

ACE-SXC

Advanced Stand-Alone Controller USB 2.0 communication



COPYRIGHT © 2007 ARCUS, ALL RIGHTS RESERVED

First edition, Oct 2007

ARCUS TECHNOLOGY copyrights this document. You may not reproduce or translate into any language in any form and means any part of this publication without the written permission from ARCUS.

ARCUS makes no representations or warranties regarding the content of this document. We reserve the right to revise this document any time without notice and obligation.

Revision History:

- 1.01 – 1st release
- 2.01 – 2nd release
- 2.02 – 3rd release

Firmware Compatibility:

†V419BL

†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.....	6
Features	6
2. Electrical and Thermal Specifications	7
Power Requirement.....	7
Temperature Ratings †	7
Digital Inputs †	7
Digital Outputs.....	7
Alarm Input.....	7
3. Dimensions	8
4. Connections.....	9
DB9 Connector	10
2-Pin Connector (5.08mm)	10
10-Pin Connector (3.81mm)	10
ACE-SXC Interface Circuit	12
Power Input.....	13
Pulse, Direction and Enable Outputs	14
Alarm Input.....	14
Limits, Home and Digital Inputs	15
Digital Outputs.....	15
5. Getting Started	16
Typical Setup	16
Windows GUI.....	17
Main Control Screen	17
A. Status.....	18
B. Axis Control	19
C. DI Status/DO Status/Enable	20
D. Configuration	20
E. Terminal	22
F. Setup	23
G. Standalone Program File Management	24
H. Standalone Program Editor	24
I. Standalone Program Compile/Download/Upload/View	25
J. Program Control.....	25
K. Variable Status	26
6. Motion Control Overview.....	27
Motion Profile and Speed	27
Position Counter.....	27
Target Move.....	27
Homing	28
Home Input Only (High speed only)	28
Home Input Only (High speed and low speed).....	29
Limit Only.....	29
Jog Move.....	30
Stopping Motor	30

Limit Switch Function	30
Alarm Input Function.....	31
Configuration Button Function.....	31
Motor Status.....	31
Digital Inputs	32
Digital Outputs.....	32
Motor Power	33
Device Number	33
Standalone Programming.....	33
Communication Time-out Feature (Watchdog).....	34
Storing to Flash.....	35
7. Connecting to DMX and ACE Drivers	36
Connecting DMX-K-DRV-11/17	36
Connecting to DMX-K-DRV-23	36
Connecting to DMX-A2-DRV-17/23	37
Connecting to ACE-SDX.....	37
8. DMX and ACE Configuration.....	38
Configuration Method #1 – Using Windows PC.....	38
Configuration Method #2 – Using the Configuration Button.....	38
9. Communication – USB	40
USB Communication API Functions.....	40
USB Communication Issues	41
10. ASCII Language Specification	42
Error Codes	44
11. Standalone Language Specification.....	45
;	45
ABORTX	45
ABS.....	45
ACC	46
DELAY	46
DI	46
DI[1-8]	47
DO.....	47
DO[1-2].....	48
ECLEARX	48
ELSE.....	48
ELSEIF	49
END	50
ENDIF.....	50
ENDSUB.....	50
ENDWHILE	51
EO	51
GOSUB	51
HLHOMEX[+ or -]	52
HOMEX[+ or -]	52
HSPD	53

IF.....	53
INC.....	54
JOGX[+ or -].....	54
LHOMEX[+ or -].....	54
LSPD.....	55
MSTX.....	55
PX.....	56
STOPX.....	56
SR[0,1].....	57
STORE.....	57
SUB.....	57
TOC.....	58
V[1-50].....	58
WAITX.....	59
WHILE.....	59
X.....	60
12. Example Standalone Programs.....	61
Standalone Example Program 1 – Single Thread.....	61
Standalone Example Program 2 – Single Thread.....	61
Standalone Example Program 3 – Single Thread.....	61
Standalone Example Program 4 – Single Thread.....	62
Standalone Example Program 5 – Single Thread.....	62
Standalone Example Program 6 – Single Thread.....	63
Standalone Example Program 7 – Multi Thread.....	64
Standalone Example Program 8 – Multi Thread.....	65

1. Introduction

ACE-SXC is a stepper controller motion product.

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

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

Features

ACE-SXC

- USB 2.0 communication
- Stand-alone programmable
- Opto-isolated I/O
 - 3 x inputs
 - 2 x outputs
 - +Limit/-Limit/Home inputs
- 1 x alarm input (TTL)
- Open-collector outputs
 - Pulse, Direction, Enable
- Homing routines:
 - Home input only (high speed)
 - Home input only (high speed + low speed)
 - Limit only
- 400 KHz maximum pulse rate output
- 12-48VDC voltage input
- DMX-K-DRV, DMX-A2-DRV and ACE-SDX driver configuration

Contacting Support

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 +48 VDC
Recommended Current (Max):	200 mA

Temperature Ratings †

Operating Temperature:	0°C to +80°C
Storage Temperature:	-65°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 PNP outputs
Max voltage at emitter:	+24 VDC
Max source current at 24VDC	†45 mA

