

**Performax 4CX-SA**  
**4-Axis**  
**Stepper Motor Controller**  
**Standalone Version**  
**Manual**



COPYRIGHT © 2005 ARCUS,  
ALL RIGHTS RESERVED

First edition, September 2005

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

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

**Revision History:**

- 1.0 – First revision**
- 1.03 – Updated ASCII commands**
- 1.04 – Added Standalone capability**
- 1.05 – Software update**

**Firmware Compatibility:**

V105

**Software Compatibility:**

V106

## Table of Contents

1. Introduction	5
2. Dimensions	6
3. Pin Descriptions	7
4. Getting Started	11
Typical Setup	11
5. Sample Program	12
6. Motion Control Feature Overview	20
Motion Profile	20
Target Motion	21
Homing	21
Jogging	22
Stopping Motor	22
Motor Position	22
Pulse Speed	22
Motor Status	22
Motor IO Status	22
Digital Outputs	23
Enable Outputs	23
Digital Inputs	23
Limit Inputs	23
Standalone Program Specification	24
Device Name	24
Storing to Flash	24
7. Communication	25
USB Communication Issues	26
8. ASCII Language Specification	27
9. Standalone Language Specification	30
;	30
ABORT	30
ABORT[axis]	30
ABS	31
ACC	31
ACC[axis]	32
DELAY	32
DI	33
DI[1-4]	33
DO	34
DO[1-4]	34
ELSE	35
ELSEIF	35
END	36
ENDIF	36
ENDSUB	36
ENDWHILE	37
EO	37

---

EO[1-4] _____	38
GOSUB _____	38
HOME[axis][+ or -] _____	39
HSPD _____	39
HSPD[axis] _____	40
IF _____	40
INC _____	41
JOG[axis] _____	41
LSPD _____	42
LSPD[axis] _____	42
MST[axis] _____	43
P[axis] _____	43
PS[axis] _____	44
STOP _____	44
STOP[axis] _____	45
STORE _____	45
SUB _____	45
U _____	46
V _____	46
WAIT _____	47
WHILE _____	48
X _____	49
Y _____	49
Z _____	50

# 1. Introduction

Performax 4CX-SA is a 4 axis standalone programmable motion controller with USB 2.0 communication.

Multiple Performax controllers can be connected via USB to make multi-axis and IO control system.

## **Performax 4CX-SA Features**

- 4 axis stepper pulse/direction open collector outputs
- Maximum output Frequency of 400K PPS
- Plus Limit, Minus Limit, and Home opto-isolated inputs per axis
- 4 general-purpose opto-isolated inputs.
- 4 general-purpose opto-isolated outputs. 100mA sink capable.
- 4 general-purpose open collector outputs. 40mA sink capable.
- USB 2.0 communication
- Standalone programmable.

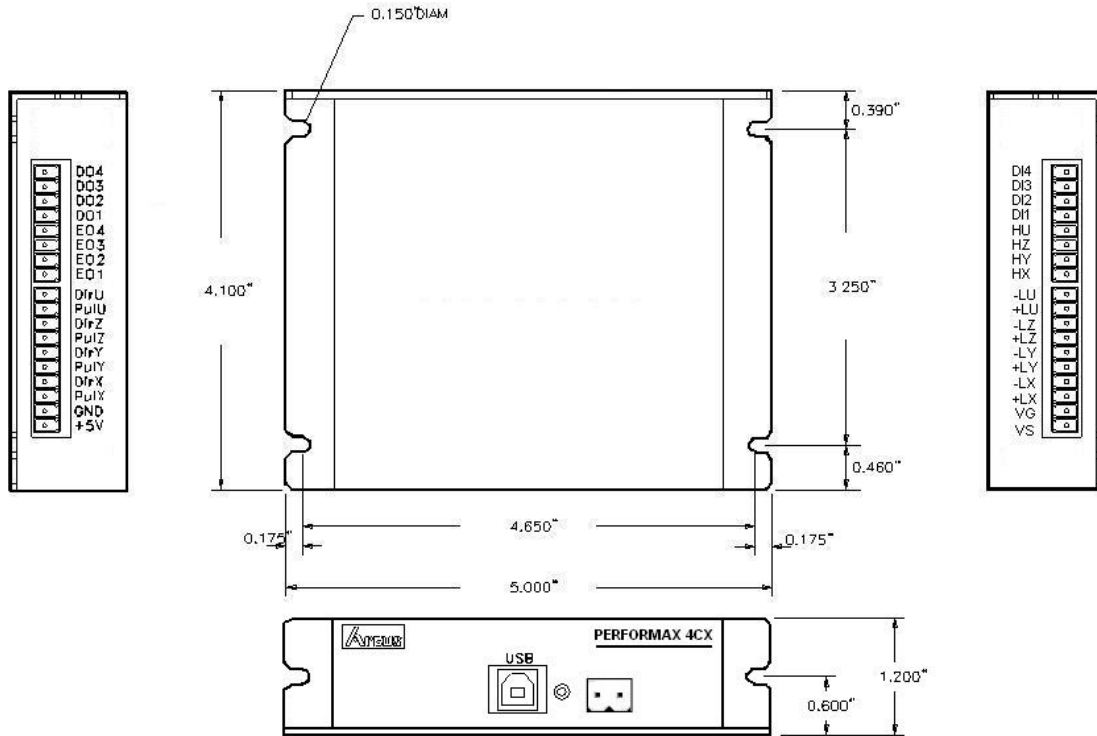
Performax 4CX comes with a Windows DLL for easy interface with the program from common programming language such as VB, VC++, and LabVIEW. Sample program in VB is provided.

## **Contacting Support**

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

Or, contact your local distributor for technical support.

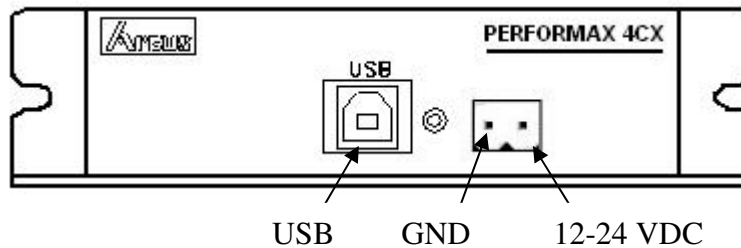
## 2. Dimensions



### 3. Pin Descriptions

#### USB and Input Power

In order for the PMX-4CX-SA to operate, it must be supplied with +12VDC to +24VDC. Power pins as well as USB port shown below.



