Performax 4ET – SA

Advanced 4-Axis Ethernet Stepper Motion Controller Standalone Version

Manual

First edition, June 2008

Revision History:

1.01 – First Revision
1.02 – Added flash reset and boot sequence
1.03 – Updated standalone max lines to 7650, added SNL status description to pg 20, updated ASCII table "=[value]", added standalone execution speed.

Firmware Compatibility:

V209

Software Compatibility:

V121

Table of Contents

1. Introduction

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

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

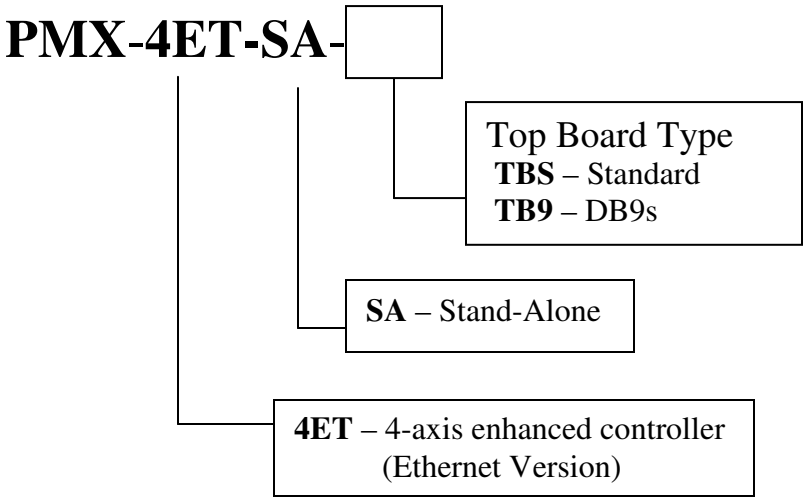
- 3D CAD/CAM
- Engraving
- Laser cutting

Performax 4ET SA Features

- Ethernet 10Mbps communication using Arcus ASCII command
- Maximum pulse output rate of 4M PPS per axis
- Trapezoidal or s-curve acceleration
- On-the-fly speed change
- Continuous linear coordinated buffered move for XYZ axes for smooth move control with buffer size of 36
- XYZU linear coordinated motion
- XY circular coordinated motion
- XY arc coordinated motion
- Opto-isolated +Limit, -Limit, Home, and Alarm inputs per axis
- Homing routine using:
  - Home input only
  - Z index encoder channel only
  - Home and Z index encoder channel
- Pulse/Dir/Enable open collector outputs per axis
- Single-ended or differential quadrature encoder inputs per axis
- 8 opto-isolated digital inputs (NPN)
- 8 opto-isolated digital outputs (NPN)
  - Option to use 4 of the digital outputs for synchronous triggering
- Standalone programmable

Model Numbers

Main Product



Contacting Support

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

Or, contact your local distributor for technical support.

2. 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.

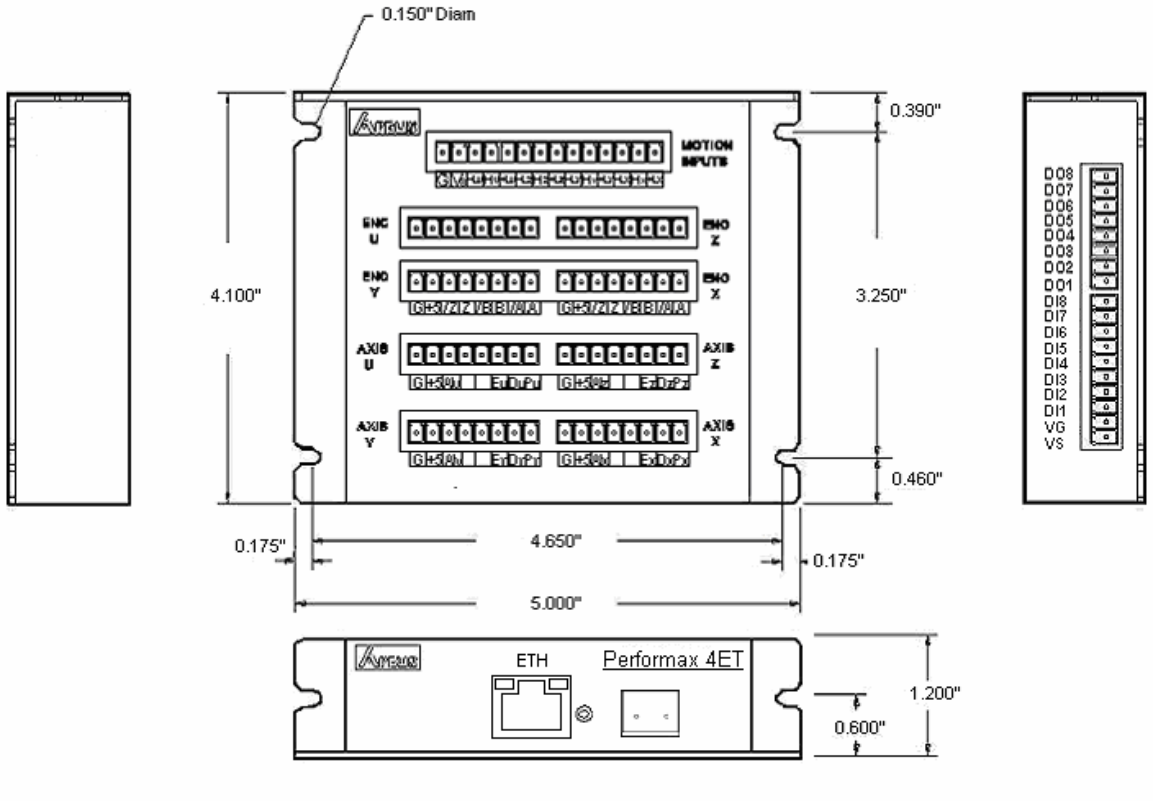
Standard Top Board (PMX-4ET-SA-TBS)

Standard Top Board consisting 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 consisting 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).

3. Dimensions

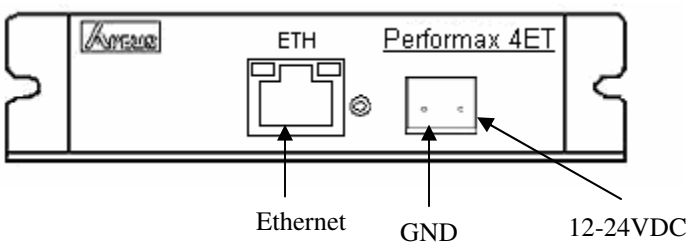


Note: The image above is of PMX-4ET-SA-TBS. The dimensions of PMX-4ET-SA-TB9 are the same, with the exception of the top board connectors.

4. Pin Descriptions

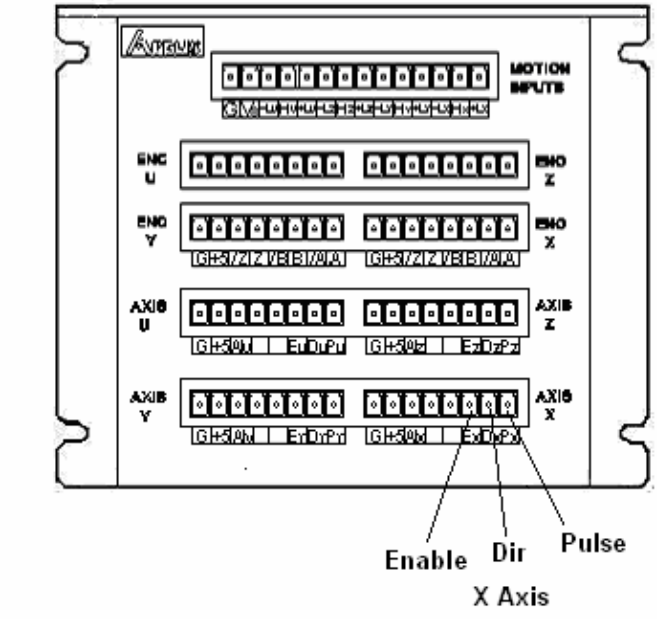
A. Connecting input power and Ethernet connection

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.



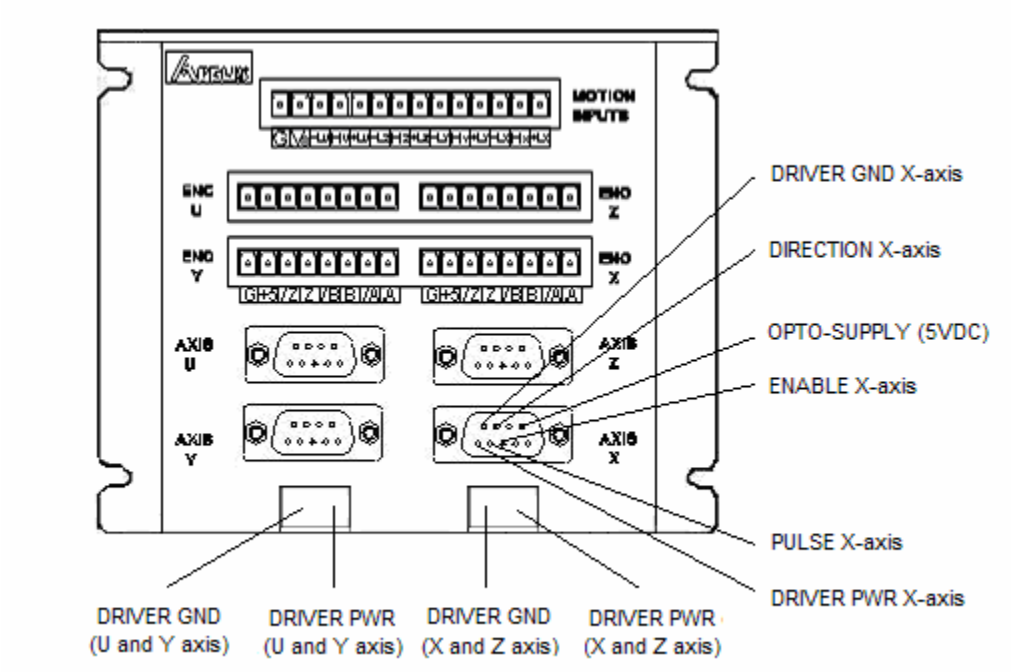
B1. Connecting to a stepper driver (PMX-4ET-SA-TBS)

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



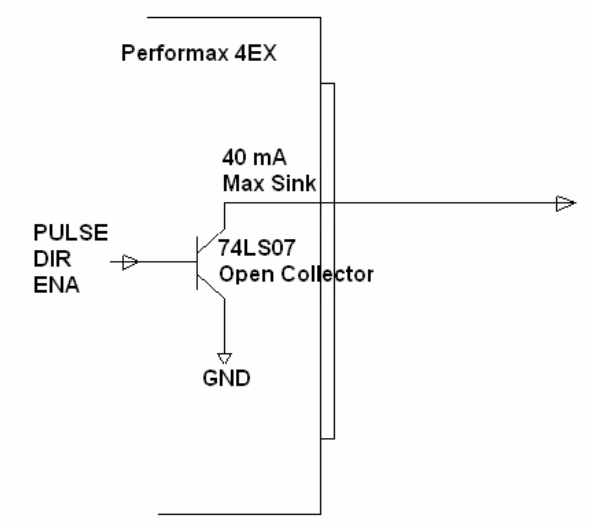
B2. Connecting to a stepper driver (PMX-4ET-SA-TB9)

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

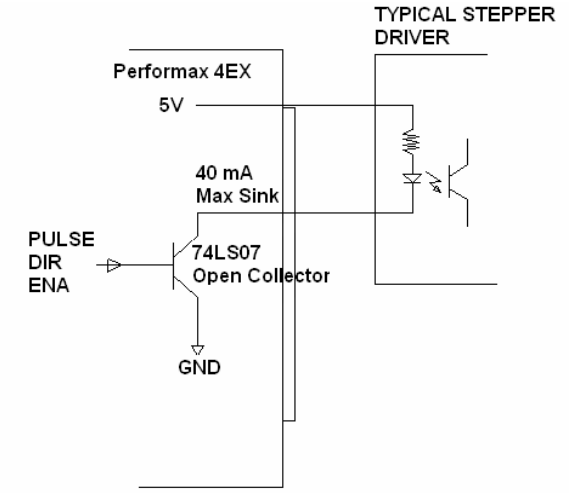


The pins on the DB9 headers can be connected directly to a DMX-A2-DRV module (pin-to-pin compatible)

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



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

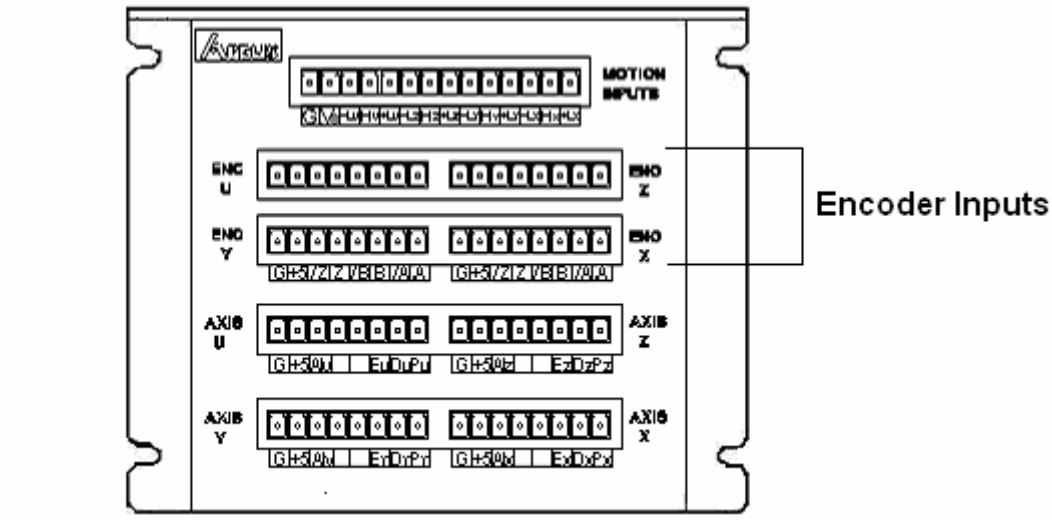


C. Connecting Encoders

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

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

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



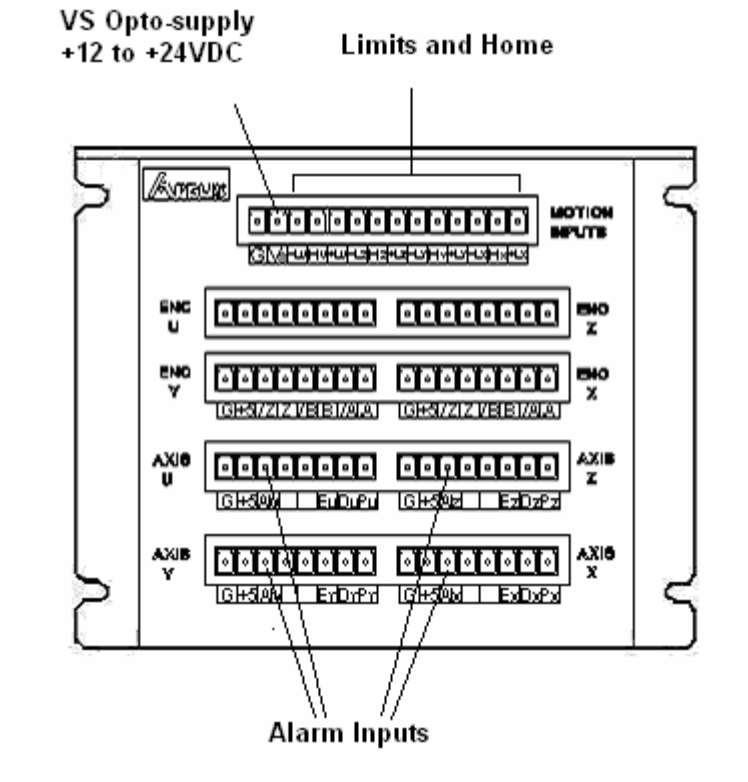
Note: Encoder pins are identical for both standard as well as DB9 top board

