DMX-K-SA

Integrated Step Motor + Encoder + Driver + Controller

First edition, April 2007

Revision History:

2.00 – 1st Release
3.15 – 2nd Release
3.16 – 3rd Release
3.17 – 4th Release
3.19 – 5th Release
3.20 – 6th Release

Firmware Compatibility:

†V403BL

†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-K-SA is an all-in-one integrated motor package that combines all the motion components in to one convenient package.

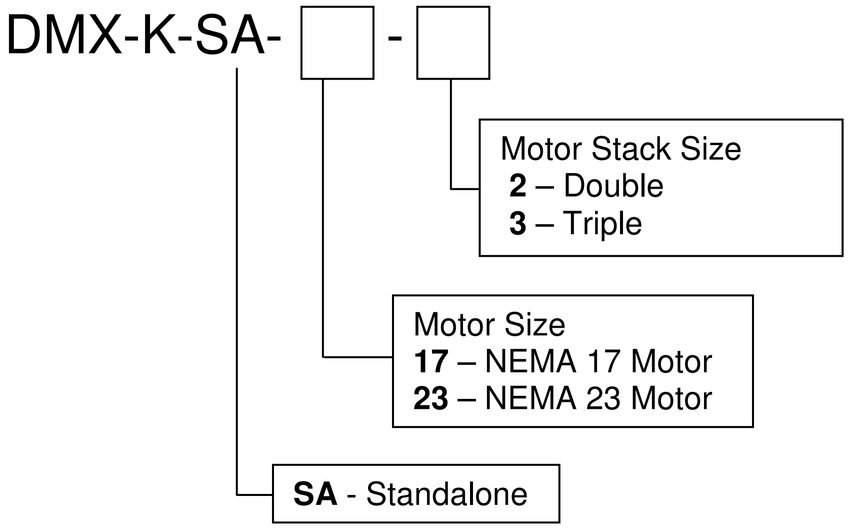
Communication to the DMX-K-SA can be established over RS-232 or RS-485. It is also possible to download a stand-alone program to the device and have it run independent of a host.

Sample source code is available to aid you in your software development.

1.1. Features

- RS-485 and RS-232 ASCII communication
  - 9600, 19200, 38400, 57600, 115200 bps
- Stepper driver
  - 12-35 VDC
  - 2.5 Amp max current setting (peak current)
  - 16 micro-step setting (fixed)
- Stepper motor
  - NEMA 17/23 motor sizes available in different stack sizes
  - 1.8° step angle
- 1000 line incremental encoder (4000 line with quadrature decoding)
  - StepNLoop closed loop control (position verification)
- Opto-isolated I/O
  - 6 x general purpose inputs
  - 3 x general purpose outputs
  - +Limit/-Limit/Home inputs
- Homing routines
  - Home input only
  - Limit inputs only
  - Z-index encoder channel only
  - Home input + Z index encoder channel
- S-curve or trapezoidal acceleration profile control
- On-the-fly speed change

1.2. Part Numbering Scheme



Contacting Support
For technical support contact: support@arcus-technology.com.
Or, contact your local distributor for technical support.

2. Electrical and Thermal Specifications

Parameter	Min	Max	Units
Main Power Input	+12	+35	V
	-	2.5	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	-	90	mA
Operating Temperature	0	+85	℃
Storage Temperature	-55	+150	℃

Table 2.0

3. Dimensions

3.1. DMX-K-SA-17

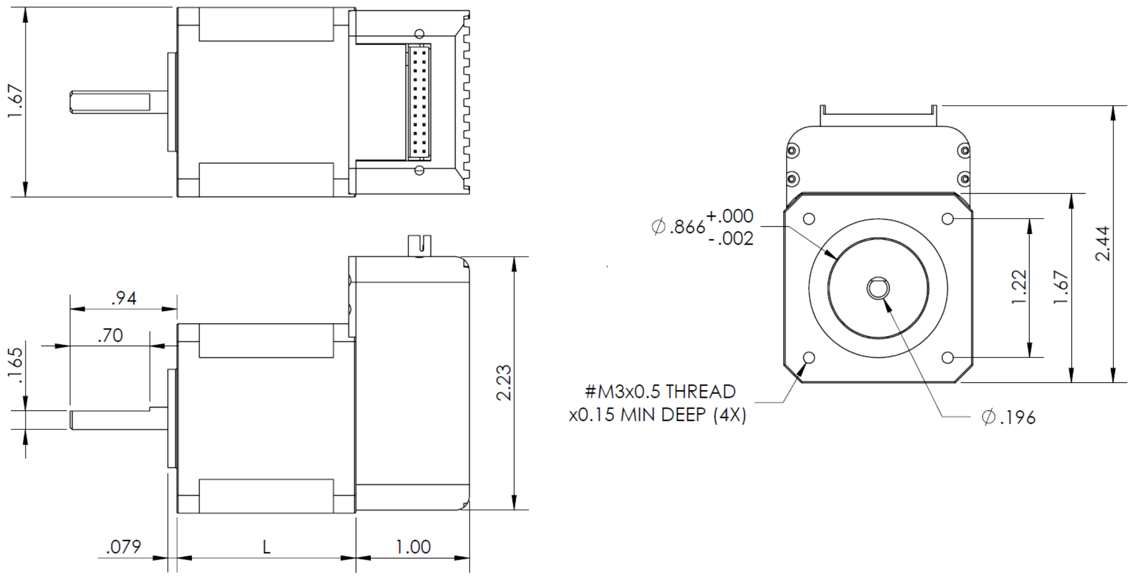


Figure 3.0

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

Table 3.0

[1] All dimensions in inches.

3.2. DMX-K-SA-23

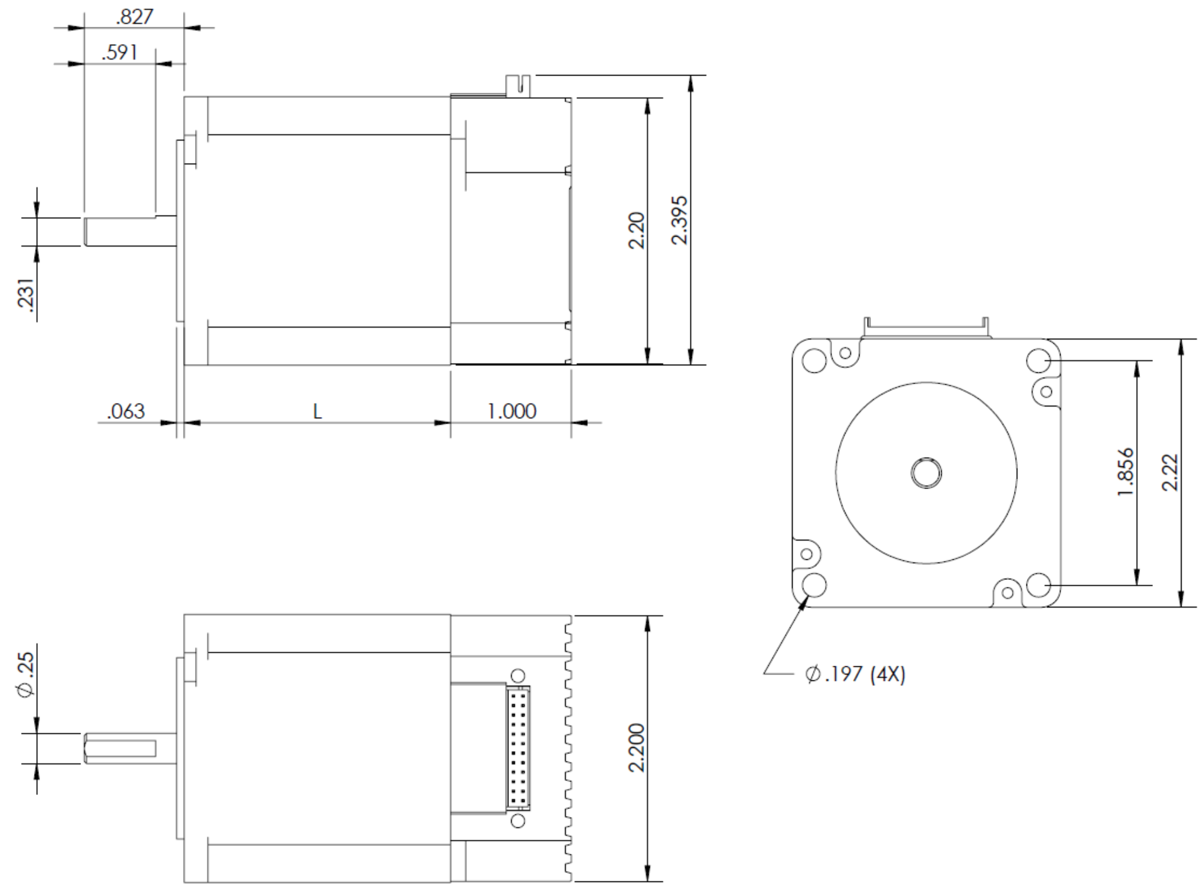


Figure 3.1

NEMA 23 Models	L (inches)
DMX-K-SA-23-2 (double stack)	2.20
DMX-K-SA-23-3 (triple stack)	2.99

Table 3.1

[1] All dimensions in inches.

4. Connectivity

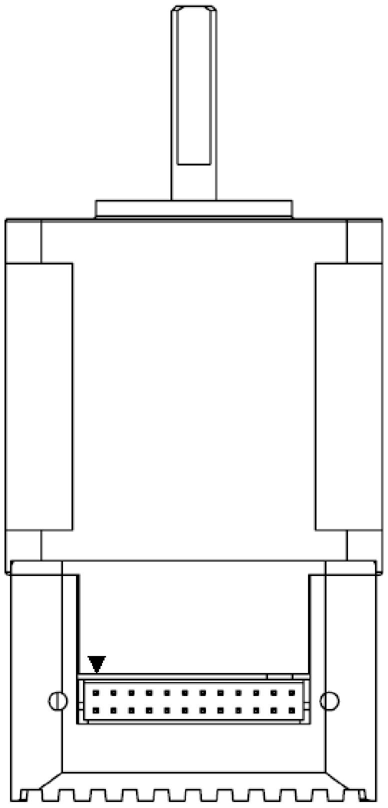


Figure 4.0

4.1. 24-Pin Connector (2.0mm)

Pin #	Wire Color	In/Out	Name	Description
1	Red	I	PWR	Power Input (+12 to +35VDC)
2	Red	I	PWR	Power Input (Shorted to pin 1)
3	Black	I	GND	Ground
4	Green/Brown	I/O	485+	RS-485+ signal
5	White	I	HOME	Home input
6	Yellow/Brown	I/O	485-	RS-485- signal
7	Black/Yellow	O	DO1/INP	Digital Output 1 / In Position
8	Yellow	I	+LIM	Plus limit input
9	Green/Red	I/O	RXD	RS-232 RXD signal
10	Yellow/White	I	-LIM	Minus limit input
11	Brown/Yellow	I	DI1	Digital Input 1
12	Brown/Yellow	I	DI3	Digital Input 3
13	Brown/Yellow	I	DI2/LT	Digital Input 2 / Latch
14	Brown/Yellow	I	DI4	Digital Input 4

15	Gray	NC	NC	No Connection
16	Brown/Yellow	I	DI5	Digital Input 5
17	Orange	I	OPTO	Opto-supply (+12 to +24VDC)
18	Brown/Yellow	I	DI6	Digital Input 6
19	Yellow/Red	I/O	TXD	RS-232 TXD signal
20	Black/Yellow	O	DO2	Digital Output 2
21	Black/Gray	I	OPTOGND	Opto-supply Ground
22	Black/Yellow	O	DO3/ALM	Digital Output 3 / Alarm
23	Gray	NC	NC	No Connection
24	Black	I	GND	Ground (Shorted to pin 3)

Table 4.0

Mating Connector Description:	24 pin 2mm dual row connector
Mating Connector Manufacturer:	HIROSE
Mating Connector Housing Part Number:	DF11-24DS-2C

4.2. DMX-K-SA-17/23 Interface Circuit

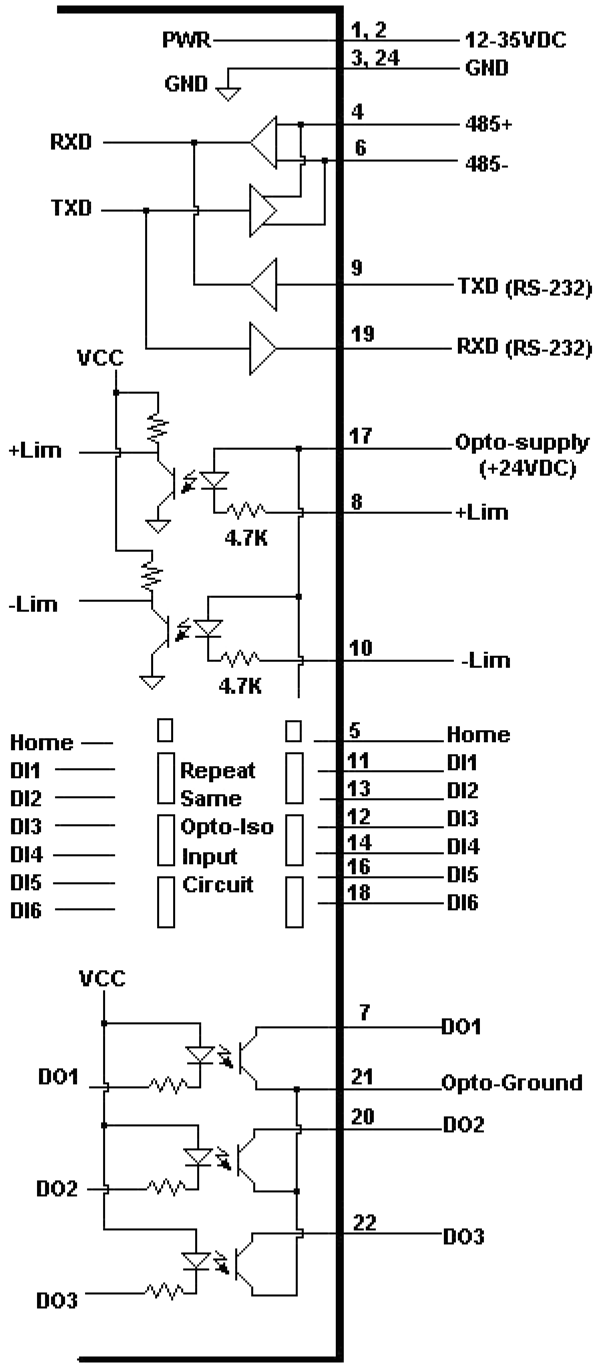


Figure 4.1

4.3. Digital Inputs, Limits, and Home

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

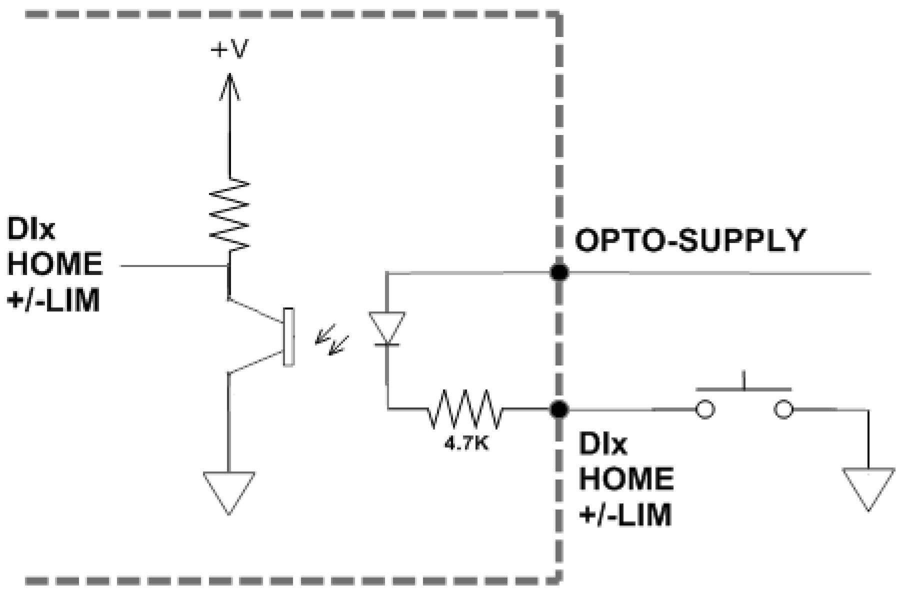


Figure 4.2

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

4.4. Digital Outputs

Figure 4.3 shows an example wiring to the digital output. All opto-isolated digital outputs will be NPN type.

