DMX-K-SA

Integrated Step Motor Encoder/Driver/Controller Manual

First edition, April 2007

Revision History:

2.00 – 1st Release
3.15 – 2nd Release
3.16 – 3rd Release
3.17 – 4th Release
3.18 – 5th Release

Firmware Compatibility:

†V340BL

†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.

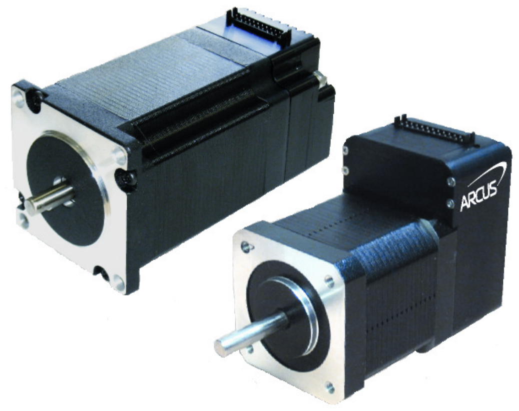
Sample source code is available to aid you in your software development.

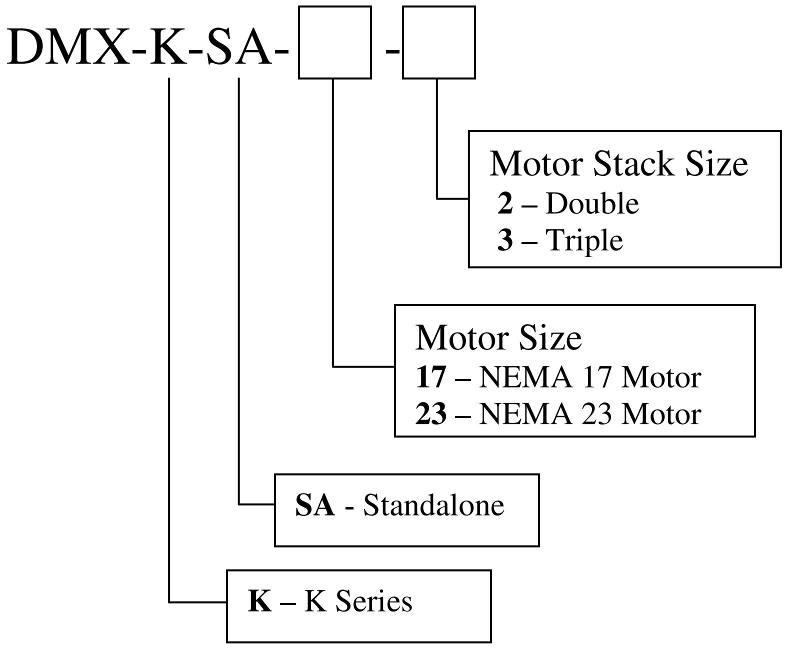
Features

DMX-K-SA-17/23

- RS-485 + RS-232 ASCII communication
    - 9600, 19200, 38400, 57600, 115200 bps
- A/B/Z differential encoder inputs
    - StepNLoop closed loop control (position verification)
- Opto-isolated I/O
    - 6 x inputs (1 x high speed position capture latch input)
    - 3 x outputs (1 x position synchronized output)
    - +Limit/-Limit/Home inputs
- Homing routines:
    - Home input only (high speed)
    - Home input only (high speed + low speed)
    - Limit only
    - Z-index encoder channel only
    - Home input + Z index encoder channel
- S-curve or trapezoidal acceleration profile control
- On-the-fly speed change
- 1000 line incremental encoder (4000 counts/rev with 4x quadrature decoding)
- Stepper driver
    - 12-35 VDC
    - 2.5 Amp max current setting (peak current)
    - 16 micro-step setting (fixed)
    - 400 KHz max pulse support
- Stepper motor
    - NEMA 17/23 motor sizes available in different stack sizes
    - 1.8° step angle

Model Numbers





<u>**Contacting Support**</u>

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

Or, contact your local distributor for technical support.

2. Electrical and Thermal Specifications

Power Requirement

Regulated Voltage:	**+12 to +35 VDC**
Current (Max):	**2.5 A (peak)**

Temperature Ratings †

Operating Temperature:	**0°C to +70°C**
Storage Temperature:	**-55°C to +150°C**

† Based on component ratings

Digital Inputs †

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

† Includes limit, home and latch

Digital Outputs

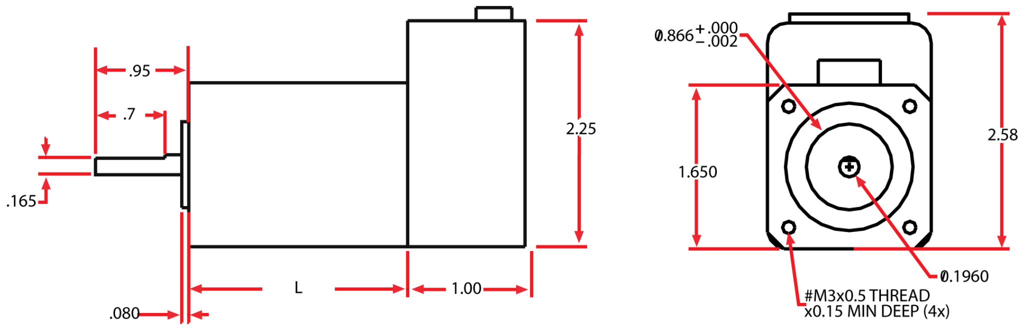
Type:	**Opto-isolated open-collector NPN outputs**
Max voltage at collector:	**+24 VDC**
Max source current at 24VDC	**†90 mA**

† A current limiting resistor is required

3. Dimensions

†All dimensions in inches

DMX-K-SA-17



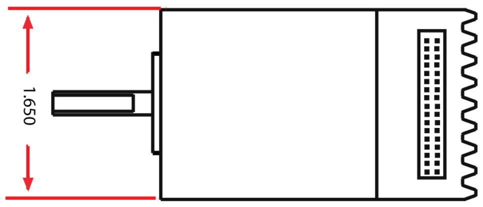
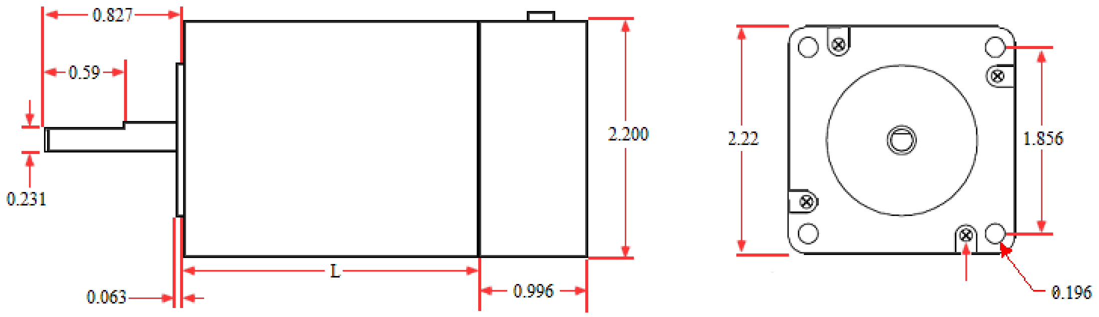


Figure 3.0

DMX-K-SA-23



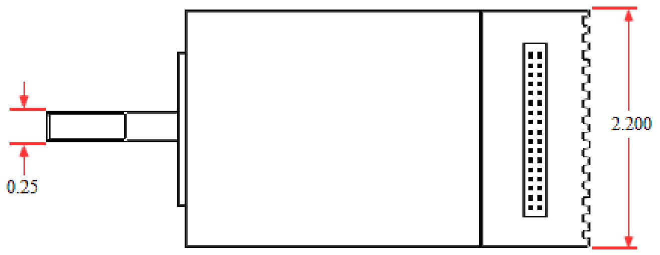


Figure 3.1

Model	L (inches)
DMX-K-SA-17-2	1.58
DMX-K-SA-17-3	1.89
DMX-K-SA-23-2	2.2
DMX-K-SA-23-3	3.1

Table 3.0

4. Motor Specifications

Electrical Specifications

NEMA Size	Stack Size	Current / Phase †	Holding Torque	Resistance/ Phase	Inductance/ Phase	Inertia
17	Double	1.7A	0.44 N-m	1.5 Ω	3.0 mH	0.28 oz-in^2
	Triple	2.0A	0.59 N-m	1.4 Ω	2.7 mH	0.37 oz-in^2
23	Double	2.8A	0.95 N-m	0.9 Ω	2.5 mH	1.64 oz-in^2
	Triple	2.8A	1.41 N-m	1.13 Ω	3.6 mH	2.62 oz-in^2

Table 4.0

† Motor current specifications are in RMS form.

