DMX-K-SA-11

Integrated NEMA 11 Step Motor + Encoder + Driver + Controller

First edition, February 2008

Revision History:

3.01 – 1st Release
3.05 – 2nd Release
3.06 – 3rd Release
3.08 – 4th Release
3.09 – 5th Release

Firmware Compatibility:

†V331BL

†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

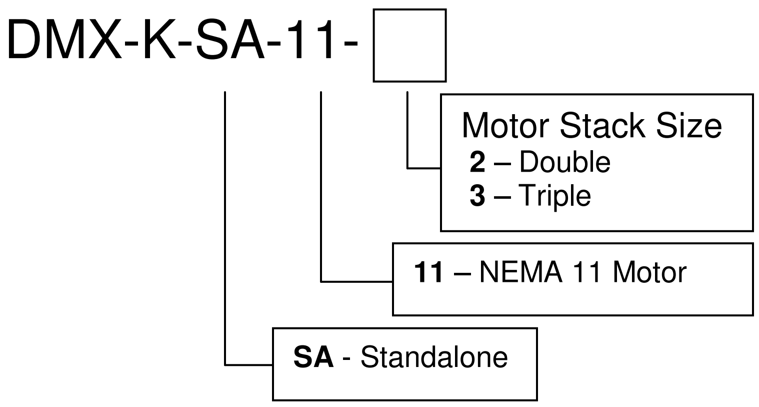
DMX-K-SA-11 is an all-in-one integrated NEMA 11 motor package that combines all the motion components into one convenient package.

Communication to the DMX-K-SA-11 can be established over RS-485. It is also possible to download a standalone program to the device and have it run independent of a host. Sample code is available to aid you in your software development.

1.1. Features

- RS-485 ASCII communication
  - 9600, 19200, 38400, 57600, 115200 bps
- Stepper driver
  - 12-24 VDC
  - 1.5 Amp max current setting (peak current)
  - 1, 2 , 4,or 16 micro-step setting
  - 30 KHz max pulse support
- Stepper motor
  - NEMA 11 motor size available in different stack sizes
  - 1.8° step angle
- Integrated incremental and absolute encoder (256 counts/rev)
- Opto-isolated I/O
  - 1 x output
  - +Limit/-Limit/Home inputs
- Trapezoidal acceleration profile control
- Homing routines:
  - Home input only
  - Limit only

1.2. Part Numbering Scheme



Contacting Support
For any technical support, contact by email at support@arcus-technology.com.
Or your local distributor for technical support.

2. Electrical and Thermal Specifications

Parameter	Min	Max	Units
Main Power Input	+12	+24	V
	-	1.5	A
Opto-supply Power Input	+12	+24	V
Digital Input Forward Diode Current	-	45	mA
Digital Output Collector Voltage	-	+24	V
Digital Output Source Current	-	90	mA
Operating Temperature	0	+85	℃
Storage Temperature	-55	+150	℃

Table 2.0

3. Dimensions

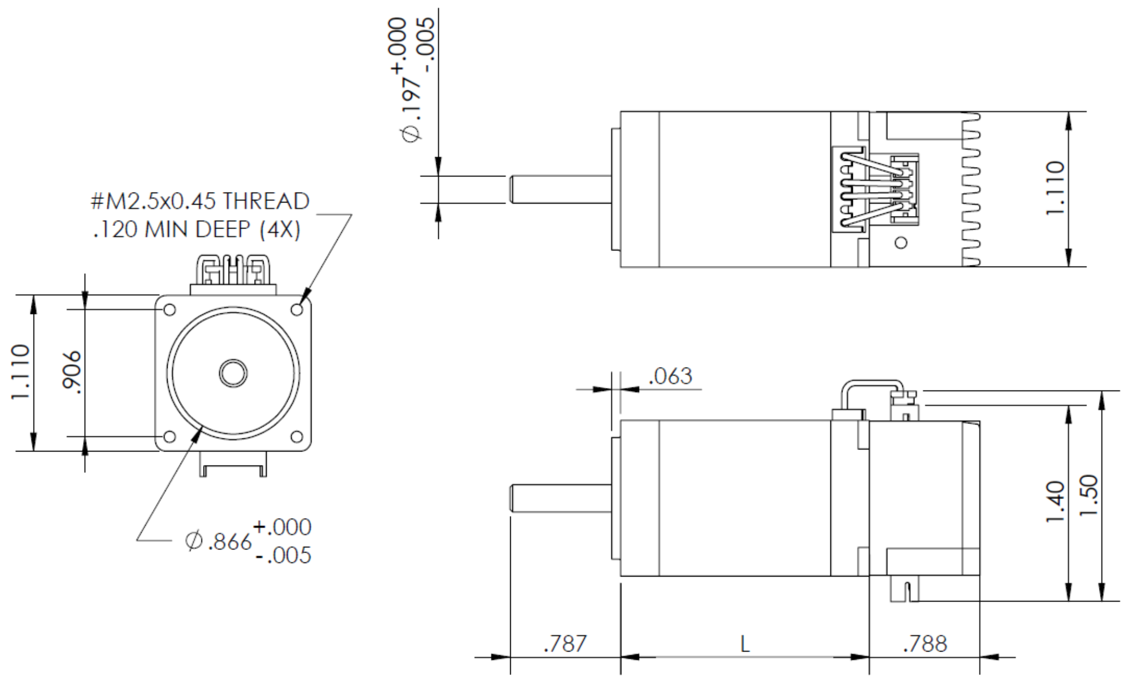


Figure 3.0

NEMA 11 Models	L (inches)
DMX-K-DRV-11-2 (double stack)	1.77
DMX-K-DRV-11-3 (triple stack)	2.01

Table 3.0

4. Connectivity

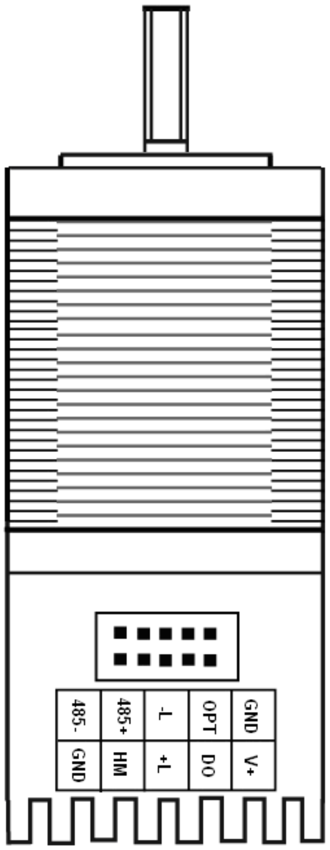


Figure 4.0

4.1. 10-Pin Connector (2mm)

Pin #	Wire Color	In/Out	Name	Description
1	Yellow/Brown	I/O	485-	RS-485 minus signal
2	Black	I	GND	Ground
3	Green/Brown	I/O	485+	RS-485 plus signal
4	White	I	HM	Home input
5	Yellow/White	I	-L	Minus limit input
6	Yellow	I	+L	Plus limit input
7	Orange	I	OPT	Opto-supply input (+12 to +24VDC)
8	Black/Yellow	O	DO	Digital output
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 (10 pin female connector)
	DF11-2428SC (female socket pin)

4.2. 4-Pin Connector (2mm)