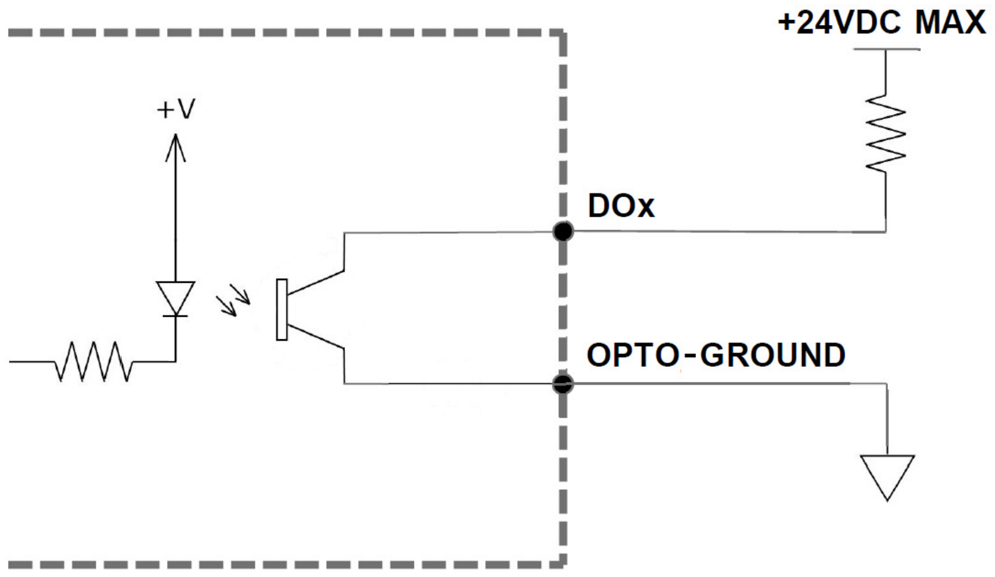


Figure 4.3

The opto-ground must be connected in order for the digital outputs to operate.

When activated, the opto-isolator for the digital output sinks the voltage on the digital output line to the opto-ground. The maximum sink current for digital outputs is 90mA. Take caution to select the appropriate external resistor so that the current does not exceed 90mA. Additionally, the pull up voltage should not exceed +24VDC.

When deactivated, the opto-isolator will break the connection between the digital output and the opto-ground. In this case, the voltage on the digital output signal will be the pull-up voltage.

5. Stepper Motor Driver Overview

5.1. Microstep

The DMX-K-SA has a fixed microstepping of 16. With the standard 1.8° stepper motor included with the DMX-K-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-K-SA will have a maximum rated driver current that is dependent on the frame and stack size of the motor. See table 5.0 for details.

Product	Max Rated Driver Current	Recommended Driver Current
DMX-K-SA-17-2	1.7 A	1.4 A
DMX-K-SA-17-3	2.0 A	1.6 A
DMX-K-SA-23-2	2.5 A	2.0 A
DMX-K-SA-23-3	2.5 A	2.0 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-K-SA has configurable current setting from 100mA to 2.5A. The 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 7.17 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-K-SA have maximum ambient operating temperature of **85 degree Celsius**. DMX-K-SA electronics are potted with heat-conductive compound 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-K-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-K-SA mounting orientation should be such that the fins are oriented vertically for better convection and heat dissipation.

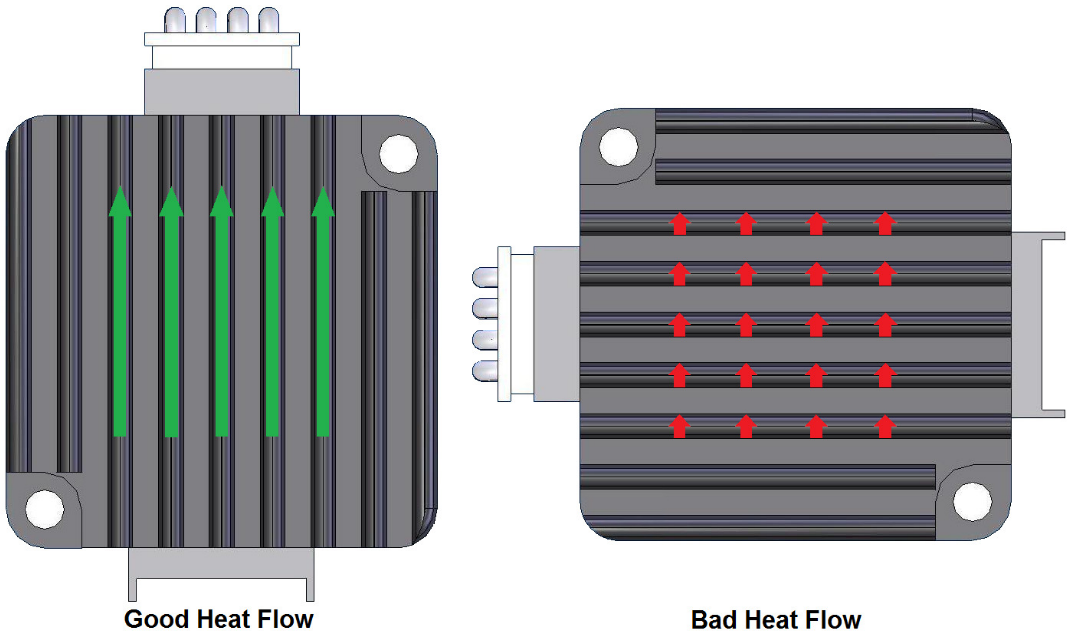


Figure 5.0

5.4. Stepper Motor Specifications

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

NEMA Size	Stack	Max Current	Resistance/Phase	Inductance/Phase	Inertia
17	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
23	Double	2.8A	0.9 Ohm	2.5 mH	1.64 oz-in^2
	Triple	2.8A	1.13 Ohm	3.6 mH	2.62 oz-in^2

Table 5.1

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

NEMA Size	Stack Size	Max Axial Force	Max Radial Force
17	Double	15N	10N
	Triple	15N	10N
23	Double	15N	75N
	Triple	15N	75N

Table 5.2

5.5. Stepper Motor Torque

The torque output of the DMX-K-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-K-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-K-SA.

See figure 5.1 for the torque curves of DMX-K-SA-17-X and figure 5.2 for the torque curves of DMX-K-SA-23-X

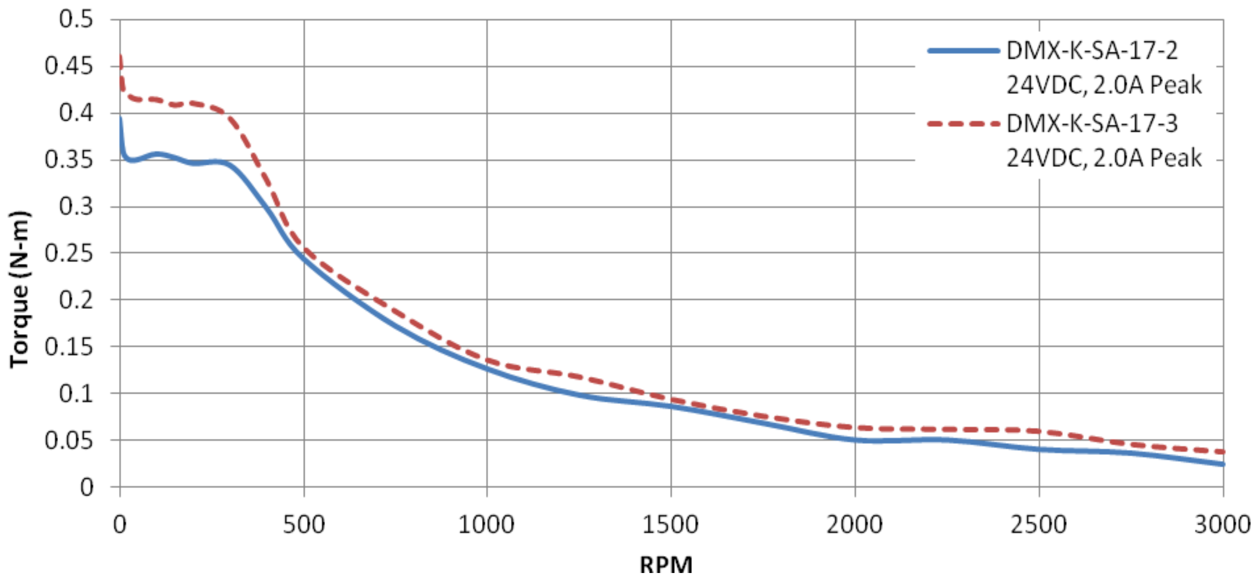


Figure 5.1

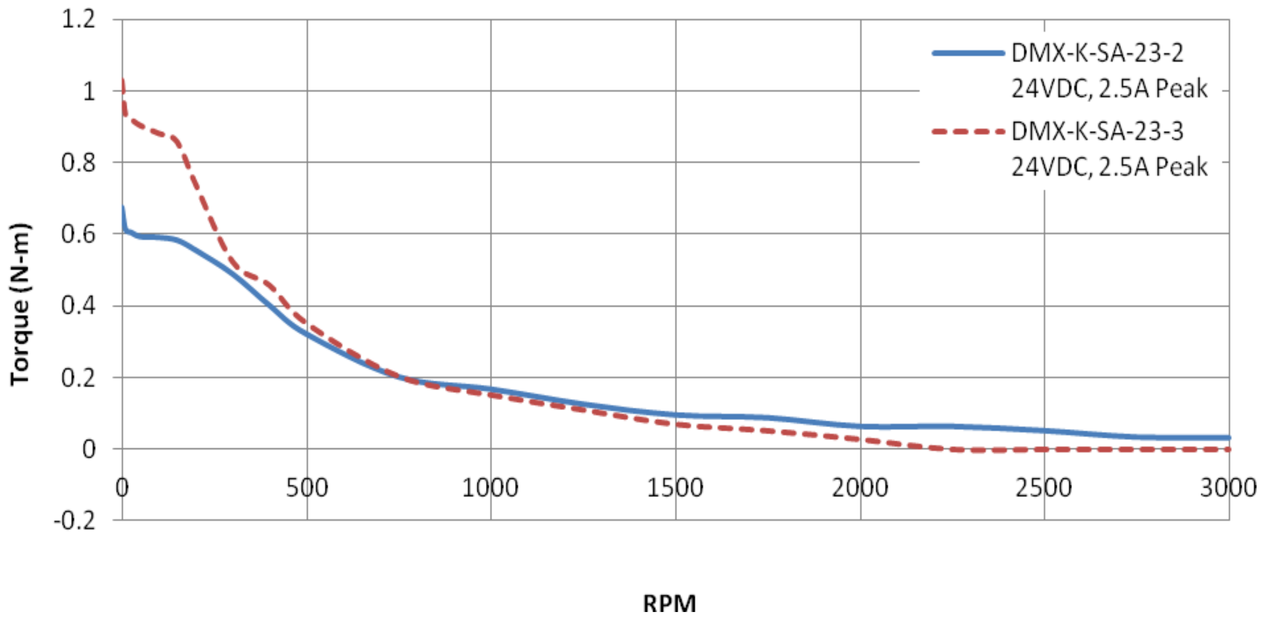

1.2
1
0.8
0.6
0.4
0.2
0
-0.2
Torque (N-m)
0
500
1000
1500
2000
2500
3000
RPM
DMX-K-SA-23-2
24VDC, 2.5A Peak
DMX-K-SA-23-3
24VDC, 2.5A Peak

Figure 5.2

6. Communication Interface

6.1. Serial Communication

The DMX-K-SA has the ability to communicate over an RS-485 or RS-232 interface. Both RS-485 and RS-232 will use an ASCII protocol. An RS-485 or RS-232 serial port on the PC or PLC can be used to communicate with the DMX-K-SA. A USB to RS-485 or USB to RS-232 converter can also be used.

6.1.1. Typical RS-485 Setup

A typical RS-485 network is shown in figure 6.0. Several techniques can be used to increase the robustness of an RS-485 network. Please see section 6.1.7 for details.

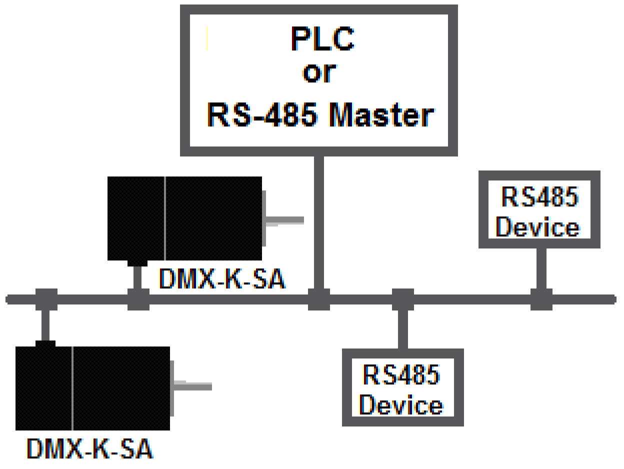


Figure 6.0

6.1.2. Typical RS-232 Setup

A typical RS-232 network is shown in figure 6.1. The RS-232 bus is typically point-to-point and does not allow for multi-dropping.