D. Connecting Limits/Home/Alarm

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

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

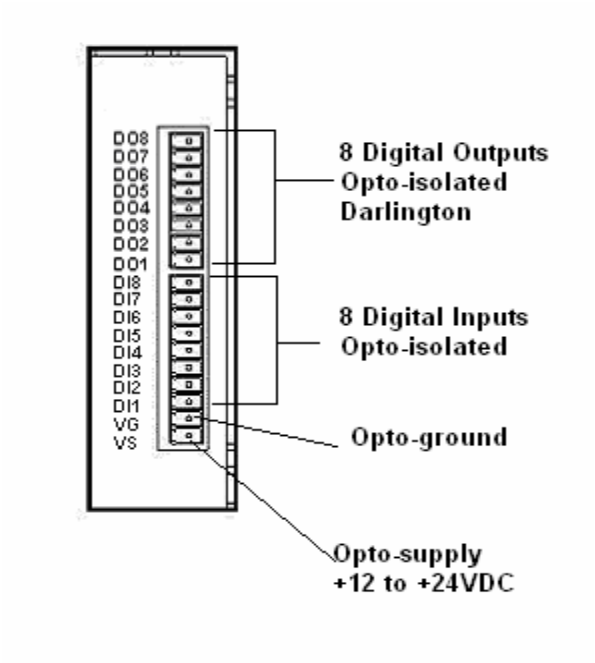
To trigger the opto-isolated inputs, sink the limit or home input signal to the ground of the Vs. For wiring diagram, see "Connecting Digital Inputs and Outputs"



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

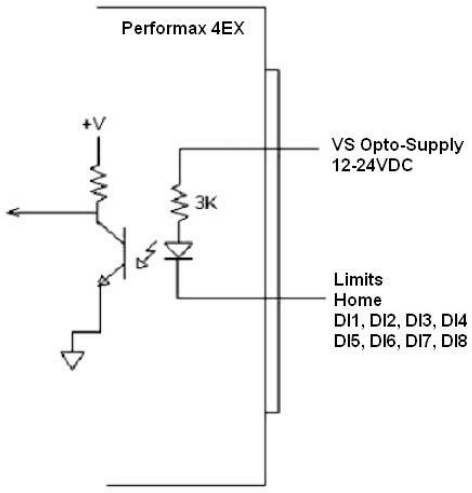
E. Connecting Digital Inputs and Outputs

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

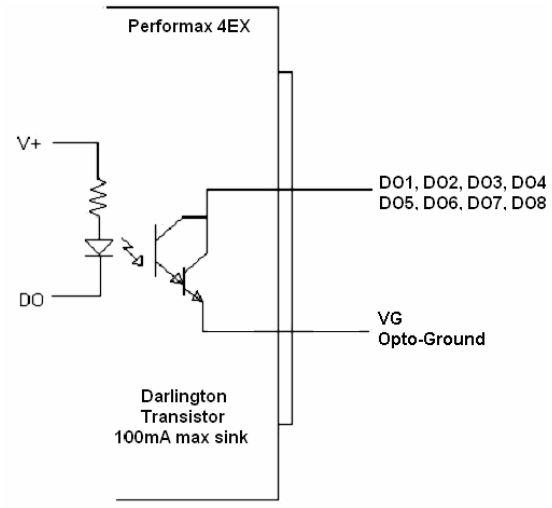


Note: Digital inputs and outputs are found on the bottom board of the PMX-4ET-SA. Therefore, pin out is the same regardless of top board choice.

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



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



For the opto-isolated outputs, the digital output signal will sink to VG opto-ground when the signal is turned on.

5. Electrical Specifications

Power Requirement

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

Communication Interfaces

Ethernet :	**Ethernet 10 Mbps - ASCII**

Pulse, Dir, Enable Outputs

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

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

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

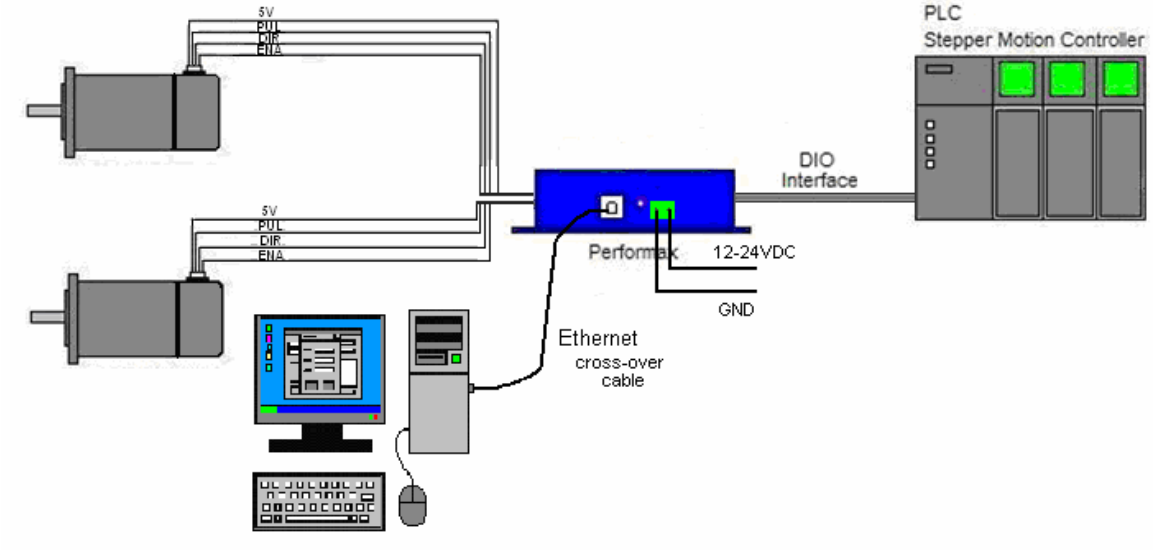
Digital Outputs

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

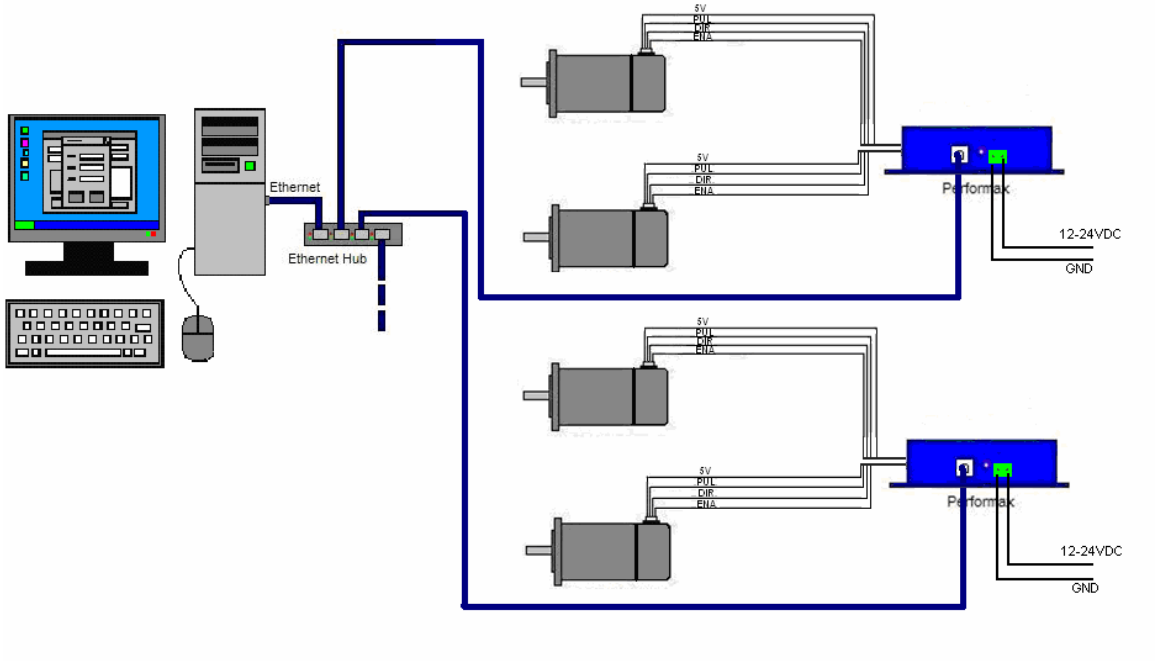
6. Getting Started

Typical Setup

Point-to-point



Network-based

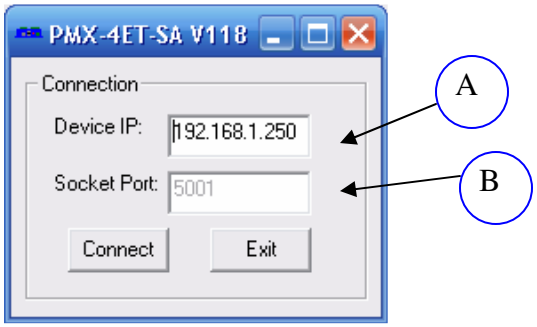


Windows GUI features

PMX-4ET-SA software comes with three different programs:

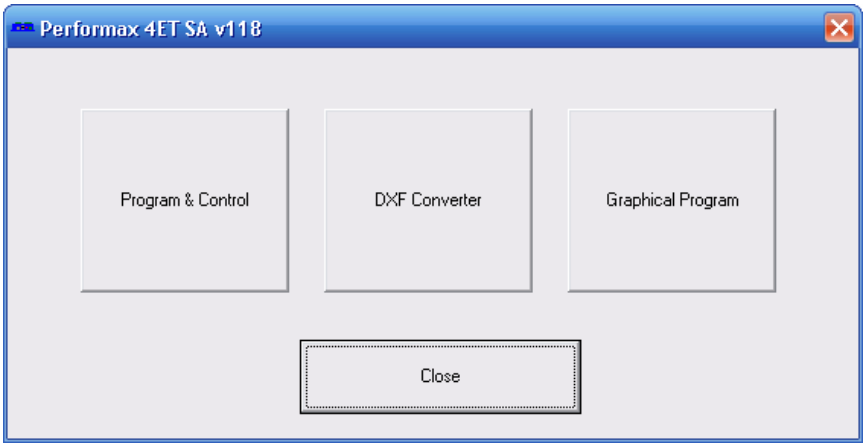
Program	Description
Program & Control	Allows the user to test all the features of the PMX-4ET-SA interactively. Also provides the interface for stand-alone programming
DXF Converter	Allows the user to load a DXF file and convert it to PMX-4ET-SA motion commands
Graphical Converter	Allows the user to create a stand-alone program graphically, instead of writing text-based code

Selecting Communication & Program

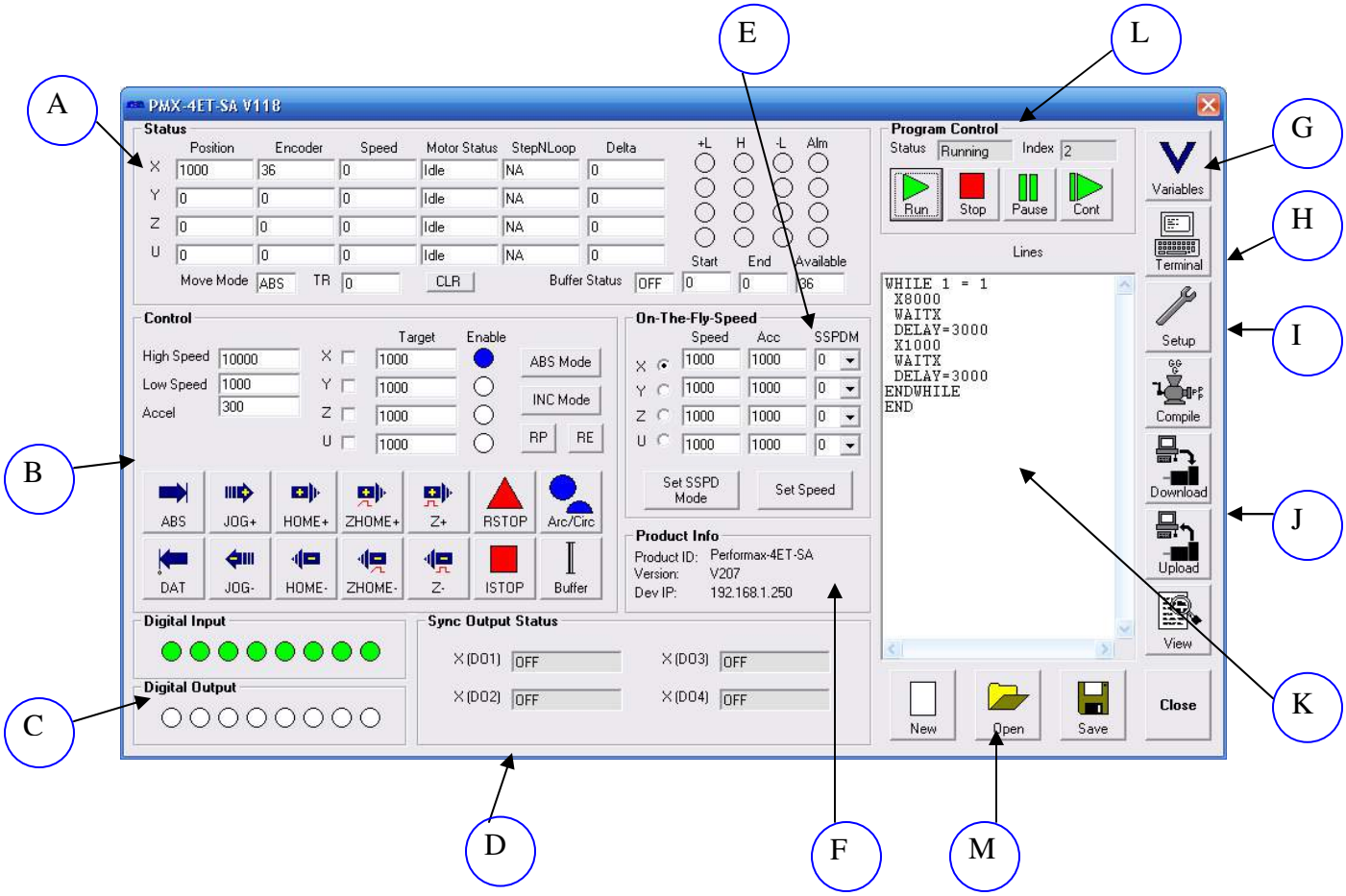


A. Device IP of the PMX-4ET-SA. The device IP later can be changed.
B. Port number of the socket that must be opened to being Ethernet TCP/IP communication. This socket number can not be changed. Default is 5001.

