DMX-J-SA

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

First edition, Oct 2007

Revision History:

1.01 - 1st revision
1.09 - 2nd release
1.10 - 3rd release

Firmware Compatibility:

†V116BL

†If your module's firmware version number is less than the listed value, contact Arcus for the appropriate documentation. Arcus reserves the right to change the firmware without notice.

Table of Contents

1. Introduction

DMX-J-SA is an integrated step motor with a microstep driver and controller.

Communication to the DMX-J-SA can be established over USB. It is also possible to download a stand-alone program to the device and have it run independent of a host.

Windows and Linux drivers, as well as sample source code, are available to aid you in your software development.

Features

DMX-J-SA

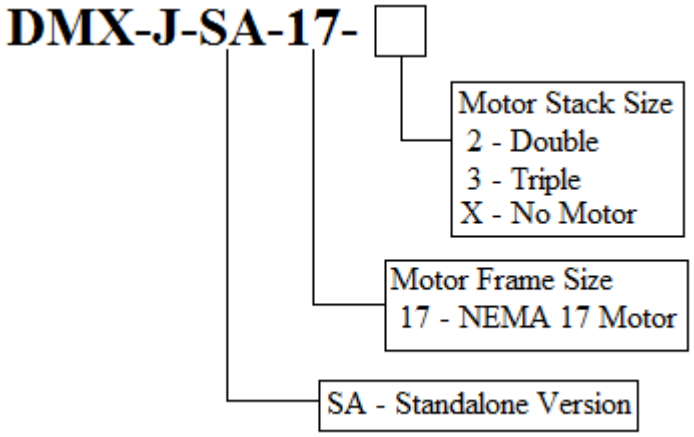
- USB 2.0 communication
- Stand-alone programmable
- 16 microstep driver (100mA to 2.0A peak current)
- NEMA 17 stepper motor
- Opto-isolated I/O
    - 2 x inputs
    - 2 x outputs
    - +Limit/-Limit/Home inputs
- Homing routines:
    - Home input only (high speed)
    - Home input only (high speed + low speed)
    - Limit only
- 12-24 VDC voltage input

Contacting Support

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

Or, contact your local distributor for technical support.

2. Part Numbering Scheme



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

3. Electrical and Thermal Specifications

Power Requirement

Regulated Voltage:	**+12 to 24 VDC**
Recommended Current (Max):	**2.0 Amp**

Temperature Ratings †

Operating Temperature:	**0°C to +85°C**
Storage Temperature:	**-55°C to +150°C**

† Based on component ratings

Digital Inputs †

Type:	**Opto-isolated NPN inputs**
Opto-isolator supply:	**+12 to +24 VDC**
Maximum diode forward current:	**45mA**

† Included limit and home

Digital Outputs

Type:	**Opto-isolated open-collector PNP outputs**
Max voltage at emitter:	**+24VDC**
Max source current at 24VDC:	**†45 mA**

† A current limiting resistor is required

4. Dimensions

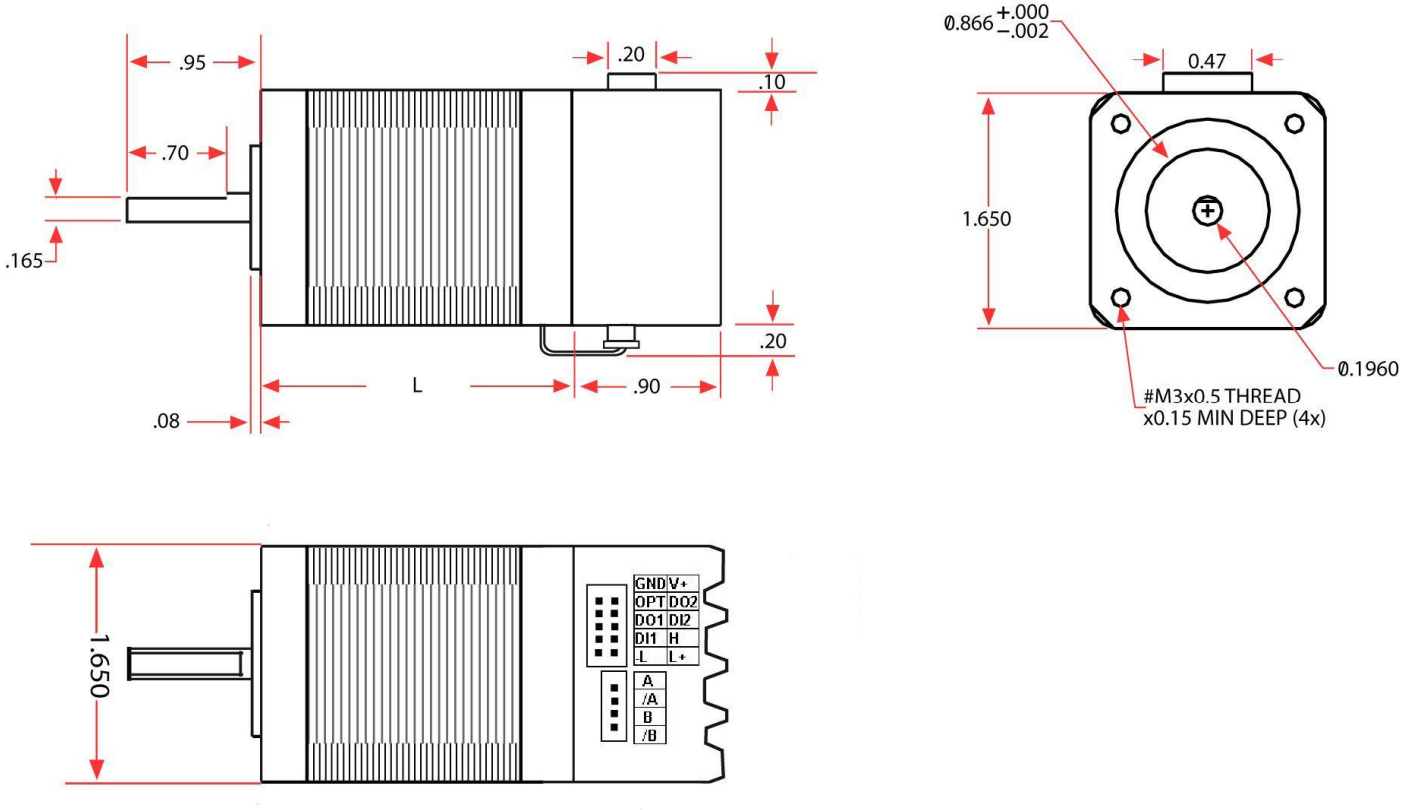
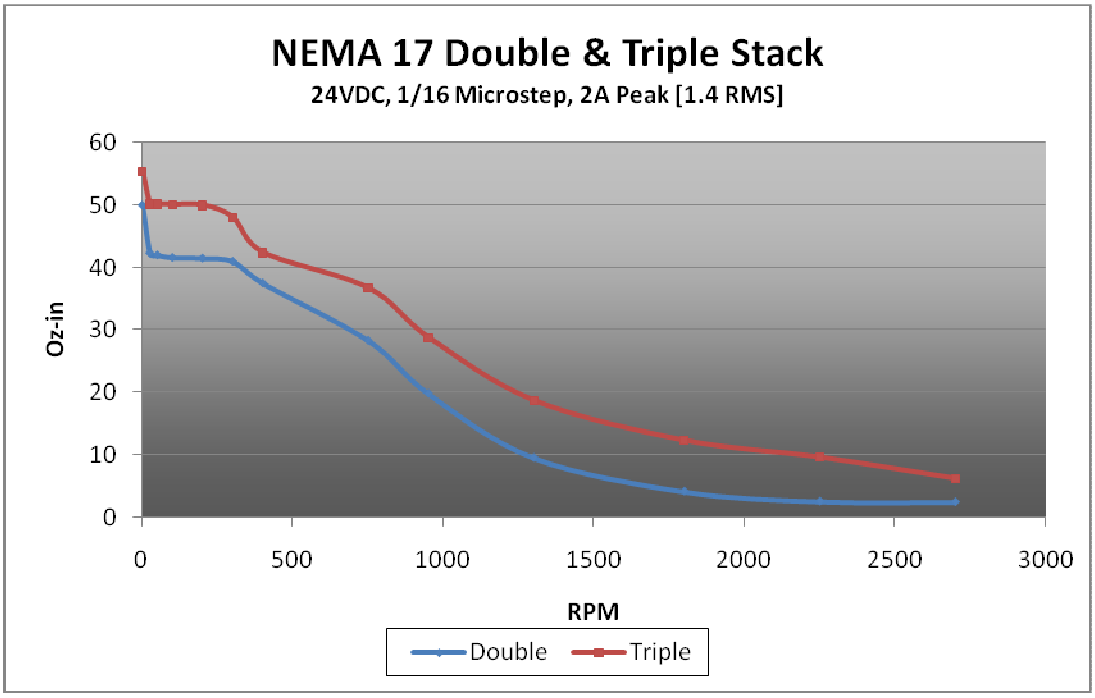


