,Y)** position of the mouse cursor within the DXF viewer box
2. **Z-axis cursor** – Whenever the Z-axis is enabled, the cursor turns the color red. Otherwise the cursor is the color white
3. **Preview of the DXF file.** For DXF preview to appear, click on “Load DXF File”

### B. Status

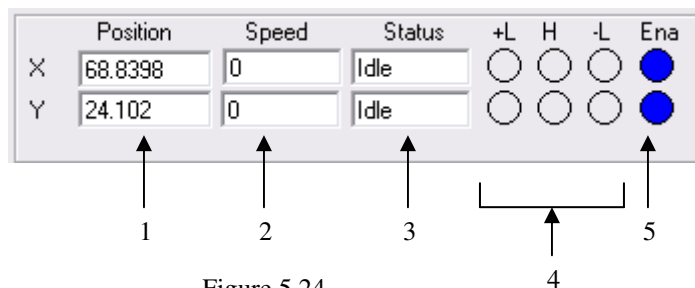


Figure 5.24

1. **Current inch/mm position (X,Y axes).**

2. **Current speed** (X,Y axes).
3. **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
4. **+Limit, -Limit, Home** status
5. **Enable** status (X,Y axes). To enable/disable the axis, click on the corresponding circle

### C. Digital Input/Output

See Section F of “Main Control Screen ”

### D. Control

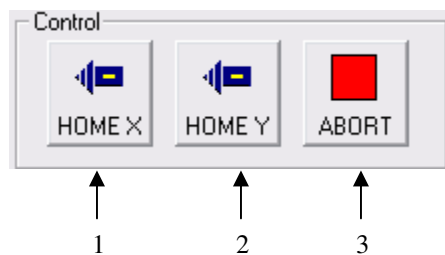


Figure 5.25

1. **Home x-axis** to the negative direction
2. **Home y-axis** to the negative direction
3. **Abort** all movement

### E. DXF Action

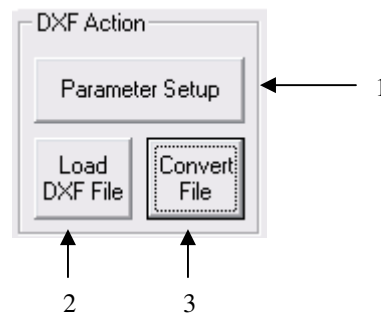


Figure 5.26

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

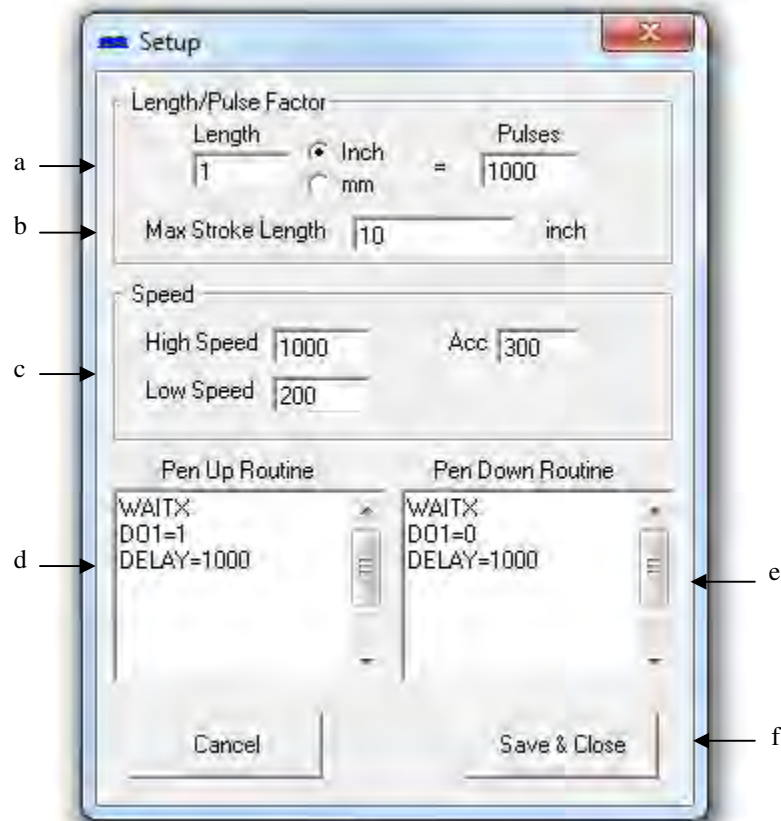


Figure 5.27

- a. Length/Pulse Factor – Select relationship between number of pulses and length of movement in terms of inch or millimeter.
  - b. Max Stroke Length – The largest allowable stroke length. This will affect the scaling of the DXF viewer box
  - c. High Speed, Low Speed and Acceleration settings
  - d. Pen Up Routine – The routine when the XY axis is **not** in position
  - e. Pen Down Routine – The routine when the XY axis is in position
  - f. Save parameters and exit setup
2. **Load DXF File** – Once the button is clicked, the following screen appears:

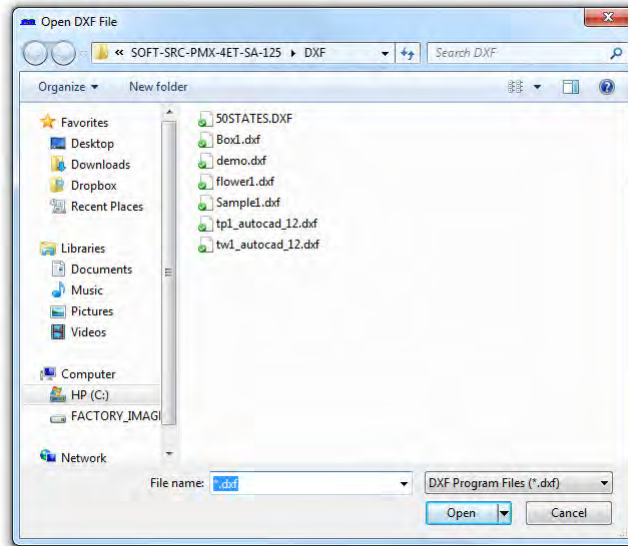


Figure 5.28

Select the desired DXF file and click “Open”. At this point, the selected DXF file will be previewed in the DXF Viewer box.

3. **Convert File** - Convert the loaded DXF file into PMX-4ET-SA compatible motion commands. The result will be loaded into the Motion Conversion Program box.

Note: The conversion scaling and speed will depend on the parameters set in the Parameter setup box.

## F. Motion Conversion Program

```

*****SET SPEED*****
HSPD=10000
LSPD=100
ACC=100
GOSUB 1 ****Pen Up
X0Y0 ****Go to start position
X26164Y34325
GOSUB 2 ****Pen Down
X27023Y41834
X20148Y44971
X15039Y39402
X18758Y32822
X26164Y34325
GOSUB 1 ****Pen Up
X21330Y39326
GOSUB 2 ****Pen Down
X21233Y37126
GOSUB 1 ****Pen Up
X20960Y29573
GOSUB 2 ****Pen Down
X20745Y19105
GOSUB 1 ****Pen Up
X21522Y28999
GOSUB 2 ****Pen Down
X21612Y20088
  
```

Figure 5.29

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 File”.

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

### G. Standalone Program Compile/Download/Upload/View

See *Section K* of “Main Control Screen”

### H. Program File Control

See *Section H* of “Main Control Screen”

### I. Program Control

See *Section J* of “Main Control Screen”

## ***DXF Converter – Important Notes:***

### Creating a compatible DXF file:

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

**Radius Size:** PMX-4ET-SA does not allow a radius larger than 134,216,773 pulses on arc or circular moves. To keep your radius moves smaller than 134,216,773, decrease the **Length/Pulse Factor**.

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

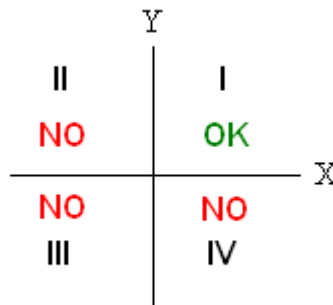


Figure 5.30

**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:

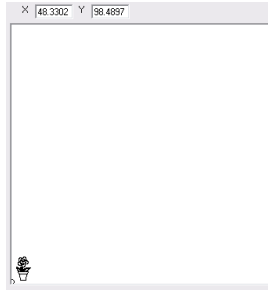


Figure 5.31

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:

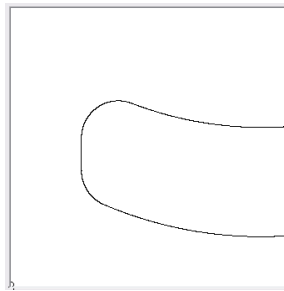


Figure 5.32

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 Auto Cad drawing that the length and width of the picture is about 100 x 100 (circled in red). In this case, the units are mm.

