

DMX-J-SA

Integrated NEMA 17 Step Motor + Driver + Controller with USB 2.0 Communication



COPYRIGHT © 2007 ARCUS, ALL RIGHTS RESERVED

First edition, Oct 2007

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

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

Revision History:

- 1.01 - First revision
- 1.02– Product upgrade
- 1.03– Manual upgrade
- 1.04– Add torque curves, update software, edit Motion Profile section, edit Standalone section
- 1.05 - Manual Revision
- 1.06 - Manual Revision
- 1.07 - Added CCR command

Firmware Compatibility:

v107

Software Compatibility:

v107

Table of Contents

1. Introduction.....	5
2. Part Numbering Scheme	5
3. Dimensions	6
4. Torque Curves.....	7
5. Connectors	8
6. Electrical Specifications.....	10
Internal Interface Circuit Overview	10
Power Input.....	11
Limits, Home and Digital Inputs	11
Digital Outputs.....	11
Communication.....	12
Microstep	12
Driver Current.....	12
Operating Temperature	13
7. Getting Started	14
Typical Setup	14
1) PC based Control	14
2) Standalone Control.....	15
8. Windows GUI.....	16
A. Motor Status.....	17
B. Motor Control.....	18
C. DI Status/DO Status/Enable.....	19
D. Configuration	20
E. Program Control	21
F. Program Text	22
G. Variables	23
9. Motion Control Overview.....	24
Motion Profile and Speed	24
Position Counter.....	25
Target Move.....	25
Home Move	25
Jog Move.....	26
Stopping Motor	26
Limit Switch Function	26
Digital Inputs	27
Digital Outputs.....	27
Motor Status.....	27
Polarity.....	28
Idle Current and Run Current	28
Standalone Program Specification.....	28
Device Number	28
Storing to Flash.....	29
10. Communication.....	30

USB Communication Issues	31
11. ASCII Language Specification	32
12. Standalone Language Specification	34
;	34
ABORTX	34
ABS	35
ACC	35
DELAY	36
DI	36
DI[1-8]	37
DO	37
DO[1-2]	38
ECLEARX	39
ELSE	39
ELSEIF	40
END	41
ENDIF	41
ENDSUB	42
ENDWHILE	42
EO	43
GOSUB	43
HOMEX[+ or -]	44
HSPD	44
IF	45
INC	46
JOGX[+ or -]	46
LSPD	47
MSTX	47
PX	48
STOPX	48
STORE	49
SUB	49
V[0-49]	50
WAITX	51
WHILE	52
X	53
Standalone Example Program 1	54
Standalone Example Program 2	54
Standalone Example Program 3	54
Standalone Example Program 4	55
Standalone Example Program 5	55
Standalone Example Program 6	56

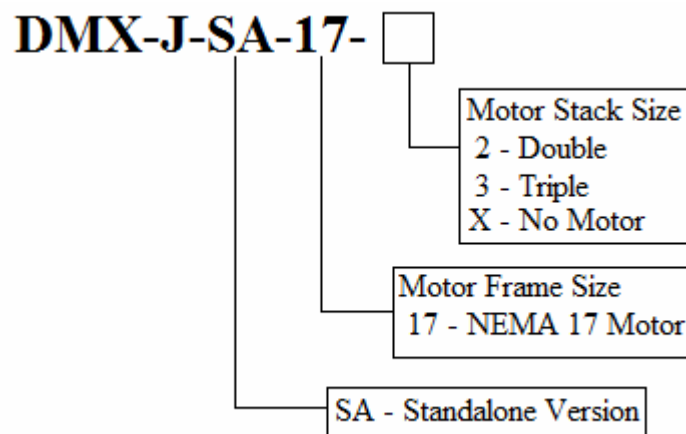
1. Introduction

DMX-J-SA is an integrated step motor + microstep driver + standalone controller with USB 2.0 communication.

DMX-J-SA Features

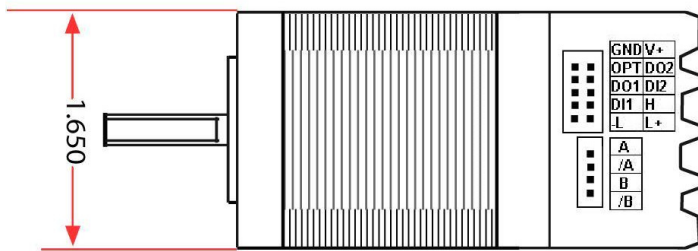
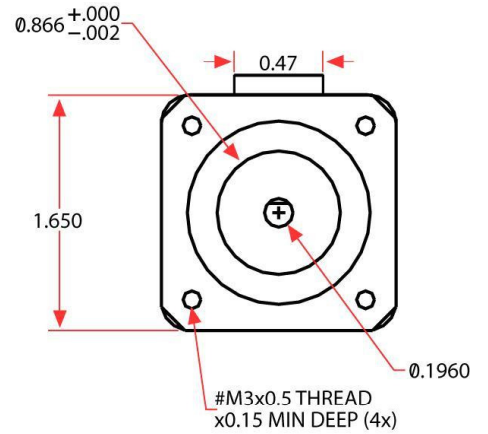
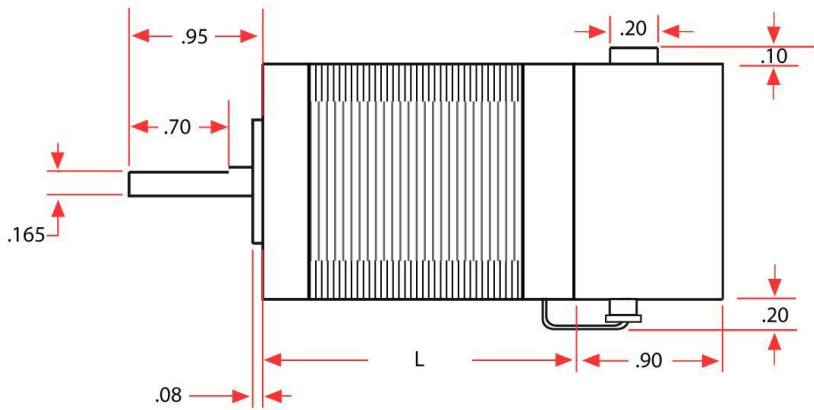
- USB 2.0 Communication
- PC based control through USB 2.0
- Standalone Control with BASIC-like programming language
- 24VDC voltage input
- 100mA to 2.0A microstep driver peak current setting
- 16 microstep
- Opto-isolated +Limit, -Limit, and Home inputs
- Two Opto-isolated Digital inputs
- Two Opto-isolated Digital outputs