† A current limiting resistor is required

Alarm Input

Type:	TTL
Voltage:	+0 to +5VDC

3. Dimensions

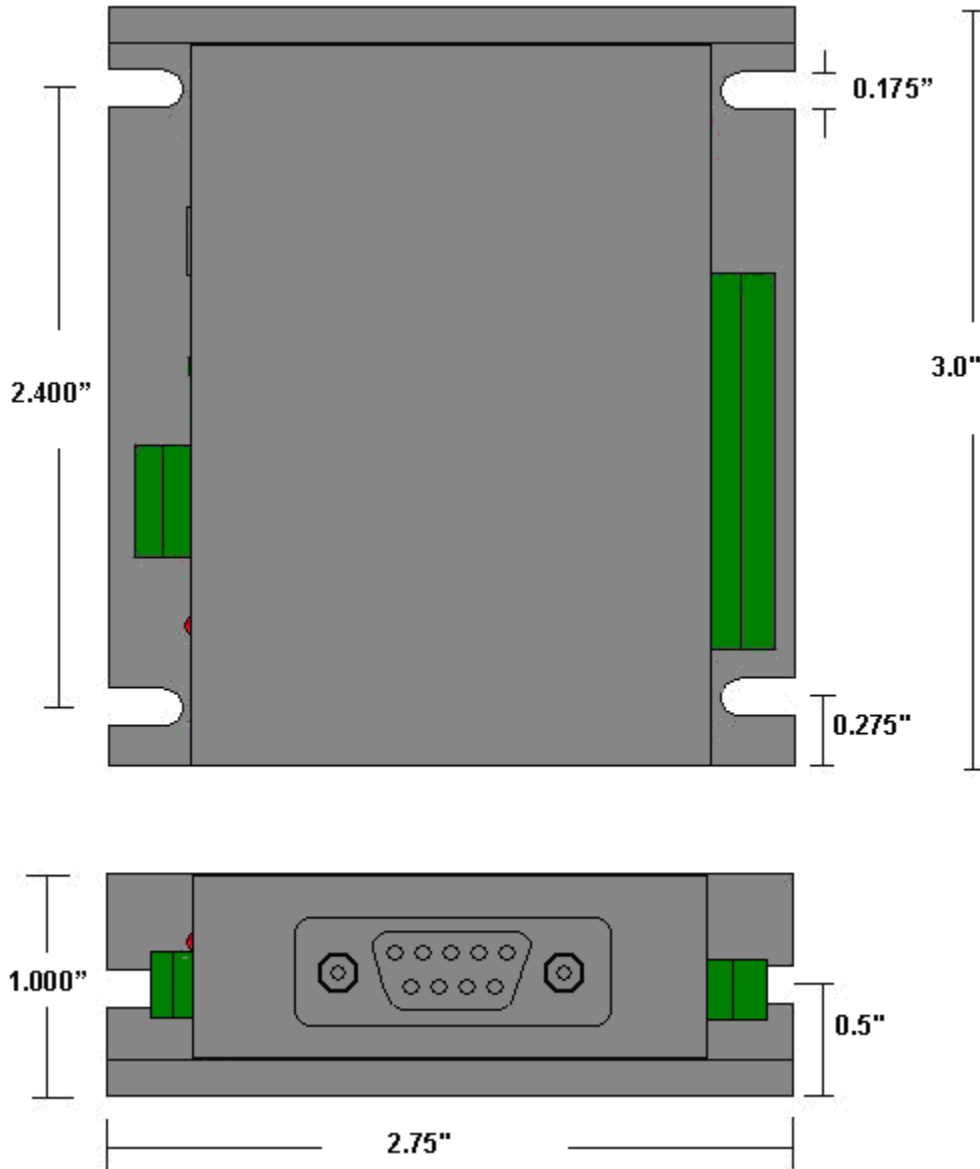


Figure 3.0

†All dimensions in inches

4. Connections

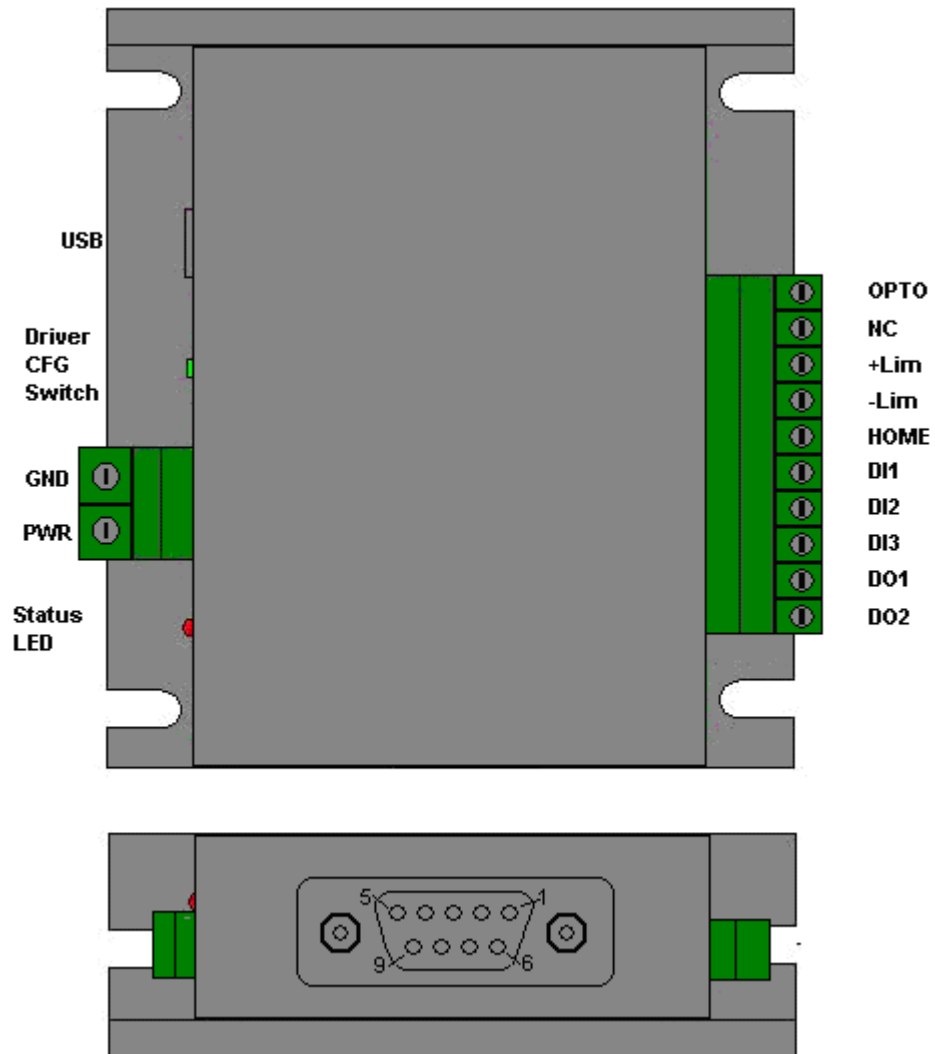


Figure 4.0

DB9 Connector

Pin #	In/Out	Name	Description
1	O	PWR_OUT	Shorted to pin 2 (PWR) of the 2-pin 5.08mm connector
2	O	PUL	Pulse output (open-collector)
3	O	ENA	Enable output (open-collector)
4	I	ALM	Alarm input (TTL)
5	O	5V+	+5VDC. Typically used for the opto-supply on the driver interface
6	O	GND_OUT	Shorted to pin 1 (GND) of the 2-pin 5.08mm connector
7	O	DIR	Direction output (open-collector)
8	RV	RV	Reserved. Do not connect
9	O	5V+	Shorted to pin 5

Table 4.1

2-Pin Connector (5.08mm)

Pin #	In/Out	Name	Description
1	I	GND	Ground
2	I	PWR	Power Input +12 to +48 VDC

Table 4.2

Mating Connector Description: 2 pin 0.2" (5.08mm) connector
 Mating Connector Manufacturer: On-Shore
 Mating Connector Manufacturer Part: †EDZ950/2

† Other 5.08mm compatible connectors can be used.

10-Pin Connector (3.81mm)

Pin #	In/Out	Name	Description
1	O	DO2	Digital Output 1
2	O	DO1	Digital Output 2
3	I	DI3	Digital Input 3
4	I	DI2	Digital Input 2
5	I	DI1	Digital Input 1
6	I	HOME	Home Input
7	I	-LIM	Minus Limit Input
8	I	+LIM	Plus Limit Input

9	RV	RV	Reserved. Do not connect
10	I	OPTO	Opto-supply input (+12 to +24VDC)

Table 4.3

Mating Connector Description: 10 pin 0.15" (3.81mm) connector
 Mating Connector Manufacturer: On-Shore
 Mating Connector Manufacturer Part: †EDZ1550/10

† Other 3.81 compatible connectors can be used.

ACE-SXC Interface Circuit

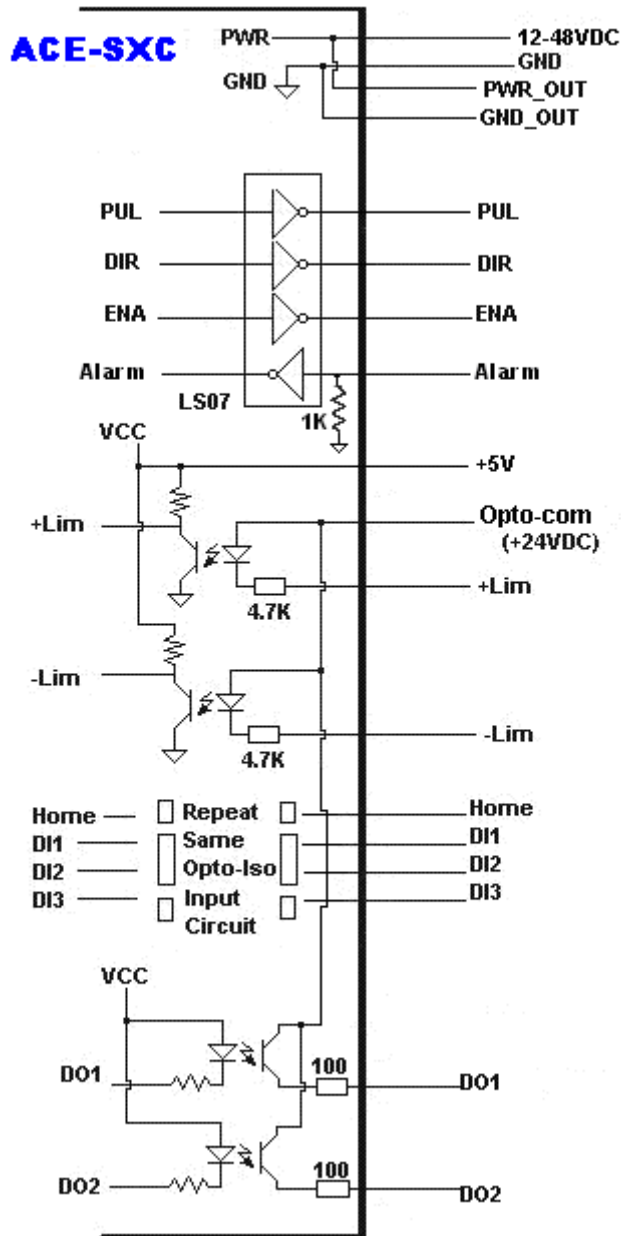


Figure 4.4

Power Input

Figure 4.5 shows that the power and ground signals that are supplied to ACE-SXC through the 2 pin connector are also available through the DB9 pin connector.

Regulated Supply Voltage Range: **+12 to 48 VDC**
 Recommended Current for power supply: **†200 mA**

† If driver is powered through the DB-9, additional current is required to power the driver.)

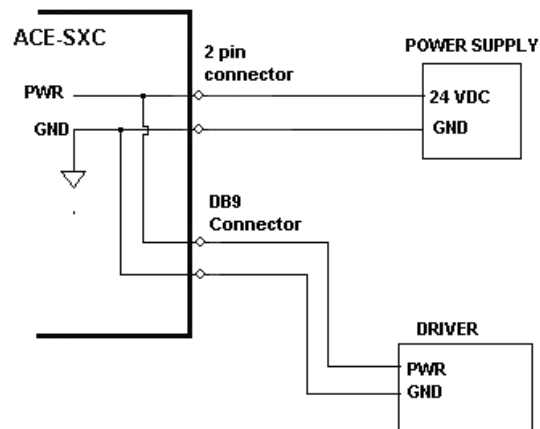


Figure 4.5

WARNING: If the driver is powered through the DB9 connector, make sure that the voltage of the power supply does not go over the maximum rated power supply voltage of the driver. For example, DMX-K-DRV maximum allowed voltage is +24VDC. If ACE-SXC is powered by +48VDC, powering the DMX-K-DRV through the DB9 connector will damage the driver.

Pulse, Direction and Enable Outputs

Enable Output is an open collector output using 74LS07.

Figure 4.6 shows an example of the pulse, direction and enable connections to a driver.

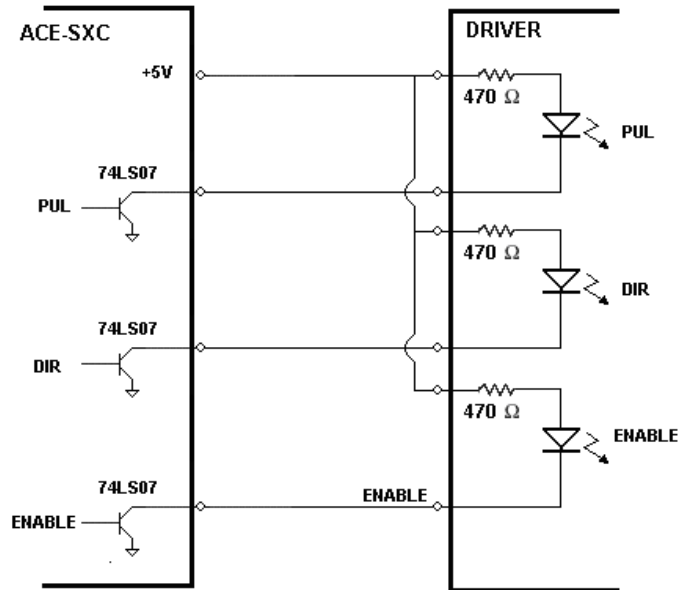


Figure 4.6

Alarm Input

Alarm input is a TTL compatible input using the 74LS07.

Figure 4.7 shows an example wiring of the alarm signal

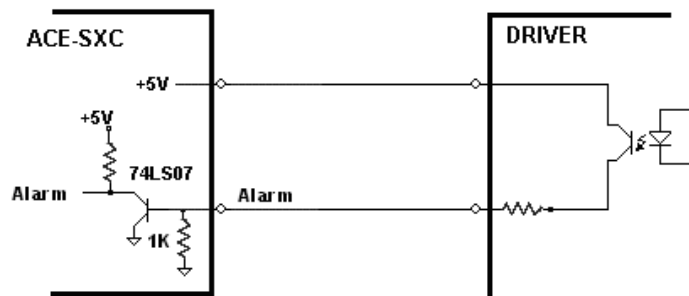


Figure 4.7

Limits, Home and Digital Inputs

Figure 4.8 shows an example wiring of the home, limit and digital inputs.