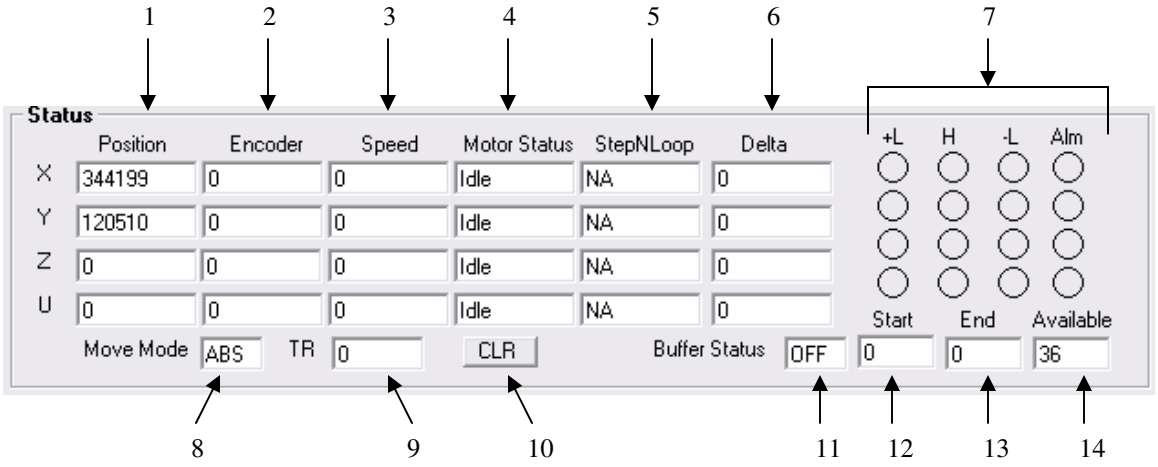
This screen allows the user to choose between the three different program types. To use a particular program, click on the corresponding button.



7. GUI: Program & Control



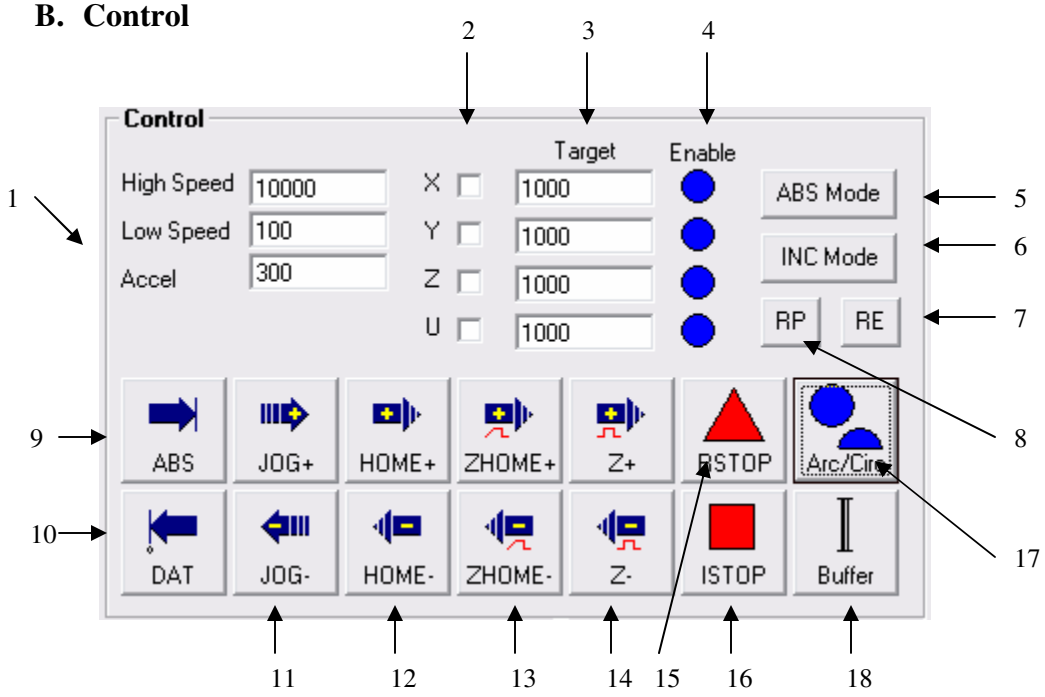
A. Status



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

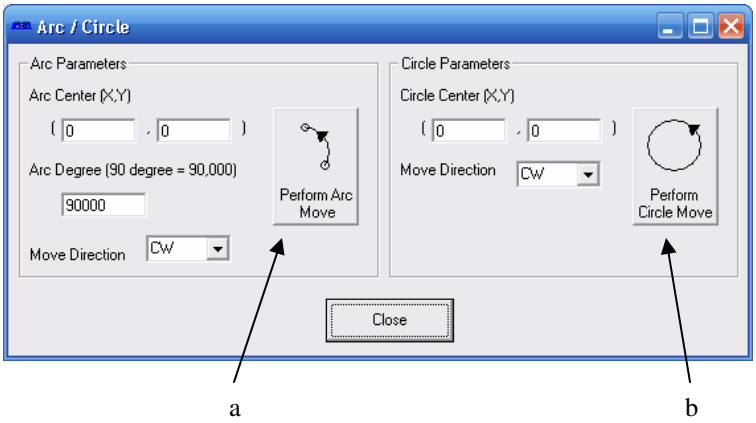
11. **Buffer move enable status**
12. **Buffer start:** This is the current index of the buffer. Note that the buffer is a 36 position ring buffer. (Used for buffer move mode only)
13. **Buffer end:** This is the current end of the buffer. Note that the buffer is a 36 position ring buffer. (Used for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

B. Control



1. **Select X/Y/Z/U** axis to control.
2. **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.
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. **Enable absolute move mode**
6. **Enable incremental move mode**
7. **Reset encoder** counter for axis
8. **Reset pulse** counter for axis. Not allowed if StepNLoop is enabled.
9. **Perform absolute move**. If more than one axis is selected, an interpolated move will result.
10. **Return to 0 position**. If more than one axis is selected, an interpolated move will result.
11. **JOG+/JOG-**: jogs the motor in positive and negative direction.

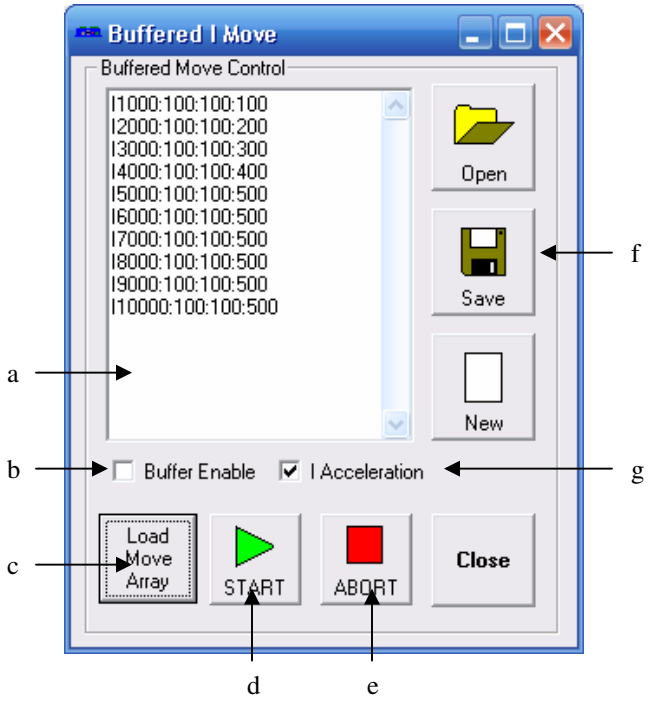
12. **HOME+/HOME-**: homing is done using only the home sensor. When the home sensor is triggered during homing, the position counter is reset to zero and the motor decelerates to low speed and stops. After homing, the position is not necessarily zero due to deceleration after the trigger of the home switch.
13. **ZHOME+/ZHOME-**: Home sensor and encoder index channel is used to home.
14. **ZOME+/ZOME-**: Only encoder index channel is used to home.
15. **RSTOP** – the motion is stopped with deceleration.
16. **ISTOP** – the motion is immediately stopped without deceleration.
17. **Arc/Circle Tool** – Clicking on this button will provide the user with an interface to perform Arc/Circle XY moves.



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.

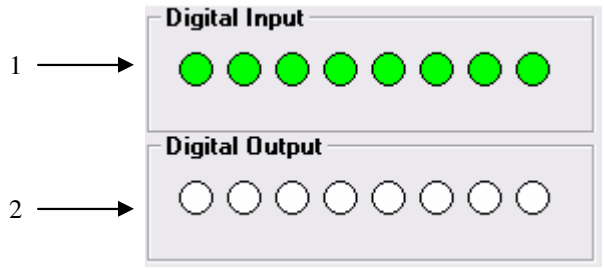
Note that after an arc or circle move is started, the position/speed values of the main control window will not begin to update until the above window is closed.

18. **Buffer move Tool** – Clicking on this button will provide the user with an interface to load buffer move commands to the PMX-4ET-SA.



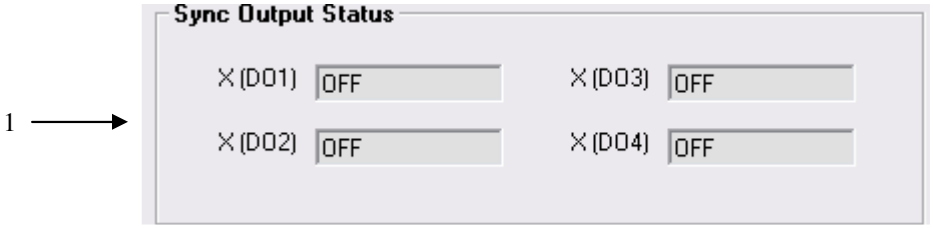
a. **Buffer Array List –** Enter the desired list of buffer commands here. Once the list is loaded, if the number of commands is greater than 36 (max buffer size), the program will automatically send the remaining commands to the PMX-4ET-SA as spaces clears up in the buffer.
b. **Buffer enable –** Enable/Disable 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.

C. Digital Input/Output



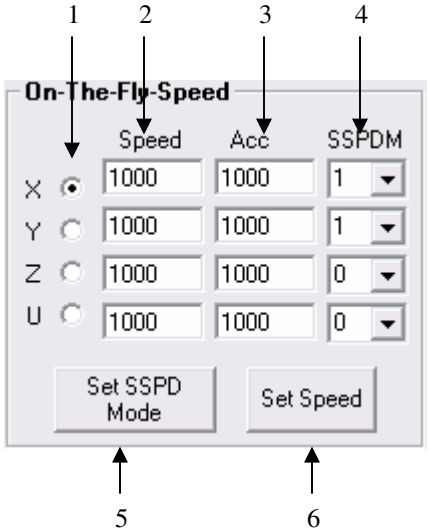
1. **Digital input status** DI1-DI8
2. **Digital output status** DO1-DO8. To turn on/off a digital output, click on the corresponding circle.

D. Sync Outputs



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

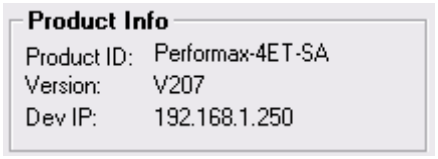
E. On-The-Fly-Speed Control



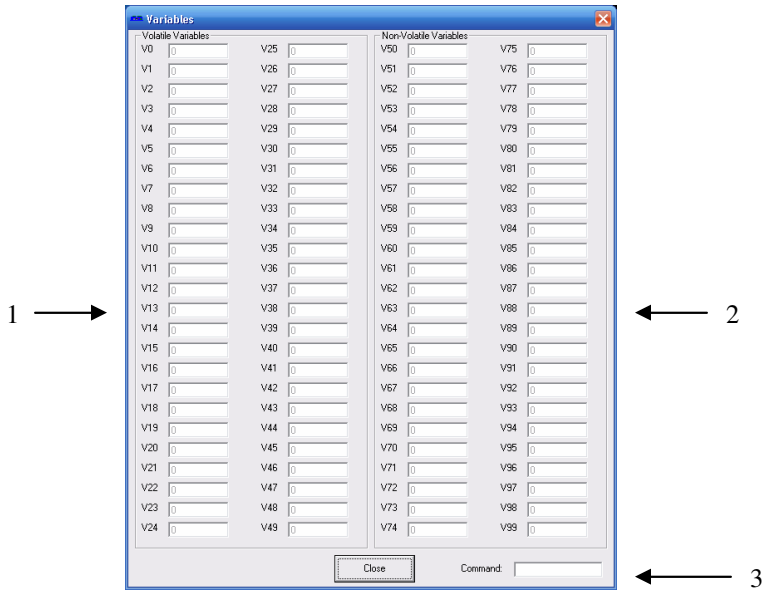
1. **Select X/Y/Z/U** axis.
2. **Select destination speed** of the axis.
3. **Select the acceleration** used during an on-the-fly speed change.

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.