Pin Name	Description	Signal Type
DO4	Digital Output 4	Opto-isolated Output
DO3	Digital Output 3	Opto-isolated Output
DO2	Digital Output 2	Opto-isolated Output
DO1	Digital Output 1	Opto-isolated Output
EO4	Enable Output 4	Open Collector Output
EO3	Enable Output 3	Open Collector Output
EO2	Enable Output 2	Open Collector Output
EO1	Enable Output 1	Open Collector Output
DirU	U Axis Direction Output	Open Collector Output
PulU	U Axis Pulse Output	Open Collector Output
DirZ	Z Axis Direction Output	Open Collector Output
PulZ	Z Axis Pulse Output	Open Collector Output
DirY	Y Axis Direction Output	Open Collector Output
PulY	Y Axis Pulse Output	Open Collector Output
DirX	X Axis Direction Output	Open Collector Output
PulX	X Axis Pulse Output	Open Collector Output
GND	Ground	Ground signal with respect to 5V supply
+5V	+5 Volt Supply Output	300mA re-settable fused output 5V supply

#### NOTES:

- Opto-isolated output sinks to opto-ground, not the 5V supply ground
- Open-collector outputs sinks to 5V supply ground
- Opto-isolated output can sink maximum of 100mA
- Open-collector output can sink maximum of 40mA

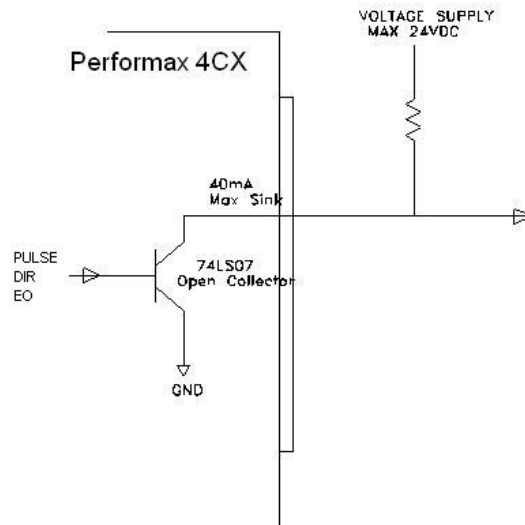
<b>Pin Name</b>	<b>Description</b>	<b>Signal Type</b>
DI4	Digital Input 4	Opto-isolated Input
DI3	Digital Input 3	Opto-isolated Input
DI2	Digital Input 2	Opto-isolated Input
DI1	Digital Input 1	Opto-isolated Input
HU	U Axis Home Input	Opto-isolated Input
HZ	Z Axis Home Input	Opto-isolated Input
HY	Y Axis Home Input	Opto-isolated Input
HX	X Axis Home Input	Opto-isolated Input
-LU	U Axis Minus Limit Input	Opto-isolated Input
+LU	U Axis Plus Limit Input	Opto-isolated Input
-LZ	Z Axis Minus Limit Input	Opto-isolated Input
+LZ	Z Axis Plus Limit Input	Opto-isolated Input
-LY	Y Axis Minus Limit Input	Opto-isolated Input
+LY	Y Axis Plus Limit Input	Opto-isolated Input
-LX	X Axis Minus Limit Input	Opto-isolated Input
+LX	X Axis Plus Limit Input	Opto-isolated Input
VG	Opto Ground	Opto-supply Ground signal with respect to Opto-supply
VS	Opto-supply	Opto-supply 12-24VDC

**NOTES:**

- All inputs are opto-isolated.
- In order for the inputs to work, supply the VS with 12 to 24 VDC.
- To trigger the input, sink the input signal to VG opto-ground.

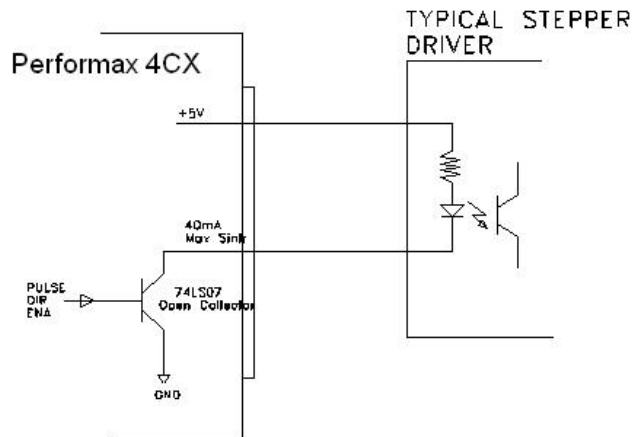
Opto-ground and 5V ground are not connected.

## Open-Collector Output Circuit

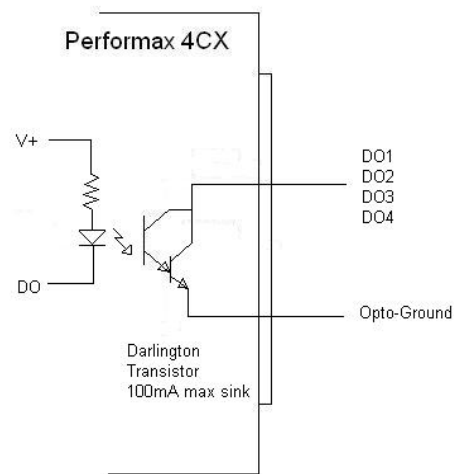


Performax 4CX open-collector outputs use 74LS07 IC. Maximum sink current is 40mA.

Typical open collector interface with stepper driver:

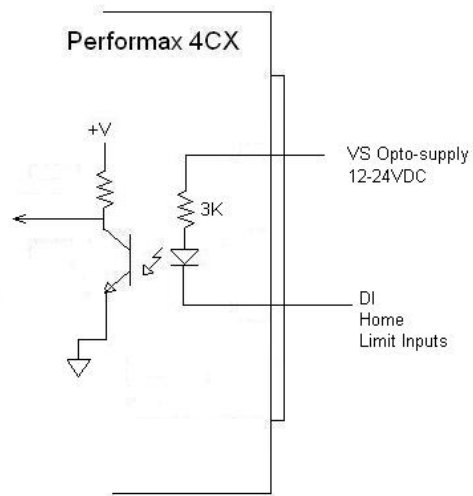


## Opto-isolated Digital Output Circuit



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

## Opto-isolated Digital Input Circuit



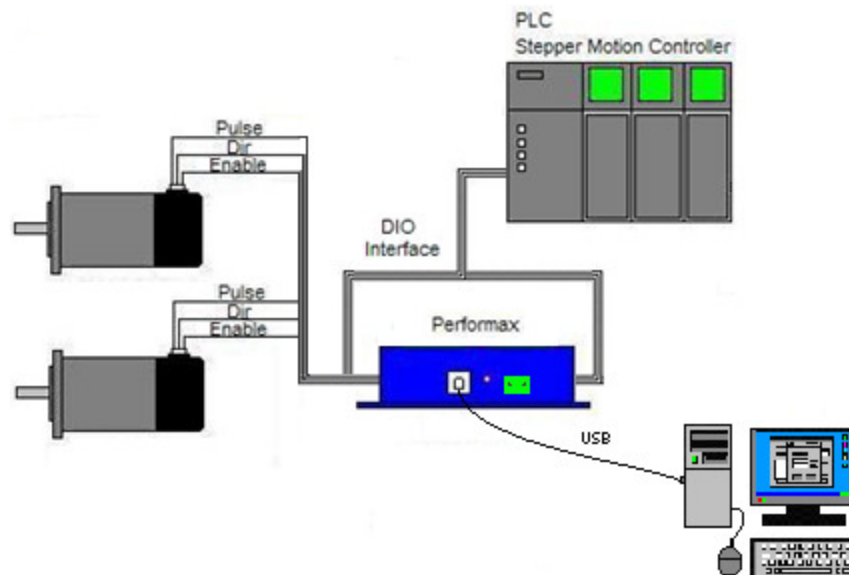
Digital inputs are opto-isolated inputs. To trigger the input, pull the signal to ground of Opto-supply.