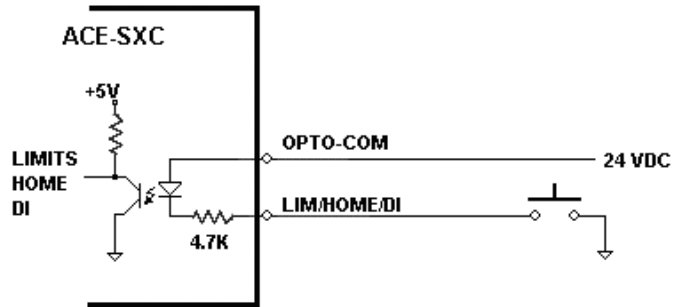


Figure 4.8

Digital inputs are active high.

Digital Outputs

Figure 4.9 shows an example wiring of the digital outputs.

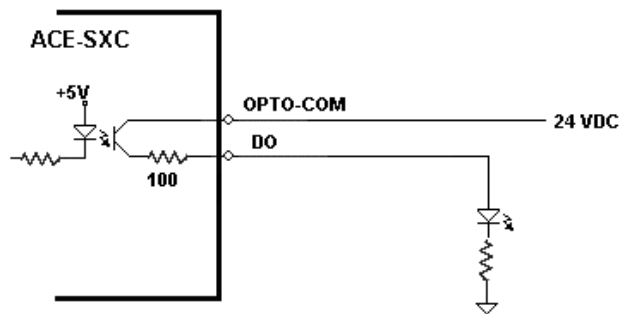


Figure 4.9

Digital outputs are active low.

5. Getting Started

Typical Setup

PC-Controlled

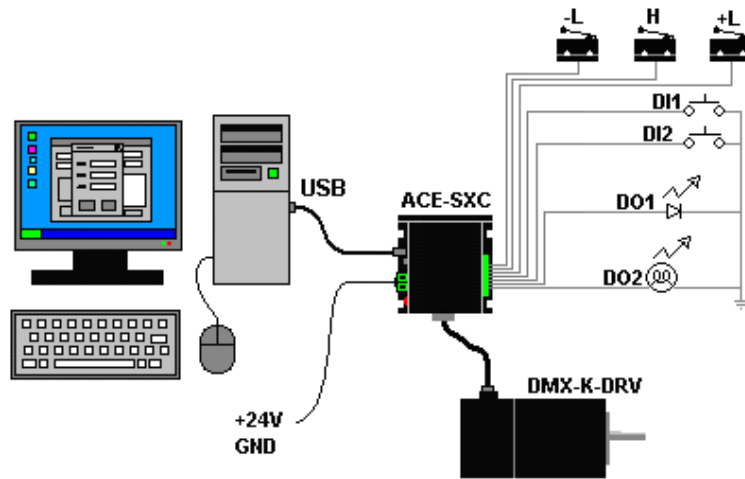


Figure 5.1

Stand-Alone Operation

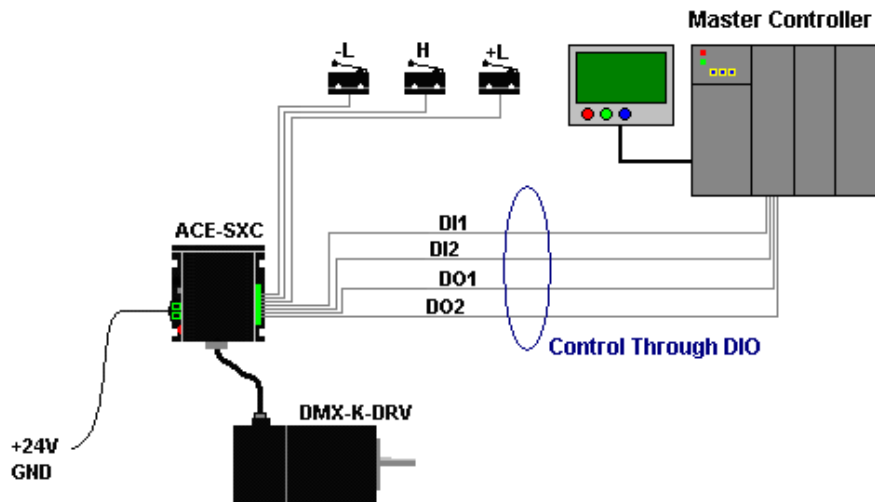


Figure 5.2

Windows GUI

ACE-SXC comes with Windows GUI program to test, program, compile, download, and debug the controller. The GUI program can also be used to configure driver settings of DMX-K-DRV, DMX-A2-DRV, and ACE-SDX.

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

Startup the ACE-SXC GUI program and you will see following screen:

Main Control Screen

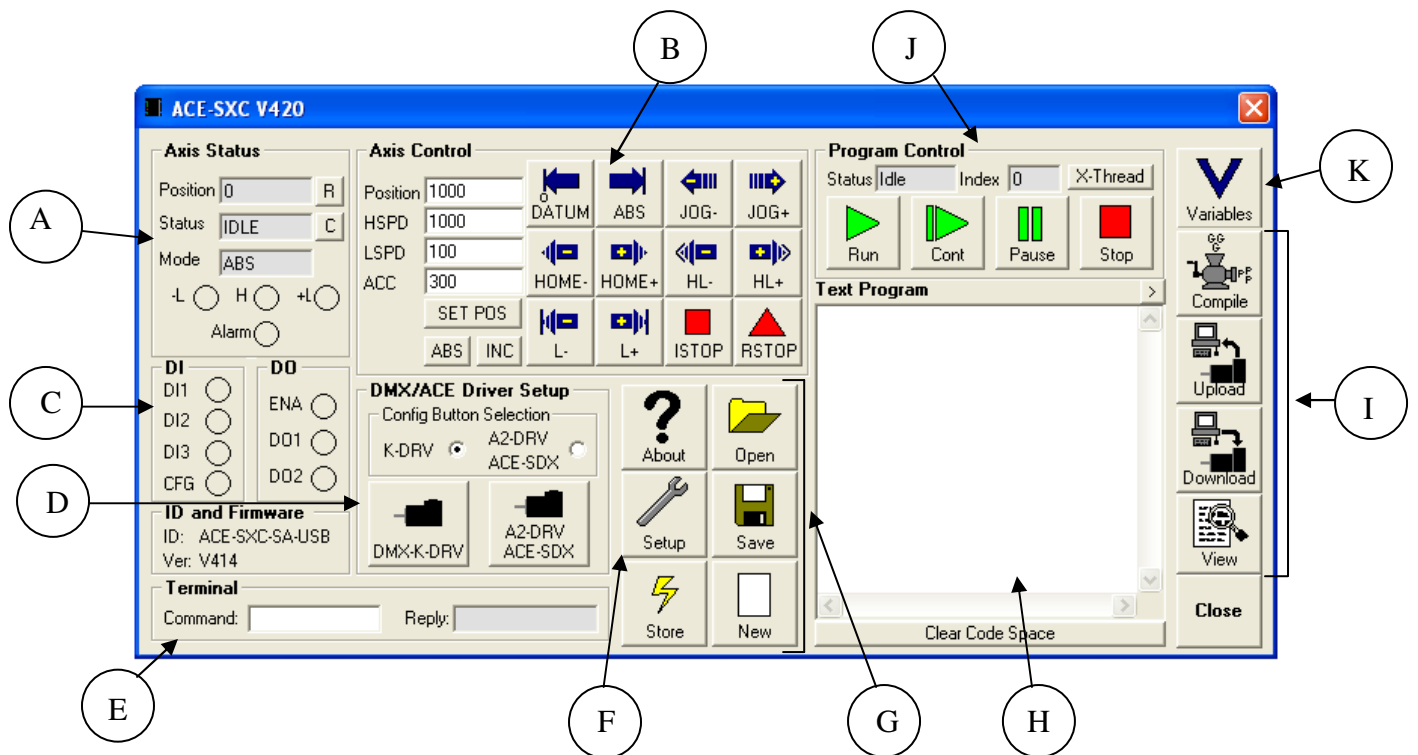


Figure 5.3

A. Status

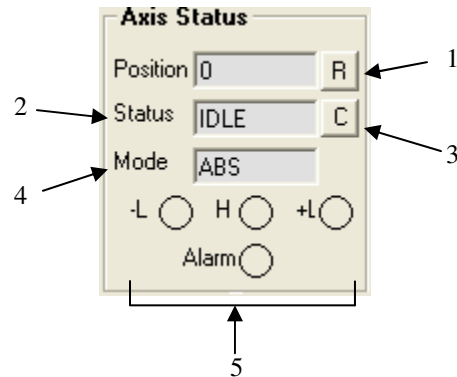


Figure 5.4

1. **Position** – Display of the current motor position value. Reset Position button is used to reset the position counter.
2. **Status** – Display of current motor status. Possible values are:
 ACCEL – acceleration in progress
 CONST – constant speed in progress
 DECEL – deceleration in progress
 -LIM ERROR – minus limit error occurred
 +LIM ERROR – plus limit error occurred
3. **Clear Error** – Clear Error button is used to clear any limit error status.
4. **Mode** – Display of the move mode of the controller. Possible values are:
 ABS – Absolute position movement
 INC – Incremental position movement
5. **Limit and Home and Alarm Input Status** – Display of limits, home and alarm input status.