F. Product Information

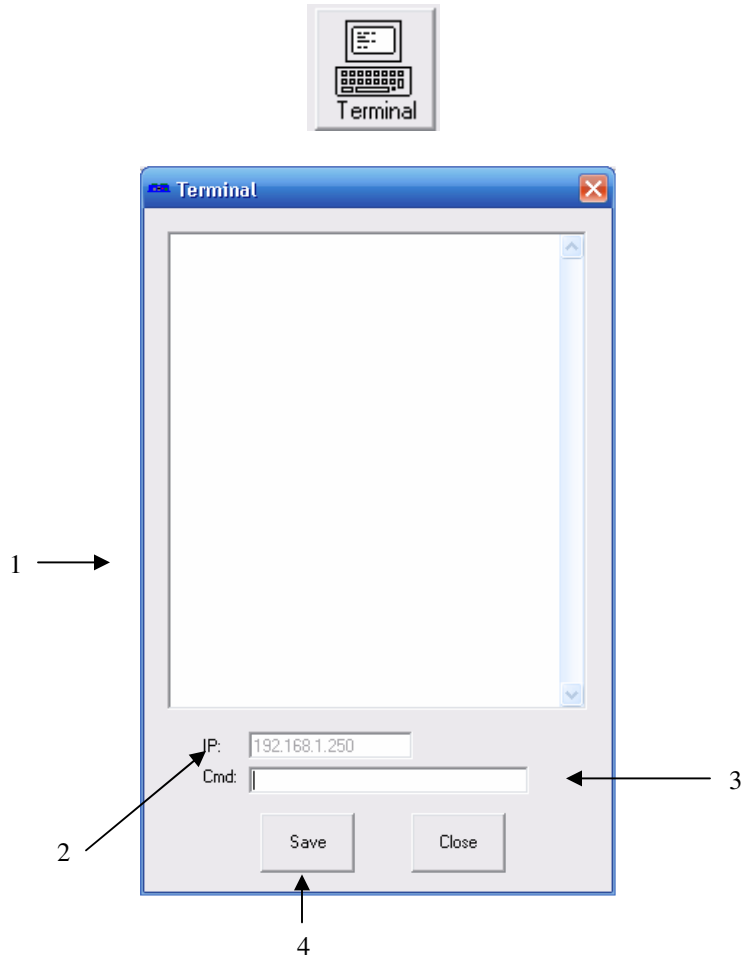


G. Variable Status



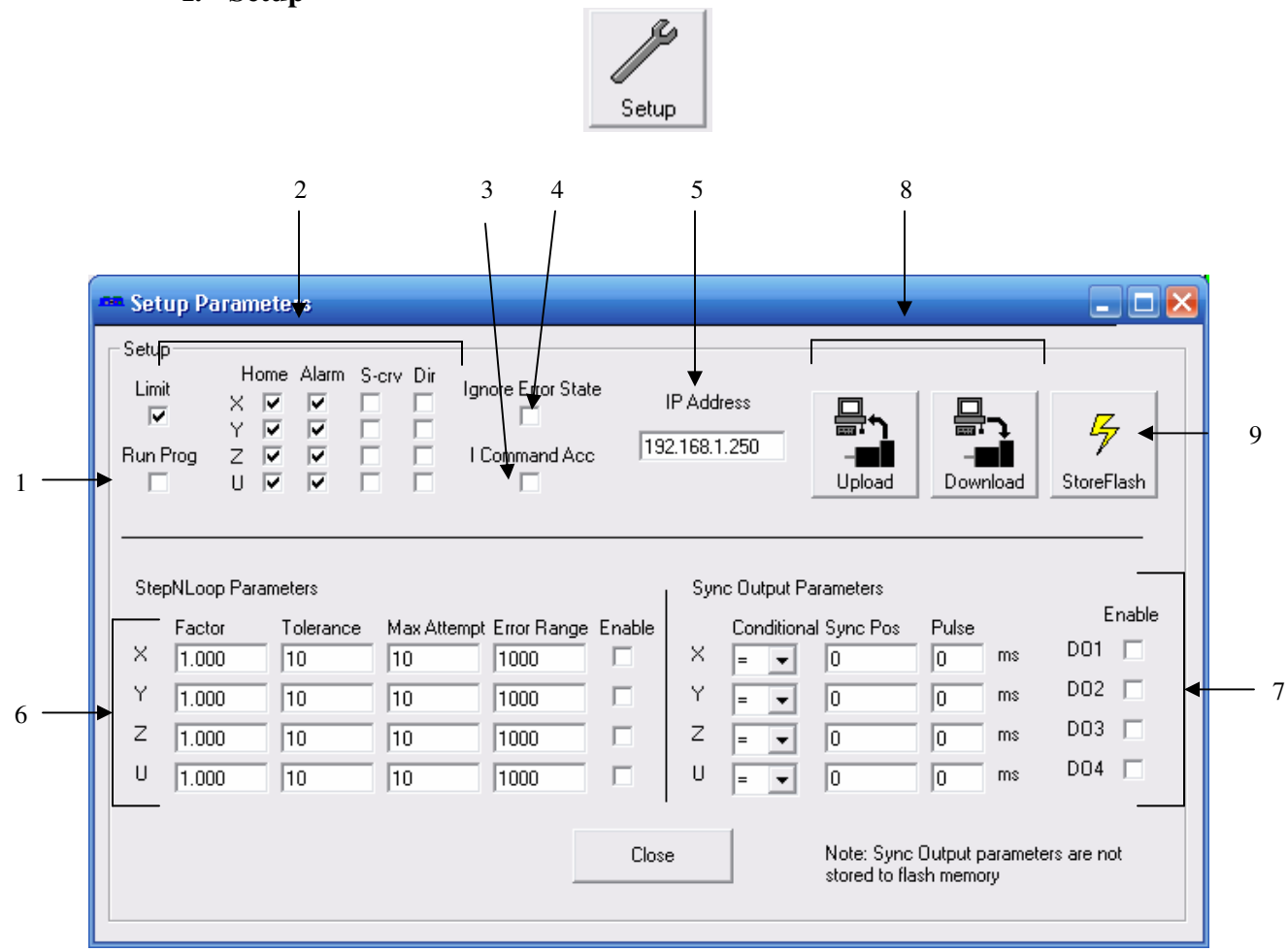
1. **Volatile Variables –** Status of volatile variable V0-V49
2. **Non-volatile Variables –** Status of non-volatile variable V50-V99
3. **Command line –** Set variables using V[0-99]=[value] syntax

H. Terminal



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

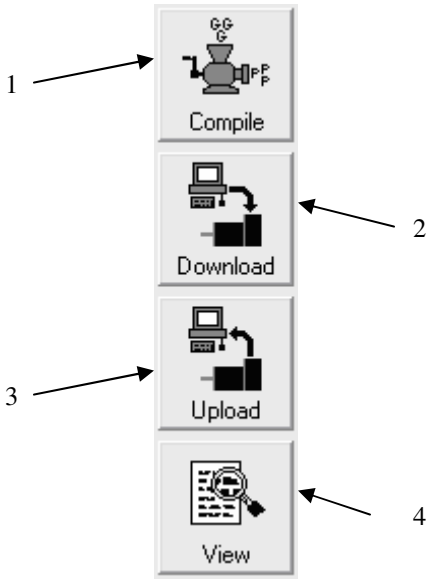
I. Setup



1. **Run Prog** – Click and perform a store to flash to have a standalone program run on boot up
2. **Polarity/S-curve**:
    a. Set home/alarm/dir polarity for X/Y/Z/U axes. Note that limit polarity is fixed either high/low for ALL axes.
    b. Set s-curve enable/disable for each axes.
3. **I accel** – Enable/Disable I move acceleration (used with buffered I commands)
4. **Ignore Error State** – Enable/Disable ignore error state parameter. If enabled, hitting a limit switch will only stop the motor. It will NOT cause the controller to go to an error state.
5. **IP Address:**
    a. Set IP address of the device
6. **Set StepNLoop** parameters

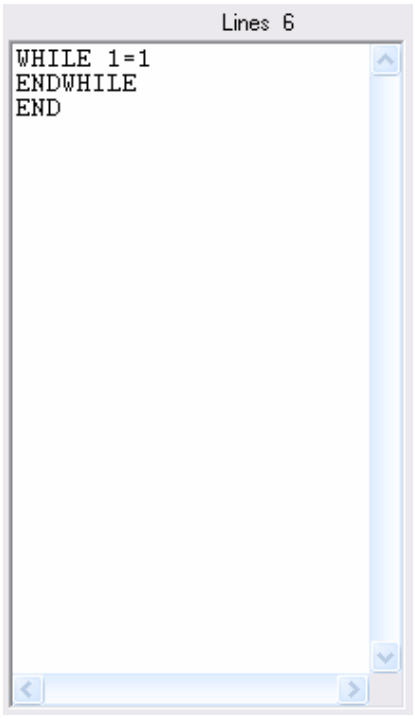
7. **Set Sync output** parameters (Note that sync output parameters are not stored to flash memory)
8. **Upload/Download** parameters to and from RAM
9. **Store** parameters to flash memory

J. Standalone Program Compile/Download/Upload/View

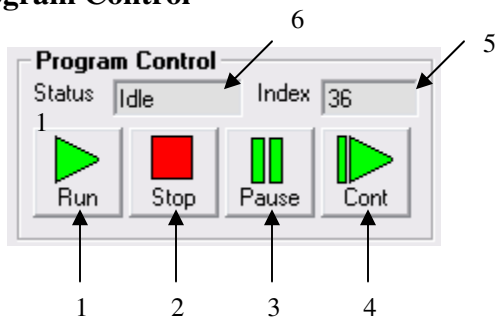


1. **Compile** – Compile the standalone program
2. **Download** – Download the compiled program
3. **Upload** – Upload the standalone program from the controller
4. **View** – View the low level compiled program

K. Standalone Program Editor



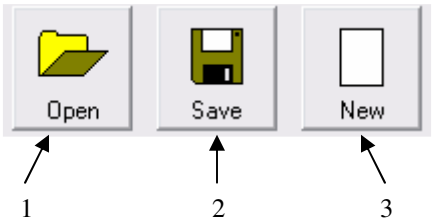
L. Standalone Program Control



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. **Index** – Current line of low-level code that is being executed.
6. **Status of standalone program**:
    i. Idle – Program is not running.
    ii. Running – Program is running.
    iii. Paused – Program is paused.

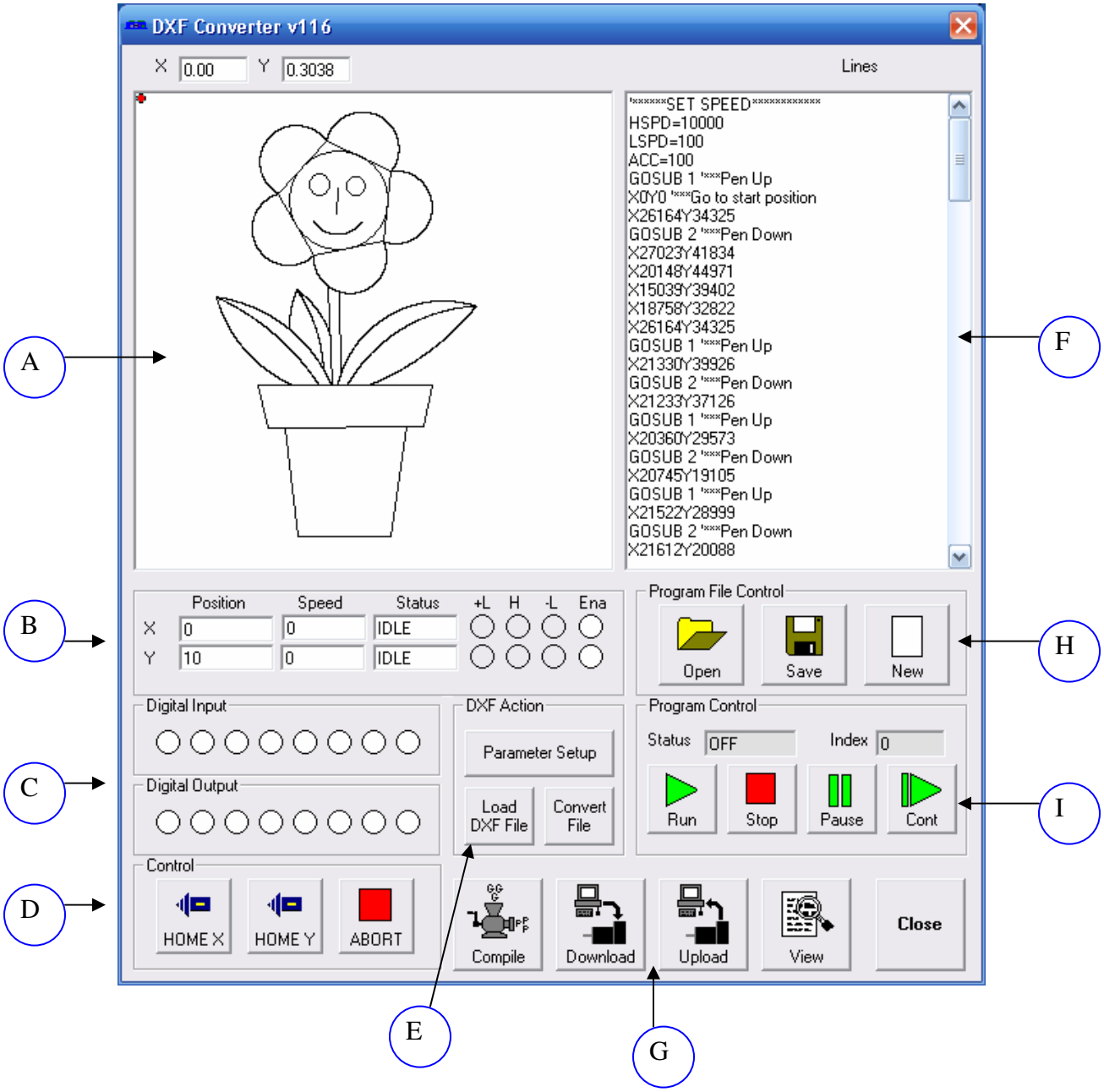
iv. Error – Program is in an error state.

M. Program File Control

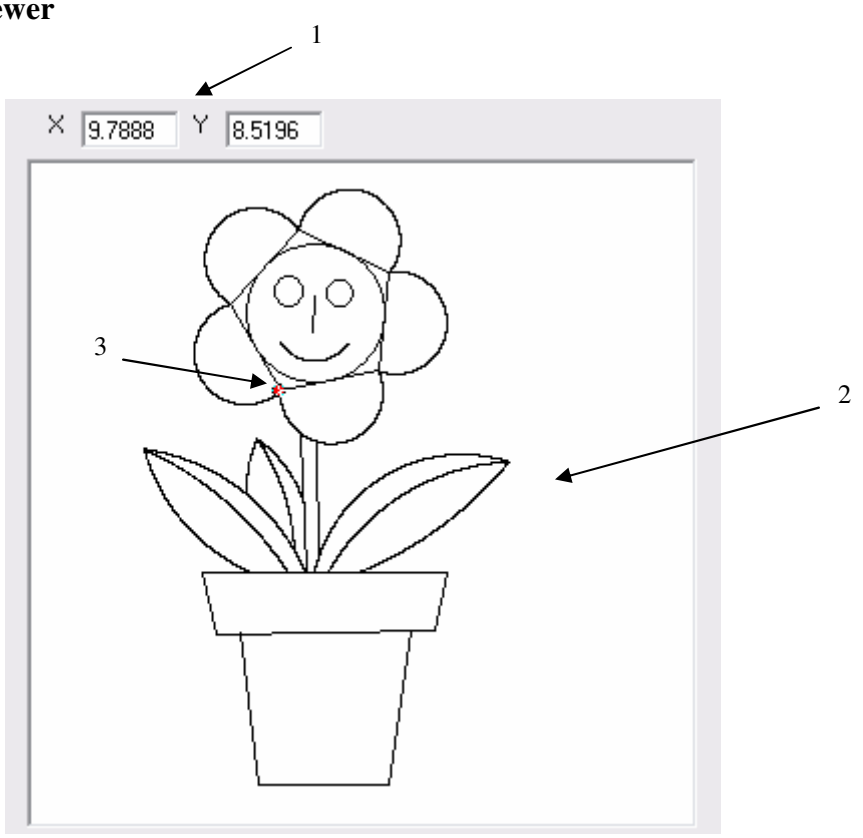


1. **Open** – Open standalone program
2. **Save** – Save standalone program
3. **New** – Clear the standalone program editor