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

Table 4.1

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

4.3. DMX-K-SA-11 Interface Circuit

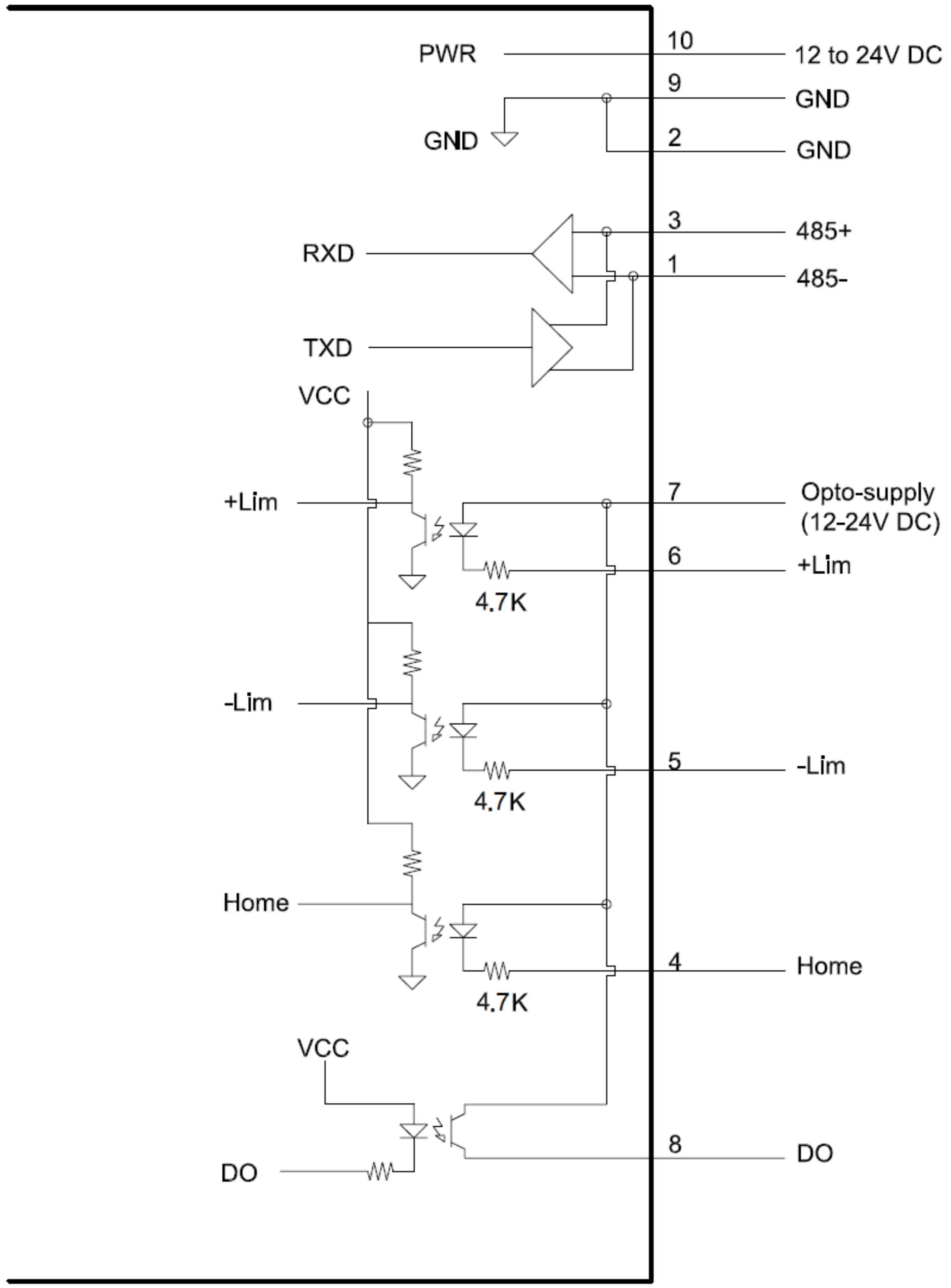


Figure 4.1

4.4. Digital Outputs

Figure 4.2 shows an example wiring to the digital output. All opto-isolated digital outputs will be PNP type.

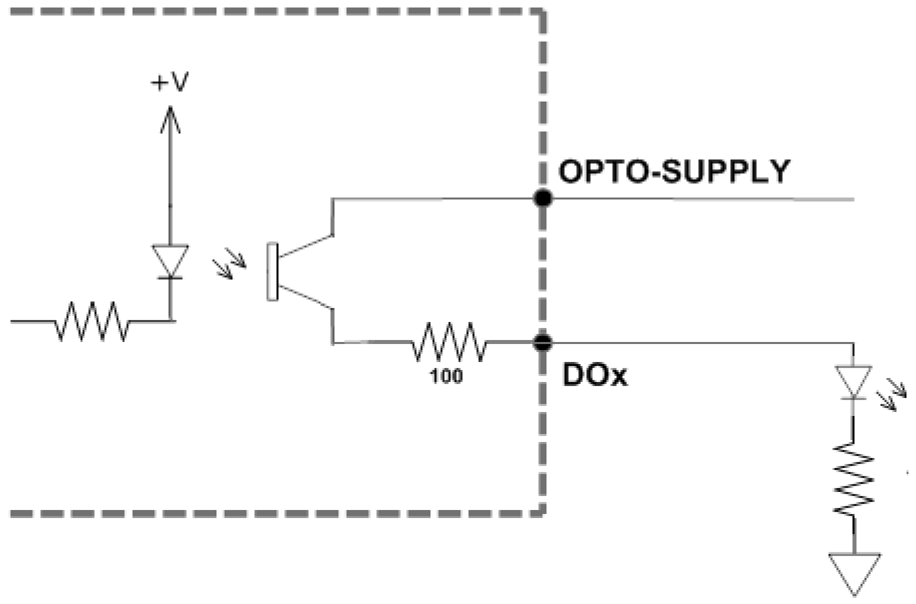


Figure 4.2

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 source current for digital outputs is 90mA. Take caution to select the appropriate external resistor so that the current does not exceed 90mA.

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

4.5. Limits and Home

Figure 4.3 shows the detailed schematic of the opto-isolated limit and home digital inputs. All opto-isolated digital inputs are NPN type.

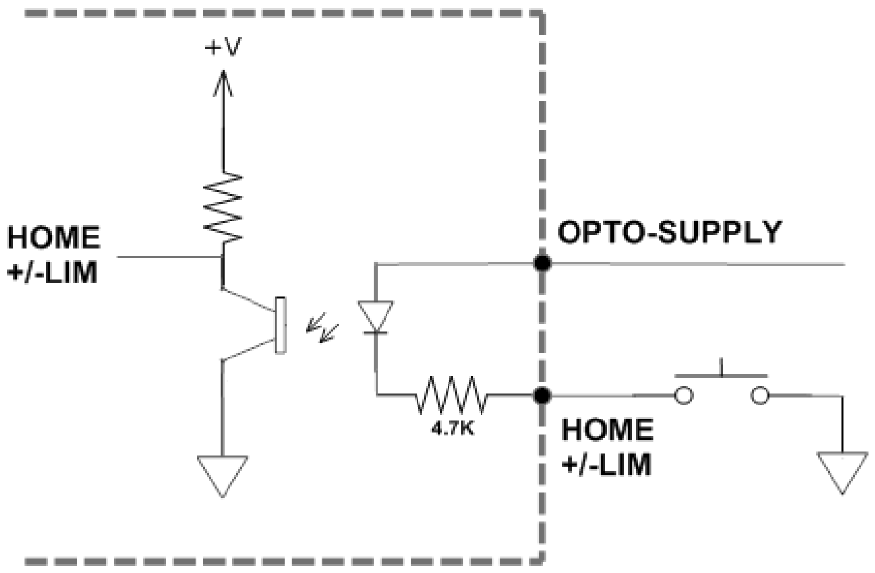


Figure 4.3

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.

5. Stepper Motor Driver Overview

5.1. Microstep

The DMX-K-SA-11 has a configurable microstepping 1, 2, 4, and 16. With the standard 1.8° stepper motor included with the DMX-K-SA-11, these microstep settings translate to 200, 400, 800, and 3200 steps per revolution, respectively.

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

5.2. Driver Current

The DMX-K-SA-11 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-11-2	1.8 A	1.5 A
DMX-K-SA-11-3	1.8 A	1.5 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-11 has configurable current setting from 100mA to 1.5A. 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 7.12 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-K-SA-11 have maximum ambient operating temperature of **85 degree Celsius**. DMX-K-SA-11 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-11 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-11 mounting orientation should be such that the fins are oriented vertically for better convection and heat dissipation.

