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
1.12 - 4th Release
1.13 - 5th Release

Firmware Compatibility:

†V303BL

†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 5

1.1. FEATURES 5
1.2. PART NUMBERING SCHEME 5

2. ELECTRICAL AND THERMAL SPECIFICATIONS 6

3. DIMENSIONS 7

4. CONNECTIVITY 8

4.1. 10-PIN CONNECTOR (2.0MM) 8
4.2. 4-PIN CONNECTOR (2.0MM) 9
4.3. USB CONNECTOR 9
4.4. DMX-J-SA INTERFACE CIRCUIT 9
4.5. DIGITAL INPUTS, LIMITS, AND HOME 10
4.6. DIGITAL OUTPUTS 11

5. STEPPER MOTOR DRIVER OVERVIEW 12

5.1. MICROSTEP 12
5.2. DRIVER CURRENT 12
5.3. OPERATING TEMPERATURE 12
5.4. STEPPER MOTOR SPECIFICATIONS 13
5.5. STEPPER MOTOR TORQUE 13

6. COMMUNICATION INTERFACE 15

6.1. USB COMMUNICATION 15
6.1.1. Typical USB Setup 15
6.1.2. USB Device Number 15
6.1.3. USB Communication API 16
6.1.4. USB Communication Issues 17
6.2. WINDOWS GUI 18

7. GENERAL OPERATION OVERVIEW 19

7.1. MOTION PROFILE AND SPEED 19
7.2. POSITION COUNTER 20
7.3. JOG MOVE 20
7.4. STOPPING MOTOR 20
7.5. POSITIONAL MOVES 20
7.6. HOMING 21
7.6.1. Home Input Only (High Speed Only) 21
7.6.2. Home Input Only (High Speed and Low Speed) 22
7.6.3. Limit Only 23
7.7. LIMIT SWITCH FUNCTION 23
7.8. MOTOR STATUS 24
7.9. DIGITAL INPUTS / OUTPUTS 24
7.9.1. Digital Inputs 24
7.9.2. Digital Outputs 25
7.10. MOTOR POWER 25
7.11. POLARITY 26
7.12. IDLE CURRENT AND RUN CURRENT 26
7.13. COMMUNICATION TIME-OUT FEATURE (WATCHDOG) 26
7.14. STANDALONE PROGRAM SPECIFICATION 27

7.14.1. Standalone Program Specification 27
7.14.2. Standalone Control 27
7.14.3. Standalone Status 27
7.14.4. Standalone Subroutines 28
7.14.5. Error Handling 28
7.14.6. Standalone Variables 28
7.14.7. Standalone Run on Boot-Up 29
7.15. STORING TO FLASH 30

8. SOFTWARE OVERVIEW 31

8.1. MAIN CONTROL SCREEN 32
8.1.1. Motor Status 32
8.1.2. Motor Control 33
8.1.3. DI Status/DO Status/Enable 34
8.1.4. Terminal 34
8.1.5. Product Information 34
8.1.6. Setup 35
8.1.7. Standalone Program File Management 36
8.1.8. Variable Status 36
8.1.9. Standalone Program Editor 37
8.1.10. Standalone Program Compile/Download/Upload/View 37
8.1.11. Program Control 38

9. ASCII LANGUAGE SPECIFICATION 39

9.1. ASCII COMMAND SET 39
9.2. ERROR CODES 41

10. STANDALONE PROGRAMMING SPECIFICATION 42

10.1. STANDALONE COMMAND SET 42
10.2. EXAMPLE STANDALONE PROGRAMS 44
10.2.1. Standalone Example Program 1 – Single Thread 44
10.2.2. Standalone Example Program 2 – Single Thread 44
10.2.3. Standalone Example Program 3 – Single Thread 44
10.2.4. Standalone Example Program 4 – Single Thread 45
10.2.5. Standalone Example Program 5 – Single Thread 45
10.2.6. Standalone Example Program 6 – Single Thread 46
10.2.7. Standalone Example Program 7 – Multi Thread 47
10.2.8. Standalone Example Program 8 – Multi Thread 48

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 motor and have it run independent of a host.

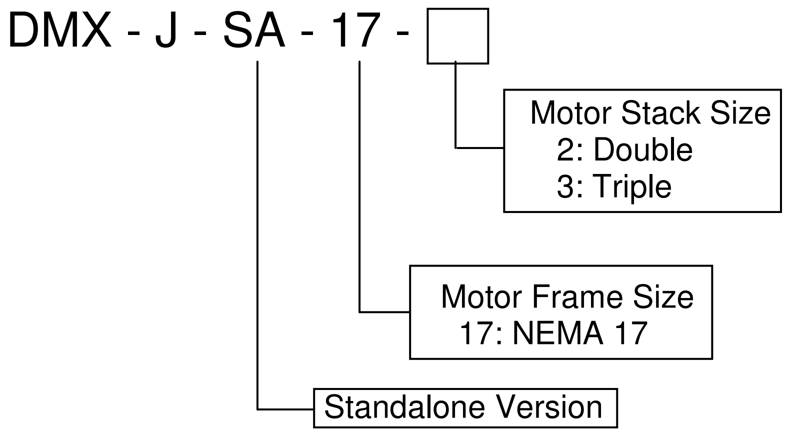
Windows and Linux drivers, as well as sample source code, are available to aid you in your software development.

1.1. Features
- USB 2.0 communication
- Standalone programming using A-SCRIPT
- NEMA 17 stepper motor
- 16 microstep driver (100mA to 2.0A peak current)
- Opto-isolated I/O
    - 2 x inputs
    - 2 x outputs
    - +Limit/-Limit/Home inputs
- Homing routines:
    - Home input only
    - Limit inputs only
- S-curve or trapezoidal motion profile

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

Or, contact your local distributor for technical support.

1.2. Part Numbering Scheme



2. Electrical and Thermal Specifications

Parameter	Min	Max	Units
Main Power Input	+12	+24	V
	-	2.0	A
Opto-supply Power Input	+12	+24	V
Digital Input Forward Diode Current	-	45	mA
Digital Output Collector Voltage	-	+24	V
Digital Output Sink Current	-	45	mA
Operating Temperature	0	+85	℃
Storage Temperature	-55	+150	℃

3. Dimensions

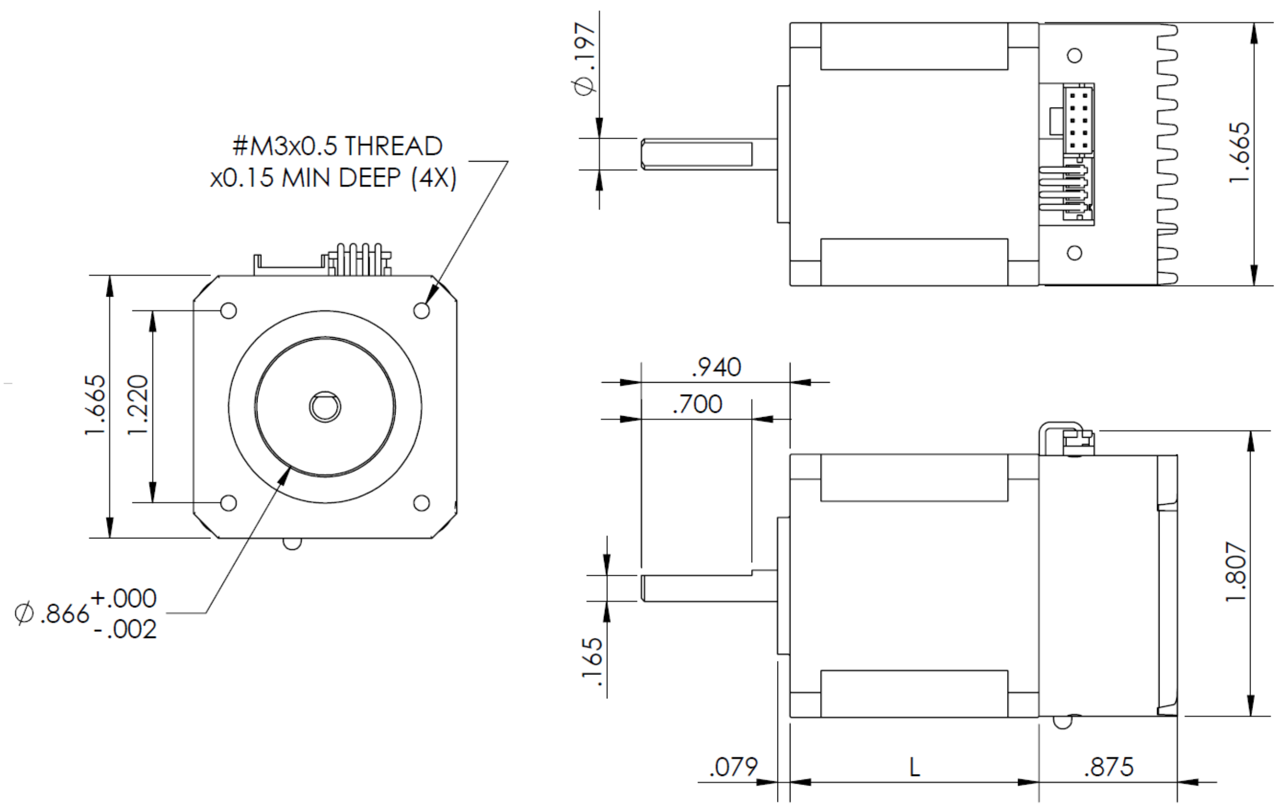


Figure 3.0

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

Table 3.0

† All dimensions in inches.