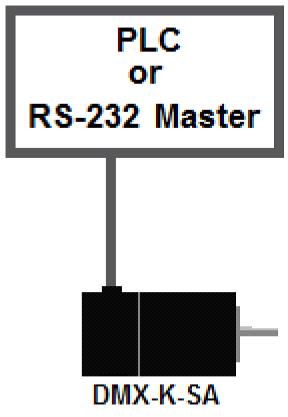


Figure 6.1

6.1.3. Communication Port Settings
The DMX-K-SA has the communication port settings shown in table 6.0.

Parameter	Setting
Byte Size	8 bits
Parity	None
Flow Control	None
Stop Bit	1

Table 6.0

DMX-K-SA provides the user with the ability to set the desired baud rate of the serial communication. In order to make these changes, first set the desired baud rate by using the **DB** command.

Return Value	Description
1	9600
2	19200
3	38400
4	57600
5	115200

Table 6.1

To write the values to the device's flash memory, use the **STORE** command. After a complete power cycle, the new baud rate will be written to memory. Note that before a power cycle is completed, the settings will not take effect.

By default, the DMX-K-SA has a baud rate setting of 9600 bps.

6.1.4. ASCII Protocol
The following ASCII protocol should be used for sending commands and receiving replies from the DMX-K-SA. Details on valid ASCII command can be found in section 9. The following protocol will be used for both RS-485 and RS-232 interfaces.

Sending Command

ASCII command string in the format of

@[DeviceName][ASCII Command][CR]

[CR] character has ASCII code 13.

Receiving Reply

The response will be in the format of

[Response][CR]

[CR] character has ASCII code 13.

Examples:

For querying the polarity
```
Send: @00POL[CR]
Reply: 7[CR]
```

For reading the digital input status
```
Send: @00DI[CR]
Reply: 8[CR]
```

For performing a store to flash memory
```
Send: @00STORE[CR]
Reply: OK[CR]
```

6.1.5. Response Type Selection

It is also possible to choose different format for the response string from the DMX-K-SA. This parameter can be set using the **RT** command.

If **RT** is set to zero using the **RT=0** command, the format will follow the protocol shown in section 6.1.4. If **RT=1**, the response string will be in the format #[DeviceName][Response][Null].

Examples:

For querying the encoder position
```
Send: @01EX[CR]
Reply: #011000[Null]
```

For jogging the motor in positive direction
```
Send: @01J+[CR]
Reply: #01OK[Null]
```

For aborting any motion in progress
```
Send: @01ABORT[CR]
Reply: #01OK[Null]
```

To write the response type parameter to flash memory, use the **STORE** command. After a complete power cycle, the new response type will take effect. Note that before a power cycle is done, the setting will not take effect.

6.1.6. Broadcasting Over RS-485

The address '00' is reserved for broadcasting over an RS-485 bus. Any ASCII command prefixed by '@00' will be processed by all DMX-K-SA modules on the RS-485 bus. When a broadcast command is received by an DMX-K-SA module, no response is sent back to the master.

The broadcast feature can still be used when communicating over RS-232 however, due to the point-to-point architecture, it is typically not necessary.

6.1.7. RS-485 Communication Issues

RS-485 communication issues can arise due to noise on the RS-485 bus. The following techniques can be used to help reduce noise issues.

Daisy Chaining
For a multi-drop RS-485 network, be sure that the network uses daisy-chain wiring. Figure 6.0 shows an example of a daisy chain network.

Number of Nodes
The maximum number of nodes recommended is 32. Increasing beyond this number will require special attention

Twisted Pair Wiring
To reduce noise, it is recommended to use twisted pair wiring for the 485+ and 485- lines. This technique will help cancel out electromagnetic interference.

Termination
For an RS-485 network, it may be required that a 120 Ohm resistor is placed in between the 485+ and 485- signals, at the beginning and end of the bus. A terminal resistor will help eliminate electrical reflections on the RS-485 network.

Note that on short communication buses, or buses with a small number of nodes, termination resistors may not be needed. Inclusion of terminal resistors when they are not needed may mask the main signal entirely.

6.2. Windows GUI

DMX-K-SA comes with a Windows GUI program to test, program, compile, download, and debug the controller. The Windows GUI has the ability to communicate via RS-485 or RS-232. See section 8 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-K-SA.

7.1. Motion Profile

By default, the DMX-K-SA incorporates a trapezoidal velocity profile as shown in Figure 7.0.

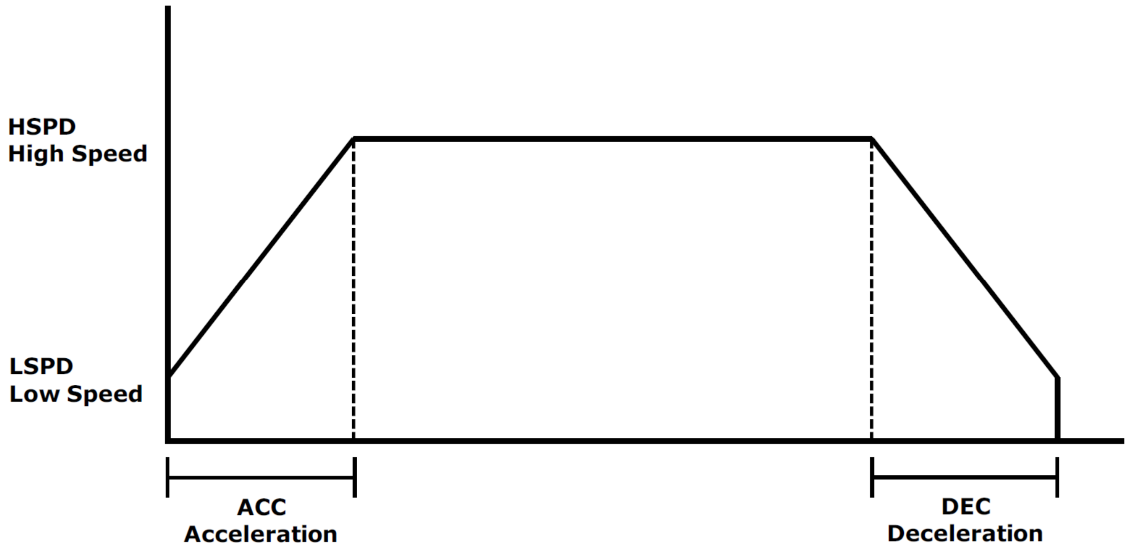


Figure 7.0

S-curve velocity profile can also be achieved by using the **SCV=0** command, as shown in Figure 7.1.

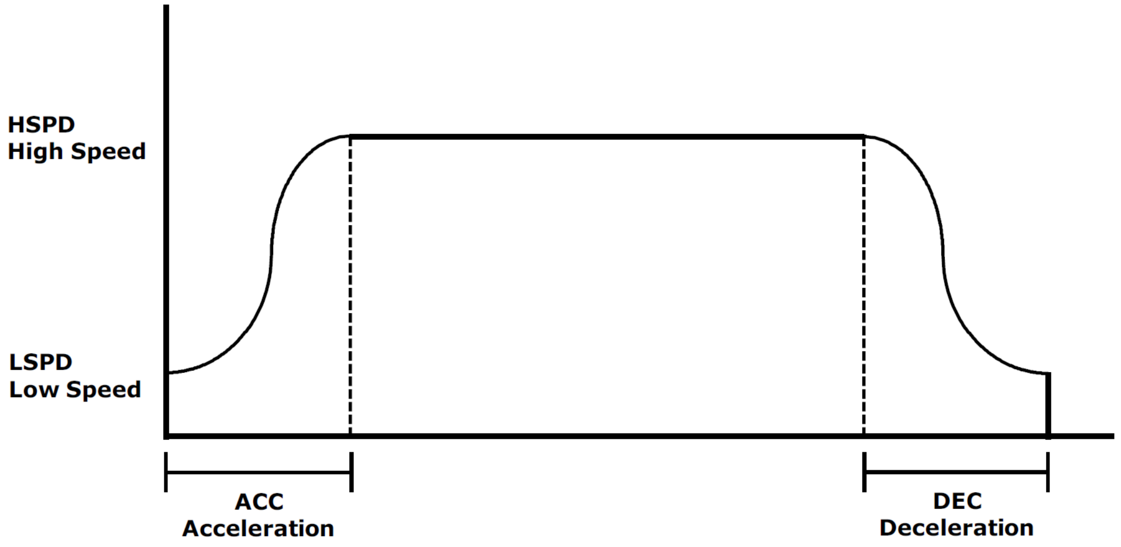


Figure 7.1

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. Depending on the voltage, current, motor type, and acceleration value, the maximum achievable speed will vary.

Standard DMX-K-SA 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. Use the **ACC** command to access the acceleration setting and the **DEC** command to access the deceleration setting. By default, the acceleration setting will be used for both the acceleration and deceleration in the motion profile. In order to decelerate using the value set in the **DEC** parameter, set the **EDEC** setting to 1.

The minimum and maximum acceleration/deceleration values depend on the high speed and low speed settings. Refer to Table A.0 and Figure A.0 in **Appendix A** for details.

ASCII	**HSPD**	**LSPD**	**ACC**	**DEC**	**EDEC**	**SCV**
Standalone	**HSPD**	**LSPD**	**ACC**	**DEC**	-	-

7.2. On-The-Fly Speed Change

An on-the-fly speed change can be achieved at any point while the motor is in motion. In order to perform an on-the-fly speed change, s-curve velocity profile must be disabled.

Before an on-the-fly speed change is performed, the correct speed window must be selected. To select a speed window, use the ASCII command **SSPDM** or the standalone command **SSPDMX**. Choosing the correct speed window will depend on the initial target speed and the final target speed. Both speeds will need to be within the same speed window.

The speed window must be set while the motor is idle. Refer to **Appendix A** for details on the speed windows.

Once the speed window has been set, an on-the-fly speed change can occur anytime the motor is in motion. The ASCII command **SSPD=[speed]** or the standalone command **SSPDX=[speed]** can be used to perform the actual speed change.

For non on-the-fly speed change moves, set the speed window to 0.

ASCII	**SSPDM**	**SSPD**
Standalone	**SSPDMX**	**SSPDX**

7.3. Position Counter

DMX-K-SA has 32 bit signed step position counter. Range of the position counter is from –2,147,483,648 to 2,147,483,647. Get the current step 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.

Similarly, the DMX-K-SA also has a 32 bit signed encoder position counter. The built in encoder will have a resolution of 1000 counts/revolution. With quadrature decoding, the resolution is increased to 4000 counts/revolution. Use the **EX** command to read and set the encoder position.

When StepNLoop closed-loop control is enabled, the **EX** command returns the encoder position and the **PX** command returns the real-time target position of the motor.

When StepNLoop closed-loop control is disabled, the **EX** command returns the encoder position and the **PX** command returns the step position. See section 7.16 for details on the StepNLoop feature.

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

7.4. Motor Power

The **EO** command can enable or disable the current to the motor. If a move command is issued to the DMX-K-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**	**-**

7.5. 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.

If a motion command is sent while the controller is already moving, the command is not processed. Instead, an error response is returned. See table 9.1 for details on error responses.

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

7.6. Stopping the 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**

7.7. Positional Moves

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

When StepNLoop is enabled, the target position in positional moves will be in units of encoder counts. When StepNLoop is disabled, the target position will be in units of motor steps. See section 7.16 for details on the StepNLoop feature.

The DMX-K-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.

If a motion command is sent while the controller is already moving, the command is not processed. Instead, an error response is returned. See table 9.1 for details on error responses.

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

7.8. On-The-Fly Target Position Change

On-the-fly target position change can be achieved using the **T[value]** command. While the motor is moving, **T[value]** will change the final destination of the motor. If the motor has already passed the new target position, it will reverse direction once the target position change command is issued.

If a **T** command is sent while the controller is not performing a target move, the command is not processed. Instead, an error response is returned. See table 9.1 for details on error responses.

ASCII	**T[pos]**
Standalone	**-**

7.9. Homing

Home search routines involve moving the motor and using the home, limit, or Z-index inputs to determine the zero reference position. Five different types of homing routines are available.

The homing routines that involve a decelerated stop will result in a final position that is non-zero. In this case the zero reference position will be the position where the deceleration occurred. The ASCII command **RZ=1** can be used to perform an automated return to the zero reference position after the deceleration is complete.

If a motion command is sent while the controller is already moving, the command is not processed. Instead, an error response is returned. See table 9.1 for details on error responses.

7.9.1. Home Input Only (High Speed Only)

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

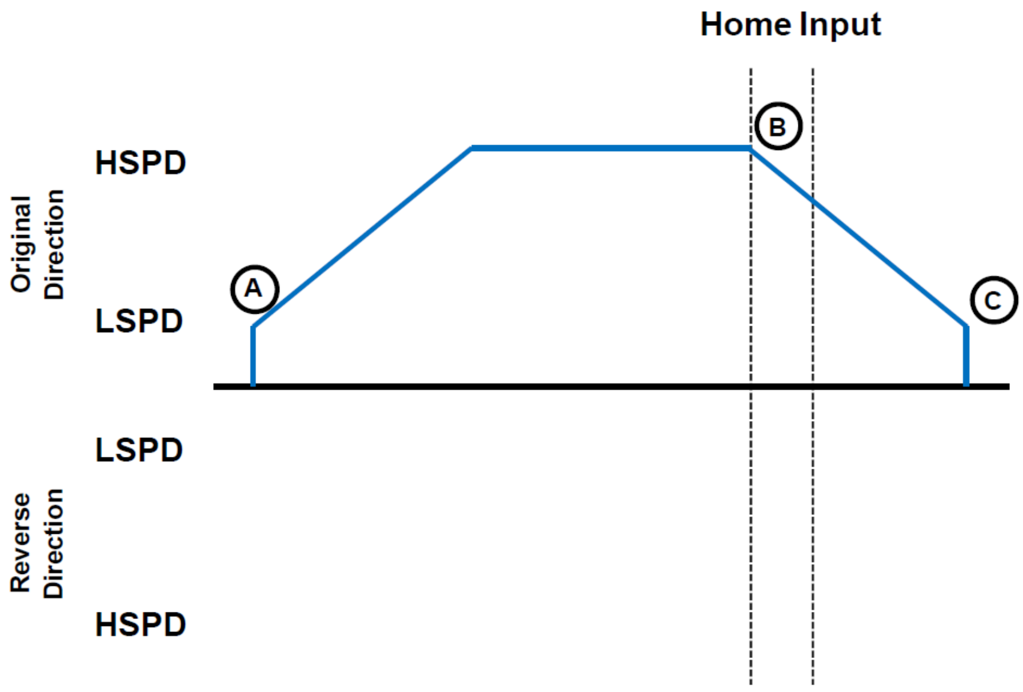


Figure 7.2

A. Starts the motor from low speed and accelerates to high speed.
B. As soon as the home input is triggered, the position counter is reset to zero and the motor begins to decelerate to low speed. As the motor decelerates, the position counter keeps counting with reference to the zero position.
C. Once low speed is reached, the motor stops. The position is non-zero however the zero position is maintained. If **RZ=1**, the motor will return to its actual zero position.