2. Part Numbering Scheme



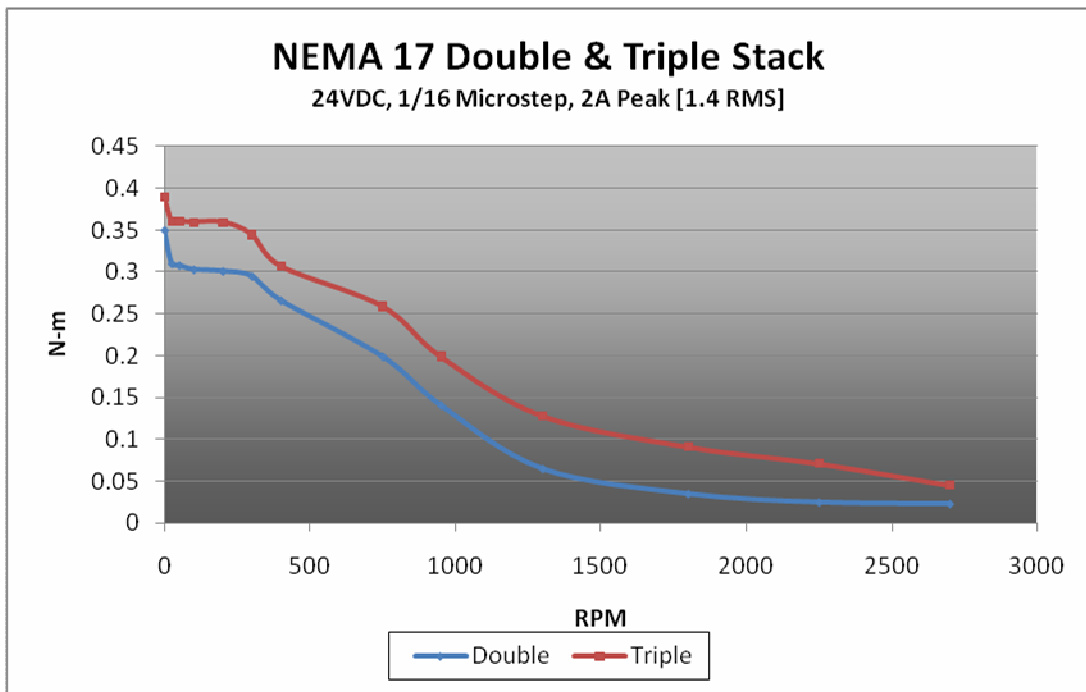
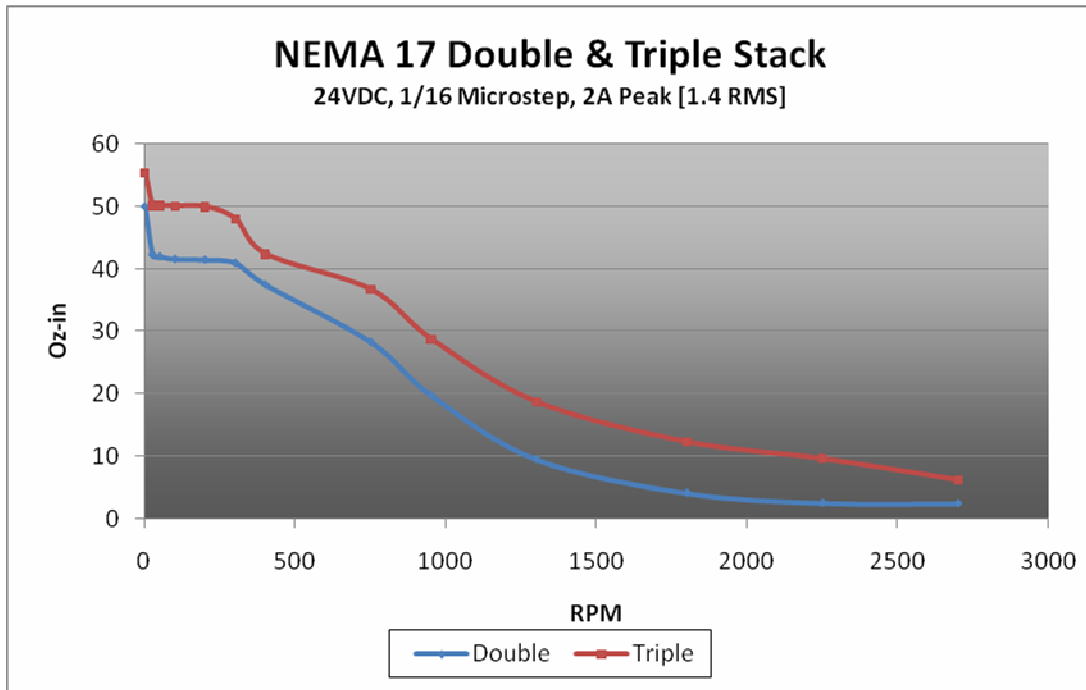
- Standard sizes available for NEMA 17 are double and triple.
- DMX-J-SA is available without a motor.

3. Dimensions

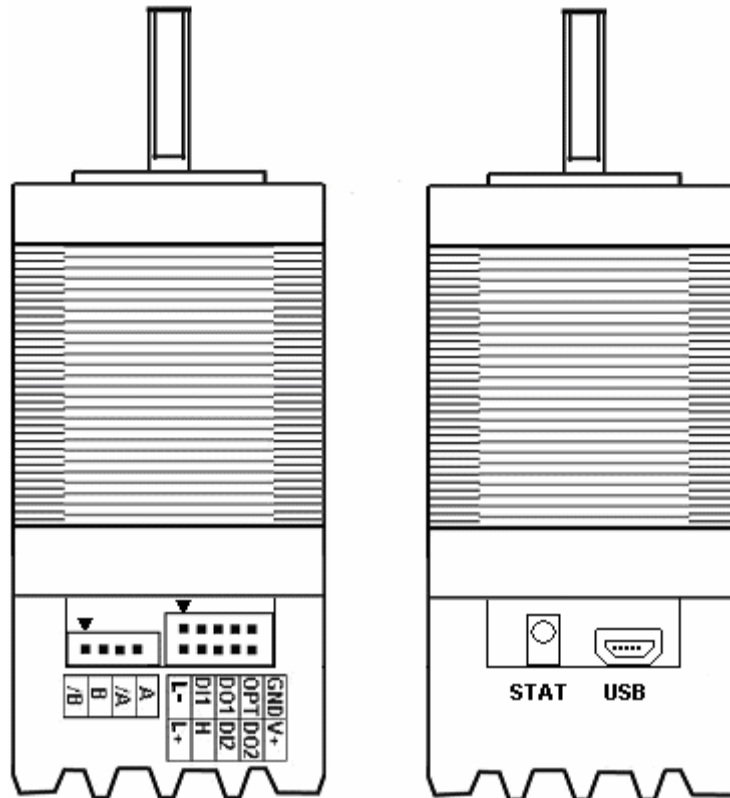


NEMA 17 Models	L (inches)
DMX-J-SA-17-2 (double stack)	1.58
DMX-J-SA-17-3 (triple stack)	1.89

4. Torque Curves



5. Connectors



10 pin Connector Pin-outs

Description	Pin	Pin	Description
-Limit Input	1	2	+Limit Input
Digital Input 1	3	4	Home Input
Digital Output 1	5	6	Digital Input 2
Opto Supply	7	8	Digital Output 2
Ground	9	10	Power V+

10 pin Mating Connector Information

Description: Female 10 pin 2mm dual row
 Manufacturer: HIROSE
 Part Number: DF11-10DS-2C (10 pin female connector)
 DF11-2428SC (female socket pin)

Important Notes:

- * Maximum current that can be handled by the connector is 2 Amps peak current.*
- * For the Power V+ and Ground lines, 24-gage wire with Teflon insulation is recommended. Check with wire manufacturer for the maximum current rating for the wire.*

4 pin connector Pin-outs

Pin	Description
1	/B
2	B
3	/A
4	A

4 pin Mating Connector Information

Description: Female 10 pin 2mm single row

Manufacturer: HIROSE

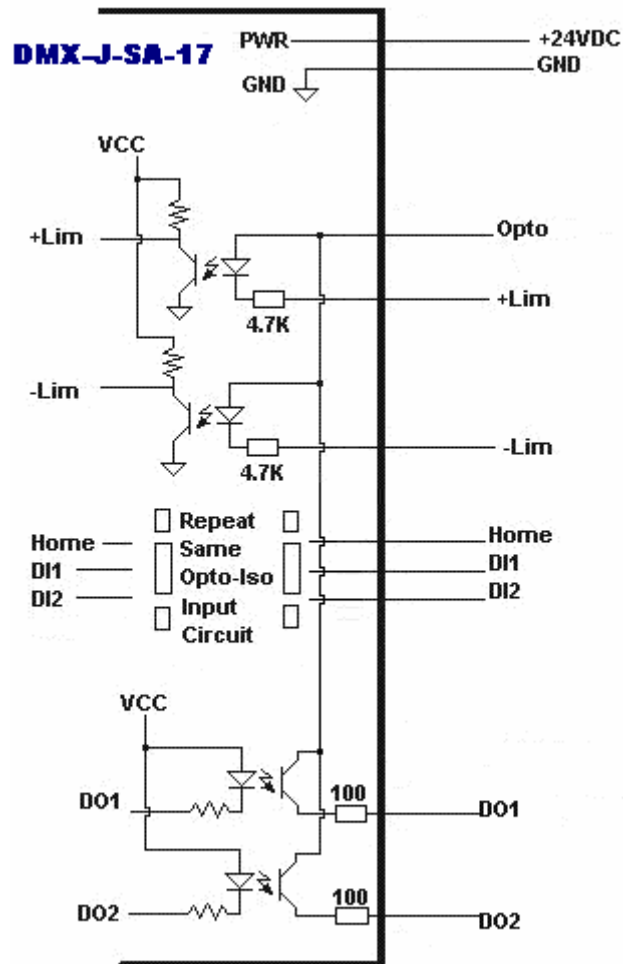
Part Number: DF3-4S-2C (4 pin female connector)
 DF3-2428SC (female pin)