4. Connectivity

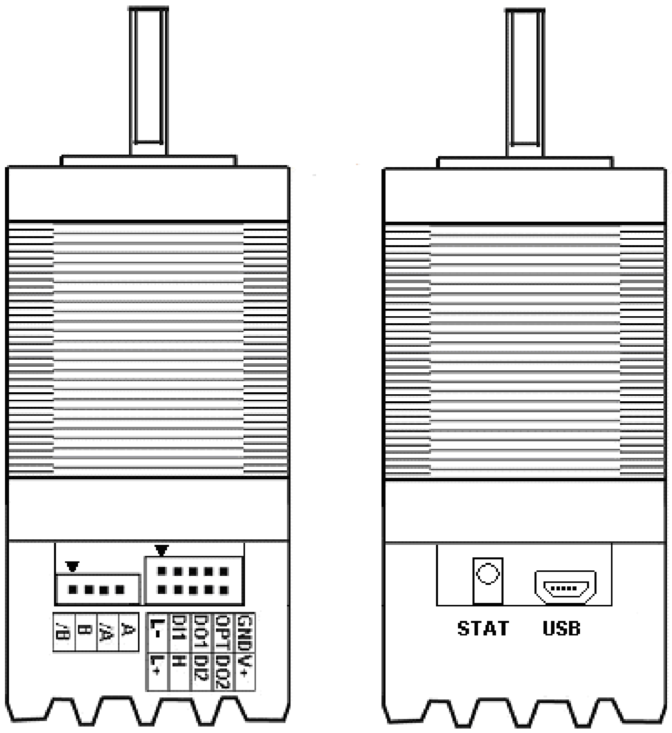


Figure 4.0

4.1. 10-Pin Connector (2.0mm)

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

Table 4.0

Mating Connector Description:	Female 10 pin 2mm dual row
Mating Connector Manufacturer:	HIROSE
Mating Connector Manufacturer Part:	DF11-10DS-2C

4.2. 4-Pin Connector (2.0mm)

Pin #	Wire Color	In/Out	Name	Description
1	Red	O	/B	Motor Winding /B
2	Red/White	O	B	Motor Winding B
3	Green	O	/A	Motor Winding /A
4	Green/White	O	A	Motor Winding A

Table 4.1

Mating Connector Description:	Female 10 pin 2mm single row
Mating Connector Manufacturer:	HIROSE
Mating Connector Manufacturer Part:	DF3-4S-2C

4.3. USB Connector

Type: Mini-B to A

4.4. DMX-J-SA Interface Circuit

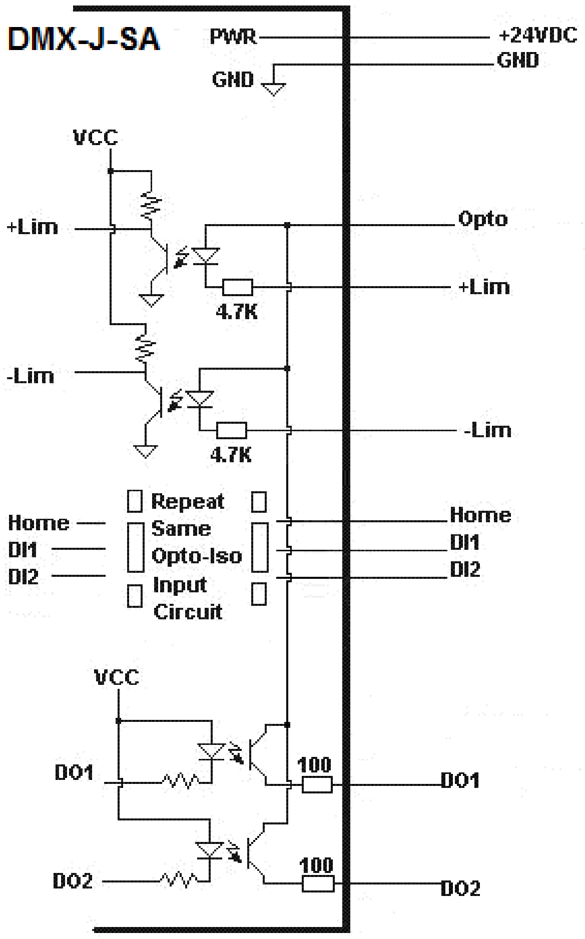


Figure 4.1

4.5. Digital Inputs, Limits, and Home

Figure 4.2 shows the detailed schematic of the opto-isolated limit, home, and general purpose inputs. All opto-isolated digital inputs are NPN type.

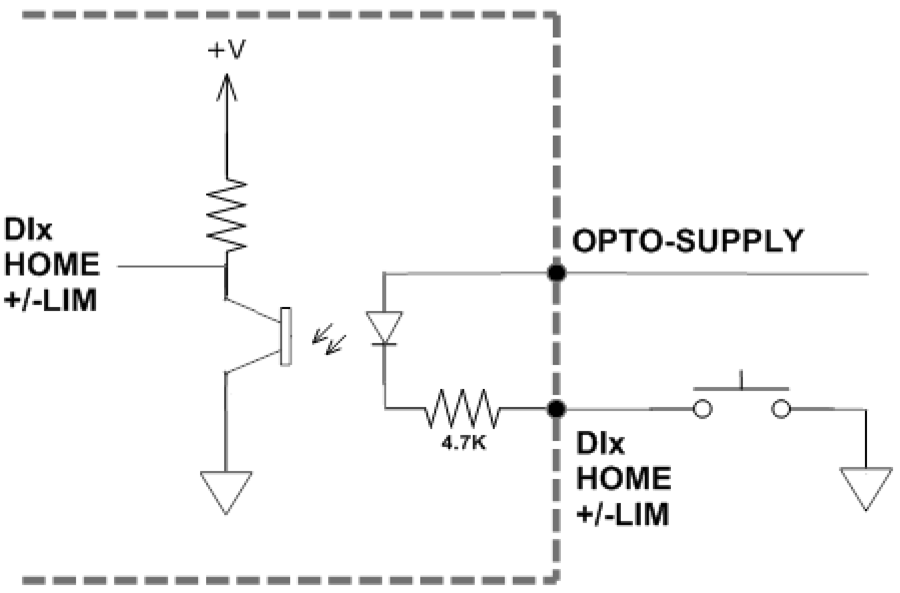


Figure 4.2

The opto-supply must be connected to +12 to +24VDC in order for the limit, home, digital inputs to operate.

When the digital input is pulled to ground, current will flow from the opto-supply to ground, turning on the opto-isolator and activating the digital input.

To de-activate the input, the digital input should be left unconnected or pulled up to the opto-supply, preventing current from flowing through the opto-isolator.

4.6. Digital Outputs

Figure 4.3 shows an example wiring of the digital outputs. All opto-isolated digital outputs will be PNP type.

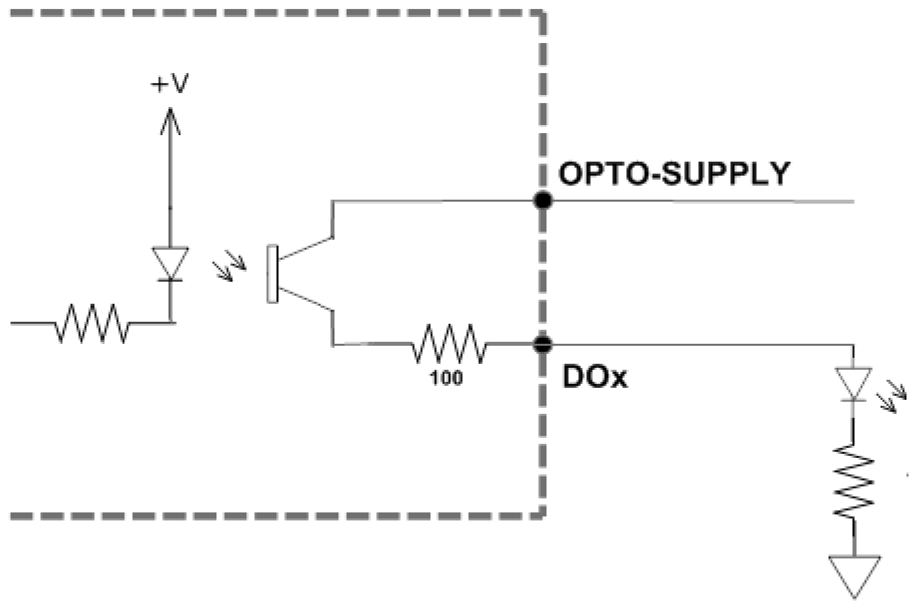


Figure 4.3

The opto-supply must be connected to +12 to +24VDC in order for the digital outputs to operate.

When activated, the opto-isolator for the digital output pulls the voltage on the digital output line to the opto-supply. The maximum sink current for digital outputs is 45mA. Take caution to select the appropriate external resistor so that the current does not exceed 45mA.

When deactivated, the opto-isolator will break the connection between the digital output and the opto-supply.

5. Stepper Motor Driver Overview

5.1. Microstep

The DMX-J-SA has a fixed microstepping of 16. With the standard 1.8° stepper motor included with the DMX-J-SA, this microstep setting will translate to 3200 steps per revolution.

The steps per revolution may change if a nonstandard motor is used.

5.2. Driver Current

The DMX-J-SA will have a maximum rated driver current that is dependent on the stack size of the motor. See table 5.0 for details.