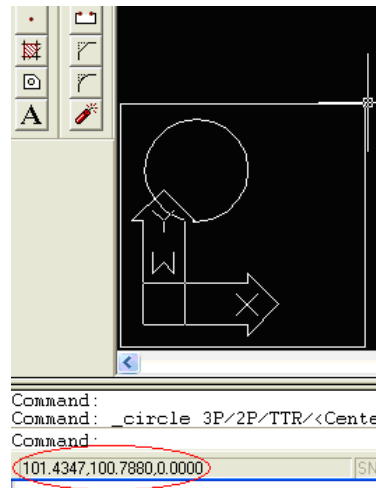


Figure 5.33

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

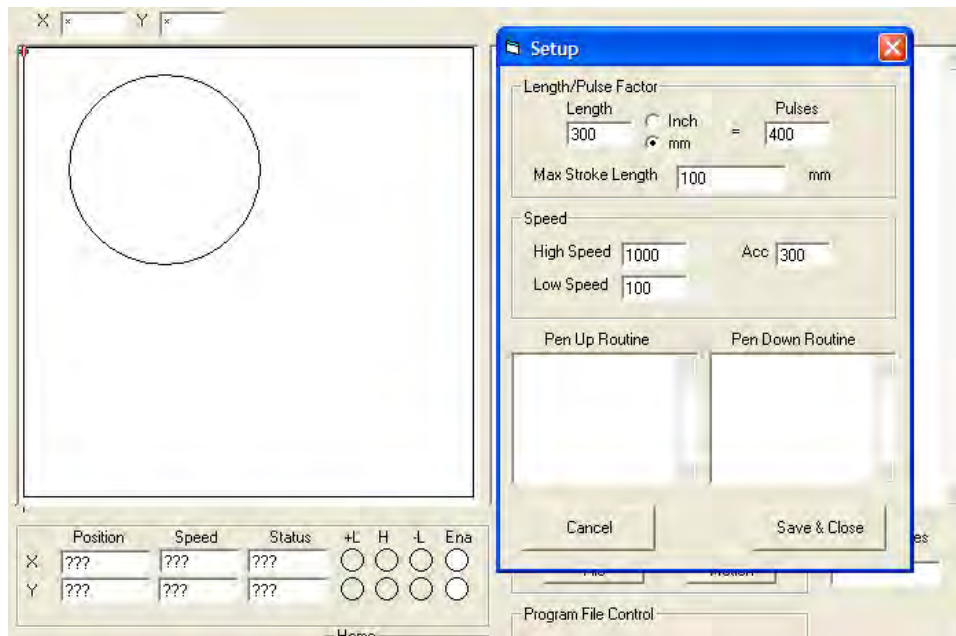


Figure 5.34

### Scaling your XY table:

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

## Graphical Programmer

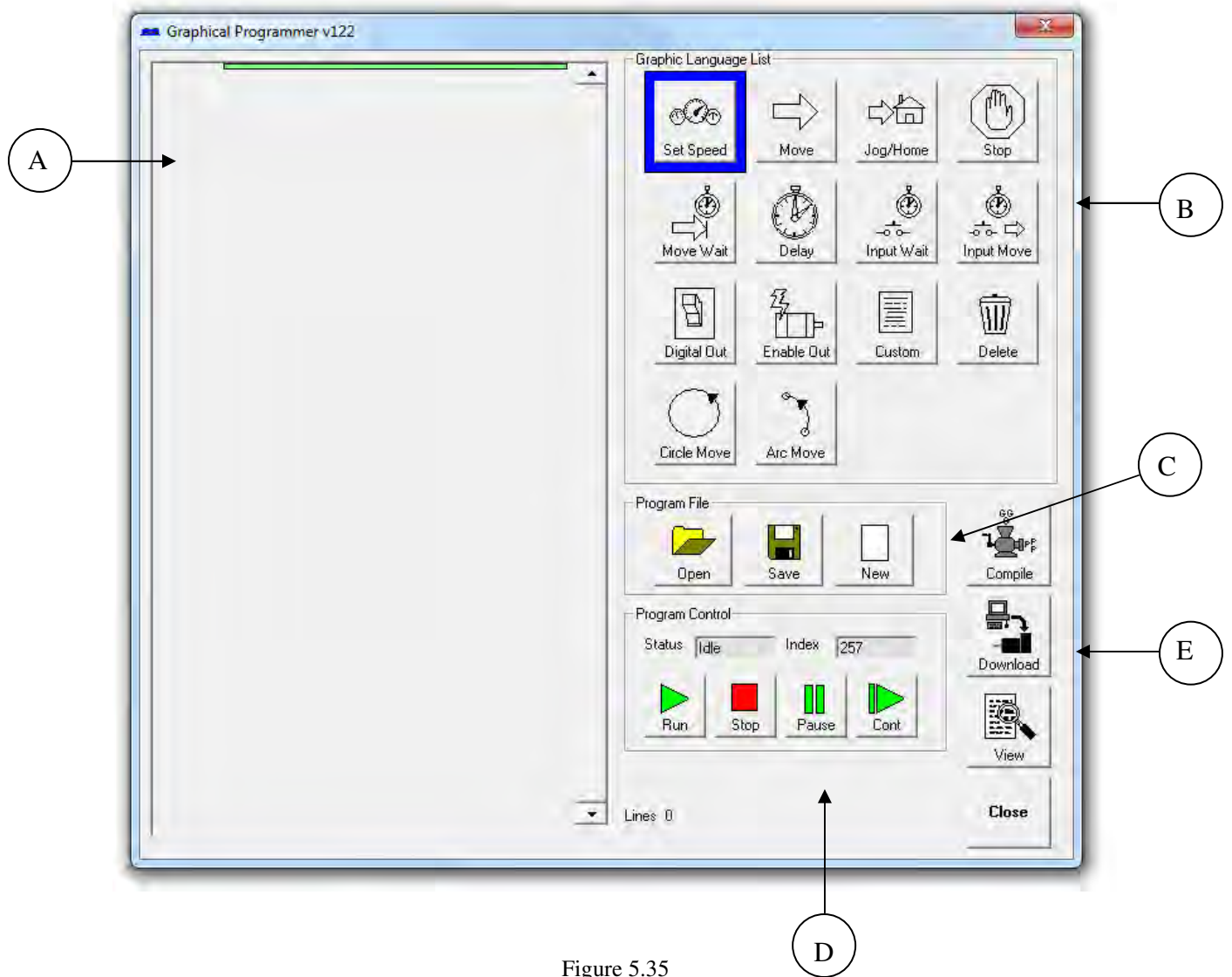


Figure 5.35

## A. While Sequence Box

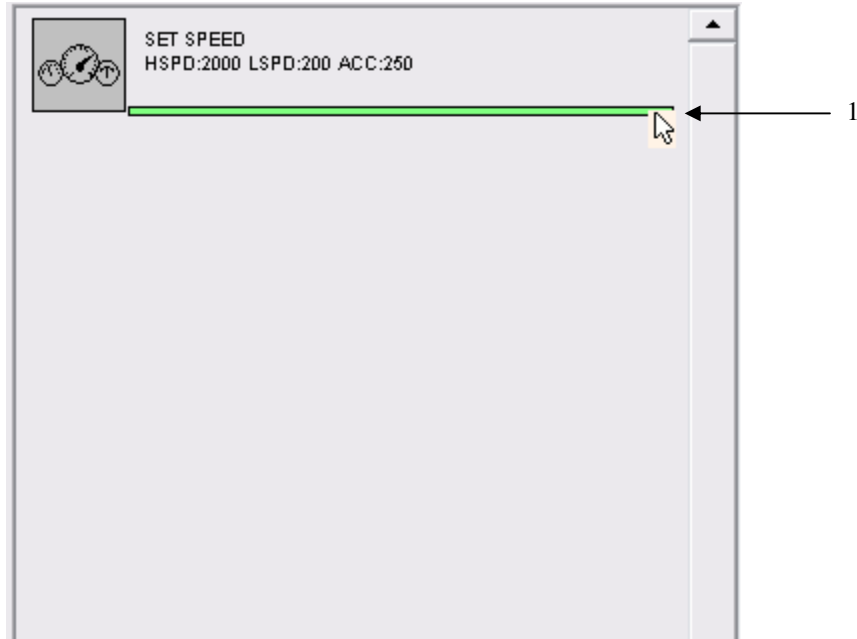


Figure 5.36

1. The following box contains the sequence that is executed in a continuous while loop. To enter processes 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 settable parameters.