## 4. Getting Started

Typical Software Setup Steps for First Time Users:

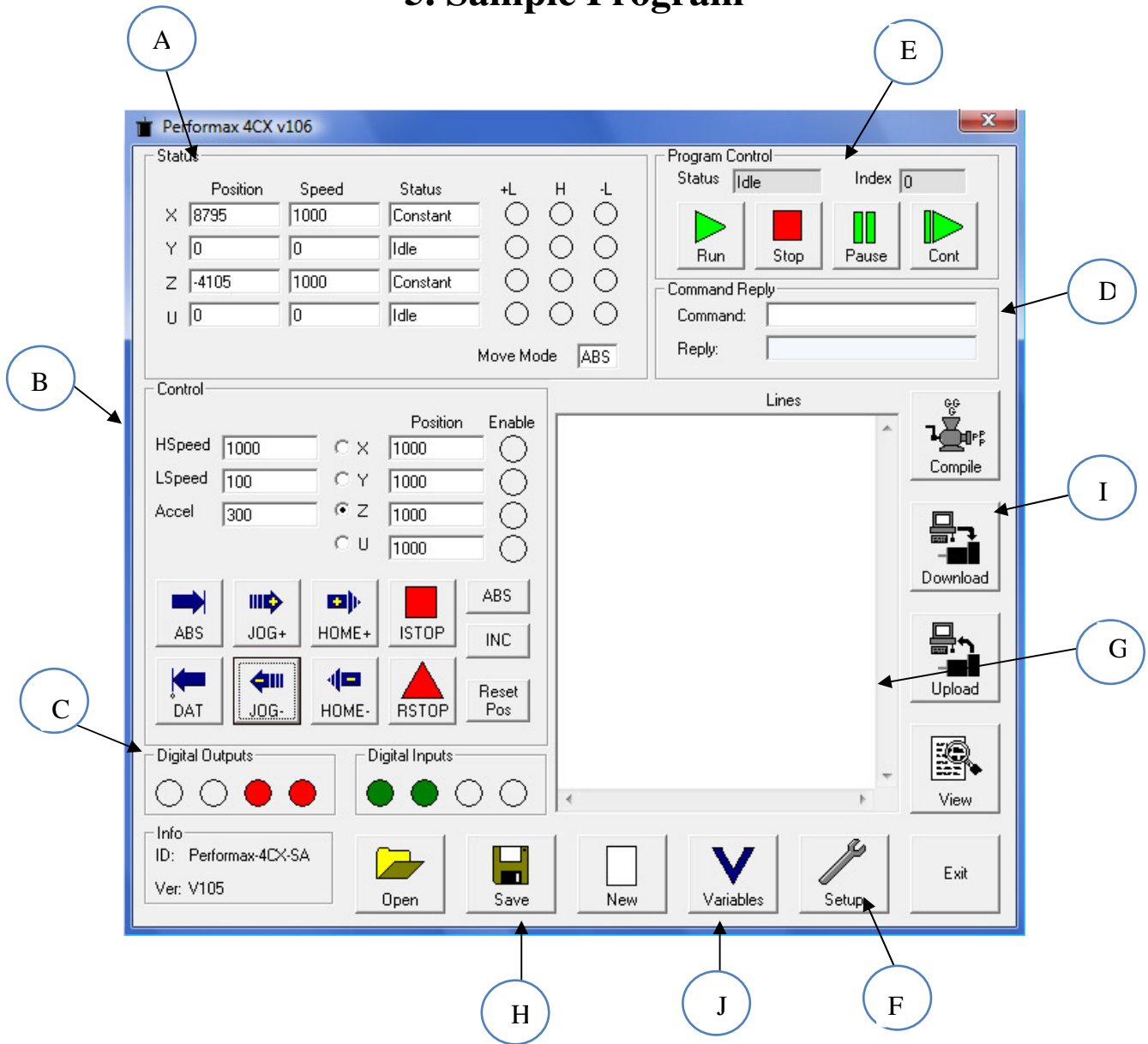
- 1) **Run Performax USB Driver Setup program.** Both the setup program and manual for the Performax USB Driver setup can be downloaded from the web site.
- 2) **Connect the Performax USB device to the Windows PC.** Go through typical Windows USB device setup. See the Performax USB Driver Setup manual for details.
- 3) **Use test program to check the USB communication and controller features.** Sample program for each of the Performax devices can be downloaded from the web site.
- 4) **Write your custom application program.** Sample application programs written in VB6, C++, or LabVIEW can be used as a starting point and these programs are available for download from the website.

### Typical Setup

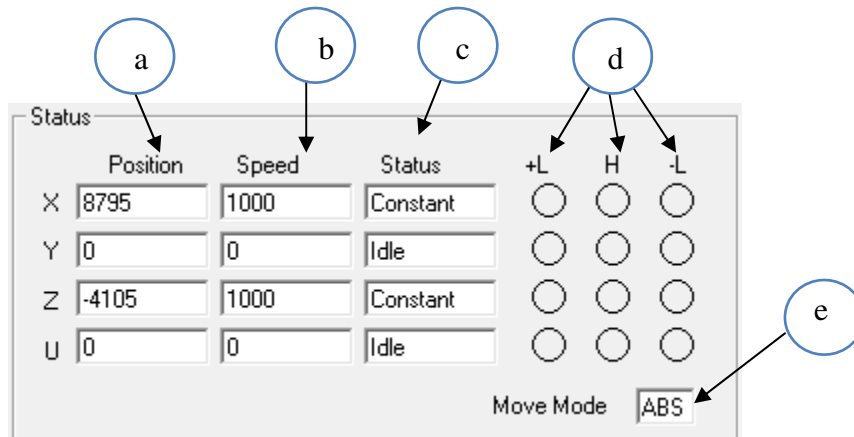


***Important Note: In order to communicate with PMX-4CX-SA through USB, proper driver must be installed first. Before connecting the PMX-4CX-SA device or running any program, please go to the Arcus web site and download the USB driver installation instruction and run the USB Driver Installation Program.***

## 5. Sample Program

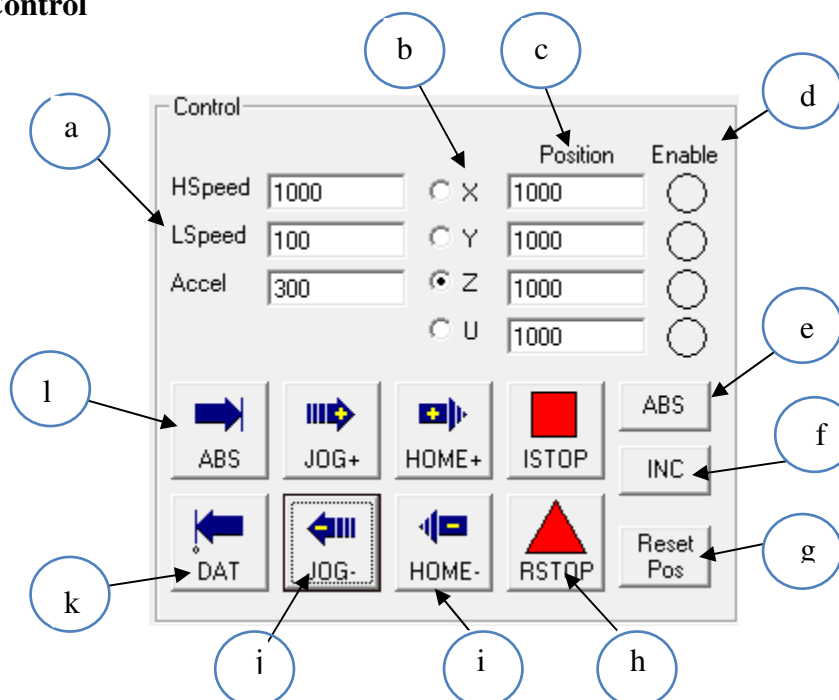


## A. Status



- a. Current pulse position (X/Y/Z/U axis).
- b. Current pulse speed (X/Y/Z/U axis).
- c. Motor status (X/Y/Z/U axis).
  - i. Idle – Motor is not moving.
  - ii. Accel – Motor is accelerating.
  - iii. Const – Motor is moving at constant speed.
  - iv. Decel – Motor is decelerating.
- d. –Limit, +Limit, Home input status (X/Y/Z/U status).
- e. Move mode.
  - a. ABS: On target movement, motor will move to target position.
  - b. INC: On target movement motor will increment by the target position.

