ACE-SXC

Advanced Standalone Controller
USB 2.0 Communication

First edition, Oct 2007

Revision History:

1.01 – 1st Release
2.01 – 2nd Release
2.02 – 3rd Release
2.04 – 4th Release
2.05 – 5th Release

Firmware Compatibility:

†V420BL

†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

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 standalone program to the device and have it run independent of a host.

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

1.1. Features

- USB 2.0 communication
- Standalone 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

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

2. Electrical and Thermal Specifications

Parameter	Min	Max	Units
Main Power Input[1]	+12	+48	V
	-	200	mA
Opto-supply Power Input	+12	+24	V
Digital Input Forward Diode Current	-	45	mA
Digital Output Collector Voltage	-	+24	V
Digital Output Source Current	-	45	mA
Alarm Input	+0	+5	V
Operating Temperature[2]	0	+80	°C
Storage Temperature[2]	-65	+150	°C

Table 2.0

[1]The supply current should match the driver current setting.
[2]Based on component ratings.

3. Dimensions

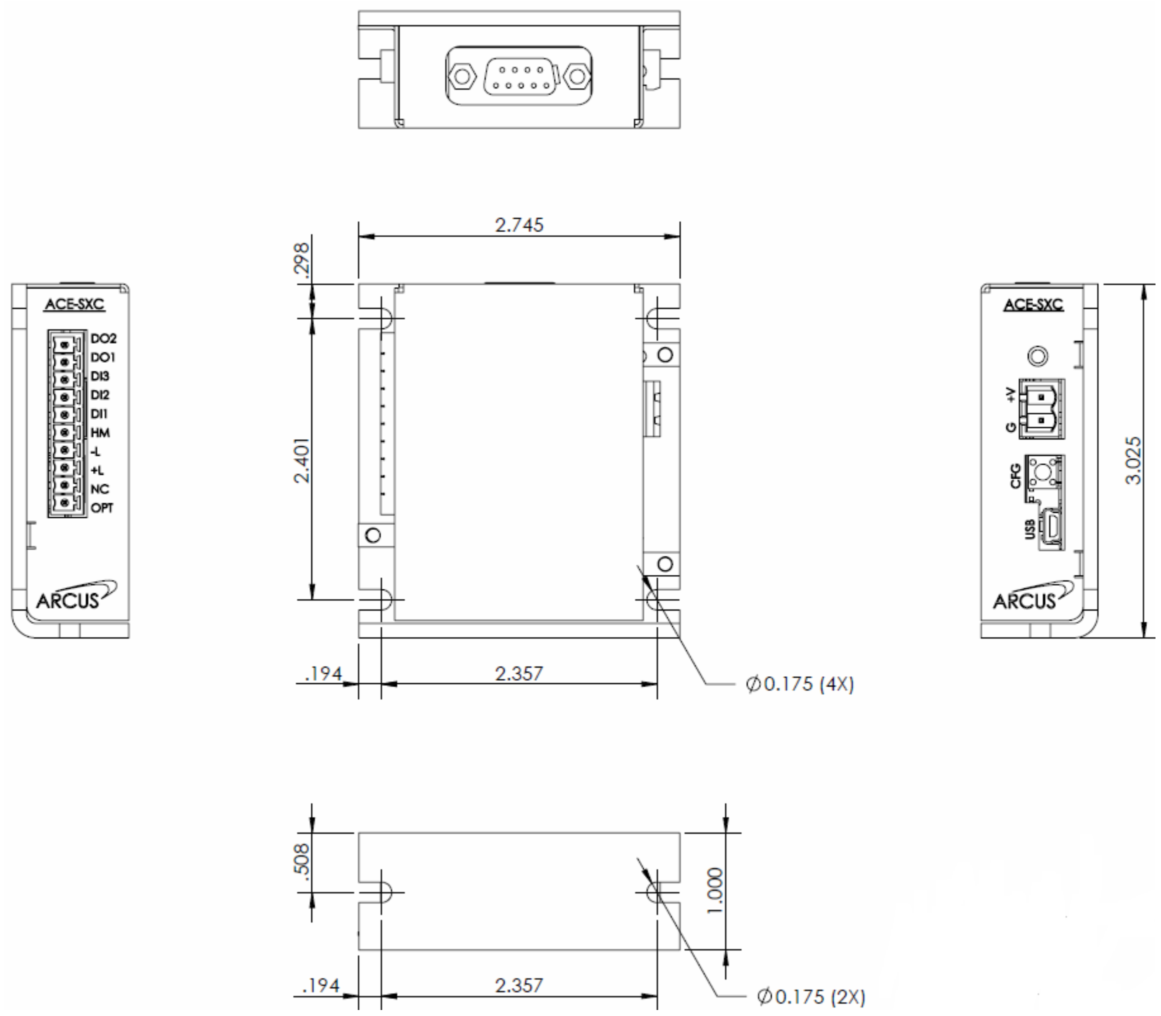


Figure 3.0

†All dimensions in inches

4. Connections

In order for ACE-SXC to operate, it must be supplied with +12VDC to +24VDC. Power pins as well as communication port pin outs are shown below.

4.1. 2-Pin Power Connector (5.08mm)

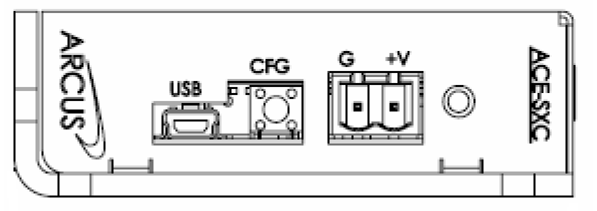


Figure 4.0

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

Table 4.0

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.

4.2. DB9 Connector

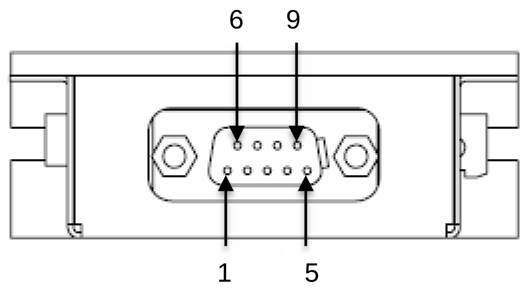


Figure 4.1

Pin #	Name	In/Out	Description
1	PWR_OUT	O	Shorted to pin 2 (PWR) of the 2-pin 5.08mm connector
2	PUL	O	Pulse output (open-collector)
3	ENA	O	Enable output (open-collector)
4	ALM	I	Alarm input (TTL)
5	5V+	O	+5VDC. Typically used for the opto-supply on the driver interface
6	GND_OUT	O	Shorted to pin 1 (GND) of the 2-pin 5.08mm connector
7	DIR	O	Direction output (open-collector)

8	RV	RV	Reserved. Do not connect
9	5V+	O	Shorted to pin 5

Table 4.1

4.3. 10-Pin DIO Connector (3.81mm)