## B. Graphic Language List Box

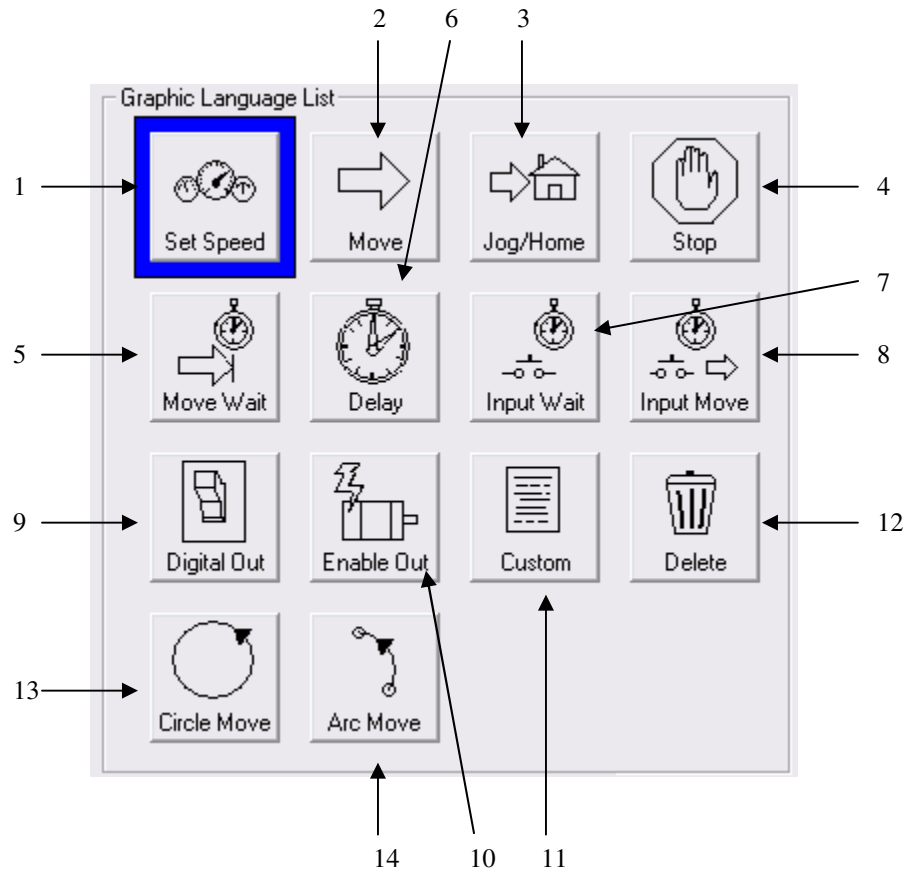


Figure 5.37

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

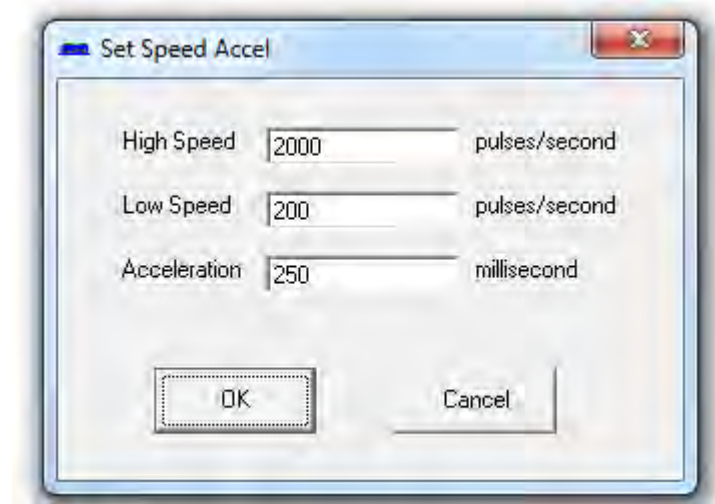


Figure 5.38

2. **Move Object:** Perform absolute move commands on selected axes

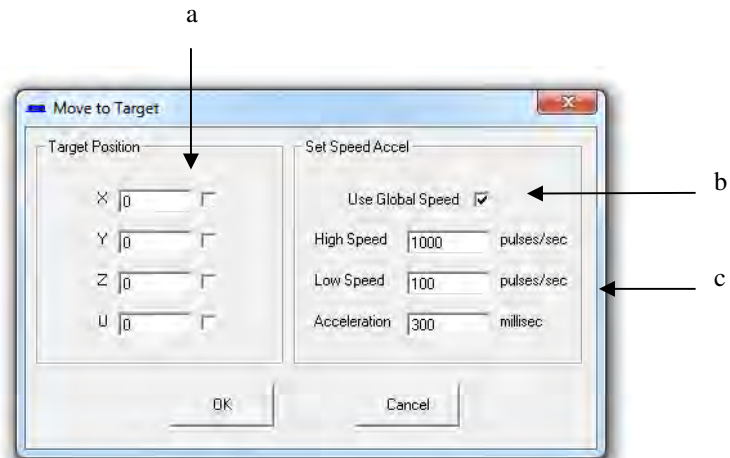


Figure 5.39

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

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

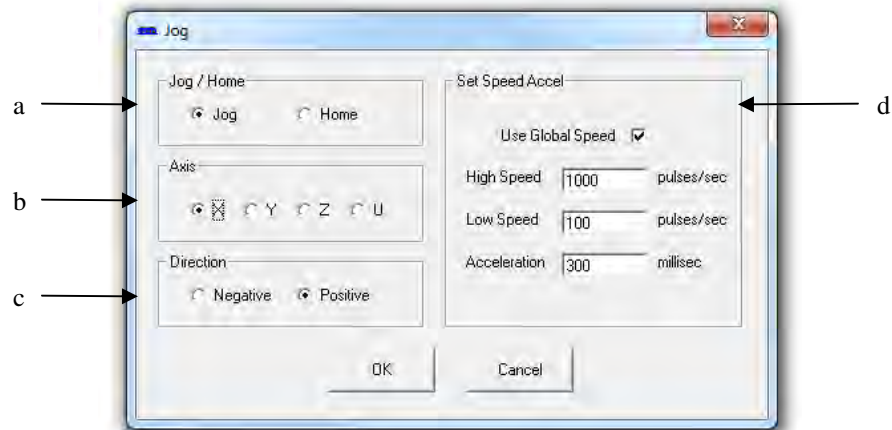


Figure 5.40

- a. Select this setting to be a jog or home move
- b. Select the axis to jog
- c. Select the direction of the move (i.e. “+” or “-“)
- d. See *Graphic Language List Box: Section 2b and 2c*

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

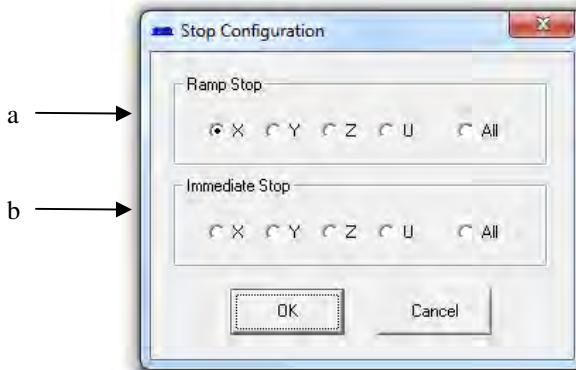


Figure 5.41

- a. If this move is a ramp stop, select 1 or all axes  
 b. If this moves is an immediate stop, select 1 or all axes
5. **Wait Move Object:** Wait until motion is done on a single axis until continuing to execute



Figure 5.42

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

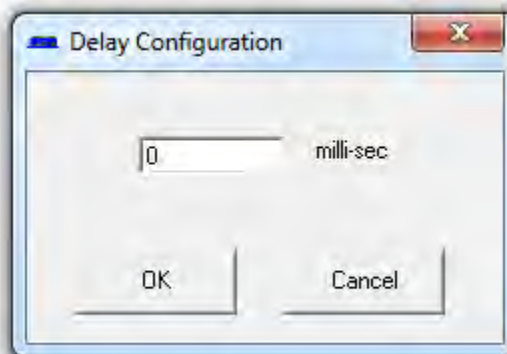


Figure 5.43

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

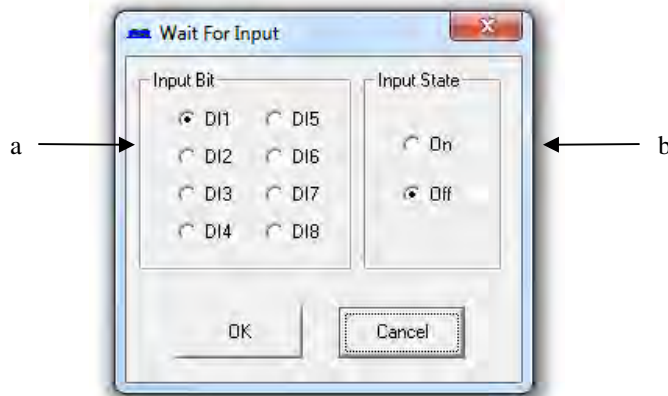


Figure 5.44

- a. Select the digital input
- b. Make the condition on or off

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

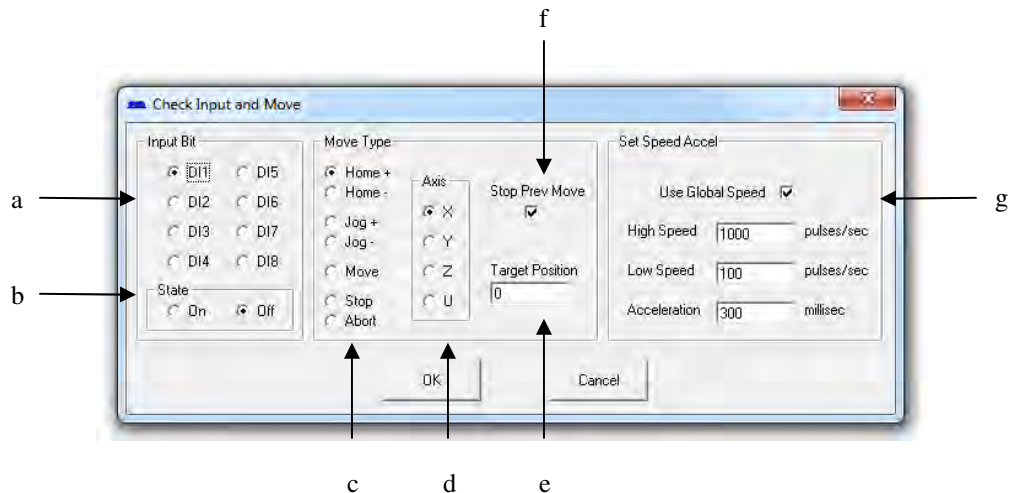


Figure 5.45

- a. Select the digital input
- b. Make the condition on or off
- c. Select move type
- d. Select axis
- e. Enter target position if “Move” type is selected
- f. Click to stop the previous move before processing this setting
- g. See *Graphic Language List Box: Section 2b and 2c*

## 9. Digital Out Object: Set digital output status

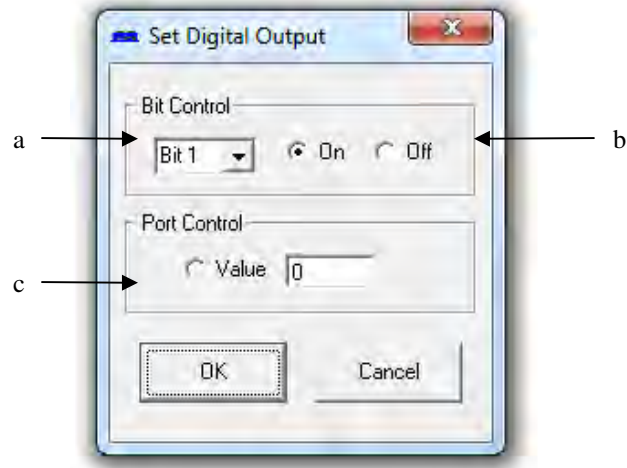


Figure 5.46

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

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

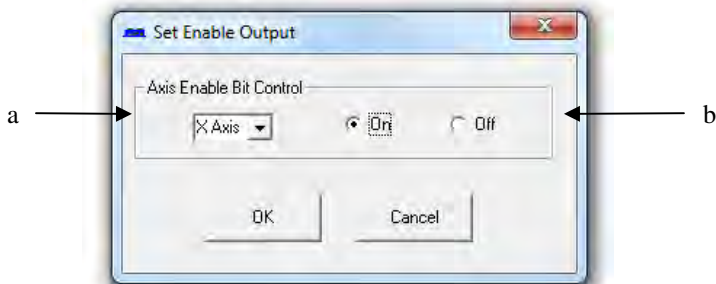


Figure 5.47

- a. Select output axis
- b. Select on/off state

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

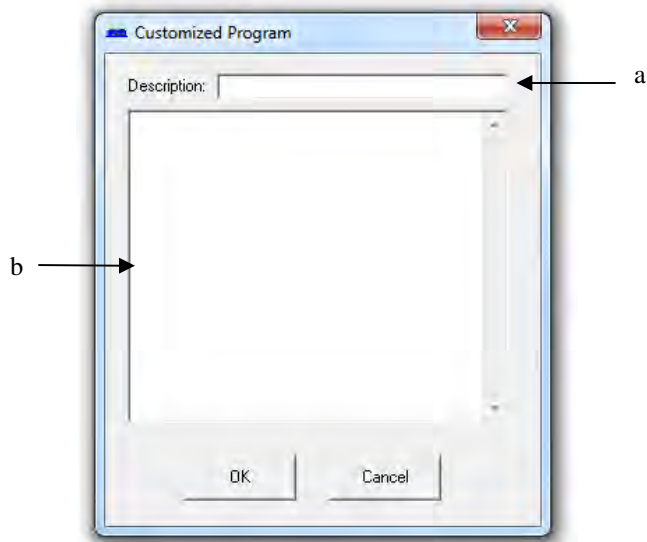


Figure 5.48

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

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



Figure 5.49

### 13. Circle Move: Create a circle interpolation move

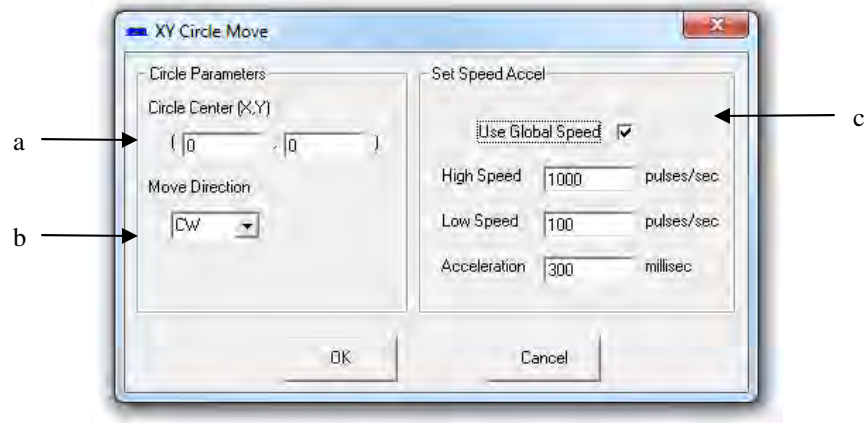


Figure 5.50

- a. Center of the circle
- b. Draw circle in CW or CCW direction
- c. See *Graphic Language List Box: Section 2b and 2c*