B. Axis Control

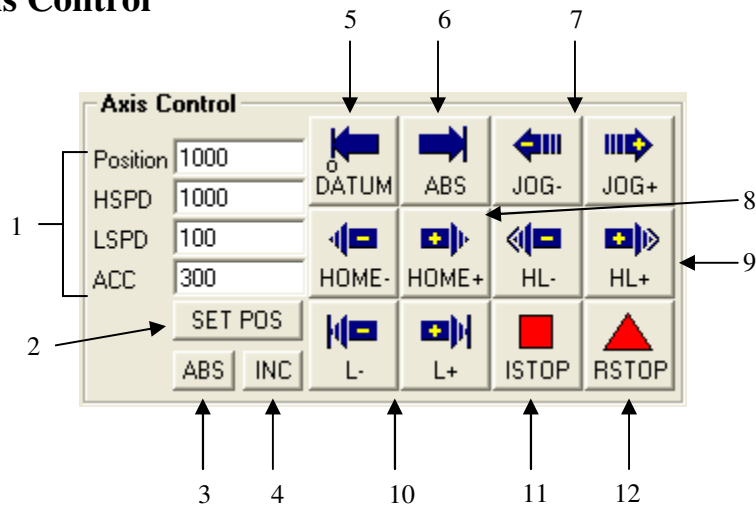


Figure 5.5

1. Target Position/Speed/Accel

Position – Set the target position. This position is the pulse position
HSPD/LSPD – Set the speed of the move. This value is in pulses/second
ACC – Set the acceleration/deceleration of the move. This value is in milliseconds

2. Set Position – Set position counter to the Target Position value.

3. ABS Mode – Set to absolute move mode. In ABS mode, all position moves go directly to the target position

4. INC Mode – Set to incremental move mode. In INC mode, all moves increment/decrement from the current position

5. DATUM – Absolute move to zero position. Maximum delta from current position to target is 262,143. If greater, then move will not perform

6. ABS – Absolute move to target position. Maximum delta from current position to target is 262,143. If greater, then the move will not perform

7. JOG+/- - Jog in plus or minus direction

8. HOME+/- - Homing in plus or minus direction

9. HL+/- - Low speed homing in the plus of minus direction

10. L+/- - Limit homing in the plus or minus direction

11. ISTOP – Immediate stop without deceleration

12. RSTOP – Stop with deceleration

C. DI Status/DO Status/Enable

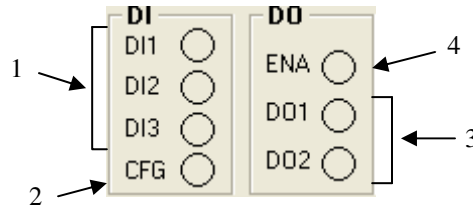


Figure 5.6

1. **Digital Input Status** – Display of the three digital input bits. If the digital input pin is grounded, the digital input is turned on.
2. **Configuration Button Input Status** – Display of the driver configuration button.
3. **Digital Output Status** – Display of the two digital output status. Digital output can be toggled by clicking on the circle.
4. **Enable Output Status** – Enable output status. Enable output can be toggled by clicking on the circle.

D. Configuration

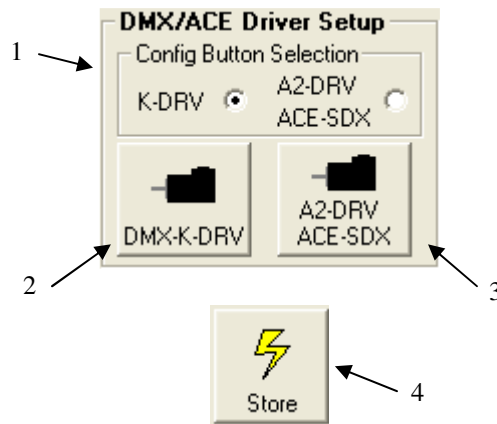


Figure 5.7

1. **Configuration Button Selection** – Configuration button is used to download the driver parameters without the use of the Windows PC. A2-DRV and ACE-SDX use the same driver parameter settings and are grouped together as one type. Select the driver type that will be used with the configuration button. Once selected, store it to flash memory.
2. **DMX-K-DRV Configuration using Windows GUI** – DMX-K-DRV can be configured from the GUI by selecting the DMX-K-DRV button.

When the DMX-K-DRV button is selected the DMX-K-DRV configuration dialog box is opened. From this dialog box, settings for

DMX-K-DRV can be uploaded or downloaded. Note that the Temperature (showing the current driver temperature as detected) and Version are read-only.

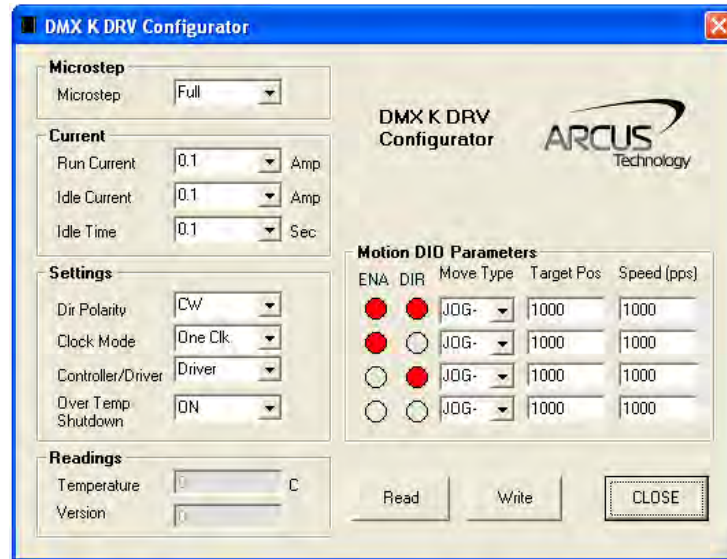


Figure 5.8

The driver settings can be stored on ACE-SXC so that the parameter values can be used by the configuration button. Click the **Close** button to download the parameters to the ACE-SXC. To store the downloaded parameters permanently to the ACE-SXC controller, make sure to store to flash before powering down the controller.

3. DMX-A2-DRV/ACE-SDX Configuration using Windows GUI –
 DMX-A2-DRV/ACE-SDX can be configured from the GUI by selecting the A2-DRV/ACE-SXC button.

When the DMX-A2-DRV/ACE-SDX button is selected, the DMX-A2-DRV/ACE-SDX configuration dialog box is opened. From this dialog box, settings for DMX-A2-DRV/ACE-SDX can be uploaded or downloaded. Note that the Temperature (showing the current driver temperature as detected) and Version are read-only.

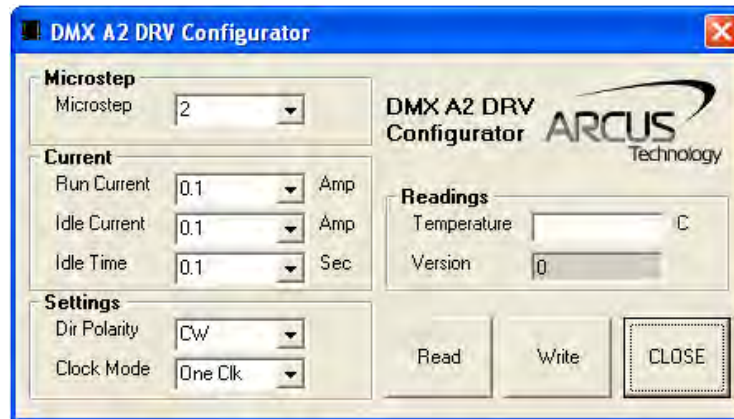


Figure 5.9

The driver settings can be stored on ACE-SXC so that the parameter values can be used by the configuration button. Click the **Close** button to download the parameters to the ACE-SXC. To store the downloaded parameters permanently to the ACE-SXC controller, make sure to store to flash before powering down the controller.

4. **Store To Flash** – Parameters on the ACE can be permanently stored to the flash memory. See Table 6.5 for variables that are stored to flash.

E. Terminal

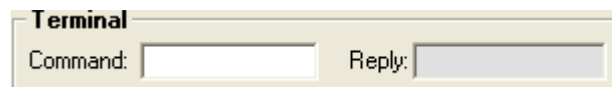


Figure 5.10

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

F. Setup

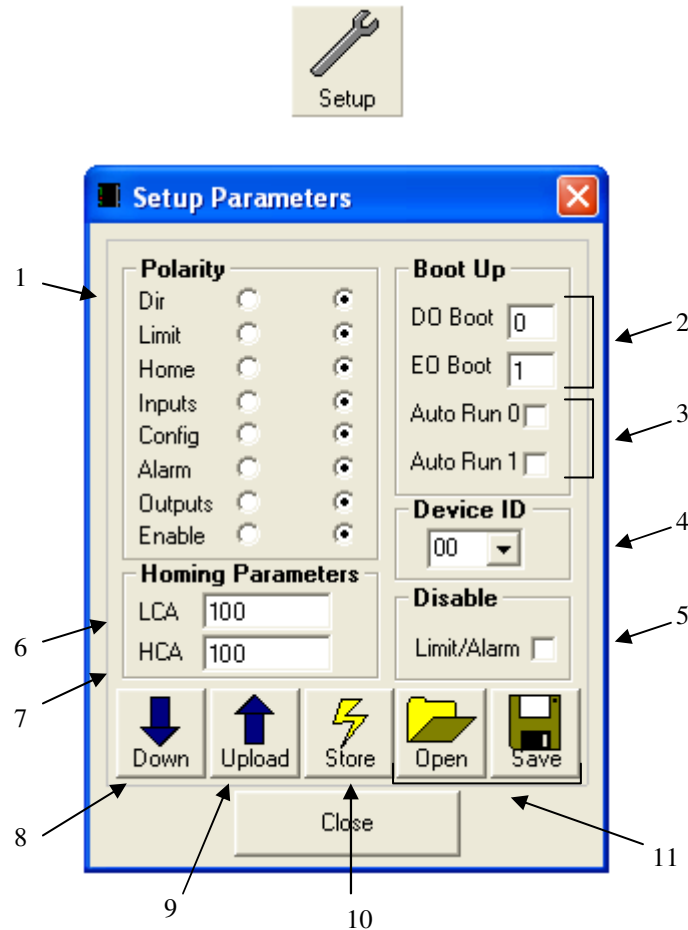


Figure 5.11

1. **Polarity** - Change the polarity of the inputs and outputs of the ACE-SXC
2. **DOBOOT/EOBOOT** - Change the boot up configuration of the enable and digital outputs
3. **Auto Run** - Run standalone programs on boot up.
4. **Device Name** - Set the Device Name [SXC01-SXC99]
5. **Limit/Alarm** - Disable the limit and alarm inputs for general purpose use
6. **LCA** - Set the Limit Correction Amount used for the limit homing routine
Refer to the Homing Overview in Section 6 for further details
7. **HCA** - Set the Home Correction Amount used for the homing routine
Refer to the Homing Overview in Section 6 for further details
8. **Down** - Download the current settings
9. **Upload** - Upload the settings currently on the ACE-SXC
10. **Store** - Store the settings to flash memory
11. **Open/Save** - Parameters can be saved to a file and read from a file