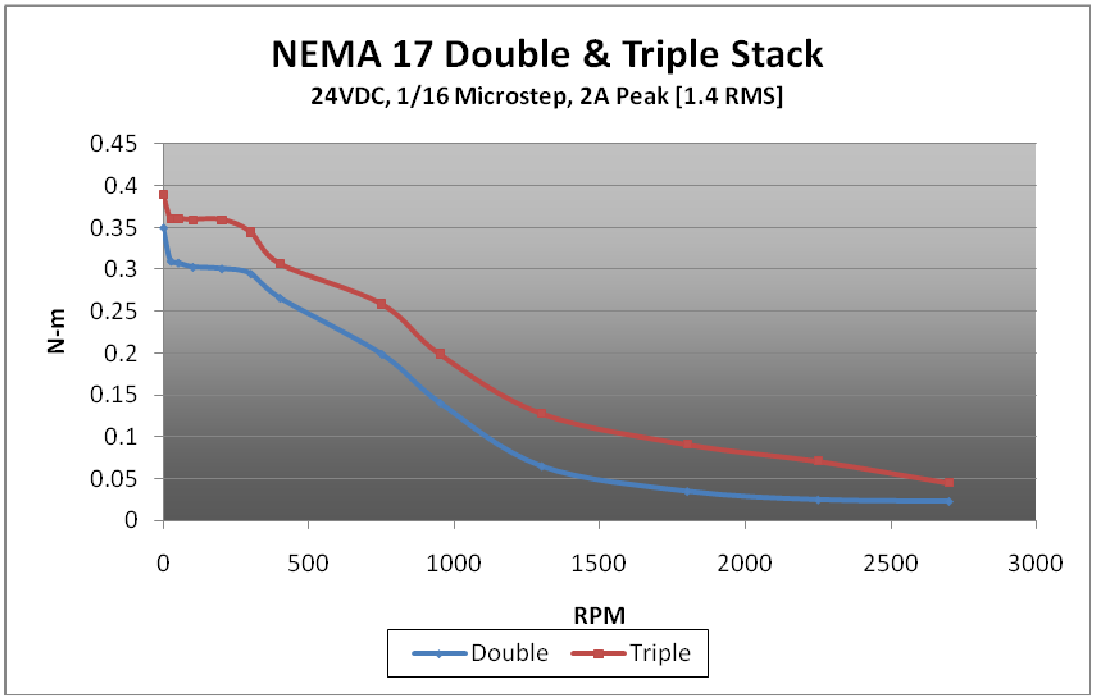
Figure 4.0

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

† All dimensions in inches

5. Torque Curves





6. Connectors

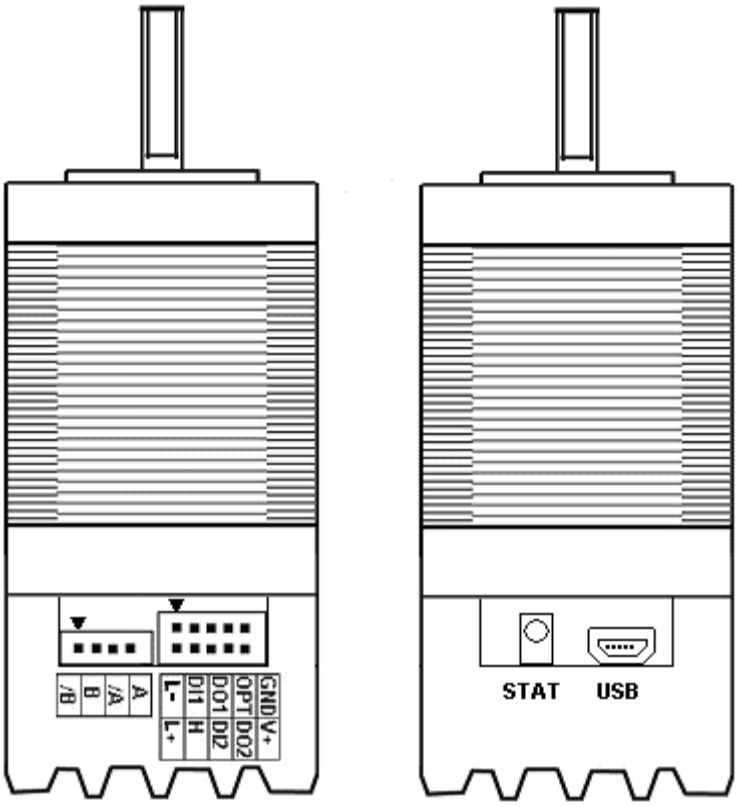


Figure 4.0

10 pin Connector (2.0mm) †

Pin #	In/Out	Name	Description
1	I	-L	Minus Limit Input
2	I	+L	Plus Limit Input
3	I	DI1	Digital Input 1
4	I	H	Home Input
5	O	DO1	Digital Output 1
6	I	DI2	Digital Input 2
7	I	OPT	Opto-supply input (+12 to +24VDC)
8	O	DO2	Digital Output 2
9	I	GND	Ground
10	I	V+	Power Input +12 to +24VDC ‡

Table 4.1

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

† Maximum current that can be handled by the connector is 2 Amps peak current.

‡ For the V+ and GND lines, 24-gage wire with Teflon insulation is recommended. Check with wire manufacturer for the maximum current rating for the wire.

4 pin Connector (2.0mm)

Pin #	In/Out	Name	Description
1	O	/B	Motor Winding /B
2	O	B	Motor Winding B
3	O	/A	Motor Winding /A
4	O	A	Moto Winding A

Table 4.2

Mating Connector Description: Female 10 pin 2mm single row
Mating Connector Manufacturer: HIROSE
Mating Connector Manufacturer Part: DF3-4S-2C (4 pin female connector)
DF3-2428SC (female pin)