### 14. Arc Move: Create an arc interpolation move

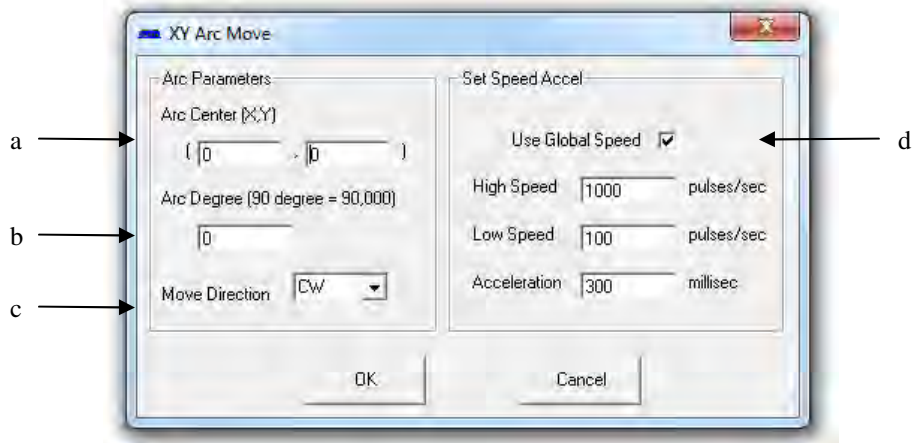


Figure 5.51

- a. Center of the arc
- b. Degree of arc drawn
- c. Draw arc in CW or CCW direction
- d. Check to use global speeds specified in the Set Speed Object
- e. If “Use Global Speed” is not checked, the move will use the following local speed settings

**C. Program File Box**

*See Section H of “Main Control Screen”*

**D. Program File Box**

*See Section J of “Main Control Screen”*

**E. Standalone Program Compile/Download/View**

*See Section K of “Main Control Screen”*

## 6. Motion Control Feature Overview

**Important Note:** All the commands described in this section are interactive commands and are not analogous to stand-alone commands. Refer to the “Standalone Language Specification” section for details regarding stand-alone commands.

### Motion Profile

By default, the PMX-4ET-SA uses trapezoidal velocity profile. See Figure 6.0.

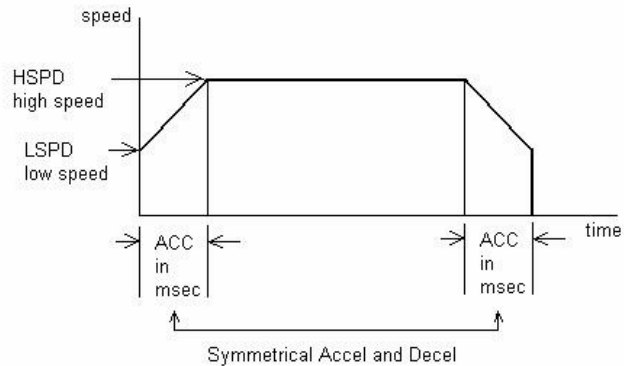


Figure 6.0

S-curve velocity profile can also be achieved by using the **SCV[axis]** command. See Figure 6.1

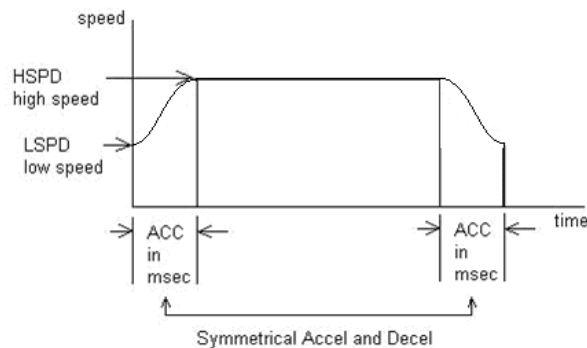


Figure 6.1

High speed and low speed are in pps (pulses/second). Use **HS[axis]** and **LS[axis]** to set/get individual high speed and low speed settings. To set/get the global high speed and low speed values use the **HS** and **LS** commands.

Acceleration and deceleration time are in milliseconds and are symmetrical. Use the **ACC[axis]** command to set/get individual acceleration/deceleration values. To set/get the global acceleration value, use the **ACC** command.

**Notes:**

By default, moves by a single axis use global speed settings, unless individual high speed, low speed and acceleration values for that axis are non-zero.

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**

It is possible to have unique acceleration and deceleration times. In order to decelerate using the value set in the **DEC[axis]** or **DEC** parameter, set **EDEC** to 1.

The minimum and maximum acceleration values depend on the high speed and low speed settings. Refer to Table A.0 and Figure A.0 in **Appendix A** for details.

***Pulse Speed***

Current pulse rate can be read using the **PS** command. For units, see Table 6.0

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

Table 6.0

This command returns the current speed of all axes. The **PS** return value has the following format:

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

***On-The-Fly Speed Change***

On-the-fly speed change can be achieved with the **SSPD[axis]** command. In order to use the **SSPD[axis]** command, s-curve velocity profile must be disabled.

**SSPD Mode**

The correct speed window must be selected in order to use the **SSPD** command. To select a speed window, use the **SSPDM[axis]** command. Refer to **Appendix A** for details.

During on-the-fly speed change operation, you must keep the initial and destination speeds within the speed window.

For non on-the-fly speed change moves, set **SSPDM[axis]** to 0.

### **Motor Status**

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

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

Table 6.1

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

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

Motor Stat [X/Y/Z/U] – Provide motor status of the axis

Buffer enabled – Buffer linear interpolated move status (0: off, 1: on)

Buffer start – The index of the current command in the buffer

Buffer end – The index of the last command in the buffer

Available Buffer – The amount of empty spaces in the buffer  
 Move mode – move mode (0: ABS, 1: INC)

### ***Individual/Linear Interpolation 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 one axis is specified, the motion will be linearly interpolated.

#### Individual/Linear Move Examples:

**[X1000]:** Move X-axis to position 1000.

**[X1000 Y1000]:** Move X-axis to position 1000, Y-axis to position 1000 using linear interpolation.

**[X1000 Y1000 Z100]:** Move X-axis to position 1000, Y-axis to position 1000, Z-axis to position 100 using linear interpolation.

**[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.

**[X1000 U800]:** Move X-axis to position 1000, U-axis to position 800 using linear interpolation.

Individual/Linear Interpolation moves can be performed in two modes: incremental mode. To set move modes, use the **INC** and **ABS** commands respectively.

#### Move Mode Examples:

**[X1000] – INC mode:** The motor will move by 1000 from the current position.

**[X1000] – ABS mode:** The motor will move to absolute position 1000.

### ***Circular Interpolation Moves***

PMX-4ET-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.

**Note:** The maximum allow radius is 134,216,773 pulses on arc or circular moves. All arc or circular moves are interpreted as absolute moves.

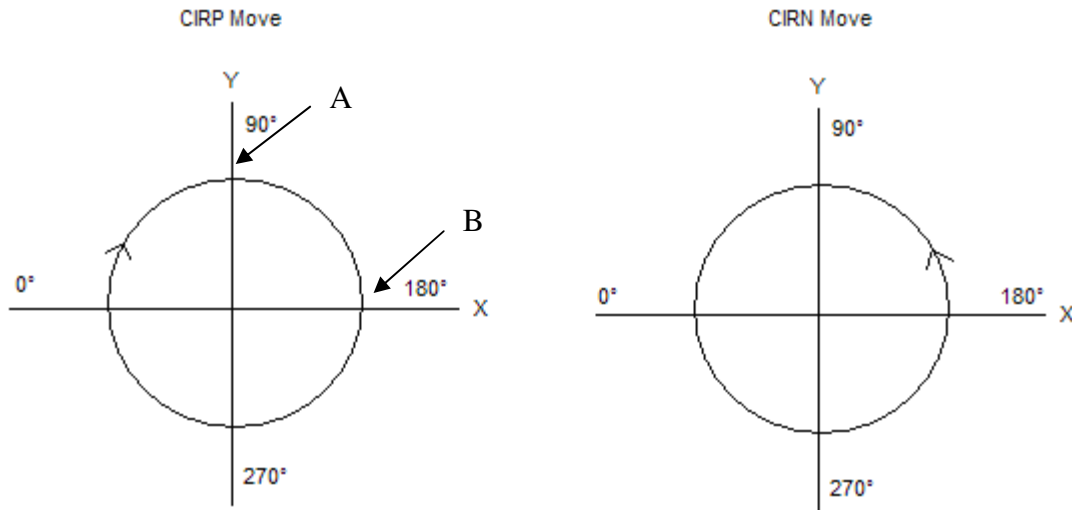


Figure 6.3

### Arc Interpolation Moves

PMX-4ET-SA supports arc interpolation moves using the **ARCPC** and **ARCNC** commands. Arcs are drawn using X,Y axes only.

**ARCPC**[X]:[Y]:[ $\theta_A$ ]:[ $\theta_R$ ] – Draw arc in CW direction where [X][Y] signifies X,Y position of the circle center,  $\theta_A$  signifies the absolute arc angle, and  $\theta_R$  signifies the relative arc angle.

**ARCNC**[X]:[Y]:[ $\theta_A$ ]:[ $\theta_R$ ] – Draw arc in CCW direction where [X][Y] signifies X,Y position of the circle center,  $\theta_A$  signifies the absolute arc angle, and  $\theta_R$  signifies the relative arc angle.

#### Notes:

The maximum allow radius is 134,216,773 pulses on arc or circular moves. All arc or circular moves are interpreted as absolute moves.

Angle values are whole number in thousandths. For example, 45 degrees is 45,000.

$\theta_A$ : The absolute angle standard is depicted in Figure 8.2. For example, to move to position A in Figure 8.2,  $\theta_A$  should always be 90,000 (90°). Likewise, to move to position B in Figure 8.2, the absolute  $\theta_A$  should always be 180,000 (180°). This is independent of the starting position and move direction.

$\theta_R$ : The relative angle is calculated by finding the total degrees that the move will take.