G. Standalone Program File Management

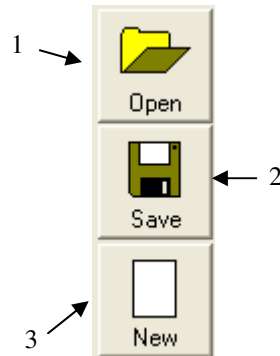


Figure 5.12

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

H. Standalone Program Editor

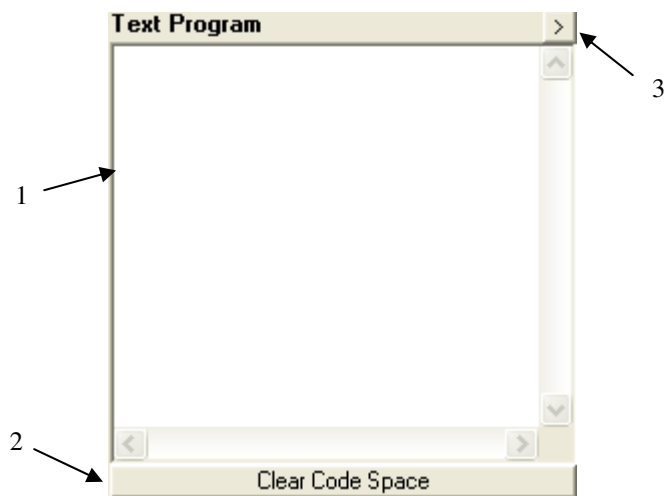


Figure 5.13

1. Write the standalone program in the Program Editor
2. **Clear Code Space** – Use this button to remove the current standalone program that is currently stored on the ACE-SXC.
3. Use this button to open a larger and easier to manage program editor.

I. Standalone Program Compile/Download/Upload/View

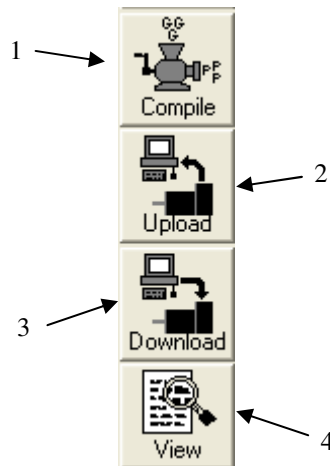


Figure 5.14

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

J. Program Control

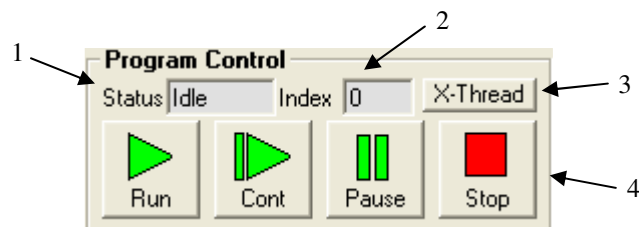


Figure 5.15

1. **Program Status** – display of program status. Possible statuses are
 - Idle – program is not running
 - Running – program is running
 - Paused – program is paused
 - Errored – program is in error status
2. **Program Index** – display of low level program index. This is the index of the low level program index.
3. **X-Thread** – Open the Program Control for standalone multi-thread operation.

4. Program Control – Program can be RUN, STOP, PAUSED, and CONTINUED.

K. Variable Status

View the status of variables 1-50. Note that this window is read-only.

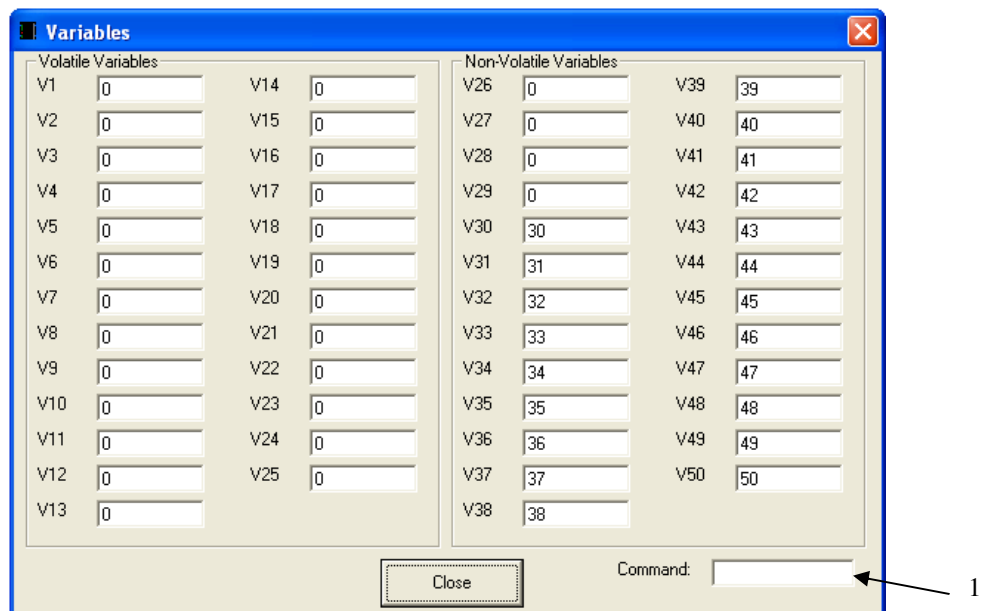


Figure 5.16

- 1. Command line** – To write to variable, use `V[1-50] = [value]` syntax.

6. Motion Control Overview

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

Motion Profile and Speed

ACE-SXC incorporates trapezoidal velocity profile as shown below.

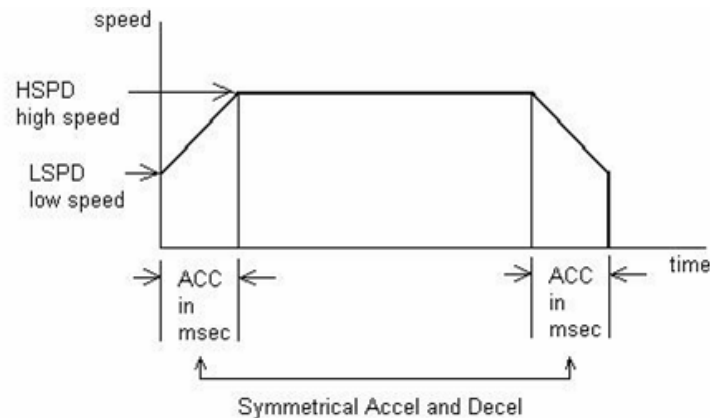


Figure 6.1

High speed and low speed are in pps (pulses/second). Use the **HSPD** and **LSPD** commands to set/get the high speed and low speed settings. Pulse output rate supported is from 100 to 400K pps.

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

Position Counter

ACE-SXC has a 32 bit signed position counter. Range of the position counter is from -2,147,483,648 to 2,147,483,647. Get the current position by using the **PX** command.

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

Target Move

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

Maximum allowable difference to target position from current position is 262,143. Maximum difference between current position and the target position has to be less than

or equal to 262,143. For example, if the current position counter is 1000, target position allowed will be between -261,143 (1,000-262,143) and 263,143 (1,000+262,143).

ACE-SXC can operate in either incremental or absolute move modes. Use **X** command to make moves. Use the **INC** and **ABS** commands to set the move mode. Use the **MODE** command to read the current move mode.

Homing

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

Home Input Only (High speed only)

Use the **H+/H-** command. Figure 6.2 shows the homing routine.

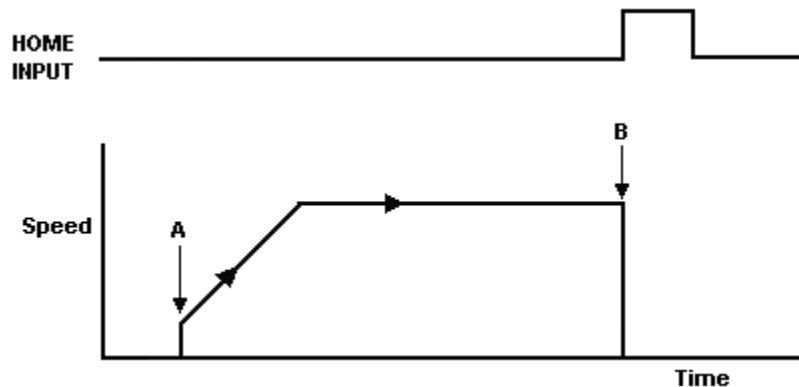


Figure 6.2

- 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 stops immediately. If the home switch is triggered in the middle of the acceleration, the motor stops immediately.

Home Input Only (High speed and low speed)

Use the **HL+/HL-** command. Figure 6.3 shows the homing routine.

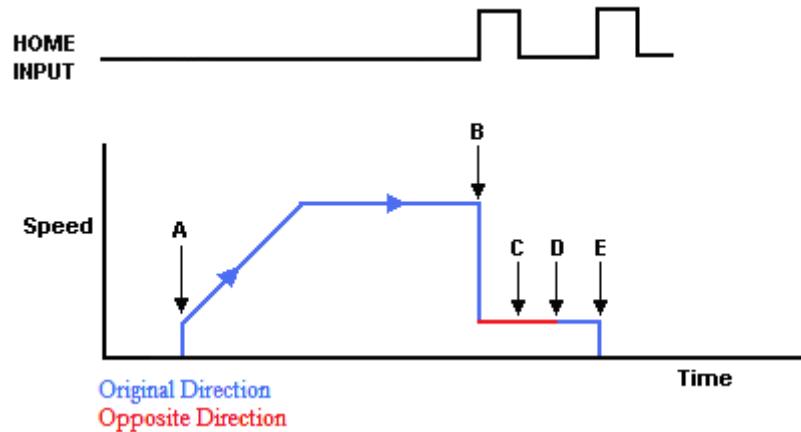


Figure 6.3

- A. Starts the motor from low speed and accelerates to high speed.
- B. As soon as the home input is triggered, the position counter is reset to zero and the motor immediately slows down to low speed.
- C. Motor is backing out from the home switch. It will continue past the home switch by the amount defined by the home correction amount (**HCA**). The direction of this movement will be the opposite of the original home command.
- D. 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.
- E. The home input is triggered again, the position counter is reset to zero and the motor immediately stops.