## B. Control



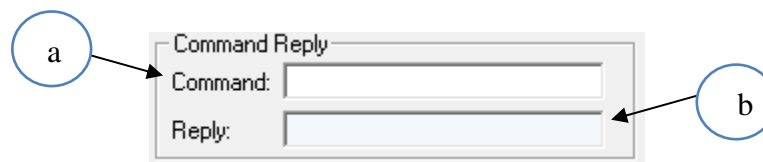
- a. Global high speed, low speed, and acceleration entered here. To give each axis individual speed parameters, enter **HSPD[axis]**, **LSPD[axis]**, and **ACC[axis]** into the command line in the “Terminal” section.
- b. Select the axis to control. The axis must also be enabled for the motor to respond.
- c. Target position used for absolute and incremental movement.
- d. Enable the driver for the indicated axis.
- e. ABS mode. For target movement, the motor will move to the absolute target position.
- f. INC mode. For target movement, the motor will increment its position to the target position.
- g. Reset the indicated motor to position zero.
- h. **RSTOP/ISTOP:**
  - a. **RSTOP** stops the indicated motor with the deceleration.
  - b. **ISTOP** stops the indicated immediately, without deceleration.
- i. **HOME+/HOME-** Homes the motor in the positive or negative direction.
- j. **JOG+/JOG-** Jogs the indicated motor in the positive or negative direction.
- k. **DAT** – Will move the motor back to position zero.
- l. **ABS** – Will move the motor using the target position.

### C. Outputs and Inputs



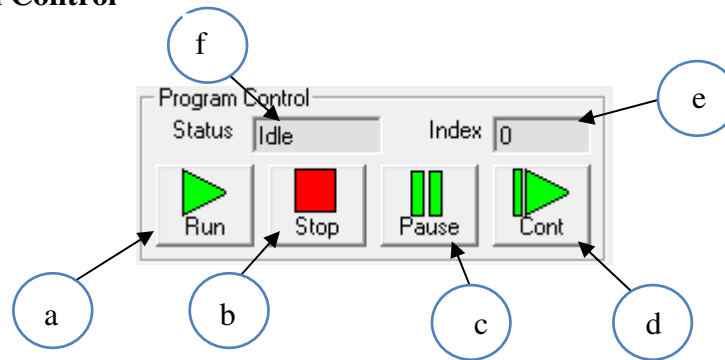
- a. Digital output status for DO1-DO4.
- b. Digital input status for DI1-DI4.

### D. Command Reply



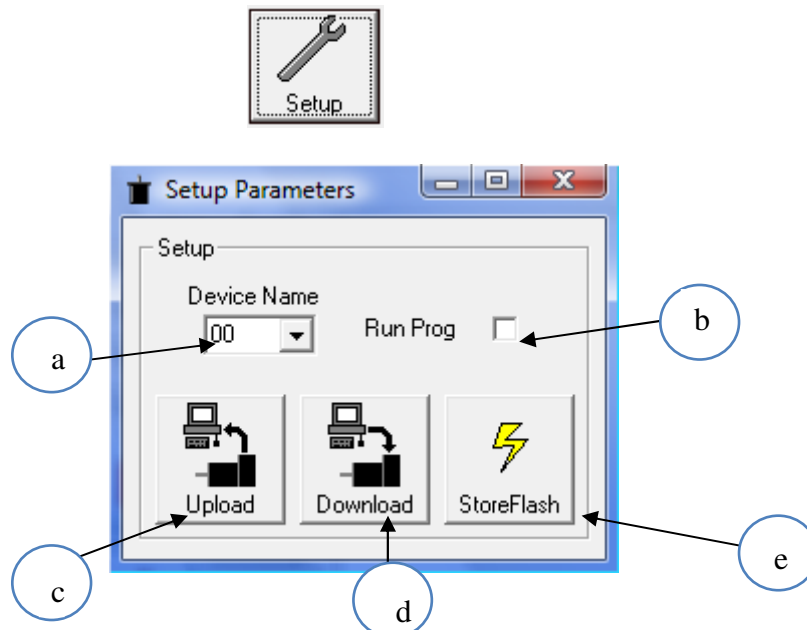
- a. Send command to the PMX-4CX-SA through this terminal.
- b. Replies from the PMX-4CX-SA will be shown here.

## E. Program Control



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

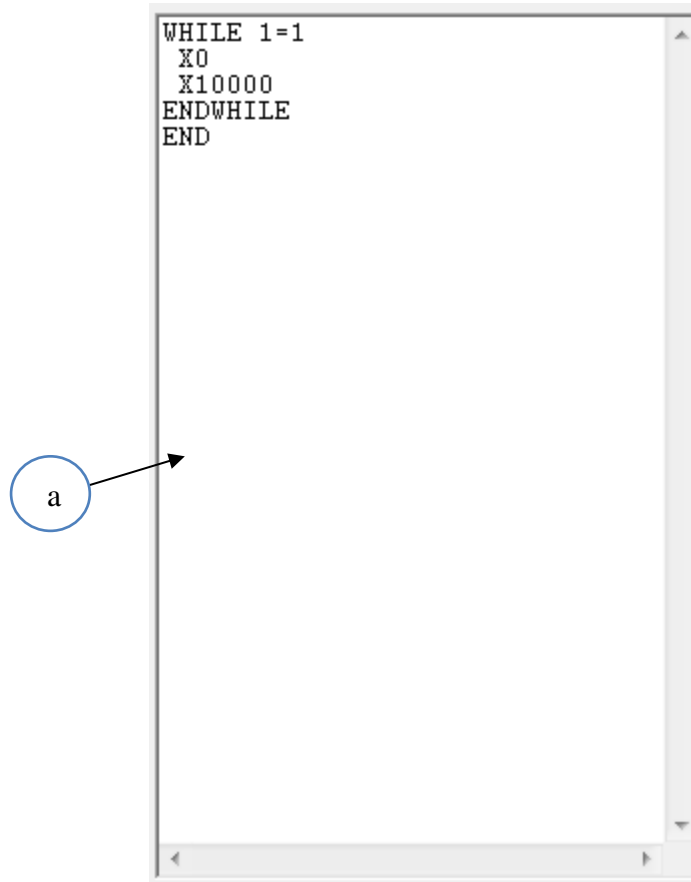
## F. Setup



- a. Device Number – Set the name of the device. Must be in the range of 4CX00-4CX99.
- b. Check this box to have the standalone program run on boot up.
- c. Upload the device name and run on boot parameter from memory.

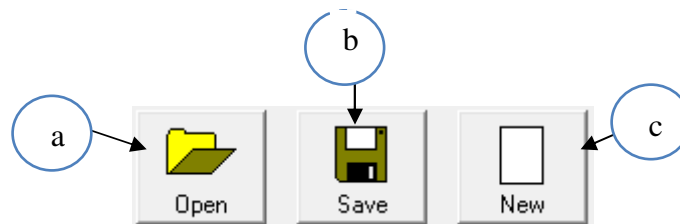
- d. Download device name and run on boot parameter to memory. Note that a store must be executed if these values are to be preserved after power cycle.
- e. Store device name and run on boot parameter to flash so that it is remembered after power cycle.