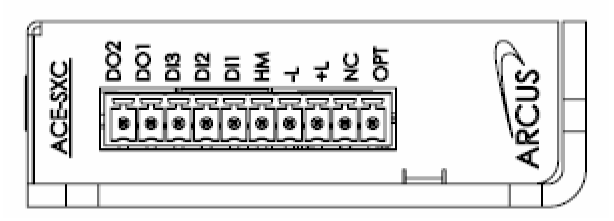


Figure 4.2

Pin #	Name	In/Out	Description
1	DO2	O	Digital Output 2
2	DO1	O	Digital Output 1
3	DI3	I	Digital Input 3
4	DI2	I	Digital Input 2
5	DI1	I	Digital Input 1
6	HM	I	Home Input
7	-L	I	Minus Limit Input
8	+L	I	Plus Limit Input
9	NC	NC	Not Connected
10	OPTO	I	Opto-supply input (+12 to +24VDC)

Table 4.2

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

4.4. ACE-SXC Interface Circuit

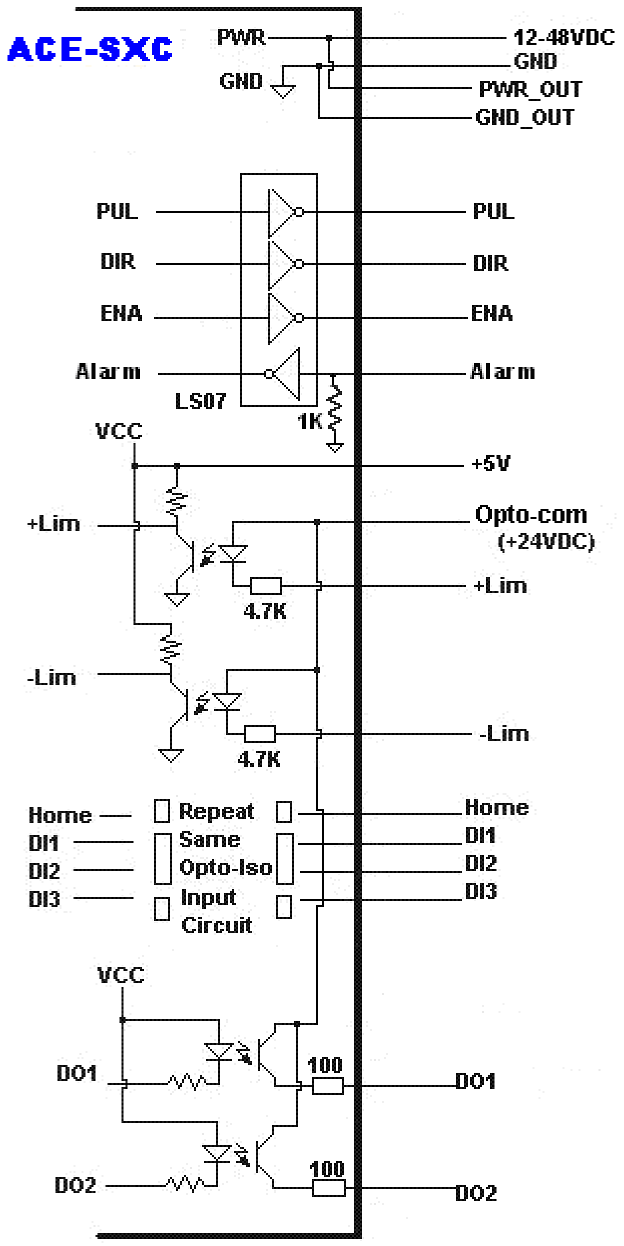


Figure 4.3

4.5. Power Input

Figure 4.4 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.)

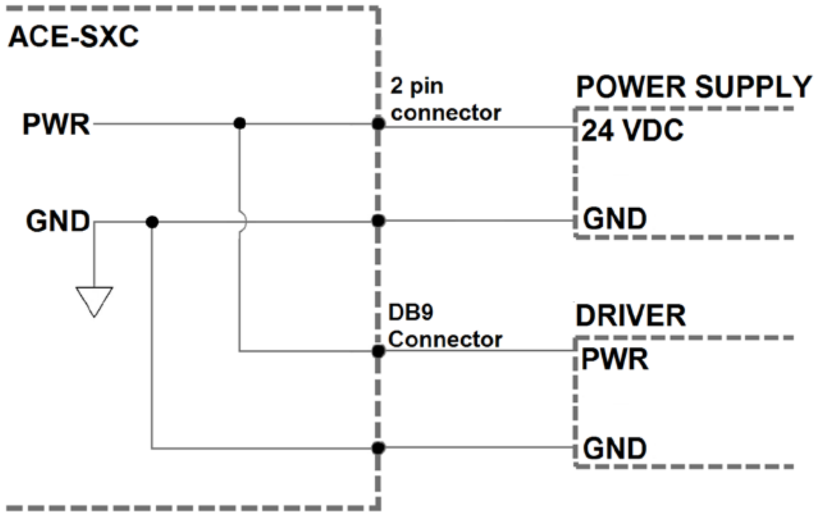


Figure 4.4

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, if a driver's maximum allowed voltage is +24VDC and the ACE-SXC is powered by +48VDC, powering the driver through the DB9 connector will damage the driver.

4.6. Pulse, Direction and Enable Outputs

The Pulse, Direction, and Enable outputs are all open collector outputs. Figure 4.5 shows the detailed schematic for these outputs. Each output is capable of sinking up to 45mA of current.

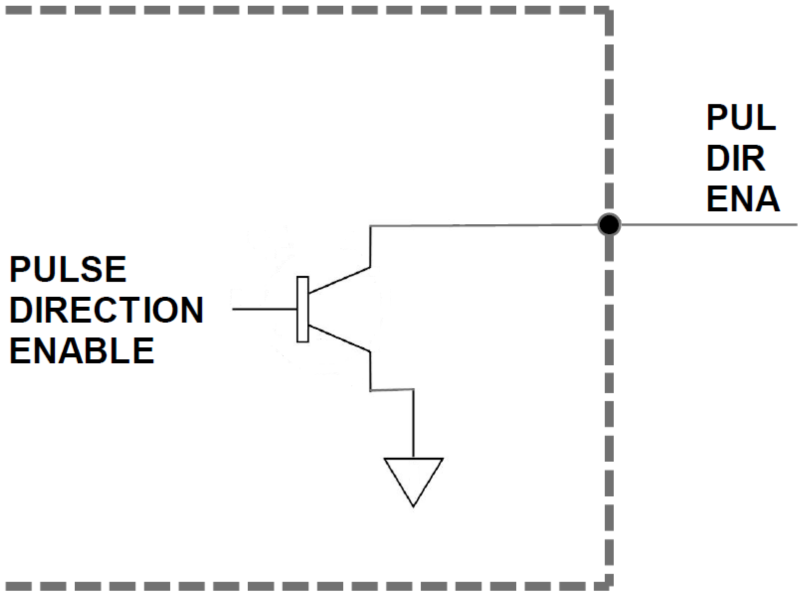


Figure 4.5

Figure 4.6 shows an example wiring diagram between the pulse, direction, and enable outputs on the ACE-SXC and the corresponding input on a typical stepper driver.