ASCII	**H+/-**	**RZ**
Standalone	**HOMEX+/-**	**-**

7.9.2. Home Input and Z-index

Use the **ZH+/-** command for ASCII mode or the **ZHOMEX+/-** command for standalone mode.. Figure 7.3 shows the homing routine.

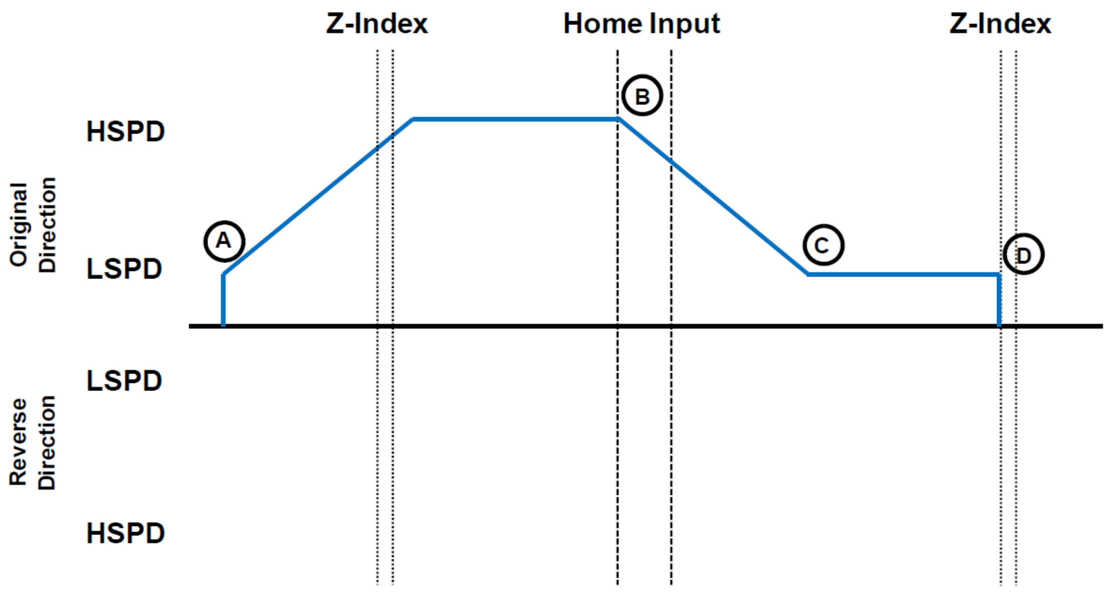


Figure 7.3

A. Issuing the command 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 motor decelerates to low speed
C. Once low speed is reached, the motor begins to search for the z-index pulse.
D. Once the z-index pulse is found, the motor stops and the position is set to zero.

ASCII	**ZH+/-**
Standalone	**ZHOMEX+/-**

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

Use the **HL+/-** command for ASCII mode and the **HLHOMEX+/-** for standalone mode. Figure 7.4 shows the homing routine.

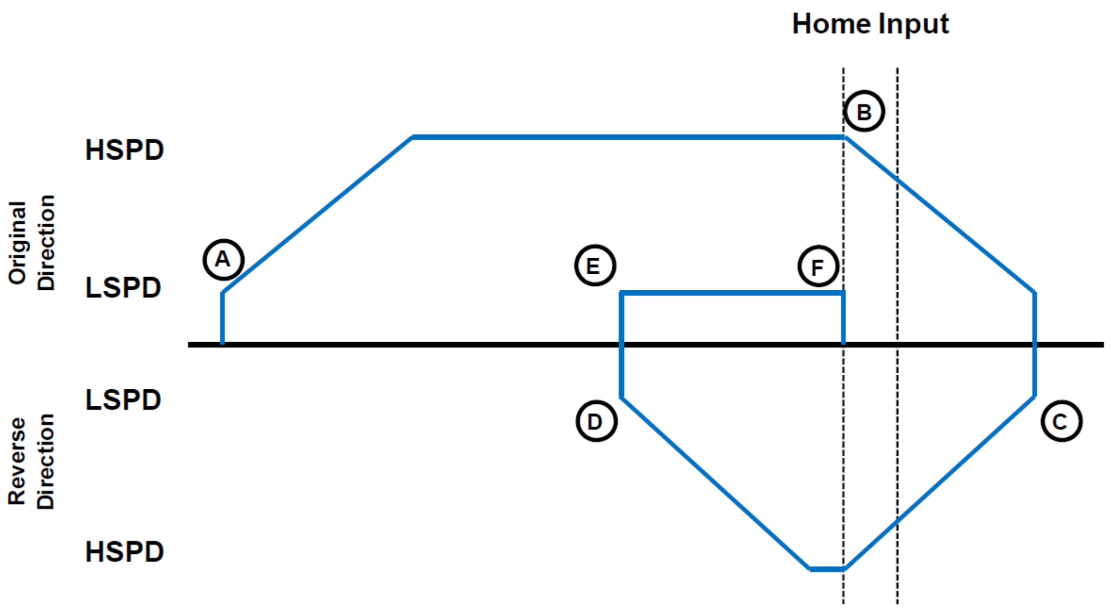


Figure 7.4

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 motor will decelerate to a stop.
C. Once deceleration is complete, the motor will accelerate and move in the opposite direction. The motor will move by the amount defined by the home correction amount (**HCA**). This length should cause the motor to clear and bypass the home input.
D. The motor has completed the home correction amount (**HCA**) move and decelerated to a stop.
E. The motor moves toward the home input again at low speed.
F. The home input is triggered again, the position counter is reset to zero and the motor stops immediately

ASCII	**HL+/-**	**HCA**
Standalone	**HLHOMEX+/-**	**-**

7.9.4. Limit Only

Use the **L+/-** command for ASCII mode or the **LHOMEX+/-** command for standalone mode. Figure 7.5 shows the homing routine.

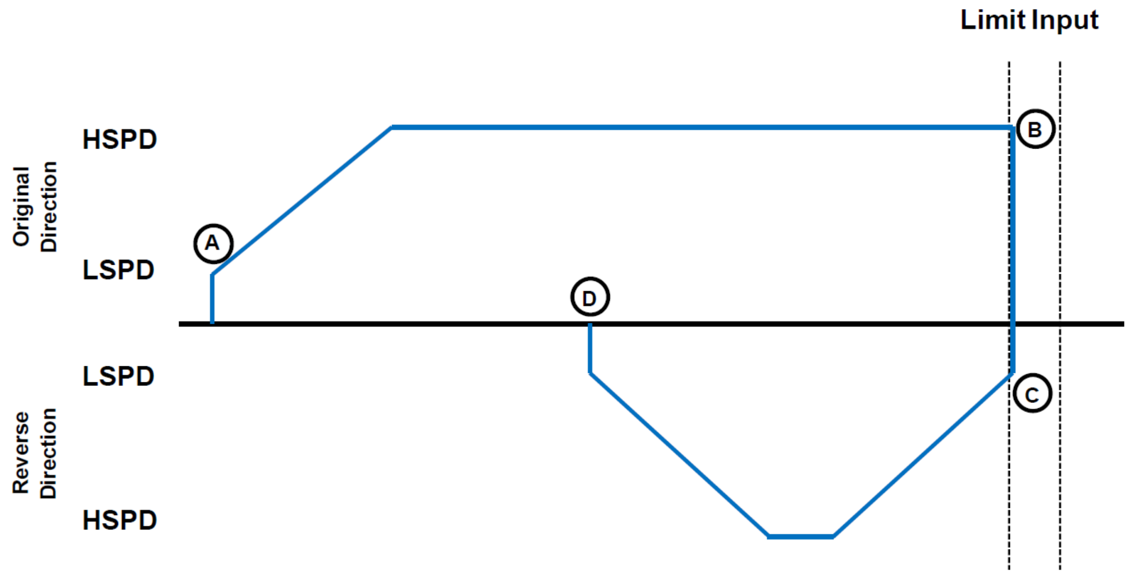


Figure 7.5

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.

ASCII	**L+/-**	**LCA**
Standalone	**LHOMEX+/-**	**-**

7.9.5. Z-Index Only

Use the **Z+/-** command for ASCII mode or the **ZOMEX+/-** command for standalone mode. Figure 7.6 shows the homing routine.

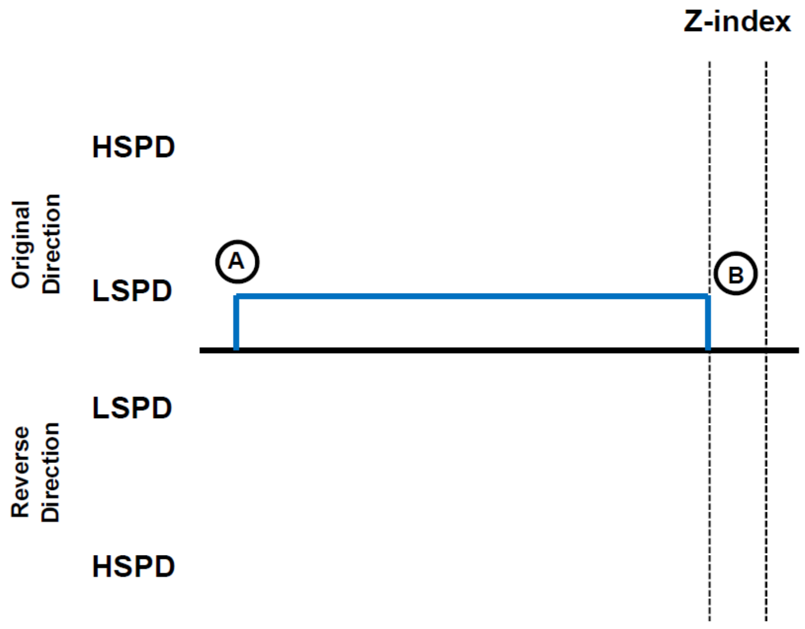


Figure 7.6

A. Issuing the command starts the motor at low speed.
B. Once the z-index pulse is found, the motor stops and the position is set to zero.

ASCII	**Z+/-**
Standalone	**ZOMEX+/-**

7.10. Limit Switch Function

Triggering 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 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 the limit error is set, the status error must be cleared, using the **CLR** command in ASCII mode or the **ECLEARX** command in the standalone mode, in order to move the motor again.

The limit error state can be ignored by setting **IERR=1**. In this case, the motor will still stop when the limit switch is triggered, however it will not enter an error state.

ASCII	**CLR**	**IERR**
Standalone	**ECLEARX**	**-**

7.11. Motor Status

Motor status can be read anytime using **MST** command. Table 7.0 shows the bit representation of the motor status.

Bit	Description
0	Motor running at constant speed
1	Motor in acceleration
2	Motor in deceleration
3	Home input switch status
4	Minus limit input switch status
5	Plus limit input switch status
6	Minus limit error. This bit is latched when minus limit is hit during motion. This error must be cleared before issuing any subsequent move commands.
7	Plus limit error. This bit is latched when plus limit is hit during motion. This error must be cleared before issuing any subsequent move commands.
8	Latch input status
9	Z-index status
10	TOC time-out status

Table 7.0

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

7.12. Digital Inputs/Outputs

DMX-K-SA comes with 6 digital inputs and 3 digital outputs.

7.12.1. Inputs

The digital input status of all 6 available inputs can be read with the DI command. Digital input values can also be referenced one bit at a time by using the **DI[1-6]** 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	Digital Input 3	DI3
3	Digital Input 4	DI4
4	Digital Input 5	DI5
5	Digital Input 6	DI6

Table 7.1

ASCII	**DI**	**DI[1-6]**
Standalone	**DI**	**DI[1-6]**

7.12.2. Digital Outputs

The **DO** command can be used to set the output voltage of all available digital outputs. The DO value must be within the range of 0-7.

Digital output values can also be referenced one bit at a time with the **DO[1-3]** 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 Command
0	Digital Output 1 (In Position)	DO1
1	Digital Output 2	DO2
2	Digital Output 3 (Alarm)	DO3

Table 7.2

If StepNLoop control and **EDO** are enabled, DO1 is used as an "In Position" status output, and DO3 is used as an "Alarm" output. See section 7.16 for details on StepNLoop control.

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 7.15 for details. Digital outputs are active low by default.

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.

ASCII	**DO**	**DO[1-3]**	**DOBOOT**	**EDO**
Standalone	**DO**	**DO[1-3]**	**-**	**-**

7.13. High Speed Latch Input

DMX-K-SA has high speed position latch feature that is triggered by DI2.

This input performs high speed position capture of both pulse and encoder positions but does not reset the pulse or encoder position counters.

When StepNLoop mode is enabled, the position value will be the current target position of the motor.

Use the **LT** command in ASCII mode or the **LTX** command in standalone mode to enable and disable latch feature. To read the latch status, use the **LTS** ASCII command or the **LTSX** standalone command. Table 7.3. details the value representation of the latch status.

Return Value	Description
0	Latch off
1	Latch on and waiting for latch trigger
2	Latch triggered

Table 7.3

Once the latch is triggered, the triggered positions can be retrieved using the **LTP** ASCII command or the **LTPX** standalone command (latched pulse position) and the **LTE** ASCII command or the **LTEX** standalone command (latched encoder position) commands.

ASCII	**LT**	**LTS**	**LTP**	**LTE**
Standalone	**LTX**	**LTSX**	**LTPX**	**LTEX**

7.14. Sync Output

DMX-K-SA has a designated synchronization digital output (DO2). The synchronization output is triggered when the encoder position value meets a defined condition. While this feature is enabled, the designated digital output (DO2) cannot be controlled by user.

Use **SYNO** command in ASCII mode or the **SYNONX** command in standalone mode to enable the synchronization output feature. Similarly the ASCII command **SYNF** or the standalone command **SYNOFFX** can be used to disable the synchronization output feature.

Use **SYNP** ASCII command or the **SYNPOSX** standalone command to read and set the synchronization position value. The synchronization output feature will use this position and the synchronization condition defined by the **SYNC** ASCII command or **SYNCFGX** standalone command to determine the output status. See table 7.4. for the available synchronization conditions.

Value	Description
1	Encoder position is equal to the sync position.
2	Encoder position is less than the sync position
3	Encoder position is greater than the sync position.
4	Encoder position changes by the sync position.

Table 7.4

Once the synchronization condition is met for options [1, 2, 3], the synchronization status will be set. The **SYNS** command in ASCII mode or the **SYNSTATX** in standalone mode can be used to query the status. Table 7.5 shows the return values of the synchronization status.