### G. Text Programming Box

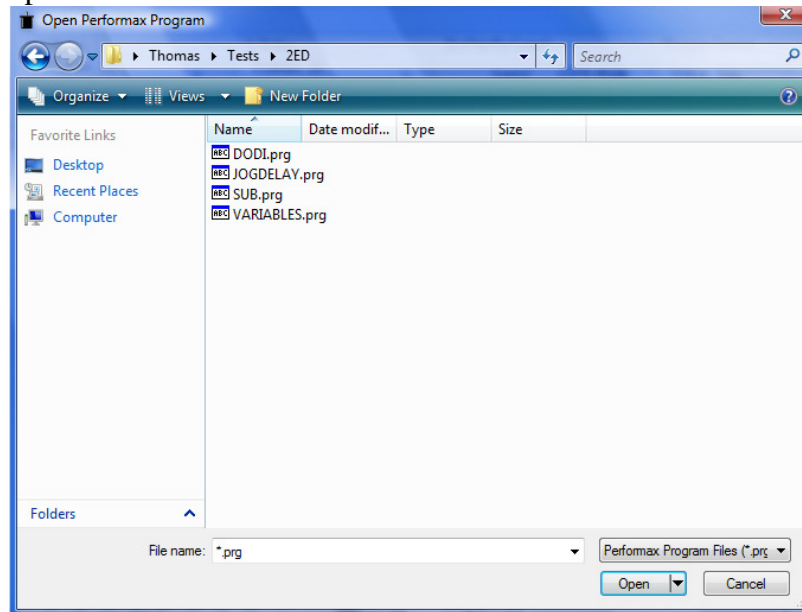


- a. Text box for standalone program. See details on programming language.

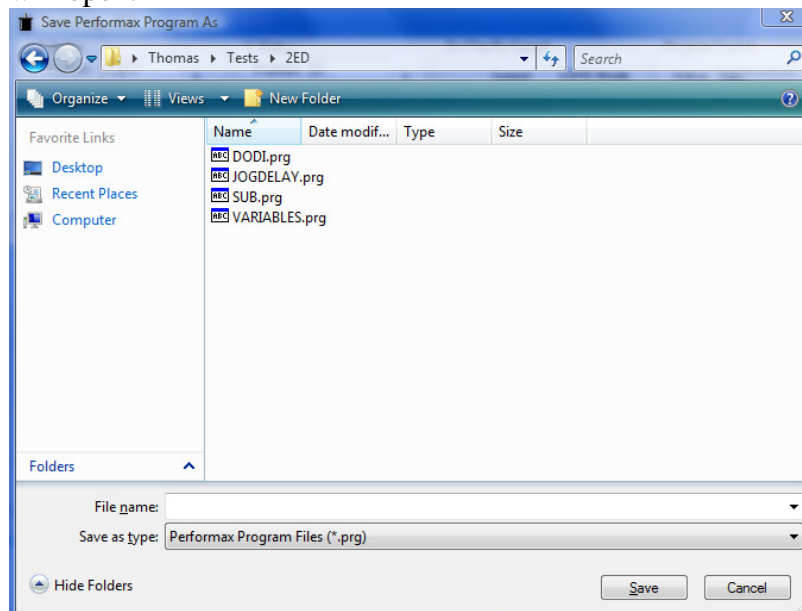
### H. Program File Control



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

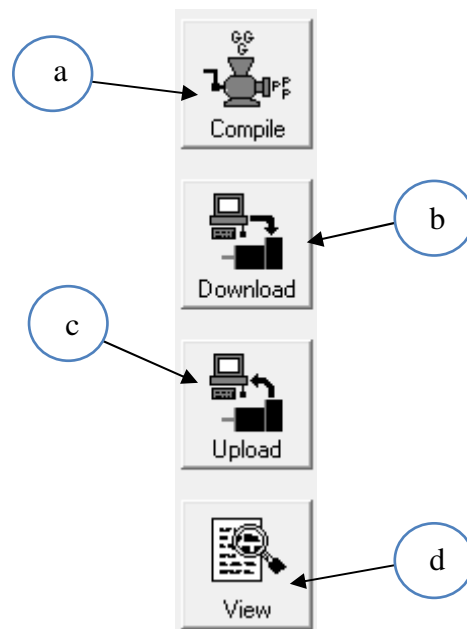


- b. Save – Standalone program in the text programming box is saved to a file. When this button is pressed, typical Windows file save dialog box will open:



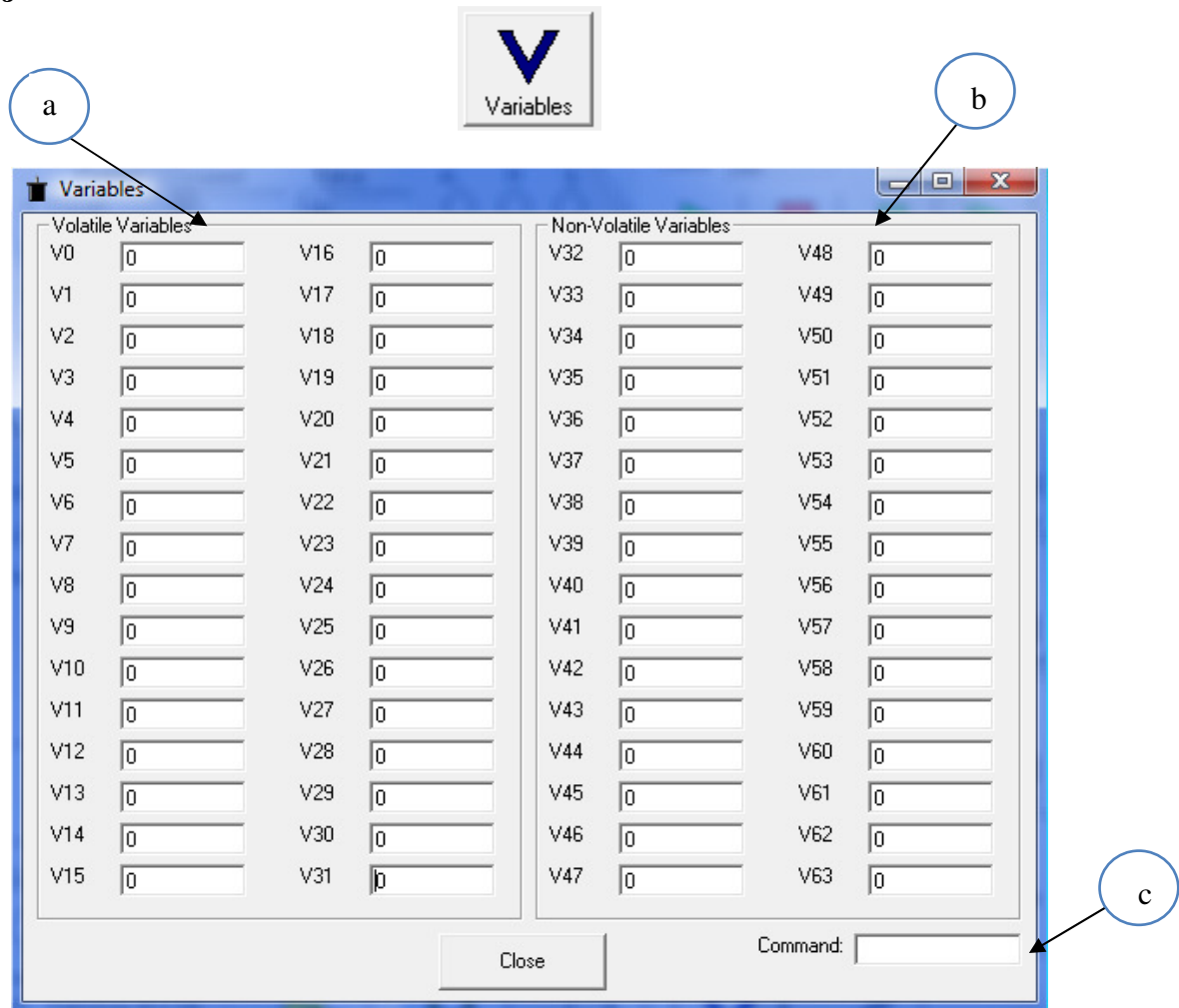
- c. New – When this button is pressed, the text programming box is cleared.