Product	Max Rated Driver Current	Recommended Driver Current
DMX-J-SA-17-2	1.7 A	1.4 A
DMX-J-SA-17-3	2.0 A	1.6 A

Table 5.0

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

DMX-J-SA has configurable current setting from 100mA to 2.0A. Driver current is set to the "Run Current" setting whenever the motor is moving. Similarly, the driver current is set to the "Idle Current" setting when the motor is idle for a period of time longer than the "Idle Time" setting. See section 7.12 for more details regarding the available driver settings.

The Run Current and the Idle Current should not go over the maximum rated current for each motor size. Use table 5.0 as a reference on maximum rated current setting.

5.3. 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, the stepper motor section typically becomes hotter than the driver section. Having the stepper motor mounted to a heat sink will better dissipate the heat generated by the motor.

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

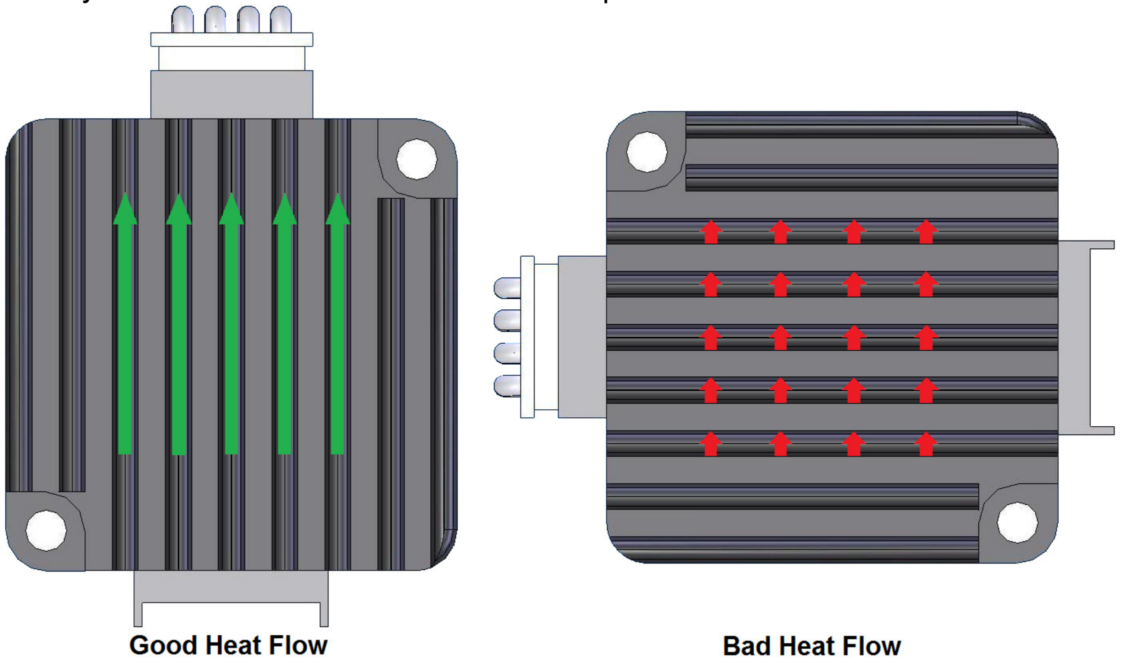


Figure 5.0

5.4. Stepper Motor Specifications

Table 5.1 details the stepper motor characteristics of the DMX-J-SA.

Stack	Max Current (RMS)	Resistance/Phase	Inductance/Phase	Inertia
Double	1.7A	1.5 Ohm	3.0 mH	0.28 oz-in^2
Triple	2.0A	1.4 Ohm	2.7 mH	0.37 oz-in^2

Table 5.1

Additionally, Table 5.2 details to max allowable forces the DMX-J-SA can tolerate on the motor shaft.

Stack Size	Max Axial Force	Max Radial Force
Double	15N	10N
Triple	15N	10N

Table 5.2

5.5. Stepper Motor Torque

The torque output of the DMX-J-SA will vary depending on the supply voltage, driver current, motor type, and target speed of the motor.

Increasing the drive current will increase the torque output, however the operating temperature will also increase. While decreasing the drive current will reduce the torque output, it will help reduce the operating temperature as well. Each application will need to adjust this setting to find the desired driver output.

Using a higher voltage to power the DMX-J-SA will allow the motor to run at faster speeds. Note that increasing the voltage will not increase the maximum holding torque of the motor.

Stepper motors in general will drop off in torque as the target speed of the motor is increased. The following torque curve shows the expected torque output based on the motor speed of the DMX-J-SA.

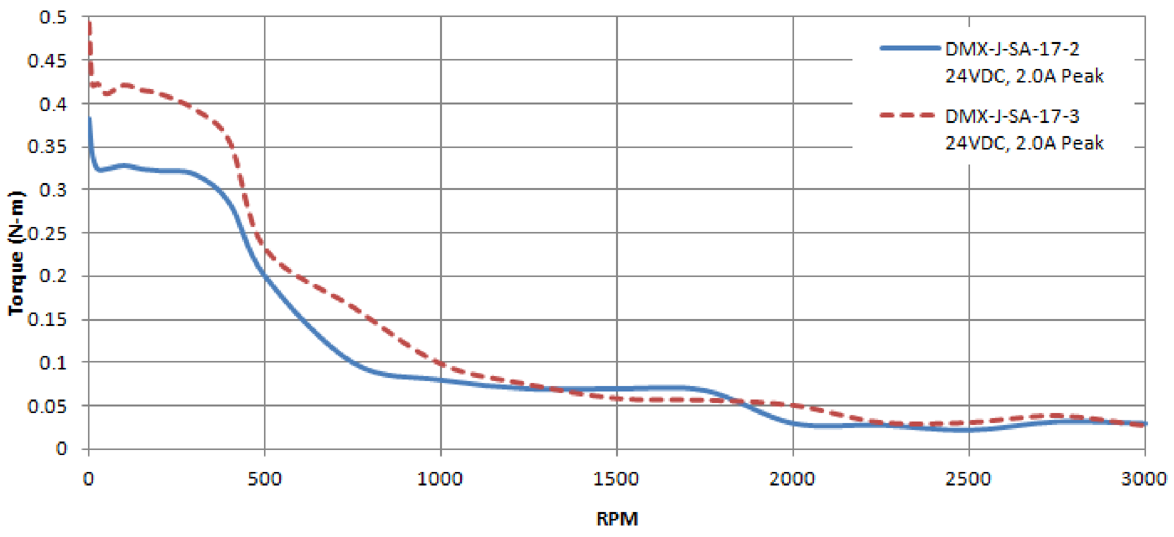


Figure 5.1

6. Communication Interface

6.1. USB Communication

DMX-J-SA USB communication is USB 2.0 compliant.

In order to communicate with DMX-J-SA via USB, the proper software driver must be first installed. Before connecting the DMX-J-SA motor or running any programs, please go to the Arcus web site, download the Arcus Drivers and Tools Setup, and run the installation.

All USB communication will be done using an ASCII command protocol.

6.1.1. Typical USB Setup

The DMX-J-SA can be connected to a PC directly via USB or through a USB hub. All USB cables should have a noise suppression choke to avoid communication loss or interruption. See a typical USB network setup in figure 6.0.

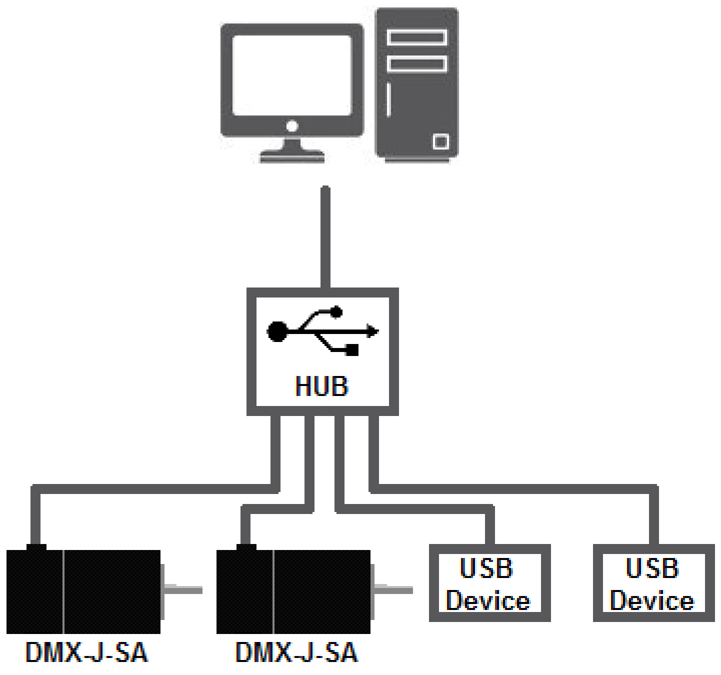


Figure 6.0

6.1.2. USB Device Number

If multiple DMX-J-SA devices are connected to the PC, each device should have a unique device number. This will allow the PC to differentiate between multiple motors. In order to make this change to a DMX-J-SA, first store the desired number using the DN command. Note that this value must be within the range [JSA00,JSA99].

For example, to change the device number, the command DN=JSA02 can be sent. The device name can also be changed through the *Setup* window of the standard DMX-J-SA software. See section 8 for details.

By default, all DMX-J-SA start with device number JSA00.