Value	Description
0	Sync output feature is off
1	Waiting for sync condition
2	Sync condition has occurred

Table 7.5

Synchronization option [4] will "auto reload' the condition to allow for a high speed synchronization output. This option will allow for 1,500 synchronization outputs per second.

ASCII	**SYNO**	**SYNF**	**SYNP**	**SYNC**	**SYNS**
Standalone	**SYNONX**	**SYNOFFX**	**SYNPOSX**	**SYNCFGX**	**SYNSTATX**

7.15. Polarity

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

Bit	Description
0	Reserved
1	Direction
2	Reserved
3	Reserved
4	Limit
5	Home
6	Latch
7	In Position Output
8	Alarm Output
9	Digital Output
10	Digital Input
11	Jump to line 0 on error
12	Enable Output

Table 7.6

The jump to line 0 polarity option defined by bit 11 indicates the return line once a standalone program has recovered from an error state. If this bit is low, the standalone program will return to the last processed line. If this bit is high, then it will return to the first line of the program.

All other polarity options indicate whether the input or output is active high or active low.

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

7.16. StepNLoop Closed Loop Control

DMX-K-SA features a closed-loop position verification algorithm called StepNLoop (SNL). The algorithm requires the use of an incremental encoder that is included in a standard DMX-K-SA.

SNL performs the following operations:

1) Position Verification: At the end of any targeted move, SNL will perform a correction if the current error is greater than the tolerance value.
2) Delta Monitoring: The delta value is the difference between the actual and the target position. When delta exceeds the error range value, the motor is stopped and the SNL Status goes into an error state. Delta monitoring is performed during moves – including homing and jogging. To read the delta value, use the **DX** command.

See Table 7.7 for a list of the SNL control parameters.

SNL Parameter	Description	Command
Tolerance	Maximum error between target and actual position that is considered "In Position". In this case, no correction is performed. Units are in encoder counts.	**SLT**
Error Range	Maximum error between target and actual position that is not considered a serious error. If the error exceeds this value, the motor will stop immediately and go into an error state.	**SLE**
Correction Attempt	Maximum number of correction tries that the controller will attempt before stopping and going into an error state.	**SLA**

Table 7.7

To enable/disable the StepNLoop feature use the **SL** ASCII command or the **SLX** standalone command. To read the StepNLoop status, use **SLS** ASCII command or the **SLSX** standalone command. See Table 7.8 for a list of the StepNLoop status return values.

Return Value	Description
0	Idle
1	Moving
2	Correcting
3	Stopping
4	Aborting
5	Jogging
6	Homing
7	Z-Homing
8	Correction range error. To clear this

	error, use **CLRS or CLR** command.
9	Correction attempt error. To clear this error, use **CLRS or CLR** command.
10	Stall Error. **DX** value has exceeded the correction range value. To clear this error, use **CLRS or CLR** command.
11	Limit Error
12	N/A (i.e. SNL is not enabled)
13	Limit homing

Table 7.8

Depending on the value of the delta, the StepNloop algorithm can have certain behaviors. See Table 7.9 for StepNLoop behavior within different scenarios.

Condition	**SNL behavior (motor is moving)**	**SNL behavior (motor is idle)**
δ <= SLT	Continue to monitor the **DX**	In Position. No correction is performed.
δ > SLT AND δ < SLE	Continue to monitor the **DX**	Out of Position. A correction is performed.
δ > SLT AND δ > SLE	Stall Error. Motor stops and signals an error.	Error Range Error. Motor stops and signals an error.
Correction Attempt > SLA	NA	Max Attempt Error. Motor stops and signals an error.

Table 7.9

Key

[δ]:	Error between the target position and actual position
SLT:	Tolerance range
SLE:	Error range
SLA:	Max correction attempt

Once StepNLoop is enabled, position move commands are in term of encoder position. For example, X1000 means to move the motor to encoder 1000 position.

Additionally, once SNL is enabled, the speed is in encoder speed. For example HSPD=1000 when SNL is enabled means that the target high speed is 1000 encoder counts per second.

If **EDO** is enabled while SNL is enabled, DO1 is dedicated as the “In Position” output and DO3 is dedicated as the “Alarm” output. In order to use the digital outputs for general purpose, disable **EDO** by setting **EDO=0**.

ASCII	**SL**	**SLS**	**SLT**	**SLA**	**SLE**	**EDO**
Standalone	**SLX**	**SLSX**	**-**	**-**	**-**	**-**

7.17. Idle Current and Run Current

DMX-K-SA allows for two current settings. The run current is used while the motor is running. The **CURR** 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 **CURI** command. The run and idle current range must be within 0mA to 2500mA. 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 **CURT** command. Units are in milli-seconds. The actual current of the motor can be read using the **CUR** command.

ASCII	**CURR**	**CURI**	**CURT**	**CUR**
Standalone	**-**	**-**	**-**	**-**

7.18. Auto-response Feature

It is possible to have the DMX-K-SA automatically send a message out on the RS-232/485 bus once it is in position. This feature prevents the master from having to constantly poll the position status of the DMX-K-SA.

Response format: $[Device Address]P[CR]

Example:

1) Device 01 is given an absolute move command from the master.
2) Once the device finishes the move, it sends the following command out on the bus: $01P[CR].

To enable/disable the feature, use the **AR** command.

To write the auto-response parameter to flash memory, use the **STORE** command. After a complete power cycle, the setting will take effect. Note that before a power cycle is done, the setting will not take effect.

ASCII	**AR**
Standalone	**-**

7.19. Communication Time-out Feature (Watchdog)

DMX-K-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 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=0**.

ASCII	**TOC**
Standalone	**TOC**

7.20. Standalone Programming

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

Standalone programs can be written to the DMX-K-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-K-SA. Each line of written standalone code creates 1-4 assembly lines of code after compilation

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

7.20.1. 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.

7.20.2. Standalone Control

The DMX-K-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 7.10

7.20.3. Standalone Status

The **SASTAT[0-1]** command can be used to determine the current status of the specified standalone program. Table 7.11 details the return values of this command.

Value	Description
0	Idle
1	Running
2	Paused

3	N/A
4	Errored

Table 7.11

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].

7.20.4. Standalone Subroutines

The DMX-K-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 subroutines can be shared by both standalone programs. Refer to section 10 for 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.

7.20.5. Error Handling

Subroutine 31 is designated for error handling. If an error occurs during standalone execution (i.e. limit error, StepNLoop 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 contains 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 7.6 for details.

7.20.6. Standalone Variables

The DMX-K-SA has 100 32-bit signed standalone variables available for general purpose use. They can be used to perform basic calculations and support integer operations. The **V[0-99]** 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 7.12

Variables V50 through V99 can be stored to flash memory using the **STORE** command. Variables V0-V49 will be initialized to zero on power up.

7.20.7. 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 to save the setting to flash memory. It will take effect on the following power cycle. See description in Table 7.13 for the bit assignment of the **SLOAD** setting.

Bit	Description
0	Standalone Program 0
1	Standalone Program 1

Table 7.13

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]**	**-**

7.21. Storing to Flash

The following items are stored to flash:

ASCII Command	Description
AR	Auto-response for in-position
CURI,CUR,CURT	Driver current settings
DB	Serial communication baud rate
DN	Device name
DNM	[1]Modbus device number
DOBOOT	DO configuration at boot-up

EDEC	Unique deceleration enable
EDO	Enable in-pos/alarm outputs
EOBOOT	EO configuration at boot-up
HCA	Home correction amount
IERR	Ignore limit error enable
LCA	Limit correction amount
POL	Polarity settings
RSM	[1]Modbus enable
RT	ASCII response type
RZ	Return to zero position after homing
SL, SLE, SLT, SLA	StepNLoop parameters
SLOAD	Standalone program run on boot-up parameter
TOC	Time-out counter reset value
V50-V99	Note that on boot-up, V0-V49 are reset to value 0

Table 7.14

[1] See "Modbus_Addition_Addendum_A" document for details on the Modbus protocol.

When a standalone program is downloaded, the program is immediately written to flash memory.

7.22. Default Settings

Following are the factory default settings when then unit is shipped from the factory.

Command	Parameter Description	Value
AR	Auto-response for in-position	Disabled
CURI	Idle current	1000 mA
CURR	Run current	1600 mA
CURT	Idle time	500 mSec
DB	Baud rate	9600
DN	Device ID	DMK01
DOBOOT	Digital output boot-up state	7
EDO	Alarm/ in position output mode	Enabled
EOBOOT	Enable output boot-up state	0
HCA	Home correction amount	1000
IERR	Ignore error state	Disabled
LCA	Limit correction amount	1000
POL (bit 1)	Direction polarity	CW
POL (bit 4)	Limit polarity	Active Low

POL (bit 5)	Home polarity	Active Low
POL (bit 6)	Latch polarity	Active Low
POL (bit 7)	In position output polarity	Active Low
POL (bit 8)	Alarm output polarity	Active Low
POL (bit 9)	Digital output polarity	Active Low
POL (bit 10)	Digital input polarity	Active Low
POL (bit 11)	Jump to line 0 on error	Disabled
POL (bit 12)	Motor enable	Active Low
RT	Response type	Do Not Append
RZ	Return to home position	Disabled
SL	StepNLoop enable	Enabled
SLA	StepNLoop maximum attempt	10
SLE	StepNLoop error range	1000
SLOAD	Run program(s) on power up	0
SLT	StepNLoop tolerance range	20
TOC	Time-out counter value (Watch-dog)	0
V0-V99	Volatile and non-volatile variables	0

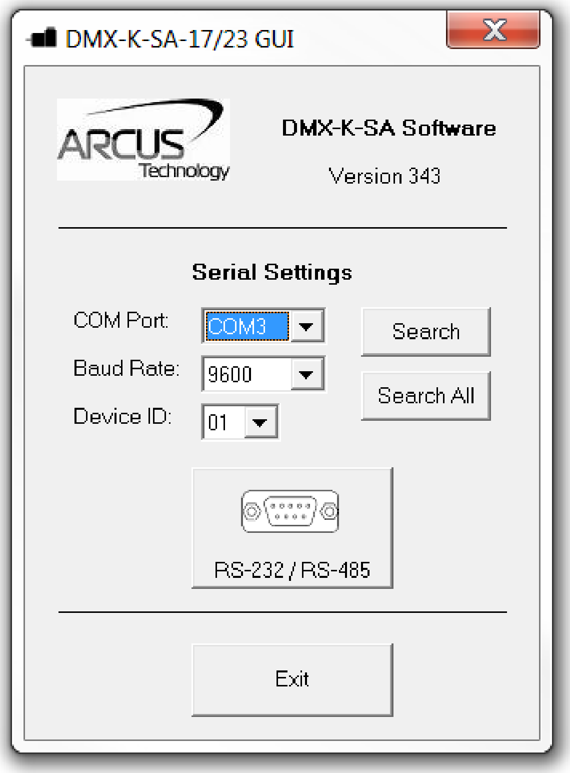
Table 7.15

8. Software Overview

The DMX-K-SA has a Windows compatible software that allows for RS232 or RS485 communication. Standalone programming, along with all other available features of the DMX-K-SA, will be accessible through the software. It can be downloaded from the Arcus Technology website.

To communicate over RS232/RS485, make sure that the DMX-K-SA is connected to the COM port.

Startup the DMX-K-SA GUI program and you will see the following screen in figure 8.0. This will allow the user to search for all the motors that are currently connected to the RS232/RS485 bus.



COM Port: COM3
Baud Rate: 9600
Device ID: 01

Figure 8.0

The first DMX-K-SA connected to the PC can be found using the Search button. If there all multiple DMX-K-SA connected to the PC, the Search All button can be used to find them.

If the search fails, or you are unable to open a connection, check the following:

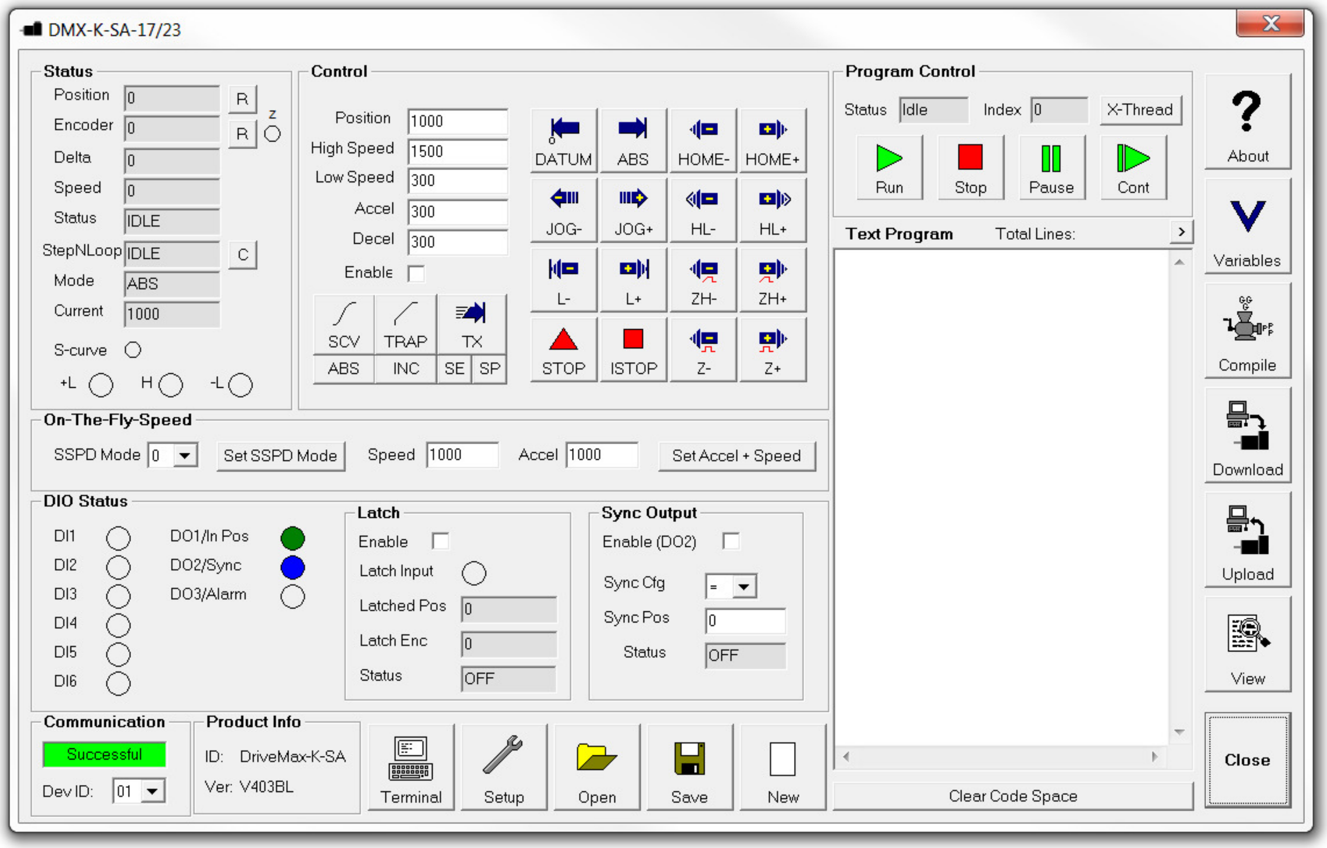
- Check power supply to DMX-K-SA. Allowable power is range is from 12VDC to 35VDC.
- Check communication wiring. If using RS-232, TXD from DMX-K-SA should be connected to RXD of the serial port and RXD from DMX-K-

SA should be connected to TXD of serial port. If using RS-485, make sure that the 485+ from DMX-K-SA is connected to 485+ of the master and 485- from DMX-K-SA is connected to 485- of the master.