Limit Only

Use the **L+/L-** command. Figure 6.4 shows the homing routine.

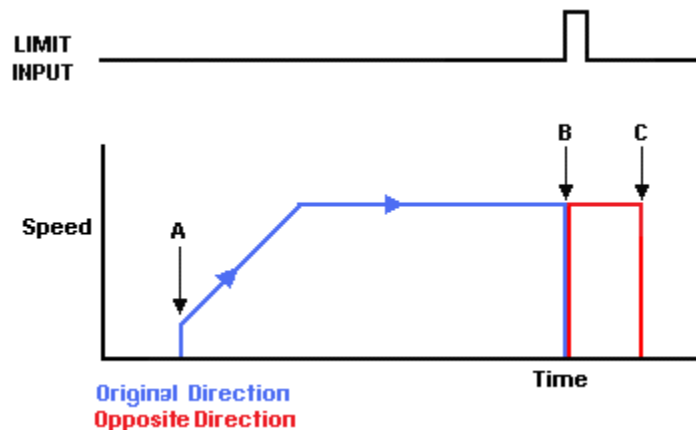


Figure 6.4

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

Notes:

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

If the home input is not used, it can also be used as a general purpose input. Digital input assignment for the home input switch is **DI6**.

If the limit switches are disabled using the disable limit feature (**DL** command), the limit inputs can also be used as general purpose inputs.

Digital inputs are active high.

Jog Move

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

Stopping Motor

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

Limit Switch Function

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

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

The limit switch function can be disabled by using the **DL** command. By disabling the limit switch function, the limits switches can be used as general purpose inputs.

Note: The **DL** command controls both the limit switch function as well as the alarm input function.

Limit inputs are active high.

Alarm Input Function

All motion is aborted when the alarm input is triggered – regardless of the direction of the motor. Once the alarm error is set, the status must be cleared (using the **CLR** command) in order to move again. The alarm switch function can be disabled using the **DL** command. By disabling the alarm switch function, the limits switches can be used as a general purpose input

Note: The **DL** command controls both the limit switch function as well as the alarm input function.

Alarm input is active high.

Configuration Button Function

Configuration button is used to configure the DMX-K-DRV, DMX-A2-DRV, and ACE-SDX. Configuration button can also be used as general purpose digital input in the standalone program.

The configuration button is active high.

Motor Status

Motor status can be read anytime using the **MST** command. The following are 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 negative direction motion. This error must be cleared before issuing any subsequent move commands (CLR command).
7	Plus limit error. This bit is latched when plus limit is hit during positive direction motion. This error must be cleared before issuing any subsequent move commands (CLR command).
8	Alarm error. This bit is latched when the alarm input is triggered while motion is in progress. This error must be cleared before issuing any subsequent move commands (CLR command).
10	Communication timeout counter alarm

Table 6.0

Digital Inputs

ACE-SXC module comes with 3 digital inputs. If the limit/alarm function is disabled, up to 8 inputs can be used as general purpose inputs

Read digital input status using the **DI** command. See description below:

Digital input values can also be referenced one bit at a time by the **DI[1-8]** 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	Configuration Button	DI4
4	Minus Limit	DI5
5	Home	DI6
6	Plus Limit	DI7
7	Alarm	DI8

Table 6.1

Digital inputs are active high.

Digital Outputs

ACE-SXC module comes with 2 digital outputs.

Read and set digital output status using the **DO** command. See description below:

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

Table 6.2

Digital output values can also be referenced one bit at a time by the **DO[1-2]** commands. Note that the indexes are 1-based for the bit references (i.e. DO1 refers to bit 0, not bit 1)

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

Digital outputs are active low.

Motor Power

Using the **EO** command, the motor power can be enabled or disabled. If the enable function is not used, the enable output can be used as a general purpose output. The enable output does not affect the move command.

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

Polarity

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

Bit	Description
0	Direction
1	Limit Input
2	Home Input
3	Digital Input
4	Configuration Button
5	Alarm Input
6	Digital Output
7	Enable Output

Table 6.3

Device Number

ACE-SXC module provides the user with the ability to set the device number of a specific device. In order to make these changes, first store the desired number using the **DN** command. Please note that this value must be within the range [SXC01,SXC99].

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, the device name is set to: **SXC00**

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 ACE-SXC 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 is first executed, and then standalone execution continues.

Calling subroutines over communication: Once a subroutine is written into the flash, they can be called via USB communication using the **GS** command. The subroutines are referenced by their subroutine number [0-31]. If a subroutine number is not defined, the ACE-SXC 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 6.4

Communication Time-out Feature (Watchdog)

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

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

Storing to Flash

The following items are stored to flash:

ASCII Command	Description
DN	Device Name
POL	Polarity Settings
DOBOOT	DO configuration at boot-up
EOBOOT	EO configuration at boot-up
LCA	Limit Correction Amount
HCA	Home Correction Amount
DL	Disable Limit setting
SLOAD	Standalone program run on boot-up parameter
BDT	Stepper driver configuration type
A1-A7, K1-K9	Stepper driver parameters
TOC	Time-out counter reset value
V26-V50	Note that on boot-up, V1-V25 are reset to value 0