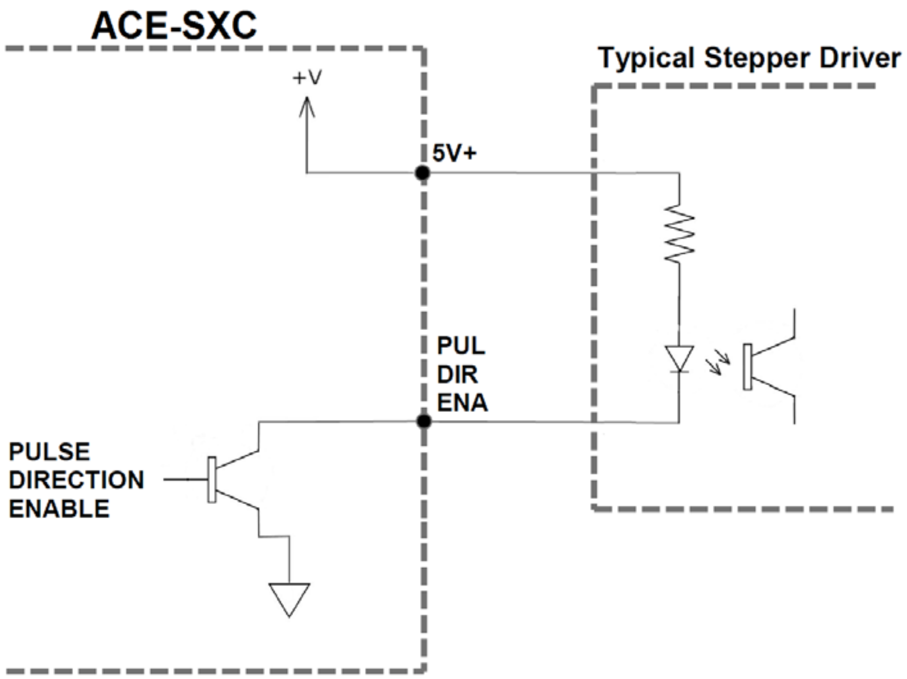


Figure 4.6

4.7. Alarm Input

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

Figure 4.7 shows an example wiring of the alarm signal

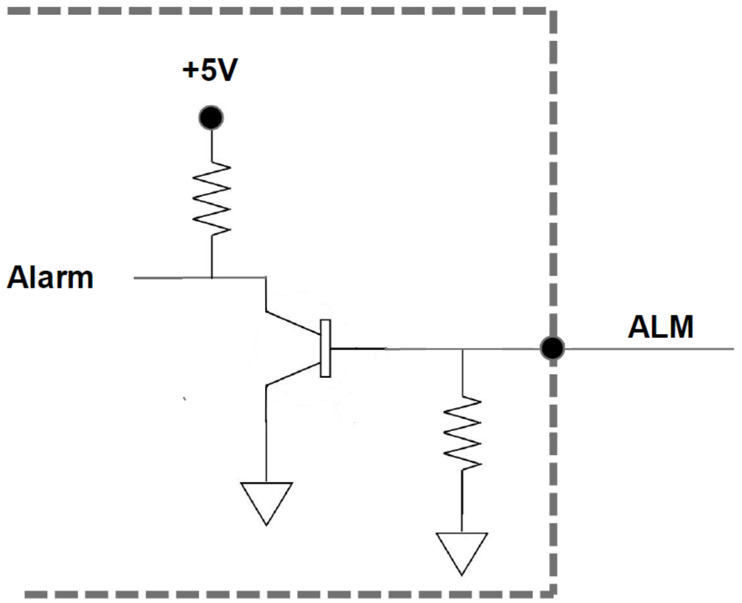


Figure 4.7

4.8. Digital Inputs, Limits, and Home

Figure 4.8 shows the detailed schematic of the opto-isolated general purpose digital inputs, limits, and home. All opto-isolated inputs are NPN type.

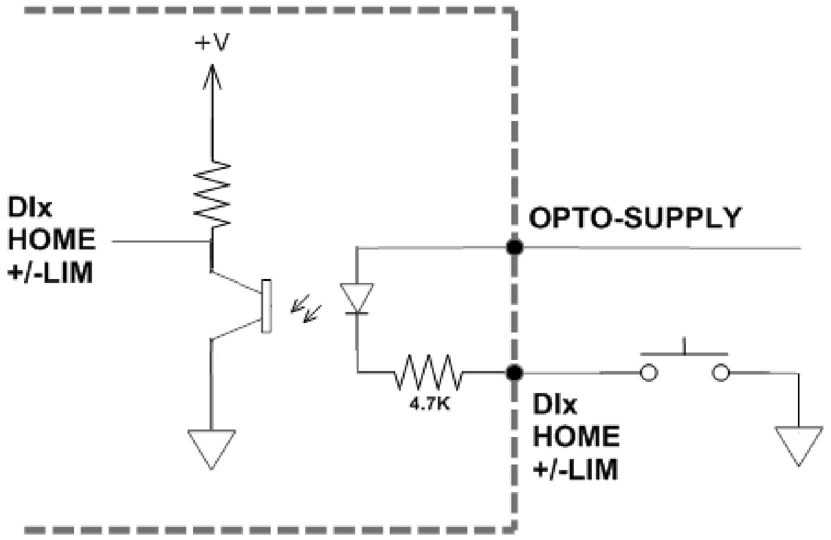


Figure 4.8

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

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

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

4.9. Digital Outputs

Figure 4.9 shows an example wiring of the digital outputs. All opto-isolated digital outputs will be PNP type.

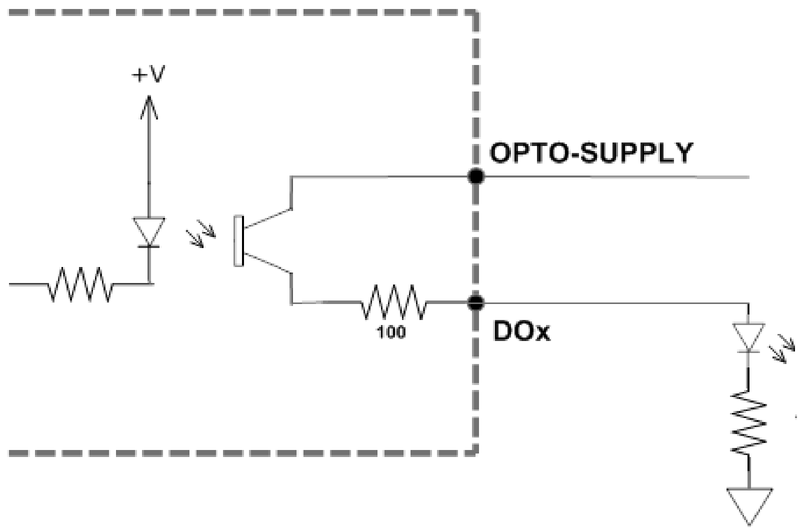


Figure 4.9

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

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

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

5. Communication Interface

5.1. USB Communication

ACE-SXC USB communication is 2.0 compliant.