USB Connector

Type: Mini-B to A

DMX-J-SA Interface Circuit

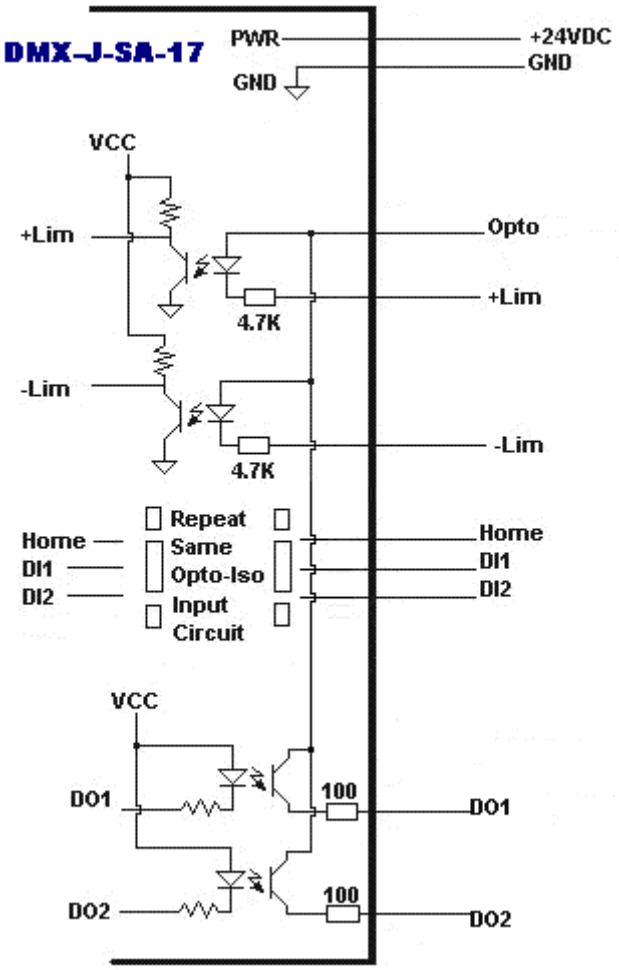


Figure 4.3

Power Input

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

Limits, Home and Digital Inputs

Figure 4.4 shows an example wiring of the home, limit, and digital inputs

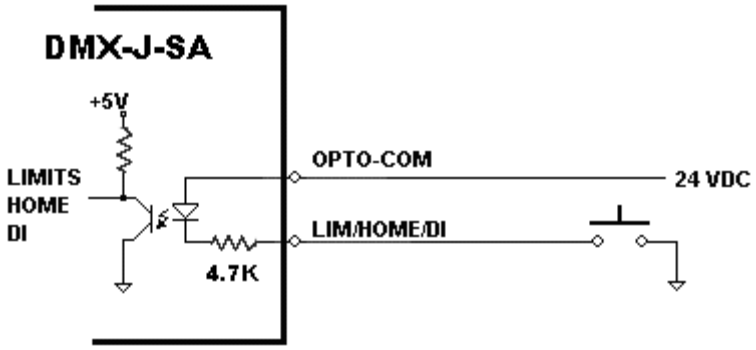


Figure 4.4

To activate an input, sink the line to opto-ground.

Digital Outputs

Figure 4.5 shows and example wiring of the digital outputs. The digital outputs will source to opto-supply.

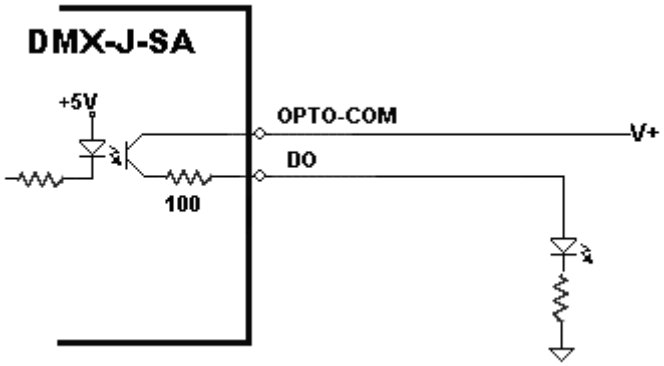


Figure 4.5

Microstep

Microstep Value: 16 microstep fixed

Driver Current

Minimum driver current value: 100 mAmp (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

Table 4.6

† 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. The 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.