## I. Compiler



- a. Compile code in the text programming box into assembly level code that the PMX-4CX-SA can understand.
- b. Download the compiled code into memory. Note that the text based code must first be compiled before download.
- c. Upload the standalone code that is currently on your PMX-4CX-SA. This automatically translates assembly level language into readable text-based code.
- d. View compiled code for easy cutting and pasting.

## J. Variables



- a. Volatile variables are not saved to flash and are reset to zero after a power cycle.
- b. Non-volatile variables can be saved to flash. Perform the **STORE** command to save these variables to flash.
- c. Send commands to the PMX-4CX-SA through this terminal.

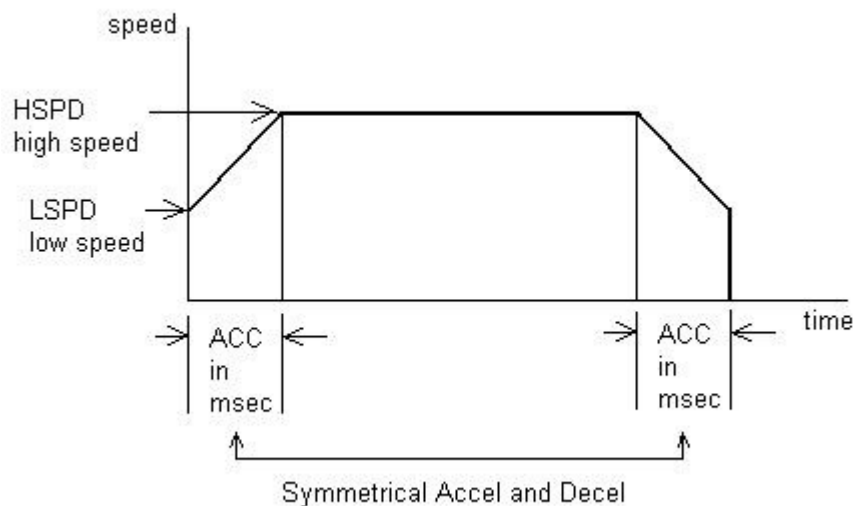
## 6. Motion Control Feature Overview

*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

Performax 4CX can generate up to 300K pulses per second pulse rate.

By default, the Performax 4CX uses trapezoidal velocity profile.



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

High speed and low speed are in pps(pulses/second). Use the **HSPD/HSPDX/HSPDY/HSPDZ/HSPDU** and **LSPD/LSPDX/LSPDY/LSPDZ/LSPDU** commands to set and get high speed and low speed settings. For example, to set the high speed of the Z axis to 15000 use **HSPDZ=15000** command. To read the current global high speed, send **HSPD** command and the reply will contain the current high speed.

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

#### Note on the acceleration setting:

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

HSPD[pps]	Minimum ACC [ms]	Accel Delta [pps]
1-8K	1	2,900
8K-16K	1	6,400
16K-40K	1	14,000
40K-80K	1	25,000
80K-160K	1	58,000
160K-400K	1	130,000

Examples:

- a) If **HSPD** = 300,000 pps, **LSPD** = 250,000 pps:
  - a. Get Speed delta:  $((300,000 - 250,000) / 130,00) = 0.38$
  - b. Max acceleration allowable:  $0.38 \times 1,000 \text{ ms} = \mathbf{380 \text{ ms}}$  (0.38 sec)
  
- b) If **HSPD** = 35,000 pps, **LSPD** = 2,000 pps:
  - a. Get Speed delta:  $((35,000 - 2,000) / 14,000) = 2.36$
  - b. Max acceleration allowable:  $2.36 \times 1000 \text{ ms} = \mathbf{2,360 \text{ ms}}$  (2.36 sec)

**Note on the speed settings:**

The minimum **LSPD** value depends on the **HSPD** settings. See the chart below for details:

HSPD[pps]	Minimum LSPD[pps]
1-8K	1
8K-16K	2
16K-40K	5
40K-80K	10
80K-160K	20
160K-300K	50

*Note: If an incorrect low speed value is used, it will be replaced by the lowest possible low speed value.*

**Target Motion**

Use the **X/Y/Z/U** command to move the axis to the target position.

Use the **ABS/INC** command to set the move commands to absolute or incremental mode.

For example, to move the motor incrementally issue the **INC** command and issue move command **X1000**. This will move the motor by 1000 from the current position. To move the motor to absolute position, issue the **ABS** command and issue target position **X500**. This will move the motor to the absolute position of 500.

**Homing**

When a homing command is sent, the motor ramps up from low speed to high speed and as soon as the home input is triggered, the motor stops and the position counter is reset to

zero. To trigger the home input switch, supply the opto-supply voltage with 12 to 24VDC and pull down (or connect) the home input signal to the ground signal. Use the **HOMEX+** and **HOMEX-** commands for homing the X axis. For Y, Z, and U axis use appropriate axis letter.

Limit switches can be used for homing by jogging the motor to the limit switch (using the **JOG[axis]+** or **JOG[axis]-** commands), waiting for the stopping of the motor, and after checking for the limit switch input status, resetting the position counter to zero using **PX/PY/PZ/PU** command.

### **Jogging**

Jogging is available for continuous speed operation. Use the **JOGX+** and **JOGX-** commands to jog in the positive or negative direction for the X axis. For other axes, use the appropriate axis letter.

### **Stopping Motor**

When motor is moving, jogging, or homing, **ABORTX/ABORTY/ABORTZ/ABORTU** command will immediately stop the indicated motor. **ABORT** command without the axis letter will stop all the axes immediately. **STOPX/STOPY/STOPZ/STOPU** command will decelerate the axis to low speed and then stop. **STOP** command without the axis letter will stop all the axes in motion with deceleration.

*Note: We recommend that a decelerated stop be used whenever possible to reduce the impact to the motor and the system.*

### **Motor Position**

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

### **Pulse Speed**

Current actual pulse rate or speed can be read using the **SX/SY/SZ/SU** command.

### **Motor Status**

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

Bit	Description
0	Generating pulse
1	Motor in acceleration
2	Motor in deceleration

### **Motor IO Status**

Motor IO status can be read anytime using the **MIOX/MIOY/MIOZ/MIOU** command. Following are bit representation of motor status:

Bit	Description
0	Minus Limit Input Status
1	Plus Limit Input Status
2	Home Input Status

### ***Digital Outputs***

Digital output command **DO** can be used to read and set the 4 bit digital outputs. All 4 bits of the digital output are opto-isolated outputs. To set and read individual digital outputs use **DO1** to **DO4** command.

For example to read the bit 3, use **DO3** command to get the status of the 3<sup>rd</sup> bit. To set the 4<sup>th</sup> bit on, use **DO4=1** command and to reset **DO4=0** command.

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

For example to set all the digital outputs on, use **DO=15** command. Number 15 corresponds to F in hex and 1111 in binary.