To save a modified device number to the DMX-J-SA's flash memory, use the STORE command. After a power cycle, the new device number will be used. Note that before a power cycle is completed, the settings will not take effect.

6.1.3. USB Communication API

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

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

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 and Performax USB modules connected to the PC.

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

- This function is used to get the Performax or 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. dwDeviceNum starts from 0.

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

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

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

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

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

- This function is used to send commands and receive replies. The 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.

6.1.4. 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 figure 6.1.

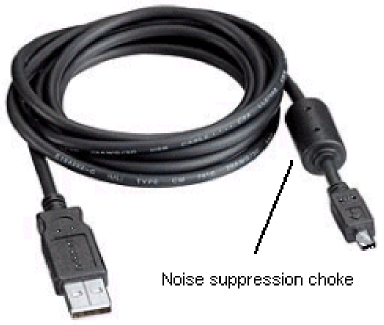


Figure 6.1

6.2. Windows GUI

DMX-J-SA comes with a Windows GUI program to test, program, compile, download, and debug the controller. The Windows GUI will perform all communication via USB. See section 9 for further details.

7. General Operation Overview

Important Note: All the commands described in this section are defined as ASCII or standalone commands. ASCII commands are used when communicating over USB. Standalone commands are used when writing a standalone program onto the DMX-J-SA.

7.1. Motion Profile and Speed

DMX-J-SA incorporates trapezoidal velocity profile as shown in Figure 7.0.

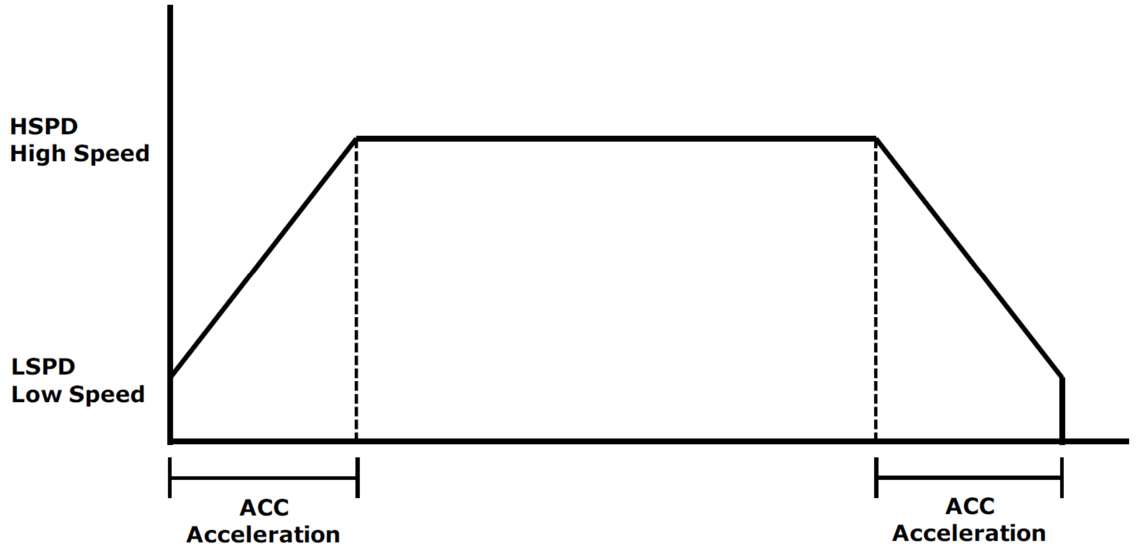


Figure 7.0

Once a typical move is issued, the motor will immediately start moving at the low speed setting and accelerate to the high speed. Once at high speed, the motor will move at a constant speed until it decelerates from high speed to low speed and immediately stops.

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, 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 * 3200 Pulse/Rev

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

ASCII	**HSPD**	**LSPD**	**ACC**
Standalone	**HSPD**	**LSPD**	**ACC**

7.2. 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.

The **PX** command can also be used to manually set the position of the motor. If the motor is moving while an attempt is made to set the position, an error will be returned and the position will remain unchanged.

ASCII	**PX**
Standalone	**PX**

7.3. Jog Move

A jog move is used to continuously move the motor without stopping. Use the **J+/J-** command when operating in ASCII mode and the **JOGX+/JOGX-** in standalone mode. Once this move is started, the motor will only stop if a limit input is activated during the move or a stop command is issued.

ASCII	**J[+/-]**
Standalone	**JOGX[+/-]**

7.4. Stopping Motor

When the motor is performing any type of move, 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 in ASCII mode and **ABORTX** in standalone. The ASCII command **STOP,** and standalone command **STOPX**, can be used to stop the motor with deceleration.

ASCII	**STOP**	**ABORT**
Standalone	**STOPX**	**ABORTX**

7.5. Positional Moves

Positional moves are used to move the motor to a desired position. The **X[target]** command should be used make positional moves

The maximum allowable difference between the target position and current position is 16,777,215. For example, if the current position counter is 1000, the maximum allowable target position will be between -16,776,215 (1,000-16,776,215) and 16,778,215 (1,000+16,776,215).

The DMX-J-SA also has the ability to move in an absolute or incremental mode. Absolute move mode will move the motor to the target position, while incremental move mode will increment the current position by the target position. The **INC** and **ABS** commands set the move mode. Use the **MM** command to read the current move mode. If the **MM** command returns 0, the motor is in absolute mode. A value of 1 will indicate the motor is in increment mode.

ASCII	**X[pos]**	**INC**	**ABS**	**MM**
Standalone	**X[pos]**	**INC**	**ABS**	**-**

7.6. Homing

Home search routines involve moving the motor and using the home or limit inputs to determine the zero reference position. Three different type of homing routines are available.

7.6.1. Home Input Only (High Speed Only)

Use the **H+/H-** command for ASCII mode or the **HOMEX+/-** command for standalone mode. Figure 7.1 shows the homing routine.

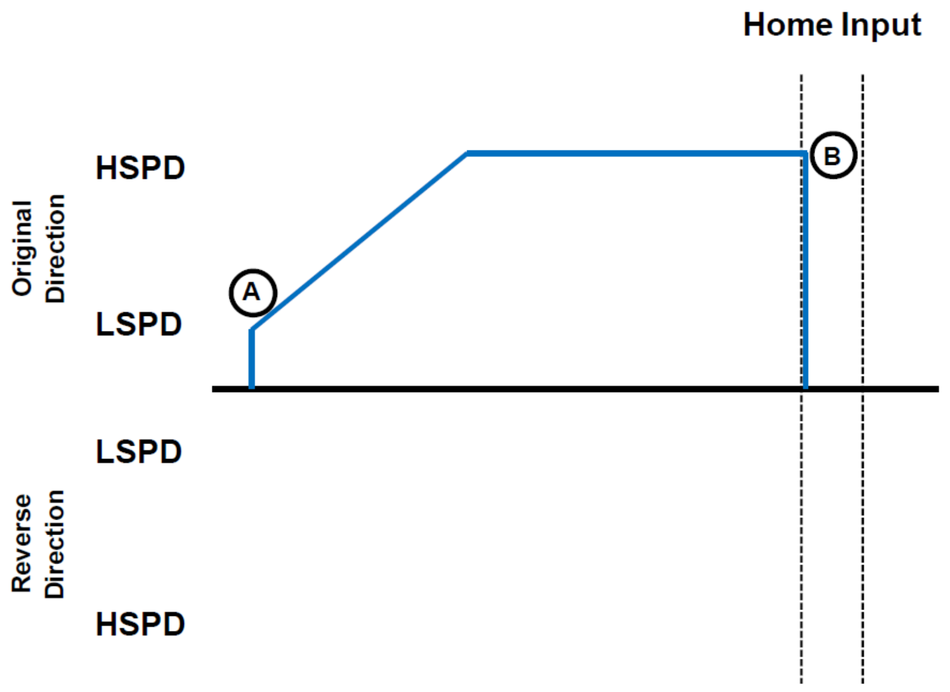


Figure 7.1

A. Starts the motor from low speed and accelerates to high speed in search of the home input.
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.

ASCII	**HX+/-**
Standalone	**HOMEX+/-**

7.6.2. Home Input Only (High Speed and Low Speed)

Use the **HL+**/**HL-** command for ASCII mode or the **HLHOMEX+/-** command for standalone mode. Figure 7.2 shows the homing routine.