- Confirm that the device name is set correctly. Default factory device name setting is "01". If this name has been changed and stored to flash, enter the new name.

Once the correct serial settings have been determined, the RS-232/RS0485 button can be used to open the Main Control Screen.

8.1. Main Control Screen



Status
Position 0
Encoder 0
Delta 0
Speed 0
Status IDLE
StepNLoop IDLE
Mode ABS
Current 1000
S-curve
+L H -L

Control
Position 1000
High Speed 1500
Low Speed 300
Accel 300
Decel 300
Enable

On-The-Fly-Speed
SSPD Mode 0
Speed 1000
Accel 1000

DIO Status
DI1 DI2 DI3 DI4 DI5 DI6
DO1/In Pos ●
DO2/Sync ●
DO3/Alarm

Latch
Enable
Latch Input
Latched Pos 0
Latch Enc 0
Status OFF

Sync Output
Enable (DO2)
Sync Cfg =
Sync Pos 0
Status OFF

Program Control
Status Idle
Index 0

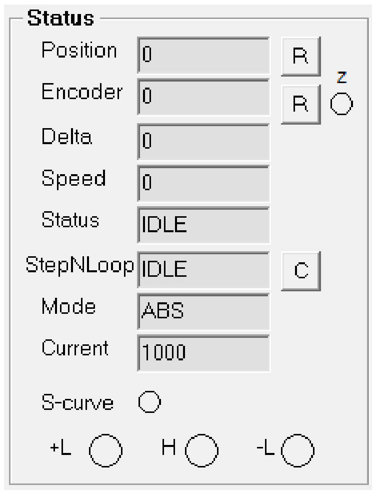
Text Program
Total Lines:

Communication
Successful
Dev ID: 01

Product Info
ID: DriveMax-K-SA
Ver: V403BL

Figure 8.1

8.1.1. Status


Status

Position	0
Encoder	0
Delta	0
Speed	0
Status	IDLE
StepNLoop	IDLE
Mode	ABS
Current	1000

Figure 8.2

1. **Position** – displays the current pulse position counter. When StepNLoop is enabled, this displays the Target position. This value can be reset to 0 with the "R" button.
2. **Encoder** – displays the current encoder position counter. This value can be reset to 0 with the "R" button.
    - **Z** – Encoder Z index channel status is displayed.
3. **Delta** – valid only for StepNLoop. Displays the difference between the target position and the actual position.
4. **Speed** – displays the current pulse speed output rate. Value is in pulses/second. While the controller is in StepNLoop mode, this value shows encoder counts/second.
5. **Status** – displays current motor status by displaying one of the following status:
    - IDLE: motor is not moving
    - ACCEL: motion is in acceleration
    - DECEL: motion is in deceleration
    - CONST: motion is in constant speed
    - -LIM ERR: minus limit error
    - +LIM ERR: plus limit error
6. **StepNLoop** – valid only when StepNLoop is enabled and displays current StepNLoop status. Any error can be cleared using the "C" button.
    - NA: StepLNoop is disabled
    - IDLE: motor is not moving
    - MOVING: target move is in progress
    - JOGGING: jog move is in progress
    - HOMING: homing is in progress
    - LHOMING: limit homing in progress
    - Z-HOMING: homing using Z-index channel in progress

- ERR-STALL: StepNLoop has stalled.
- ERR-LIM: plus/minus limit error

7. **Mode** – displays current move mode
   - ABS: all the move commands by X[pos] command will be absolute moves
   - INC: all the move commands by X[pos] command will be increment moves.
8. **Current** – Actual driver current
9. **S-curve Status** – Displays whether the moves are in trapezoidal or S-curve acceleration.
10. **Limit/Home Input Status** – Limit and Home input status.

8.1.2. Control

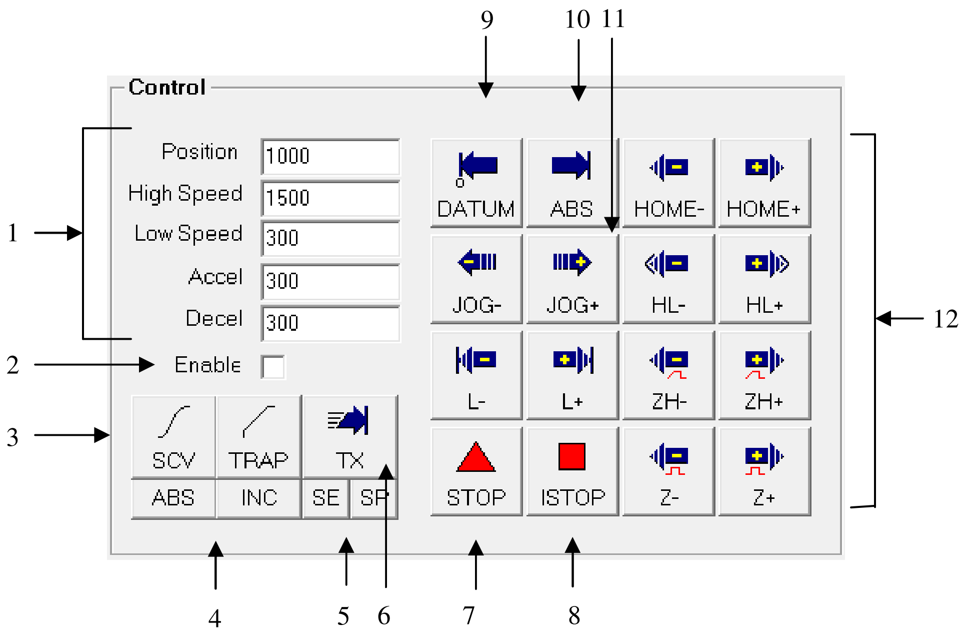


Figure 8.3

1. **Target Position/Speed/Accel**
   - Position: use this to set the target position. For normal open loop mode, this position is the pulse position and when StepNLoop is enabled this target position is in encoder position.
   - High/Low Speed: use this to set the speed of the move. For normal open loop mode, this value is in pulses/second and when StepNLoop is enabled this value is in encoder counts/second.
   - Accel: acceleration value in milliseconds.
   - Decel: deceleration value in milliseconds.
2. **Enable Driver Power** – use this button to enable and disable the power to the micro-step driver.
3. **Select Acceleration Mode** – use these buttons to select trapezoidal or S-curve acceleration mode.

4. **Select Move Mode** – use these buttons to select absolute or incremental move mode.
5. **Set Position** – use these buttons to set the encoder or pulse position to “Position” value.
6. **On-the-fly target change** – Change the target position on-the-fly
7. **Ramp Stop** – use this button to stop the motion with deceleration.
8. **Immediate Stop** – use this button to stop the motion immediately. *We recommend that ramp stop be used whenever possible to reduce the impact to the motor and the system.*
9. **Move back to zero** – use this to move the motor to the zero target position. (zero encoder position when in StepNLoop and zero pulse position when in open loop).
10. **Perform Absolute Move** – use this to move the motor to the target position. When in absolute mode, the axis will move to the absolute target position. When in incremental mode, the axis will move incrementally.
11. **Jogging** – jog motor in either positive or negative direction
12. **Perform Homing** – Five different homing routines are available
    - HOME: homing is done using only the home switch.
    - HL: homing is done using the home switch + a low speed
    - L: homing is done using the limit switch
    - ZH: homing is done using the home switch first and then the Z index channel of the encoder.
    - Z: homing is done only using the Z index channel of the encoder.

8.1.3. On-The-Fly Speed Change

Set the speed on the fly. On the fly speed change feature can only be used if the controller is already in motion.

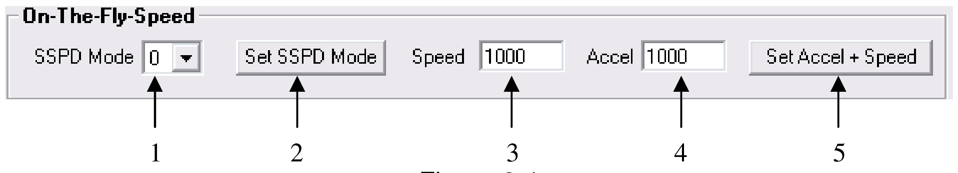
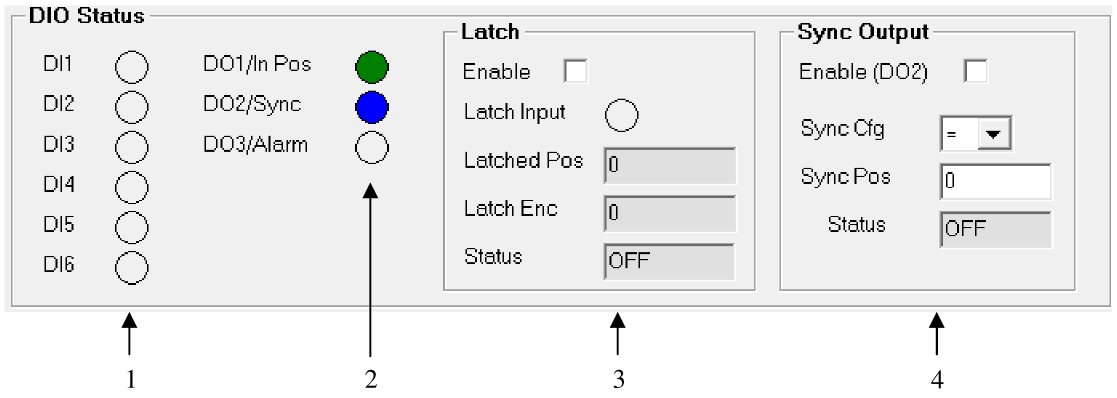


Figure 8.4

1. **SSPD Mode** – Before the controller starts motion, set the SSPDM parameter. To see which value to use, see the on-the-fly speed change section.
2. **Set SSPDM Mode** – Set the SSPDM parameter. Note that if an on-the-fly speed change operation is to be used, this parameter must be set before the controller goes starts motion.
3. **Speed** – Once the “Set Speed” button is clicked, the speed will change on-the-fly to the desired speed.

4. **Accel** – The acceleration/deceleration use for the on-the-fly speed change operation.
5. **Set Accel + Speed** – Start the on-the-fly speed operation.

8.1.4. DIO Status



Latched Pos: 0
Latch Enc: 0
Status: OFF
Sync Cfg: =
Sync Pos: 0
Status: OFF

Figure 8.5

1. **Digital Input Status** – displays the current digital input status.
2. **Digital Output Status and Control** – digital outputs
   a. DO1 - When StepNLoop is enabled, DO1 is used as In-Position output unless **EDO=0**.
   b. DO2 - When Sync Output is enabled, DO2 is used for Sync Digital Output function.
   c. DO3 - When StepNLoop is enabled, DO3 is used as Alarm output unless **EDO=0**.
3. **Latch** - encoder and pulse positions can be captured/latched with an input trigger. DI2 is used as the latch input.
4. **Sync Output** – digital outputs can be triggered.

8.1.5. Communication

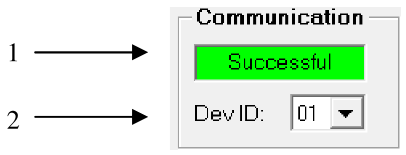
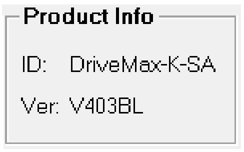


Figure 8.6

This section displays the current communication status with the selected device. The current device ID the software is using is also displayed. To communicate with a different DMX-K-SA on-the-fly, select another ID number from this drop-down box.