8. GUI: DXF Converter

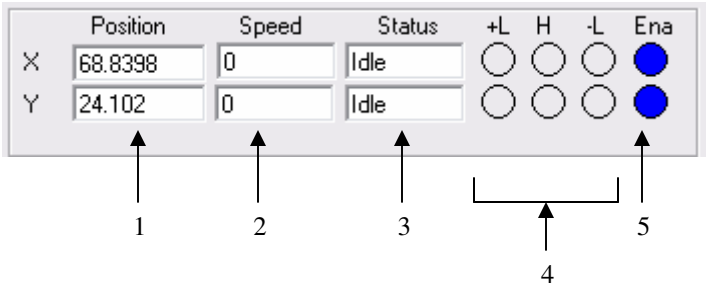


A. DXF Viewer



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

B. Status

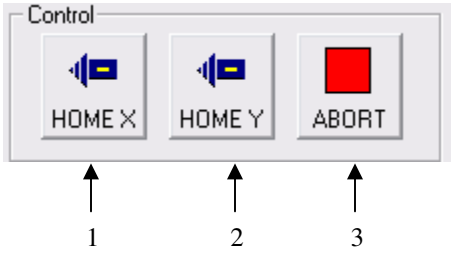


1. **Current inch/mm position** (X,Y axes).
2. **Current speed** (X,Y axes).
3. **Motor status** (X,Y axes)
   i. Idle – motor is not moving.
   ii. Accel – motor is accelerating
   iii. Const – motor is running in constant speed
   iv. Decel – motor is decelerating
   v. +LimError – plus limit error
   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

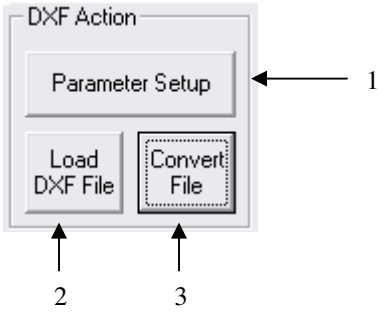
*See **Section C** of "GUI: Program & Control"*

D. Control

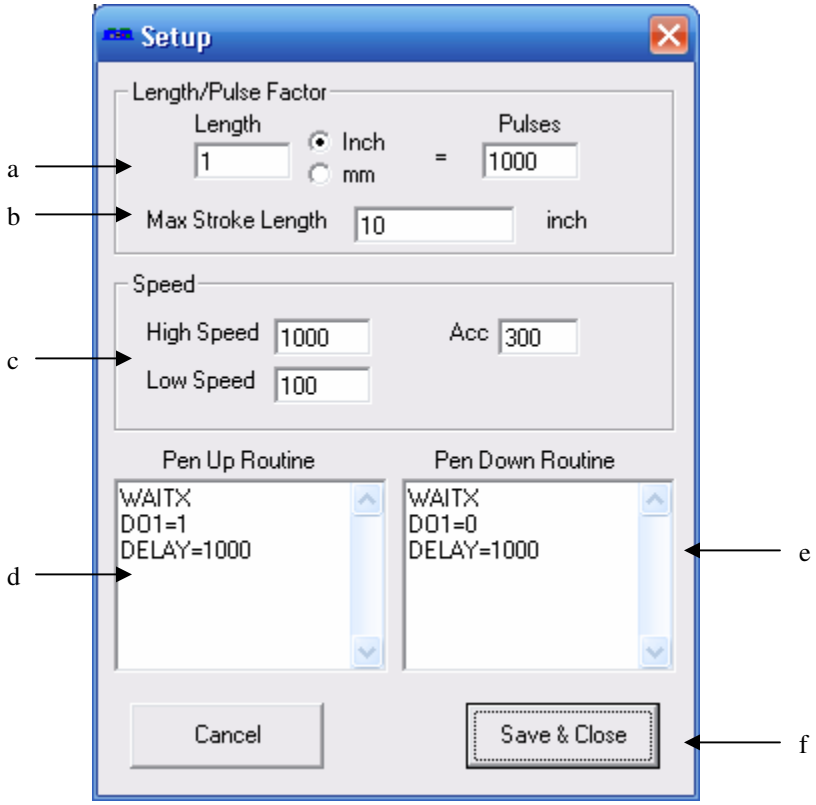


1. **Home x-axis** to the negative direction
2. **Home y-axis** to the negative direction
3. **Abort** all movement

E. DXF Action



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



   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

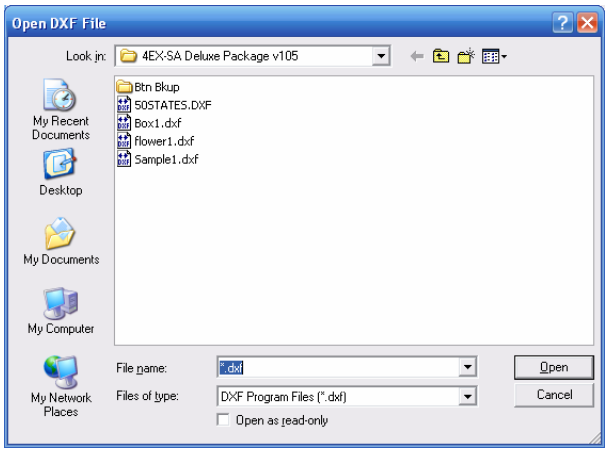
   c. High Speed, Low Speed and Acceleration settings

   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:

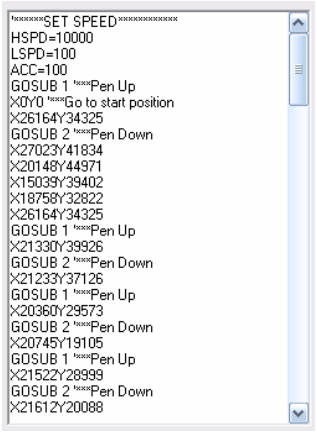


Select the desired DXF file and click "Open". At this point, the selected DXF file will be previewed in the DXF Viewer box.

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



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 J** of "GUI: Program & Control"*

H. Program Control

*See **Section L** of "GUI: Program & Control"*

I. Program File Control

*See **Section M** of "GUI: Program & Control"*

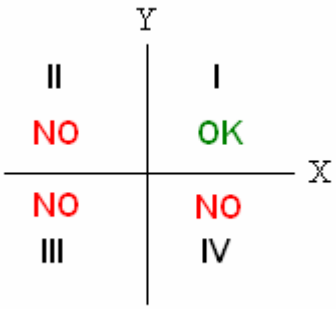
DXF Converter – Important Notes:

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 46399 pulses on arc or circular moves. To keep your radius moves smaller than 46399, decrease the **Length/Pulse Factor**.

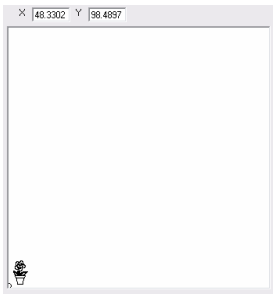
Picture positioning: A DXF file cannot contain an any or part of an image that is not in quadrant I (i.e. all x,y positions of the DXF need to be positive). See figure below:



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

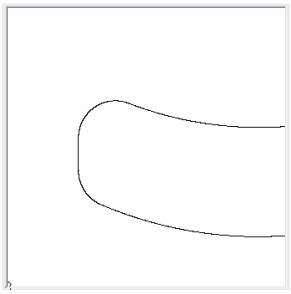
Scaling the DXF Viewer Box:

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



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

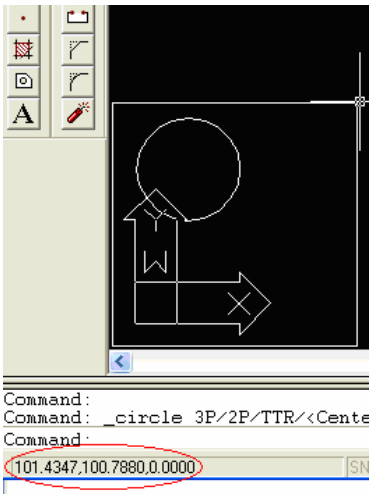
In the case where you do not see any picture or the picture is cut off, the window is zoomed in too much. See below:



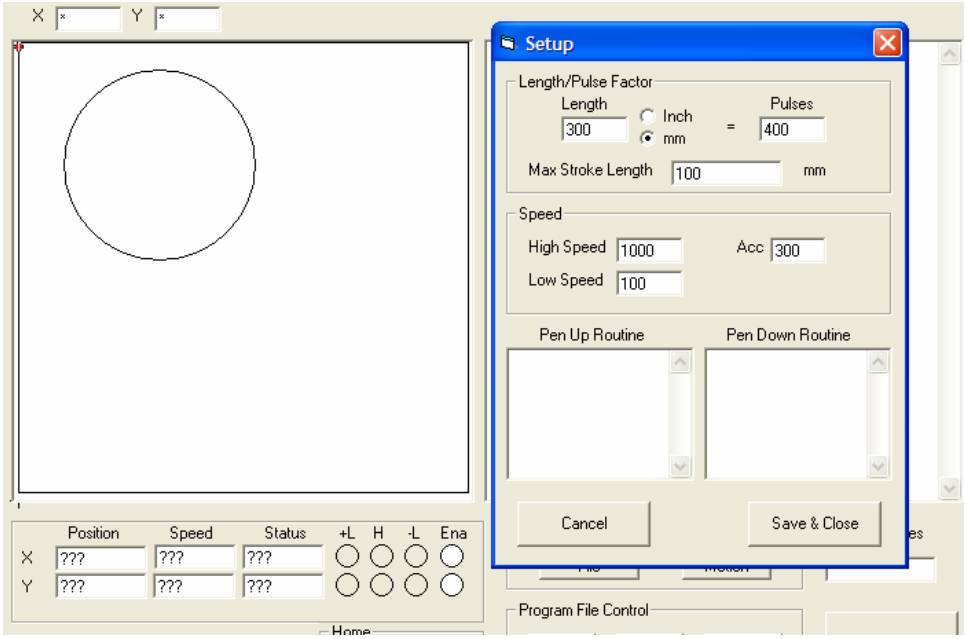
To zoom out, decrease the **Max Stroke Length** parameter.

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

For example, you can see in below in 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.



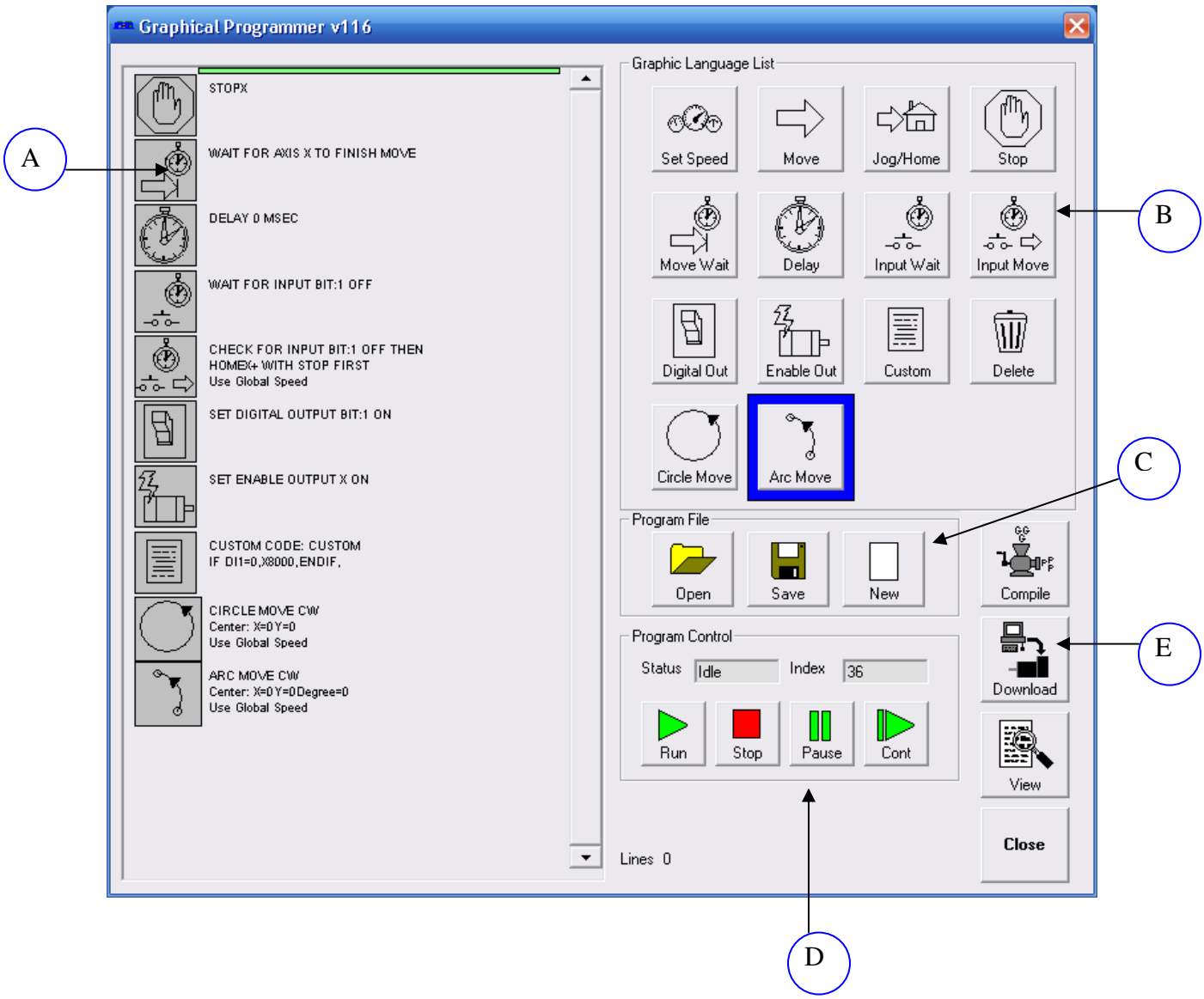
When loading the DXF, the **Max Stroke Length** should be set to 100 in order to properly show the picture. See below:



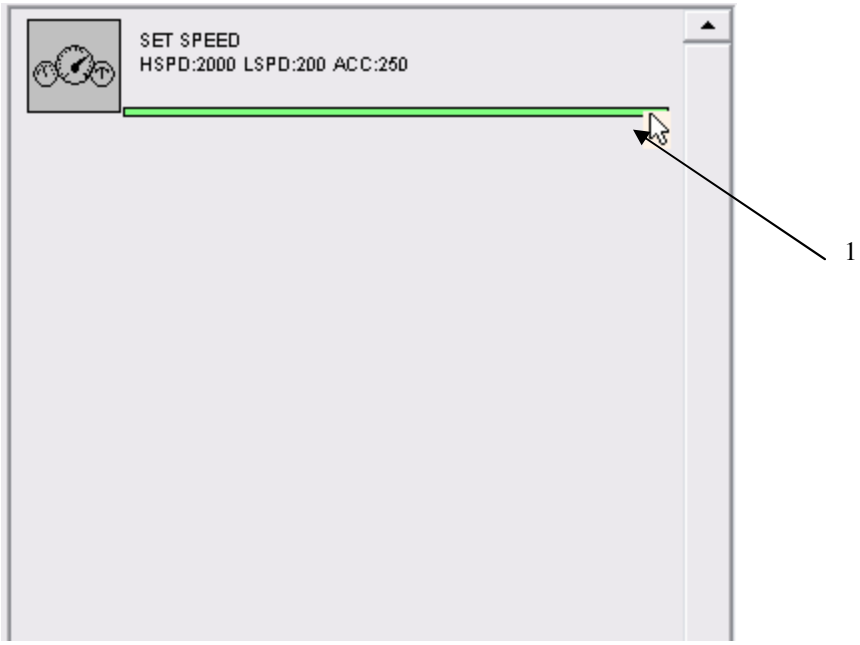
Scaling your XY table:

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

9. GUI: Graphical Programmer



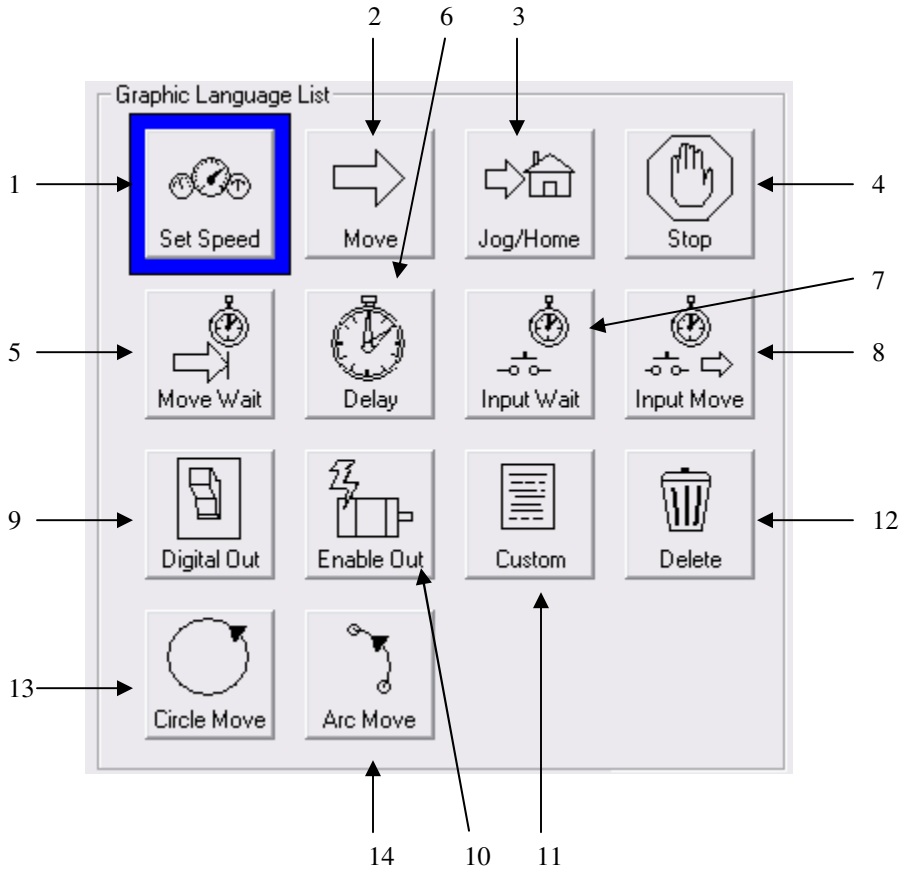
A. While Sequence Box



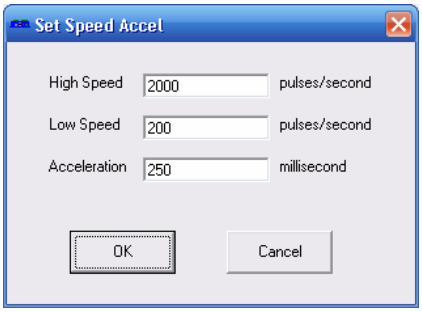
1. The following box contains the sequence that is executed in a continuous while loop. To enter things into the while loop, first click on an item in the Graphic Language List box and then move the cursor directly under the last item of the sequence (see above).

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

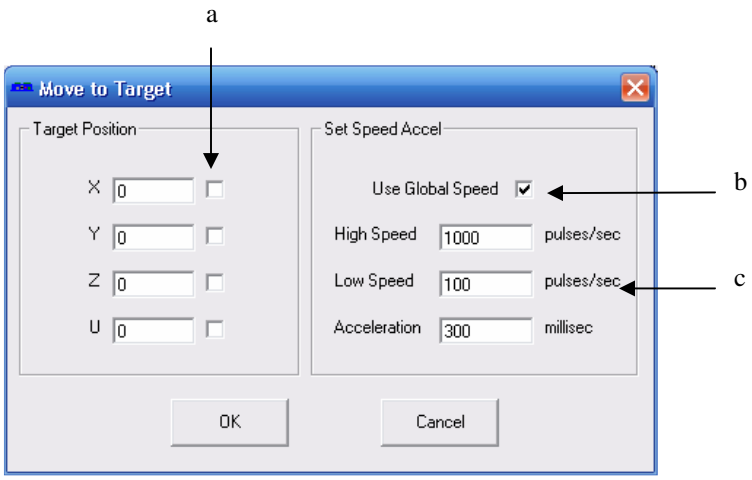
B. Graphic Language List Box



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.

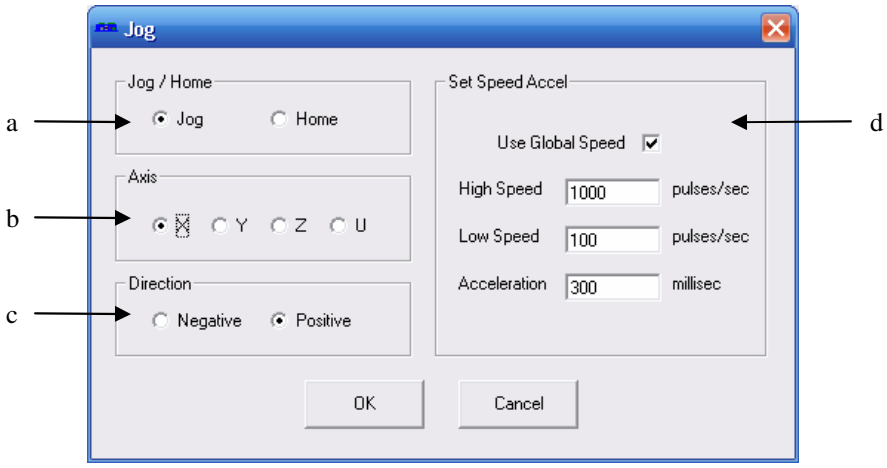


2. **Move Object:** Perform absolute move commands on selected axes



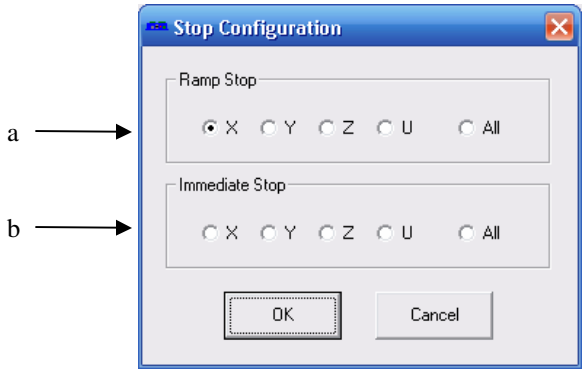
   a. Select 1-4 axes to move. If more than one axis is selected, the move will be linear interpolated.
   b. Check to use global speeds specified in the Set Speed Object
   c. If "Use Global Speed" is not checked, the move will use the following local speed settings

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



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***

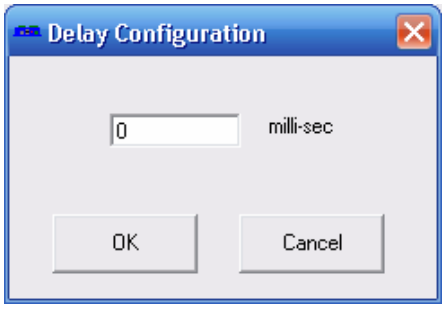
4. **Stop Object:** Perform a ramp stop or immediate stop on 1 or all axes



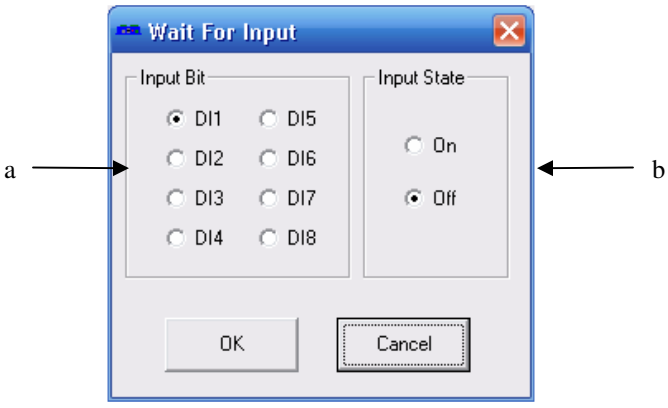
a. If this move is a ramp stop, select 1 or all axes
b. If this moves is an immediate stop, select 1 or all axes

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

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

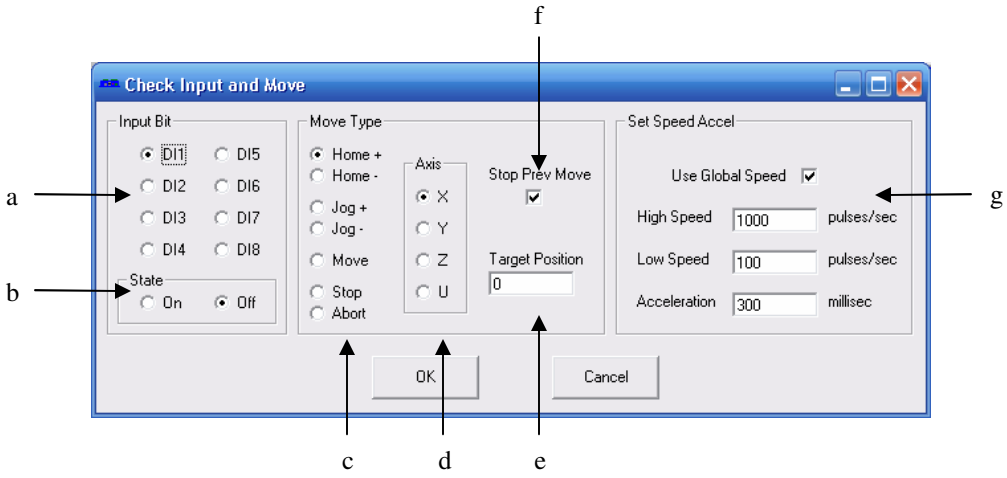


7. **Wait Input Object:** Wait until a single input is on/off before continuing to execute



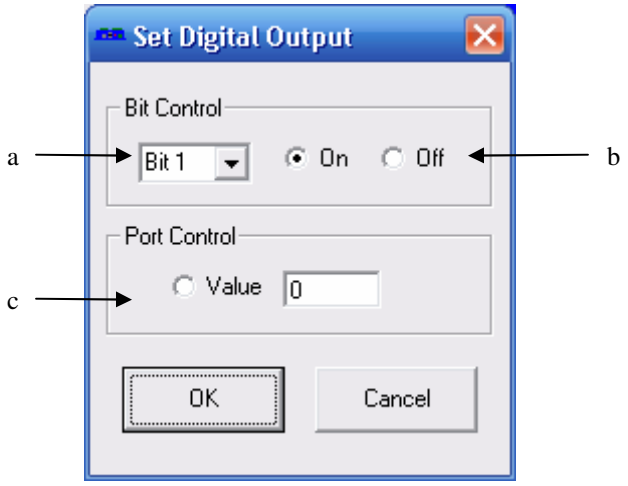
a. Select the digital input
b. Make the condition on or off

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



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 first stop the previous move before processing this setting
g. *See* ***Graphic Language List Box: Section 2b and 2c***

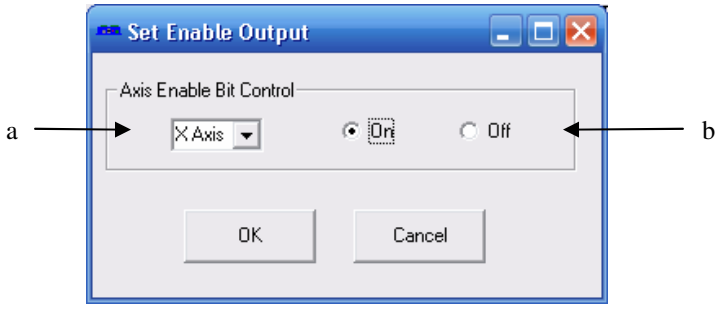
9. **Digital Out Object:** Set digital output status



a. Select output bit

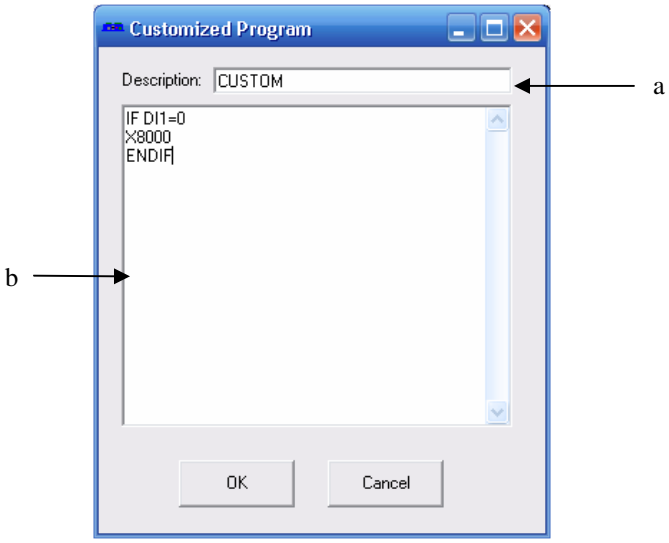
b. Select on/off
c. To set entire 8-bit output status, click the "Value" radio button and enter the 8-bit number in the field

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



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

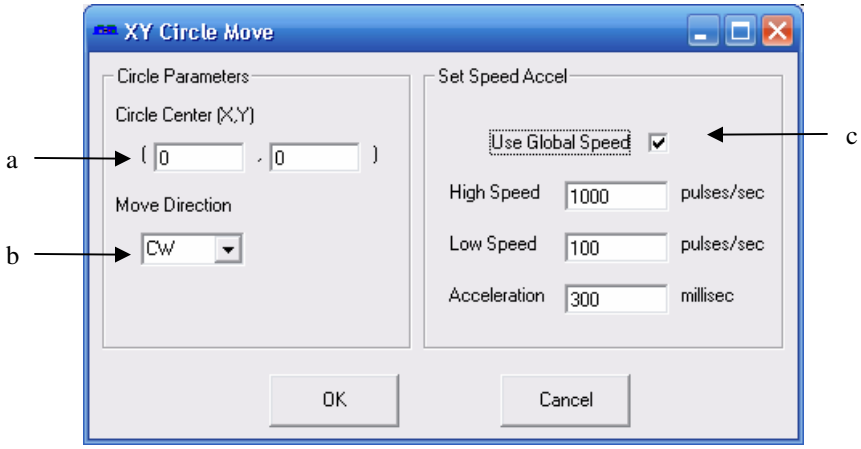
11. **Custom**: Write a custom program to insert into the sequence.



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.