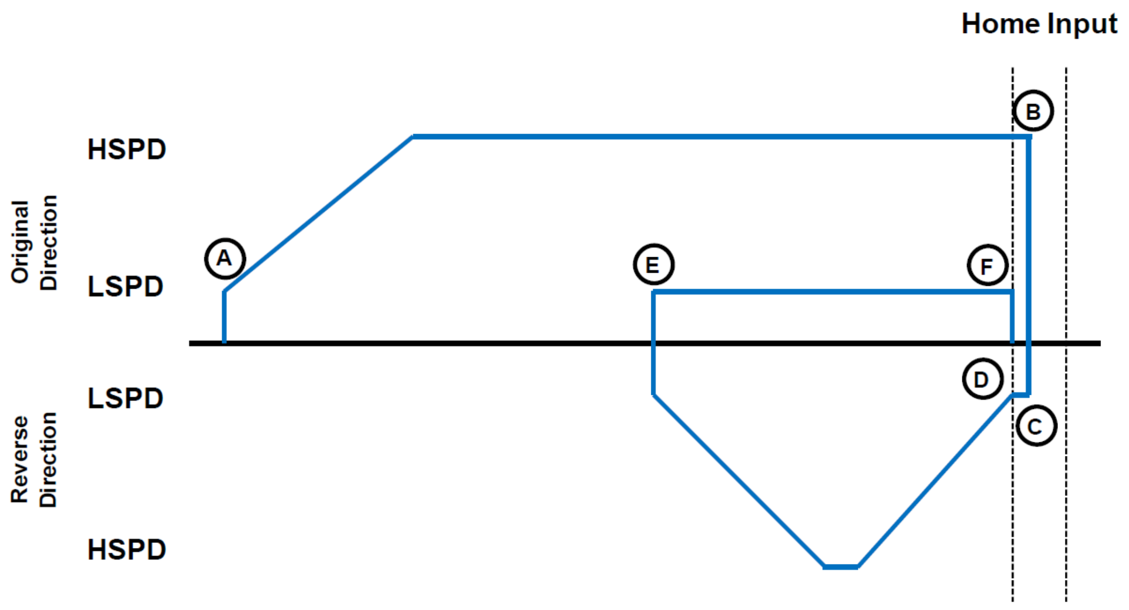


Figure 7.2

A. Starts the motor from low speed and accelerates to high speed in search of the home input.
B. As soon as the home input is triggered, the position counter is reset to zero and the motor immediately stops.
C. The motor moves at low speed in the reverse direction until the home input has been cleared.
D. Once the home input is cleared, the motor will continue to move in the reverse direction by the amount defined by the home correction amount (**HCA**). It will ramp up to high speed for this movement.
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 immediately stops.

ASCII	**HLX+/-**
Standalone	**HLHOMEX+/-**

7.6.3. Limit Only

Use the **L+**/**L-** command in ASCII mode or the **LHOMEX+/-** in standalone mode. Figure 7.3 shows the homing routine.

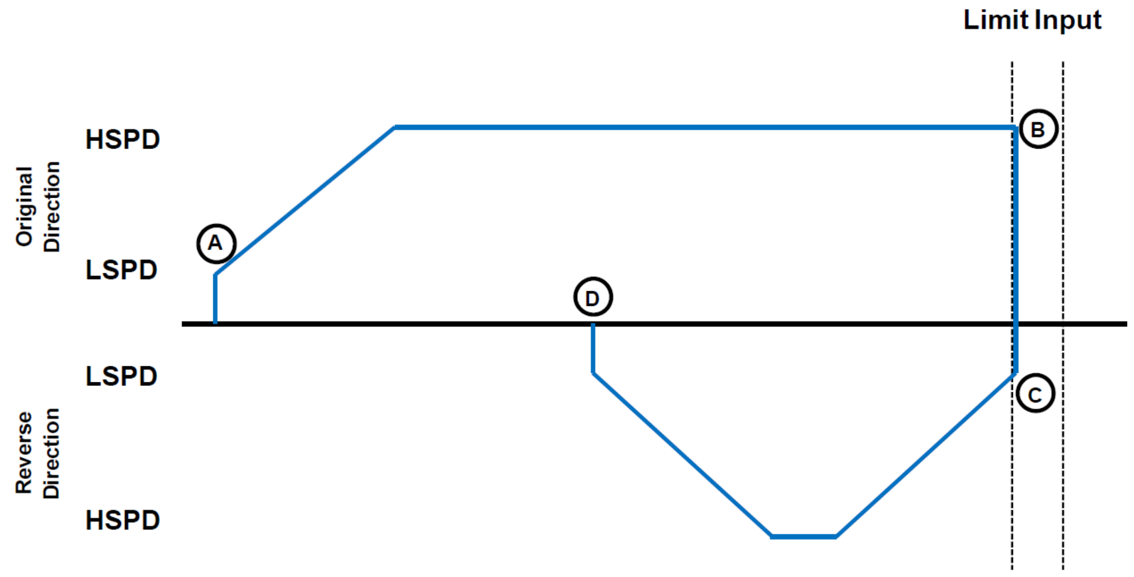


Figure 7.3

A. Starts the motor from low speed and accelerates to high speed in search of the specified limit input.
B. As soon as the relevant limit input is triggered, the motor immediately stops motion.
C. The motor position will be set to the limit correction amount (**LCA**). It will the move in the reverse direction at high speed.
D. Once the limit correction amount move is complete, the motor position will read zero.

Notes:
To trigger a home or limit input switch, supply the opto-supply voltage with 12VDC to 24VDC and connect the home or limit input signal to opto-supply ground.

If home and limit inputs are not used, it can also be used as a general purpose input. See section 7.1 for the digital input assignment.

7.7. Limit Switch Function

With the limit switch function enabled, triggering of the limit switch while the motor is moving will stop the motion immediately. For example, 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 will apply 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.

ASCII	**CLR**	**DL**
Standalone	**ECLEARX**	**-**

7.8. Motor Status

Motor status can be read anytime using the **MST** command. Table 7.0 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.
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.
10	Communication timeout counter alarm

Table 7.0

ASCII	**MST**
Standalone	**MSTX**

7.9. Digital Inputs / Outputs

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

7.9.1. Digital Inputs

The digital input status of all five available inputs can be read with the **DI** command.

Digital inputs values can also be referenced one bit at a time by using the **DI[1-5]** commands. Note that the indexes are 1-based for the bit references. For example, DI1 refers to bit 0, not bit 1. See Table 7.1 for details.

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 7.1

By default, if a digital input is on (i.e. input is pulled to GND of opto-supply), the bit status is 0. Otherwise, the bit status is 1. This can be changed using the polarity setting. See section 7.12 for details.

ASCII	**DI**	**DI[1-5]**
Standalone	**DI**	**DI[1-5]**

7.9.2. Digital Outputs

The **DO** command can be used to set the output voltage of both available digital outputs. The DO value must be within the range of 0-3.

Digital output values can also be referenced one bit at a time with the **DO[1-2]** commands. Note that the indexes are 1-based for the bit references. For example, DO1 refers to bit 0, not bit 1. See Table 7.2 for details.

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

Table 7.2

If digital output is turned on, the corresponding bit of the **DO** command is 1. Otherwise, the bit status is 0. The voltage level of the digital output when it is on or off is determined by the polarity setting. See section 7.12 for details.
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.

ASCII	**DO**	**DO[1-2]**	**DOBOOT**
Standalone	**DO**	**DO[1-2]**	**-**

7.10. Motor Power

The **EO** command can enable or disable the current to the motor. If a move command is issued to the DMX-J-SA while it is disabled, the move command will still process however the motor will not be able to physically move.

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.

ASCII	**EO**	**EOBOOT**
Standalone	**EO**	**-**

7.11. Polarity

The **POL** command can be used to configure the polarity of the signals in Table 7.3.

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 7.3

†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.

ASCII	**POL**
Standalone	**-**

7.12. Idle Current and Run Current

DMX-J-SA allows for two current settings. The run current is used while the motor is running. The **CUR** command can be used to set the run current. The idle current is used when the motor is idle and can be set using the **ACR** command. The run and idle current range must be within 0mA to 2000mA. A setting of 0mA will result in the motor disabling completely.

To set the amount of time the motor needs to be idle before changing from run current to idle current, use the **ICN** command. Units are in centi-seconds. The actual current of the motor can be read using the **CCR** command.

ASCII	**CUR**	**ACR**	**CCR**	**ICN**
Standalone	**-**	**-**	**-**	**-**

7.13. Communication Time-out Feature (Watchdog)

DMX-J-SA allows for the user to trigger an alarm if the master has not communicated with the device for a set period of time. When an alarm is triggered, bit 10 of the **MST** parameter is turned on. The time-out value is set by the **TOC** command. Units are in milliseconds. This feature will typically be used in standalone mode.

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

7.14. Standalone Program Specification

Standalone programming allows the controller to operate of a user defined program that is stored in the internal memory of the DMX-J-SA. The standalone program can be run independently of USB communication or while communication is active.

Standalone programs can be written to the DMX-J-SA using the Windows GUI described in section 8. Once a standalone program is written by the user, it is then compiled and downloaded to the DMX-J-SA. Each line of written standalone code creates 1-4 assembly lines of code after compilation

The DMX-J-SA has the ability to store and operate two separate standalone programs simultaneously.

7.14.1. 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.

7.14.2. Standalone Control

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 section 10. The following assignments can be used to control a standalone program.

Value	Description
0	Stop standalone program
1	Start standalone program
2	Pause standalone program
3	Continue standalone program

Table 7.4

7.14.3. Standalone Status

The **SASTAT[0-1]** command can be used to determine the current status of the specified standalone program. Table 7.5 details the return values of this command.

Value	Description
0	Idle
1	Running

2	Paused
3	N/A
4	Errored

Table 7.5

The **SPC[0-1]** command can also be used to find the current assembled line that the specified standalone program is executing. Note that the return value of the **SPC[0-1]** command is referencing the assembly language line of code that does not directly transfer to the pre-compiled user generated code. The return value can range from [0-1274].

7.14.4. Standalone Subroutines

The DMX-J-SA has the capabilities of using up to 32 separate subroutines. Subroutines are typically used to perform functions that are repeated throughout the operation of the standalone program. Note that subroutine can be shared by both standalone programs. Refer to section 10 on further details on how to define subroutines.

Once a subroutine is written into the flash, they can be called via USB communication using the **GS** command. Standalone programs can also jump to subroutine using the **GOSUB** command. The subroutines are referenced by their subroutine number [SUB 0 - SUB 31]. If a subroutine number is not defined, the controller will return with an error.