Torque Curve – NEMA 17

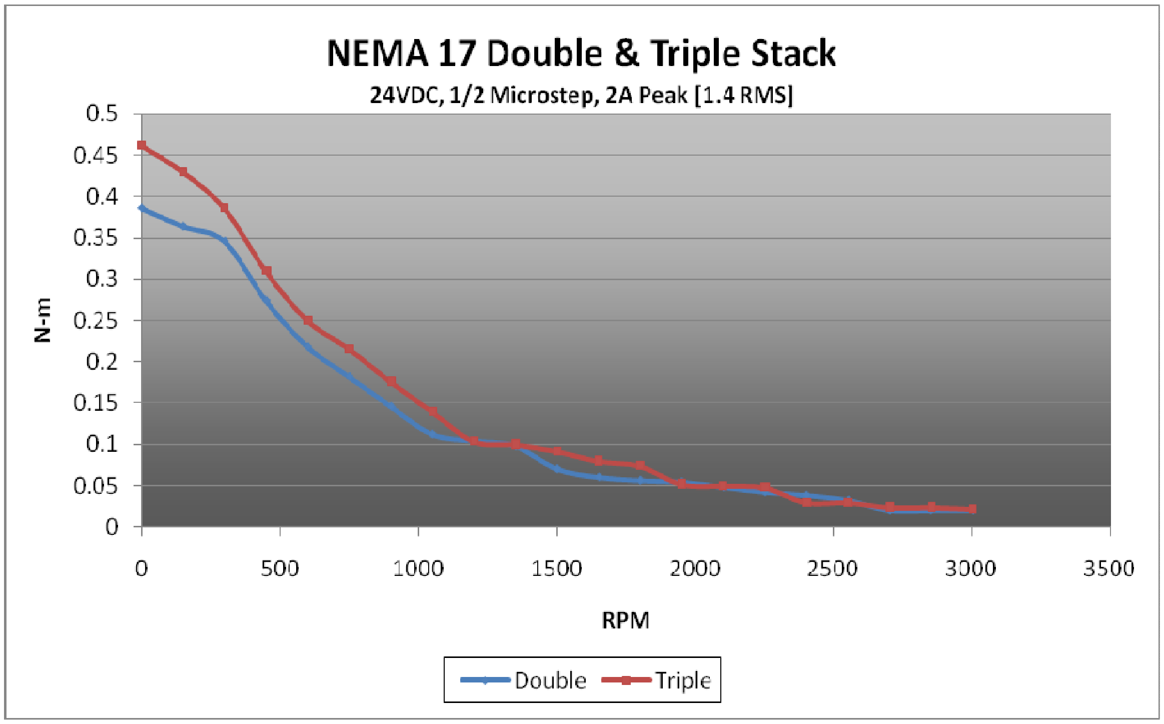


Figure 4.0

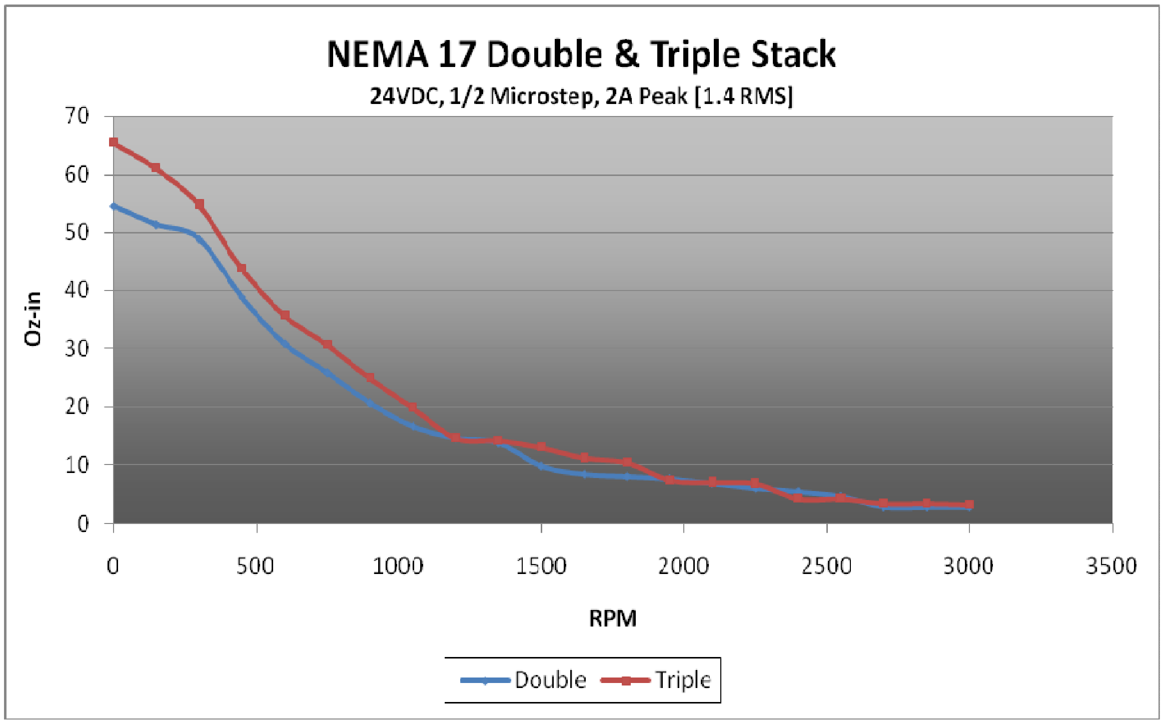


Figure 4.1

Torque Curve – NEMA 23

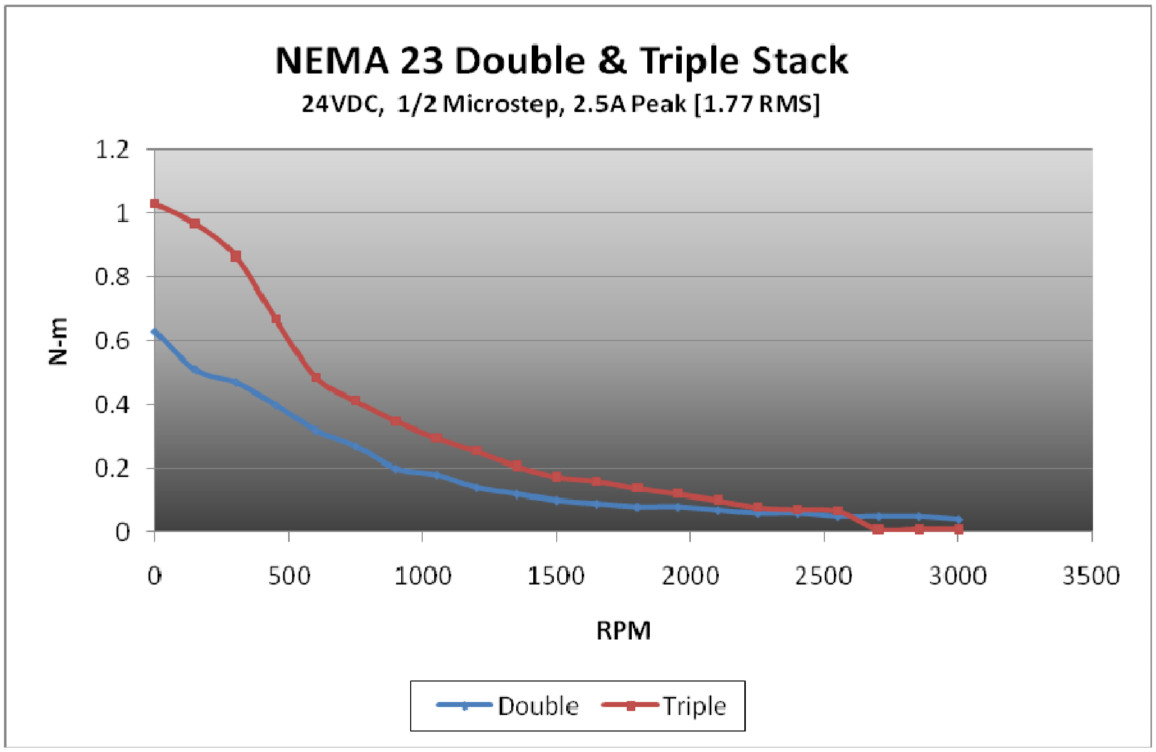


Figure 4.2

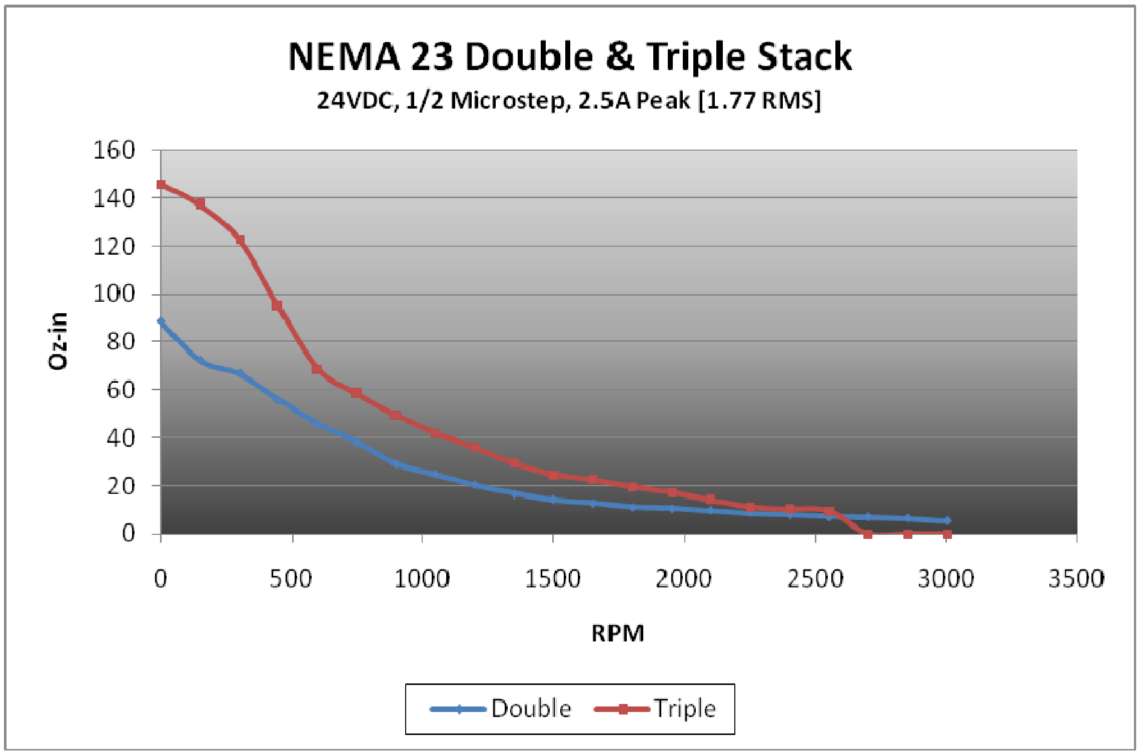

NEMA 23 Double & Triple Stack
24VDC, 1/2 Microstep, 2.5A Peak [1.77 RMS]
Oz-in
160
140
120
100
80
60
40
20
0
0
500
1000
1500
2000
2500
3000
3500
RPM
Double
Triple

Figure 4.3

5. Connections

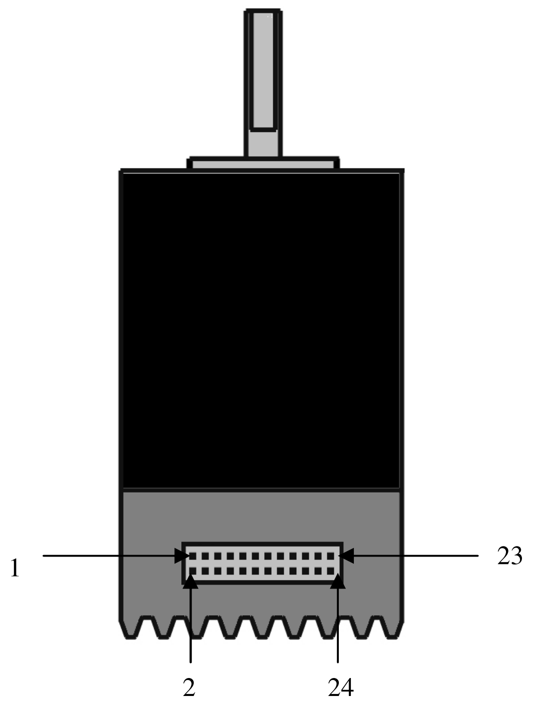


Figure 5.0

24-Pin Connector (2mm)

Pin #	In/Out	Name	Description
1	I	PWR	+12 to +35VDC power input
2	I	PWR	+12 to +35VDC power input. Shorted to pin 1
3	I	GND	Ground
4	I/O	485+	RS-485 plus signal
5	I	HOME	Home input
6	I/O	485-	RS-485 minus signal
7	O	DO1/INP	Digital Output 1
8	I	+LIM	Plus limit input
9	I/O	RXD	RS-232 RXD signal
10	I	-LIM	Minus limit input
11	I	DI1	Digital Input 1
12	I	DI3	Digital Input 3
13	I	DI2/LT	Digital Input 2 / Latch
14	I	DI4	Digital Input 4
15	NC	NC	No Connection
16	I	DI5	Digital Input 5

17	I	OPTO	+12-24VDC opto-supply
18	I	DI6	Digital Input 6
19	I/O	TXD	RS-232 TXD signal
20	O	DO2	Digital Output 2
21	I	OPTOGND	Opto-supply ground
22	O	DO3/ALM	Digital Output 3 / Alarm
23	NC	NC	No Connection
24	I	GND	Ground. Shorted to pin 3

Table 5.0

Mating Connector Description:	24 pin 2mm dual row connector
Mating Connector Manufacturer:	HIROSE
Mating Connector Housing Part Number:	DF11-24DS-2C
Mating Connector Pin Part Number:	DF11-2428SC

Junction Board

JBD-K-SA is a junction board that allows users to easily access the signals on the DMX-K-SA. The accessory comes with a 1ft long cable to connect to the DMX-K-SA.

JBD-K-SA is not included in the purchase of DMX-K-SA.

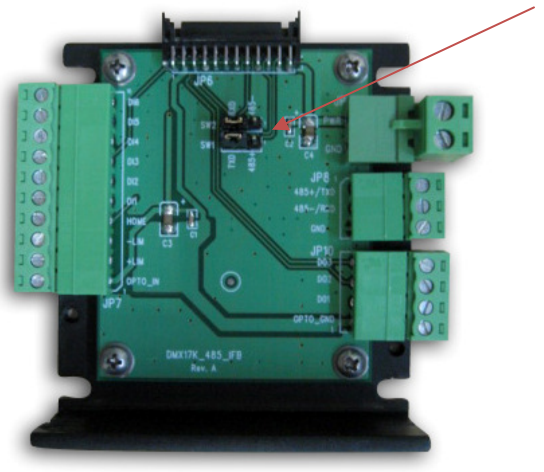


Figure 5.1

Note that JP8 is used for both RS-485 and RS-232 communication.   In order to switch between the two communication methods, the jumpers SW1 and SW2 must be placed on the appropriate pins. The configuration shown in Figure 5.1 is set for RS-232.

DMX-K-SA-17/23 Interface Circuit

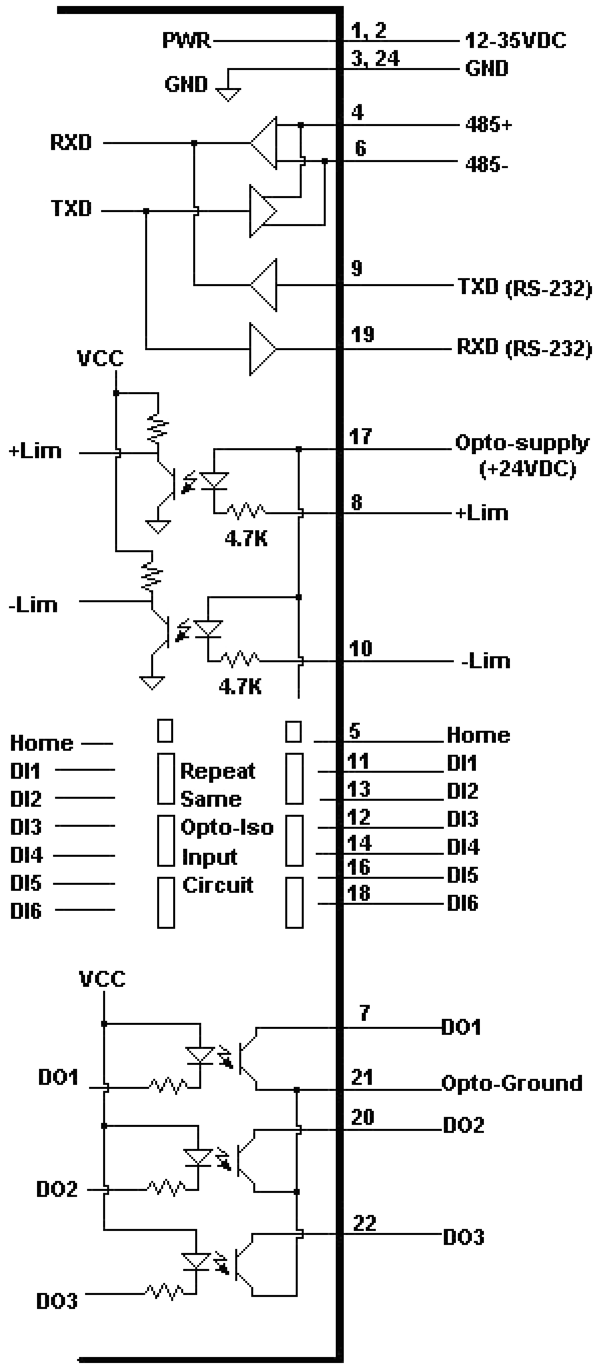


Figure 5.2

Digital Outputs

Figure 5.3 shows an example wiring to the digital output.