### ***Enable Outputs***

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

### ***Digital Inputs***

The digital input command **DI** can be used to read the 4 bits of digital input. The 4 bits of the digital inputs are opto-isolated inputs. To read individual digital inputs use the **DI1** to **DI4** command.

For example, reply to the **DI** command of 15 will correspond to F in hex and 1111 in binary.

### ***Limit Inputs***

If the positive limit switch is triggered while moving in positive direction, the motor will immediately stop. The same applies for the negative limit when the motor is moving in the negative direction.

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

To read the limit switch input status, use the **MIOX/MIOY/MIOZ/MIOU** command.

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

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

Memory size: 1785 assembly lines ~ 10.5 KB.

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

#### **Calling subroutines over communication:**

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

#### ***Device Name***

Multiple Performax 4CX-SA devices can be connected to the USB port of the Windows PC, only requiring that each Performax 4CX-SA device needs to have a unique device ID number. When configuring the device ID, make sure to connect only one device at a time and use the **DN** command. For example, to set the Performax 4CX-SA device, use **DN=4CX03** command to set the device number to 4CX03. Once device number is set, the controller needs to have the power cycled for the new ID to take effect. The device number is stored in the flash memory and is remembered even after the power cycle.

#### ***Storing to Flash***

The following items are stored to flash:

- Device Name
- Automatic program run on boot
- Second half of general purpose variables (V32-V63)

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

## 7. Communication

Performax USB communication is USB 2.0 compliant.

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

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

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

### USB Communication API Functions

For USB communication, the following DLL API functions are provided:

**BOOL fnPerformaxComGetNumDevices**(OUT LPDWORD lpNumDevices);

- This function is used to get total number of Performax and Performax USB modules connected to the PC.

**BOOL fnPerformaxComGetProductString**(IN DWORD dwNumDevices,  
OUT LPVOID lpDeviceString,  
IN DWORD dwOptions);

- This function is used to get the Performax product string. This function is used to find out Performax USB module product string and its associated index number. Index number starts from 0.

**BOOL fnPerformaxComOpen**(IN DWORD dwDeviceNum,  
OUT HANDLE\* pHandle);

- This function is used to open communication with the Performax USB module and to get the communication handle. dwDeviceNum starts from 0.

**BOOL fnPerformaxComClose**(IN HANDLE pHandle);

- This function is used to close communication with the Performax USB module.

**BOOL fnPerformaxComSetTimeouts**(IN DWORD dwReadTimeout,  
DWORD dwWriteTimeout);

- This function is used to set the communication read and write timeouts. Values are in milliseconds. This must be set for the communication to work. Typical value of 1000 msec is recommended.

**BOOL fnPerformaxComSendRecv**(IN HANDLE pHandle,  
 IN LPVOID wBuffer,  
 IN DWORD dwNumBytesToWrite,  
 IN DWORD dwNumBytesToRead,  
 OUT LPVOID rBuffer);

This function is used to send a command and receive a reply. The number of bytes to read and write must be 64 characters.

### **USB Communication Issues**

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

- 1) **Multi-threading:** Be sure that your application does not employ multi-thread processing. See “important note” in the beginning of this section.
- 2) **Buffer Flushing:** If USB communication begins from an unstable state (i.e. your application has closed unexpectedly, it is recommended to first flush the USB buffers of the PC and the USB device. See the following function prototype below:

**BOOL fnPerformaxComFlush**(IN HANDLE pHandle)

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

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



## 8. ASCII Language Specification

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

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

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

Command	Description	Return	Min	Max
ABORT	Immediately stops all the axes in motion.	OK		
ABORTX ABORTY ABORTZ ABORTU	Immediately stops the motor if in motion. For decelerate stop, use STOP command.	OK		
ABS	Changes the move mode to absolute mode. All target move commands X/Y/Z/U command will be absolute move after ABS command is issued.	OK		
ACC	Returns the current global acceleration.	Acceleration value in milliseconds	1	1000
ACCX ACCY ACCZ ACCU	Returns current acceleration value in milliseconds for the indicated motor.	Acceleration value in milliseconds	1	1000
ACC=[Value]	Sets the global acceleration value in milliseconds.	OK	1	1000
ACCX=[Value] ACCY=[Value] ACCZ=[Value] ACCU=[Value]	Sets acceleration value in milliseconds for the indicated motor. Example: ACC=300	OK	1	1000
DI	Returns 4 bits of general purpose digital input status	[0-15]	0	15
DI[1-4]	Returns 4 bits of general purpose digital input status		0	15
DN	Returns 4 character device ID. First three characters are “4CX”. Next two characters are any character from ‘00’ to ‘99’.	5 digit character	4CX00	4CX99
DN=[Value]	Sets 5 character device ID. First three characters must be “4CX” and next two characters must be from ‘00’ to ‘99’. If device ID is changed, power cycle must be done for the new device	OK	4CX00	4CX99

	number to be recognized by the PC. Once the device number is set, it is stored in the non-volatile flash. Example: DN=4CX01			
DO	Returns 4 bits of general purpose digital output status	[0-15]	0	15
DO=[4 bit Value]	Sets the digital output bits.	OK	0	15
DO[1-4]	Returns the digital output bit status	[0-1]	0	1
DO[1-4] = [Value]	Sets the digital output bit	OK	0	1
EO	Returns 4 bits of enable output value.	[0-15]	0	15
EO=[Value]	Sets 4 bits of enable outputs.	OK	0	15
EO[1-4]	Returns bit of enable output value.	[0-1]	0	1
EO[1-4]=[Value]	Set bit of enable outputs.	OK	0	1
GS[SubNumber]	Call a defined subroutine	OK		
HOMEX+ HOMEY+ HOMEZ+ HOMEU+	Homes the motor in the positive direction	OK		
HOMEX- HOMEY- HOMEZ- HOMEU-	Homes the motor in the negative direction	OK		
HSPD	Returns the global high speed settings for the indicated motor.	Value in pps	1	300K
HSPDX HSPDY HSPDZ HSPDU	Returns high speed setting for the indicated motor.	Value in pps	1	300K
HSPD=[Value]	Sets the global high speed value.	OK	1	300K
HSPDX=[Value] HSPDY=[Value] HSPDZ=[Value] HSPDU=[Value]	Sets high speed value for the indicated motor.	OK	1	300K
ID	Return product ID.	Performax-4CX-SA		
INC	Changes the move mode to incremental mode. All target move commands X/Y/Z/U command will be incremental move after INC command is issued.	OK		
JOGX+ JOGY+ JOGZ+ JOGU+	Jogs the motor in positive direction	OK		
JOGX- JOGY-	Jogs the motor in negative direction	OK		

JOGZ- JOGU-				
LSPD	Returns the global low speed setting.	Value in pps	1	300K
LSPDX LSPDY LSPDZ LSPDU	Returns Low Speed setting for the indicated motor.	Value in pps	1	300K
LSPD=[Value]	Sets the global low speed value.	OK	1	300K
LSPDX=[Value] LSPDY=[Value] LSPDZ=[Value] LSPDU=[Value]	Sets Low Speed value for the indicated motor.	OK	1	300K
MIO	Returns axis IO status	Bit 0 – minus limit Bit 1 – plus limit Bit 2 – home		
MST	Returns motor status	Bit 0 – pulse generating Bit 1 – accelerating Bit 2 – decelerating		
PX PY PZ PU	Returns current position value for the indicated motor.	Position value in 32 bit		
PX=[Value] PY=[Value] PZ=[Value] PU=[Value]	Sets the current position value for the indicated motor.	OK		
SX SY SZ SU	Returns current pulse speed		1	300K
STOP	Will use a decelerated stop to stop all axes in motion.	OK		
STOPX STOPY STOPZ STOPU	Decelerates to stop the motor if in motion. For immediate stop, use ABORT command.	OK		
STORE	Store device setting and variables (V32-V63) to flash.	OK		
VER	Returns current firmware version number	Vxxx		
V[0-63]	Returns general purpose variable register.	Variable value in 32 bits	-4.29G	4.29G
V[0-63]=[value]	Sets value to general purpose register	OK	-4.29G	4.29G
X[value] Y[value] Z[value] U[value]	Moves the motor to absolute/incremental position value using the HSPD, LSPD, and ACC values.	OK		