7.14.5. Error Handling

Subroutine 31 is designated for error handling. If an error occurs during standalone execution (i.e. limit error), the standalone program will automatically jump to SUB 31.

If SUB 31 is not defined, the program will cease execution and go into error state.

If SUB 31 is defined by the user, the code within SUB 31 will be executed. Typically the code within subroutine 31 will contain the standalone command **ECLEARX** in order to clear the current error. Section 10 will contain examples of using subroutine 31 to perform error handling.

The return jump from subroutine 31 will be determined by bit 6 of the **POL** register. This setting will determine if the standalone program will jump back to the beginning of the program or to the last performed line. See table 7.3 for details.

7.14.6. Standalone Variables

The DMX-J-SA has 50 32-bit signed standalone variables available for general purpose use. They can be used to perform basic calculations and support integer operations. The **V[0-49]** command can be used to access the specified variables.

The syntax for all available operations can be found below. Note that these operations can only be performed in standalone programming.

Operator	Description	Example
+	Integer Addition	V1=V2+V3
-	Integer Subtraction	V1=V2-V3
*	Integer Multiplication	V1=V2*V3
/	Integer Division (round down)	V1=V2/V3
%	Modulus	V1=V2%5
>>	Bit Shift Right	V1=V2>>2
<<	Bit Shift Left	V1=V2<<2
&	Bitwise AND	V1=V2&7
|	Bitwise OR	V1=V2|8
~	Bitwise NOT	V1=~V2

Table 7.6

Variables V25 through V49 can be stored to flash memory using the **STORE** command. Variables V0-V24 will be initialized to zero on power up.

7.14.7. Standalone Run on Boot-Up

Standalone can be configured to run on boot-up using the **SLOAD** command. Once this command has been issued, the **STORE** command will be needed save the setting to flash memory. It will take effect on the following power cycle. See description in Table 7.7 for the bit assignment of the **SLOAD** setting.

Bit	Description
0	Standalone Program 0
1	Standalone Program 1

Table 7.7

Standalone programs can also be configured to run on boot-up using the Windows GUI. See section 8 for details.

ASCII	**SR[0-1]**	**SASTAT[0-1]**	**SPC[0-1]**	**GS[0-31]**	**V[0-100]**	**SLOAD**
Standalone	**SR[0-1]**	**-**	**-**	**GOSUB[0-31]**	**V[0-100]**	**-**

7.15. Storing to Flash

The following items are stored to flash. To store to flash, use the **STORE** command.

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, V0-V24 are reset to value 0

Table 7.8

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

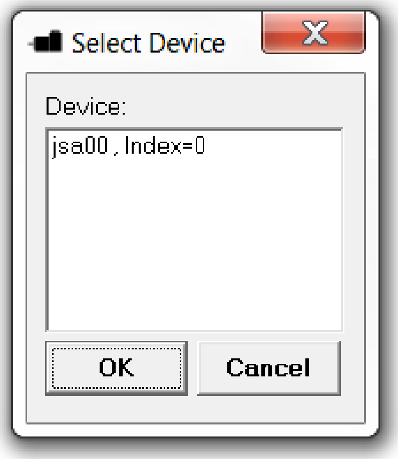
8. Software Overview

The DMX-J-SA has a Windows compatible software that allows for USB communication. Standalone programming, along with all other available features of the DMX-J-SA, will be accessible through the software. It can be downloaded from the Arcus Technology website.

To communicate over USB, make sure that the Arcus Technology USB driver is installed properly before powering on and connecting the motor.

Startup the DMX-J-SA GUI program and you will see the following screen in figure 8.0. This will allow the user to select the desired DMX-J-SA for control.

If multiple DMX-J-SA are connected to the PC, they will be listed here. Note that each DMX-J-SA connected to the PC will need a unique device name. Refer to section 6.1.2. for details.



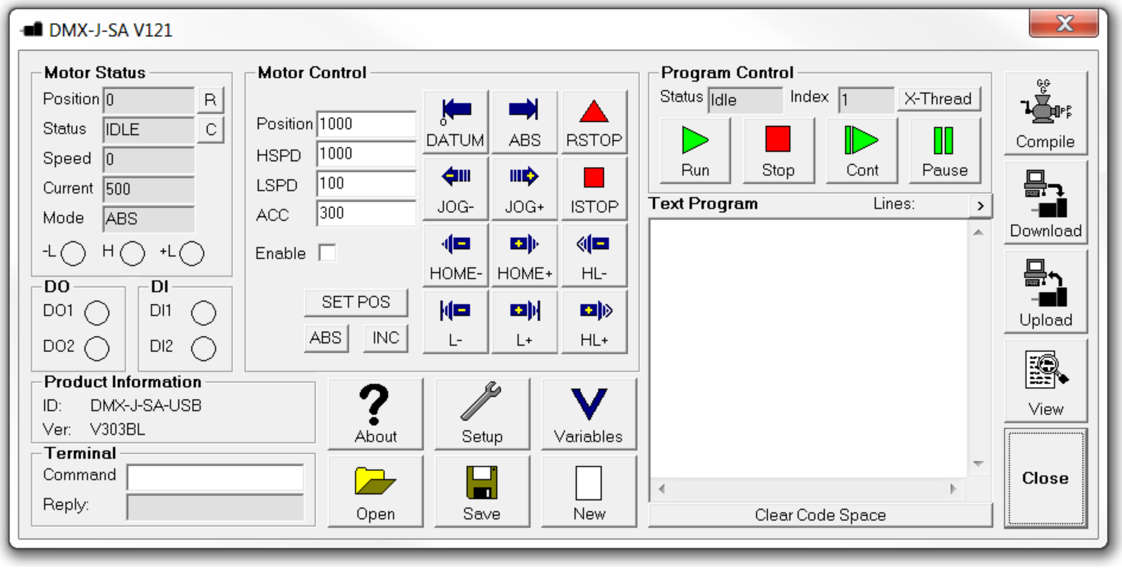
jsa00 , Index=0

Figure 8.0

Selecting the desired device and clicking OK will open the Main Control Screen.

8.1. Main Control Screen

Once communication is established, the Main Control Screen will appear. This screen allows the user to move the motor, set digital outputs, observe digital inputs, and download standalone programs.


DMX-J-SA V121

Motor Status: Position 0, Status IDLE, Speed 0, Current 500, Mode ABS

Motor Control: Position 1000, HSPD 1000, LSPD 100, ACC 300

Program Control: Status Idle, Index 1

Product Information: ID: DMX-J-SA-USB, Ver: V303BL

Figure 8.1

8.1.1. Motor Status

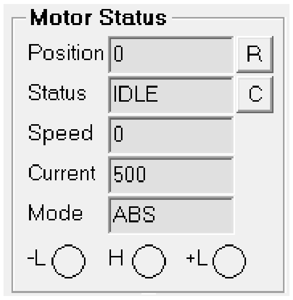

Motor Status: Position 0, Status IDLE, Speed 0, Current 500, Mode ABS

Figure 8.2

1. **Position** - Displays the current position of the motor. Units are in pulses.
   - **R** - Click this button to reset the position counter
2. **Status** - Displays the current motor status of the motor. Possible values include IDLE, ACCEL, DECEL, CONST, -LIM ERROR, and +LIM ERROR. See section 7.8 for details.
   - **C** - Click this button to clear any motor status errors.
3. **Speed** - The current speed of the motor. Units are in pulses/second.
4. **Current** - Displays the motor current. Units is in mA.
5. **Mode** - Displays the current move mode. Possible values include ABS and INC. See section 7.5 for details.
6. **-L/H/+L** - The current status of the +/- limit and home inputs.

8.1.2. Motor Control

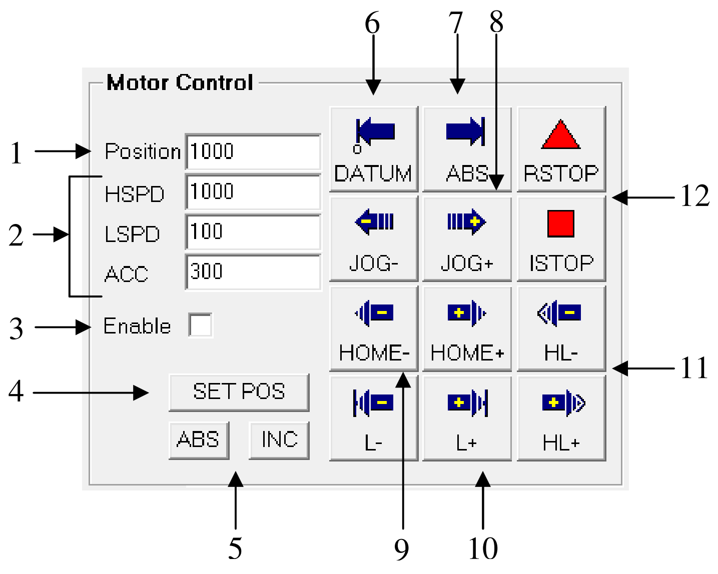


Figure 8.3

1. **Position –** Target position to move to for position moves.
2. **Speed Parameters** – Set the high speed, low speed, and acceleration values used during movement.
3. **Enable** – Enable the motor.
4. **SET POS –** 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.
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/RSTOP** – Stop the motor immediately or with a decelerated ramp down.