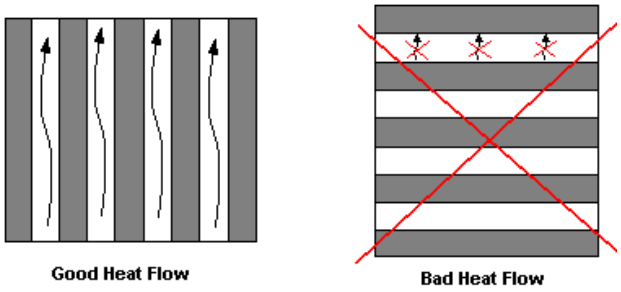


Figure 4.7

7. Getting Started

Typical Setup

PC-Controlled

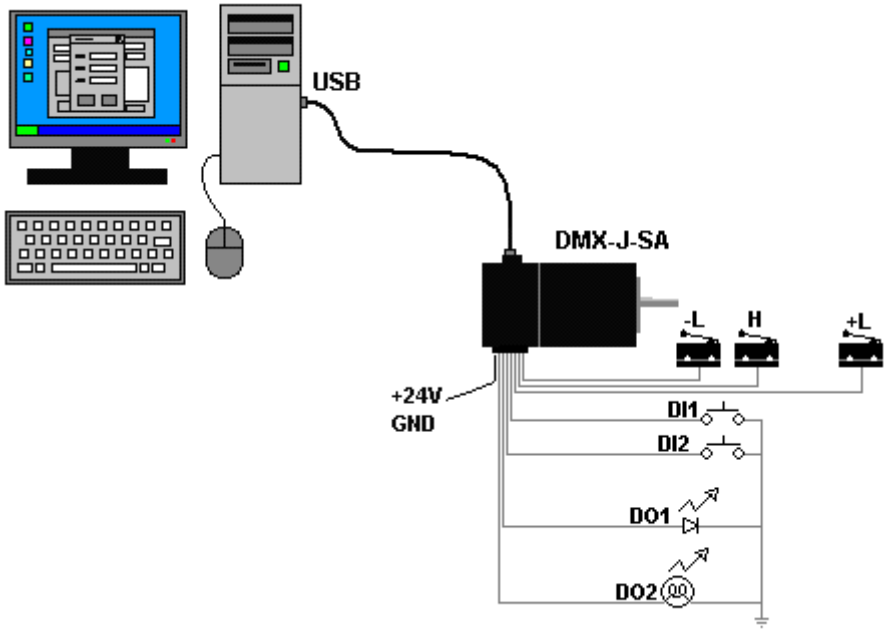


Figure 7.1

Stand-alone Operation

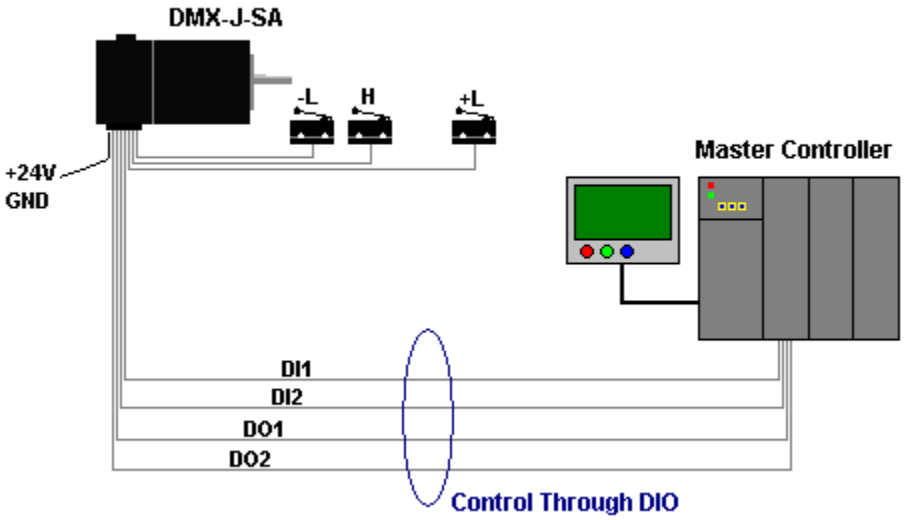


Figure 7.2

Windows GUI

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

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

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

Main Control Screen

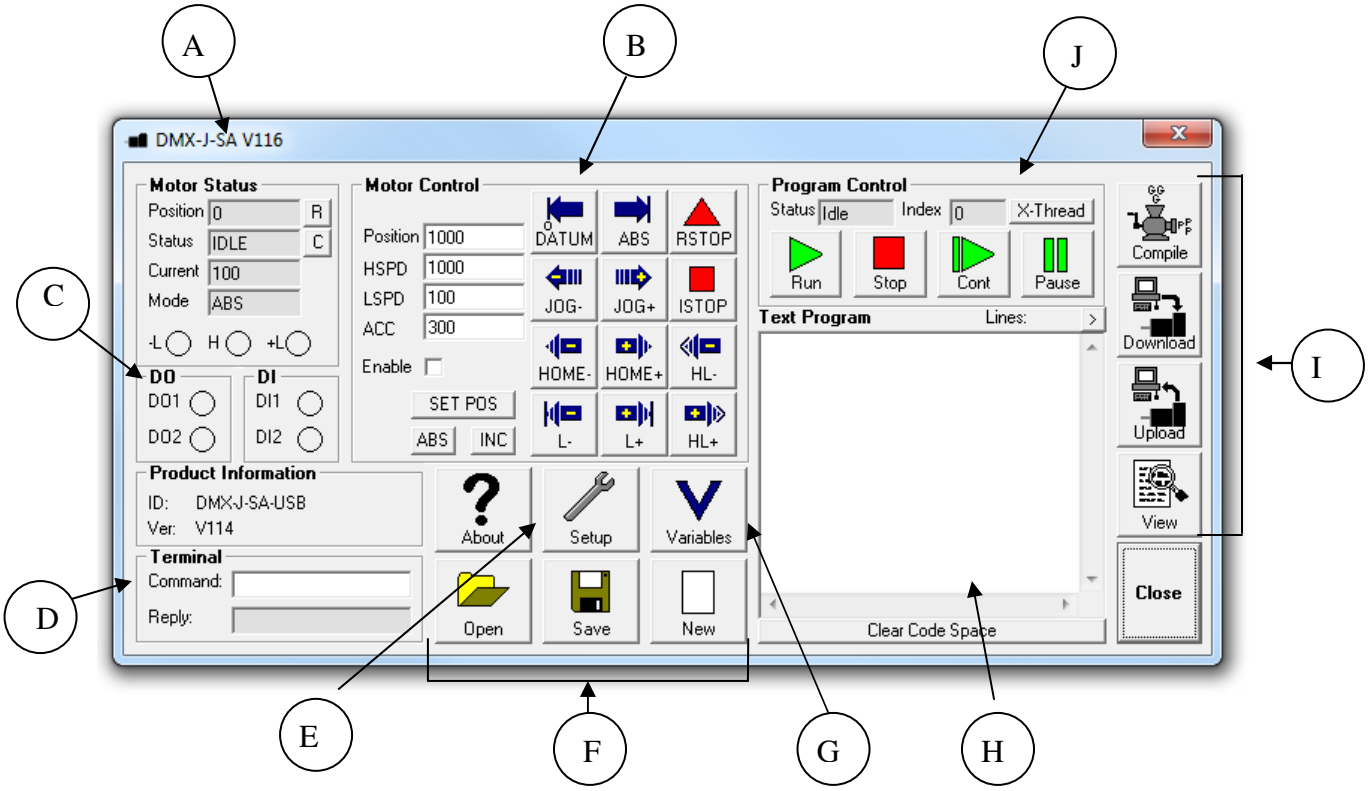


Figure 7.3

A. Motor Status

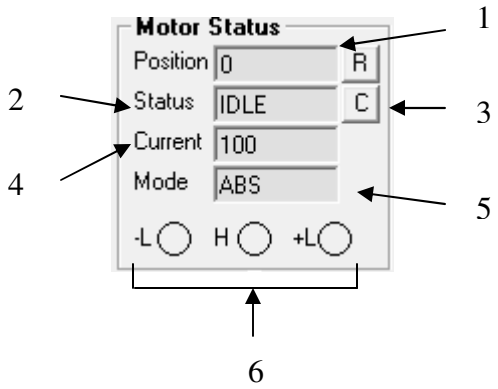


Figure 7.4

1. Position – Display of current motor position value
2. 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
3. Clear Error – Clear Error button is used to clear the Limit error status.
4. Current – Display of driver current.
5. Mode – Displays the move mode of the motor (**ABS/INC)**.
6. Limit and Home Input Status – Display of limits and home input status.

B. Motor Control

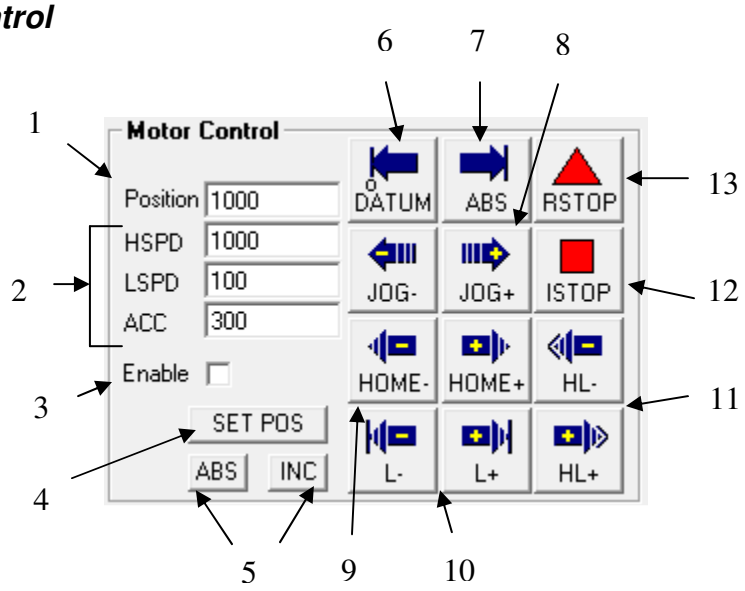


Figure 7.5

1. **Target Position –** Target position to move to for targeted position move.
2. **Speed Parameters** – Set the high speed, low speed, and acceleration values used during movement.
3. **Enable** – Enable the motor.
4. **Set Position –** Set Position button. Sets the position counter to the Position value.
5. **ABS/INC** – Set the move mode of the controller.
6. **DATUM** – Absolute move to zero position. Maximum delta from current position to target is 262,143. If greater, then move will not perform.
7. **ABS** – Absolute move to target position. Maximum delta from current position to target is 262,143. If greater, then the move will not perform.
8. **JOG+/-** - Jog the motor in the specified direction
9. **HOME+/- -** Home the motor in the specified direction
10. **L+/-** - Limit home the motor in the specified direction
11. **HL+/- -** Home the motor using high speed and low speed in the specified direction.
12. **ISTOP -** Immediate stop without deceleration
13. **RSTOP** – Stop with deceleration

C. DI Status/DO Status/Enable

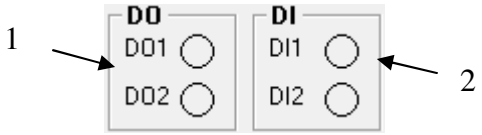


Figure 7.6

1. **Digital Output Status** – Display of the two digital output status. Digital output can be toggled by clicking on the circle.
2. **Digital Input Status –** Display of the two digital input bits. If the digital input pin is grounded, the digital input is turned on.

D. Terminal

Figure 7.7

Interactive ASCII commands can be sent and replies can be received. See interactive commands for details.