## 9. Standalone Language Specification

Version 1.21

;

Description:

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

Syntax:

; [Comment Text]

Examples:

```

;***This is a comment
JOGX+           ;***Jogs X axis to positive direction
DELAY=1000      ;***Wait 1 second
  
```

### **ABORT**

Description:

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

Syntax:

ABORT

Examples:

```

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

### **ABORT[axis]**

Description:

**Motion:** Immediately stops individual axis without deceleration.

Syntax:

ABORT[axis]

Examples:

```

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

## ABS

Description:

**Motion:** Changes all move commands to absolute mode.

Syntax:

ABS

Examples:

```

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

## ACC

Description:

**Read:** Get acceleration value

**Write:** Set acceleration value.

Value is in milliseconds.

Range is from 1 to 10,000.

Syntax:

**Read:** [variable] = ACC

**Write:** ACC = [value]

ACC = [variable]

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

IF ACC=[value]  
ENDIF

Examples:

```

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

## **ACC[axis]**

Description:

**Read:** Get individual acceleration value

**Write:** Set individual acceleration value.

Value is in milliseconds.

Range is from 1 to 10,000.

Syntax:

**Read:** [variable] = ACC[axis]

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

ACC[axis] = [variable]

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

IF ACC[axis]=[value]  
ENDIF

Examples:

ACCX=300 ;\*\*\*Sets the X acceleration to 300 milliseconds

V3=500 ;\*\*\*Sets the variable 3 to 500

ACCX=V3 ;\*\*\*Sets the X acceleration to variable 3 value of 500

## **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 seconds

ABORT ;\*\*\*Stop with deceleration all axes including X axis

PX=0 ;\*\*\*Sets the current X position to 0

PY=0 ;\*\*\*Sets the current Y position to 0

PZ=0 ;\*\*\*Sets the current Z position to 0

PU=0 ;\*\*\*Sets the current U position to 0

## **DI**

Description:

**Read:** Gets the digital input value

Performax 4CX has 4 digital inputs

Syntax:

**Read:** [variable] = DI

**Conditional:** IF DI=[variable]  
ENDIF

IF DI=[value]  
ENDIF

Examples:

```
IF DI=15
    DO=1          ;***If all digital inputs are triggered, set DO=1
ENDIF
```

## **DI[1-4]**

Description:

**Read:** Gets the digital input value

Performax 4CX has 4 digital inputs

Syntax:

**Read:** [variable] = DI[1-4]

**Conditional:** IF DI[1-4]=[variable]  
ENDIF

IF DI[1-4]=[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 4CX has 4 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[1-4]**

### Description:

**Read:** Gets the individual digital output value

**Write:** Sets the individual digital output value

Performax 4CX has 4 digital outputs

### Syntax:

**Read:** [variable] = DO[1-4]

**Write:** DO[1-4] = [0 or 1]

DO[1-4] = [variable]

**Conditional:** IF DO[1-4]=[variable]

ENDIF

IF DO[1-4]=[0 or 1]

ENDIF

### Examples:

DO2=1 ;\*\*\*Turn DO2 on

DO4=1 ;\*\*\*Turn DO4 on

## **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
- 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
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 previous subroutine.

### Syntax:

ENDSUB

### Examples:

GOSUB 1

END

SUB 1

X0

ENDSUB

## **ENDWHILE**

Description:

Indicate end of WHILE loop

Syntax:

ENDWHILE

Examples:

```

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

## **EO**

Description:

**Read:** Gets the enable output value

**Write:** Sets the enable output value

Performax 4CX 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 4CX has 4 enable outputs.

Syntax:

**Read:** [variable] = EO[1-4]

**Write:** EO[1-4] = [0 or 1]

EO[1-4] = [variable]

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

IF EO=[value]  
ENDIF

Examples:

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

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

## **GOSUB**

Description:

Perform go to subroutine operation

Subroutine range is from 1 to 32.

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

Syntax:

GOSUB [subroutine number]

[Subroutine Number] range is 1 to 32

Examples:

```
GOSUB 1
END
```

```
SUB 1
    X0
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 300,000.

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 300,000.

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 incremental 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 300,000.

### Syntax:

**Read:** [variable]=LSPD

**Write:** LSPD=[long value]

LSPD=[variable]

**Conditional:** IF LSPD=[variable]

ENDIF

IF LSPD=[value]

ENDIF

### Examples:

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

V1=500 ;\*\*\*Sets the variable 1 to 500

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

## **LSPD[axis]**

### Description:

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

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

Range is from 1 to 300,000.

### Syntax:

**Read:** [variable]=LSPD[axis]

**Write:** LSPD[axis]=[long value]

LSPD[axis]=[variable]

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

ENDIF

IF LSPD[axis]=[value]

ENDIF

### Examples:

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 50

## ***MST[axis]***

Description:

**Command:** Get motor status of axis

Syntax:

MST[Axis]

Examples:

```
IF MSTX=0
    DI=6
ELSEIF MSTY=0
    DI=3
ELSEIF MSTZ=0
    DI=2
ELSEIF MSTU=0
    DI=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
```

## ***STOP***

Description:

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

Syntax:

STOP

Examples:

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

## **STOP[axis]**

### Description:

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

### Syntax:

STOP[axis]

### Examples:

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

## **STORE**

### Description:

Store device settings and second half of general purpose variable registers (V32-V63) to flash.

### Syntax:

STORE

### Example:

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

## **SUB**

### Description:

Indicates start of subroutine

### Syntax:

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

### Examples:

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

## U

Description:

**Command:** Perform U axis move to target location

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 4CX has 64 variables [V0-V63]

Syntax:

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

*Special case for BIT NOT:*

V[Variable Number] = ~[Argument]

[Argument] can be any of the following:

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

[Operation] can be any of the following

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

Examples:

```

V1=12345      ;***Set Variable 1 to 123
V2=V1+1      ;***Set Variable 2 to V1 plus 1
V3=DI        ;***Set Variable 3 to digital input value
V4=DO        ;***Sets Variable 4 to digital output value
V5=~EO       ;***Sets Variable 5 to bit NOT of enable output value

```

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

## **WAIT**

Description:

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

Syntax:

```

WAIT[axis]
X[variable]

```

Examples:

```

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

```

## **WHILE**

### Description:

Perform WHILE loop

### Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

### Examples:

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

## X

Description:

**Command:** Perform X axis move to target location

Syntax:

X[value]  
X[variable]

Examples:

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

Y3000 ;\*\*\*Move Y to 3000

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

Syntax:

Y[value]  
Y[variable]

Examples:

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

Z3000 ;\*\*\*Move Z to 3000

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

Syntax:

Z[value]

Z[variable]

Examples:

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

Y1000 ;\*\*\*Move Y to 1000

V10 = 1200 ;\*\*\*Set variable 10 value to 1200

ZV10 ;\*\*\*Move Z Axis to variable 10 value

## **Contact Information**

Arcus Technology, Inc.

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

[www.arcus-technology.com](http://www.arcus-technology.com)