#### Arc Move Examples

Example 1:

Arc start position: (0,1000)  
 Arc start absolute angle: 90,000 (90°)

Arc end position: (1000,0) in CW direction  
 Move amount (degrees): 90,000 (90°)  
 Move command: ARCP0:0:180000:90000

**Example 2:**

Arc start position: (-1000,0)  
 Arc start absolute angle: 0 (0°)  
 Arc end position: (0,1000) in CCW direction  
 Move amount (degrees): 270,000 (270°)  
 Move command: ARCNO:0:450000:270000

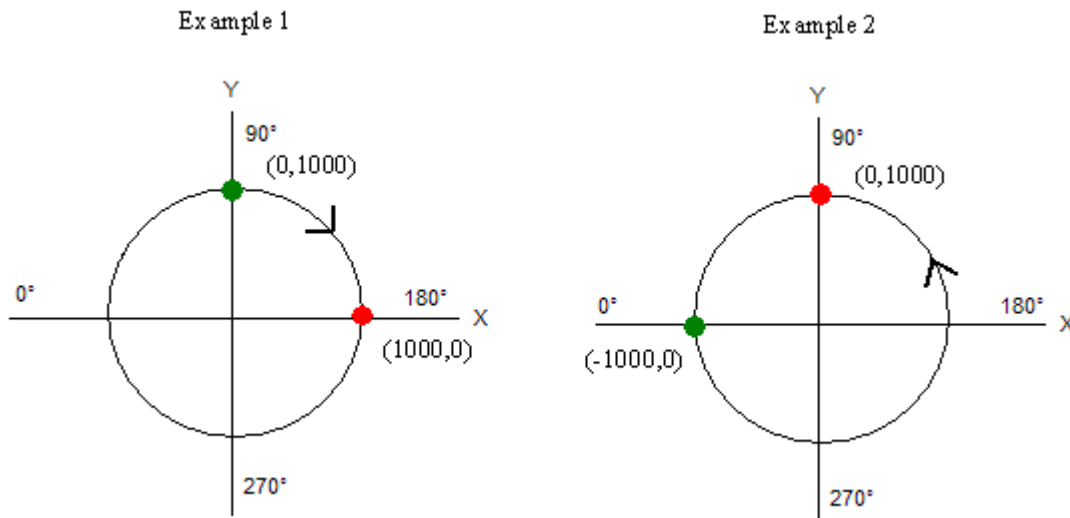


Figure 6.4

**Buffered Linear Interpolation Moves**

PMX-4ET-SA supports buffered linear coordinated motions for X, Y, and Z-axes using the **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 pps, use the following command - **I1000:2000:3000:250**.

**Notes:**

Manual Acceleration Control: When **IACC=0**, acceleration/deceleration must be done manually. To control the acceleration or deceleration manually, gradually increase or decrease the speed value for each interpolated move.

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

Linear interpolation buffer move size is 36 points. To turn on and off buffer move, use the **BO** and **BF** command respectively. When enabled, as soon as the first I command is issued the motion will start.

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

Buffered move operation cannot be used while StepNLoop is enabled.

For command information for buffer move status, see *Motor Status* section.

### **On-The-Fly Target Position Change**

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

**Note:** If a **T** command is sent while the controller is not performing a target move, the command is not processed. Instead, an error response is returned.

### **Homing**

Home search sequence involves moving the motor towards the home or limit switches and then stopping when the relevant input is detected.

The PMX-4ET-SA has four different homing routines. Use the **H** command to perform them.

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

Four homing modes:

0 – Home Input Only (High speed only)

1 – Limit Input Only

2 – Home Input and Z-Index

3 – Z-Index Only

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

### **MODE 0 : Home Input Only (High speed only)**

Figure 6.5 shows the homing routine.

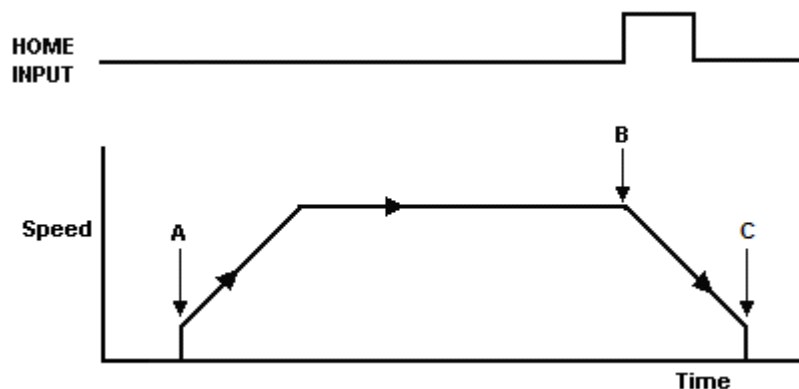


Figure 6.5

- A. Starts the motor from low speed and accelerates to high speed.
- B. As soon as the home input is triggered, the position counter is reset to zero and the motor begins to decelerate to low speed. As the motor decelerates, the position counter keeps counting with reference to the zero position.
- C. Once low speed is reached, the motor stops. The position is non-zero.

**MODE 1 : Limit Only**

Figure 6.6 shows the homing routine.

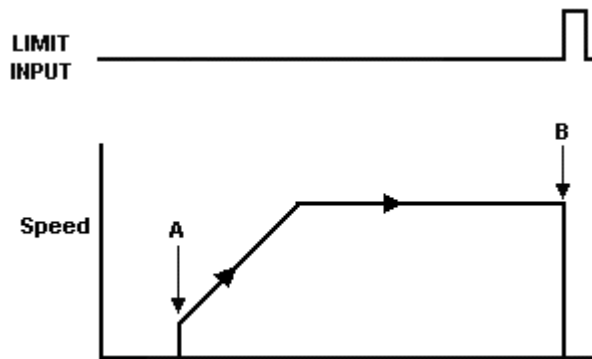


Figure 6.6

- A. Issuing a limit home command starts the motor from low speed and accelerates to high speed.
- B. The corresponding limit is triggered, the motor stops immediately and the position is set to zero.

**MODE 2 : Home and Z-index**

Figure 6.7 shows the homing routine.

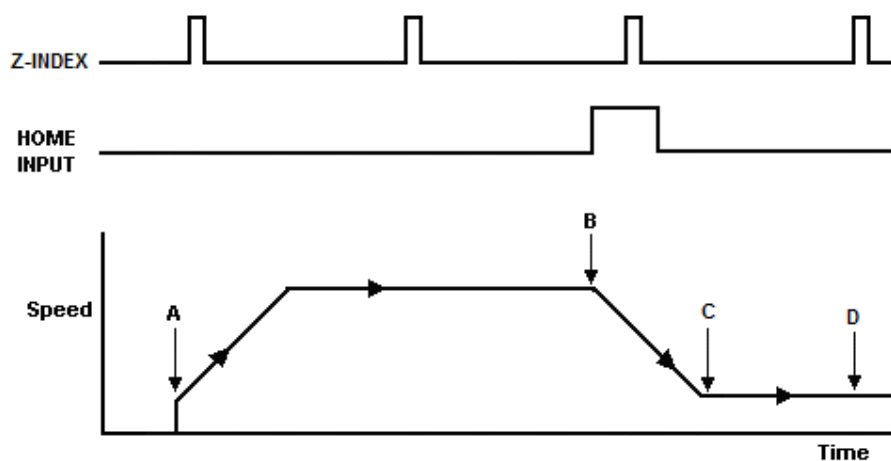


Figure 6.7

- A. Issuing a limit home command starts the motor from low speed and accelerates to high speed.

- B. As soon as the home input is triggered, the motor decelerates to low speed
- C. Once low speed is reached, the motor begins to search for the z-index pulse.
- D. Once the z-index pulse is found, the motor stops and the position is set to zero.

### **MODE 3 : Z-index only**

Figure 6.8 shows the homing routine.

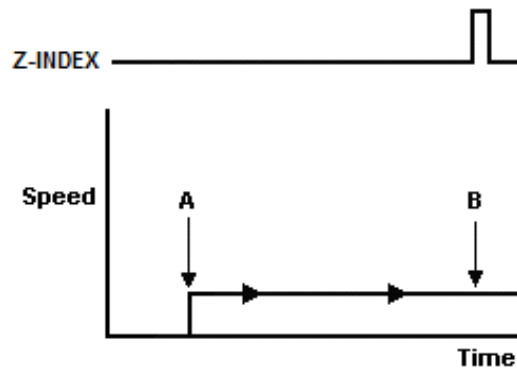


Figure 6.8

- A. Issuing a limit home command starts the motor at low speed.
- B. Once the z-index pulse is found, the motor stops and the position is set to zero.

### **MODE 4 : Home Input Only (High speed and low speed)**

Figure 6.9 shows the homing routine.

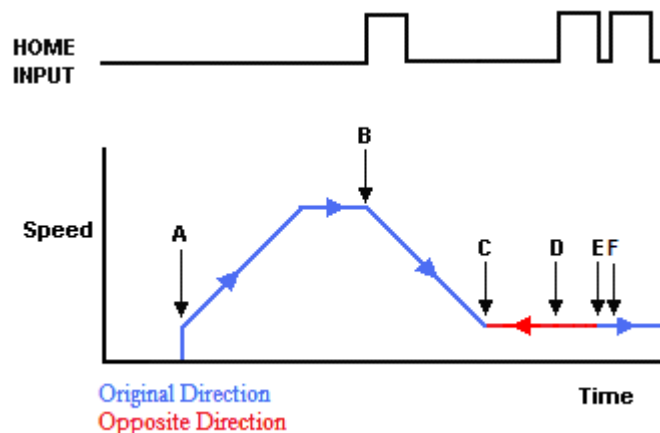


Figure 6.9

- A. Starts the motor from low speed and accelerates to high speed.
- B. As soon as the home input is triggered, the position counter is reset to zero and the motor decelerates to low speed.
- C. Once low speed is reached, the motor reverses direction to search for the home switch.
- D. Once the home switch is reached, it will continue past the home switch until the home switch is off.

- E. The motor is now past the home input. The motor now moves back towards the home switch at low speed.
- F. The home input is triggered again, the position counter is reset to zero and the motor stops immediately

### **Jogging**

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

**J[axis][direction + or -]**

### **Stopping**

When the motor is moving, the **ABORT[axis]** command will immediately stop an individual axis. Use the **ABORT** command to immediately stop ALL axes.

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

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

### **Polarity**

Using the **PO[axis]** command to get and set polarity of the signal below. The format is as an integer with the following bit assignment:

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

Table 6.2

### **Motor Position**

Motor positions can be read using the **PP** command which returns the pulse position of all 4 axes. The return value has the following format:

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

Encoder positions can be read using **PE** command which returns the encoder position of all 4 axes. Encoders are set to 4X reading. The return value has the following format:

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

To manually set/get the pulse position of an individual axis, use the **P[axis]** command. Note that setting the pulse position is not allowed if StepNLoop is enabled.