E. Setup

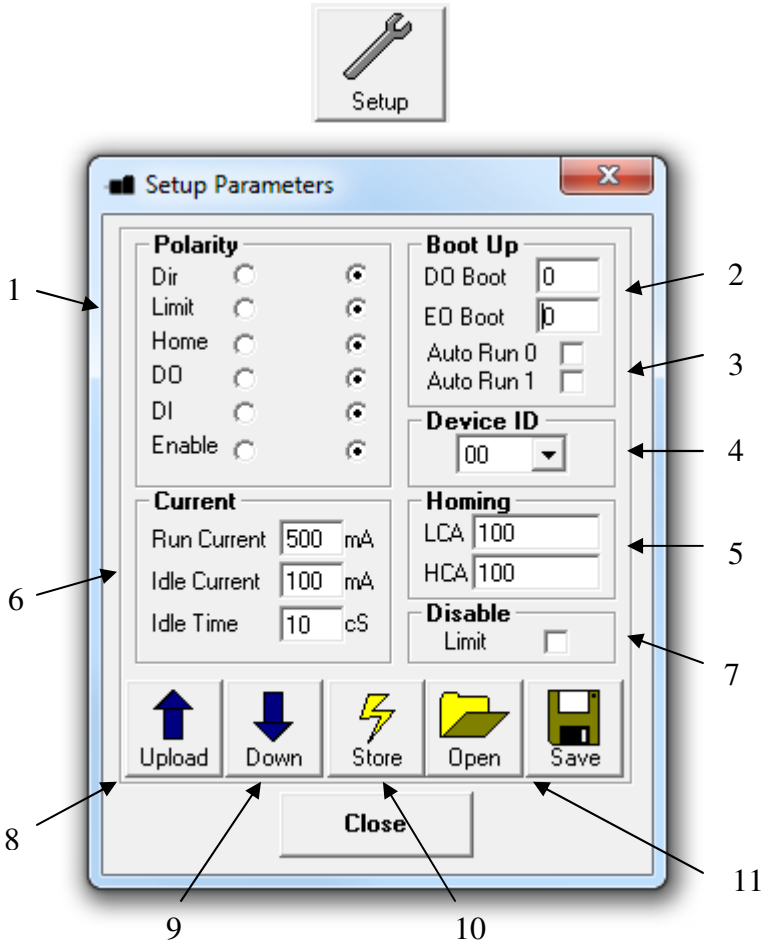


Figure 7.8

1. **Polarity** - Change the polarity of the inputs and outputs of the DMX-J-SA.
2. **DOBOOT/EOBOOT** - Change the boot-up configuration of the digital and enable outputs.
3. **Auto Run** - Run stand-alone programs on boot-up.
4. **Device Name** - Set the Device Name [JSA00-JSA99].
5. **LCA/HCA** - Set the Limit/Home Correction Amount used for their respective homing routines. Refer to the Homing Overview in Section 8.
6. **Current** - Change the Run Current, Idle Current, and Idle Time.
7. **Disable Limit** - Disable the limit inputs for general purpose use.
8. **Upload -** Upload the settings currently on the DMX-J-SA.
9. **Down** - Download the current settings.
10. **Store -** Store the settings to flash memory.
11. **Open/Save** - Parameters can be saved to a file and read from a file.

F. Standalone Program File Management

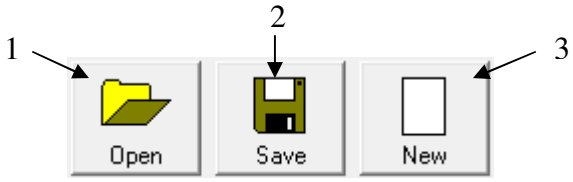


Figure 7.9

1. **Open -** Open a stand-alone program.
2. **Save** - Save a stand-alone program.
3. **New** - Clear the stand-alone program editor.

G. Variable Status

View the status of variables 0-49. Note that this window is read-only.

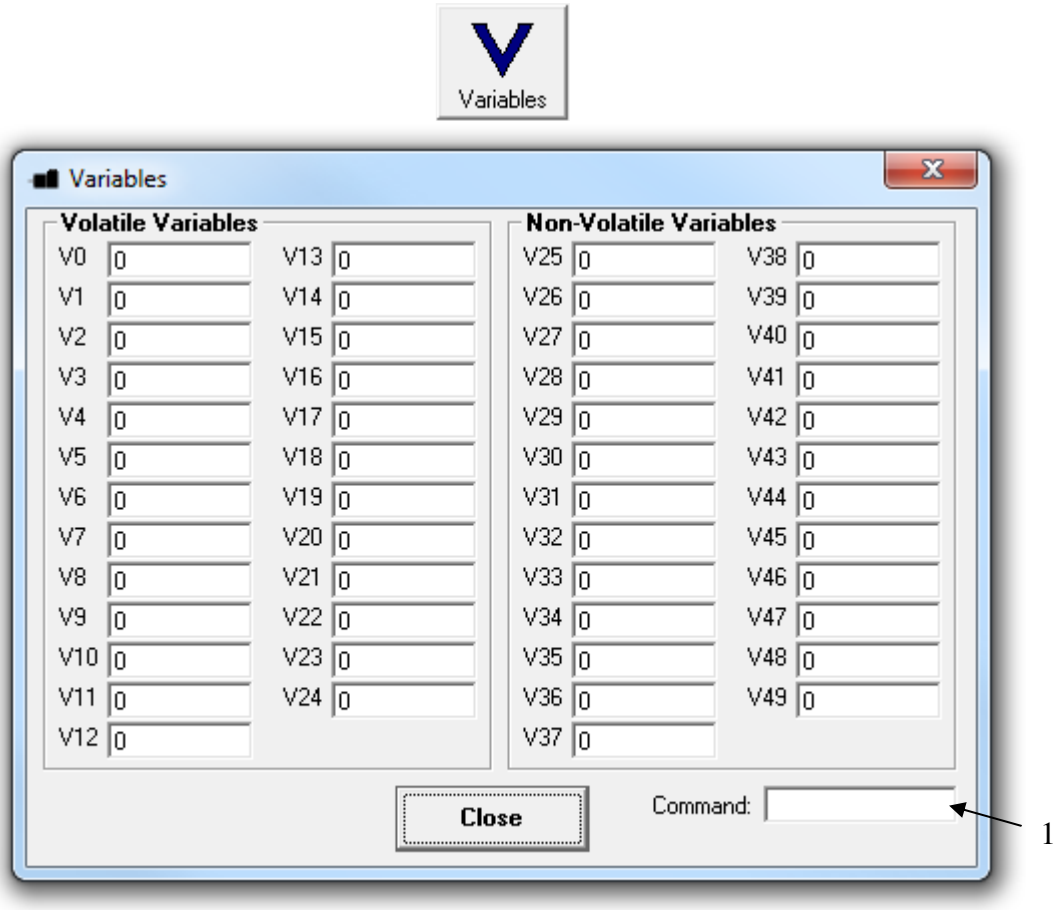


Figure 7.10

1. **Command Line** - To write to a variable, use V[0-49]=[value] syntax.

H. Standalone Program Editor

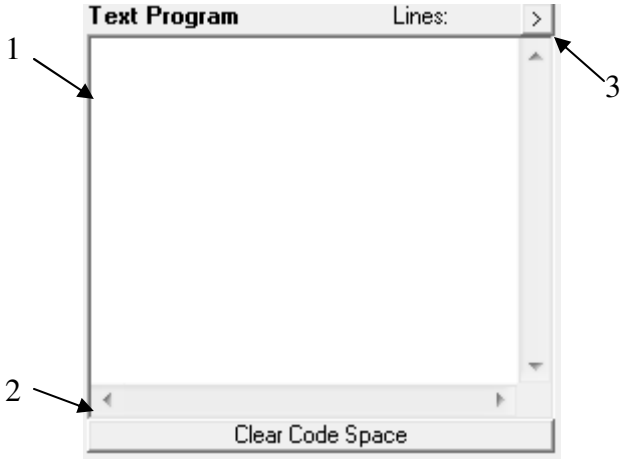


Figure 7.11

1. Write the standalone program in the Program Editor.
2. **Clear Code Space** - Use this button to remove the current standalone program that is stored on the DMX-J-SA.
3. Use this button to open an larger and easier to manage program editor.