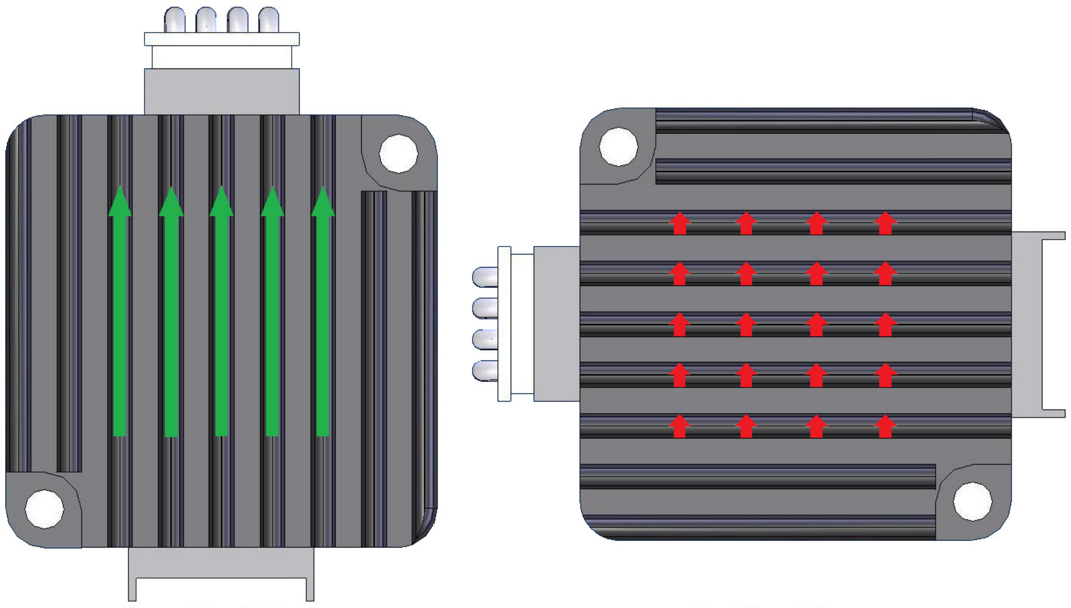


Figure 5.0

5.4. Stepper Motor Specifications

Table 5.1 details the standard stepper motor characteristics of the DMX-K-SA-11.

NEMA Size	Stack	Max Current	Resistance/Phase	Inductance/Phase	Inertia
11	Double	1.8A	1.3 Ohm	0.8 mH	0.07 oz-in^2
	Triple	1.8A	1.9 Ohm	1.7 mH	0.10 oz-in^2

Table 5.1

Additionally, Table 5.2 details to max allowable forces the DMX-K-SA-11 can tolerate on the motor shaft.

NEMA Size	Stack Size	Max Axial Force	Max Radial Force
11	Double	5N	8N
	Triple	5N	8N

Table 5.2

5.5. Stepper Motor Torque

The torque output of the DMX-K-SA-11 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-11 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-11.

See figure 5.1 for the torque curves of DMX-K-SA-11-X.

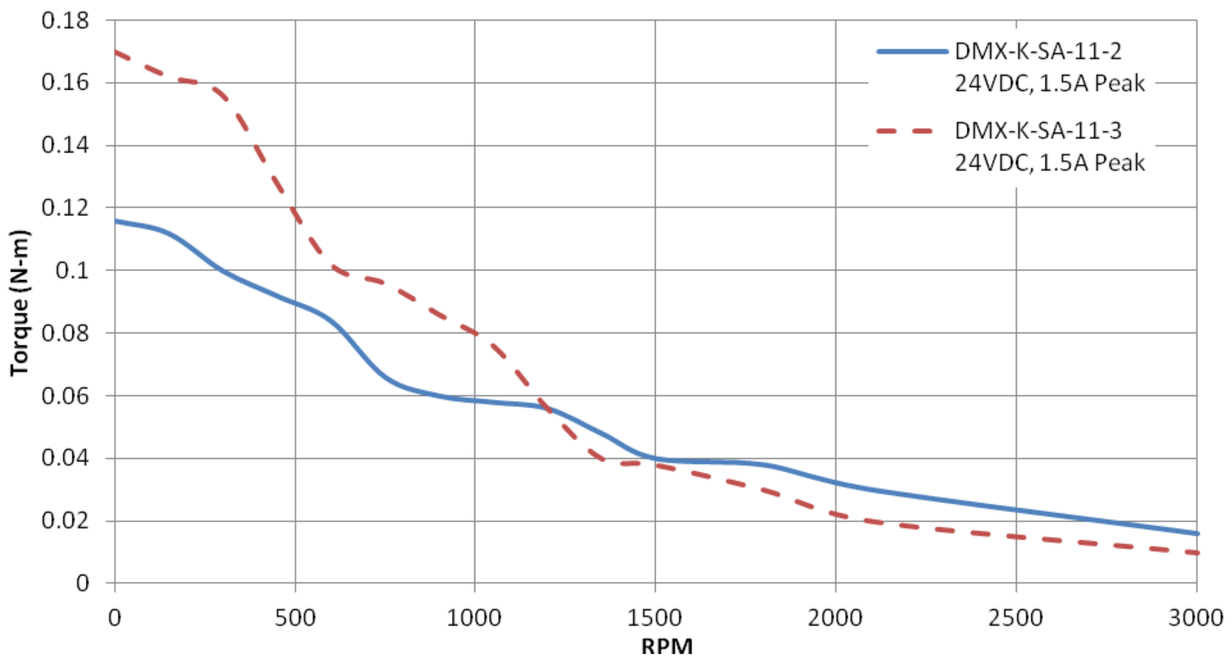


Figure 5.1

6. Communication Interface

6.1. Serial Communication

The DMX-K-SA-11 communicates over an RS-485 interface using an ASCII protocol. An RS-485 serial port on the PC or PLC can be used to communicate with the DMX-K-SA-11. A USB to RS-485 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.

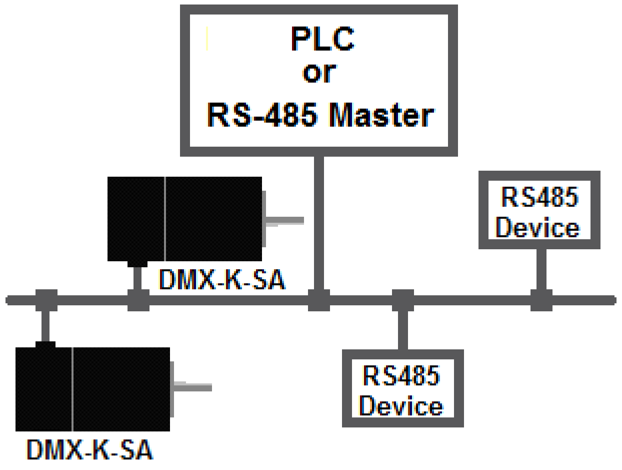


Figure 6.0

6.1.2. Communication Port Settings

The DMX-K-SA-11 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-11 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-11 has a baud rate setting of 9600 bps.

6.1.3. ASCII Protocol

The following ASCII protocol should be used for sending commands and receiving replies from the DMX-K-SA-11. Details on valid ASCII commands can be found in section 9. The following protocol will be used for RS-485 communication.