8.1.3. DI Status/DO Status/Enable

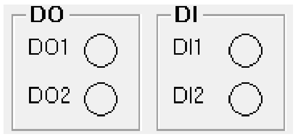


Figure 8.4

1. **Digital Output Status** – Display of the two digital output statuses. Click on the digital output circle to toggle the specified output.
2. **Digital Input Status –** Display of the two digital input bits.

8.1.4. Terminal

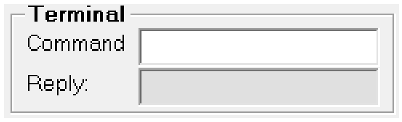
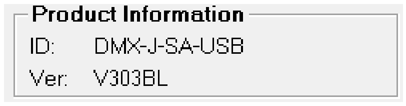


Figure 8.5

The terminal dialog box allows for manual sending of ASCII commands. The reply to the sent ASCII command will be displayed in the Reply field.

8.1.5. Product Information



Product Information

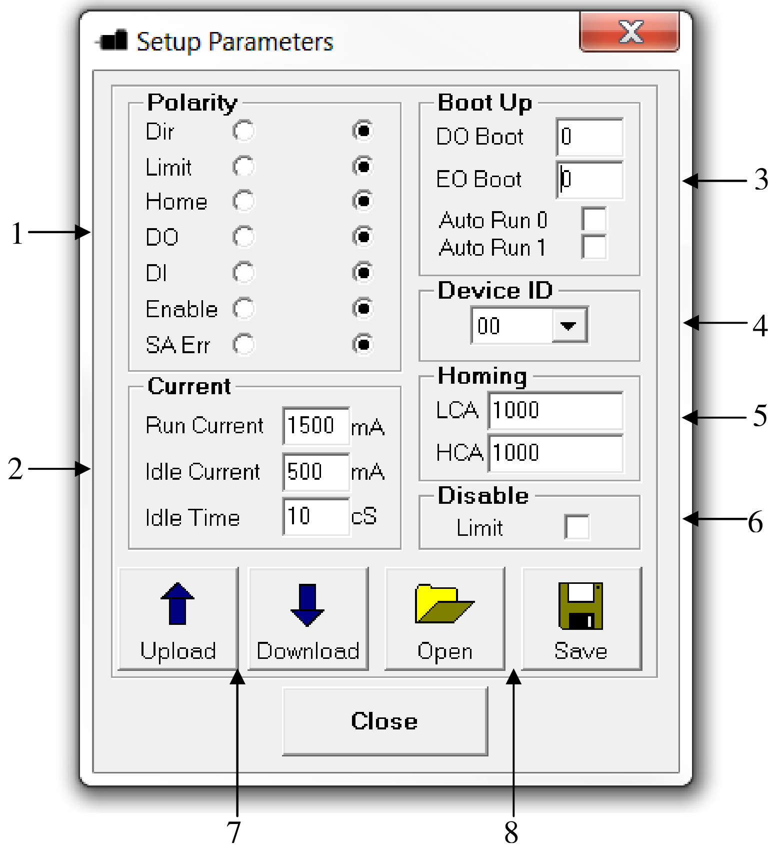
ID: DMX-J-SA-USB

Ver: V303BL

Figure 8.6

This section displays the product ID of the DMX-J-SA and the current firmware revision on the motor.

8.1.6. Setup



Setup Parameters

Polarity
Dir ○ ●
Limit ○ ●
Home ○ ●
DO ○ ●
DI ○ ●
Enable ○ ●
SA Err ○ ●

Current
Run Current 1500 mA
Idle Current 500 mA
Idle Time 10 cS

Boot Up
DO Boot 0
EO Boot 0
Auto Run 0 ☐
Auto Run 1 ☐

Device ID
00

Homing
LCA 1000
HCA 1000

Disable
Limit ☐

Figure 8.7

1. **Polarity** – the following polarity parameters can be configured
   - Dir: motor direction polarity
   - Limit: limit input polarity
   - Home: home input polarity
   - DO: digital output polarity
   - DI: digital input polarity
   - Enable: enable output polarity
   - SA Err: standalone return line after error handling polarity
2. **Current** – This section allows you to change the available driver settings
   - Run Current: motor current setting when the motor is moving [mA]
   - Idle Current: motor current setting when the motor is idle [mA]
   - Idle Time: the time required after the motor becomes idle for the motor to change from run current to idle current [centi-seconds]
3. **Boot Up**
   - DO Boot: The status of the digital outputs on power up

- EO Boot: the status of the enable output on power up
- Auto Run 0/1: start the specified standalone program on power up

4. **Device ID** – Set the Device Name [JSA00-JSA99].
5. **Homing**
   - LCA: set the Limit Correction Amount for limit homing.
   - HCA: set the Home Correction Amount for homing using the home input.
6. **Disable Limit** – Disable the limit inputs for general purpose use.
7. **Upload/Download** – Upload the settings currently on the DMX-J-SA or download the setting shown
8. **Open/Save** – Parameters can be saved to a file or read from a file

8.1.7. Standalone Program File Management

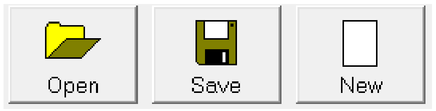
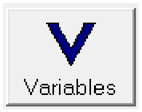
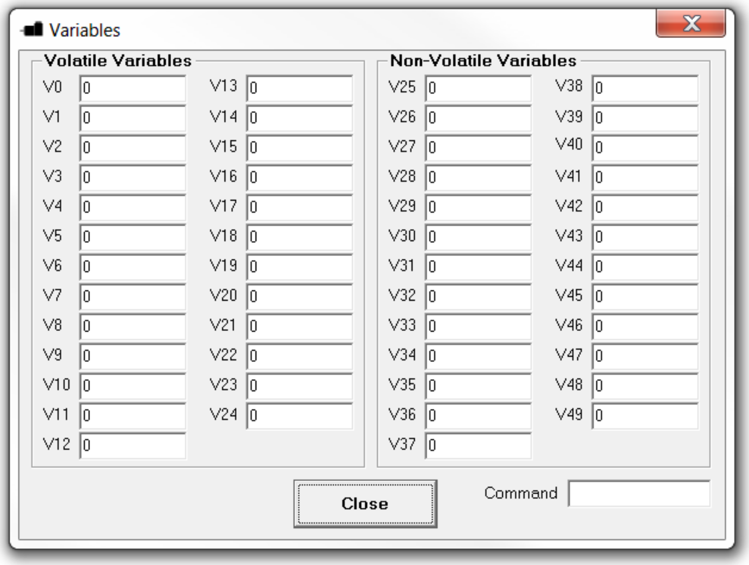


Figure 8.8

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

8.1.8. Variable Status





V0	0	V13	0	V25	0	V38	0
V1	0	V14	0	V26	0	V39	0
V2	0	V15	0	V27	0	V40	0
V3	0	V16	0	V28	0	V41	0
V4	0	V17	0	V29	0	V42	0
V5	0	V18	0	V30	0	V43	0
V6	0	V19	0	V31	0	V44	0
V7	0	V20	0	V32	0	V45	0
V8	0	V21	0	V33	0	V46	0
V9	0	V22	0	V34	0	V47	0
V10	0	V23	0	V35	0	V48	0
V11	0	V24	0	V36	0	V49	0
V12	0			V37	0		

Figure 8.9

1. Display the current values of standalone variables V0-V49
2. **Command Line** – To write to a variable, use V[0-49]=[value] syntax.

8.1.9. Standalone Program Editor

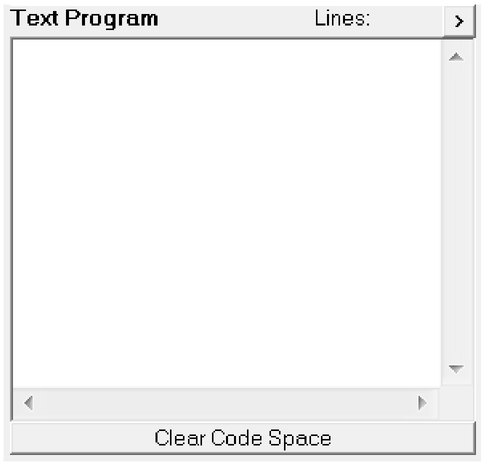


Figure 8.10

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.

8.1.10. Standalone Program Compile/Download/Upload/View

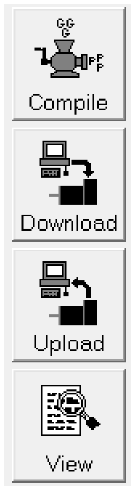


Figure 8.11

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.

8.1.11. Program Control

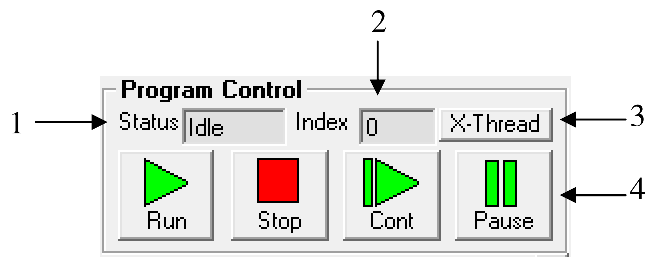


Figure 8.12

1. **Program Status** – display of program status. The following values are available.
   - 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.
3. **X-Thread** – Open the Program Control for standalone multi-thread operation.
4. **Program Control** – the following program control options are available.
   - **Run**: Run the standalone program
   - **Stop**: Stop the standalone program
   - **Cont**: Continue the standalone program
   - **Pause:** Pause the standalone program

9. ASCII Language Specification

Important Note: All the commands described in this section are interactive commands and are not analogous to standalone commands. Refer to section 10 for details regarding standalone commands.