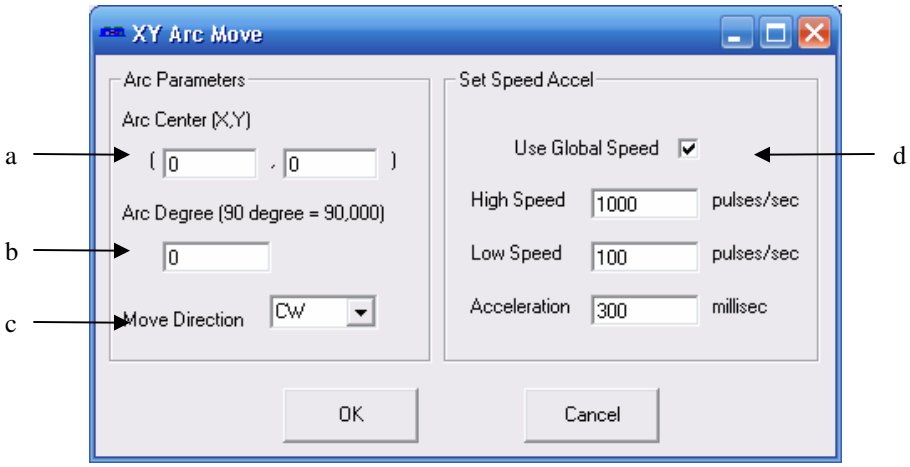
13. **Circle Move:** Create a circle interpolation move



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



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 Control Box

*See **Section M** of "GUI: Program & Control"*

D. Program File Box

*See **Section L** of "GUI: Program & Control"*

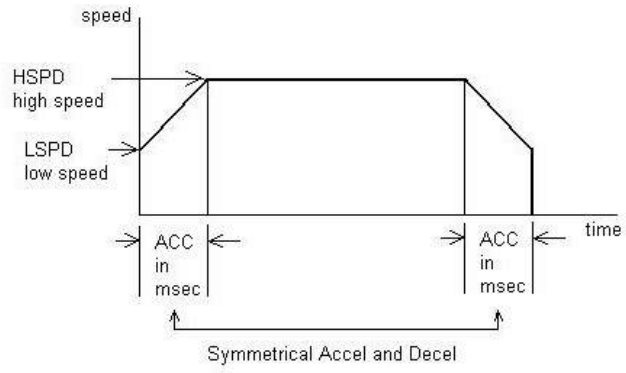
10. Motion Control Feature Overview

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

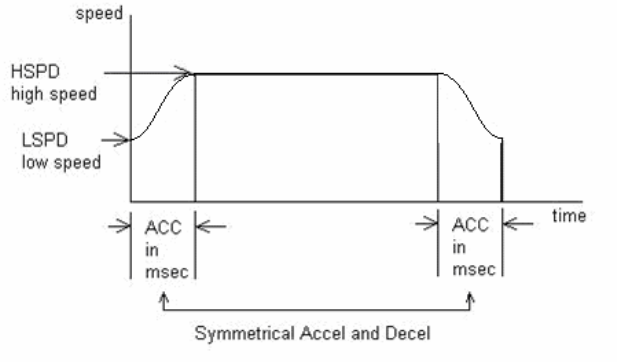
Motion Profile

PMX-4ET-SA can generate up to 4M pulses per second pulse rate.

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



PMX-4ET-SA can also use an s-curve velocity profile by issuing the **SCV[axis]** command. S-curve velocity profile is shown below:



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

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.

By default, all moves use the global speed settings, unless ALL parameters (i.e. high speed, low speed, and acceleration) for a certain axis are 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
```

Interpolated moves use the global speed settings at all times.

Notes on speed and acceleration settings:

The minimum value of the low speed setting for an axis depends on the high speed setting of the axis. If a low speed setting is used that is outside of the corresponding window, the controller will go to an error state. See chart below:

SPEED WINDOWS

SSPDM value	Lowest Speed [pps]	Highest Speed [pps]
0	SSPD not used	SSPD not used
1	1	65,000
2	2	130,000
3	5	325,000
4	10	650,000
5	20	1,300,000
6	50	3,200,000
7	100	6,000,000

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

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

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

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

ACCELERATION WINDOWS

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

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

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

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

Examples:

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

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

On-The-Fly Speed Change

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

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

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

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

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

2) To set acceleration of the on-the-fly speed change, use the **ACC[axis]** command. Set the acceleration before calling the **SSPD[axis]** command. See *"ACCELERATION WINDOWS"* chart in previous section.

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

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

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

Pulse Speed

Current pulse rate can be read using the **PS** command. For units, see chart below:

PMX-4ET-SA mode	**Speed units**
StepNLoop disabled	Pulse / sec
ALL interpolated moves	Pulse / sec
StepNLoop enabled and non-interpolated move	Encoder counts / sec

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]

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

9	Alarm Error

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

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

Individual/Linear Interpolation moves have two modes, incremental mode, done by issuing the **INC** command, and absolute mode, done by issuing the **ABS** command. Under incremental mode, the command **X1000** will move the motor by 1000 from the current position. Under absolute mode, the command **X1000** will move the motor to position 1000.

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.

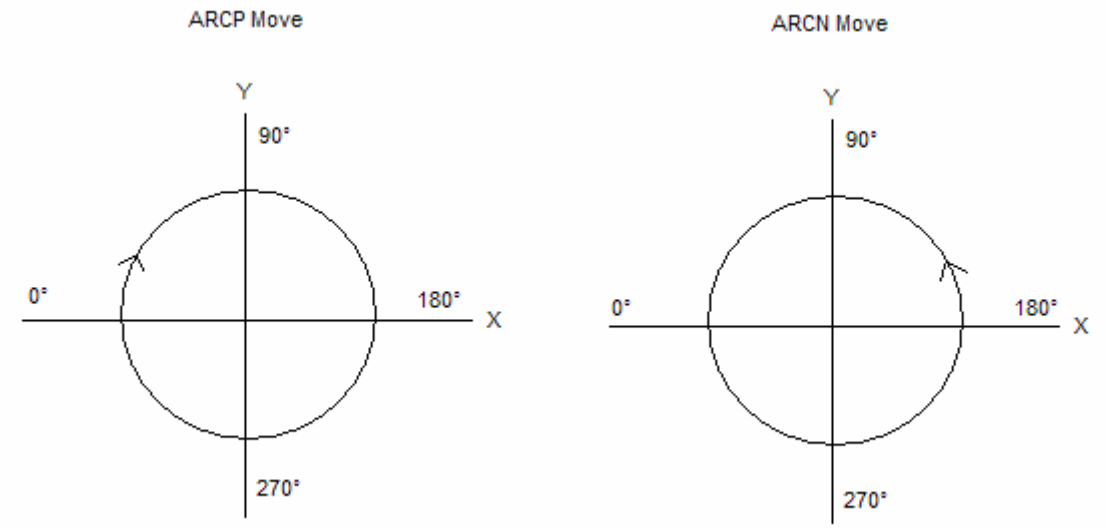
Notes: PMX-4ET-SA does not allow a radius larger than 46399 pulses on arc or circular moves. All arc or circular moves are interpreted as absolute moves (even if the PMX-4ET-SA is in incremental move mode)

Arc Interpolation Moves

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

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

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



Note: PMX-4ET-SA does not allow a radius larger than 46399 pulses on arc or circular moves. All arc or circular moves are interpreted as absolute moves (even if the PMX-4ET-SA is in incremental move mode)

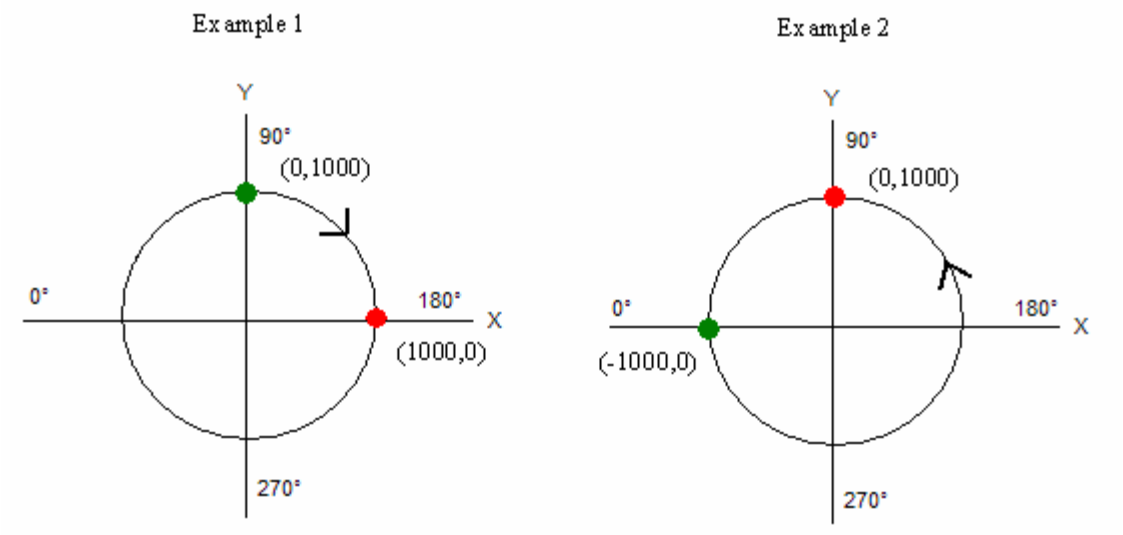
Arc Move Example

Example 1:

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

Example 2:

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



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, use following command* ***I1000:2000:3000:250***

Manual Acceleration Control

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

Automatic Acceleration Control

To have the controller acceleration or deceleration automatically, enable automatic buffered move acceleration with the **IACC** command. In this case, the speed acceleration profile will be automatically generated between sequential buffered moves. The acceleration value used for automatic acceleration control is in the global acceleration value (**ACC** command).

Linear Interpolations buffer move size is 36 points. Buffered move mode is turned on with the **BO** command turned off with the **BF** command. 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 can not be used while StepNLoop is enabled.

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

Homing

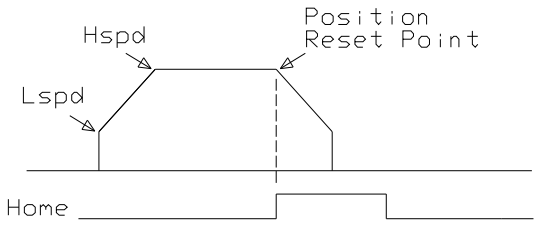
Use the **H** command for homing the motor. Use the following format:

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

Four homing modes are available.

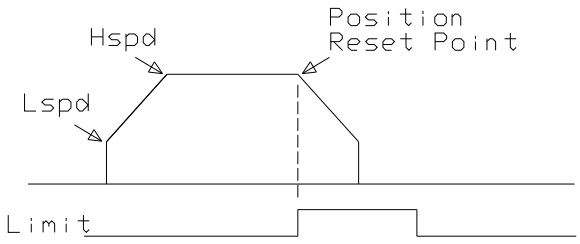
0 - Using home switch
1 - Using limit switch
2 - Using home switch and encoder index channel
3 – Using encoder index channel only

Homing Mode 0



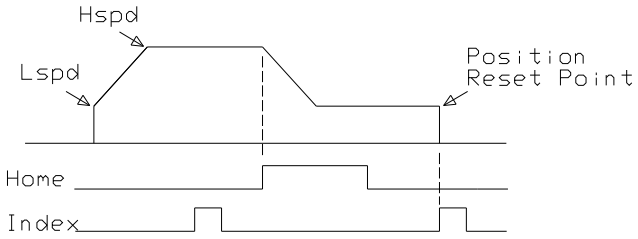
In homing mode 0, the axis ramps from low speed to high speed, then maintains the high speed until the home sensor is triggered. At the home sensor trigger, pulse and encoder position counters reset to zero and the deceleration is done to ensure smooth ramp down to low speed. At the end of the home routine,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

Homing Mode 1



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

Homing Mode 2



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

Homing Mode 3

In homing mode 3, the axis starts at low speed and maintains the low speed until the index channel of the encoder is triggered at which point the motion stops and pulse and encoder position counters are reset to zero.

Homing Examples

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

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

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

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

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 decelerate stop, use **STOP[axis]** command. Use **STOP** command to decelerate 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

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

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

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

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

Ignore Error State Mode

It is possible to have the controller stop when a limit/alarm is triggered, but not go into an error state. To do this, enable the "Ignore Error State Mode" with the **IERR** command.

Enable Outputs

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

Digital Outputs

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

Sync Outputs

PMX-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

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

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

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

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

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

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

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

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

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

Digital Inputs

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

StepNLoop Closed Loop Control

PMX-4ET-SA has closed loop position control algorithm called StepNLoop control for accurate positioning of the motor using the integrated encoder.

StepNLoop control does the following operations:

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

Following are configuration required for StepNLoop control:

SNL Parameter	Description
Pulse/Encoder Ratio	Number of pulse counts/number of encoder counts per revolution of motor. Use **SLR[axis]** command to set the ratio. Value must be in the range [0.001 , 999.999].
Tolerance Range	When the actual encoder position is within desired encoder position by this tolerance range, no position correction is done. Use **SLT[axis]** command to set the tolerance range.
Correction Range	When the actual encoder position is within desired encoder position by this correction range, position correction is done when idle. If the actual encoder position is outside of correction range, the motor status goes to error state. Use **SLE[axis]** command to set the correction range.
Correction Attempt Number	This is the maximum number of correction tries that the controller will attempt. If the correction cannot be done within this number of tries, the motor status goes to error state. Use **SLA[axis]** command to set the maximum correction attempt number.

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

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

Following are the StepNLoop status values:

Value	Description
0	Idle
1	Moving
2	Correcting
3	Stopping
4	Aborting
5	Jogging
6	Homing
7	ZHoming
8	Correction Range Error. To clear this error, use **CLR[axis]** command
9	Correction Attempt Error. To clear this error, use **CLR[axis]** command.
10	Stall Error. **DX[axis]** value has exceeded the **SLE[axis]** value. To clear this error, use **CLR[axis]** command.
11	Limit Error
12	N/A

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

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

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

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

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

Device IP Address

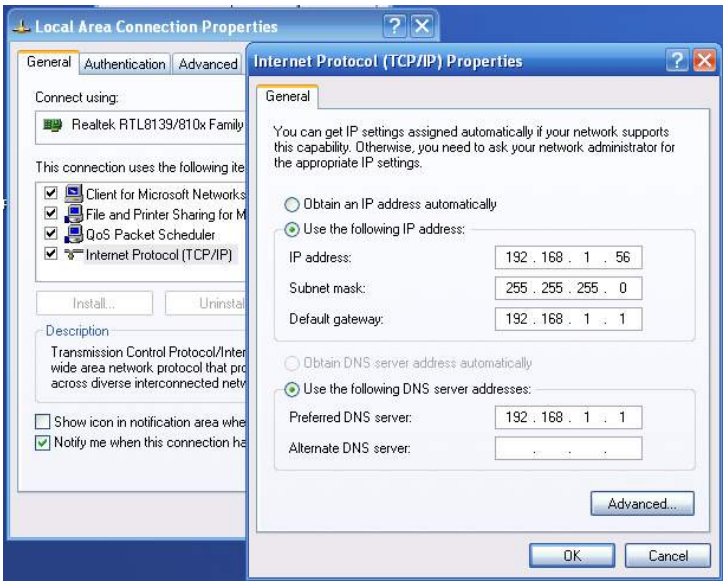
Set the IP address of the PMX-4ET-SA module using the **IP** command. See default IP/socket settings below:

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:



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 Program Specification

Standalone Program Specification:
Memory size: 7650 assembly lines ~ 44 KB.
Execution speed: ~ 2 ms per compiled line.
Note: Each line of pre-compiled code equates to 1-4 lines of assembly lines.

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

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

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

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.

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	**OFF**	ON	OFF	OFF	ON
3	**ON**	ON	OFF	OFF	ON
4	ON	ON	**ON**	OFF	ON
5	ON	ON	**OFF**	OFF	ON
6	ON	ON	OFF	OFF	**OFF**
7	ON	ON	OFF	OFF	**ON**

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

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:

- Device IP
- Polarity settings
- S-curve settings
- StepNLoop settings
- Buffered interpolated move automatic acceleration (**IACC**)
- Ignore Error State mode (**IERR**)
- Second half of general purpose variables (V50-V99)

- Automatic program run on power up

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

11. Ethernet Communication Protocol

PMX-4ET-SA is 10Mbps Ethernet ASCII communication.

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]

12. ASCII Language Specification

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

Command	Description	Return
ABORT	Immediately stops all the motor if in motion. Abort turns off the buffered move.	OK
ABORTX ABORTY ABORTZ ABORTU	Immediately stops individual motor if in motion. Abort turns off the buffered move.	OK
ABS	Turns on absolute move mode	OK
ACC	Returns current global acceleration value in milliseconds.	
ACC=[Value]	Sets global acceleration value in milliseconds.	OK
ACCX ACCY ACCZ ACCU	Returns current individual acceleration value in milliseconds.	
ACCX=[value] ACCY=[value] ACCZ=[value] ACCU=[value]	Sets individual acceleration value in milliseconds.	OK
ARCP[X]:[Y]:[θ]	XY Arc interpolation move (CW direction)	OK
ARCN[X]:[Y]:[θ]	XY Arc interpolation move (CCW direction)	OK
BF	Disable buffered move	OK
BO	Enable buffer move on	OK
CIRP[X]:[Y]	XY Circular interpolation move (CW direction)	OK
CIRN[X]:[Y]	XY Circular interpolation move (CCW direction)	OK
CLRX CLRY CLRZ CLRU	Clears motor limit or alarm status bit. Also clears a StepNLoop errors	OK
DI	Returns 8 bits of general purpose digital input.	[0-255]
DI[1-8]	Returns bit status of general purpose digital input.	[0,1]
DO	Returns 8 bits of general purpose digital output value.	[0-255]
DO=[value]	Sets 8 bits of general purpose digital output.	OK
DO[1-8]	Returns bit of general purpose digital output value.	[0,1]
DO[1-8]=[value]	Sets bit of general purpose digital output.	OK
DN	Return device name	[4ET00-4ET99]
DN=[value]	Set device name. value must be in the range [4ET00, 4ET99]	OK
DXX DXY DXZ DXU	Get StepNLoop delta value of axis	
EO	Returns 4 bits of enable output value.	[0-15]
EO=[value]	Sets 4 bits of enable outputs.	OK
EO[1-4]	Returns bit of enable output value.	[0,1]
EO[1-4]=[value]	Set bit of enable outputs.	OK
EX=[value] EY=[value]	Set encoder value of axis	OK

EZ=[value] EU=[value]		
GS[0-31]	Call a defined subroutine	OK
HS	Returns global high speed setting	[1-6,000,000]
HS=[value]	Sets global high speed	OK
HSX HSY HSZ HSU	Returns individual high speed setting	[1-6,000,000]
HSX=[value] HSY=[value] HSZ=[value] HSU=[value]	Sets individual high speed	OK
HX[+/-][mode] HY[+/-][mode] HZ[+/-][mode] HU[+/-][mode]	Homes the motor in plus [+] or minus [-] direction using different homing mode.	OK
I[X axis]: [Y axis]: [Z axis]: [speed]	XYZ interpolated move. Target move values are separated by ':' character. Last value is the constant speed that will be used in the move.	OK
IACC	Get automatic acceleration during buffer interpolated move status	[0-1]
IACC=[0 or 1]	Set automatic acceleration during buffer interpolated move status	OK
IERR	Get the ignore limit/alarm error status	[0-1]
IERR=[0 or 1]	Set the ignore limit/alarm error status	OK
INC	Enable incremental move mode	OK
JE	Get joystick enable status	[0-15]
JE=[value]	Set joystick enable status	OK
JX[+/-] JY[+/-] JZ[+/-] JU[+/-]	Jogs the motor in plus [+] or minus [-] direction.	OK
LS	Returns global low speed setting	[1-6,000,000]
LS=[value]	Sets global low speed	OK
LSX LSY LSZ LSU	Returns individual low speed setting	[1-6,000,000]
LSX=[value] LSY=[value] LSZ=[value] LSU=[value]	Sets individual low speed	OK
MST	Returns all motor status, buffer move status, and move mode status Motor Status Bit 0 – accelerating Bit 1 – decelerating Bit 2 – constant speed Bit 3 – alarm input Bit 4 - + limit input Bit 5 - -limit input	[X motor status]: [Y motor status]: [Z motor status]: [U motor status]: [Buffer enabled]: [Buffer start]: [Buffer end]: [Available Buffer]: [MoveMode]:

	Bit 6 – home input Bit 7 - + limit error Bit 8 - - limit error Bit 9 – alarm error	
PE	Returns current encoder counter values of all 4 axes	[X Enc Position]: [Y Enc Position]: [Z Enc Position]: [U Enc Position]
POX POY POZ POU	Returns polarity setup Bit 0 – home input polarity Bit 1 – alarm polarity Bit 2 – limit polarity (X axis control polarity of all limits)	[0-7]
POX=[value] POY=[value] POZ=[value] POU=[value]	Sets polarity Bit 0 – home input polarity Bit 1 – alarm polarity Bit 2 – limit polarity (X axis control polarity of all limits)	OK
PP	Returns current pulse counter values of all 4 axes	[X Pulse Position]: [Y Pulse Position]: [Z Pulse Position]: [U Pulse Position]
PS	Returns current pulse speed values of all 4 axes	[X Speed]: [Y Speed]: [Z Speed]: [U Speed]
PX=[value] PY=[value] PZ=[value] PU=[value]	Set position value of axis	OK
SASTAT	Get standalone program status 0 – Stopped 1 – Running 2 – Paused 4 – In Error	[0-4]
SA[LineNumber]	Get standalone line	Single line of compiled code
SA[LineNumber]=[Value]	Set standalone line	OK
SCVX SCVY SCVZ SCVU	Returns the s-curve control	[0,1]
SCVX=[0 or 1] SCVY=[0 or 1] SCVZ=[0 or 1] SCVU=[0 or 1]	Enable or disable s-curve. If disabled, trapezoidal acceleration/ deceleration will be used.	OK
SLAX SLAY SLAZ SLAU	Get StepNLoop maximum attempt value of axis	
SLAX=[value] SLAY=[value] SLAZ=[value] SLAU=[value]	Set StepNLoop maximum attempt value of axis	OK
SLEX SLEY	Get StepNLoop error range value of axis	

SLEZ SLEU		
SLEX=[value] SLEY=[value] SLEZ=[value] SLEU=[value]	Set StepNLoop error range value of axis	OK
SLRX SLRY SLRZ SLRU	Get StepNLoop ratio of axis (ppr / cpr)	[0.001-999.999]
SLRX=[value] SLRY=[value] SLRZ=[value] SLRU=[value]	Set StepNLoop ratio of axis (ppr / cpr)	OK
SLSX SLSY SLSZ SLSU	Get StepNLoop status of axis	[0-12]
SLTX SLTY SLTZ SLTU	Get StepNLoop tolerance of axis	
SLTX=[value] SLTY=[value] SLTZ=[value] SLTU=[value]	Set StepNLoop tolerance of axis	OK
SLX SLY SLZ SLU	Get StepNLoop enable of axis	[0,1]
SLX=[value] SLY=[value] SLZ=[value] SLU=[value]	Set StepNLoop enable of axis	OK
SLOAD	Returns RunOnBoot parameter	[0,1]
SLOAD=[0 or 1]	0 – Do NOT run standalone program on boot up 1 – Run standalone program on boot up	OK
SR=[Value]	Control standalone program: 0 – Stop standalone program 1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program	OK
SPC	Get program counter for standalone program	[0-1784]
SSPDX=[value] SSPDY=[value] SSPDZ=[value] SSPDU=[value]	PMX on-the-fly speed change. In order to use this command on a certain axis, S-curve control must be disabled for the corresponding axis. Use SCV[axis] command to enable and disable s-curve acceleration/ deceleration control.	OK
SSPDMX SSPDMY SSPDMZ SSPDMU	Get on-the-fly speed change mode for each axis	[0-7]
SSPDMX=[value] SSPDMY=[value] SSPDMZ=[value]	Set on-the-fly speed change mode for each axis.	OK

SSPDMU=[value]		
STOP	Performs ramp down to low speed and stop if the motor is moving. (All axes)	OK
STOPX STOPY STOPZ STOPU	Performs ramp down to low speed and stop if the motor is moving. (Individual axis)	OK
STORE	Store parameters to flash	OK
SYNXC SYNYC SYNZC SYNUC	Read sync output configuration for each axis 1 – trigger when encoder equals position 2 – trigger when encoder is greater than position 3 – trigger when encoder is less than position	[1-3]
SYNXC=[value] SYNYC=[value] SYNZC=[value] SYNUC=[value]	Set sync output configuration for each axis 1 – trigger when encoder equals position 2 – trigger when encoder is greater than position 3 – trigger when encoder is less than position	OK
SYNXF SYNYF SYNZF SYNUF	Turn of sync output for each axis	OK
SYNXO SYNYO SYNZO SYNUO	Turn on sync output for each axis	OK
SYNXP SYNYP SYNZP SYNUP	Get trigger position for each axis	28 bit signed number
SYNXP=[value] SYNYP=[value] SYNZP=[value] SYNUP=[value]	Set trigger position for each axis	28 bit signed number
SYNXT SYNYT SYNZT SYNUT	Get pulse width time (ms). Only applicable if sync output configuration is set to 1.	[0-10]
SYNXT=[value] SYNYT=[value] SYNZT=[value] SYNUT=[value]	Set pulse width time (ms). Only applicable if sync output configuration is set to 1.	OK
TR	Get timer register value	[0-1,000,000]
TR=[value]	Set timer register value (ms)	OK
V[0-99]	Get standalone variable value	
V[0-99]=[Value]	Write standalone variable value	OK
VER	Returns controller firmware version	V[#]
X[target X] Y[target Y] Z[target Z] U[target U]	Individual/interpolation move command	OK

13. 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
X2000           ;***Move X axis to position 2000
ABORT           ;***Stop immediately all axes including X axis
```