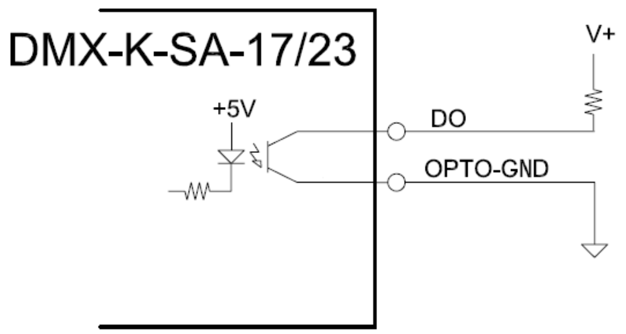


Figure 5.3

WARNING: The maximum sink current for digital outputs is 90 mA. Take caution to select the appropriate external supply and pull-up resistance to limit the sink current below this level.

Digital Inputs

Figure 5.4 shows the detailed schematic of the opto-isolated inputs.

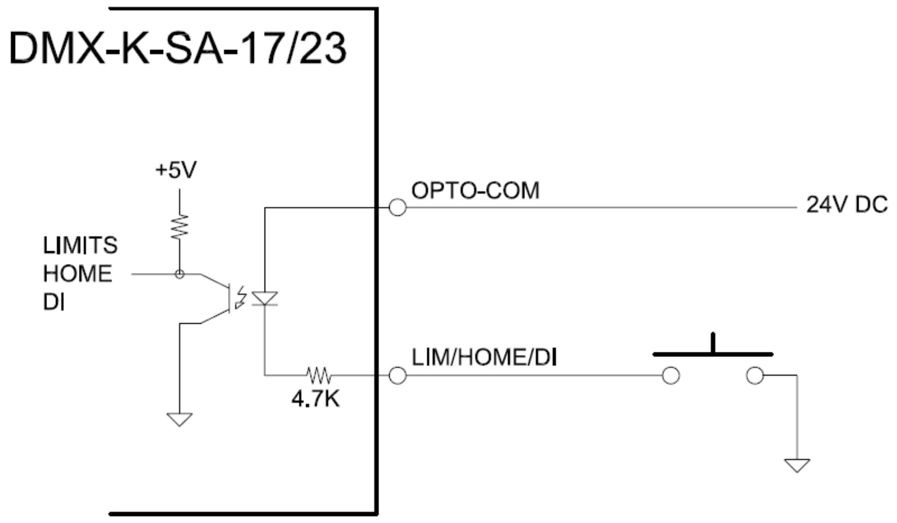


Figure 5.4

6. Getting Started

Typical Setup

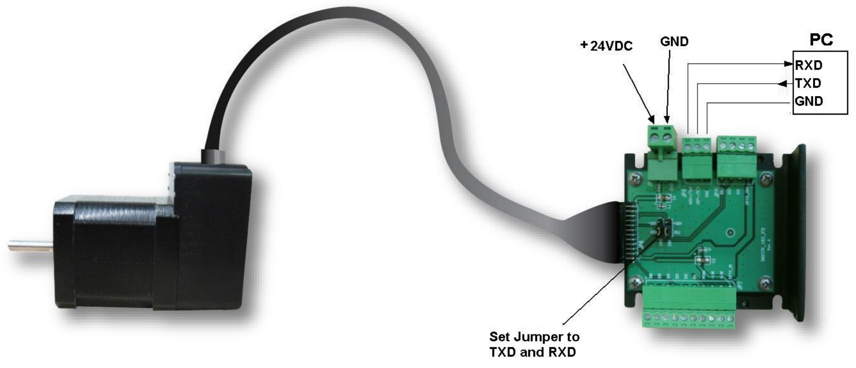


Figure 6.0

There are two ways to communicate with DMX-K-SA series product: RS-232 and RS-485.

RS-232

When the DMX-K-SA unit is shipped from the factory, default communication setting is RS-232 at 9600 baud rate.

Note that RS-232 is a point-to-point protocol. See figure below:

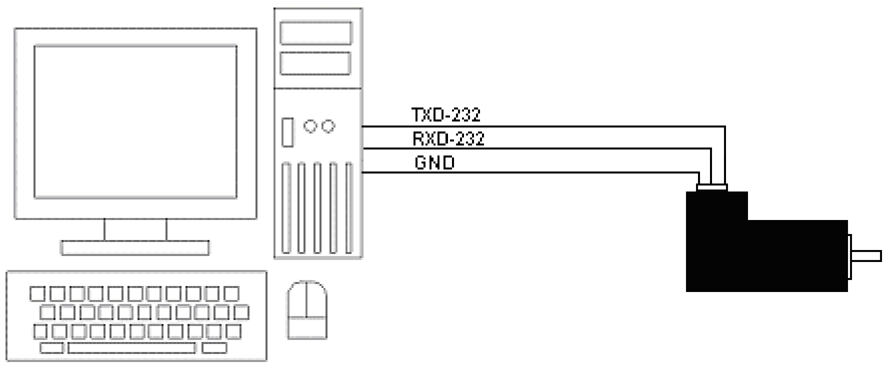


Figure 6.1

RS-485

If RS-485 communication is required, first you need to communicate using RS-232 and use the Windows program to change the communication method to RS-485, download the setup, and store to flash. Once communication method is changed, you need to reboot the module for the new parameter to take effect.

When communicating via RS-485, it is recommended to add a 120 Ohm terminating resistor between 485+ and 485- signal on the last module.

Below is a typical RS-485 master and multi-slave network.

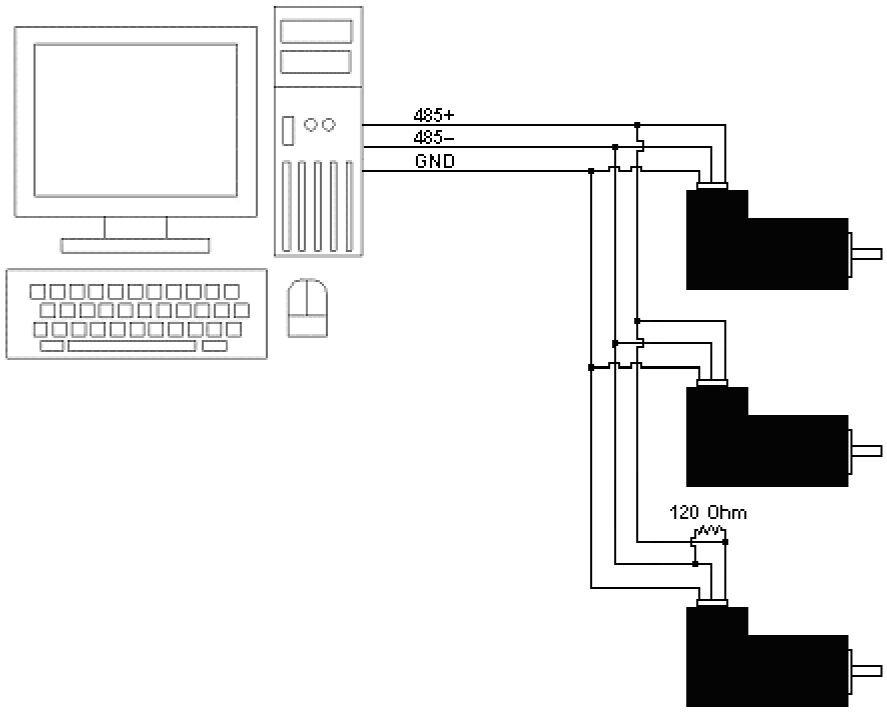


Figure 6.2

Windows GUI

DMX-K-SA comes with user friendly Windows Program to quickly communicate, test, program, and debug the DMX-K-SA unit.

Start the DMX-K-SA program.

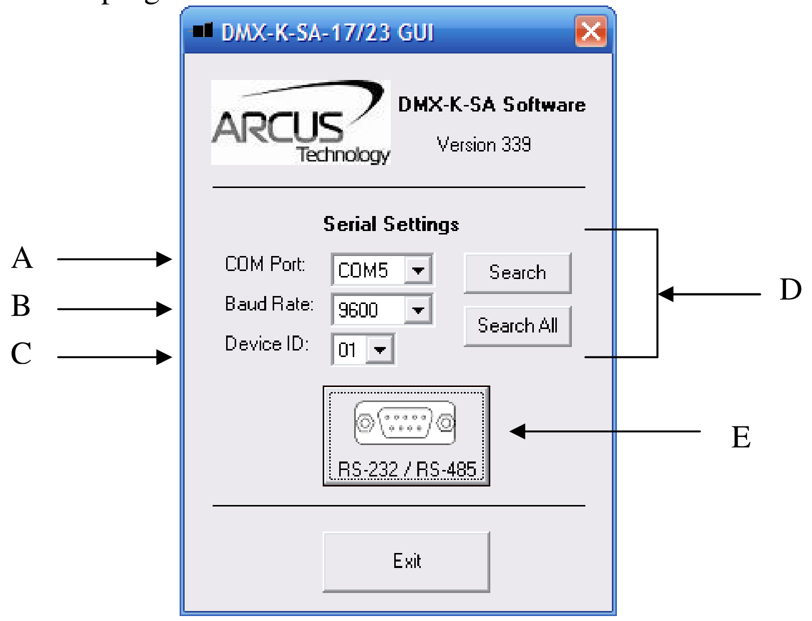


Figure 6.3

A. Select COM port
B. 9600 baud rate is the default communication baud rate that is used. If you have set your DMX-K-SA module to operate at a different baud rate, select the correct baud rate here.
C. Select the device ID (name) of your DMX-K-SA
D. If communication port or the baud rate is not known, use these buttons to search for the device.
E. Open communication with the DMX-K-SA module.

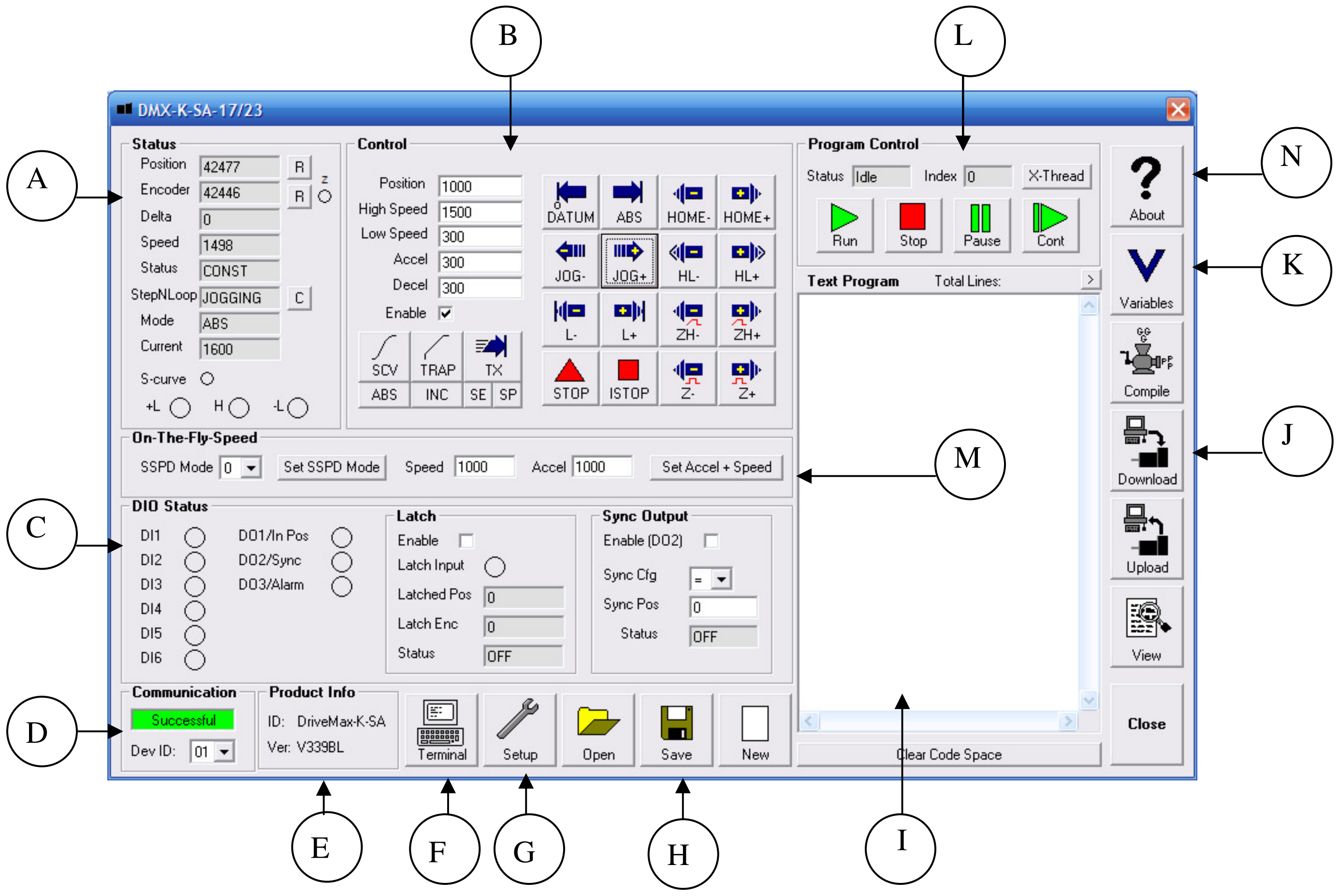
Note: When connecting for the first time, we recommend that a search is done to find out the COM port number where the DMX-K-SA is connected.