In order to communicate with ACE-SXC via USB, the proper software driver must first be installed. Before connecting the ACE-SXC controller, or running any programs, please go to the Arcus-Technology website, download the Arcus Drivers and Tools Setup, and run the installation.

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

5.1.1. Typical USB Setup

The ACE-SXC can be connected to a PC directly via USB or through a USB hub. All USB cables should have a noise suppression choke to avoid communication loss or interruption. See a typical USB network setup in Figure 5.0 below.

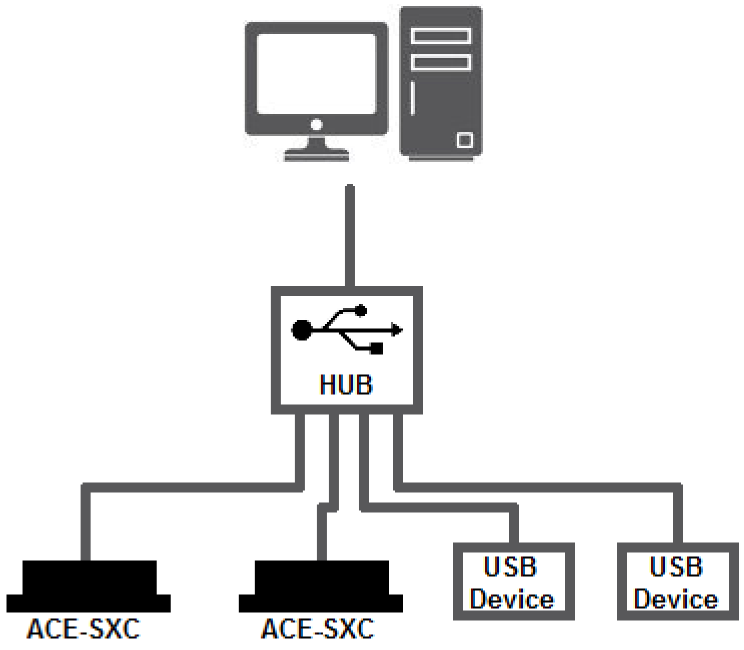


Figure 5.0

5.1.2. USB Communication API

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

Typical communication transaction time between PC and ACE-SXC for sending a command from a PC and getting a reply from the controller using the

fnPerformaxComSendRecv() API function is in single digit milliseconds. This value will vary with the CPU speed of the PC as well as the type of command.

For USB communication, the following DLL API functions are provided.

BOOL **fnPerformaxComGetNumDevices**(OUT LPDWORD lpNumDevices);

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

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

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

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

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

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

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

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

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

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

- This function is used to send commands and receive replies. The number of bytes to read and write must be 64 characters.

BOOL **fnPerformaxComFlush**(IN HANDLE pHandle)

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

5.1.3. 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 commands 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 the issue.

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

   BOOL **fnPerformaxComFlush**(IN HANDLE pHandle)

2. USB Cable**:** Another source of USB communication issues may come from the USB cable. Confirm that the USB cable being used has a noise suppression choke. See Figure 5.1.

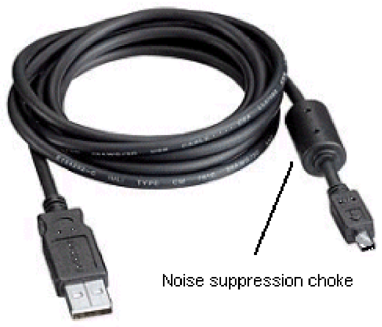


Figure 5.1

5.2. Device Number

If multiple ACE-SXC devices are connected to the PC, each device should have a unique device number. This will allow the PC to differentiate between multiple controllers. In order to make this change to an ACE-SXC, first store the desired number using the **DN** command. Note that this value must be within the range [SXC00 – 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: Device name is set to: **SXC00**

5.3. Windows GUI

The ACE-SXC comes with a Windows GUI program to test, program, compile, download, and debug the controller. The Windows GUI will perform all communication via USB . See section 7 for further details.

6. General Operation Overview

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

6.1. Motion Profile

By default, the ACE-SXC uses a trapezoidal velocity profile as shown in Figure 6.0.

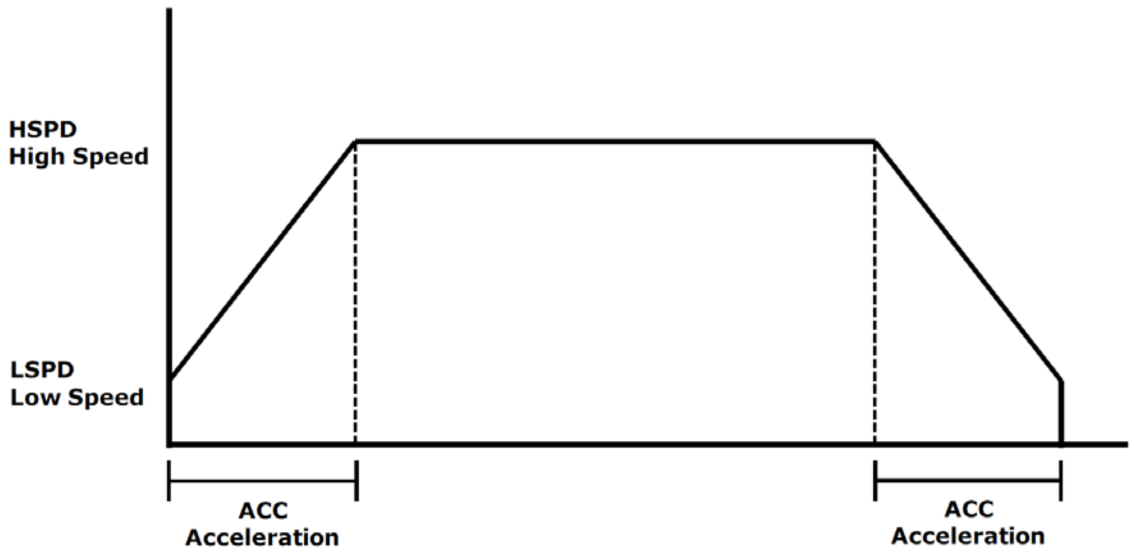


Figure 6.0

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

High speed and low speed are in pps (pulses/second). Pulse output rate supported is from 100 to 400K pps. Use ASCII commands **HSPD** and **LSPD** to set/get high speed and low speed settings. The same commands will be used in standalone mode.