I. Standalone Program Compile/Download/Upload/View

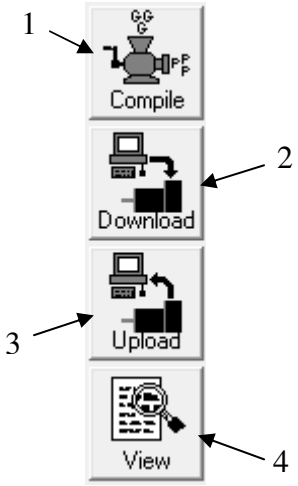


Figure 7.12

1. **Compile -** Compile the standalone program.
2. **Download -** Download the standalone program to flash memory.
3. **Upload -** Upload the standalone program from the controller.
4. **View -** View the low level compiled program.

J. Program Control

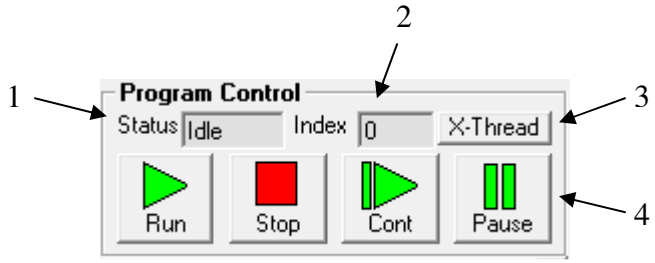


Figure 7.13

1. **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.
2. **Program Index** – display of low level program index. This is the index of the low level program index.
3. **X-Thread -** Open the Program Control for standalone multi-thread operation.
4. **Program Control** – Program can be RUN, STOP, PAUSED, and CONTINUED.

8. Motion Control Overview

Note: All commands described in this section are interactive commands only. Refer to the "Standalone Language Specification" section for the stand-alone code API.

Motion Profile and Speed

DMX-J-SA incorporates trapezoidal velocity profile as shown in Figure 8.0.

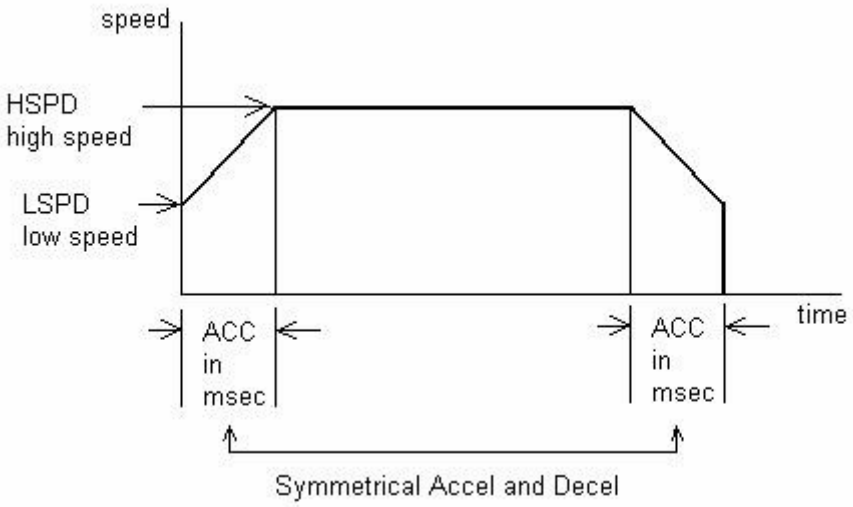


Figure 8.0

High speed and low speed are in pps(pulses/second). Use the **HSPD** and **LSPD** command to set/get the high speed and low speed settings. 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 translates to 3200 pulses/rev. To convert rotational speed to pulse speed, use the following formula.

Pulse/Sec = RPS * 53.33

Acceleration and deceleration time is in milliseconds and are symmetrical. Use the **ACC** command to set/get the acceleration/deceleration setting. Acceleration range is from 10msec to 1000msec.

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. Get the current position by using the **PX** command.

Note: If a motion command is sent while the controller is already moving, the command is not processed. Instead, an error response is returned. See Section 10 for further information on error responses.

Target Move

Target move, also known as absolute move, is used to move the motor to the desired position from the current position.

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).

DMX-J-SA can operate in either incremental or absolute move modes. Use the **X** command to make moves. Use the **INC** and **ABS** commands to set the move mode. Use the **MM** command to read the current move mode.

Homing

Home search sequence involves moving the motor towards the home or limit switches and then stopping when the relative input is detected. The DMX-J-SA has three different homing routines.

Home Input Only (High speed only)

Use the **H+/H-** commands. Figure 8.1 shows the homing routine.

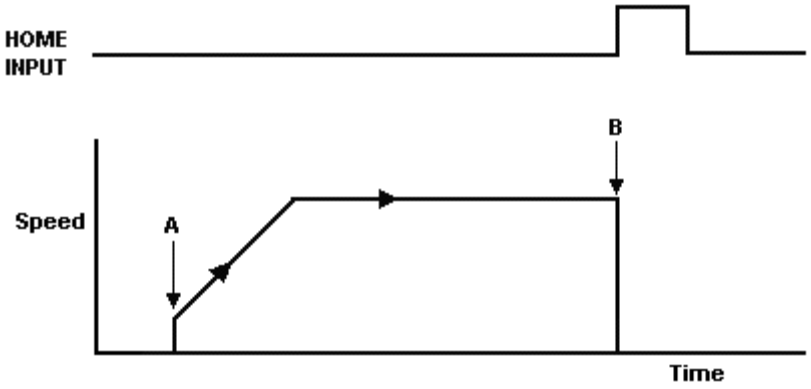


Figure 8.1

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

Home Input Only (High speed and low)

Use the **HL+/HL-** command. Figure 8.2 shows the homing routine.

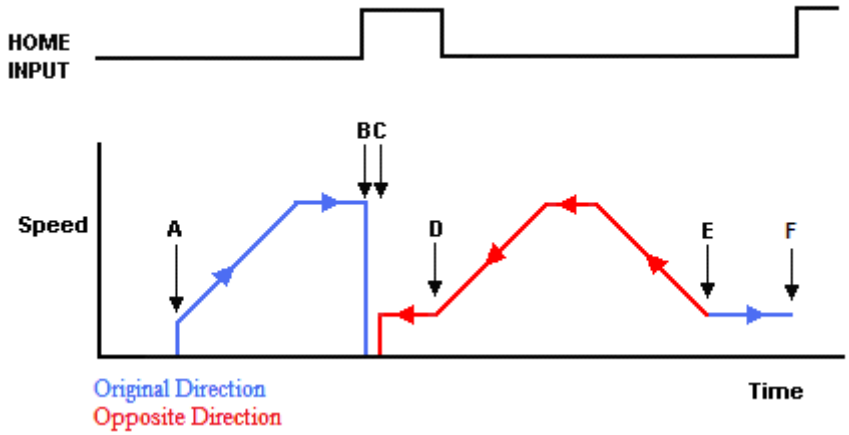


Figure 8.2

A. Starts the motor from low speed and accelerates to high speed.
B. As soon as the home input is triggered, the position counter is reset to zero and the motor immediately slows down to low speed.
C. The motor reverses direction to search for the home switch.
D. Once the home switch is reached, it will continue past the home switch by the amount defined by the home correction amount (**HCA**) at high speed.
E. The motor is now past the home input by the amount defined by the home correction amount (**HCA**). The motor now moves back towards the home switch at low speed.
F. The home input is triggered again, the position counter is reset to zero and the motor stops immediately

Limit Only

Use the **L+/L-** command. Figure 8.3 shows the homing routine.