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: @01POL[CR]
Reply: 7[CR]
```

For reading the digital input status
```
Send: @01DI[CR]
Reply: 1[CR]
```

For performing a store to flash memory
```
Send: @01STORE[CR]
Reply: OK[CR]
```

6.1.4. Response Type Selection

It is also possible to choose different format for the response string from the DMX-K-SA-11. 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.

6.1.5. 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.

6.1.6. 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. Device Number

If multiple DMX-K-SA-11 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-K-SA-11, first store the desired number using the DN command. Note that this value must be within the range [DMK01, DMK99].

For example, to change the device number, the command **DN=DMK02** can be sent. The device name can also be changed through the Setup window of the standard DMX-K-SA-11 software. See section 8 for details.

By default, all DMX-K-SA-11 start with device number DMK01.

To save a modified device number to the DMX-K-SA-11'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.

6.3. Windows GUI

DMX-K-SA-11 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. Motion Control 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-11.

7.1. Motion Profile

DMX-K-SA-11 has trapezoidal velocity profile as shown figure 7.0.

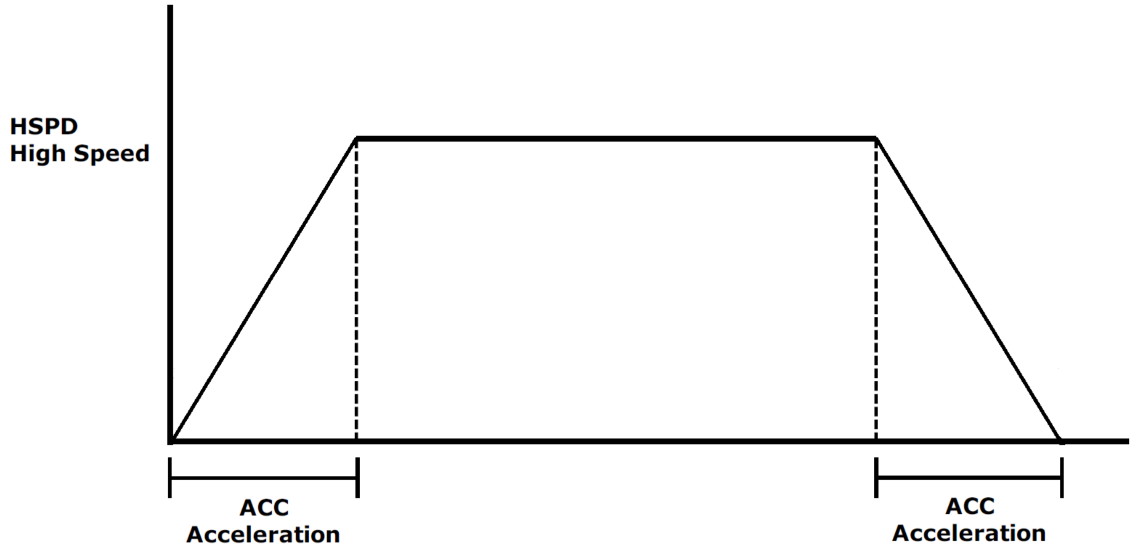


Figure 7.0

Once a typical move is issued, the motor will immediately start moving from idle speed 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 a complete stop.

The high speed setting is in pps (pulses/second). Use the **HSPD** command to set/get the high speed setting. Depending on the voltage, current, motor type, microstep, and acceleration value, the maximum achievable speed will vary. Acceleration time is in milliseconds. Use the **ACC** command to access the acceleration setting.

Standard DMX-K-SA-11 comes with 1.8 degree motor with 1, 2, 4, and 16 configurable microstep setting. To convert rotational speed to pulse speed, use the following formula.

Pulse/Sec = RPS * 200 Pulse/Rev * microstep

The high speed setting will be restricted by the acceleration setting. Similarly, the acceleration will be restricted by the high speed setting. The **HSPD** and **ACC** are limited by the following equation:

HSPD x ACC < 6,000,000

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

7.2. Position Counter

DMX-K-SA-11 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-11 also has integrate encoder that can provide both the incremental encoder position or the absolute encoder position. The built in encoder will have a resolution of 256 counts/revolution for both incremental and absolute encoder readings.

The incremental encoder position uses a 32-bit signed counter. Use the **EX** command to read and set the incremental encoder position. The incremental will be reset to 0 after each power cycle.

The absolute encoder position uses an read-only 8-bit unsigned counter. Use the **EA** command to read the absolute encoder position. The absolute encoder will constantly preserve the current position of the motor, even during power down.

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

7.3. 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-11 while it is disabled, the move command will still process however the motor will not be able to physically move.

ASCII	**EO**
Standalone	**EO**

7.4. 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[+/-]**

7.5. 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**

7.6. 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 target position will be in units of motor steps.

The DMX-K-SA-11 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**	**-**

7.7. Homing

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

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.

7.7.1. Home Input Only (High Speed Only)

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

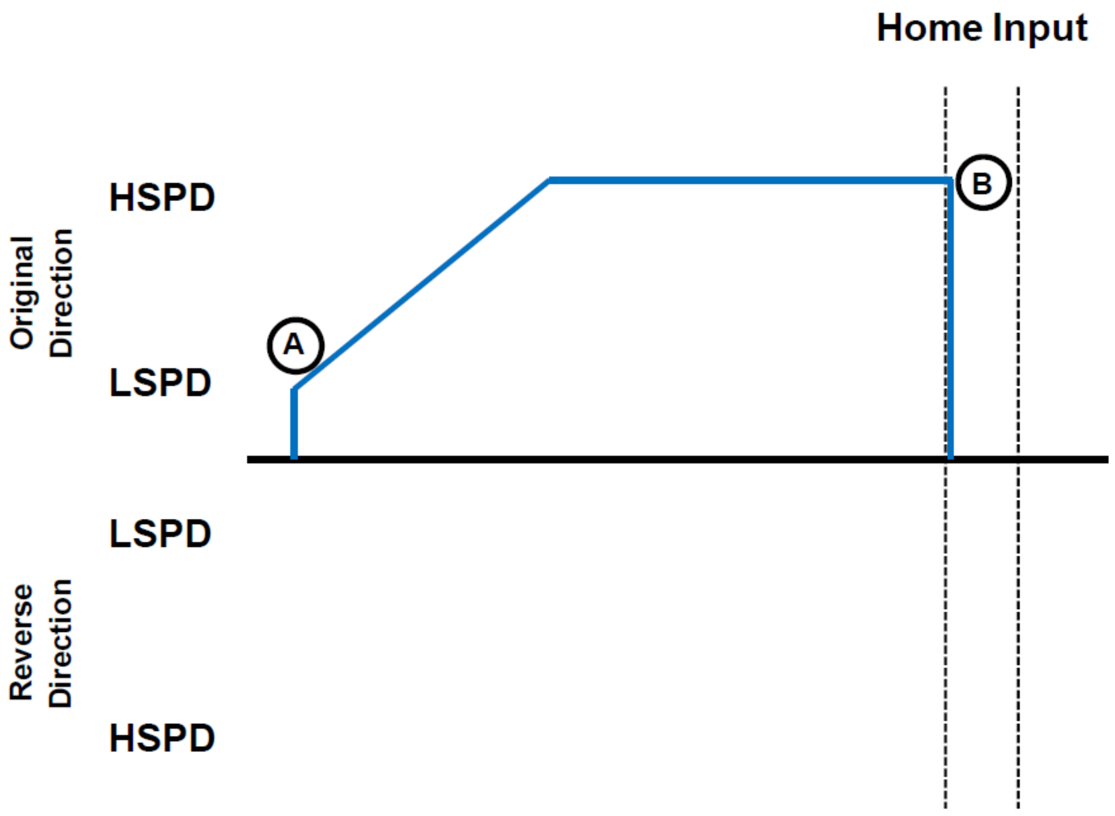


Figure 7.1

A. Starts the motor from low speed and accelerates to high speed.
B. As soon as the home input is triggered, the position counter is reset to zero and the motor stops immediately.

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

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

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

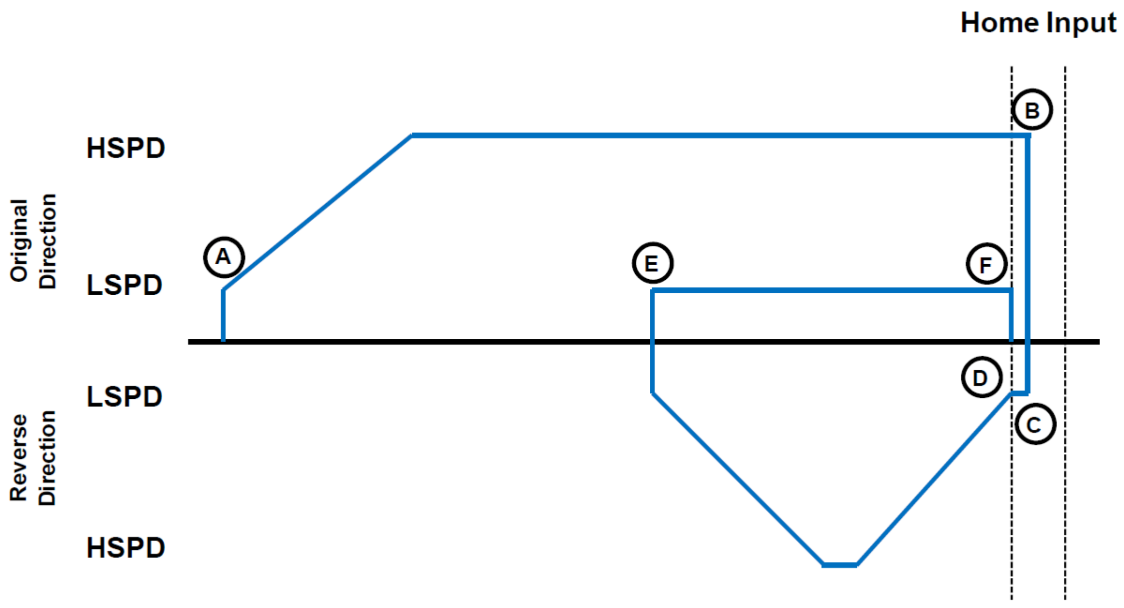


Figure 7.2

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 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+/-**	**-**

7.7.3. Limit Only

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

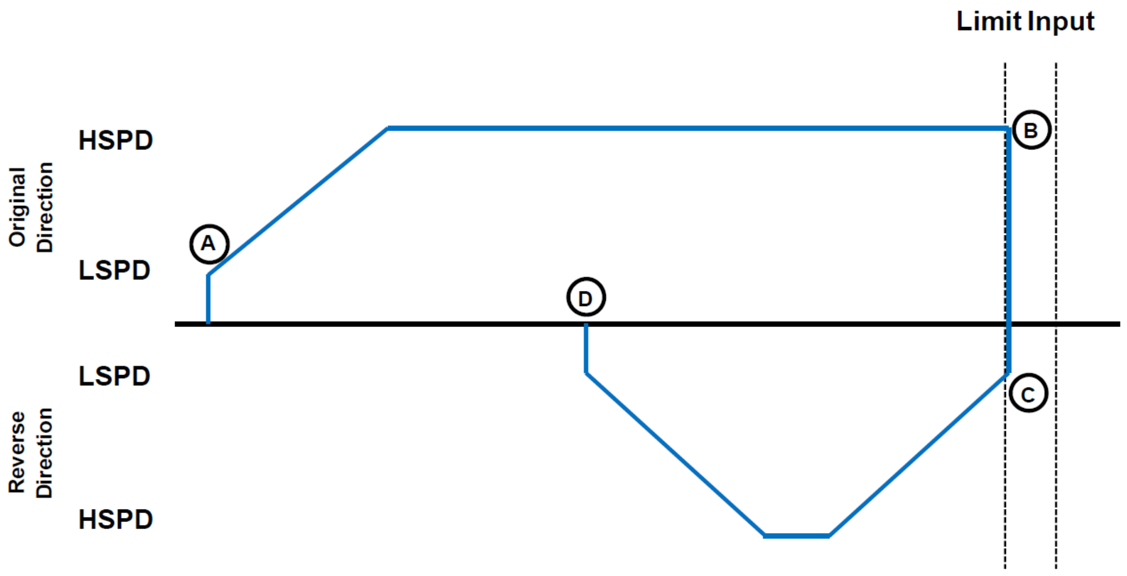


Figure 7.3

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.

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