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.

Main Control Screen



Status
Position 42477
Encoder 42446
Delta 0
Speed 1498
Status CONST
StepNLoop JOGGING
Mode ABS
Current 1600
S-curve ○
+L ○ H ○ -L ○
Z ○

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

Communication
Successful
Dev ID: 01

Product Info
ID: DriveMax-K-SA
Ver: V339BL

Program Control
Status Idle
Index 0

Text Program
Total Lines:

Figure 6.4

A. Status

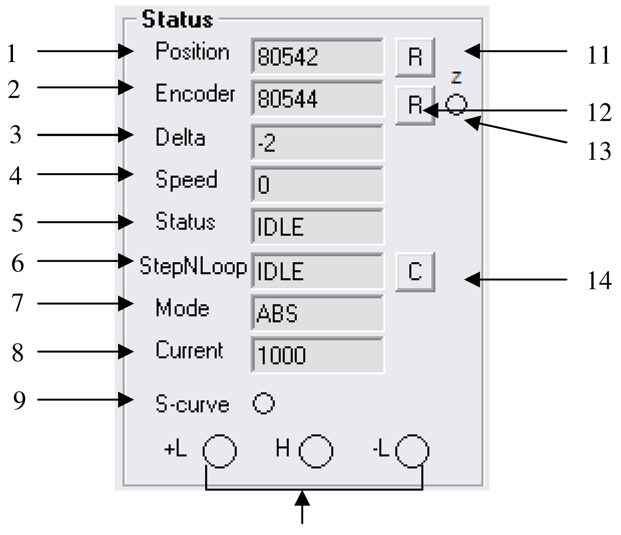
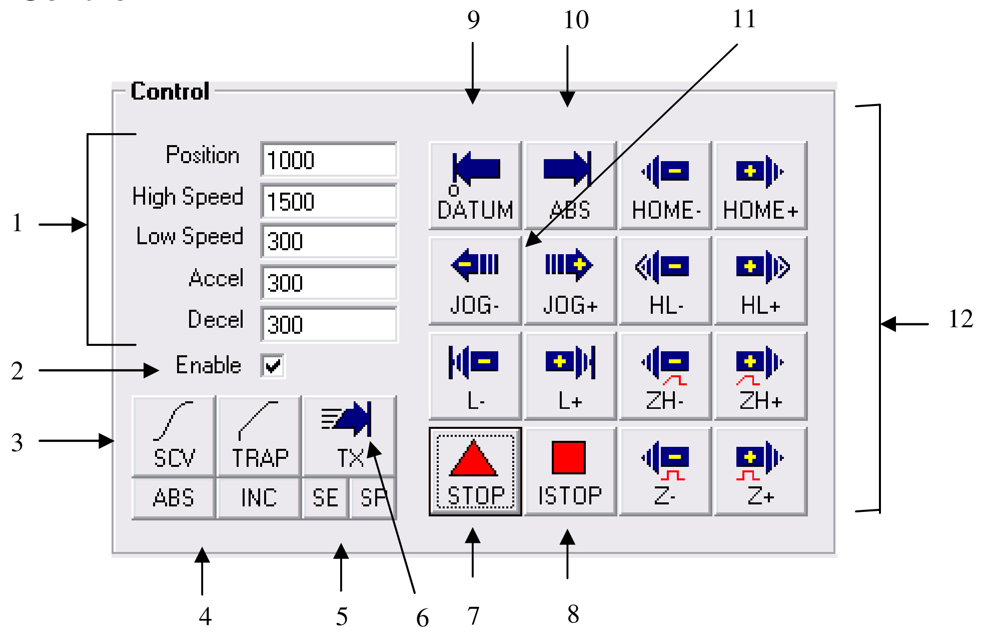


Figure 6.5

1. **Pulse Counter** – displays the current pulse position counter. When StepNLoop is enabled, this displays the Target position.
2. **Encoder Counter** – displays the current encoder position counter.
3. **Delta Counter** – 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. **Motion Status** – displays current motion 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 Status** – valid only when StepNLoop is enabled and displays current StepNLoop status by displaying one of the following:
    - 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. **Move 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.
11. **Reset Pulse Counter** – Pulse counter can be reset to zero using this button.
12. **Reset Encoder Counter** – Encoder counter can be reset to zero using this button.
13. **Encoder Z Index Channel Status** – Encoder Z index channel status is displayed.
14. **Reset StepNLoop and Motor Error** – When the StepNLoop or motor status is in error, use this button to clear the error. StepNLoop status will return to IDLE after error is cleared.

B. Control



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

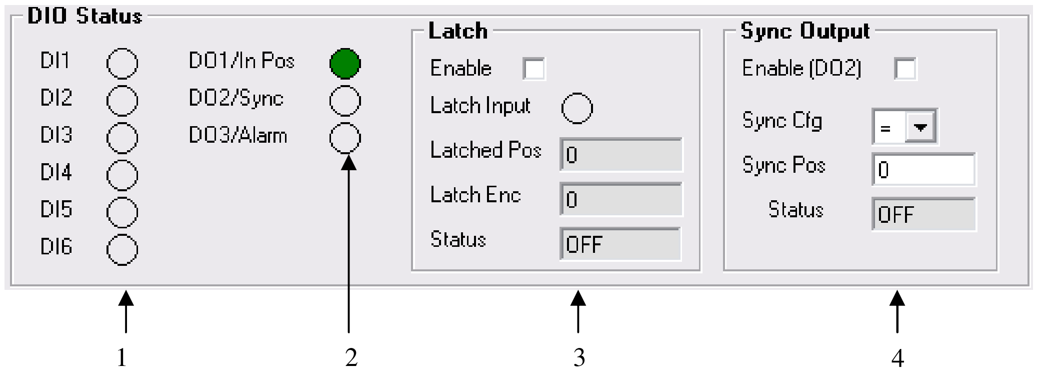
Figure 6.6

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.

C. Digital Input / Output



Latched Pos: 0
Latch Enc: 0
Status: OFF

Sync Cfg: =
Sync Pos: 0
Status: OFF

Figure 6.7

1. **Digital Input Status** – digital inputs

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

D. Communication

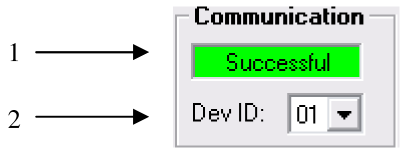
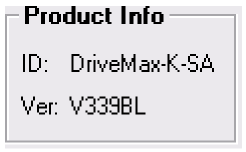


Figure 6.8

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

E. Product Info

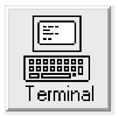


ID: DriveMax-K-SA
Ver: V339BL

Figure 6.9

Displays the product ID as well as the firmware version

F. Terminal



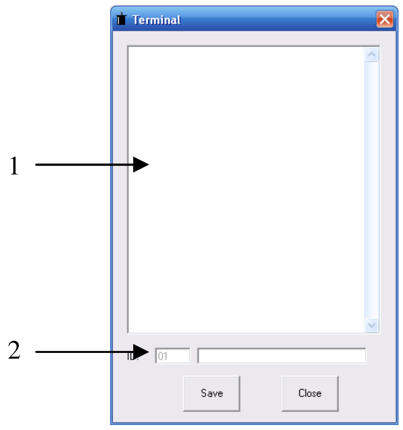
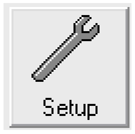


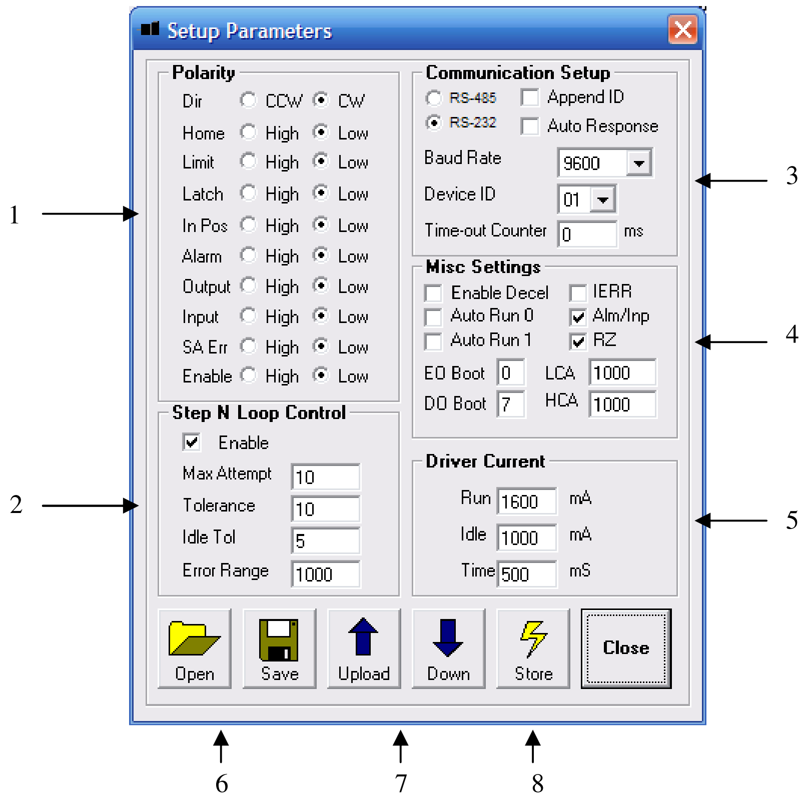
Figure 6.10

Terminal dialog box allows manual testing of the commands from a terminal screen as shown in Figure 6.10

1. **Response Window** – Displays the response, if any, from the command line
2. **Address** – 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.

G. 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	5
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	500	mS

Figure 6.11

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 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** – Following micro-step driver settings can be configured:
   - Run Current: 100mA to 2.5Amp
   - Idle Current: 100mA to 2.5Amp
   - Idle Time:1 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, it needs to be downloaded to take effect.
8. **Store** – The downloaded parameters can be permanently stored on the non-volatile memory.

H. Standalone Program File Management

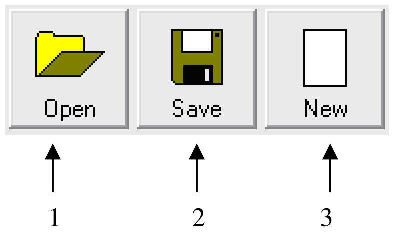


Figure 6.12

1. **Open** – Open standalone program

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

I. Standalone Program Editor

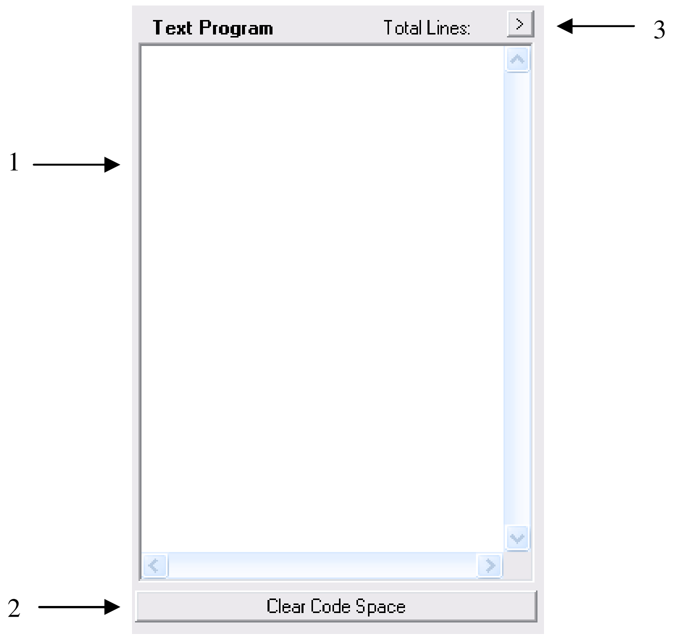


Figure 6.13

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.