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

### **Limits and Alarm**

If positive limit switch is triggered while moving in positive direction, the motor will immediately stop and the motor status bit for positive limit error is set. The same is for the negative limit while moving in the negative direction.

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

Once the limit or alarm error is set, use the **CLR[axis]** command to clear the error.

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

### **Digital Inputs/Outputs and Enable Outputs**

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

#### Inputs

Read digital input status using the **DI** command.

Digital input values can also be referenced one bit at a time by the **DI[1-8]** commands. Note that the indexes are 1-based for the bit references (i.e. DI1 refers to bit 0, not bit 1)

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

Table 6.3

If digital input is on (i.e. input is pulled to GND), the bit status is 0. Otherwise, the bit status is 1.

### Digital Outputs

The digital output status can be controlled using the **DO** command. DO value must be within the range of 0-255.

Digital output values can also be referenced one bit at a time by the **DO[1-8]** commands. Note that the indexes are 1-based for the bit references (i.e. DO1 refers to bit 0, not bit 1)

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

Table 6.4

If digital output is turned on (i.e. the output is pulled to VS), the bit status is 1. Otherwise, the bit status is 0.

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

### Enable Outputs

The enable output status can be controlled using the **EO** command. EO value must be within the range of 0-15.

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

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

Table 6.5

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

### **Sync Outputs**

PMX-4ET-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

Table 6.6

**Note:** While feature is enabled for an axis, the corresponding digital output cannot 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 **LESS** than the sync position.
- 3 – Turns output on when the encoder position is **GREATER** 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

## StepNLoop Closed Loop Control

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

SNL performs the following operations:

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

See Table 6.7 for a list of the SNL control parameters.

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

Table 6.7

†A convenient way to find the StepNLoop ratio is to set EX=0, PX=0 and move the motor +1000 pulses. The ratio can be calculated by dividing 1000 by the resulting EX value. Note that the value must be positive. If it is not, then the direction polarity must be adjusted. This test can be performed on all axes that require StepNLoop. See Table 8.2 for details.

To enable/disable the SNL feature use the **SL[axis]** command. To read the SNL status, use **SLS[axis]** command to read the status.

See Table 6.8 for a list of the **SLS[axis]** return values.

Return Value	Description
0	Idle
1	Moving
2	Correcting
3	Stopping
4	Aborting
5	Jogging
6	Homing
7	Z-Homing
8	Correction range error. To clear this error, use <b>CLRS</b> or <b>CLR</b> command.
9	Correction attempt error. To clear this error, use <b>CLRS</b> or <b>CLR</b> command.
10	Stall Error. <b>DX</b> value has exceeded the correction range value. To clear this error, use <b>CLRS</b> or <b>CLR</b> command.
11	Limit Error
12	N/A (i.e. SNL is not enabled)
13	Limit homing

Table 6.8

See Table 6.9 for SNL behavior within different scenarios.

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

Table 6.9

Key

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

**Notes:**

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

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

**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. Also note that the speed during an arc/circular interpolation move is calculated as pulse/sec, NOT encoder counts/sec.

***Device IP Address***

Set the IP address of the PMX-4ET-SA module using the **IP** command. See default IP/socket settings below:

**IP: 192.168.1.250**

**Port: 5001**

Note: To begin communication with a factory default device, configure the PC with the following settings:

IP = 192.168.1.xxx  
Subnet Mask = 255.255.255.0

See sample configuration below:

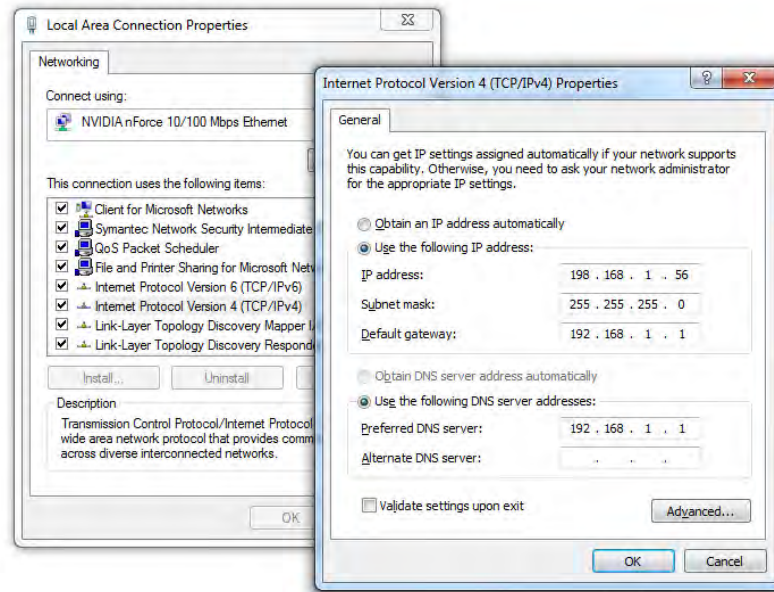


Figure 6.10

### **Changing the IP Address:**

PMX-4ET-SA provides the user with the ability to set the device IP of the module.

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

### **Standalone Programming**

#### **Standalone Program Specification:**

Memory size: 7,650 assembly lines.

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

**WAIT Statement:** When writing a standalone program, it is generally necessary to wait until a motion is completed before moving on to the next line. In order to do this, the WAIT statement must be used. See the examples below:

In the example below, the variable V1 will be set immediately after the X10000 move command begins; it will not wait until the controller is idle.

```
X10000          ;* Move to position 0
V1=100
```

Conversely, in the example below, the variable V1 will not be set until the motion has been completed. V1 will only be set once the controller is idle.

```
X10000          ;* Move to position 0
WAITX         ;* Wait for the move to complete
V1=100
```

**Multi-Threading:** PMX-4ET-SA supports the simultaneous execution of up to 4 standalone programs. Programs 0,1,2,3 are controlled via the **SR0**, **SR1**, **SR2** and **SR3** commands respectively. For examples of multi-threading, please refer to the **Example Stand-alone Programs** section.

**Note:** Sub-routines can be shared by different threads.

**Error Handling:** If an error occurs during standalone execution (i.e. limit error), 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 will be executed. The return jump line will be determined by value of the **SAP** register. If the value is 0, the return jump line will be the line that caused the error. Otherwise, the return jump line will be line 0.

**Calling subroutines over communication:** Once a subroutine is written into the flash, they can be called via Ethernet communication using the **GS** command. The subroutines are referenced by their subroutine number [0-31]. If a subroutine number is not defined, the controller will return with an error.

**Standalone Run on Boot-Up:** Standalone can be configured to run on boot-up using the **SLOAD** command. See description below:

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

Table 6.10

### **Timer Register**

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

### **Communication Time-out Feature (Watchdog)**

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

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

---

## ***Boot-up Sequence***

### **Initial Boot-up:**

PMX-4ET-SA takes approximately 5 seconds to boot up from the moment that power is supplied to the 2-pin connector.

### **Hard Reset detection:**

After initial boot up, the PMX-4ET-SA will begin to look for a hard reset input sequence. If the first input pattern is not detected within 5 seconds, boot-up will skip to “Connection detection”.

However, if the first input pattern is detected within 5 seconds, AND the full reset sequence is reached, the flash memory will be reset to factory defaults. Once the flash is reset, a power cycle needs to be performed in order to communicate via factory default settings. See *Hard Reset (Flash Memory)* section for details.

### **Connection detection:**

If the hard reset input sequence is not detected, the device begins to look for an Ethernet connection. If an Ethernet connection is not detected within 7 seconds, the controller automatically times out and goes to the ***Connection Time-out State***. On the other hand, if, a connection is detected within the 7 second time frame, the controller will go to ***Full Connection State*** at the time of detection.

Note: During connection detection, the term “Ethernet connection” does not mean that a socket connection has been established. Instead, it means that the device is connected to an enabled / active Ethernet network / PC.

- ***Connection Time-out State:*** The controller could not detect an Ethernet connection in the allowable time frame. In this case, any standalone program that is set to run on boot-up will begin execution. Note that once this state is entered, it will no longer be possible to communicate with the controller via Ethernet. To communicate via Ethernet, a power cycle must be performed to restart the boot-up sequence.
- ***Full Connection State:*** The controller found an Ethernet connection in the allowable time frame. In this case, communication established by opening a TCP/IP socket connection. Note that it is possible to close and re-open the connection multiple times. Any stand-alone program that is set to run on boot-up will also begin execution.

### ***Hard Reset (Flash Memory)***

PMX-4ET-SA comes with the ability to reset all the flash parameters to factory default settings. This is especially useful if the user has forgotten the device IP.

Hard reset is done by triggering the DI1/DI2/DI3/DI4/DI5 digital inputs in a particular sequence on boot up (See *Boot-up Sequence* section). There are a total of 7 steps that

must be met in sequence. Each step has an input pattern that must be met before moving on to the next step. See chart below:

### Hard Reset Step Input Conditions

Step Condition	DI1	DI2	DI3	DI4	DI5
1	ON	ON	OFF	OFF	ON
2	<b>OFF</b>	ON	OFF	OFF	ON
3	<b>ON</b>	ON	OFF	OFF	ON
4	ON	ON	<b>ON</b>	OFF	ON
5	ON	ON	<b>OFF</b>	OFF	ON
6	ON	ON	OFF	OFF	<b>OFF</b>
7	ON	ON	OFF	OFF	<b>ON</b>

Table 6.11

**Notes:**

- ON: signal is pulled to GND of opto-supply
- OFF: signal is not pulled to GND of opto-supply
- For each condition, only one signal needs to change state. This signal is in bold

At each step, the LED status will toggle. This is a tool to help signal to the user when to create the next step condition. See chart below:

### Hard Reset Step Motor Enable Status

Step Condition	Pre-trigger LED status	Post-triggered LED status
1	ON	OFF
2	OFF	ON
3	ON	OFF
4	OFF	ON
5	ON	OFF
6	OFF	ON
7	ON	OFF

Table 6.12

The controller will poll for the input pattern at each step for up to 10 sec. If the condition is not reached within the allotted 10 sec, the controller will stop looking for the hard reset sequence and continue its normal boot-up sequence (*See Boot-up Sequence* section). The motor will start as disabled.

However, once the condition for a step is met, it will immediately start looking for the next sequence (i.e. it is not necessary to wait the full 10 sec to trigger the next step).

If the PMX-4ET-SA successfully triggers steps 1-7 in sequence, the flash is reset to factory default. At the end of the flash reset operation, the LED will flash slowly for a few seconds. At the end of this sequence, the LED will remain off.

Once the flash is reset, a power cycle needs to be performed in order to communicate via factory default settings.