USB Connector

Type: Mini-B to A

6. Electrical Specifications

Internal Interface Circuit Overview

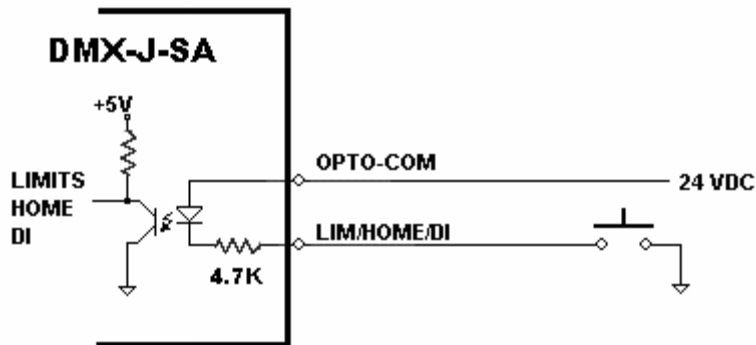


Power Input

Regulated Supply Voltage Range: **+12 to 24 VDC**
 Recommended Current for power supply: **2.0 Amp**

Limits, Home and Digital Inputs

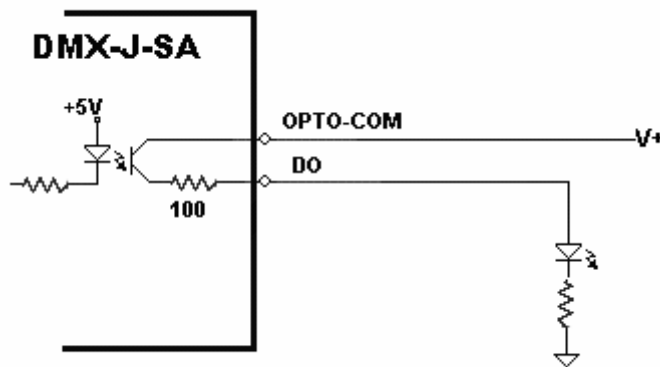
Type: **Opto-isolated inputs**
 Opto voltage supply input: **+24 VDC**
 Maximum diode forward current: **50mA**



4.7K resistor is built in to the limit, home and digital inputs to limit the current across the diode of the opto-isolator.

Digital Outputs

Type: **Opto-isolated open-emitter transistor output**
 Maximum emitter current: **50 mA**



Communication

Communication: **USB 2.0**
 Connector: **Mini-B to A**

Microstep

Microstep Value: 16 microstep fixed

Driver Current

Minimum driver current value: 100 mA (peak)
 Maximum driver current value: See below chart

Product	Maximum Rated Driver Peak Current Setting (Amp)	Recommended Driver Peak Current Setting (Amp)
DMX-J-SA-17-2	1.7	1.4
DMX-J-SA-17-3	2.0	1.6

Note: Setting the driver current higher than the maximum rated current will overheat the motor and the driver and potentially damage the unit. Recommendation is to use the current setting that is in the range of 60-80% of the maximum rated current of the motor.

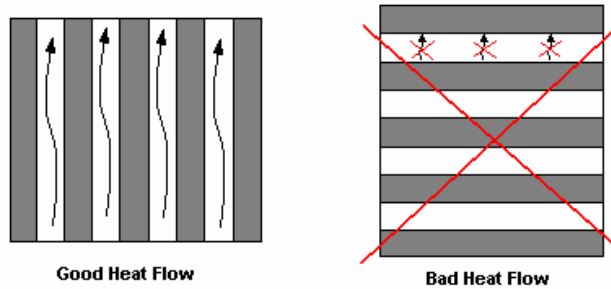
DMX-J-SA has configurable current setting from 100mA to 2.0A. Driver current is set to Run Current whenever the motor is moving. Driver current is set to Idle Current when the motion is idle for the duration set by the Idle Time. Run Current and the Idle Current should not go over the maximum rated current for each motor size. Use the chart above as a reference on maximum rated current setting.

Operating Temperature

Electronic components used in DMX-J-SA have maximum ambient operating temperature of **85 degree Celsius**. DMX-J-SA electronics are potted with heat-conductive compound to the housing to evenly distribute the heat and reduce any hot spots in the driver. Housing also has integrated fins to better dissipate the heat.

DMX-J-SA should be mounted securely to a metallic bracket that can also act as a heat-sink. During operation, step motor section typically becomes hotter than the driver section. Having the step motor mounted to a heat sink will better dissipate the heat generated by the step motor.

DMX-J-SA mounting orientation should be such that the fins are oriented vertically for better convection and heat dissipation.

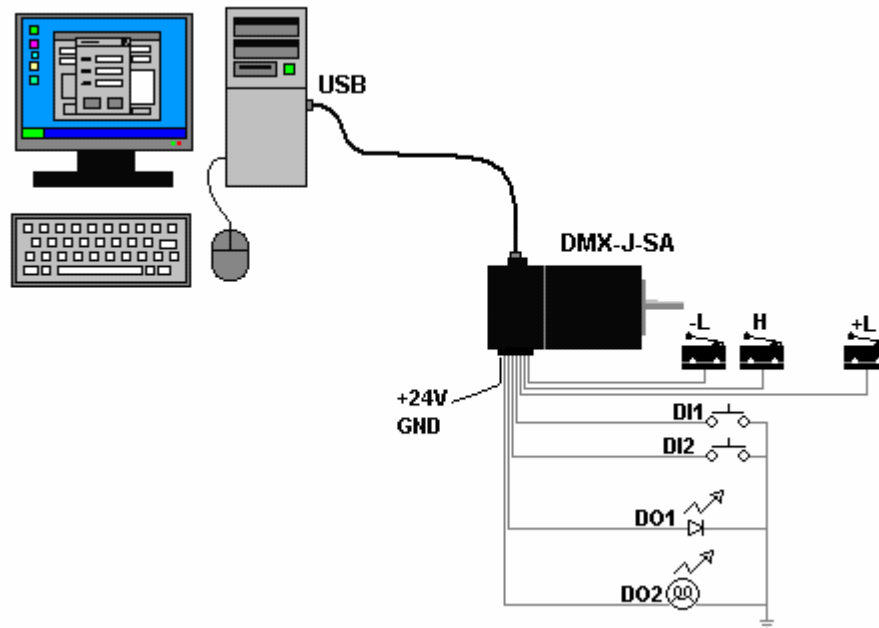


7. Getting Started

DMX-J-SA is an integrated stepper motor + driver + controller with USB 2.0 communication. DMX-J-SA can work in two modes as shown below.

Typical Setup

- 1) **PC based Control** - As slave to a Windows PC through USB 2.0 communication as shown below.

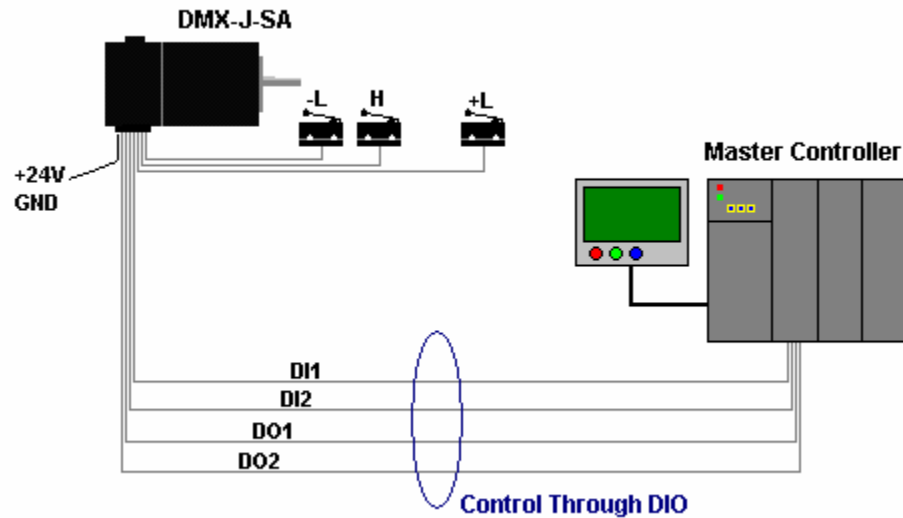


Common Windows programming language (Visual Basic, Visual C++, .Net, LabView, etc.) can be used to develop PC based control system. Requirements for PC based control are the following:

- i. Windows XP/VISTA compatible
- ii. DLL functions can be called from the program.

Sample VB6, VC++, LabView source code programs can be downloaded from the Arcus website.