J. Standalone Processing

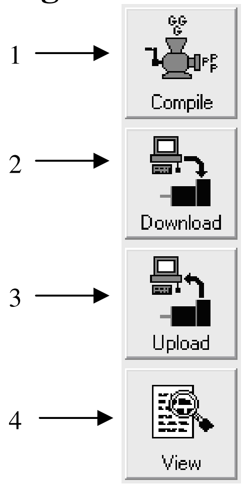
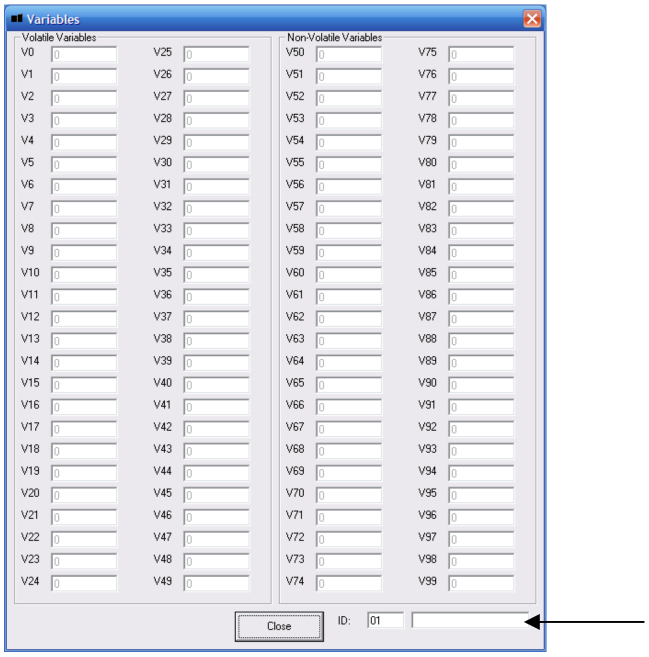


Figure 6.14

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

K. 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

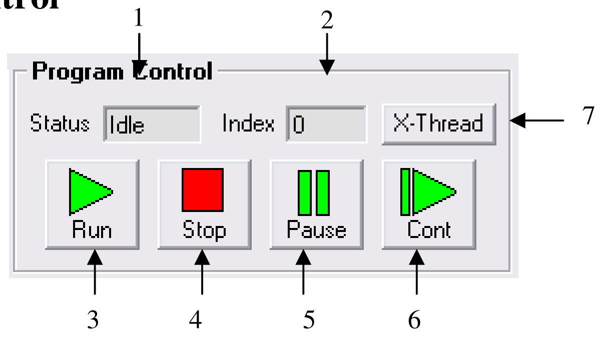
ID: 01

Figure 6.15

View the status of variables 0-99. Note that this window is read-only.

1. **Command line** – To write to variable, use V[0-99] = [value] syntax.

L. Program Control



Status Idle
Index 0

Figure 6.16

1. **Program Status** – program status shows here. Following are possible program status: Idle, Running, Errored and Paused.

2. **Index** – 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** – program is run.
4. **Stop** – program is stopped.
5. **Pause** – program that is running can be paused.
6. **Continue** – program that is paused can be continued
7. **X-Thread –** Open the Program Control for standalone multi-thread operation.

M. 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.

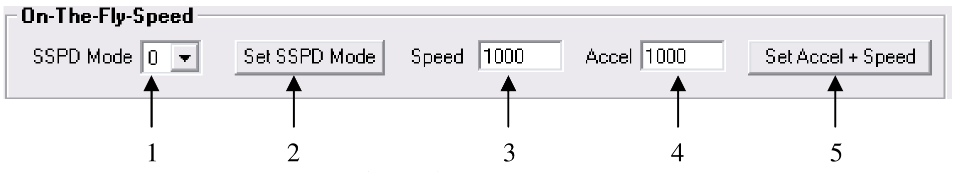


Figure 6.17

1. **On-the-fly speed mode** – Before setting the controller into motion, set the SSPDM parameter. To see which value to use, see the on-the-fly speed change section.
2. **Set SSPDM** – 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. **Desired Speed** – Once the "Set Speed" button is clicked, the speed will change on-the-fly to the desired speed.
4. **Desired Acc/Dec** – The acceleration/deceleration use for the on-the-fly speed change operation.
5. **Set SSPD[value]** – Start the on-the-fly speed operation

N. About

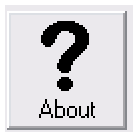


Figure 6.18

Click this button to display the GUI version as well as the firmware version of the controller/driver. If the firmware version is not up to date, the unsupported features will be listed.

7. Motion Control Overview

Important Note: All the commands described in this section are interactive commands and are not analogous to stand-alone commands. Refer to the "Standalone Language Specification" section for details regarding stand-alone commands.

Built-in encoder

DMX-K-SA comes with a 1000 line encoder. With quadrature decoding, 4000 count/rev resolution is reached. Use the **EX** command to read and set the encoder position. Pulse position can be read and set using the **PX** command.

When StepNLoop closed-loop control is *enabled*:

EX command returns encoder position
PX command returns the real-time target position of your move

When StepNLoop closed-loop control is *disabled*:

EX command returns encoder position
PX command returns pulse position

Built-in Microstep Driver

DMX-K-SA has an integrated micro-step driver. The micro-step setting is fixed at 16. With a 1.8° motor, this results in a 3200 step/rev resolution.

Motion Profile

By default, a trapezoidal velocity profile is used. See Figure 7.0.

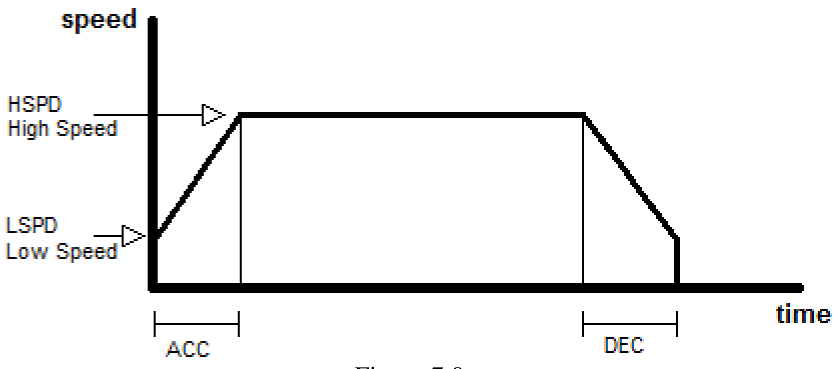


Figure 7.0

High speed and low speed are in pps (pulses/second). Use **HSPD/LSPD** commands to modify the high speed and low speed settings.

Acceleration and deceleration time are in milliseconds. Use the **ACC/DEC** command to modify the acceleration and deceleration values.

S-curve velocity profile can also be achieved by using the **SCV** command. See Figure 7.1.

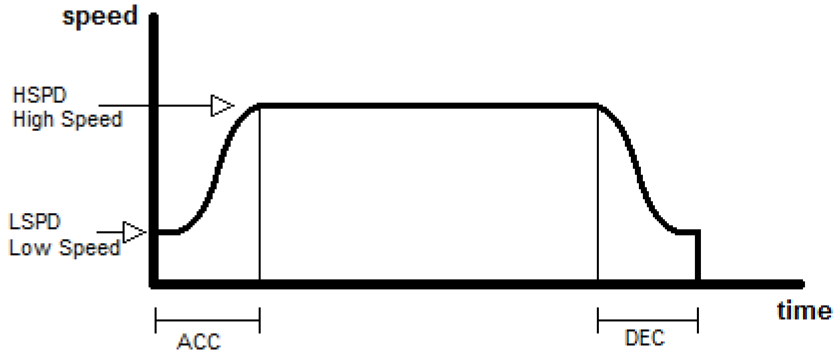


Figure 7.1

Notes:
By default, the deceleration is defined by the value set in the **ACC** parameter. In order to decelerate using the value set in the **DEC** parameter, set **EDEC** to 1.

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

On-The-Fly Speed Change

On-the-fly speed change can be achieved with the **SSPD** command. In order to use the **SSPD** command, s-curve velocity profile must be disabled.

SSPD Mode
The correct speed window must be selected in order to use the SSPD command. To select a speed window, use the **SSPDM** command. Refer to **Appendix A** for details.

During on-the-fly speed change operation, you must keep the initial and destination speeds within the speed window.

For non on-the-fly speed change moves, set **SSPDM=0**.

Digital Inputs/Outputs

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

Inputs
Read digital input status using the **DI** command.

Digital input values can also be referenced one bit at a time by the **DI[1-6]** commands. Note that the indexes are 1-based for the bit references (i.e. DI1 refers to bit 0, not bit 1)

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 7.0

Outputs

Use the **DO** command to drive digital outputs. DO value must be within the range of 0-7. Digital output values can also be referenced one bit at a time by the **DO[1-3]** commands. Note that the indexes are 1-based for the bit references (i.e. DO1 refers to bit 0, not bit 1)

Bit	Description	Bit-Wise Command
0	Digital Output 1 (In Position)	DO1
1	Digital Output 2	DO2
2	Digital Output 3 (Alarm)	DO3

Table 7.1

If StepNLoop control and **EDO** are enabled, DO1 is used as an “In Position” status output, and DO3 is used as an “Alarm” output.

To use DO1 and DO3 as general purpose outputs while StepNLoop is enabled, set **EDO=0.**

The initial state of the 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.

Motor Power

Using the **EO** command, the motor power can be enabled or disabled.

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

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 7.2

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

Positional Moves

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

Note: If a motion command is sent while the controller is already moving, the command is not processed. Instead, an error response is returned.

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.

Note: 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.

Jogging

Jogging is available for continuous speed operation. Use **J+** and **J-** commands to jog in positive or negative direction.

Stopping Motor

When the motor is moving, jogging, or homing, using the **ABORT** command will immediately stop the motor. Using the **STOP** command will decelerate the motor to low speed before stopping.

Homing

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

<u>Home Input Only (High Speed Only)</u>

Use the **H+/H-** command. Figure 7.2 shows the homing routine.

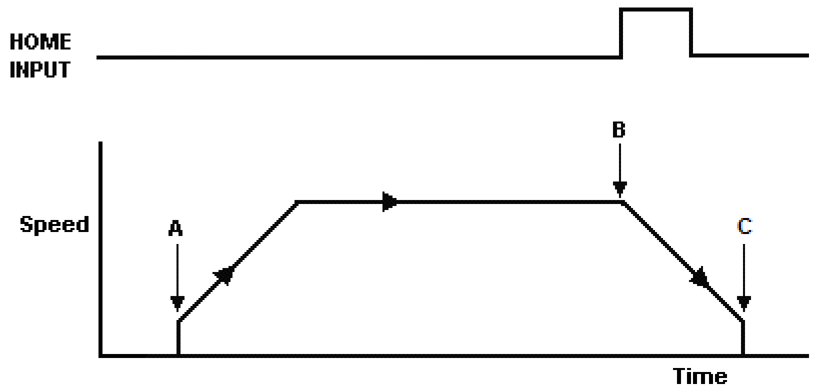


Figure 7.2

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.

Note: For **H**, **HL** homing routines, it is possible to have the motor automatically return to the zero position. To do so, set the **RZ=1**.

<u>Home Input and Z-index</u>

Use the **ZH+/ZH-** command. Figure 7.3 shows the homing routine.

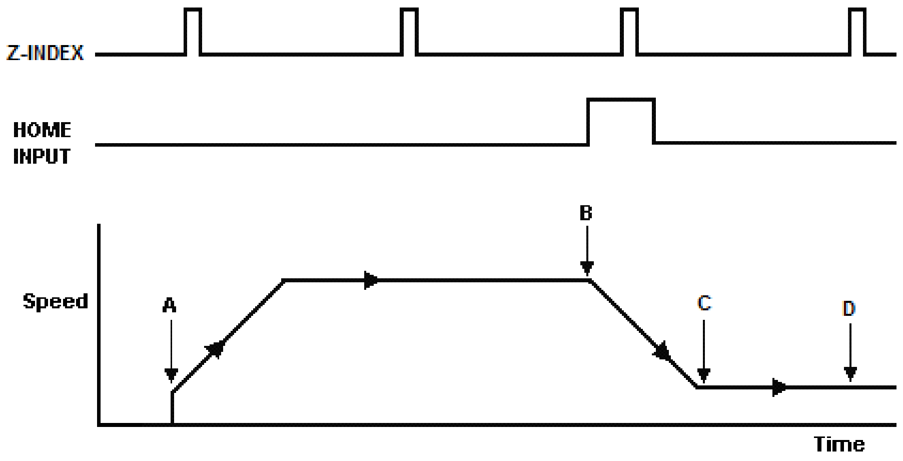


Figure 7.3

A. Issuing the command starts the motor from low speed and accelerates to high speed.
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.