7.8. 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**	**-**

7.9. Motor Status

Motor status can be read anytime using the **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	Minus limit input switch status
4	Plus limit input switch status
5	Home input switch status
6	Minus limit error. This bit is latched when minus limit is hit during negative direction motion.
7	Plus limit error. This bit is latched when plus limit is hit during positive direction motion.
10	Communication timeout counter alarm

Table 7.0

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

7.10. Digital Inputs/Outputs

DMX-K-SA-11 comes with 3 digital inputs and 1 digital output.

7.10.1. Inputs

The digital input status of all 3 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[1-3]** 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	Minus Limit	DI1
1	Plus Limit	DI2

2	Home	DI3

Table 7.1

ASCII	**DI**	**DI[1-3]**
Standalone	**DI**	**DI[1-3]**

7.10.2. 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 0-1.

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
0	Digital Output 1

Table 7.2

The voltage level of the digital output when it is on or off is determined by the polarity setting. See section 7.11 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**	**DOBOOT**
Standalone	**DO**	**-**

7.11. Polarity

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

Bit	Description
0	Dir (CW or CCW)
1	Limit
2	Home
3	Digital Output
4	Jump to line 0 on error†

Table 7.3

The jump to line 0 polarity option defined by bit 4 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.

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

7.12. 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 1500mA. 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	**-**	**-**	**-**	**-**

7.13. Communication Time-out Feature (Watchdog)

DMX-K-SA-11 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**

7.14. 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-11. 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-11 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-11. Each line of written standalone code creates 1-4 assembly lines of code after compilation

The DMX-K-SA-11 has the ability to store and operate two separate standalone programs simultaneously.

7.14.1. Standalone Program Specification

Memory size:510 assembly lines ~ 3.0 KB.
Note: Each line of pre-compiled code equates to 1-4 lines of assembly lines.

7.14.2. Standalone Control

The DMX-K-SA-11 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 [0-509].

7.14.4. Standalone Subroutines

The DMX-K-SA-11 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.

7.14.5. 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 contains examples of using subroutine 31 to perform error handling.

The return jump from subroutine 31 will be determined by bit 4 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-K-SA-11 has 32 32-bit signed standalone variables available for general purpose use. They can be used to perform basic calculations and support integer operations. The **V[0-31]** 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 V16 through V31 can be stored to flash memory using the **STORE** command. Variables V0-V15 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 to 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.

ASCII Command	Description
CURI,CUR	Driver current settings
DB	Serial communication baud rate
DN	Device name
DOBOOT	DO configuration at boot-up
HCA	Home correction amount
DL	Disable limit switch function
LCA	Limit correction amount
POL	Polarity settings
RT	ASCII response type
SLOAD	Standalone program run on boot-up parameter
TOC	Time-out counter reset value
V16-V31	Note that on boot-up, V0-V15 are reset to value 0

Table 7.8

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

7.16. Default Settings

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

Command	Parameter Description	Value
CURI	Idle Current	500 mA
CURR	Run Current	500 mA
MS	Microstep	16
DB	Baud Rate	9600
DN	Device ID	DMK01
DOBOOT	Digital Output Boot-up State	1
HCA	Home Correction Amount	1000
DL	Disable Limit	Disabled
LCA	Limit Correction Amount	1000
POL (bit 0)	Direction Polarity	CW
POL (bit 1)	Limit Polarity	Active Low
POL (bit 2)	Home Polarity	Active Low
POL (bit 3)	Digital Output Polarity	Active Low
POL (bit 4)	Jump to line 0 on error	Active Low
RT	Response type	Do Not Append
SLOAD	Run Program(s) on Power Up	0
TOC	Time-out counter value (Watch-dog)	0
V16-V31	Non-volatile variables	0

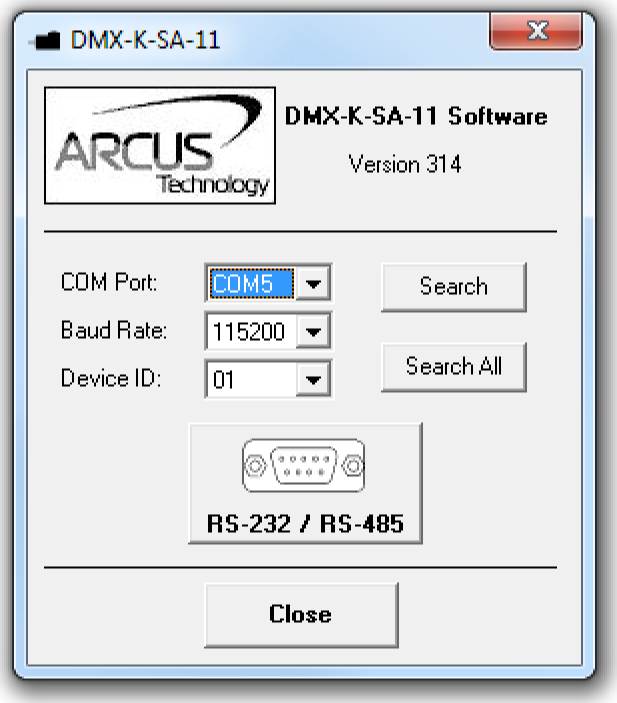
Table 7.9

8. Software Overview

The DMX-K-SA-11 has a Windows compatible software that uses RS485 communication. Standalone programming, along with all other available features of the DMX-K-SA-11, will be accessible through the software. It can be downloaded from the Arcus Technology website.