8.1.6. Product Info


Product Info

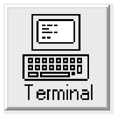
ID: DriveMax-K-SA

Ver: V403BL

Figure 8.7

Displays the product ID as well as the firmware version

8.1.7. Terminal



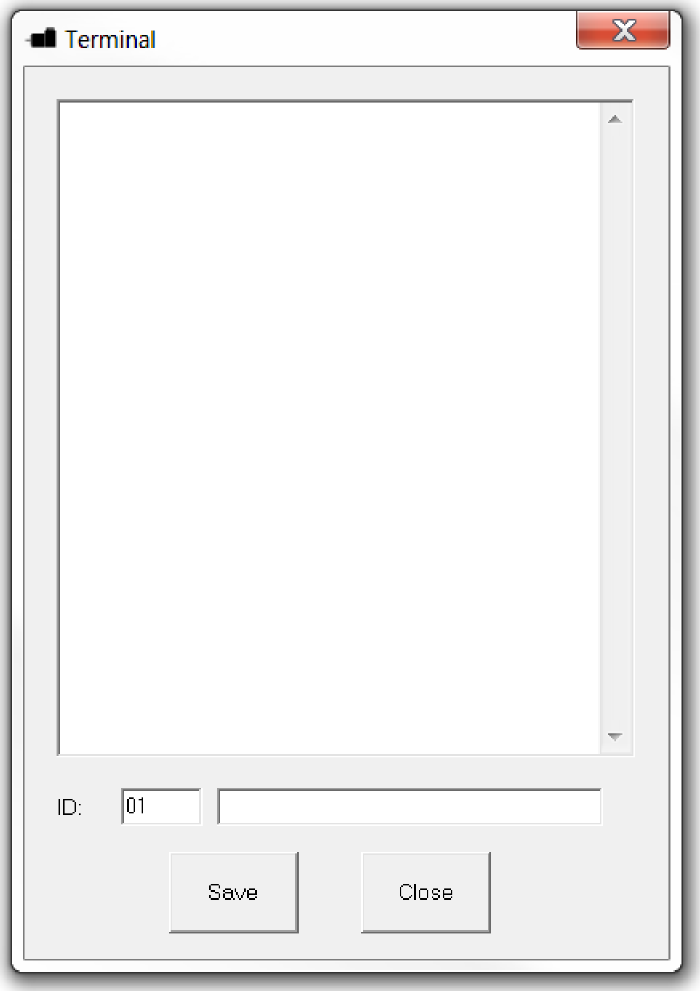
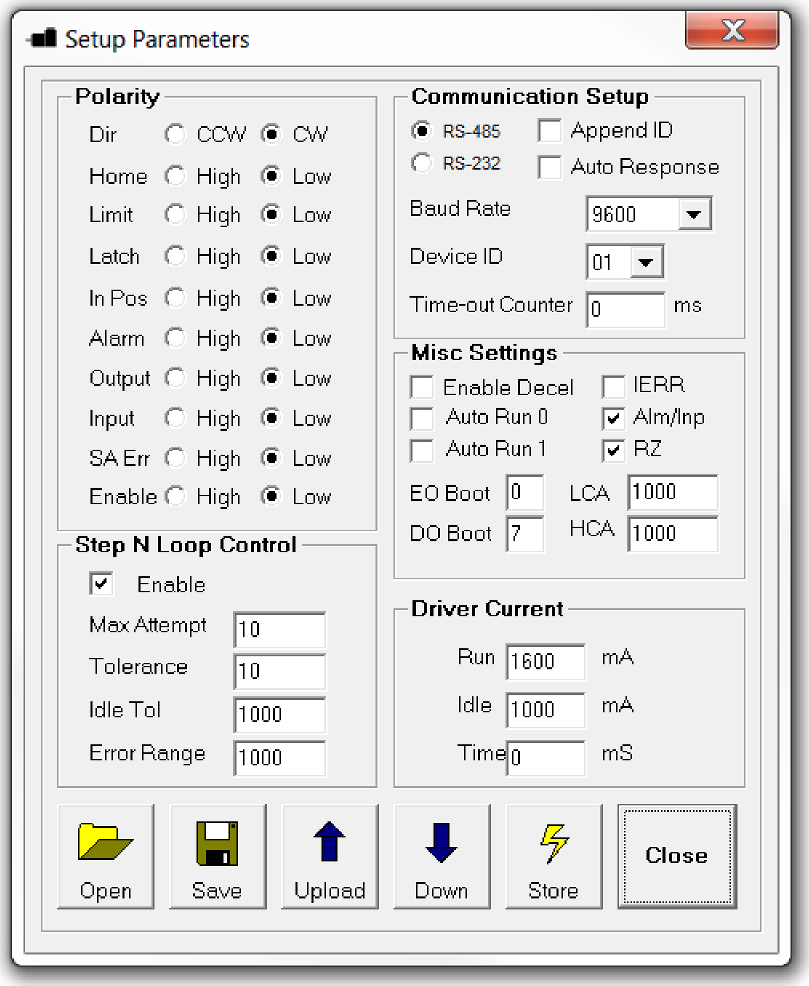


Figure 8.8

Terminal dialog box allows manual testing of the commands. The command with the desired device ID can be entered into the command line. The response from the DMX-K-SA will be displayed in the response window.

8.1.8. Setup



Polarity

Dir	CCW	[✓ CW]
Home	High	[✓ Low]
Limit	High	[✓ Low]
Latch	High	[✓ Low]
In Pos	High	[✓ Low]
Alarm	High	[✓ Low]
Output	High	[✓ Low]
Input	High	[✓ Low]
SA Err	High	[✓ Low]
Enable	High	[✓ Low]

Step N Loop Control

☑ Enable

Max Attempt 10

Tolerance 10

Idle Tol 1000

Error Range 1000

Communication Setup

[✓ RS-485] RS-232

☐ Append ID

☐ Auto Response

Baud Rate 9600

Device ID 01

Time-out Counter 0 ms

Misc Settings

☐ Enable Decel ☐ IERR

☐ Auto Run 0 ☑ Alm/Inp

☐ Auto Run 1 ☑ RZ

EO Boot 0 LCA 1000

DO Boot 7 HCA 1000

Driver Current

Run 1600 mA

Idle 1000 mA

Time 0 mS

Figure 8.9

1. **Polarity Setup** – The following polarity parameters can be configured
   - Dir: direction of the motion (clockwise or counter-clockwise)
   - Home: home input polarity
   - Limit: limit input polarity
   - Latch: latch input polarity

- In Pos: In position output (used when StepNLoop is enabled and EDO=1)
- Alarm: Alarm output (used when StepNLoop is enabled and EDO=1)
- Output: digital output polarity
- Input: digital input polarity
- SA Err: stand-alone error jump line.
  - Low: jump to previous line
  - High: jump to line 0
- Enable: motor enable output polarity

2. **StepNLoop Control** – Using the encoder input, StepNLoop control allows closed loop position verification and correction for the moves. See StepNLoop control section 7.16 for details.
3. **Communication Setup**
   - Select RS-485 or RS-232 communication
   - Device ID appended to controller's response
   - Enable/Disable in position auto-response feature
   - Serial communication baud rate can be selected to support different communication speed.
   - Device ID configuration allows multiple devices on the serial communication network.
   - Time-out counter is a watch-dog timer for communication (ms)
4. **Misc Settings**
   - Enable Decel: Allow for unique deceleration and acceleration values
   - Auto Run 0: Run stand-alone program 0 on boot-up.
   - Auto Run 1: Run stand-alone program 1 on boot-up.
   - IERR: Ignore limit error
   - Alm/Inp: Digital output alarm / in-position.
   - RZ: Return to zero position after homing routines
   - EO Boot: Configure enable output boot-up state
   - DO Boot: Configure digital output boot-up state
   - LCA: Set limit correction amount
   - HCA: Set home correction amount
5. **Driver Setting** – The following micro-step driver settings can be configured:
   - Run Current: 100mA to 2.5Amp
   - Idle Current: 100mA to 2.5Amp
   - Idle Time:0 to 100 centi-second (10 centi-second = 1 second)
6. **Open/Save** – Configuration values can be saved to a file and read from a file.
7. **Upload/Download** – Configuration values can be uploaded and downloaded. Note that if the configuration values are changed, they need to be downloaded to take effect.
8. **Store** – The downloaded parameters can be permanently stored on the non-volatile memory.

8.1.9. Standalone Program File Management

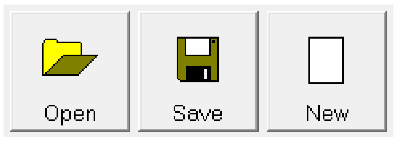


Figure 8.10

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

8.1.10. Standalone Program Editor

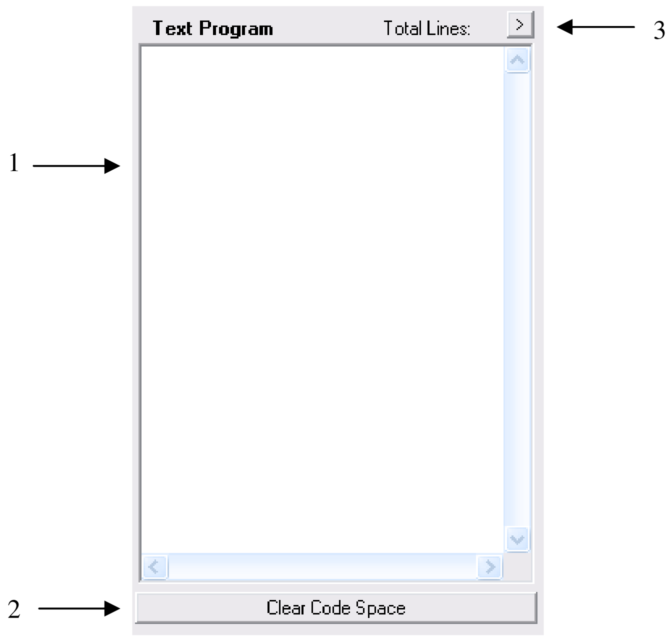


Figure 8.11

1. Write the standalone program in the Program Editor
2. Use this button to remove the current standalone program
3. Use this button to open a larger and easier to manage program editor.

8.1.11. Standalone Processing

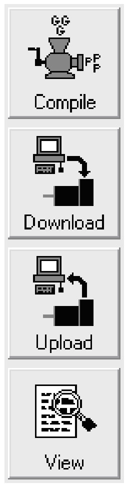


Figure 8.12

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

8.1.12. Program Control

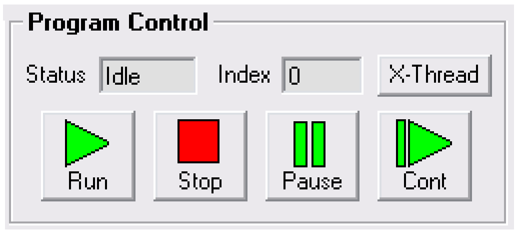
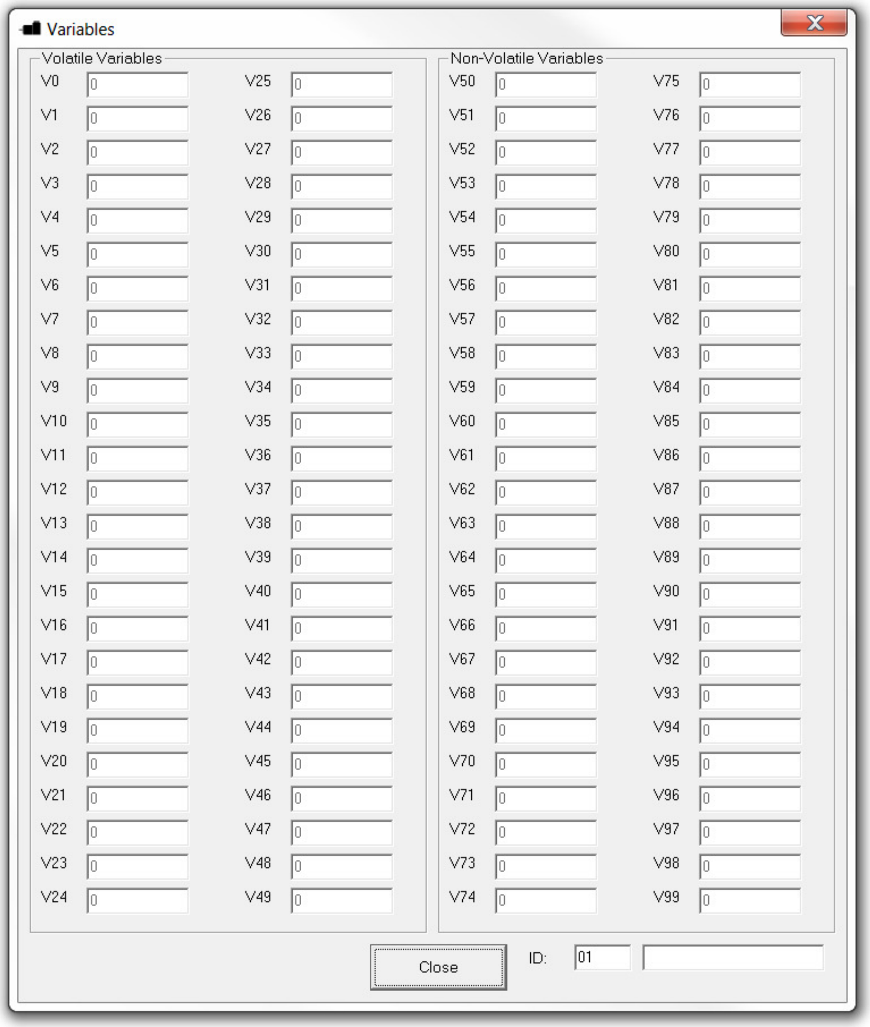


Figure 8.13

1. **Status** – The program status is shown here. Idle, Running, Errored, and Paused are possible program statuses.
2. **Index** – The program that is downloaded is in the form of low-level code. Each line of the low level code has a line number which shows here.
3. **Run** – Run the program.
4. **Stop** – Stop the program.
5. **Pause** – Pause the program if it is already running.
6. **Continue** – Continue the program if it has been paused.
7. **X-Thread –** Open the Program Control for standalone multi-thread operation.

8.1.13. Variable Status



Volatile Variables				Non-Volatile Variables			
V0	0	V25	0	V50	0	V75	0
V1	0	V26	0	V51	0	V76	0
V2	0	V27	0	V52	0	V77	0
V3	0	V28	0	V53	0	V78	0
V4	0	V29	0	V54	0	V79	0
V5	0	V30	0	V55	0	V80	0
V6	0	V31	0	V56	0	V81	0
V7	0	V32	0	V57	0	V82	0
V8	0	V33	0	V58	0	V83	0
V9	0	V34	0	V59	0	V84	0
V10	0	V35	0	V60	0	V85	0
V11	0	V36	0	V61	0	V86	0
V12	0	V37	0	V62	0	V87	0
V13	0	V38	0	V63	0	V88	0
V14	0	V39	0	V64	0	V89	0
V15	0	V40	0	V65	0	V90	0
V16	0	V41	0	V66	0	V91	0
V17	0	V42	0	V67	0	V92	0
V18	0	V43	0	V68	0	V93	0
V19	0	V44	0	V69	0	V94	0
V20	0	V45	0	V70	0	V95	0
V21	0	V46	0	V71	0	V96	0
V22	0	V47	0	V72	0	V97	0
V23	0	V48	0	V73	0	V98	0
V24	0	V49	0	V74	0	V99	0

ID: 01

Figure 8.14

View the status of variables 0-99. Note that this window is read-only. A command line is available to send commands to the DMX-K-SA.

9. ASCII Language Specification

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

DMX-K-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 RS-232 and RS-485 communication, the commands detailed in table 9.0 are valid.