Acceleration times are in milliseconds. Use the **ACC** command to set/get acceleration values. Acceleration range is from 10 msec to 1000 msec.

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

6.2. Motor Position
The ACE-SXC has a 32 bit signed step position counter. Range of the position counter is from –2,147,483,648 to 2,147,483,647. Motor position can be read using the ASCII command **PX**, which returns the pulse position.

To manually set/get the position of the motor, use the **PX=[value]** command.

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

6.3. Motor Power
The **EO** command can be used to enable or disable the current to the motor. The effect of the enable output signal will depend on the characteristics of the motor drive.

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

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

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

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

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

6.5. Stopping
When the motor is performing any type of move, motion can be stopped abruptly or with deceleration. It is recommended to use decelerated stops so that there is less impact on the system. Use the **ABORT**(ASCII) or **ABORTX**(standalone) command to immediately stop the motor. To employ deceleration on a stop, use the **STOP** (ASCII) or **STOPX** (standalone) command to stop the motor.

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

6.6. Positional Moves

The ACE-SXC can perform positional moves in absolute or incremental mode. For absolute mode, the **ABS** command should be used and, for incremental mode, the **INC** command should be used. These commands should be sent before the move command is issued. The move mode will remain in absolute or incremental mode until it is changed. The **MODE** command can be used to read the current move mode.

Return Value	Description
0	ABS (Absolute)
1	INC (Incremental)

Table 6.0

In absolute mode, the axis will move by the specified target position. In incremental mode, the axis will increase or decrease its current position by the specified target position.

Use the **X** command to make moves. For example, the **X1000** command will move the axis to position 1000 if performed in absolute mode.

The 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).

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

ASCII	**ABS**	**INC**	**MODE**	**X[target]**
Standalone	**ABS**	**INC**	**-**	**X[target]**

6.7. Homing

Home search routines involve moving the motor and using the home or limit inputs to determine the zero reference position. The zero reference position will be preserved as the position is marked when the home trigger is detected. If a motion command is sent while the controller is already moving, the command is not processed. Instead, an error response is returned. See table 9.1 for details on error responses.

The ACE-SXC has three different homing routines. The syntax for the home command in ASCII and standalone mode can be found below.

6.7.1. MODE 1: Home Input Only (High Speed Only)

Use the **H[+/-]** command in ASCII mode or the **HOMEX[+/-]** command in standalone mode to issue a homing command that uses the home input only. Figure 6.1 shows the homing routine.

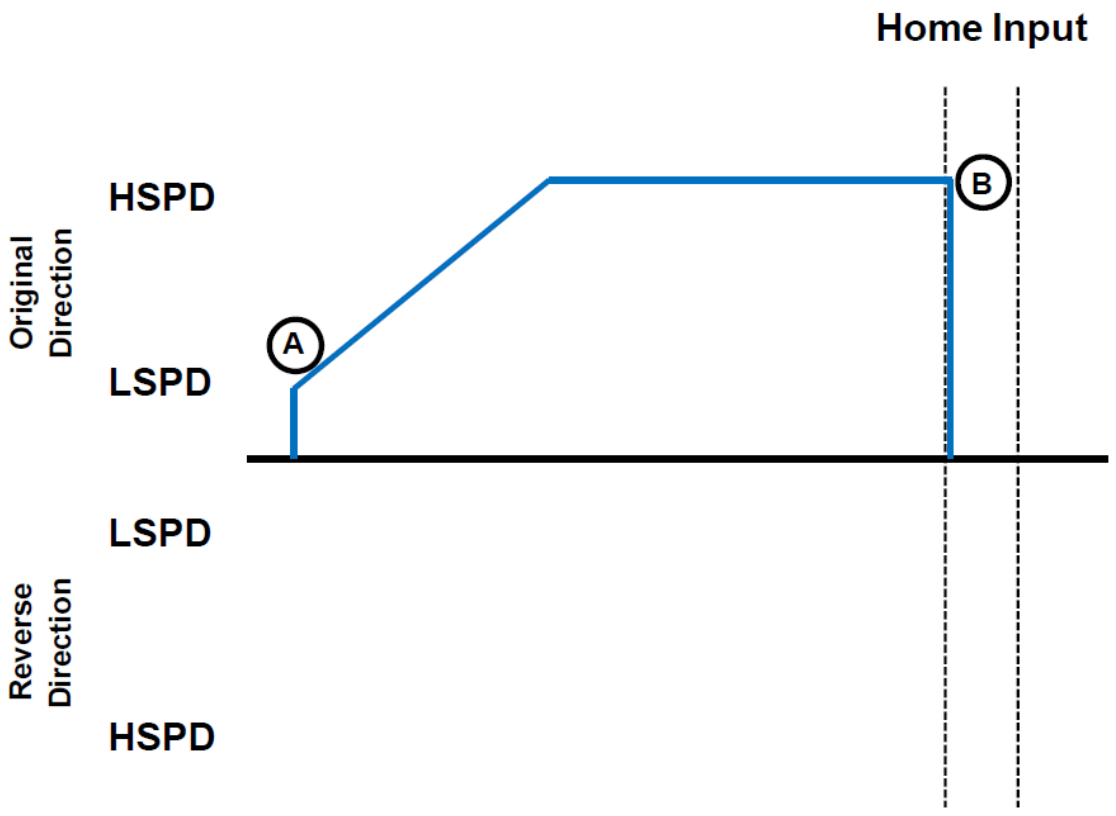


Figure 6.1

A. Issuing the command starts the motor from low speed and accelerates to high speed in search of the home input.
B. As soon as the home input is triggered, the position counter is reset to zero and the motor stops.

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

6.7.2. MODE 2: Limit Only

Use the **L[+/-]** command for ASCII mode or the **LHOMEX[+/-]** command for standalone mode. Figure 6.2 shows the homing routine.