To communicate over RS485, make sure that the DMX-K-SA-11 is connected to the functional COM port on the PC.

Startup the DMX-K-SA-11 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 RS485 bus.



COM Port: COM5
Baud Rate: 115200
Device ID: 01

Figure 8.0

The first DMX-K-SA-11 connected to the PC can be found using the Search button. If there all multiple DMX-K-SA-11 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:

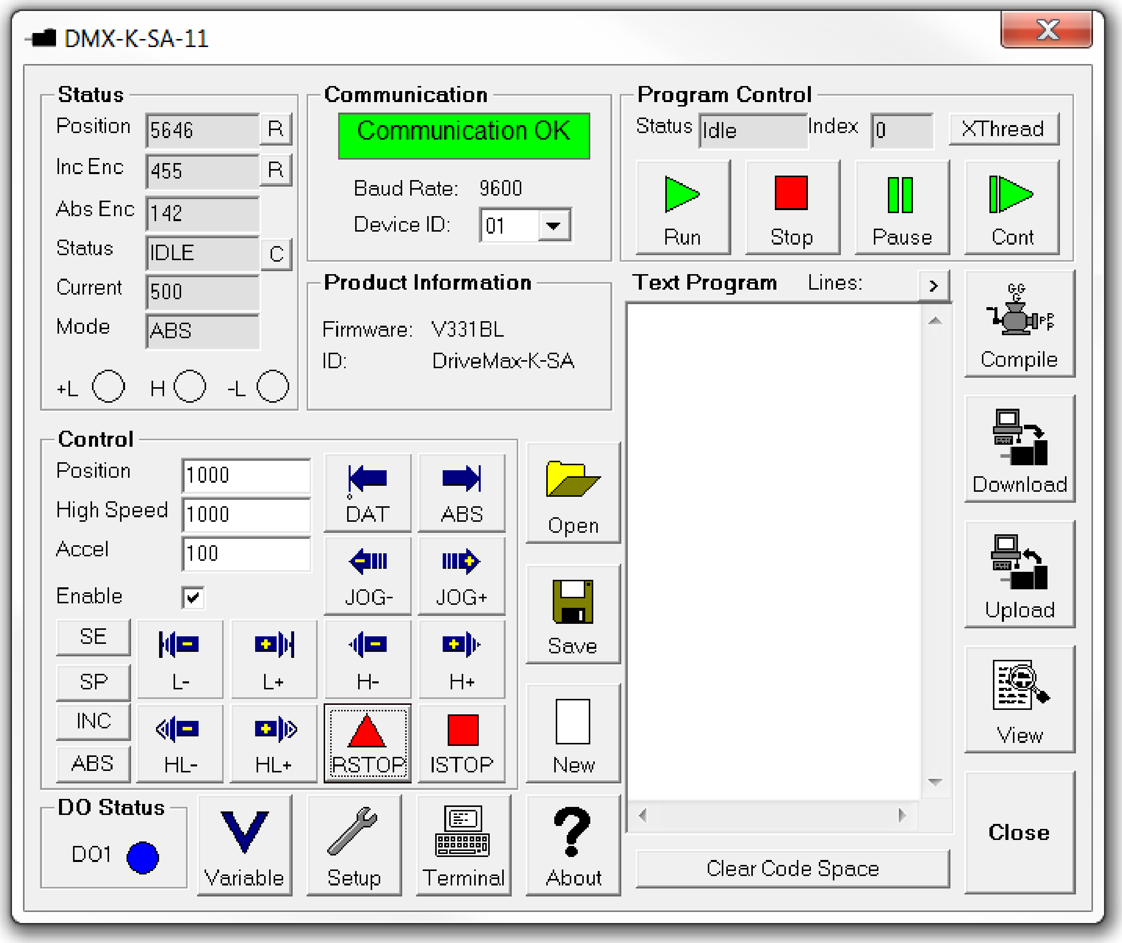
- Check power supply to DMX-K-SA-11. The allowable power range is from 12VDC to 24VDC.
- Check communication wiring. If using RS-232, TXD from DMX-K-SA should be connected to RXD of the serial port and RXD from DMX-K-SA should be connected to TXD of serial port. If using RS-485, Make sure that the 485+ from DMX-K-SA-11 is connected to 485+ of the

master and 485- from DMX-K-SA-11 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-485 button can be used to open the Main Control Screen.

8.1. Main Control Screen


DMX-K-SA-11

Status
Position 5646
Inc Enc 455
Abs Enc 142
Status IDLE
Current 500
Mode ABS

Communication
Communication OK
Baud Rate: 9600
Device ID: 01

Product Information
Firmware: V331BL
ID: DriveMax-K-SA

Program Control
Status Idle
Index 0

Control
Position 1000
High Speed 1000
Accel 100
Enable ✓

DO Status
DO1 ●

Figure 8.1

8.1.1. Status

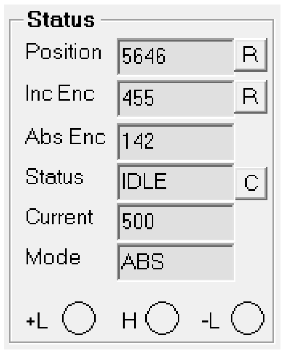


Figure 8.2

1. **Position** – displays the current pulse position. The pulse position can be reset using the adjacent R button
2. **Inc Enc** – displays the current incremental encoder position. The encoder position can be reset using the adjacent R button.
3. **Abs Enc** - displays the current absolute encoder position. The absolute encoder position will read from 0-255.
4. **Status** – displays the current motor status by display one of the following. The C button can be used to clear an error.
    - IDLE – motor is not moving.
    - ACCEL – motor is accelerating
    - CONST – motor is running in constant speed
    - DECEL – motor is decelerating
    - ERROR – limit error
5. **Current** – displays active current value. Value is in mA.
6. **Mode** – displays the current move mode that is used during positional moves. The following modes are available.
    - ABS – Target movement moves to the absolute target position
    - INC – Target movement increments/decrements by the target position amount
7. **H/+L/-L** – displays the –Limit, +Limit, and Home input status

8.1.2. Control

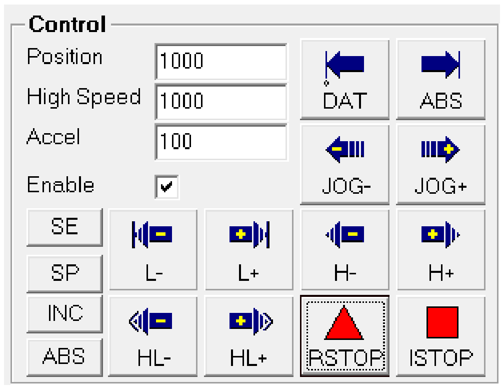


Figure 8.3

1. **Position** – enter the desired target position entered here. This value will be used when performing a position move.
2. **High Speed**/**Accel** – set the high speed and acceleration parameters
3. **Enable** – motor power is turned on or off using this check box. When motor is disabled, no motion is done.
4. **SE**/**SP** – Set the pulse position or encoder position. Positions are set using the position box value.
5. **INC**/**ABS** – select absolute or incremental move mode
6. **HL+**/**HL-** – homing is done using only the home sensor at high speed and low speed.
7. **L+**/**L-** – homing is done using the Limit inputs.
8. **RSTOP**/**ISTOP** – motion is either stopped with deceleration using RSTOP or without deceleration using ISTOP.
9. **H+**/**H-** – homing is done using only the home sensor at high speed.
10. **JOG+**/**JOG-** – jogs the motor in positive and negative direction.
11. **ABS** – moves the motor to the target absolute position using the high speed and the low speed and the acceleration values.
12. **DAT** – moves the motor to the zero target position.

8.1.3. Digital Output

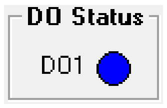
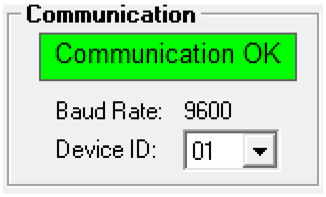


Figure 8.4

To toggle the digital output available on the DMX-K-SA-11, click on the circle.

8.1.4. Communication



Communication

Communication OK

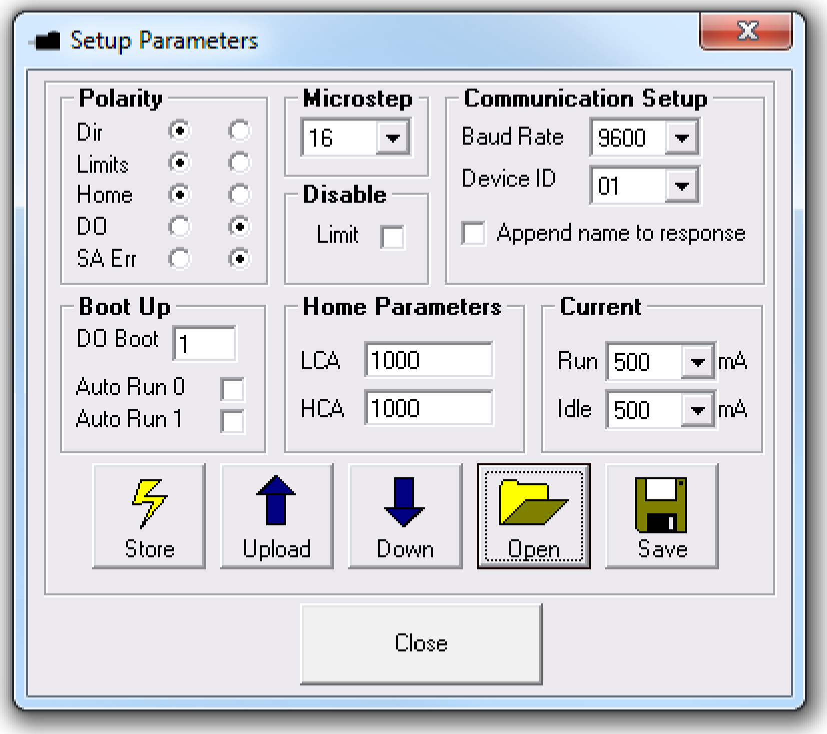
Baud Rate: 9600

Device ID: 01

Figure 8.5

1. Communication Status – Displays communication status with the selected device.
2. Device ID – Device ID of the communicating DMX-K-SA-11. To communicate with a different DMX-K-SA-11 on-the-fly, select another ID number from this drop-down box.

8.1.5. Setup



Setup Parameters

Polarity

Dir	●	○
Limits	●	○
Home	●	○
DO	○	●
SA Err	○	●

Microstep: 16

Disable: Limit ☐

Communication Setup

Baud Rate: 9600

Device ID: 01

☐ Append name to response

Boot Up

DO Boot: 1

Auto Run 0 ☐

Auto Run 1 ☐

Home Parameters

LCA: 1000

HCA: 1000

Current

Run: 500 mA

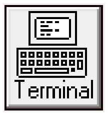
Idle: 500 mA

Figure 8.6

1. **Boot Up Setup -** the following boot up parameters can be configured
   - DO Boot: digital output configuration