DMX-J-SA ASCII protocol is case sensitive. All commands should be in upper case letters.

An invalid command is returned with a “?”. Always check for the proper reply when a command is sent.

For USB communication, the commands detailed in table 10.0 are valid.

9.1. ASCII Command Set

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.	[10-1000]ms
ACC=[Value]	Set the acceleration value. [10-1000]ms Example: ACC=300	OK
ACR	Return the idle current of the motor.	[0-2000]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.	[0-2000]mA
CLR	Clear limit error.	OK
CUR	Return the run current of the motor. [100-2000]mA	[0-2000]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 7.1
DI[1-5]	Return the individual status of the digital input.	Refer to Table 7.1
DL	Return the disable limit function status. 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 7.2
DO=[Value]	Set the digital outputs.	OK
DO[1-2]	Return the status of the individual digital output.	Refer to Table 7.2
DO[1-2] =[Value]	Set the individual digital output. Refer to Table 7.2.	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	OK

	memory.	
H+	Home the motor in positive direction.	OK
H-	Home the motor in negative direction.	OK
HCA	Return the home correction amount.	[0-16777215]
HCA=[Value]	Set the home correction amount.	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.	[100-200000]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.	[0-16777215]
LCA=[Value]	Set the limit correction amount.	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 7.0
POL	Return the current polarity.	Refer to Table 7.3
POL=[Value]	Set the polarity. Refer to Table 7.3.	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. Refer to Table 7.5.	[0-4]
SA[0-1274]	Return the standalone line.	
SA[0-1274]=[Value]	Set the standalone line.	OK
SLOAD	Return the standalone program run on boot parameter. See Table 7.7.	[0-3]
SLOAD=[0-3]	Set the standalone program run on boot parameters. See Table 7.7.	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. See Table 7.4.	OK
STOP	Stop all motion using deceleration.	OK
STORE	Store settings to flash. Refer to Table 7.8.	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]	Perform a positional move.	OK

Table 9.0

9.2. 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 DMX-J-SA.
?LimErrored	Motor is in limit error state.
?Moving	A move or position change command is sent while the DMX-J-SA is outputting pulses.
?Overmove	An absolute or increment move is issued with a move amount greater than 16,777,215.
?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 9.1

10. Standalone Programming Specification

Important Note: All the commands described in this section are standalone language commands and are not analogous to ASCII commands. Refer to section 9 for details regarding ASCII commands.

10.1. Standalone Command Set

Command	R/W	Description	Example
;	-	Comment notation. Comments out any text following ; in the same line.	;This is a comment
ABORTX	W	Immediately stop all motion	ABORTX
ABS	W	Set the move mode to absolute mode	ABS X1000 ;move to position 1000

ACC	R/W	Set/get the acceleration setting. Unit is in milliseconds	ACC=500
			ACC=V1

DELAY=[Value]	W	Set a delay in milliseconds. Assigned value is a 32-bit unsigned integer.	DELAY=1000 ;1 second

DI	R	Return status of digital inputs. See Table 7.1 for bitwise assignment.	IF DI=0
			  DO=1 ;Turn on DO1
			ENDIF

DI[1-5]	R	Get individual bit status of digital inputs. Will return [0,1]. See Table 7.1 for bitwise assignment.	IF DI1=0
			  DO=1 ;Turn on DO1
			ENDIF

DO	R/W	Set/get digital output status. See Table 7.2 for bitwise assignment.	DO=2 ;Turn on DO2
DO[1-2]	R/W	Set/get individual bit status of digital outputs. Range for the bit assigned digital outputs is [0,1].	DO2=1 ;Turn on DO2
ECLEARX	W	Clear any motor status errors. See Table 7.0 for types of errors.	ECLEARX
EO	R/W	Set/get the enable output status.	EO=1 ;Enable the motor
GOSUB [0-31]	-	Call a subroutine that has been previously stored to flash memory	GOSUB 0 END
HLHOMEX[+/-]	W	Home the motor using the home input at low and high speeds in the specified direction. See section 7.6.2 for details	HLHOMEX+ ;positive home WAITX ;wait for home move
HOMEX[+/-]	W	Home the motor using the home input at high speed in the specified direction. See section 7.6.1 for details.	HOMEX- ;negative home WAITX ;wait for home move

HSPD	R/W	Set/get the high speed setting. Unit is in pulses/second.	HSPD=1000
			HSPD=V1

IF
ELSEIF
ELSE
ENDIF
	-	Perform a standard IF/ELSEIF/ELSE conditional. Any command with read ability can be used in a conditional
		
		ENDIF should be used to close off an IF statement.
			IF DI1=0
			  DO=1 ;Turn on DO1
			ELSEIF DI2=0
			  DO=2; Turn on DO2
			ELSE
			  DO=0; Turn off DO
			ENDIF

		Conditions [=, >, <, >=, <=, !=] are available	
INC	W	Set the move mode to incremental mode	INC X1000 ;increment by 1000
JOGX[+/-]	W	Move the motor indefinitely in the specified direction	JOGX+ JOGX-
LHOMEX[+/-]	W	Home the motor using the limit inputs in the specified directions. See section 7.6.3 for details.	LHOMEX+ ;positive home WAITX

LSPD	R/W	Set/get the low speed setting. Units are in pulses/second	LSPD=100
			LSPD=V3

MSTX	R	Get the current motor status of the motor. See Table 7.0 for motor status assignment.	
PRG [0-1] END	-	Used to define the beginning and end of a main program. The DMX-J-SA can have up to two main programs	PRG 0 ;main program END

PX	R/W	Set/get the current motor position.	PX=1000 ;Set to 1000
			V1=PX ;Read current position

SR[0,1]=[Value]	W	Set the standalone control for the specified program. See Table 7.4	SR0=0 ;Turn off program 0
STOPX	W	Stop all motion using a decelerated stop	STOPX
STORE	-	Store settings to flash	STORE

SUB [0-31]
ENDSUB
	-	Defines the beginning of a subroutine. ENDSUB should be used to define the end of the subroutine.	**SUB 1**
			DO=4
			ENDSUB

TOC	W	Sets the communication time-out parameter. Units is in milliseconds	**TOC=1000 ;1 SECOND TIME-OUT**

V[0-49]	R/W	Set/get standalone variables.
		
		The following operations are available:
		[+] Addition
		[-] Subtraction
		[*] Multiplication
		[/] Division
		[%] Modulus
		[>>] Bit shift right
		[<<] Bit shift left
		[&] Bitwise AND
		[|] Bitwise OR
		[~] Bitwise NOT
			V1=12345 ;Set V1 to 12345
			V2=V1+1;Set V2 to V1 + 1
			V3=DI ;Set V3 to DI
			V4=DO ;SET V4 TO DO
			V5=~EO ;SET V5 TO NOT EO

WAITX	W	Wait for current motion to complete before processing the next line	**X1000 ;MOVE TO POSITION 1000** WAITX ;wait for move

WHILE
ENDWHILE
	-	Perform a standard WHILE loop within the standalone program. ENDWHILE should be used to close off a WHILE loop.
		
		Conditions [=, >, <, >=, <=, !=] are available.
			WHILE 1=1 ;FOREVER LOOP
			DO=1 ;Turn on DO1
			DO=0 ;Turn off DO1
			ENDWHILE

X[position]	W	If in absolute mode, move the motor to [position]. If in incremental mode, move the motor to [current position] + [position].	**X1000**

Table 10.0

10.2. Example Standalone Programs

10.2.1. 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
```

10.2.2. 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
```

10.2.3. 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
```

10.2.4. 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               ;* End of IF statement
ENDWHILE                    ;* Go back to WHILE statement
END
```

10.2.5. 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               ;* End of IF statement
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
```

10.2.6. 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                   ;* End of IF statement
        V1=MSTX                 ;* Store the motor status to variable 1
        V2=V1&7                 ;* Get first 3 bits
        IF V2!=0                ;* If any of the first 3 bits are high (moving)
                DO1=1           ;* Turn on DO1
        ELSE                    ;* If all 3 first bits are low (idle)
                DO1=0           ;* Turn off DO1
        ENDIF                   ;* End of IF statement
ENDWHILE                        ;* Go back to WHILE statement
END

SUB 31                          ;* Start of subroutine 31
        ECLEARX                 ;* Clear the current error
ENDSUB                          ;* End of subroutine 31
```

10.2.7. 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 20000 pulses/sec
LSPD=500                ;* Set low speed to 500 pulses/sec
ACC=500                 ;* Set acceleration to 500 msec
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 of IF statement
ENDWHILE                ;* Go back to WHILE statement
END                     ;* End Program 1

SUB 31                  ;* Start of subroutine 31
    ECLEARX             ;* Clear the current error
ENDSUB                  ;* End of subroutine 31
```

10.2.8. 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 pulses/sec
LSPD=500                    ;* Set low speed to 500 pulses/sec
ACC=500                     ;* Set acceleration to 500 msec
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               ;* End of IF statement
        ENDWHILE            ;* Go back to WHILE statement
END                         ;* End Program 1

SUB 31                      ;* Start of subroutine 31
        ECLEARX             ;* Clear the current error
ENDSUB                      ;* End of subroutine 31
```

Contact Information

Arcus Technology, Inc.

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

www.arcus-technology.com