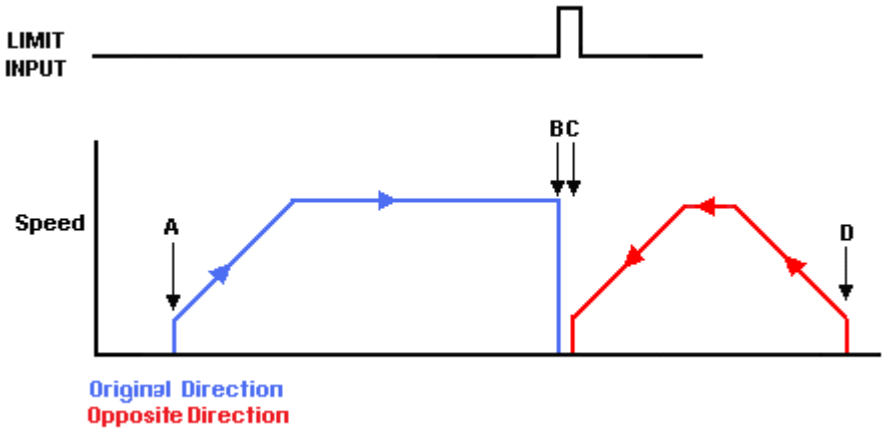


Figure 8.3

A. Issuing a limit home command starts the motor from low speed and accelerates to high speed.
B. As soon as the relevant limit input is triggered, the motor position is set to the limit correction amount (**LCA**) and the motor immediately reverses direction and returns to the zero position.
C. The zero position is reached and the motor immediately stops.

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

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

Jog Move

Jog move is used to continuously move the motor without stopping. Use the **J+/J-** command.

Stopping Motor

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

Limit Switch Function

With the limit switch function enabled, triggering of the limit switch in motion will stop the motion immediately depending on the direction of the motion. If the positive limit switch is triggered while moving in the positive direction, the motor will immediately stop and the motor status bit for positive limit error is set. The same is for the negative limit while moving in the negative direction.

Once limit error is set, the status must be cleared (using the **CLR** command) in order to move again.

The 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. Digital input assignments are: **DI3** for –Limit and **DI5** for +Limit.

Communication Time-out Feature (Watchdog)

ACE-SXC allows for the user to trigger an alarm if the master has not communicated with the device for a set period of time. When an alarm is triggered, bit 10 of the **MST** parameter is turned on. The time-out value is set by the **TOC** command. Units are in milliseconds. This feature is usually used in stand-alone mode. Refer to the **Example Stand-alone Programs** section for an example.

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

Motor Status

Motor status can be read anytime using the **MST** command. Table 8.0 shows 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 the minus limit is hit during negative direction motion. This error must be cleared before issuing any subsequent move commands (**CLR** command).
7	Plus limit error. This bit is latched when the plus limit is hit during positive direction motion. This error must be cleared before issuing any subsequent move commands (**CLR** command).
10	Communication timeout counter alarm

Table 8.0

Digital Inputs

DMX-J-SA module comes with 2 digital inputs. If the limit function is disabled (**DL**), up to 5 inputs can be used as general purpose inputs

Read digital input status using the **DI** command. See Table 8.1 for description.

Bit	Description	Bit-Wise Command
0	Digital Input 1	DI1
1	Digital Input 2	DI2
2	Negative Limit	DI3
3	Home	DI4
4	Positive Limit	DI5

Table 8.1

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 Table 8.2 for description.

Bit	Description	Bit-Wise Commands
0	Digital Output 1	DO1
1	Digital Output 2	DO2

Table 8.2

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).

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

Digital outputs are active low.

Motor Power

Using the **EO** command, the motor power can be enabled or disabled. The enable output does not affect the move command.

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

Polarity

Using **POL** command, polarity of following signals in Table 8.3 can be configured.

Bit	Description
0	Direction
1	Limit
2	Home
3	Digital Output
4	Digital Input
5	Enable Output
6	Jump to line 0 on error†

Table 8.3

†Used for error handling within standalone operation. If this bit is on, the line that is executed after SUB31 is called will be line 0. Otherwise, it will be the line that caused the error.

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.

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.

By default, the device name is set to: **JSA00**

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.

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

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

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

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

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

V1=100

Multi-Threading: The DMX-J-SA supports the simultaneous execution of two standalone programs. Program 0 is controlled via the **SR0** command and program 1 is controlled via the **SR1** command. For examples of multi-threading, please refer to the **Example Stand-alone Programs** section.

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

Error Handling: If an error occurs during standalone execution (i.e. limit error), the program automatically jumps to SUB 31. If SUB 31 is NOT defined, the program will cease execution and go to error state. If SUB 31 is defined by the user, the code within SUB 31 will be executed. The return jump line will be determined by bit 12 of the **POL** register. See Table 8.3 for details.

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 DMX-J-SA will return with an error.

Standalone Run on Boot-Up: Standalone can be configured to run on boot-up using the **SLOAD** command. See Table 8.4 for description.

Bit	Description
0	Standalone Program 0
1	Standalone Program 1

Table 8.4

Storing to Flash

The following items are stored to flash:

ASCII Command	Description
DN	Device Name
POL	Polarity Settings
CUR	Run Current
ACR	Idle Current
ICN	Idle Time
DOBOOT	DO configuration at boot-up
EOBOOT	EO configuration at boot-up
LCA	Limit Correction Amount
HCA	Home Correction Amount
DL	Disable Limit setting
SLOAD	Standalone program run on boot-up

	parameter
TOC	Time-out counter reset value
V25-V49	Note that on boot-up, V1-V25 are reset to value 0

Table 8.5

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

9. Communication

DMX-J-SA communication is USB 2.0 compliant.

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

The typical time needed for a send/reply transaction using the **fnPerformaxComSendRecv**() API function is in the single digit milliseconds range. This value will vary slightly with CPU speed of the PC and the type of ASCII command issued.

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.

BOOL **fnPerformaxComFlush**(IN HANDLE pHandle)

- Flushes the communication buffer on the PC as well as the USB controller. It is recommended to perform this operation right after the communication handle is opened.

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) **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

2) **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:

Figure 9.1

10. ASCII Language Specification

Note: All the commands described in this section are all interactive commands only. Please refer to the "Standalone Language Specification" section for stand-alone code API.

DMX-J-SA language is case sensitive. All commands should be in capital letters. Invalid commands are returned with "?". Always check for proper reply when command is sent.

Command	Description	Return
ABORT	Immediately stop the motor if in motion.	**OK**
ABS	Set the move mode to absolute mode.	**OK**
ACC	Return current acceleration value in milliseconds.	**[10-1,000]ms**
ACC=[Value]	Set the acceleration value. [10-1000]ms Example: ACC=300	**OK**
ACR	Return the idle current of the motor.	**[100-2,000]mA**
ACR=[Value]	Set the idle current of the motor. [100-2000]mA Example: ACR=500	**OK**
CCR	Return the actual current of the motor.	**[100-2,000]mA**
CLR	Clear limit error	**OK**
CUR	Return the run current of the motor. [100-2000]mA	**[100-2,000]mA**
CUR=[Value]	Set the run current of the motor. [100-2000]mA Example: CUR=1000	**OK**
DI	Return the status of digital inputs.	**Refer to Table 8.1**
DI[1-5]	Return the individual status of digital status.	**Refer to Table 8.1**
DL	Return the disable limit function. 0 - limit function is enabled 1 - limit function is disabled	**[0,1]**
DL=[Value]	Set the disable limit function 0 - limit function is enabled 1 - limit function is disabled	**OK**
DO	Return the status of digital outputs.	**Refer to Table 8.2**
DO=[Value]	Set the digital outputs.	**OK**
DO[1-2]	Return the status of the individual digital output.	**Refer to Table 8.2**
DO[1-2] =[Value]	Set the individual digital output. Refer to Table 8.2	**OK**
DOBOOT	Return the DO configuration at boot-up	**[0-3]**
DOBOOT=[Value]	Set the DO configuration at boot-up	**OK**
EO	Return the driver enable status 0 - Motor power enabled 1 - Motor power disabled	**[0,1]**
EO=[0,1]	Enable (1) or disable (0) the motor power	**OK**
DN	Get device name	[JSA00-JSA99]
DN=[Value]	Set device name	OK
EOBOOT	Return the EO configuration at boot-up	**[0,1]**
EOBOOT=[Value]	Set the EO configuration at boot-up	**OK**
GS[0-31]	Call a subroutine that has been previously stored to flash memory.	**OK**
H+	Home the motor in positive direction	**OK**
H-	Home the motor in negative direction	**OK**