   - Auto Run: automatic run on boot up for the specified standalone program

2. **Polarity Setup -** the following polarity parameters can be configured
    - Dir: direction of the motion (clockwise or counter-clockwise)
    - Limits: limit input polarity
    - Home: home input polarity
    - DO: digital output polarity
    - SA Err: standalone error jump line
        - Low: jump to previous line
        - High: jump to line 0
3. **Disable Limit** - disable the limit input so they can be used as general purpose inputs
4. **Microstep** - set the driver to 1, 2, 4, or 16 microstep.
5. **Home Parameters**
    - LCA: limit correction amount
    - HCA: home correction amount
6. **Communication Setup**
    - Baud Rate – set the baud rate ranging from 9600 bps, 19200 bps, 38400 bps, 57600 bps, 115200 bps.
    - Device ID – Device name ranges from DMK01 to DMK99.
    - Set response type. If checked, the device name will be appended to the response of the device.
7. **Current**– Run current is used when the motor is running. Idle current is used when the motor is not moving. Minimum current setting is 100 mA and maximum current setting is 1500 mA. Depending on the model of the motor, the current setting should not go above the recommended maximum rated current of the motor.
8. **Open**/**Save** – Configuration values can be saved to a file and read from a file.
9. **Upload**/**Download** – Configuration values can be uploaded and downloaded. Note that if the configuration values are changed, it needs to be downloaded to take effect.
10. **Store** – The downloaded parameters can be permanently stored on the non-volatile memory.

8.1.6. Terminal



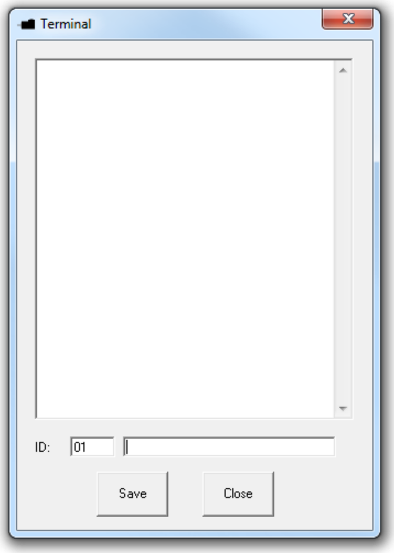


Figure 8.7

1. **Response Window** – this large text box displays the response history, if any, from the command line
2. **ID** – Select the address of the DMX-K-SA module which you wish to communicate. Selecting address **‘00’** will send a broadcast command which will be received by all DMX-K-SA modules on a RS-485 bus.
3. **Command Line** – Type the commands here to send them to the DMX-K-SA-11. When sending commands, you do not need to type the device name. For example, when you want to know what the motor status is, type MST and you will see a number on the reply that represents the status of the motor. Press the Enter key to send the command.
4. **Save** – Save the text in the response window into a text file. When this button is pressed, typical Windows file save dialog box will open.

8.1.7. Standalone Program File Management

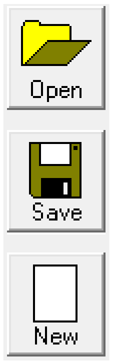


Figure 8.8

1. **Open** - standalone program is loaded to the editor box. When this button is pressed, typical Windows file open dialog box will open.
2. **Save** – standalone program in the text editor is saved to a file. When this button is pressed, typical Windows file save dialog box will open.
3. **New** – when this button is pressed, the text editor is cleared.

8.1.8. Text Programming Box

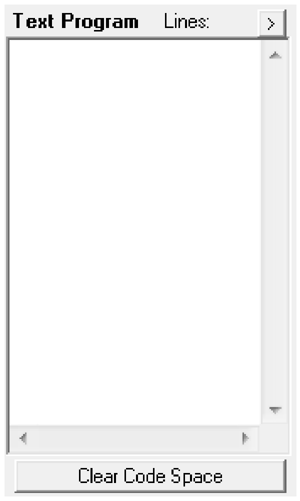


Figure 8.10

Compose standalone code in this text box. Compile and Download to store the code to the DMX-K-SA-11. The Clear Code Space button allows the user to erase the entire standalone code space.

8.1.9. Standalone Program Process

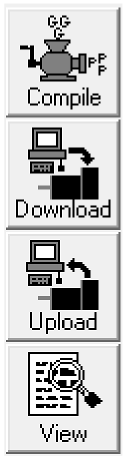


Figure 8.11

1. **Compile** – Compile the standalone program.
2. **Download** – Download the compiled program to the DMX-K-SA.
3. **Upload** – Upload the standalone program from the DMX-K-SA.
4. **View** – View the low level compiled program.

8.1.10. Program Control

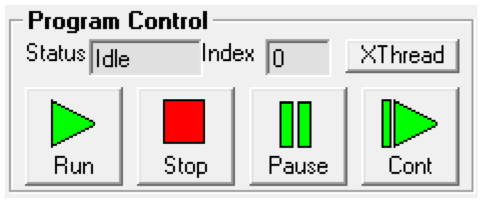
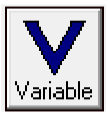
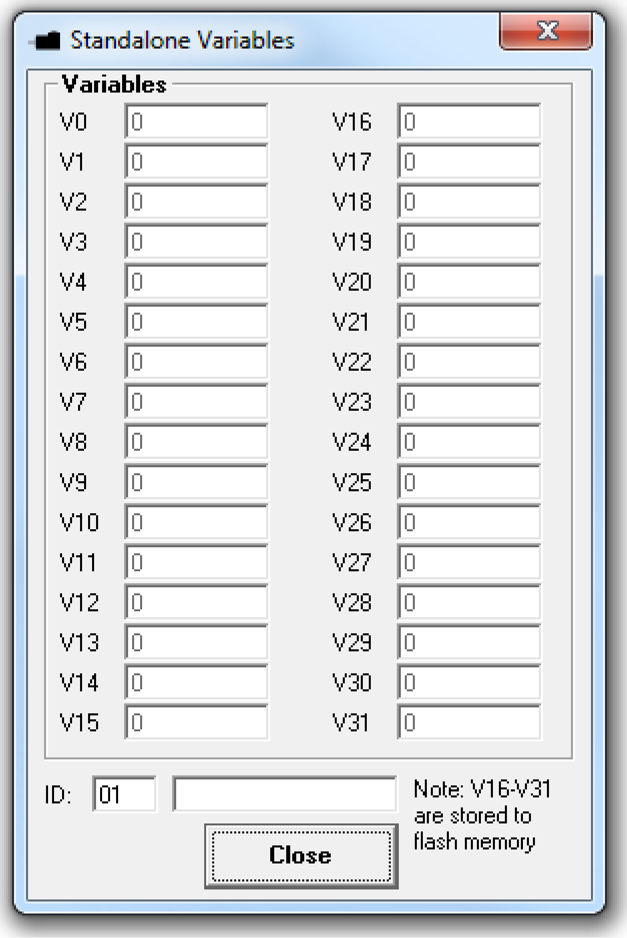


Figure 8.12

1. **Status** – displays the current program status. Following are possible program status: Idle, Running, Paused, and Error.
2. **Index** – displays the line of low level standalone code that is current being processed. Note that this does not directly correspond to the line of code written by the user.
3. **Run** – run the standalone program.
4. **Stop** – stop the standalone program.
5. **Pause** – pause the standalone program.
6. **Continue** – continue a paused standalone program.
7. **X-Thread –** Open the Program Control for standalone multi-thread operation.

8.1.11. Variables





Variables

V0	0	V16	0
V1	0	V17	0
V2	0	V18	0
V3	0	V19	0
V4	0	V20	0
V5	0	V21	0
V6	0	V22	0
V7	0	V23	0
V8	0	V24	0
V9	0	V25	0
V10	0	V26	0
V11	0	V27	0
V12	0	V28	0
V13	0	V29	0
V14	0	V30	0
V15	0	V31	0

ID: 01

Note: V16-V31 are stored to flash memory

Figure 8.6

View the status of variables 0-31. ASCII commands can be sent to the DMX-K-SA-11 through the available text box..

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-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. For decelerate stop, use STOP command.	OK