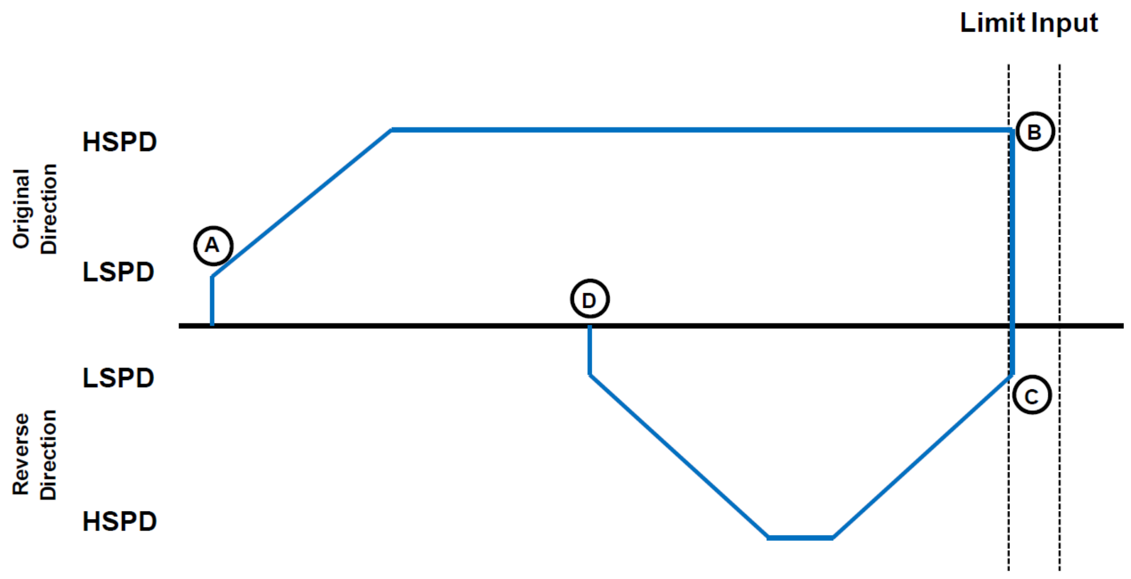


Figure 6.2

A. Issuing the command starts the motor from low speed and accelerates to high speed in search of the specified limit input.
B. If the limit correction amount (LCA) is set to zero, the motor will immediately stop when the relevant limit input is triggered (Finish Homing). Otherwise, the position counter will be set to the same value as the LCA.
C. The motor will start moving in the opposite direction. First accelerating to high speed, maintaining speed, and then decelerating.
D. Once position counter has reached zero, the motor stops.

ASCII	**L[+/-]**
Standalone	**LHOMEX[+/-]**

6.7.3. MODE 3: Home Input Only (High Speed and Low Speed)

Use the **HL[+/-]** command for ASCII mode and the **HLHOMEX[+/-]** for standalone mode. Figure 6.3 shows the homing routine.

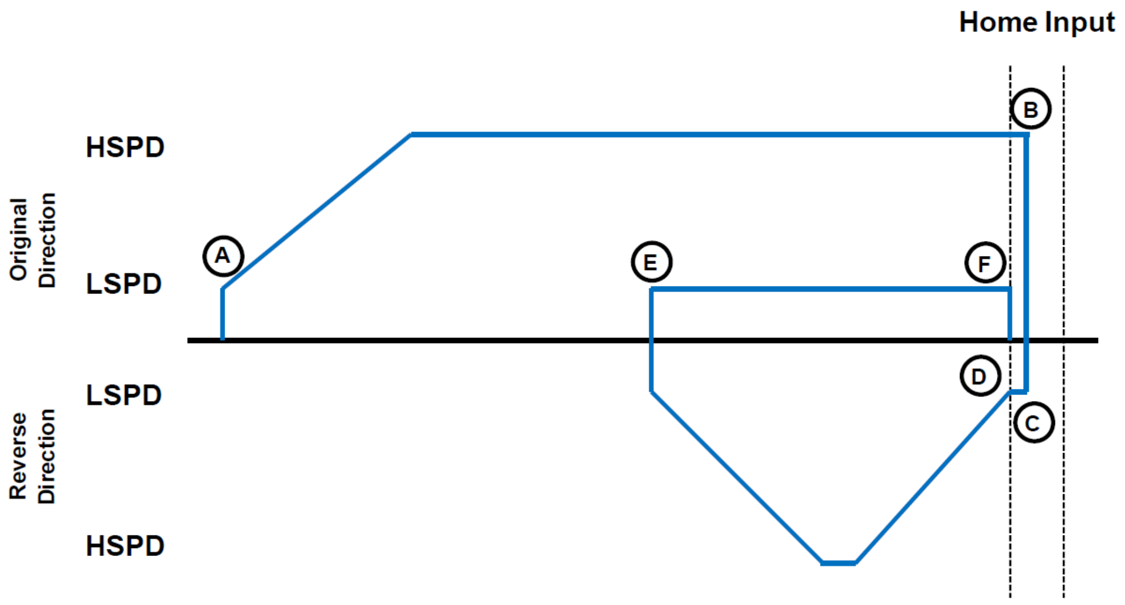


Figure 6.3

- **A.** Starts the motor from low speed and accelerates to high speed.
- **B.** As soon as the home input is triggered, the motor switches to low speed.
- **C.** The motor reverses direction until the home input is cleared.
- **D.** Once the home input is cleared, the motor accelerates to high speed and moves until the HCA (Home Correction Amount) is reached.
- **E.** The motor moves in the original direction at low speed.
- **F.** When the home input is triggered, the motor stops and the position is set to zero.

ASCII	**HL[+/-]**
Standalone	**HLHOMEX[+/-]**

6.8. Limits And Alarm 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.

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.

ASCII	**CLR**
Standalone	**ECLEARX**

6.9. 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 6.1

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.

6.10. Digital Inputs / Outputs / Configuration Button

The ACE-SXC module comes with three digital inputs and two digital outputs. If the limit/alarm function is disabled, up to eight inputs can be used as general purpose inputs.

6.10.1. Digital Inputs

Read digital inputs status using the **DI** command.

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). Digital inputs are active high.

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 6.2

If a digital input is on (i.e. input is pulled to GND of opto-supply), the bit status is 0. Otherwise, the bit status is 1.

ASCII	**DI**	**DI[1-8]**
Standalone	**DI**	**DI[1-8]**

6.10.2. Digital Outputs

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

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).

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

Table 6.3

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

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

6.10.3. Configuration Button

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.

6.11. Polarity

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

Bit	Description
0	Direction
1	Limit Input
2	Not Used
3	Digital Input
4	Configuration Button
5	Alarm Input
6	Digital Output
7	Enable Output

Table 6.4

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

6.12. Communication Time-Out 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 standalone mode. Refer to the **Example Standalone Programs** section for an example.

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

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

6.13. Standalone Program Specification

Standalone programming allows the controller to execute a user defined program that is stored in the internal memory of the ACE-SXC. The standalone program can be run independently of USB communication or while communication is still active.

Standalone programs can be written to the ACE-SXC using the Windows GUI described in section 7. Once a standalone program is written by the user, it is then compiled and downloaded to the ACE-SXC. Each line of written standalone code creates ~1-4 assembly lines of code after compilation.

The ACE-SXC can store and operate up to two separate standalone programs simultaneously.

6.13.1. Standalone Program Specification

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

6.13.2. Standalone Control

The ACE-SXC supports the simultaneous execution of two standalone programs. All programs can be controlled by the **SR[0-1]** command, where Program 0 uses command **SR0**, and Program 1 uses command **SR1**. For examples of multi-threading, please refer to section 10. The following assignments can be used for the **SR[0-1]** command.