- 2) **Standalone Control** - As a standalone controller with its own program running and DMX-J-SA can interface with another controller through the digital IO. Digital IO can be used to perform operations such as trigger a sequence of motions or report the motion completion and error status.



Standalone program is written using a BASIC-like language using the DMX-J-SA GUI Windows program that can be downloaded from the Arcus website.

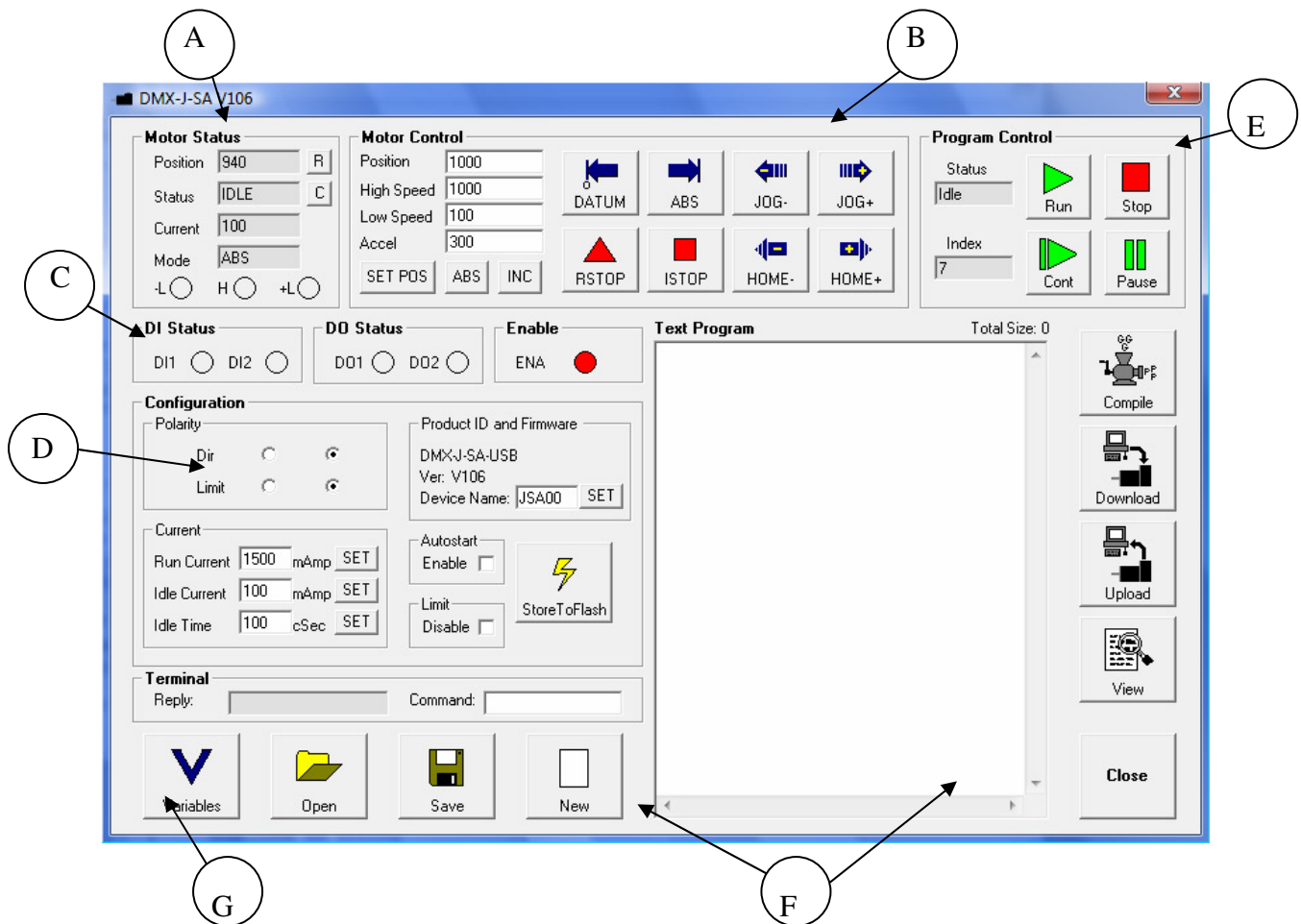
8. Windows GUI

DMX-J-SA comes with Windows GUI program to test, program, compile, download, and debug the controller.

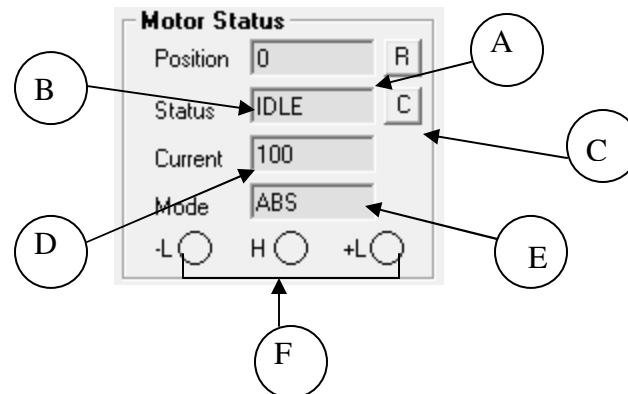
Important Note: In order to communicate with DMX-J-SA through USB, proper driver must be installed first. Before connecting the DMX-J-SA device or running any program, please go to the Arcus web site and download the USB driver installation instruction and run the USB Driver Installation Program.

Make sure that the USB driver is installed properly before running the controller.

Startup the DMX-J-SA GUI program and you will see following screen:

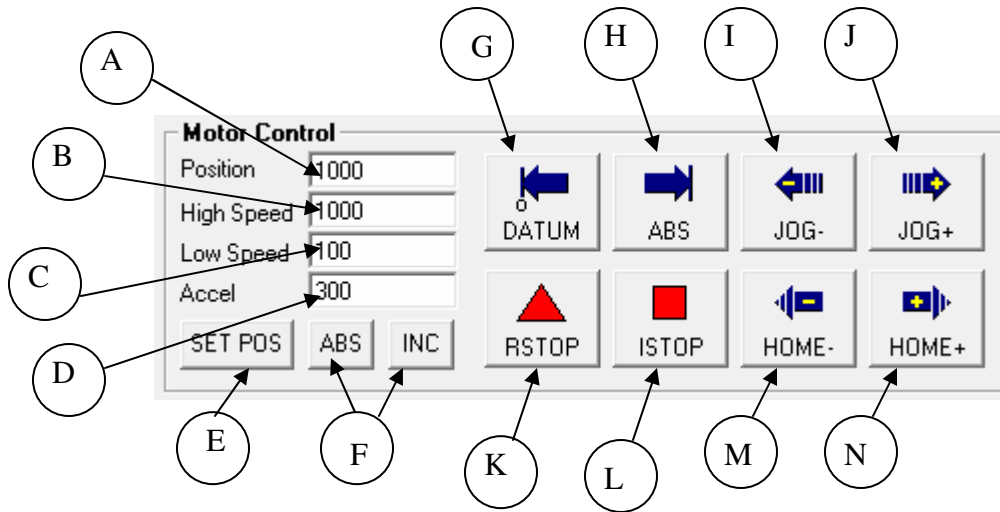


A. Motor Status



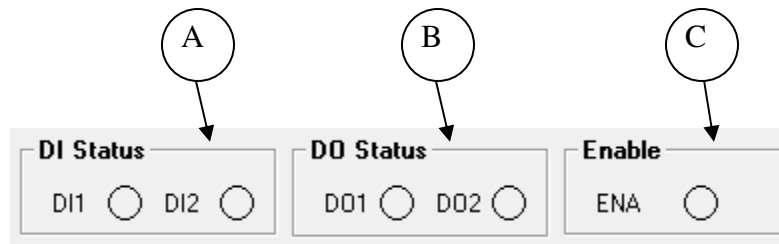
- A. Position** – Display of current motor position counter value
- B. Status** – Display of current motor status. Possible values are
 - ACCEL – acceleration in progress
 - CONST – constant speed in progress
 - DECEL – deceleration in progress
 - LIM ERROR – minus limit error occurred
 - +LIM ERROR – plus limit error occurred
- C. Clear Error** – Clear Error button is used to clear the Limit error status.
- D. Current** – Display of driver current.
- E. Mode** – Displays the move mode of the motor (**ABS/INC**).
- F. Limit and Home Input Status** – Display of limits and home input status.