ACC

Description:

Read: Get acceleration value
Write: Set acceleration value.

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

Syntax:

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

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

IF ACC=[value]
ENDIF
```

Examples:

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

ACC[axis]

Description:

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

Value is in milliseconds.

Syntax:

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

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

IF ACC[axis]=[value]
ENDIF

Examples:

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

ARC

Description:

Motion: Perform arc move using X and Y axis.

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

Syntax:

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

Examples:

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

CIR

Description:

Motion: Perform circle move using X and Y axis.

Specify clockwise or counter-clockwise and the center location.

Syntax:

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

Examples:

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

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

DELAY

Description:

Set a delay (1 ms units)

Syntax:

Delay=[Number] (1 ms units)

Examples:

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

DI

Description:

Read: Gets the digital input value

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

ELSEIF

Description:

Perform ELSEIF condition check as a part of the IF statement

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

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

END

Description:

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

Note: Subroutine definitions should be written AFTER the END statement

Syntax:

END

Examples:

```
X0
X1000
END
```

ENDIF

Description:

Indicates end of IF operation

Syntax:

ENDIF

Examples:

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

ENDSUB

Description:

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

Syntax:

ENDSUB

Examples:

```
GOSUB 1
END

SUB 1
    X0
    X1000
ENDSUB
```

ENDWHILE

Description:

Indicate end of WHILE loop

Syntax:

ENDWHILE

Examples:

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

EO

Description:

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

Performax 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
    X1000
ENDSUB
```

HOME[axis][+ or -]

Description:

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

Syntax:

HOME[Axis][+ or -]

Examples:

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

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

HSPD

Description:

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

Range is from 1 to 6,000,000.

Syntax:

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

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

IF HSPD=[value]
ENDIF
```

Examples:

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

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

HSPD[axis]

Description:

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

Range is from 1 to 6,000,000.

Syntax:

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

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

IF HSPD[axis]=[value]
ENDIF
```

Examples:

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

IF

Description:

Perform IF condition check

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

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

INC

Description:

Command: Changes all move commands to incremental mode.

Syntax:

INC

Examples:

```
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
```

LSPD

Description:

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

Range is from 1 to 6,000,000.

Syntax:

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

Conditional: IF LSPD=[variable]
ENDIF

IF LSPD=[value]
ENDIF

Examples:

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

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

LSPD[axis]

Description:

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

Range is from 1 to 6,000,000.

Syntax:

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

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

IF LSPD[axis]=[value]
ENDIF

Examples:

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

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

MST[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 can not be used for the corresponding axis.

Examples:

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

SL[axis]

Description:

Write: Set individual StepNLoop enable.

Range is from 0 or 1

Syntax:

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

Examples:

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

SLS[axis]

Description:

Command: Get the StepNLoop status of axis

Syntax:

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

Examples:

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

SSPD[axis]

Description:

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

Syntax:

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

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

Examples:

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

SSPDM[axis]

Description:

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

Syntax:

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

Examples:

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

STOP

Description:

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

Syntax:

STOP

Examples:

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

STOP[axis]

Description:

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

Syntax:

STOP[axis]

Examples:

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

SYN[axis]C

Description:

Write: Set sync output configuration for axis

Syntax:

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

Examples:

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

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

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

SYN[axis]F

Description:
Write: Disable sync output for axis

Syntax:
Write: SYN[axis]F

Examples:
See SYN[axis]C

SYN[axis]O

Description:
Write: Enable sync output for axis

Syntax:
Write: SYN[axis]O

Examples:
See SYN[axis]C

SYN[axis]P

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

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

Examples:
See SYN[axis]C

SYN[axis]S

Description:
Read: Get status for sync output of axis

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

Examples:
See SYN[axis]C

SYN[axis]T

Description:

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

Syntax:

Write: SYN[axis]T=[value]

Examples:

See SYN[axis]C

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
        X1000
ENDSUB
```

TR

Description:

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

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

Syntax:

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

Conditional: IF TR=[variable]
ENDIF

Examples:

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

U

Description:

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

Syntax:

U[value]
U[variable]

Examples:

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

V

Description:

Assign to variable.
Performax 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

Description:

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

Syntax:

WAIT[axis]
X[variable]

Examples:

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

WHILE

Description:

Perform WHILE loop

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

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

X

Description:

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

Syntax:

X[value]
X[variable]

Examples:

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

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

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

Y

Description:

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

Syntax:

Y[value]
Y[variable]

Examples:

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

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

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

Z

Description:

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

Syntax:

Z[value]
Z[variable]

Examples:

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

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

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

ZHOME[axis][+ or -]

Description:

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

Syntax:

ZHOME[Axis][+ or -]

Examples:

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

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

ZOME[axis][+ or -]

Description:

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

Syntax:

ZOME[Axis][+ or -]

Examples:

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

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

Contact Information

Arcus Technology, Inc.

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

www.arcus-technology.com