### **Storing to Flash**

The following items are stored to flash:

ASCII Command	Description
IP	IP address
DOBOOT	DO configuration at boot-up
EDEC	Unique deceleration enable
EOBOOT	EO configuration at boot-up
IACC	Automatic I-move acceleration/deceleration enable
IERR	Ignore limit error enable
POL[axis]	Polarity settings
SAP	Jump return line select (Stand-alone error handling)
EOP	EO polarity
DIP	DI polarity
DOP	DO polarity
SCV[axis]	S-curve enable
SL[axis], SLR[axis], SLE[axis], SLT[axis], SLA[axis]	StepNLoop parameters
SLOAD	Standalone program run on boot-up parameter
TOC	Time-out counter reset value
V50-V99	Note that on boot-up, V0-V49 are reset to value 0

Table 6.13

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

## 7. Ethernet Communication Protocol

PMX-4ET-SA is 10Mbps Ethernet ASCII communication over TCP/IP.

Communication between the PC/PLC and PMX-4ET-SA is done using standard socket programming.

### **Socket Settings**

Port: 5001

### **ASCII Protocol**

Sending Command

ASCII command string in the format of

[ASCII Command][NUL]

*[NUL] character has ASCII code 0.*

Receiving Reply

The response will be in the format of

[Response][NUL]

*[NUL] character has ASCII code 0.*

Examples:

For querying the x-axis polarity

Send: POX[NUL]

Reply: 7[NUL]

For jogging the x-motor in positive direction

Send: JX+[NUL]

Reply: OK[NUL]

For aborting any motion in progress

Send: ABORT[NUL]

Reply: OK[NUL]

## 8. 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 the current global acceleration value in milliseconds	
ACC=[Value]	Sets the global acceleration value in milliseconds	OK
ACCX ACCY ACCZ ACCU	Returns the current individual acceleration value in milliseconds	
ACCX=[value] ACCY=[value] ACCZ=[value] ACCU=[value]	Sets the individual acceleration value in milliseconds	OK
ARCP[X]:[Y]:[θ <sub>A</sub> ]:[θ <sub>R</sub> ]	XY Arc interpolation move (CW direction)	OK
ARCN[X]:[Y]:[θ <sub>A</sub> ]:[θ <sub>R</sub> ]	XY Arc interpolation move (CCW direction)	OK
BF	Disable buffered move	OK
BO	Enable buffer move	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 the motor limit or alarm status bit. Also clears StepNLoop errors	OK
DEC	Returns the current global deceleration value in milliseconds	
DEC=[Value]	Sets the global deceleration value in milliseconds	OK
DECX DECY DECZ DECU	Returns the current individual deceleration value in milliseconds	
DECX=[value] DECY=[value] DECZ=[value] DECU=[value]	Sets the individual deceleration value in milliseconds	OK
DI	Returns the 8 bit status of general purpose digital inputs	[0-255]
DI[1-8]	Returns the individual bit status of the specified general purpose digital input.	[0,1]
DIP	Returns the digital input polarity	[0,1]
DIP=[0 or 1]	Sets the digital input polarity	OK
DO	Returns the 8 bit status of general purpose digital outputs	[0-255]
DO=[value]	Sets the 8 bit status of the general purpose digital output	OK
DO[1-8]	Returns the individual bit status of the specified general purpose digital output	[0,1]
DO[1-8]=[value]	Sets the bit of the specified general purpose digital output	OK

DOP	Returns the digital output polarity	[0,1]
DOP=[0,1]	Sets the digital output polarity	
DN	Returns device name	[4ET00-4ET99]
DN=[value]	Sets device name. value must be in the range [4ET00, 4ET99]	OK
DXX DXY DXZ DXU	Returns the StepNLoop delta value of the specified axis	32-bit Number
EDEC	Returns the enable deceleration status	[0,1]
EDEC=[0 or 1]	Sets the enabled deceleration status	OK
EO	Returns the 4 bit status of the enable outputs	[0-15]
EO=[value]	Sets the 4 bits status of the enable outputs.	OK
EO[1-4]	Returns the individual bit status of the specified enable output	[0,1]
EO[1-4]=[value]	Sets the individual bit status of the specified enable outputs	OK
EOP	Returns the enable output polarity	[0,1]
EOP=[0,1]	Sets the enable output polarity	
EX=[value] EY=[value] EZ=[value] EU=[value]	Sets encoder value of the specified axis	OK
GS[0-31]	Calls a defined subroutine	OK
HS	Returns the global high speed setting	[1-4,000,000]
HS=[value]	Sets the global high speed	OK
HSX HSY HSZ HSU	Returns the individual high speed setting	[1-4,000,000]
HSX=[value] HSY=[value] HSZ=[value] HSU=[value]	Sets the 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	Returns the automatic acceleration during buffer interpolated move status	[0-1]
IACC=[0 or 1]	Sets automatic acceleration during buffer interpolated move status	OK
IERR	Returns the ignore limit/alarm error status	[0-1]
IERR=[0 or 1]	Sets the ignore limit/alarm error status	OK
INC	Sets incremental move mode	OK
JE	Returns the joystick enable status	[0-15]
JE=[value]	Sets the joystick enable status	OK
JX[+/-] JY[+/-] JZ[+/-] JU[+/-]	Jogs the motor in plus [+] or minus [-] direction.	OK

LS	Returns the global low speed setting	[1-4,000,000]
LS=[value]	Sets the global low speed	OK
LSX LSY LSZ LSU	Returns the individual low speed setting	[1-4,000,000]
LSX=[value] LSY=[value] LSZ=[value] LSU=[value]	Sets the individual low speed	OK
MST	Returns the all motor status, buffer move status, and move mode status	See Table 6.1
PE	Returns the 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 the polarity setup	See Table 6.2
POX=[value] POY=[value] POZ=[value] POU=[value]	Sets the polarity	OK
PP	Returns the current pulse counter values of all 4 axes	[X Pulse Position]: [Y Pulse Position]: [Z Pulse Position]: [U Pulse Position]
PS	Returns the current pulse speed values of all 4 axes	[X Speed]: [Y Speed]: [Z Speed]: [U Speed]
PX=[value] PY=[value] PZ=[value] PU=[value]	Sets the position value of axis	OK
SASTAT	Returns the standalone program status 0 – Stopped 1 – Running 2 – Paused 4 – In Error	[0-4]
SA[LineNumber]	Returns the standalone line	Single line of compiled code
SA[LineNumber]=[Value]	Sets the standalone line	OK
SAP	Get the return jump line (standalone error handling)	[0,1]
SAP=[0 or 1]	Sets the return jump line (standalone error handling)	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]	Enables or disable s-curve. If disabled, trapezoidal acceleration/ deceleration will be used.	OK
SLAX	Returns the StepNLoop maximum attempt value of axis	

SLAY SLAZ SLAU		
SLAX=[value] SLAY=[value] SLAZ=[value] SLAU=[value]	Sets the StepNLoop maximum attempt value of axis	OK
SLEX SLEY SLEZ SLEU	Returns the StepNLoop error range value of axis	
SLEX=[value] SLEY=[value] SLEZ=[value] SLEU=[value]	Sets the StepNLoop error range value of axis	OK
SLRX SLRY SLRZ SLRU	Returns the StepNLoop ratio of axis (ppr / cpr)	[0.001-999.999]
SLRX=[value] SLRY=[value] SLRZ=[value] SLRU=[value]	Sets the StepNLoop ratio of axis (ppr / cpr)	OK
SLSX SLSY SLSZ SLSU	Returns the StepNLoop status of axis	[0-12]
SLTX SLTY SLTZ SLTU	Returns the StepNLoop tolerance of axis	
SLTX=[value] SLTY=[value] SLTZ=[value] SLTU=[value]	Sets the StepNLoop tolerance of axis	OK
SLX SLY SLZ SLU	Returns the StepNLoop enable of axis	[0,1]
SLX=[value] SLY=[value] SLZ=[value] SLU=[value]	Sets the StepNLoop enable of axis	OK
SLOAD	Returns the RunOnBoot parameter	[0-15]
SLOAD=[value]	Set the 4 bit RunOnBoot parameter	OK
SR[0-3]=[value]	Controls the specified standalone program: 0 – Stop standalone program 1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program	OK
SPC[0-3]	Returns the program counter for the specified standalone program	[0-7650]
SSPDY=[value] SSPDZ=[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	OK

SSPDU=[value]	disable s-curve acceleration/ deceleration control.	
SSPDMX SSPDMY SSPDMZ SSPDMU	Returns the on-the-fly speed change mode for each axis	[0-7]
SSPDMX=[value] SSPDMY=[value] SSPDMZ=[value] SSPDMU=[value]	Sets the 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	Stores the parameters to flash. See Table 6.13	OK
SYNXC SYNYC SYNZC SYNUC	Returns the 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=[value] SYNYC=[value] SYNZC=[value] SYNUC=[value]	Sets the 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 off sync output for the specified axis	OK
SYNXO SYNYO SYNZO SYNUO	Turns on sync output for the specified axis	OK
SYNXP SYNYP SYNZP SYNUP	Returns the trigger position for the specified axis	28 bit signed number
SYNXP=[value] SYNYP=[value] SYNZP=[value] SYNUP=[value]	Sets the trigger position for the specified axis	28 bit signed number
SYNXT SYNYT SYNZT SYNUT	Returns the pulse width time (ms). Only applicable if sync output configuration is set to 1.	[0-10]
SYNXT=[value] SYNYT=[value] SYNZT=[value] SYNUT=[value]	Sets the pulse width time (ms). Only applicable if sync output configuration is set to 1.	OK
T[axis]=[value]	Set on-the-fly target position change	OK
TOC	Returns the communication time-out parameter. Value is in milliseconds.	32-bit number
TOC=[value]	Set the communication time-out parameter.	OK
TR	Returns the timer register value	[0-1,000,000]
TR=[value]	Sets the timer register value (ms)	OK
V[0-99]	Returns the standalone variable value	

V[0-99]=[Value]	Sets the specified standalone variable value	OK
VER	Returns the controller firmware version	VXXX
X[target X] Y[target Y] Z[target Z] U[target U]	Individual/interpolation move command	OK

Table 8.0

### Error Codes

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

Error Code	Description
?[Command]	The ASCII command is not understood by the PMX-4ET-SA
?PULSING	A move or position change command is sent while the PMX-4ET-SA is outputting pulses.
?LIMIT	A move command is sent while the axis has a limit error.
?ALARM	A move command is sent while the alarm is on.
?StatError	A move command is issued while the controller is in error state.
?Index out of Range	The index for the command sent to the controller is not valid.
?Sub not Initialized	Call to a subroutine using the <b>GS</b> command is not valid because the specified subroutine has not been defined.
?Timer Running	An attempt to set the timer register while it is running has been made.
?In incremental mode	A circle or arc interpolation move has been issue while the PMX-4ET-SA is in incremental mode.
?BUFFER FULL	An attempt to add a move to a full buffer has been made.
?BUFON	Buffer is on.
?BUFOFF	Buffer is off.
?SSPD mode not initialized	An on-the-fly speed change was issued without initialized the SSPDM parameter.
?Bad SSPD Command	An on-the-fly speed change was issued with an invalid speed.
?Speed out of range	An on-the-fly speed change was issued with a speed outside of the allowable window defined by the SSPDM parameter.
?S-CURVE ON	An on-the-fly speed change was issued while s-curve acceleration is enabled.