ABS	Set move mode to absolute	OK
ACC	Return the current acceleration value in milliseconds.	[0-6,000,000]
ACC=[Value]	Sets acceleration value in milliseconds.	OK
CLR	Clears limit error	OK
CUR	Return the actual current setting value. Units are in mA.	[100-1500]
CURI	Return idle current setting. Units are in mA.	[100-1500]
CURI=[100-1500]	Set idle current.	OK
CURR	Return run current setting	[100-1500]
CURR=[100-1500]	Set run current	OK
DB	Return baud rate setting	See Table 6.1
DB=[0-5]	Set baud rate. See Table 6.1	OK
DI	Return status of digital inputs. See table 7.1.	[0-7]
DI[1-3]	Return status of individual input	[0,1]
DL	Return disable limit status	[0,1]
DL=[0,1]	Set disable limit status	OK
DN	Return device name	DMK01-DMK99
DN=[Value]	Set device name.	OK
DO	Return digital output status	[0,1]
DO=[0,1]	Set digital output status	OK
EA	Return the absolute encoder position	[0-255]
EO	Return the driver power enable status.	[0,1]
EO=[0 or 1]	Enables or disables motor power.	OK
EX	Return the incremental encoder count value	32-bit signed number
EX=[Value]	Set the incremental encoder count value	OK
HSPD	Returns high speed setting	[0-30,000]
HSPD=[Value]	Sets 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 in positive direction (with low	OK

	speed)	
HL-	Homes the motor in negative direction (with low speed)	OK
ID	Returns product ID	DriveMax-K-SA
INC	Set move mode to incremental	OK
J+	Jogs the motor in positive direction	OK
J-	Jogs the motor in negative direction	OK
L+	Limit home in the positive direction	OK
L-	Limit home in the negative direction	OK
LCA	Returns the limit correction amount	28-bit number
LCA=[Value]	Sets the limit correction amount.	OK
MM	Return move mode status	[0,1]
MS	Returns microstep setting	[1,2,4,16]
MS=[microstep]	Sets microstep setting. Available microstep values are 1, 2, 4, and 16.	OK
MST	Returns motor status	See Table 7.0
POL	Returns current polarity	See Table 7.3
POL=[Value]	Sets polarity. See Table 7.3.	OK
PX	Returns current position value	32-bit number
PX=[value]	Sets the current position value	OK
RT	Return response type parameter	[0,1]
RT=[0,1]	Set response type parameter	OK
SASTAT[0,1]	Return the standalone program status 0 – Idle 1 – Running 2 – Paused 4 – In Error	[0-4]
SLOAD	Returns RunOnBoot parameter	See Table 7.7
SLOAD=[value]	Set RunOnBoot parameter. See table 7.7.	OK
SPC[0,1]	Return program counter for standalone program	[0-509]
SR[0,1]=[Value]	Control standalone program: 0 – Stop standalone program 1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program	OK
STORE	Store setup values to flash memory.	OK
STOP	Decelerated to stop the motor if in motion. For immediate stop, use ABORT command	OK
TOC	Return time-out counter. Units are in milliseconds.	32-bit number
TOC=[value]	Set time-out counter. Units are in milliseconds.	OK
V[0-31]	Return the standalone variable value	32-bit number
V[0-31]=[Value]	Set the standalone variable value	OK
VER	Returns current firmware software version number	VXXX
X[value]	Moves the motor to absolute position value using the HSPD and ACC values.	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
?Index out of Range	The index for the command sent to the controller is not valid
?Moving	A move or position change command is sent while the controller is outputting pulses
?Current Error Range	An invalid current value was used when trying to set the run or idle current

Table 9.1

10. Standalone Language 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[1-3]	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=1 ;Turn on DO1
EA	R	Get the current absolute encoder position	V1=EA ;Read abs encoder
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 incremental encoder position.	EX=1000 ;Set to 1000
			V1=EX ;Read inc 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 7.7.2 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 7.7.1 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.
		
		Conditions [=, >, <, >=, <=, !=] are
			IF DI1=0
			  DO=1 ;Turn on DO1
			ELSEIF DI2=0
			  DO=2; Turn on DO2
			ELSE
			  DO=0; Turn off DO
			ENDIF

		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+
LHOMEX[+/-]	W	Home the motor using the limit inputs in the specified directions. See section 7.7.3 for details.	LHOMEX+ ;positive home WAITX

LSPD	R/W	Set/get the low speed setting. Unit it 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[0-31]	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.	WHILE 1=1 ;Forever loop

		ENDWHILE should be used to close off a WHILE loop.
		
		Conditions [=, >, <, >=, <=, !=] are available.
			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 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
```

Contact Information

Arcus Technology, Inc.

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

www.arcus-technology.com