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

Table 6.5

6.13.3. Standalone Status

The **SASTAT[0-1]** command can be used to determine the current status of the specified standalone program. Table 6.6 details the return values of this command.

Value	Description
0	Idle
1	Running
2	Paused
3	N/A
4	Errored

Table 6.6

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

6.13.4. Standalone Subroutines

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

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

6.13.5. Error Handling

Subroutine 31 is designated for error handling. If an error occurs during standalone execution (i.e. limit error,), the standalone program will automatically jump to SUB 31. If SUB 31 is not defined, the program will cease execution and go into error state.

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

The return jump from subroutine 31 will be determined by the ASCII command **SAP**. Write a "0" to this setting to have the standalone program jump back to the last performed line. Write a "1" to this setting to have the standalone program jump back to the first line of the program.

6.13.6. Standalone Variables

The ACE-SXC has 50 32-bit signed standalone variables available for general purpose use. They can be used to perform basic calculations and support integer operations. The **V[1-50]** command can be used to access the specified variables. The syntax for all available operations can be found below. Note that these operations can only be performed in standalone programming.

Operator	Description	Example
+	Integer Addition	V1=V2+V3
-	Integer Subtraction	V1=V2-V3
*	Integer Multiplication	V1=V2*V3

/	Integer Division (round down)	V1=V2/V3
%	Modulus	V1=V2%5
>>	Bit Shift Right	V1=V2>>2
<<	Bit Shift Left	V1=V2<<2
&	Bitwise AND	V1=V2&7
|	Bitwise OR	V1=V2|8
~	Bitwise NOT	V1=~V2

Table 6.7

Variables V26 through V50 can be stored to flash memory using the **STORE** command. Variables V1-V25 will be initialized to zero on power up.

6.13.7. Standalone Run On Boot-Up

Standalone can be configured to run on boot-up using the **SLOAD** command. Once this command has been issued, the **STORE** command will be needed to save the setting to flash memory. It will take effect on the following power cycle. See description in table 6.8 for the bit assignment of the **SLOAD** setting.

Bit	Description
0	Standalone Program 0
1	Standalone Program 1

Table 6.8

Standalone programs can also be configured to run on boot-up using the Windows GUI. See section 7 for details.

6.13.8. 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 X1000 move command begins; it will not wait until the controller is idle.

```
X1000       ;* Move to position 1000
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.