Home Input Only (High Speed and Low Speed)

Use the **HL+/HL-** command. Figure 7.4 shows the homing routine.

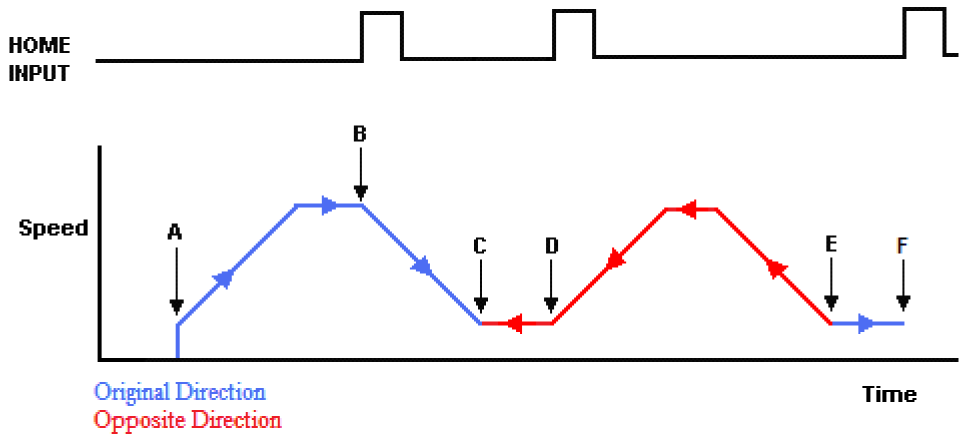


Figure 7.4

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

Note: For **H**, **HL** homing routines, it is possible to have the motor automatically return to the zero position. To do so, set the **RZ=1**.

Limit Only

Use the **L+/L-** command. Figure 7.5 shows the homing routine.

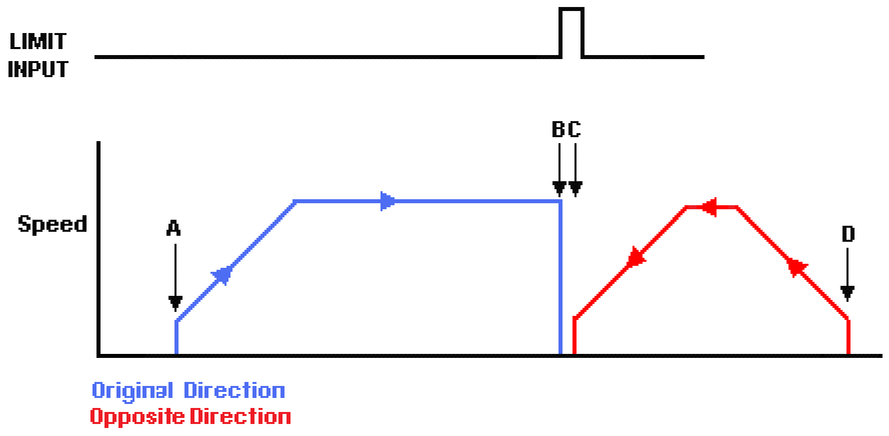


Figure 7.5

A. Issuing the command starts the motor from low speed and accelerates to high speed.
B. The corresponding limit is triggered and the motor stops immediately.
C. The motor reverses direction by the amount defined by the limit correction amount (**LCA**) at high speed.
D. The zero position is reached.

Z-Index Only

Use the **Z+/Z-** command. Figure 7.6 shows the homing routine.

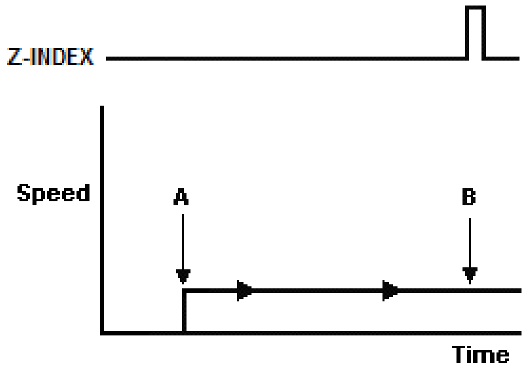


Figure 7.6

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.

Motor Position

Motor position can be set and read by using the **PX** command.
Encoder position can be set and read by using the **EX** command.

Motor Status

Motor status can be read anytime by reading the response to the **MST** command. The following is the bit representation of 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 using the **CLR** command 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 using the **CLR** command before issuing any subsequent move commands.
8	Latch input status
9	Z-index status
10	TOC time-out status

Table 7.3

Examples:

- When motor status value is 0, motor is idle and all input switches are off.
- When motor status value is 2, motor is in acceleration.
- When motor status value is 9, motor is moving in constant high speed and home input switch is on.
- When motor status value is 64, motor is in minus limit error. Use **CLR** command to clear the error before issuing any more move commands.

Limit Inputs

If 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 is for the negative limit while moving in the negative direction. Once the limit error is set, use the **CLR** command to clear the error. Once the error is cleared, move the motor out of the limit switch.

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.

Latch Input

DMX-K-SA has high speed position latch input DI2.

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

Note: When StepNLoop mode is enabled, the position value should be ignored.
Use the **LT** command to enable and disable latch feature. To read the latch status, use **LTS** command.

Following are return value description for **LTS** command.

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

Table 7.4

Once the latch is triggered, the triggered position can be retrieved using **LTP** (latched pulse position) and **LTE** (latched encoder position) commands.

Sync Output

DMX-K-SA has a designated synchronization digital output (DO2). The synchronization signal output is triggered when the encoder position value meets the set condition.

While this feature is enabled, the designated digital output (DO2) cannot be controlled by user.

Use **SYNO** to enable the synchronization output feature.

Use **SYNF** to disable the synchronization output feature.

Use **SYNP** to read and set the synchronization position value. (28-bit signed number)

Use **SYNC** to set the synchronization condition.

1 – Turn the output on when the encoder position is EQUAL to sync position.
If the synchronization output is done during motion, the sync output pulse will turn on only when the encoder position and sync position are equal.
2 – Turns output on when the encoder position is LESS than the sync position.
3 – Turns output on when the encoder position is GREATER than sync position.

Use **SYNS** to read the synchronization output status.

0 – Sync output feature is off
1 – Waiting for sync condition
2 – Sync condition occurred

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.

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 7.5 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**
Idle Tolerance	After correction, if the remaining error is greater than this value, an additional correction will be performed using idle current instead of run current.	**SLM**

Table 7.5

To enable/disable the SNL feature use the **SL** command. To read the SNL status, use **SLS** command to read the status.

See Table 7.6 for a list of the **SLS** 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 7.6

See Table 7.7 for SNL 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 and error.	Error Range Error. Motor stops and signals and error.
Correction Attempt > SLA	NA	Max Attempt Error. Motor stops and signals and error.

Table 7.7

Key

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

Notes:

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

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**.

Idle Current and Run Current

DMX-K-SA allows for two current settings.

Run Current: Current used while the motor is running. Set using the **CURR** command
Idle Current: Current used when the motor is idle. Set using the **CURI** command

To set the amount of time the motor needs to be idle before changing to idle current, use the **CURT** command. Units are in ms.

To read the actual current at anytime, use the **CUR** command

When setting idle and run current, the range must be within 100mA to 2500mA, unless the user wishes to have the motor become disabled during idle state. To do this, set the idle current to 0.

RS-232/RS-485 Selection:

In order to choose between RS-232 or RS-485 communication, use the **CM** command.

A value of 0 corresponds to RS-232 communication. A value of 1 corresponds to RS-485 communication. By default, the **CM** value is set to 0.

To write the value to flash memory, use the **STORE** command. After a complete power cycle, the new communication method will be used. Note that before a power cycle is done, the setting will not take effect.

Device Number

DMX-K-SA module provides the user with the ability to modify the unique device number. In order to make these changes, first store the desired number using the **DN** command. This value must be within the range [DMK01,DMK99].

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

By default: Device name is set to: **DMK01**

Baud Rate Setting

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 store the desired baud rate by using the **DB** command.

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

Table 7.9

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

By default: Baud rate is set to: **1 (9600 bps)**

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 a DMX-K-SA module, no response is sent back to the master.

Response type selection

It is possible to choose between two types of response string formats. This parameter can be set using the **RT** command.

Format 1 (default): [Response][Null]

Examples:

For querying the encoder position

```
Send:   @01EX[CR]
Reply:  1000[Null]
```

For jogging the motor in positive direction

```
Send:   @01J+[CR]
Reply:  OK[Null]
```

For aborting any motion in progress

```
Send:   @01ABORT[CR]
```

To achieve this response string type, send command **RT=0**.

Format 2: #[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 achieve this response string type, send command **RT=1**.

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.

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:

a) Device 01 is given an absolute move command from the master.
b) 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.

Standalone Programming

Standalone Program Specification:
Memory size: 1275 assembly lines ~ 7.5 KB.
Note: Each line of pre-compiled code equates to 1-4 lines of assembly lines.

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

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

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

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

```
X10000          ;* Move to position 0
WAITX           ;* Wait for the move to complete
V1=100
```

Multi-Threading: The DMX-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 the **Example Stand-alone Programs** section.

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

Error Handling: If an error occurs during standalone execution (i.e. limit error), the program automatically jumps to SUB 31. If SUB 31 is NOT defined, the program will cease execution and go to error state. If SUB 31 is defined by the user, the code within SUB 31 will be executed. The return jump line will be determined by bit 11 of the **POL** register. See Table 7.4 for details.

Calling subroutines over communication: Once a subroutine is written into the flash, they can be called via serial communication using the **GS** command. The subroutines are referenced by their subroutine number [0-31]. If a subroutine number is not defined, the controller will return with an error.

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

Bit	Description
0	Standalone Program 0
1	Standalone Program 1

Table 7.10

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**.

Storing to Flash

The following items are stored to flash:

ASCII Command	Description
AR	Auto-response for in-position
CM	Communication method (RS-232/RS-485)
CURI,CUR,CURT	Driver current settings
DB	Serial communication baud rate
DN	Device name
DNM	†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	†Modbus enable
RT	ASCII response type
RZ	Return to zero position after homing
SL, SLR, SLE, SLT, SLA, SLM	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 7.11

† See "Modbus_Addition_Addendum_A" document for details.

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

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
CM	Communication Method	RS-232
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 Pos 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	Enabled
SLA	StepNLoop Maximum Attempt	10
SLE	StepNLoop Error Range	1000
SLM	StepNLoop Idle Tolerance	5
SLOAD	Run Program(s) on Power Up	0
SLT	StepNLoop Tolerance Range	20
TOC	Time-out counter value (Watch-dog)	0
V50-V99	Non-volatile variables	0

Table 7.12

8. Communication Protocol

Communication protocol and commands are the same for both RS-232 and RS-485.

Sending Command to DMX-K-SA

ASCII command string in the format of

@[DeviceName][ASCII Command][CR]

[CR] character has ASCII code 13.

Receiving Reply from DMX-K-SA

The response will be in the format of

[Response][Null]

[Null] character has ASCII code 0.

Examples:

For querying the encoder position

```
Send:   @01EX[CR]
Reply:  1000[Null]
```

For jogging the motor in positive direction

```
Send:   @01J+[CR]
Reply:  OK[Null]
```

For aborting any motion in progress

```
Send:   @01ABORT[CR]
Reply:  OK[Null]
```

Notes:

The address **'00'** is reserved for broadcasting over a 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 a DMX-K-SA module, no response is sent back to the master.

There is an option to have the device name appended to the beginning of the response string. This can be achieved by using the **RT** command. See "Response Type Selection"

9. ASCII Language Specification

Important Note: All the commands described in this section are interactive commands and are not analogous to stand-alone commands. Refer to the "Standalone Language Specification" section for details regarding stand-alone commands.