B. Motor Control



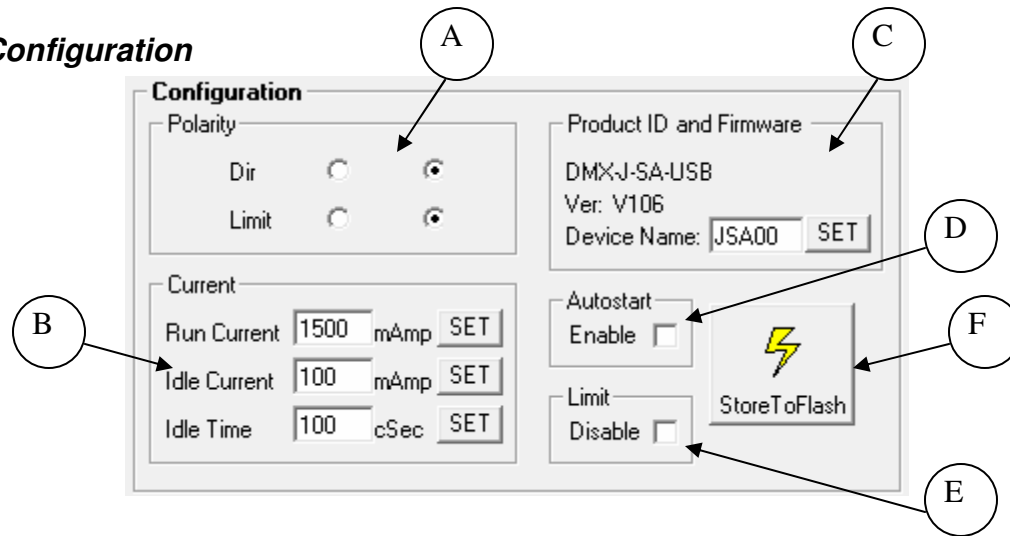
- A. Target Position** – Target position to move to for targeting position move.
- B. High Speed** – High Speed Value. Range from 100 to 200,000.
- C. Low Speed** – Low Speed Value. Range from 100 to 200,000.
- D. Acceleration** – Acceleration Value. Range from 10 to 1000.
- E. Set Position** – Set Position button. Uses the Position value as the Set Position
- F. ABS/INC** – Set the move mode of the controller.
- G. DATUM** – Absolute move to zero position. Maximum delta from current position to target is 262,143. If greater, then move will not perform.
- H. ABS** – Absolute move to target position. Maximum delta from current position to target is 262,143. If greater, then the move will not perform.
- I. JOG-** - Jog to minus direction
- J. JOG+** - Jog to plus direction
- K. RSTOP** – Stop with deceleration
- L. ISTOP** – Immediate stop without deceleration
- M. HOME-** - Homing in minus direction
- N. HOME+** - Homing in plus direction

C. DI Status/DO Status/Enable



- A. Digital Input Status** – Display of the two digital input bits. If the digital input pin is grounded, the digital input is turned on.
- B. Digital Output Status** – Display of the two digital output status. Digital output can be toggled by clicking on the circle.
- C. Enable Output Status** – Enable output status. Enable output can be toggled by clicking on the circle. When the enable is on, the driver is enabled and the motor is energized.

D. Configuration



A. Polarity – Motor rotation direction can be configured. Limit switch input can be configured. Limit switch polarity setting is valid only when the limit switch function is enabled.

B. Current – Motor drive current configuration. Driver current is set to Run Current when ever the motor is moving. Driver current is set to Idle Current when the motion is idle for the duration set by the Idle Time. Run Current and the Idle Current should not go over the maximum rated current for each motor size. Minimum current setting is 100mA and maximum is 2000mA.

C. Product ID and Firmware – Displays the USB product ID and the firmware version number. Device name can be changed so that multiple DMX-J-SA can be connected on the USB. When entering the new Device Name, make sure to enter in the format of JSAXX where XX ranges from 00 to 99.

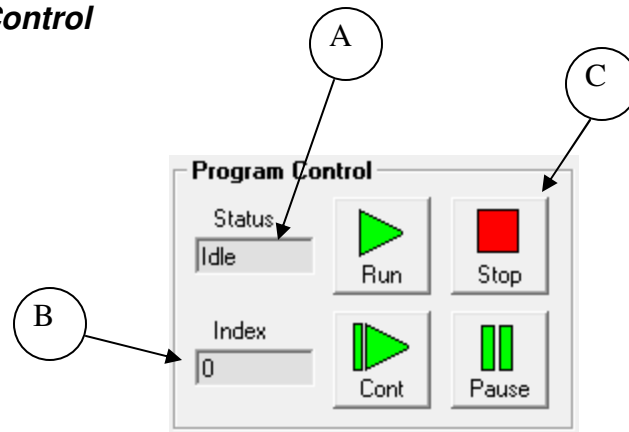
D. Autostart – Autostart is for automatic startup of the standalone program. If this is checked, the standalone program startup up automatically when the unit is powered.

E. Limit Disable – Limit Switch function can be disabled so that the limit switch inputs can be used as general purpose inouts.

F. Store To Flash – Parameters on the DMX-J can be permanently stored on the flash memory. When the unit is powered, the parameter values stored on the flash is loaded and used. Following are parameters that are stored on the flash.

- Direction Polarity
- Limit Polarity
- Autostart Enable
- Disable Limit switch function
- Device Name
- Run/Idle Currents and Idle Time
- Non-volatile variables (V25-V49)

E. Program Control



A. Program Status – display of program status. Possible statuses are

Idle – program is not running

Running – program is running

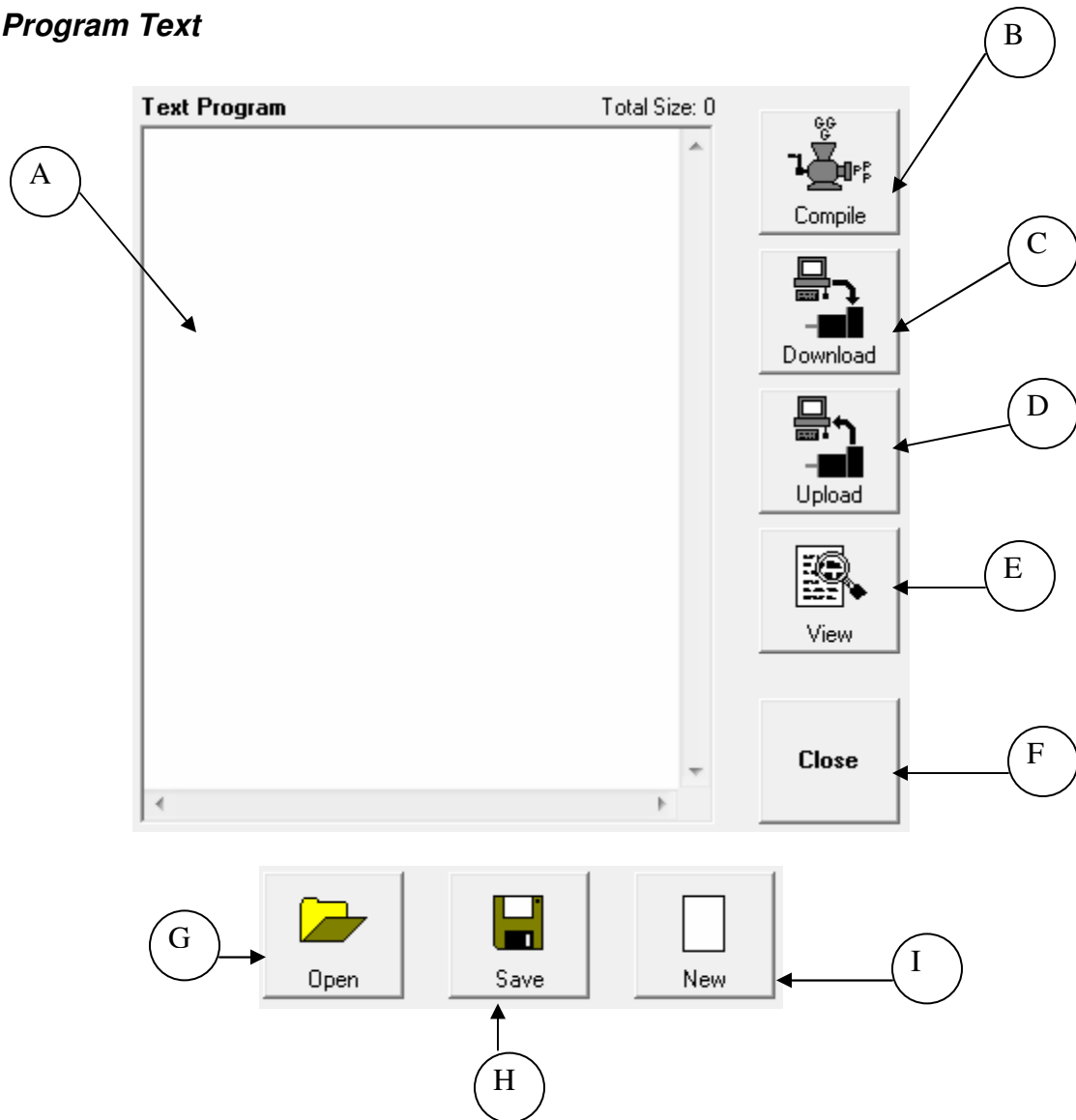
Paused – program is paused

Error – program is in error state.