Table 6.5

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

7. Connecting to DMX and ACE Drivers

Connecting DMX-K-DRV-11/17

Cable Accessory: CBL-DB_9M-DF11_10F-L1-G24-V1

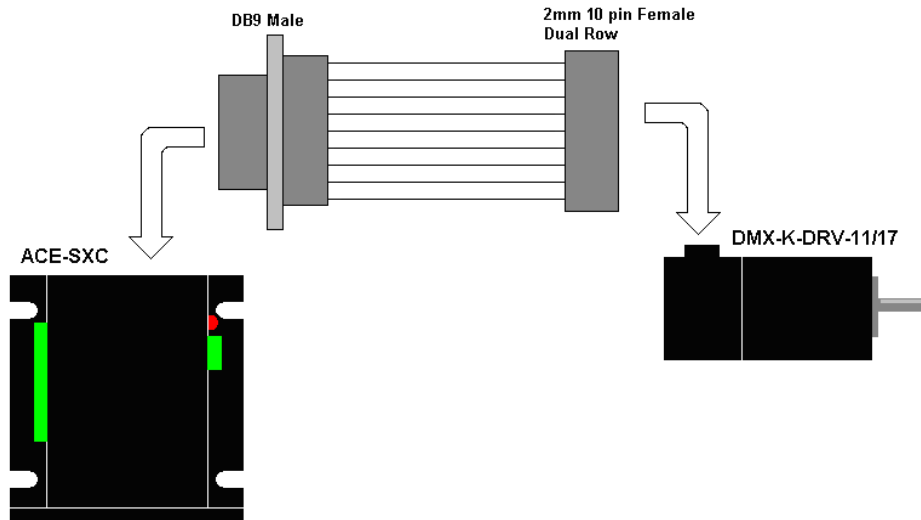


Figure 7.1

Connecting to DMX-K-DRV-23

Cable Accessory: CBL-DB_9M-DF3_10-L1-G24-V1

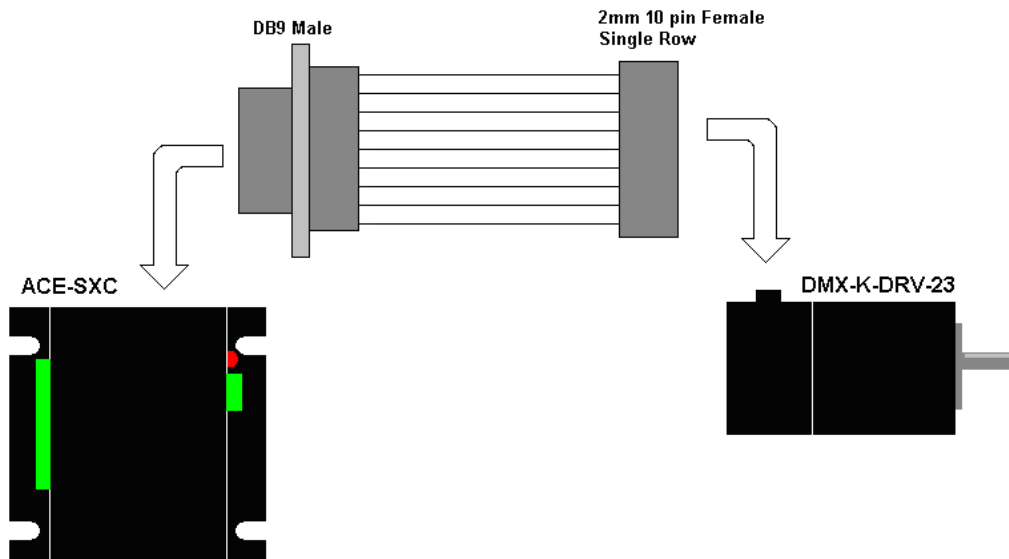


Figure 7.2

Connecting to DMX-A2-DRV-17/23

Cable Accessory: CBL-DB_9M-DB_9F-L1-G24-V1

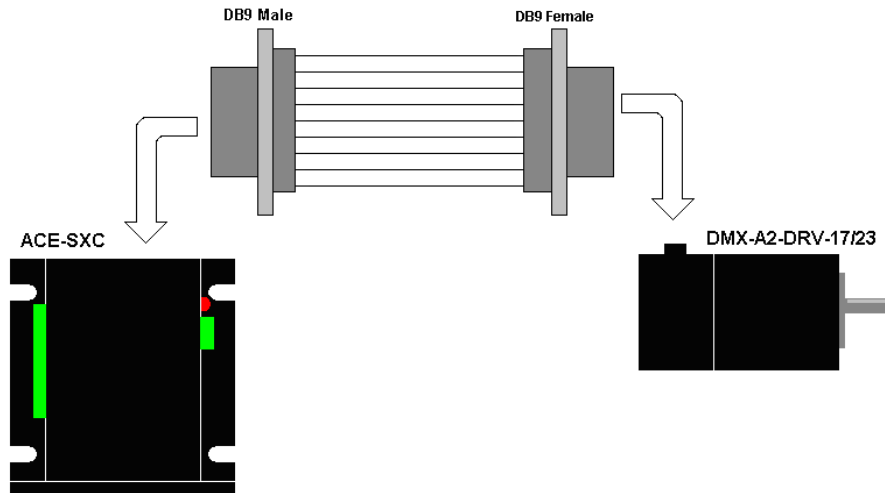


Figure 7.3

Connecting to ACE-SDX

Cable Accessory: CBL-DB_9M-I_8F-L1-G24-V1

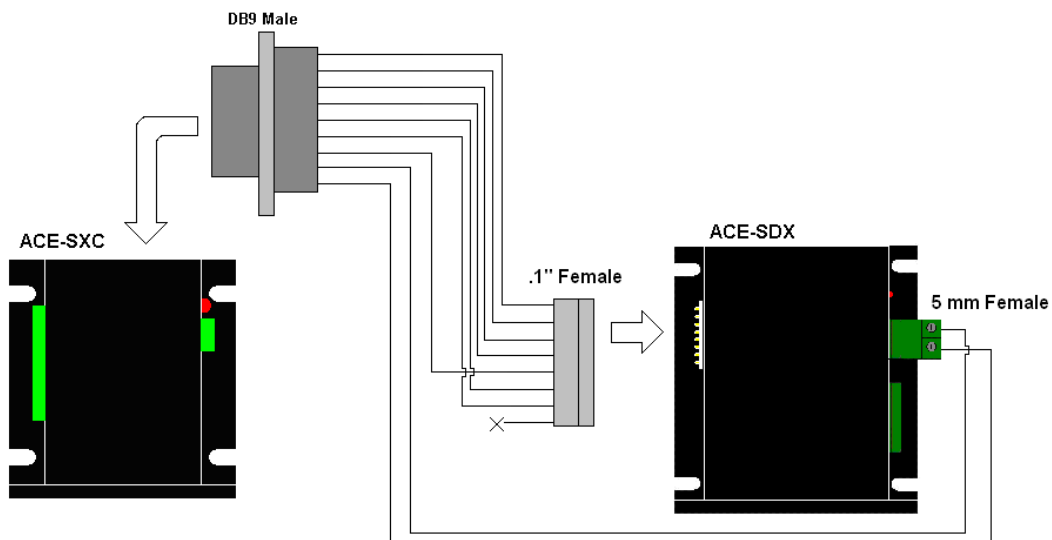


Figure 7.4

8. DMX and ACE Configuration

ACE-SXC can be used to configure the driver settings for the following products.

DMX-K-DRV-11/17/23
 DMX-A2-DRV-17/23
 ACE-SDX

ACE-SXC uses patent pending Dynamic Configuration method of reading and writing of the driver setting through control lines: PULSE/DIR/ENABLE/ALARM.

There are two ways to configure the DMX-K/DMX-A2/ACE-SDX using ACE-SXC.

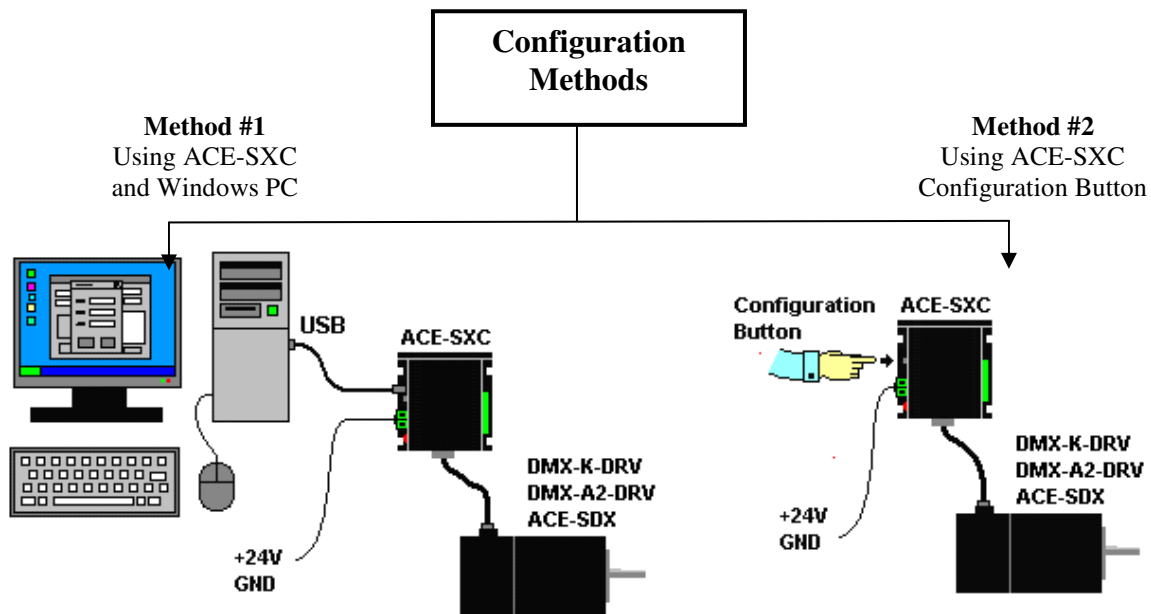


Figure 8.1

Configuration Method #1 – Using Windows PC

Method #1 uses the Windows PC and the ACE-SXC GUI program to upload and download the driver parameters. For detailed description, refer to the **Getting Started** section.

Configuration Method #2 – Using the Configuration Button

Method #2 uses the configuration button on the ACE-SXC controller to download the driver parameters.

On the ACE-SXC controller, the driver type needs to be stored to flash so that when the button configuration is used, the correct driver configuration is performed. There are two types types for button configuration: 1) K-DRV and 2) A2-DRV/ACE-SDX.

Once the correct driver type is selected and the driver parameter values are stored to flash memory, driver parameters can be downloaded from ACE-SXC to DMX-K-DRV without the use of Windows PC using the configuration button on the ACE-SXC. To configure the driver through the configuration button, follow the steps below.

- 1) Power the ACE-SXC controller using 24VDC power supply.
- 2) Connect the control cable between ACE-SXC and DMX-K-DRV/A2-DRV/ACE-SDX. All the control signals (Pulse/Dir/Enable/Alarm) must be connected to work properly.
- 3) Press and hold down the configuration button for 3 seconds. The LED on ACE-SXC controller will start blinking quickly indicating that the configuration is ready to start.
- 4) While the LED is blinking quickly, release the button and press the button again to start the configuration of the connected driver. While the configuration is being processed, the LED is turned off. Configuration takes about 3 seconds. If the button is not pressed again within 3 seconds during the quick blinking state, the LED will stop blinking and configuration will be aborted.
- 5) If the configuration is done properly, the LED will blink quickly for 3 seconds. If the configuration is not done properly, LED will blink slowly for 3 seconds.

9. Communication – USB

ACE-SXC USB communication is USB 2.0 compliant.

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

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

USB Communication API Functions

For USB communication, following DLL API functions are provided.

BOOL fnPerformaxComGetNumDevices(OUT LPDWORD lpNumDevices);

- This function is used to get total number of all types of Performax and Performax USB modules connected to the PC.

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

- This function is used to get the Performax or Performax product string. This function is used to find out Performax USB module product string and its associated index number. Index number starts from 0.

**BOOL fnPerformaxComOpen(IN DWORD dwDeviceNum,
OUT HANDLE* pHandle);**

- This function is used to open communication with the Performax USB module and to get communication handle. dwDeviceNum starts from 0.

BOOL fnPerformaxComClose(IN HANDLE pHandle);

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

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

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

BOOL fnPerformaxComSendRecv(IN HANDLE pHandle,

```

IN LPVOID wBuffer,
IN DWORD dwNumBytesToWrite,
IN DWORD dwNumBytesToRead,
OUT LPVOID rBuffer);

```

- This function is used to send command and get reply. Number of bytes to read and write must be 64 characters.

BOOL *fnPerformaxComFlush*(IN HANDLE pHandle)

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

USB Communication Issues

A common problem that users may have with USB communication is that after sending a command from the PC to the device, the response is not received by the PC until another command is sent. In this case, the data buffers between the PC and the USB device are out of sync. Below are some suggestions to help alleviate this issue.

- 1) **Buffer Flushing:** If USB communication begins from an unstable state (i.e. your application has closed unexpectedly), it is recommended to flush the USB buffers of the PC and the USB device right after the communication handle is opened. See the following function prototype below:

```

BOOL fnPerformaxComFlush(IN HANDLE pHandle)

```

Note: *fnPerformaxComFlush* is only available in the most recent PerformaxCom.dll which is not registered by the standard USB driver installer. A sample of how to use this function along with this newest DLL is available for download on the website

- 2) **USB Cable:** Another source of USB communication issues may come from the USB cable. Confirm that the USB cable being used has a noise suppression choke. See photo below:



Figure 9.1

10. ASCII Language Specification

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

ACE-SXC language is case sensitive. All command should be in capital letters. Invalid command is returned “?”. Always check for proper reply when command is sent.