Table 8.1

## 9. Standalone Language Specification

;

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
WAITX
X2000          ; ***Move X axis to position 2000
WAITX

```

## **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.

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 absolute and relative angle. Angle is in whole number in thousandth. For example, 45 degrees is 45,000.

Syntax:

ARC[P for CW, N for CCW][Center X]:[Center Y]:[Angle<sub>A</sub>]:[Angle<sub>R</sub>]

Examples:

\*\*\* move 90 degrees CW to absolute angle 90 degrees

ARCP1000:0:90000:90000

WAITX

\*\*\*\* move 180 degrees CCW to absolute angle 0 degrees

ARCNO:1000:0:180000

WAITX

## ***CIR***

Description:

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

Specify clockwise or counter-clockwise and the center location.

Syntax:

CIR[P for CW, N for CCW] [Center X]:[Center Y]

Examples:

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

WAITX

CIRNO:2000 ;\*\*\*Using X 0 and Y 2000 perform circular move (CCW)

WAITX

## ***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 4ET 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 4ET 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 4ET 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 4ET 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. Also clears StepNLoop error.

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
    WAITX
ELSE
    X-1000    ;***If V1 is not 1, then move to -1000
    WAITX
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
    WAITX
ELSEIF V1=2
    X2000
    WAITX
ELSEIF V1=3
    X3000
    WAITX
ELSE
    X0
    WAITX
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
WAITX
X1000
WAITX
END
```

## **ENDIF**

### Description:

Indicates end of IF operation

### Syntax:

ENDIF

### Examples:

```
IF V1=1
    X1000
    WAITX
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
    WAITX
    X1000
    WAITX
ENDSUB
```

## **ENDWHILE**

Description:

Indicate end of WHILE loop

Syntax:

ENDWHILE

Examples:

```

WHILE V1=1          ;***While V1 is 1 continue to loop
    X0
    WAITX
    X1000
    WAITX
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 4ET 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 4ET 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 0 to 31.

**Note:** Subroutine definitions should be written AFTER the END statement

Subroutine 31 is reserved for error handling

Syntax:

```
GOSUB [subroutine number]
[Subroutine Number] range is 0 to 31
```

Examples:

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

## **HLHOME[axis][+ or -]**

Description:

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

Syntax:

```
HLHOME[Axis][+ or -]
```

Examples:

```
HLHOMEX+ ;***Low speed homes X axis in positive direction
WAITX
HLHOMEZ- ;***Low speed homes Z axis in negative direction
WAITZ
```

## **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  
 WAITX  
 HOMEZ- ;\*\*\*Homes Z axis in negative direction  
 WAITZ

## **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 4,000,000.

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 4,000,000.

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
    WAITX
ENDIF
```

## ***INC***

Description:

**Command:** Changes all move commands to incremental mode.

Syntax:

```
INC
```

Examples:

```
INC                ;***Change to increment 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
```

## ***LHOME[axis][+ or -]***

Description:

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

Syntax:

LHOME[Axis][+ or -]

Examples:

```
LHOMEX+ ;***Limit homes X axis in positive direction
WAITX
LHOMEZ- ;***Limit homes Z axis in negative direction
WAITZ
```

## **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 4,000,000.

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 4,000,000.

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[axis]***

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 cannot 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.

### **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

```

### **SR[0,3]**

Description:

**Write:** Set the standalone control for the specified standalone program

Syntax:

```

Write: SR[0-3] = [0-3]
        SR[0-3] = [0-3]

```

Examples:

```

IF DI1=1           ; If digital input 1 is on
    SR0=0          ; Turn off standalone program 0
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 cannot 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

```

## **STORE**

Description:

**Command:** Store all values to flash

Syntax:

```
STORE
```

Examples:

```

V80=EX       ;***Put encoder value in V80
DELAY=1000   ;***Wait 1 second
STORE        ;***Store V80 to non-volatile flash

```

## ***SUB***

Description:

Indicates start of subroutine

**Note:** Subroutine definitions should be written AFTER the END statement

Subroutine 31 is reserved for error handling

Syntax:

```
SUB [subroutine number]
    [Subroutine Number] range is 0 to 31
```

Examples:

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

## ***SYNCFG[axis]***

Description:

**Write:** Set sync output configuration for axis

Syntax:

```
Write: SYNCFG[axis]=[value]
    SYNCFG[axis]=[variable]
```

Examples:

```
SYNCFGX=1          ;*** Set sync output configuration to 1 for x-axis
SYNPOSX=3000      ;*** Set sync output position to 3000 for x-axis
SYNTIMEX=10       ;*** Set sync output pulse time to 10 ms for x-axis
SYNONX            ;*** Turn on sync output for x-axis
V1=1              ;*** Wait until sync output is triggered for x-axis
WHILE V1 != 2
    V1=SYNSTATX
ENDWHILE
SYNOFFX           ;*** Disable sync output for x-axis
```

## ***SYNOFF[axis]***

Description:

**Write:** Disable sync output for axis

Syntax:

```
Write: SYNOFF[axis]
```

Examples:

See SYNCFG[axis]

### ***SYNON[axis]***

Description:

**Write:** Enable sync output for axis

Syntax:

**Write:** SYNON[axis]

Examples:

See SYNCFG[axis]

### ***SYNPOS[axis]***

Description:

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

Syntax:

**Write:** SYNPOS[axis]=[value]

**Write:** SYNPOS[axis]=[variable]

Examples:

See SYNCFG[axis]

### ***SYNSTAT[axis]***

Description:

**Read:** Get status for sync output of axis

Syntax:

**Read:** [variable] = SYN[axis]S

Examples:

See SYNCFG[axis]

### ***SYNTIME[axis]***

Description:

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

Syntax:

**Write:** SYNTIME[axis]=[value]

Examples:

See SYNCFG[axis]

### ***TOC***

Description:

Sets the communication time-out parameter. Value is in milli-seconds.

Syntax:

TOC=[long value]

Examples:

TOC=10000 ;\*\*\*Sets time-out parameter to 10 seconds

### ***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
    WAITX
  ELSEIF TR>5000
    X3000
    WAITX
  ELSE
    X8000
    WAITX
  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
WAITU
V10 = 1200      ;***Set variable 10 value to 1200
UV10           ;***Move U Axis to variable 10 value
WAITU
```

## V[index]

Description:

Assign to variable.  
 Performax 4ET has 100 variables [V0-V99]

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[axis]**

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
  WAITX
  X1000
  WAITX
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
WAITX
X2000Y3000 ;***Move X to 2000 and Y to 3000 in linear interpolation move
WAITX
V10 = 1200  ;***Set variable 10 value to 1200
XV10       ;***Move X Axis to variable 10 value
WAITX
```

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

```

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

```

## 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
WAITZ
Y1000Z2000U3000 ;***Move Y to 1000, Z to 2000, U to 3000
WAITY
V10 = 1200 ;***Set variable 10 value to 1200
ZV10 ;***Move Z Axis to variable 10 value
WAITZ

```

### 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

```

## 10. Example Standalone Programs

### ***Standalone Example Program 1 – Single Thread***

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

```

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

```

### ***Standalone Example Program 2 – Single Thread***

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

```

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

```

### ***Standalone Example Program 3 – Single Thread***

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

```

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

```

### **Standalone Example Program 4 – Single Thread**

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

```

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

```

### **Standalone Example Program 5 – Single Thread**

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

```

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

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

```

### **Standalone Example Program 6 – Single Thread**

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

```

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

```

### **Standalone Example Program 7 – Multi Thread**

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

```

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

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

```

### **Standalone Example Program 8 – Multi Thread**

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

```

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

PRG 1                                ;* Start of Program 1
WHILE 1=1                             ;* Forever loop
    V1=MSTX&2048                     ;* Get bit time-out counter alarm variable
    IF V1 = 1024                     ;* If time-out counter alarm is on
        SR0=0                        ;* Stop program 0
        ABORTX                       ;* Abort the motor
        DO=0                          ;* Set DO=0
        DELAY=3000;                  ;* Delay 3 seconds
        SR0=1                        ;* Turn program 0 back on
        DO=1                          ;* Set DO=1
    ENDIF
ENDWHILE                             ;* Go back to WHILE statement
END                                   ;* End Program 1

```

## Appendix A: Speed Settings

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

Table A.0

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

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

### **Acceleration/Deceleration Range**

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

The minimum acceleration/deceleration setting for a given high speed and low speed is shown in Table A.0.

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

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

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

Figure A.0

Examples:

- a) If **HSPD** = 20,000 pps, **LSPD** = 10,000 pps:
  - a. Min acceleration allowable: **1 ms**
  - b. Max acceleration allowable:  
 $((20,000 - 10000) / 50) \times 1,000 \text{ ms} = \mathbf{200,000 \text{ ms}}$  (200 sec)
  
- b) If **HSPD** = 900,000 pps, **LSPD** = 9,000 pps:
  - a. Min acceleration allowable: **1 ms**
  - b. Max acceleration allowable:  
 $((900,000 - 9,000) / 1500) \times 1000 \text{ ms} = \mathbf{594,000 \text{ ms}}$  (594 sec)

## Acceleration/Deceleration Range – Positional Move

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

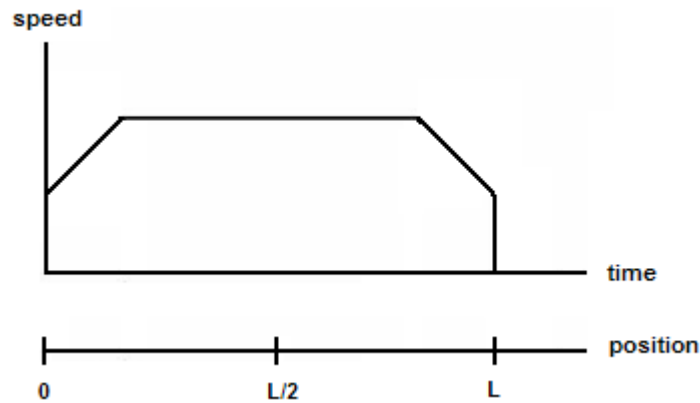


Figure A.1

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

## **Contact Information**

Arcus Technology, Inc.

3159 Independence Dr  
Livermore, CA 94551  
925-373-8800

[www.arcustechnology.com](http://www.arcustechnology.com)

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