B. Program Index – display of low level program index. This is the index of the low level program index.

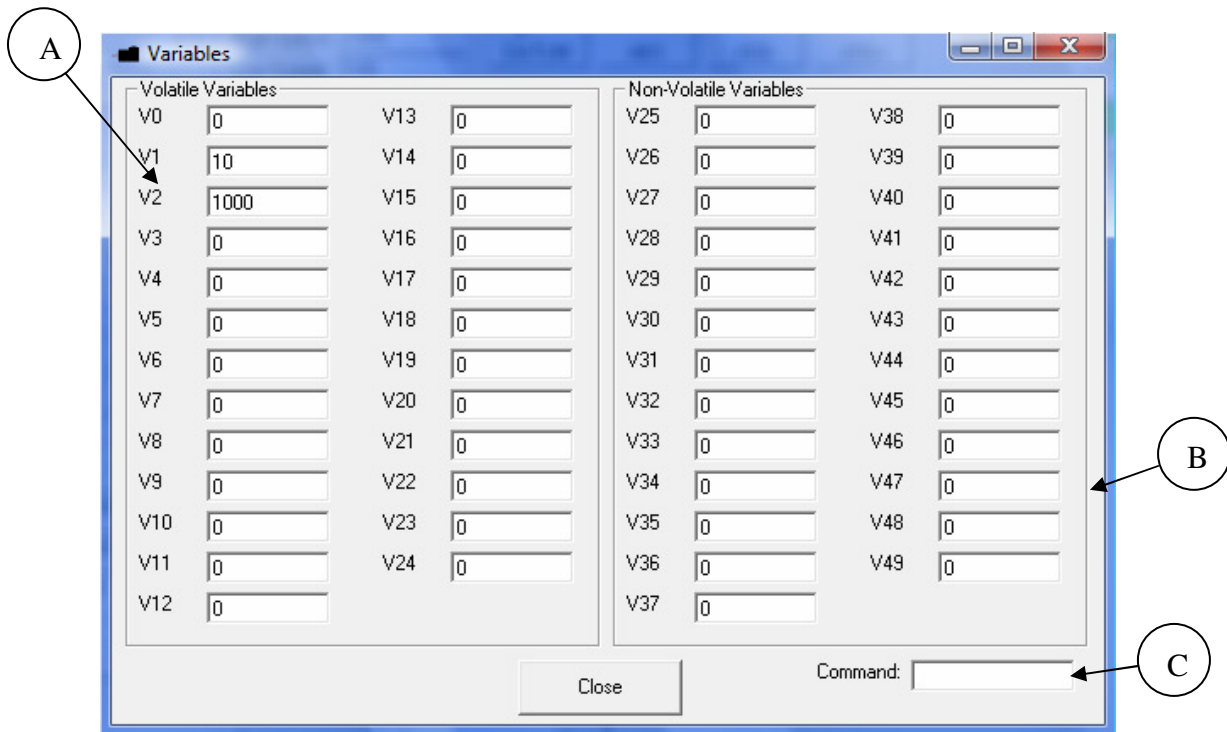
C. Program Control – Program can be RUN, STOP, PAUSED, and CONTINUED.

F. Program Text



- A. Text Program** - Program text can be entered and edited here.
- B. Compile** – Program is compiled to low level
- C. Download** – Compiled program is downloaded to the controller.
- D. Upload** – Program in the controller is uploaded and decompiled to high level.
- E. View** – Compiled low-level program can be viewed.
- F. Close** – DMX-J-SA GUI program is closed.
- G. Open** – Open program
- H. Save** – Save program
- I. New** – Clear the program text section

G. Variables



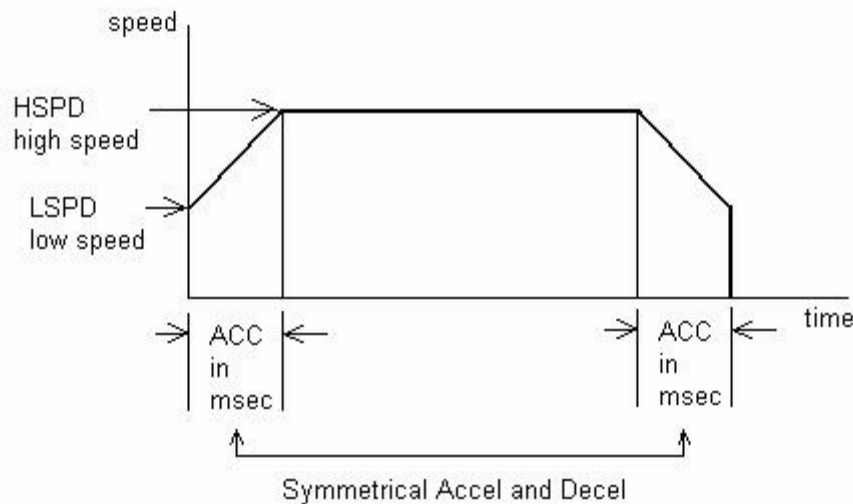
- A. Volatile Variables** – These variables are not stored to flash on the **STORE** command.
- B. Non-Volatile Variables** – These variables are stored to flash on the **STORE** command.
- C. Command Line** – To write to a variable, use `V[0-49]=[value]` syntax.

9. Motion Control Overview

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

Motion Profile and Speed

DMX-J-SA incorporates trapezoidal velocity profile as shown below.



Acceleration and deceleration time, set by using the **ACC** command, is in milliseconds and are symmetrical. Acceleration range is from 10 msec to 1000 msec.

High speed and low speed are in pps(pulses/second). Use the **HSPD** and **LSPD** command to retrieve and set low speed and high speed values. The supported pulse output rate is from 100 to 200K pulses/second. Depending on the voltage, current, the motor type and acceleration value, the maximum achievable speed will vary. Standard DMX-J-SA-17 comes with 1.8 degree motor with 16 microstep fixed setting which means 3200 pulses/rev. To convert rotational speed to pulse speed, use the following formula.

$$\text{Pulse/Sec} = \text{RPM} * 53.33$$

For example to achieve 3000 RPM (rev/min) 160K pulses/second speed is required.

Note on low speed and high speed settings:

The minimum **LSPD** value allowable depends on the **HSPD** value. If the **LSPD** is set below its minimum allowable value, movement will not be allowed.

Lowest Speed [pps]	Highest Speed [pps]
1	8K
2	16K
5	40K
10	80K
20	160K
50	200K

Note: If the low speed is set below its requirement for its corresponding high speed, the lowest allowable low speed value will be used.

Position Counter

DMX-J-SA has 32 bit signed position counter. Range of the position counter is from -2,147,483,648 to 2,147,483,647. Motor position can be read and set using the **PX** 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.

Target Move

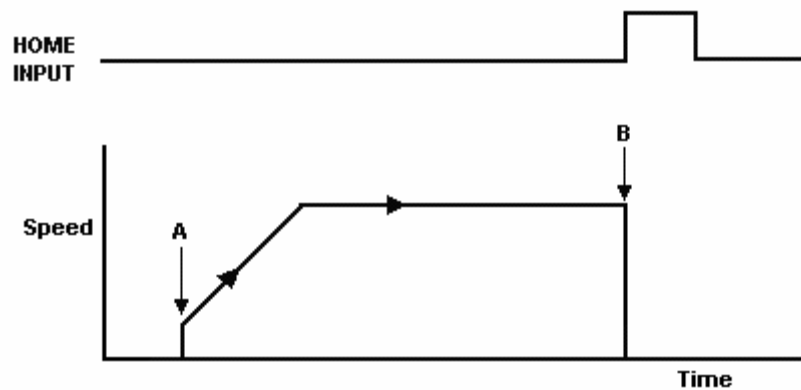
Target move, also known as absolute move, is used to move the motor to the desired position from the current position. Use the **X** command to move the axis to the target position.