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. (10-1000)	ms
ACC=[Value]	Sets acceleration value in milliseconds. (10-1000) Example: ACC=300	OK
BDT	Get driver configuration type	1 – DMX-K-DRV 2 – DMX-A2-DRV / ACE-SDX
BDT=[0,1]	Set driver configuration type	OK
CLR	Clears limit error as well as StepNLoop error	OK
DI	Return status of digital inputs	Refer to Table 6.1
DI[1-8]	Return individual status of inputs	Refer to Table 6.1
DO	Return status of digital outputs	Refer to Table 6.2
DO=[Value]	Set digital output 2 bit number. Digital output is writable only if DIO is disabled.	OK
DO[1-2]	Get status if individual output	Refer to Table 6.2
DO[1-2]=[Value]	Set individual output. Refer to Table 6.2	OK
DL	Get disable limit/alarm function 0 – limit/alarm function is enabled 1 – limit/alarm function is disabled	[0,1]
DL=[Value]	Set disable limit/alarm function 0 – limit/alarm function is enabled 1 – limit/alarm function is disabled	OK
DN	Get device number	[SXC00-SXC99]
DN=[value]	Set device number	OK
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
GS[0-31]	Call a subroutine that has been previously stored to flash memory	OK
H+	Homes the motor in the positive direction	OK
H-	Homes the motor in the negative direction	OK
HCA	Returns the home correction amount	[0-262,143]

HCA=[Value]	Sets the home correction amount. This value cannot be greater than 262,143.	OK
HL+	Homes the motor at low speed in the positive direction	OK
HL-	Homes the motor at low speed in the negative direction	OK
HSPD=[Value]	Sets High Speed.	OK
ID	Returns product ID	ACE-SXC-SA-USB
INC	Set move mode to incremental	OK
J+	Jogs the motor in the positive direction	OK
J-	Jogs the motor in the negative direction	OK
L+	Limit homes the motor in the positive direction	OK
L-	Limit homes the motor in the negative direction	OK
LCA	Returns the limit correction amount	[0-262,143]
LCA=[Value]	Sets the limit correction amount. This value cannot be greater than 262,143.	OK
LSPD	Returns Low Speed Setting	PPS
LSPD=[Value]	Sets Low Speed	OK
MODE	Get move mode status	0 – Absolute move mode 1 – Incremental move mode
MST	Returns motor status	Refer to Table 6.0
POL	Returns current polarity	Refer to Table 6.3
POL=[value]	Sets polarity. Refer to Table 6.3	OK
PX	Returns current position value	32-bit number
PX=[value]	Sets the current position value	OK
SASTAT	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]	OK
SLOAD	Returns RunOnBoot parameter	0-1
SLOAD=[Value]	Bit 0 – Standalone program 0 0 – Do NOT run standalone program on boot up 1 – Run standalone program on boot up Bit 1 – Standalone program 1	OK
SR[0,1]=[Value]	Control standalone program: 0 – Stop standalone program 1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program	OK
SPC	Get program counter for standalone program	[0-1274]
STORE	Store settings to flash	OK
TOC	Get communication time out parameter	ms
TOC=[Value]	Set communication time-out parameter	OK
V[1-50]	Read variables 1-50	32-bit number

V[1-50]=[value]	Set variables 1-50	OK
VER	Get firmware version	VXXX
X[value]	Moves the motor to absolute position value using the HSPD, LSPD, and ACC values. Maximum allowed incremental move amount is 262143. For example, if current position is 100000, target move must be between 362143 and -162143	OK

Table 10.1

Error Codes

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

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

Table 10.2

11. Standalone Language Specification

;

Description:

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

Syntax:

; [Comment Text]

Examples:

```

;***This is a comment
JOGX+           ;***Jogs axis to positive direction
DELAY=1000      ;***Wait 1 second
ABORT           ;***Stop immediately all axes including X axis

```

ABORTX

Description:

Motion: Immediately stop motion without deceleration.

Syntax:

ABORTX

Examples:

```

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

```

ABS

Description:

Command: Changes all move commands to absolute mode.

Syntax:

ABS

Examples:

```

ABS             ;***Change to absolute mode
PX=0           ;***Change position to 0
X1000          ;***Move X axis to position 1000
WAITX
X3000          ;***Move X axis to position 3000
WAITX

```

ACC

Description:

Read: Get acceleration value

Write: Set acceleration value.

Value is in milliseconds.

Syntax:

Read: [variable] = ACC

Write: ACC = [value]

ACC = [variable]

Examples:

ACC=300 ;***Sets the acceleration to 300 milliseconds

V3=500 ;***Sets the variable 3 to 500

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

DELAY

Description:

Set a delay (1 ms units)

Syntax:

Delay=[Number] (1 ms units)

Examples:

JOGX+ ;***Jogs axis to positive direction

DELAY=10000 ;***Wait 10 second

ABORTX ;***Stop axis

DI

Description:

Read: Gets the digital input value. ACE-SXC has 8 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[1-8]

Description:

Read: Gets the digital input value. ACE-SXC has 8 digital inputs.

Digital inputs are active high

Syntax:

Read: [variable] = DI[1-8]

Conditional: IF DI[1-8]=[variable]
ENDIF

IF DI[1-8]=[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

ACE-SXC has 2 digital outputs.

Digital outputs are active low.

Syntax:

Read: [variable] = DO

Write: DO = [value]

DO = [variable]

Conditional: IF DO=[variable]
ENDIF

IF DO=[value]
ENDIF

Examples:

```
DO=3      ;***Turn on both bits
```

DO[1-2]

Description:

Read: Gets the individual digital output value

Write: Sets the individual digital output value

ACE-SXC has 2 digital outputs.

Digital outputs are active low.

Syntax:

Read: [variable] = DO[1-2]

Write: DO[1-2] = [0 or 1]

DO[1-2] = [variable]

Conditional: IF DO[1-2]=[variable]

ENDIF

IF DO[1-2]=[0 or 1]

ENDIF

Examples:

DO1=1 ;***Turn DO1 on

DO2=1 ;***Turn DO2 on

ECLEARX

Description:

Write: Clears motor error status.

Syntax:

Write: ECLEARX

Examples:

ECLEARX ;***Clears motor error

ELSE

Description:

Perform ELSE condition check as a part of IF statement

Syntax:

ELSE

Examples:

IF V1=1
X1000 ;***If V1 is 1, then move to 1000
WAITX

ELSE
X-1000 ;***If V1 is not 1, then move to -1000
WAITX

ENDIF

ELSEIF

Description:

Perform ELSEIF condition check as a part of the IF statement

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

```

IF V1=1
    X1000
    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
END
```

ENDIF

Description:

Indicates end of IF operation

Syntax:

ENDIF

Examples:

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

ENDSUB

Description:

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

Syntax:

ENDSUB

Examples:

```
GOSUB 1
END

SUB 1
    X0
    WAITX
    X1000
    WAITX
ENDSUB
```

ENDWHILE

Description:

Indicate end of WHILE loop

Syntax:

ENDWHILE

Examples:

```

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

EO

Description:

Read: Gets the enable output value

Write: Sets the enable output value

Syntax:

Read: [variable] = EO

Write: EO = [value]

EO = [variable]

Conditional: IF EO=[variable]
ENDIF

IF EO=[value]
ENDIF

Examples:

```

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

GOSUB

Description:

Perform go to subroutine operation

Subroutine range is from 0 to 31.

Note: Subroutine definitions should be written AFTER the END statement.

Subroutine 31 is reserved for error handling.

Syntax:

GOSUB [subroutine number]

[Subroutine Number] range is 0 to 31
 Examples:
 GOSUB 0
 END

```

SUB 0
  X0
  WAITX
  X1000
  WAITX
ENDSUB
  
```

HLHOMEX[+ or -]

Description:

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

Syntax:

HLHOMEX[+ or -]

Examples:

```

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

HOMEX[+ or -]

Description:

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

Syntax:

HOMEX[+ or -]

Examples:

```

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

HSPD

Description:

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

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

Range is from 1 to 400,000.

Syntax:

Read: [variable] = HSPD

Write: HSPD = [value]

HSPD = [variable]

Examples:

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

V1=2500 ;***Sets the variable 1 to 2,500

HSPD=V1 ;***Sets the high speed to variable 1 value of 2500

IF

Description:

Perform IF condition check

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

```
IF V1=1
    X1000
    WAITX
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.

Range is from 1 to 400,000.

Syntax:

Read: [variable]=LSPD

Write: LSPD=[long value]

LSPD=[variable]

Examples:

LSPD=1000 ;***Sets the start low speed to 1,000 pulses/sec

V1=500 ;***Sets the variable 1 to 500

LSPD=V1 ;***Sets the start low speed to variable 1 value of 500

MSTX

Description:

Read: Get motor status

Syntax:

Read: [variable]=MSTX

Conditional: IF MSTX=[variable]

ENDIF

IF MSTX=[value]

ENDIF

Examples:

IF MSTX=0

DO=3

ELSE

DO=0

ENDIF

PX

Description:

Read: Gets the current pulse position

Write: Sets the current pulse position

Syntax:

Read: Variable = PX

Write: PX = [value]

PX = [variable]

Conditional: IF PX=[variable]
ENDIF

IF PX=[value]
ENDIF

Examples:

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

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
WAITX
```

SR[0,1]

Description:

Write: Set the standalone control for the specified standalone program

Syntax:

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

Examples:

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

STORE

Description:

Command: Store all values to flash

Syntax:

STORE

Examples:

```
V40=PX           ;***Put position value in V40
DELAY=1000       ;***Wait 1 second
STORE            ;***Store V40 to non-volatile flas
```

SUB

Description:

Indicates start of subroutine

Syntax:

SUB [subroutine number]

[Subroutine Number] range is 0 to 31

Examples:

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

TOC

Description:

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

Syntax:

TOC=[long value]

Examples:

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

V[1-50]

Description:

Assign to variable.

ACE-SXC has 100 variables [V1-V50]

Syntax:

V[Variable Number] = [Argument]

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

Special case for BIT NOT:

V[Variable Number] = ~[Argument]

[Argument] can be any of the following:

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

[Operation] can be any of the following

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

Examples:

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

WAITX

Description:

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

Syntax:

WAITX

Examples:

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

WHILE

Description:

Perform WHILE loop

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

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

X

Description:

Command: Perform X axis move to target location

Syntax:

X[value]

X[variable]

Examples:

ABS ;*** Absolute move mode

X10000 ;*** Move to position 10000

WAITX

V10 = 1200 ;*** Set variable 10 value to 1200

XV10 ;*** Move axis to variable 10 value

WAITX

12. Example Standalone Programs

Standalone Example Program 1 – Single Thread

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

```

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

```

Contact Information

Arcus Technology, Inc.

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

www.arcustechnology.com

The information in this document is believed to be accurate at the time of publication but is subject to change without notice.