9.1. ASCII Command Set

Command	Description	Return
ABORT	Immediately stops the motor if in motion	OK
ABS	Set move mode to absolute mode	OK
ACC	Return current acceleration value in milliseconds	milliseconds
ACC=[Value]	Sets acceleration value in milliseconds	OK
AR	Return auto-response feature status	[0,1]
AR=[0 or 1]	Set auto-response feature status	OK
CLR	Clears limit error as well as StepNLoop error	OK
CUR	Return real-time current	[0-2500]mA
CURI	Return idle current setting	[0-2500]mA
CURI=[Value]	Set idle current of the motor [0-2500]mA	OK
CURR	Return run current setting	[0-2500]mA
CURR=[Value]	Set run current or the motor [0-2500]mA	OK
CURT	Return driver idle time setting	milliseconds
CURT=[Value]	Set driver idle time setting	OK
DB	Return the current baud rate of the device	Refer to Table 6.1
DB=[Value]	Set the baud rate of the device	OK
DEC	Return deceleration value in milliseconds.	milliseconds
DEC=[Value]	Set deceleration value in milliseconds	OK
DI	Return status of digital inputs	Refer to Table 7.1
DI[1-6]	Return individual bit status of digital inputs	0,1
DO	Return status of digital outputs	Refer to Table 7.2
DO=[Value]	Set digital outputs. Refer to Table 7.2	OK
DO[1-3]	Return status of individual digital output	Refer to Table 7.2
DO[1-3] = [Value]	Set individual digital output. Refer to Table 7.2	OK
DOBOOT	Return the DO configuration at boot-up	[0-7]
DOBOOT=[Value]	Set DO configuration at boot-up	OK
DN	Return device name	[DMK01-DMK99]
DN=[Value]	Set device name	OK
DNM	Return Modbus device number	1-127
DNM=[Value]	Set Modbus device number	OK
DX	Returns the delta value during StepNLoop control	28-bit number
EO	Returns driver power enable 0 - Motor power enabled	[0,1]

	1 - Motor power disabled	
EO=[0,1]	Enable (1) or disable (0) motor power	OK
EOBOOT	Return the EO configuration at boot-up	[0,1]
EOBOOT=[0,1]	Set the EO configuration at boot-up	OK
EDEC	Return the unique deceleration setting	[0,1]
EDEC=[0,1]	Set the unique deceleration setting	OK
EDO	Returns enable alarm/in pos mode status 0 - Disabled 1 - Enabled	[0,1]
EDO=[0,1]	Enable (1) or disable (0) alarm/in pos mode	OK
EX	Returns current encoder counter value	28-bit number
EX=[Value]	Sets the current encoder counter value	OK
GS[0-31]	Call a subroutine that has been previously stored to flash memory	OK
HSPD	Returns high speed setting	PPS
HSPD=[Value]	Set the high speed setting	OK
H+	Homes the motor in positive direction	OK
H-	Homes the motor in negative direction	OK
HCA	Returns the home correction amount	28-bit number
HCA=[Value]	Sets the home correction amount	OK
HL+	Homes the motor at low and high speed in positive direction	OK
HL-	Homes the motor at low and high speed in negative direction	OK
IERR	Return the ignore limit error setting	[0,1]
IERR=[0,1]	Set the ignore limit error setting	OK
ID	Return the product ID	DriveMax-K-SA
INC	Set move mode to incremental mode	OK
J+	Jogs the motor in positive direction	OK
J-	Jogs the motor in negative direction	OK
L+	Limit home the motor in positive direction	OK
L-	Limit home the motor in negative direction	OK
LCA	Return the limit correction amount	28-bit number
LCA=[Value]	Set the limit correction amount	OK
LSPD	Return the low speed setting	PPS
LSPD=[Value]	Set the low speed setting	OK
LT=[0,1]	Enable or disable position latch feature	OK
LTE	Return the latched encoder position	28-bit number
LTP	Return the latched pulse position	28-bit number
LTS	Return the latch status	See Table 7.3
MM	Return move mode status 0 - Absolute mode 1- Incremental mode	[0,1]
MST	Return the current motor status	See Table 7.0
POL	Return the polarity setting	See Table 7.6
POL=[value]	Sets polarity	OK
PS	Return the current pulse speed	PPS
PX	Return the current position value	28-bit number
PX=[value]	Set the current position value	OK
RSM	Return the Modbus enable setting	[0,1]
RSM= [0,1]	Set the Modbus enable setting	OK
RT	Return the response type setting	[0,1]
RT= [0,1]	Set the response type setting	OK
RZ	Return the return to zero setting	[0,1]

RZ=[0,1]	Set the return to zero setting	OK
SASTAT[0,1]	Return standalone program status. Refer to Table 7.11	[0-4]
SA[0-1274]	Return the standalone line	
SA[0-1274]=[Value]	Set the standalone line	OK
SCV	Returns the s-curve control 0 - Trapezoidal motion profile 1 - S-curve motion profile	[0,1]
SCV=[0,1]	Set the s-curve enable setting	OK
SL	Return the StepNLoop enable status 0 - Disable StepNLoop 1 - Enable StepNLoop	[0,1]
SL=[0,1]	Enable or disable StepNLoop Control	OK
SLA	Return the maximum number of StepNLoop control attempts	32-bit number
SLA=[value]	Set the maximum number of StepNLoop control attempt	OK
SLE	Return the StepNLoop correction range value	28-bit number
SLE=[value]	Set the StepNLoop correction range value	OK
SLS	Return the current StepnLoop control status	Refer to Table 7.8
SLT	Return the StepNLoop tolerance value	32-bit
SLT=[value]	Set the StepNLoop tolerance value	OK
SLOAD	Return the standalone program run on boot parameter. See Table 7.13	[0-3]
SLOAD=[value]	Set the standalone program run on boot parameters. See Table 7.13	OK
SPC[0,1]	Return program counter for the specified standalone program	[0-1274]
SR[0,1]=[Value]	Control standalone program. Refer to Table 7.10	OK
SSPD[value]	Perform an on-the-fly speed change	OK
SSPDM	Return the on-the-fly speed change mode	[0-9]
SSPDM=[value]	Set the on-the-fly speed change mode	OK
STOP	Stop the motor using deceleration if in motion	OK
STORE	Store settings to flash. Refer to Table 7.14	OK
SYNC	Return the current sync output configuration. Refer to Table 7.4	[0-4]
SYNC=[0-4]	Set the sync output configuration. Refer to table 7.4	OK
SYNF	Turn off sync output	OK
SYNO	Turn on sync output	OK
SYNP	Return the sync output trigger position	28-bit number
SYNP=[Value]	Set the sync output trigger position	28 bit number
T[Value]	Perform an on-the-fly target position change	OK
TOC	Return the time-out counter (ms)	32-bit number
TOC=[Value]	Set the time-out counter (ms)	OK
V[0-99]	Read variables 0-99	32-bit number
V[0-99]=[Value]	Set variables 0-99	OK
VER	Return firmware version	VXXXBL
X[Value]	Perform a positional move	OK
Z+	Home the motor in positive direction using the Z index encoder channel	OK
Z-	Homes the motor in negative direction using the Z index encoder channel	OK
ZH+	Homes the motor in positive direction using the home switch and then Z index encoder channel	OK

ZH-	Homes the motor in negative direction using the home switch and then Z index encoder channel	OK

Table 9.0

9.2. Error Codes

If an ASCII command cannot be processed by the DMX-K-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 controller
?Bad SSPD Command	SSPD move parameter is invalid
?Enable RS-485 Mode	Modbus mode cannot be set because communication needs to first be set to RS-485
?Index out of Range	The index for the command sent to the controller is not valid
?Motor is not moving	T[] command is invalid because a target position move is not in operation
?Moving	A move or position change command is sent while the controller is outputting pulses
?SCV ON	Cannot perform SSPD move because s-curve is enabled
?Speed out of range	SSPD move parameter is out of the range of the SSPDM speed window
?State Error	A move command is issued while the controller is in error state
?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

DEC	R/W	Set/get the deceleration setting. Unit is in milliseconds.	DEC=500
			DEC=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[1-6]	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[1-3]	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
DRVIC	W	Set the idle current of the motor. Unit is in mA.	DRVIC=100 ;Set to 100mA
DRVIT	W	Set the idle time of the motor. Unit is in milliseconds.	DRVIT=1000 ;Set to 1000mS
DRVRC	W	Set the run current of the motor. Unit is in mA.	DRVRC=1500 ;Set to 1500mA
ECLEARX	W	Clear any motor status errors.	ECLEARX
EO	R/W	Set/get the enable output status.	EO=1 ;Enable the motor

EX	R/W	Set/get the current encoder position.	EX=1000 ;Set to 1000
			V1=EX ;Read current encoder

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 7.9.3 for details.	HLHOMEX+ ;positive home WAITX ;wait for home move
HOMEX[+/-]	W	Home the motor using the home input at high speed in specified direction. See section 7.9.1 for details.	HOMEX- ;negative home WAITX ;wait for home move
HSPD	R/W	Set/get the high speed setting. Unit is	HSPD=1000

		in pulses/second.	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.
		
		Conditions [=, >, <, >=, <=, !=] are available
			IF DI1=0
			  DO=1 ;Turn on DO1
			ELSEIF DI2=0
			  DO=2; Turn on DO2
			ELSE
			  DO=0; Turn off DO
			ENDIF

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+
LHOMEX[+/-]	W	Home the motor using the limit inputs in the specified directions. See section 7.9.4 for details.	LHOMEX+ ;positive home WAITX

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

LTEX	R	Get the latched encoder position.	V1=LTEX ;Set to latch encoder
LTPX	R	Get the latched pulse position.	V1=LTPX ;Set to latch position
LTSX	R	Get the current latch status.	V1=LTSX ;Set to latch status

LTX	W	Enable/disable the high speed latch feature.	LTX=0 ;Disable latch
			LTX=V1 ;Set latch to V1

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
PSX	R	Get the current motor speed.	V2=PSX ;Set V2 to speed

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

SLSX	R	Get the current StepNLoop status.	V3=SLSX ;Set to status

SLX	W	Enable/disable StepNLoop closed loop mode.	SLX=1 ;Enable StepNLoop
			SLX=0 ;Disable StepNLoop

SR[0,1]=[Value]	W	Set the standalone control for the specified program. See Table 7.10.	SR0=0 ;Turn off program 0

SSPDMX	W	Set the SSPD mode. Must be done before move command.	SSPDMX=1 ;Set SSPD mode
			SSPDMX=V2
			JOGX+ ;Jog the motor

SSPDX	W	Perform an on-the-fly speed change. SSPDMX must be set first.	JOGX+ ;Jog the motor
			DELAY=1000 ;Wait 1 second
			SSPDX=1000 ;Change speed
			SSPDX=V1

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	SUB 1 DO=4

		the end of the subroutine.	ENDSUB
SYNCFGX	W	Set the sync output configuration. Refer to Table 7.4.	SYNCFGX=4 ;Interval setting
SYNOFFX	W	Disable the sync output configuration.	SYNOFFX ;Turn off sync
SYNONX	W	Enable the sync output configuration.	SYNONX ;Turn on sync

SYNPOSX	W	Set the sync output reference position.	SYNPOSX=1000 ;Set position
			SYNPOSX=V1 ;Set position

SYNSTATX	R	Get the current sync output status.	V1=SYNSTATX ;Get status
TOC	W	Sets the communication time-out parameter. Units is in milliseconds.	TOC=1000 ;1 second time-out

V[0-99]	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
ZHOMEX[+/-]	W	Home the motor using the home input and Z-index. See section 7.9.2 for details.	ZHOMEX+ ;home in + dir WAITX
ZOMEX[+/-]	W	Home the motor using the Z-index only. See section 7.9.5 for details.	ZOMEX- ;home in - dir WAITX

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 1000
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
        X1000       ;* Move to zero
        WAITX       ;*Wait for X-axis move to complete
        X0          ;* Move to 1000
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
        X1000       ;* Move to zero
        WAITX       ;*Wait for X-axis move to complete
        X0          ;* Move to 1000
        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
                        X1000           ;* Move to zero
                        WAITX           ;*Wait for X-axis move to complete
                        X0              ;* Move to 1000
                ENDIF
        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
        ENDWHILE                ;* Go back to WHILE statement
        END

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

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 position 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
        ELSEIF DI2=1        ;* If digital input 2 is on
                X2000       ;* Move to 2000
        ELSEIF DI3=1        ;* If digital input 3 is on
                X3000       ;* Move to 3000
        ELSEIF DI5=1        ;* If digital input 5 is on
                HOMEX-      ;* Home the motor in negative direction
        ENDIF
        V1=MSTX             ;* Store the motor status to variable 1
        V2=V1&7             ;* Get first 3 bits
        IF V2!=0
                DO1=1
        ELSE
                DO1=0
        ENDIF
ENDWHILE                    ;* Go back to WHILE statement
END
```

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 0
        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
```

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 0
     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
```

A. Speed Settings

HSPD value [PPS] [1]	Speed Window [SSPDM]	Min. LSPD value	Min. ACC [ms]	δ	Max ACC [ms]
< 16 K	0,1	10	2	500	((HSPD – LSPD) / δ) × 1000
< 30 K	2	10	1	1 K	
< 80 K	3	15	1	2 K	
< 160 K	4	25	1	4 K	
< 300 K	5	50	1	8 K	

Table A.0

[1] If StepNLoop is enabled, the [HSPD range] values needs to be transposed from PPS (pulse/sec) to EPS (encoder counts/sec) using the following formula:

EPS = PPS / 0.8

A.1 Acceleration/Deceleration Range

The allowable acceleration/deceleration values depend on the **LSPD** and **HSPD** settings.

The minimum acceleration/deceleration setting for a given high speed and low speed is shown in Table A.0.

The maximum acceleration/deceleration setting for a given high speed and low speed can be calculated using the formula:

Note: The ACC parameter will be automatically adjusted if the value exceeds the allowable range.

$$\boxed{\textbf{Max ACC = ((HSPD – LSPD) / δ) × 1000 [ms]}}$$

Figure A.0

Examples:

a) If **HSPD** = 20,000 pps, **LSPD** = 100 pps:
   a. Min acceleration allowable: **1 ms**
   b. Max acceleration allowable:
      ((20,000 – 100) / 1,000) x 1,000 ms = **19900 ms** (19.9 sec)

b) If **HSPD** = 900,000 pps, **LSPD** = 1000 pps:
   a. Min acceleration allowable: **1 ms**
   b. Max acceleration allowable:
      ((900,000 – 1000) / 39,000) x 1000 ms = **23050** ms (23.05 sec)

A.2. Acceleration/Deceleration Range – Positional Move

When dealing with positional moves, the controller automatically calculates the appropriate acceleration and deceleration based on the following rules.

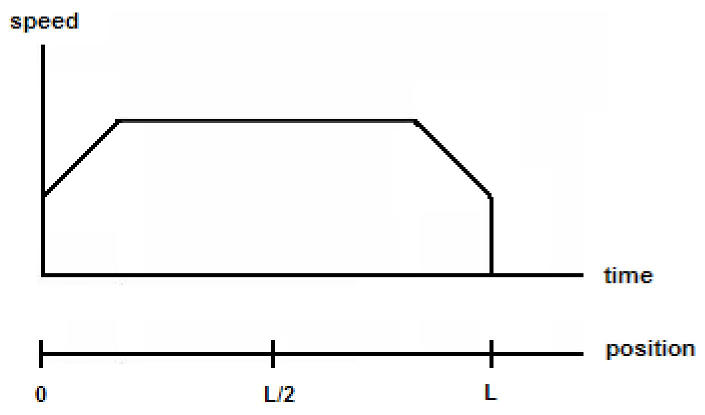


Figure A.1

1) ACC vs. DEC 1: If the theoretical position where the controller begins deceleration is less than L/2, the acceleration value is used for both ramp up and ramp down. This is regardless of the EDEC setting.
2) ACC vs. DEC 2: If the theoretical position where the controller begins constant speed is greater than L/2, the acceleration value is used for both ramp up and ramp down. This is regardless of the EDEC setting.
3) Triangle Profile: If either (1) or (2) occur, the velocity profile becomes triangle. Maximum speed is reached at L/2.

Contact Information

Arcus Technology, Inc.

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

www.arcus-technology.com