Target motion has two modes, incremental mode, done by issuing the **INC** command, and absolute mode, done by issuing the **ABS** command. Under incremental mode, the command **X1000** will move the motor by 1000 from the current position. Under absolute mode, the command **X1000** will move the motor to absolute position 1000. To read which move mode the device is in use the **MM** command.

Maximum allowable difference to target position from current position is 262,143. Maximum difference between current position and the target position has to be less than or equal to 262,143. For example, if the current position counter is 1000, target position allowed will be between -261,143 (1,000-262,143) and 263,143 (1,000+262,143).

Home Move

Home search sequence involves moving the motor towards the home switch and then stopping when the home input is detected. To home the motor use the **H+ / H-** command. The following sequence shows the homing routine.



- A. Issuing home command starts the motor from low speed and accelerates to high speed.
- B. As soon as the home input is triggered, the position counter is reset to zero and the motor stops immediately. If the home switch is triggered in the middle of the acceleration, the motor stops immediately.

To trigger the home input switch, supply the opto-supply voltage with 24VDC and connect the home input signal to opto-supply ground.

If home switch is not used, home input can also be used as general purpose input. Digital input assignment for home input switch is **DI4**.

Jog Move

Jog move is used to continuously move the motor without stopping. To jog the motor, use the **J+/J-** command.

Stopping Motor

When motor is moving, jogging, or homing, motion can be stopped abruptly, using the **ABORT** command, or with deceleration, using the **STOP** command. It is recommended to use decelerate and stop command so that there is less impact to the system.

Limit Switch Function

With limit switch function enabled, triggering of the limit switch in motion will stop the motion immediately depending on the direction of the motion. If positive limit switch is triggered while moving in positive direction, motor will immediately stop and the motor status bit for positive limit error is set. Same is for negative limit while moving in negative direction. Once limit error is set, status must be cleared using the **CLR** command in order to move again. Once the error is cleared, move the motor out of the limit switch.

Limit switch function can also be disabled using the **DL** command. By disabling the limit switch function, the limits switches can be used as general purpose inputs. When limit function is disabled, digital input assignments are: **DI3** for -Limit and **DI5** for +Limit.

Digital Inputs

DMX-J-SA module comes with 3 digital inputs. If the limit function is disabled, up to 5 inputs can be used as general purpose inputs

Read digital input status using the **DI** command. See description below:

Bit	Description
0	Digital Input 1
1	Digital Input 2
2	Negative Limit
3	Home
4	Positive Limit

Digital input values can also be referenced one bit at a time by the **DI[1-5]** commands. Note that the indexes are 1-based for the bit references (i.e. DI1 refers to bit 0, not bit 1)

Digital inputs are active high.

Digital Outputs

DMX-J-SA module comes with 2 digital outputs.

Read/set digital output status using the **DO** command. See description below:

Bit	Description
0	Digital Output 1
1	Digital Output 2

Digital output values can also be referenced one bit at a time by the **DO[1-2]** commands. Note that the indexes are 1-based for the bit references (i.e. DO1 refers to bit 0, not bit 1)

Digital outputs are active low.

Motor Status

Motor status can be read anytime using the **MST** command. The following is the bit representation for motor status.

Bit	Description
0	Motor running at constant speed
1	Motor in acceleration
2	Motor in deceleration
3	Home input switch status
4	Minus limit input switch status
5	Plus limit input switch status
6	Minus limit error. This bit is latched when minus limit is hit during

	negative direction motion. This error must be cleared before issuing any subsequent move commands.
7	Plus limit error. This bit is latched when plus limit is hit during positive direction motion. This error must be cleared before issuing any subsequent move commands.

Polarity

Using **POL** command, polarity of following signals can be configured:

Bit	Description
0	Direction
1	Limit

Idle Current and Run Current

DMX-J-SA allows for two current settings.

Run Current: Current used while the motor is running. Set using the **CUR** command

Idle Current: Current used when the motor is idle. Set using the **ACR** command

To set the amount of time the motor needs to be idle before changing to idle current, use the **ICN** command. Units are in centi-seconds.

When setting idle and run current, the range must be within 100mA to 2000mA, unless the user wishes to have the motor become disabled during idle state. To do this, set the idle current to 0.

The actual current of the motor can be read using the **CCR** command.

Standalone Program Specification

Standalone Program Specification:

Memory size: 1275 assembly lines ~ 7.5 KB.

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

Error Handling:

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

Device Number

DMX-J-SA module provides the user with the ability to set the device number of a specific device. In order to make these changes, first store the desired number using the

DN command. Please note that this value must be within the range [JSA00-JSA99]. To write the values to the device's flash memory, use the **STORE** command. After a complete power cycle, the new device ID will be written to memory. Note that before a power cycle is completed, the settings will not take effect.

Storing to Flash

The following items are stored to flash:

- Device name
- Polarity settings
- Current settings
- Automatic program run on boot
- Second half of general purpose variables (V25-V49)

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

10. Communication

DMX-J-SA uses Performax communication, which is USB 2.0 compatible.

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

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

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

USB Communication API Functions

For USB communication, following DLL API functions are provided.

BOOL fnPerformaxComGetNumDevices(OUT LPDWORD lpNumDevices);

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

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

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

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

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

BOOL fnPerformaxComClose(IN HANDLE pHandle);

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

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

- This function is used to set the communication read and write timeout. Values are in milliseconds.

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

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

USB Communication Issues

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

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

BOOL **fnPerformaxComFlush**(IN HANDLE pHandle)

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

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



11. 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.

Following are interactive commands that can be sent from Windows program.

Command	Value Range	Unit	Description	Return
ABORT			Aborts the motion in progress.	OK
ABS			Sets the move mode to absolute mode	OK
ACC			Returns Acceleration value	10 to 1000
ACC=[value]	10 to 1000	msec	Sets Acceleration value	OK
ACR			Returns Idle Current	100 to 2000
ACR=[value]	100 to 2000	mA	Sets Idle Current	OK
CCR	100 to 2000	mA	Gets actual current of motor	100 to 2000
CLR			Clears Limit error	OK
CUR			Returns Run Current	100 to 2000
CUR=[value]	100 to 2000	mA	Sets Run Current	OK
DI	0 to 31		Returns the Digital Input status	0 to 31
DI[bit]			Returns Digital Input bit status DI1, DI2 – Digital inputs 1 and 2 DI3 – Minus Limit DI4 – Home DI5 – Plus Limit	0 or 1
DL			Return Disable Limit	0 or 1
DL=[value]	0 – disable 1 – enable		Sets Disable Limit	OK
DO			Returns Digital Output value.	0 to 3
DO=[value]			Sets Digital Output value.	
DO[bit]			Returns Status of Digital Output bit Bit 1,2 – DO1 and DO2	0 or 1
DO[bit] =[value]	0 – off 1 – on		Sets Digital Output bit	OK
EO			Returns Enable status	0 or 1
EO=[value]	0 – off 1 – on		Enables the motor power	OK
H+			Homes the motor in positive direction	OK
H-			Homes the motor in minus direction	OK
HSPD		pps	Returns high speed value	100 to 200000
HSPD=[value]	100 to 200,000	pps	Sets high speed value	OK
ICN		csec	Returns Idle Time. Each unit is 100 msec. For example, if ICN is 10, that means 1 sec will be the idle time.	1 to 100
ICN=[value]	1 to 100	csec	Sets Idle Time	OK
ID			Returns the product ID	DMX-J-SA-USB
INC			Sets the move mode to incremental mode	OK
J+			Jog the motor in positive direction	OK
J-			Jog the motor in minus direction	OK
LSPD		pps	Returns low speed	100 to 200000
LSPD=[value]	100 to 200,000	pps	Sets lows speed	OK
MM			Returns the current move mode 0 – ABS mode 1 – INC mode	0 or 1
MST			Returns the motor status Bit 0 – In Constant Speed Bit 1 – Accelerating	0 to 255