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

Append "@XX" to the command before sending, where "XX" is the device number. Ex: To send the "J+" command to device number 05, send the following: **"@05J+"**

Command	Description	Return
ABORT	Immediately stops the motor if in motion. For decelerate stop, use STOP command. This command can also be used to clear a StepNLoop error	OK
ABS	Set move mode to absolute	OK
ACC	Returns current acceleration value in milliseconds.	Milli-seconds
ACC=[Value]	Sets acceleration value in milliseconds. Example: ACC=300	OK
AR	Get auto-response feature status	1 or 0
AR=[0 or 1]	Set auto-response feature status	OK
CLR	Clears limit error as well as StepNLoop error	OK
CM	Get RS-232/RS-485 communication mode	1 or 0
CM=[0 or 1]	Set RS-232/RS-485 communication mode 0 – RS-232 1 – RS-485	OK
CUR	Get real-time current	0mA, 100mA to 2500mA
CURI	Get idle current setting	0mA, 100mA to 2500mA
CURI=[Value]	Set idle current. To have motor become disabled during idle state, set this value to 0.	OK
CURR	Get run current setting	0mA, 100mA to 2500mA
CURR=[Value]	Set run current	OK
CURT	Get driver idle time setting	milliseconds
CURT=[Value[	Set driver idle time setting	OK
DB	Return the current baud rate of the device	See Table 6.10
DB=[Value]	Set the baud rate of the device	OK
DEC	Get deceleration value in milliseconds. Only used if EDEC=1	Milli-seconds
DEC=[Value]	Set deceleration value in milliseconds. Only used if EDEC=1	OK
DI	Return status of digital inputs	See Table 6.2
DI[1-6]	Get individual bit status of digital inputs	0,1
DO	Return status of digital outputs	3-bit number
DO=[Value]	Set digital output 3 bit number. Note that DO1 and DO3 can only be used as general purpose when EDO=0 or SNL is disabled	OK
DO[1-3]	Return status of individual digital output	1 or 0
DO[1-3] = [Value]	Set individual digital output	OK
DOBOOT	Get DO boot-up state	See Table 6.3
DOBOOT=[Value]	Set DO boot-up state	OK
DN	Get device name	[DMK01-DMK99]
DN=[Value]	Set device name	OK
DNM	Get 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.	1 – Motor power enabled 0 – Motor power disabled
EO=[0 or 1]	Enables (1) or disable (0) motor power.	OK
EOBOOT	Get EO boot-up state	0 or 1
EOBOOT=[Value]	Set EO boot-up state	OK
EDEC	Get unique deceleration enable	0 or 1
EDEC=[Value]	Set unique deceleration enable	OK
EDO	Returns enable alarm/in pos mode status	1 – enabled 0 – disabled
EDO=[0 or 1]	Enables (value 1) or disable (value 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]	Sets High Speed.	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 with low speed)	OK
HL-	Homes the motor in negative direction (with low speed)	OK
IERR	Get ignore limit error enable	0 or 1
IERR=[Value]	Set ignore limit error enable	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 homing in positive direction	OK
L-	Limit homing in negative direction	OK
LCA	Returns the limit correction amount	28-bit number
LCA=[Value]	Sets the limit correction amount.	OK
LSPD	Returns Low Speed Setting	PPS
LSPD=[Value]	Sets Low Speed	OK
LT=[0 or 1]	Enable or disable position latch feature	OK
LTE	Returns latched encoder position	28-bit number
LTP	Returns latched pulse position	28-bit number
LTS	Returns latch status.	See Table 6.6
MM	Get move mode status	0 – Absolute move mode 1 – Incremental move mode
MST	Returns motor status	See Table 6.5
POL	Returns current polarity	See Table 6.4
POL=[value]	Sets polarity.	OK
PS	Returns current pulse speed	PPS
PX	Returns current position value	28-bit number
PX=[value]	Sets the current position value	OK
RSM	Get Modbus enable	0 or 1
RSM= [0 or 1]	Set Modbus enable	OK
RT	Get response type value	0 or 1
RT= [0 or 1]	Set response type value	OK

RZ	Get return zero enable. Used during homing	0 or 1
RZ=[0 or 1]	Set return zero enable. Used during homing	OK
SASTAT[0,1]	Get standalone program status 0 – Stopped 1 – Running 2 – Paused 4 – In Error	0-4
SA[LineNumber]	Get standalone line LineNumber: [0,1274]	
SA[LineNumber]=[Value]	Set standalone line LineNumber: [0,1274]	
SCV	Returns the s-curve control	0 or 1
SCV=[0 or 1]	Enable or disable s-curve. If disabled, trapezoidal acceleration/ deceleration will be used.	OK
SL	Returns StepNLoop enable status	0 – StepNLoop Off 1 – StepNLoop On
SL=[0 or 1]	Enable or disable StepNLoop Control	OK
SLA	Returns maximum number of StepNLoop control attempt	28-bit number
SLA=[value]	Sets maximum number of StepNLoop control attempt	OK
SLE	Returns StepNLoop correction range value.	28-bit number
SLE=[value]	Sets StepNLoop correction range value.	OK
SLM	Returns StepNLoop idle tolerance value	28-bit number
SLM=[value]	Set StepNLoop idle tolerance value	OK
SLR	Returns StepNLoop ratio value	[0.001 – 999.999]
SLR=[factor]	Sets StepNLoop ratio value. Must be in the range [0.001 – 999.999]	OK
SLS	Returns current status of StepNLoop control	See Table 6.8
SLT	Returns StepNLoop tolerance value	32-bit
SLT=[value]	Sets StepNLoop tolerance value.	OK
SLOAD	Returns RunOnBoot parameter	See Table 6.12
SLOAD=[value]	Set RunOnBoot parameter	See Table 6.12
SPC[0,1]	Get program counter for standalone program	[0-1274]
SR[0,1]=[Value]	Control standalone program: 0 – Stop standalone program 1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program	OK
SSPD[value]	On-the-fly speed change. In order to use this command, S-curve control must be disabled. Use SCV command to enable and disable s-curve acceleration/ deceleration control. Note that an “=” sign is not used for this command.	OK
SSPDM	Return on-the-fly speed change mode	[0-9]
SSPDM=[value]	Set on-the-fly speed change mode	OK
STOP	Stops the motor using deceleration if in motion.	OK
STORE	Store settings to flash	OK
SYNC	Read sync output configuration 1 – trigger when encoder equals position 2 – trigger when encoder is LESS than position 3 – trigger when encoder is GREATER than position	1,2,3
SYNC=	Set sync output configuration 1 – trigger when encoder equals position 2 – trigger when encoder is LESS than position 3 – trigger when encoder is GREATER than position	OK
SYNF	Turn off sync output	OK

SYNO	Turn on sync output	OK
SYNP	Get trigger position	28 bit signed number
SYNP=	Set trigger position	28 bit signed number
T[value]	On-the-fly target change	OK
TOC	Get time-out counter (ms)	32-bit number
TOC=[value]	Set time-out counter (ms)	OK
V[0-99]	Read variables 0-99	28-bit number
V[0-99]=[value]	Set variables 0-99	OK
VER	Get firmware version	VXXX
X[value]	Moves the motor to absolute position value using the HSPD, LSPD, and ACC values.	OK
Z+	Homes the motor in positive direction using the Z index encoder channel ONLY.	OK
Z-	Homes the motor in negative direction using the Z index encoder channel ONLY.	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

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 Language Specification

;

Description:

  Comment notation. In programming, comment must be in its own line.

Syntax:

```
; [Comment Text]
```

Examples:

```
; ***This is a comment
JOGX+                  ;***Jogs axis to positive direction
```

ABORTX

Description:

  Motion: Immediately stop motion without deceleration.

Syntax:

```
ABORTX
```

Examples:

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

ABS

Description:

  Command: Changes all move commands to absolute mode.

Syntax:

```
ABS
```

Examples:

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

ACC

Description:

  Read: Get acceleration value

  Write: Set acceleration value.

Syntax:

  Read: [variable] = ACC

  Write: ACC = [value]

  ACC = [variable]

Examples:

```
ACC=300      ;***Sets the acceleration to 300 milliseconds
V3=500       ;***Sets the variable 3 to 500
ACC=V3       ;***Sets the acceleration to variable 3 value of 500
```

DEC

Description:

Read: Get deceleration value
Write: Set deceleration value.
Value is in milliseconds.

Syntax:

Read: [variable] = DEC
Write: DEC = [value]
DEC = [variable]

Examples:

```
DEC=300      ;***Sets the deceleration to 300 milliseconds
V3=500       ;***Sets the variable 3 to 500
DEC=V3       ;***Sets the deceleration to variable 3 value of 500
```

DELAY

Description:

Set a delay (1 ms units)

Syntax:

Delay=[Number] (1 ms units)

Examples:

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

DI

Description:

Read: Gets the digital input value. DMX-K-SA has 6 digital inputs.

Digital inputs are active high

Syntax:

Read: [variable] = DI

Conditional:

```
IF DI=[variable]
ENDIF

IF DI=[value]
ENDIF
```

Examples:

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

DI[1-6]

Description:

Read: Gets the digital input value. DMX-K-SA has 6 digital inputs.

Syntax:

Read: [variable] = DI[1-6]

Conditional:
```
IF DI[1-6]=[variable]
ENDIF
IF DI[1-6]=[0 or 1]
ENDIF
```

Examples:
```
IF DI1=0
        DO=1            ;***If digital input 1 is triggered, set DO=1
ENDIF
```

DO

Description:

Read: Gets the digital output value

Write: Sets the digital output value

DMX-K-SA has 3 digital outputs.

Syntax:

Read: [variable] = DO

Write:
```
DO = [value]
DO = [variable]
```

Conditional:
```
IF DO=[variable]
ENDIF
IF DO=[value]
ENDIF
```

Examples:
```
DO=3            ;***Set DO to 3
```

DO[1-3]

Description:

Read: Gets the individual digital output value

Write: Sets the individual digital output value

DMX-K-SA has 3 digital outputs.

Syntax:

Read: [variable] = DO[1-3]

Write:
```
DO[1-3] = [0 or 1]
DO[1-3] = [variable]
```

Conditional:
```
IF DO[1-3]=[variable]
ENDIF
IF DO[1-3]=[0 or 1]
ENDIF
```

Examples:

```
DO1=1           ;***Turn DO1 on
DO2=1           ;***Turn DO2 on
```

DRVIC

Description:

Write: Sets the driver idle current parameter

Syntax:

Write: DRVIC=[value]

Examples:

```
WHILE 1=1
        IF DI1 = 0                    ;***If DI1 is triggered, execute
                SL=0                  ;***Disable StepNLoop
                DRVIT=1               :***Idle-time set to 1 cent-sec
                DRVIC=100             ;***Idle-current set to 100 mA
                DRVRC=1000            ;***Run-current set to 1000 mA
        ENDIF
ENDWHILE
```

DRVIT

Description:

Write: Sets the driver idle time parameter

Syntax:

Write: DRVIT=[value]

Examples: See DRVIC

DRVRC

Description:

Write: Sets the driver run current parameter

Syntax:

Write: DRVRC=[value]

Examples: See DRVIC

ECLEARX

Description:

Write: Clears motor error status. Also clears a StepNLoop error.

Syntax:

Write: ECLEARX

Examples:

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

ELSE

Description:

Perform ELSE condition check as a part of IF statement

Syntax:

ELSE

Examples:

```
IF V1=1
        X1000           ;***If V1 is 1, then move to 1000
        WAITX
ELSE
        X-1000          ;***If V1 is not 1, then move to -1000
        WAITX
ENDIF
```

ELSEIF

Description:

Perform ELSEIF condition check as a part of the IF statement

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

```
IF V1=1
        X1000
        WAITX
ELSEIF V1=2
        X2000
        WAITX
ELSEIF V1=3
        X3000
        WAITX
ELSE
        X0
        WAITX
ENDIF
```

END

Description:

Indicate end of program.
Program status changes to idle when END is reached.

Note: Subroutine definitions should be written AFTER the END statement

Syntax:

```
END
```

Examples:

```
X0
WAITX
X1000
WAITX
END
```

ENDIF

Description:

Indicates end of IF operation

Syntax:

```
ENDIF
```

Examples:

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

ENDSUB

Description:

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

Notes: Subroutine definitions should be written AFTER the END statement
Sub 31 is reserved for error handling

Syntax:

```
ENDSUB
```

Examples:

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

ENDWHILE

Description:

Indicate end of WHILE loop

Syntax:

ENDWHILE

Examples:

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

EO

Description:

Read: Gets the enable output value

Write: Sets the enable output value

Syntax:

Read: [variable] = EO

Write: EO = [value]

EO = [variable]

Conditional: IF EO=[variable]

ENDIF

IF EO=[value]

ENDIF

Examples:

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

EX

Description:

Read: Gets the current encoder position

Write: Sets the current encoder position

Syntax:

Read: [variable] = E[axis]

Write: EX = [value]

EX = [variable]

Conditional: IF EX=[variable]

ENDIF

IF EX=[value]

ENDIF

Examples:

```
EX=0                ;***Sets the current encoder position to 0
```

GOSUB

Description:

Perform go to subroutine operation
Subroutine range is from 0 to 31.
Notes: Subroutine definitions should be written AFTER the END statement
Sub 31 is reserved for error handling

Syntax:

GOSUB [subroutine number]
[Subroutine Number] range is 0 to 31

Examples:

```
GOSUB 0
END

SUB 0
        X0
        WAITX
        X1000
        WAITX
ENDSUB
```

HLHOMEX[+ or -]

Description:

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

Syntax:

HLHOMEX[+ or -]

Examples:

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

HOMEX[+ or -]

Description:

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

Syntax:

HOMEX[+ or -]

Examples:

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

HSPD

Description:

Read: Gets high speed. Value is in pulses/second

Write: Sets high speed. Value is in pulses/second.