HCA	Return the home correction amount.	**[0-262,143]**
HCA=[Value]	Set the home correction amount. This value cannot be greater than 262,143.	**OK**
HL+	Home the motor at low and high speed in the positive direction.	**OK**
HL-	Home the motor at low and high speed in the negative direction.	**OK**
HSPD	Return the high speed setting	**[100-200,000]pps**
HSPD=[Value]	Set the high speed setting. [100-200000]pps	**OK**
ICN	Return the idle time in centiseconds.	**[1-100]cs**
ICN=[Value]	Set the idle time. [1-100]cs Example: ICN=10	**OK**
ID	Return the product ID	**DMX-J-SA-USB**
INC	Set the move mode to incremental mode	**OK**
J+	Jog the motor in the positive direction	**OK**
J-	Jog the motor in the negative direction	**OK**
L+	Limit home the motor in the positive direction.	**OK**
L-	Limit home the motor in the negative direction.	**OK**
LCA	Return the limit correction amount.	**[0-262,143]**
LCA=[Value]	Set the limit correction amount. This value cannot be greater than 262,143.	**OK**
LSPD	Return the low speed setting.	**[100-200000]pps**
LSPD=[Value]	Set the low speed setting. [100-200000]pps	**OK**
MM	Return the current move mode 0 – ABS mode 1 – INC mode	**[0,1]**
MST	Return the motor status	**Refer to Table 8.0**
POL	Return the current polarity.	**Refer to Table 8.3**
POL=[Value]	Set the polarity. Refer to Table 8.3.	**OK**
PX	Return the current position value.	**32-bit number**
PX=[Value]	Set the current position value.	**OK**
SASTAT[0,1]	Return the standalone program status 0 – Idle 1 – Running 2 – Paused 4 - In Error	**[0-4]**
SA[Line]	Return the standalone line. Line: [0-1274]	
SA[Line]=[Value]	Set the standalone line. Line: [0-1274]	**OK**
SLOAD	Return the RunOnBoot parameter.	**[0-3]**
SLOAD=[Value]	Bit 0 - Stand-alone program 0 0 - Do NOT run stand-alone program on boot up 1 - Run stand-alone program on boot up Bit 1 - Stand-alone program on boot up	**OK**
SPC[0,1]	Return the program counter for the specified stand-alone program	**[0-1274]**
SR[0,1]=[Value]	Control the stand-alone program: 0 - Stop stand-alone program 1 - Run stand-alone program 2 - Pause stand-alone program	**OK**

	3 - Continue stand-alone program	
STOP	Stop all motion using deceleration.	**OK**
STORE	Store settings to flash. Refer to Table 8.5.	**OK**
TOC	Return the communication time-out parameter. Value is in milliseconds.	**32-bit number**
TOC=[Value]	Set the communication time-out parameter.	
V[0-49]	Read variables 0-49.	**32-bit number**
V[0-49]=[Value]	Set variables 0-49.	**OK**
VER	Get the firmware version.	**VXXX**
X[Value]	Move the motor to absolute position value using the HSPD, LSPD, and ACC values. Maximum allowed incremental move amount is 262143. for example, if current position is 100000, target move must be between 362143 and -162143.	**OK**

Table 10.0

Error Codes

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

Error Code	Description
?[Command]	The ASCII command is not understood by the ACE-SXC
?LimErrored	Motor is in limit error state
?Moving	A move or position change command is sent while the ACE-SXC is outputting pulses.
?Overmove	An absolute or increment move is issued with a move amount greater than 262,143.
?State Error	A move command is issued while the controller is in error state.
?Current Range	The idle or run current is set to a value outside of [100-2000]mA.
?Index out of Range	The index for the command sent to the controller is not valid.
?Sub not Initialized	Call to a subroutine using the **GS** command is not valid because the specified subroutine has not been defined.

Table 10.1

11. 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
WAITX
X3000           ;***Move X axis to position 3000
WAITX
```

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[1-5]

Description:

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

Digital inputs are active high

Syntax:

Read: [variable] = DI[1-5]

Conditional: IF DI[1-5]=[variable]
ENDIF

IF DI[1-5]=[0 or 1]
ENDIF

Examples:

```
IF DI1=0
```

```
        DO=1            ;***If digital input 1 is triggered, set DO=1
    ENDIF
```

DN

Description:

Read: Gets the device name
Write: Sets the device name

Syntax:

Read: [variable] = DN
Write: DN=[value]
DN=[variable]
Conditional: IF DN=[variable]
ENDIF

IF DN=[value]
ENDIF

Examples:

```
DN=2            ;***Set device name to JSA02
```

DO

Description:

Read: Gets the digital output value
Write: Sets the digital output valueDMX-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
    WAITX
ELSE
    X-1000       ;***If V1 is not 1, then move to -1000
    WAITX
ENDIF
```

ELSEIF

Description:

Perform ELSEIF condition check as a part of the IF statement

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

- = Equal to

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

Examples:

```
IF V1=1
        X1000
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
WAITX
END
```

ENDIF

Description:

Indicates end of IF operation

Syntax:

```
ENDIF
```

Examples:

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

ENDSUB

Description:

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

Syntax:

ENDSUB

Examples:

```
GOSUB 1
END

SUB 1
    X0
    WAITX
    X1000
    WAITX
ENDSUB
```

ENDWHILE

Description:

Indicate end of WHILE loop

Syntax:

ENDWHILE

Examples:

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

EO

Description:

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

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
```

HLHOMEX[+ or -]

Description:

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

Syntax:

HLHOMEX[+ or -]

Examples:

```
HLHOMEX+  ;***Homes the motor at low speed in the positive direction
WAITX
```

HOMEX[+ or -]

Description:

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

Syntax:

HOMEX[+ or -]

Examples:

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

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)
WAITX
```

```
X2000                ;***Move axis to position 3000 (1000+2000)
WAITX
```

JOGX[+ or -]

Description:

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

Syntax:

```
JOGX[+ or -]
```

Examples:

```
JOGX+        ;***Jogs axis in positive direction
DELAY=1000
STOPX
WAITX
JOGX-        ;***Jogs axis in negative direction
DELAY=1000
STOPX
WAITX
```

LHOMEX[+ or -]

Description:

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

Syntax:

```
LHOMEX[+ or -]
```

Examples:

```
LHOMEX+    ;***Limit homes axis in positive direction
WAITX
```

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
```

SR[0,1]

Description:

Write: Set the stand-alone control for the specified stand-alone program

Syntax:

Write: SR[0,1] = [0-3]
SR[0,1] = [0-3]

Examples:

```
IF DI1=1            ; If digital input 1 is on
    SR0=0           ; Turn off standalone program 0
ENDIF
```

STOPX

Description:

Command: Stop all 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

Note: The STORE command can only be issued a maximum of 30 times during one program run. If this amount is exceeded the standalone program will go into error state.

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
```

TOC

Description:

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

Syntax:

TOC=[long value]

Examples:

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

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
WAITX
V10 = 1200      ;***Set variable 10 value to 1200
XV10            ;***Move axis to variable 10 value
WAITX
```

12. Example Standalone Programs

Standalone Example Program 1 – Single Thread

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

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

Standalone Example Program 2 – Single Thread

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

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

Standalone Example Program 3 – Single Thread

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

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

Standalone Example Program 4 – Single Thread

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

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

Standalone Example Program 5 – Single Thread

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

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

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

Standalone Example Program 6 – Single Thread

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

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

Standalone Example Program 7 – Multi Thread

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

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

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

Standalone Example Program 8 – Multi Thread

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

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

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

Contact Information

Arcus Technology, Inc.

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

www.arcustechnology.com