			Bit 2 – Decelerating Bit 3 – Home Input Switch (Also DI4) Bit 4 – Minus Limit Switch (Also DI3) Bit 5 – Plus Limit Switch (Also DI5) Bit 6 – Minus Limit Error Bit 7 – Plus Limit Error	
POL			Returns Polarity Bit 0 – Direction Polarity Bit 1 – Limit Polarity (Valid when Disable Limit is off)	0 to 3
POL=[value]	0 to 3		Sets polarity	OK
PX			Returns the pulse position counter	-2147483648 to 2147483647
PX=[value]	-2147483648 to 2147483647		Sets position counter.	OK
SASTAT	Get the current standalone program status		Returns standalone program status 0 – Idle 1 – Running 2 – Paused	0 to 3
SA[Line]			Get standalone line. Line: [0-1274]	
SA[Line]=[Value]			Sets standalone line. Line: [0-1274]	OK
SLOAD			Return automatic run mode.	0 or 1
SLOAD=[value]	0 – auto run off 1 – auto run on		Sets automatic run mode	OK
SPC			Returns the program counter for the standalone program	[0-1274]
SR=[value]	0 – Stop 1 – Run 2 – Pause 3 – Continue		Controls Standalone program	OK
STOP			Performs ramp down stop if motor is in motion	OK
STORE			Store parameters to Flash	OK
V[var#]			Return variable value. Var # from 0 to 49	-2147483648 to 2147483647
V[var#]=[value]	-2147483648 to 2147483647		Sets variable value. Var # from 0 to 49	OK
VER			Returns the current firmware version	VXXX
X[Value]	Maximum delta from current position to target = 262,143		Performs absolute move	OK

12. 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 axis to positive direction
DELAY=1000      ;***Wait 1 second
ABORT           ;***Stop immediately all axes including X axis

```

ABORTX

Description:

Motion: Immediately stop motion without deceleration.

Syntax:

ABORTX

Examples:

```

JOGX+           ;***Jogs axis to positive direction
DELAY=1000      ;***Wait 1 second
ABORTX          ;***Stop axis immediately

```

ABS

Description:

Command: Changes all move commands to absolute mode.

Syntax:

ABS

Examples:

```

ABS           ;***Change to absolute mode
PX=0         ;***Change position to 0
X1000        ;***Move X axis to position 1000
X3000        ;***Move X axis to position 3000
  
```

ACC

Description:

Read: Get acceleration value

Write: Set acceleration value.

Value is in milliseconds.

Syntax:

Read: [variable] = ACC

Write: ACC = [value]

ACC = [variable]

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
  
```

DELAY

Description:

Set a delay (1 ms units)

Syntax:

Delay=[Number] (1 ms units)

Examples:

```
JOGX+           ;***Jogs axis to positive direction
DELAY=10000     ;***Wait 10 second
ABORTX          ;***Stop axis
```

DI

Description:

Read: Gets the digital input value. DMX-J-SA has 5 digital inputs.

Digital inputs are active high

Syntax:

Read: [variable] = DI

Conditional: IF DI=[variable]
ENDIF

IF DI=[value]
ENDIF

Examples:

```
IF DI=0
    DO=1           ;***If all digital inputs are off, set DO=1
ENDIF
```

DI[1-8]

Description:

Read: Gets the digital input value. DMX-J-SA has 5 digital inputs.

Digital inputs are active high

Syntax:

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

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

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

Examples:

```
IF DI1=0
    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

DMX-J-SA has 2 digital outputs.

Digital outputs are active low.

Syntax:

Read: [variable] = DO

Write: DO = [value]

DO = [variable]

Conditional: IF DO=[variable]
ENDIF

IF DO=[value]
ENDIF

Examples:

```
DO=3      ;***Turn on both bits
```

DO[1-2]

Description:

Read: Gets the individual digital output value

Write: Sets the individual digital output value

DMX-J-SA has 2 digital outputs.

Digital outputs are active low.

Syntax:

Read: [variable] = DO[1-2]

Write: DO[1-2] = [0 or 1]

DO[1-2] = [variable]

Conditional: IF DO[1-2]=[variable]
ENDIF

IF DO[1-2]=[0 or 1]
ENDIF

Examples:

DO1=1 ;***Turn DO1 on

DO2=1 ;***Turn DO2 on

ECLEARX

Description:

Write: Clears motor error status.

Syntax:

Write: ECLEARX

Examples:

```
ECLEARX                ;***Clears motor error
```

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  
WAITX  
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
    WAITX
    X1000
    WAITX
ENDSUB
```

ENDWHILE

Description:

Indicate end of WHILE loop

Syntax:

ENDWHILE

Examples:

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

EO

Description:

Read: Gets the enable output value

Write: Sets the enable output value

Syntax:

Read: [variable] = EO

Write: EO = [value]

EO = [variable]

Conditional: IF EO=[variable]
ENDIF

IF EO=[value]
ENDIF

Examples:

```
EO=1           ;***Energize motor
```

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 0
END
```

```
SUB 0
    X0
    WAITX
    X1000
    WAITX
ENDSUB
```

HOMEX[+ or -]

Description:

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

Syntax:

HOMEX[+ or -]

Examples:

HOMEX+ ;***Homes axis in positive 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 200,000.

Syntax:

Read: [variable] = HSPD

Write: HSPD = [value]

HSPD = [variable]

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

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 position to 0
X1000        ;***Move axis to position 1000 (0+1000)
X2000        ;***Move axis to position 3000 (1000+2000)

```

JOGX[+ or -]

Description:

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

Syntax:

JOGX[+ or -]

Examples:

```

JOGX+       ;***Jogs axis in positive direction
JOGX-       ;***Jogs 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 200,000.

Syntax:

Read: [variable]=LSPD

Write: LSPD=[long value]

LSPD=[variable]

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

MSTX

Description:

Read: Get motor status

Syntax:

Read: [variable]=MSTX

Conditional: IF MSTX=[variable]

ENDIF

IF MSTX=[value]

ENDIF

Examples:

IF MSTX=0

DO=3

ELSE

DO=0

ENDIF

PX

Description:

Read: Gets the current pulse position

Write: Sets the current pulse position

Syntax:

Read: Variable = PX

Write: PX = [value]

PX = [variable]

Conditional: IF PX=[variable]
ENDIF

IF PX=[value]
ENDIF

Examples:

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

STOPX

Description:

Command: Stop all axes if in motion with deceleration.

Previous acceleration value is used for deceleration.

Syntax:

STOPX

Examples:

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

STORE

Description:

Command: Store all values to flash

Syntax:

STORE

Examples:

```
V40=PX           ;***Put position value in V40
DELAY=1000       ;***Wait 1 second
STORE            ;***Store V40 to non-volatile flash
```

SUB

Description:

Indicates start of subroutine

Syntax:

SUB [subroutine number]

[Subroutine Number] range is 0 to 31

Note: SUB 31 is reserved for error handling.

Examples:

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

V[0-49]

Description:

Assign to variable.
DMX-J-SA has 50 variables [V0-V49]

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
```

WAITX

Description:

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

Syntax:

WAITX

Examples:

```
X10000      ;***Move axis to position 10000
WAITX      ;***Wait until axis move is done
DO=3       ;***Set digital output
X3000      ;***Move axis to 3000
WAITX      ;***Wait until axis move is done
```

WHILE

Description:

Perform WHILE loop

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

```

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

X

Description:

Command: Perform X axis move to target location

Syntax:

X[value]
X[variable]

Examples:

```
ABS          ;***Absolute move mode
X10000       ;***Move to position 10000
V10 = 1200   ;***Set variable 10 value to 1200
XV10         ;***Move axis to variable 10 value
```

Standalone Example Program 1

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

```

HSPD=20000      ;* Set the high speed to 20000 pulses/sec
LSPD=1000       ;* Set the low speed to 1000 pulses/sec
ACC=300         ;* Set the acceleration to 300 msec
EO=1            ;* Enable the motor power
X1000           ;* Move to 1000
X0              ;* Move to 1000
END             ;* End of the program

```

Standalone Example Program 2

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

```

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

```

Standalone Example Program 3

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

```

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

```

Standalone Example Program 4

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

```

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

```

Standalone Example Program 5

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

```

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

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

```

Standalone Example Program 6

Task: If digital input 1 is on, move to position 1000. If digital input 2 is on, move to position 2000. If digital input 3 is on, move to 3000. If digital input 5 is on, home the motor in negative direction. Use digital output 1 to indicate that the motor is moving or not moving. **Note that in order to have digital input 3 and 5 working as digital input instead of limit switches, disable the limit switch function.**

```

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

```

Contact Information

Arcus Technology, Inc.

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

www.arcus-technology.com