Syntax:

Read: [variable] = HSPD

Write: HSPD = [value]
HSPD = [variable]

Examples:

```
HSPD=10000  ;***Sets the high speed to 10,000 pulses/sec

V1=2500     ;***Sets the variable 1 to 2,500
HSPD=V1     ;***Sets the high speed to variable 1 value of 2500
```

IF

Description:

Perform IF condition check

Syntax:

IF [Argument 1] [Comparison] [Argument 2]

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

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

INC

Description:

Command: Changes all move commands to incremental mode.

Syntax:

```
INC
```

Examples:

```
INC                   ;***Change to incremental mode
PX=0                  ;***Change position to 0
X1000                 ;***Move axis to position 1000 (0+1000)
WAITX
X2000                 ;***Move axis to position 3000 (1000+2000)
WAITX
```

JOGX[+ or -]

Description:

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

Syntax:

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

Examples:

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

LHOMEX[+ or -]

Description:

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

Syntax:

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

Examples:

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

LSPD

Description:

Read: Get low speed. Value is in pulses/second.

Write: Set low speed. Value is in pulses/second.

Syntax:

Read: [variable]=LSPD

Write: LSPD=[long value]

LSPD=[variable]

Examples:

```
LSPD=1000   ;***Sets the start low speed to 1,000 pulses/sec
V1=500      ;***Sets the variable 1 to 500
LSPD=V1     ;***Sets the start low speed to variable 1 value of 500
```

LTX

Description:

Write: Set latch enable
Range is [0,1]

Syntax:

Write: LTX=[0,1]
LTX=[variable]

Examples:

```
LTX=1                          ;***Enable latch
WHILE 1=1
    V2=LTSX                    ;***Get latch status
    IF LTSX = 2
        V3=LTEX                ;***Get latch encoder value if latch is triggered
        V4=LTPX                ;***Get latch position value if latch is triggered
    ENDIF
ENDWHILE
```

LTEX

Description:

Read: Get latch encoder value

Syntax:

Read: [variable]=LTEX

Examples:

See LTX

LTPX

Description:

Read: Get latch position value

Syntax:

Read: [variable]=LTPX

Examples:

See LTX

LTSX

Description:

Read: Get latch status

Syntax:

Read: [variable]=LTSX

Examples:

See LTX

MSTX

Description:

Read: Get motor status

Syntax:

Read: [variable]=MSTX

Conditional: IF MSTX=[variable]
ENDIF
IF MSTX=[value]
ENDIF

Examples:

```
IF MSTX=0
        DO=3
ENDIF
```

PX

Description:

Read: Gets the current pulse position

Write: Sets the current pulse position

Syntax:

Read: Variable = PX

Write: PX = [value]
PX = [variable]

Conditional: IF PX=[variable]
ENDIF
IF PX=[value]
ENDIF

Examples:

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

PS

Description:

Read: Get the current pulse speed

Syntax:

Read: Variable = PS

Conditional: IF PS=[variable]
ENDIF
IF PS=[value]
ENDIF

Examples:

```
JOGX+           ;***Jogs axis to positive direction
DELAY=1000      ;***Wait 1 second
ABORTX          ;***Stop without deceleration
V1=PS           ;***Sets variable 1 to pulse speed
```

SCVX

Description:

Write: Set s-curve enable.
Range is from 0 or 1

Syntax:

Write: SCVX=[0 or 1]
SCVX=[variable]

Note: If s-curve is enabled for an axis, on-the-fly speed feature can not be used for the corresponding axis.

Examples:

```
SCVX=1        ;***Sets axis to use s-curve acceleration: on-the-fly speed
              ; change is NOT allowed for this axis.
```

SLX

Description:

Write: Set StepNLoop closed-loop mode
Range is from 0 or 1

Syntax:

Write: SL=[0 or 1]

Examples:

```
SL=1   ;***Sets axis to closed-loop mode
```

SLSX

Description:

Read: Get StepNLoop status

Syntax:

Read: [variable]=SLSX

Conditional: IF SLSX =[variable]
ENDIF
IF SLSX =[value]
ENDIF

Examples:

```
IF SLSX != 0
    ECLEARX
ELSE
    ECLEARSX
ENDIF
```

SSPDX

Description:

Write: Set on-the-fly speed change for an individual axis.

Syntax:

Write: SSPDX=[value]
SSPDX=[variable]

Note: If s-curve is enabled for an axis, on-the-fly speed feature cannot be used for the corresponding axis.

Examples:
```
SCVX=0          ;***Disable s-curve acceleration
HSPD=1000       ;***X-axis high speed
LSPD=100        ;***Set low speed
ACC=100         ;***Set acceleration
JOGX+           ;***Jogs to positive direction
DELAY=1000      ;***Wait 1 second
SSPDX=3000      ;***Change speed on-the-fly to 3000 PPS
```

SSPDMX

Description:

Write: Set individual on-the-fly speed change mode

Syntax:

Write: SSPDMX=[0-9]
SSPDMX=[variable]

Examples:
```
SCVX=0          ;***Disable s-curve acceleration
HSPD=1000       ;***X-axis high speed
LSPD=100        ;***Set low speed
ACC=100         ;***Set acceleration
JOGX+           ;***Jogs to positive direction
DELAY=1000      ;***Wait 1 second
SSPDMX=1        ;***Set on-the-fly speed change mode to 1
ACC=20000       ;***Set acceleration to 20 seconds
SSPDX=190000    ;***Change speed on-the-fly to 190000 PPS
```

STOPX

Description:

Command: Stop all axes if in motion with deceleration.
Previous acceleration value is used for deceleration.

Syntax:

STOPX

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

STORE

Description:

Command: Store all values to flash

Syntax:

STORE

Examples:
```
V80=EX          ;***Put encoder value in V80
DELAY=1000      ;***Wait 1 second
STORE           ;***Store V80 to non-volatile flash
```

SYNCFGX

Description:

Write: Set sync output configuration

Syntax:

Write: SYNCFGX=[value]
SYNCFGX =[variable]

Examples:

```
SYNCFGX =1          ;*** Set sync output configuration to 1
SYNPOSX=3000        ;*** Set sync output position to 3000
SYNONX              ;*** Turn on sync output feature
V1=1                ;*** Wait until sync output is triggered
WHILE V1 != 2
        V1=SYNSTATX
ENDWHILE
SYNOFFX             ;*** Disable sync output feature
```

SYNOFFX

Description:

Write: Disable sync output feature

Syntax:

Write: SYNOFFX

Examples:

See SYNCFGX

SYNONX

Description:

Write: Enable sync output feature

Syntax:

Write: SYNONX

Examples:

See SYNCFGX

SYNPOSX

Description:

Write: Set sync output position.

Syntax:

Write: SYNPOSX=[value]
Write: SYNPOSX=[variable]

Examples:

See SYNCFGX

SYNSTATX

Description:

Read: Get status for sync output

Syntax:

Read: [variable] = SYNS

Examples:

See SYNCFGX

SUB

Description:

Indicates start of subroutine

Notes: Subroutine definitions should be written AFTER the END statement
Sub 31 is reserved for error handling

Syntax:

SUB [subroutine number]
[Subroutine Number] range is 0 to 31

Examples:

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

TOC

Description:

Write: Set deceleration value.
Value is in milliseconds.

Syntax:

Write: TOC = [value]

Examples:

```
TOC=300     ;***Sets the time-out watchdog to 300 milliseconds
```

V[0-99]

Description:

Assign to variable.

DMX-K-SA has 100 variables [V0-V99]

Syntax:

V[Variable Number] = [Argument]

V[Variable Number] = [Argument1][Operation][Argument2]

Special case for BIT NOT:

V[Variable Number] = ~[Argument]

[Argument] can be any of the following:

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

[Operation] can be any of the following

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
>>	Bit Shift Right
<<	Bit Shift Left
&	Bit AND
|	Bit OR
~	Bit NOT

Examples:

```
V1=12345          ;***Set Variable 1 to 123
V2=V1+1           ;***Set Variable 2 to V1 plus 1
V3=DI             ;***Set Variable 3 to digital input value
V4=DO             ;***Sets Variable 4 to digital output value
V5=~EO            ;***Sets Variable 5 to bit NOT of enable output value
```

WAITX

Description:

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

Syntax:

WAITX

Examples:

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

WHILE

Description:

Perform WHILE loop

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

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

X

Description:

Command: Perform X axis move to target location

Syntax:

X[value]
X[variable]

Examples:

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

ZHOMEX[+ or -]

Description:

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

Syntax:

ZHOMEX[+ or -]

Examples:

```
ZHOMEX+     ;***Z Homes axis in positive direction
WAITX
ZHOMEX-     ;***Z Homes axis in negative direction
WAITX
```

ZOMEX[+ or -]

Description:

Command: Perform Zoming (homing only using Z-index) using current high speed, low speed, and acceleration.

Syntax:

ZOMEX[+ or -]

Examples:

```
ZOMEX+      ;***Zomes axis in positive direction
WAITX
ZOMEX-      ;***Zomes axis in negative direction
WAITX
```

11. Example Standalone Programs

Standalone Example Program 1 – Single Thread

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

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

Standalone Example Program 2 – Single Thread

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

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

Standalone Example Program 3 – Single Thread

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

```
HSPD=20000          ;* Set the high speed to 20000 pulses/sec
LSPD=1000           ;* Set the low speed to 1000 pulses/sec
ACC=300             ;* Set the acceleration to 300 msec
EO=1                ;* Enable the motor power
V1=0                ;* Set variable 1 to value 0
WHILE V1<10         ;* Loop while variable 1 is less than 10
        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
```

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
```

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
```

Standalone Example Program 6 – Single Thread

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

```
HSPD=20000                ;* Set the high speed to 20000 pulses/sec
LSPD=1000                 ;* Set the low speed to 1000 pulses/sec
ACC=300                   ;* Set the acceleration to 300 msec
EO=1                      ;* Enable the motor power
WHILE 1=1                 ;* Forever loop
        IF DI1=1          ;* If digital input 1 is on
                X1000     ;* Move to 1000
        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
```

Standalone Example Program 7 – Multi Thread

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

```
PRG 0                           ;* Start of Program 0
HSPD=20000                      ;* Set high speed to 20000pps
LSPD=500                        ;* Set low speed to 500pps
ACC=500                         ;* Set acceleration to 500ms
WHILE 1=1                       ;* Forever loop
        X0                      ;* Move to position 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
```

Standalone Example Program 8 – Multi Thread

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

```
PRG 0                    ;* Start of Program 0
HSPD=1000                ;* Set high speed to 1000 pps
LSPD=500                 ;* Set low speed to 500pps
ACC=500                  ;* Set acceleration to 500ms
TOC=5000                 ;* Set time-out counter alarm to 5 seconds
EO=1                     ;* Enable motor
WHILE 1=1                ;* Forever loop
        X0               ;* Move to position 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
```

Appendix A: Speed Settings

HSPD value [PPS] †	Speed Window [SSPDM]	Min. LSPD value	Min. ACC [ms]	δ	Max ACC setting [ms]
1 - 16 K	0,1	10	2	500	((HSPD – LSPD) / δ) × 1000
16K - 30 K	2	10	1	1 K	
30K - 80 K	3	15	1	2 K	
80K - 160 K	4	25	1	4 K	
160K - 300 K	5	50	1	8 K	
300K - 800 K	6	100	1	18 K	
800K - 1.6 M	7	200	1	39 K	
1.6 M - 3.0 M	8	400	1	68 K	
3.0 M – 6.0 M	9	500	1	135 K	

Table A.0

†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 / Step-N-Loop Ratio

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.

$$\boxed{\textbf{Max ACC} = ((\textbf{HSPD} - \textbf{LSPD}) / \delta) \times \textbf{1000 [ms]}}$$

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)

Acceleration/Deceleration Range – Positional Move

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

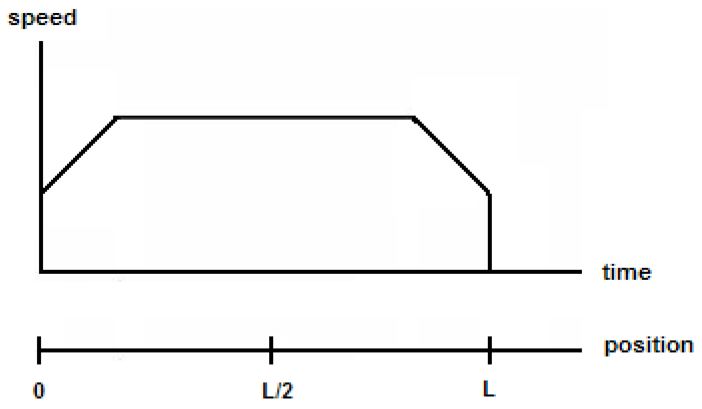


Figure A.1

1) <u>ACC vs. DEC 1:</u> 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) <u>ACC vs. DEC 2:</u> 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) <u>Triangle Profile:</u> 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