```
X1000           ;* Move to position 1000
WAITX           ;* Wait for the move to complete
V1=100
```

6.14. 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 6.9

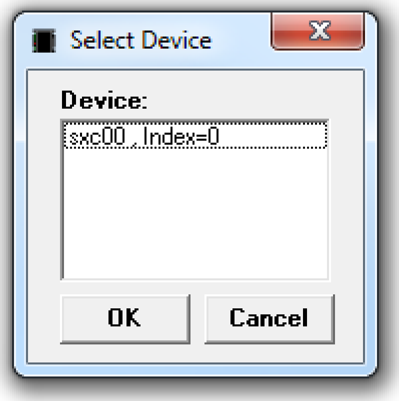
Note: When a standalone program is downloaded, the program is immediately written to flash memory.

7. Software Overview

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

To communicate over a USB connection, make sure that the ACE-SXC is connected to one of the available ports on the PC.

Startup the ACE-SXC GUI program and you will see the following screen in Figure 7.0. This will allow the user to select from all the ACE-SXC devices currently connected to the USB network. To choose a particular device, select it and then press OK.



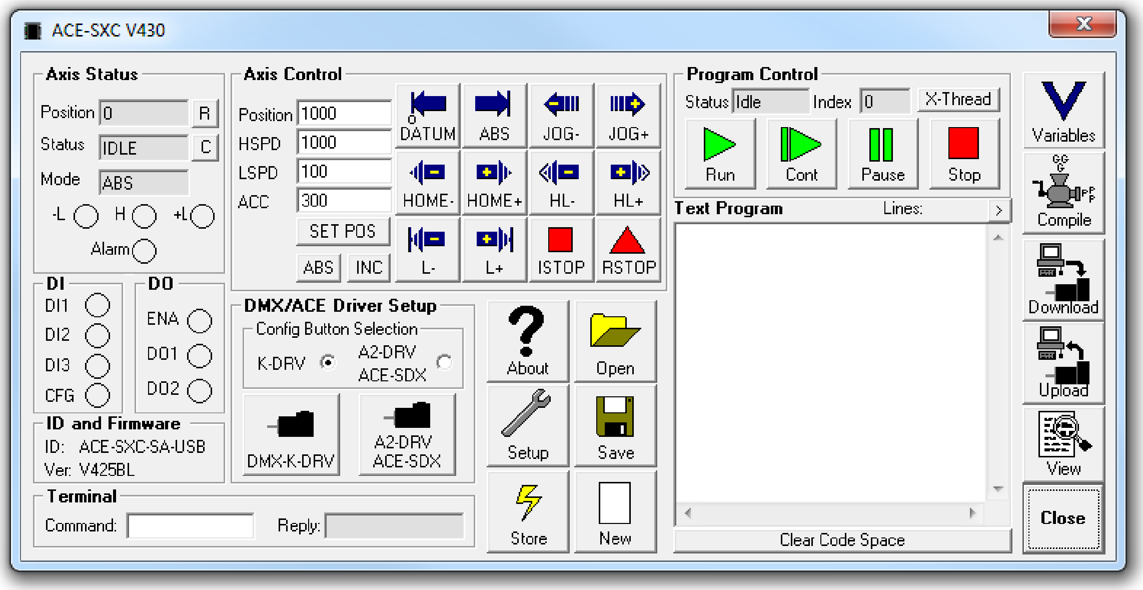
sxc00 , Index=0

Figure 7.0

Note: In order to communicate with ACE-SXC via USB, the proper driver must first be installed. Before connecting the ACE-SXC device or running any program, please go to the Arcus-Technology website, download the USB driver installation instruction and run the USB Driver Installation Program.

7.1. Main Control Screen

The Main Control Screen provides accessibility to all the available function on the ACE-SXC. All features can be tested and verified.



Axis Status: Position 0, Status IDLE, Mode ABS

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

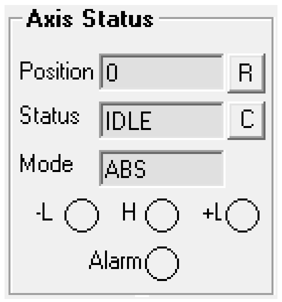
Program Control: Status Idle, Index 0

Config Button Selection: [✓ K-DRV] A2-DRV ACE-SDX

ID and Firmware
ID: ACE-SXC-SA-USB
Ver: V425BL

Figure 7.1

7.1.1. Status



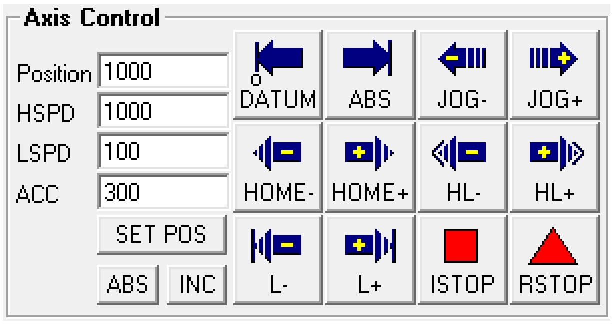
Position 0
Status IDLE
Mode ABS

Figure 7.2

1. **Position –** Display of the current motor position value. Reset Position button is used to reset the counter.
2. **Status –** Displays current motor status. Possible values are:
    a. **ACCEL** – acceleration in progress
    b. **CONST** – constant speed in progress
    c. **DECEL** – deceleration in progress
    d. **-LIM ERROR** – minus limit error occurred
    e. **+LIM ERROR** – plus limit error occurred
3. **Clear Error –** Clear Error button is used to clear any error status.
4. **Mode –** Displays the move mode of the controller. Possible values are:
    a. **ABS** – Absolute position movement.
    b. **INC** – Incremental position movement.

5. **Limit, Home, and Alarm Input Status –** Displays states of limit, home, and alarm inputs.

7.1.2. Control



Position 1000
HSPD 1000
LSPD 100
ACC 300

Figure 7.3

1. **Target Position/Speed/Accel**
    a. **Position** – Set the target position. This position is the pulse position
    b. **HSPD/LSPD** – Set the speed of the move. This value is in pulses/second
    c. **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 or minus direction.
10. **L+/- –** Limit homing in the plus or minus direction.
11. **ISTOP –** Immediate stop without deceleration.
12. **RSTOP –** Stop with deceleration.

7.1.3. Digital Input / Output / Enable

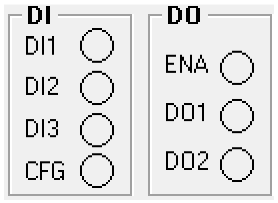


Figure 7.4

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.

7.1.4. Configuration

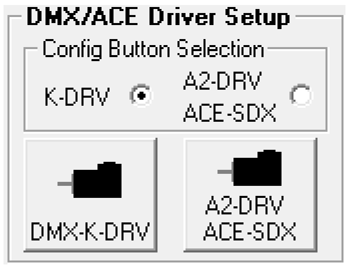


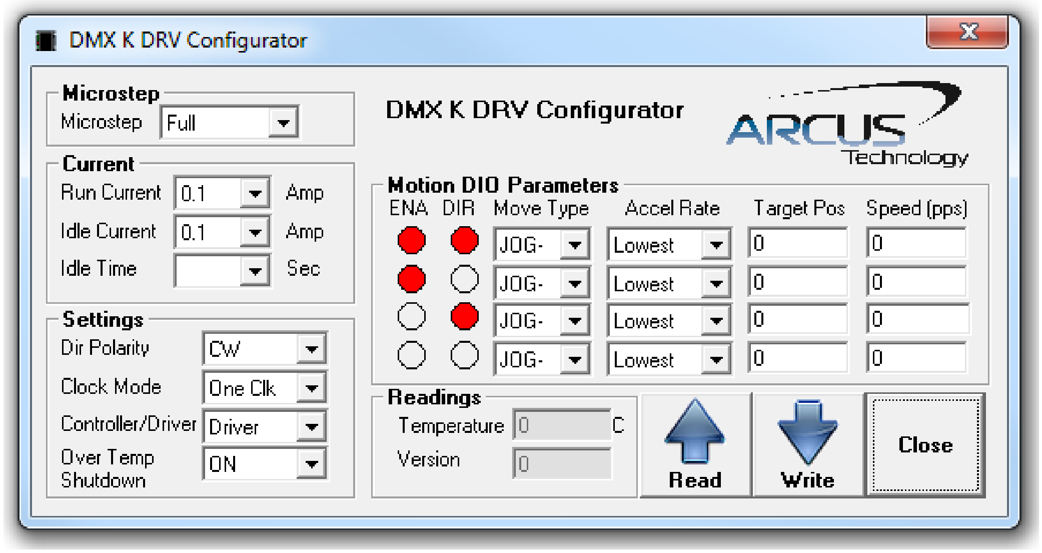
Figure 7.5

Figure 7.6

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.


DMX K DRV Configurator

Microstep

Microstep	Full

Current

Run Current	0.1	Amp
Idle Current	0.1	Amp
Idle Time		Sec

Settings

Dir Polarity	CW
Clock Mode	One Clk
Controller/Driver	Driver
Over Temp Shutdown	ON

DMX K DRV Configurator

Motion DIO Parameters

ENA	DIR	Move Type	Accel Rate	Target Pos	Speed (pps)
●	●	JOG-	Lowest	0	0
●	○	JOG-	Lowest	0	0
○	●	JOG-	Lowest	0	0
○	○	JOG-	Lowest	0	0

Readings

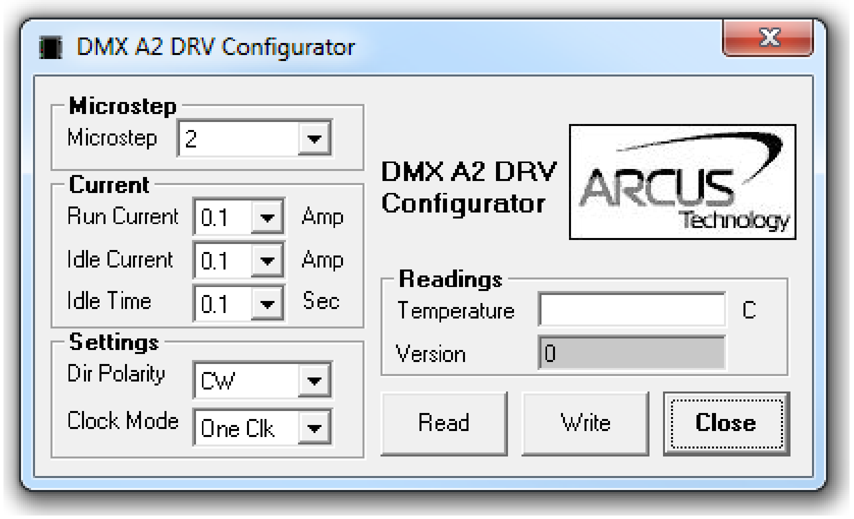
Temperature	0	C
Version	0	

Figure 7.7

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.


DMX A2 DRV Configurator

Microstep

Microstep	2

Current

Run Current	0.1	Amp
Idle Current	0.1	Amp
Idle Time	0.1	Sec

Settings

Dir Polarity	CW
Clock Mode	One Clk

DMX A2 DRV Configurator

Readings

Temperature		C
Version	0	

Figure 7.8

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.

4. **Store To Flash –** Parameters on the ACE can be permanently stored to the flash memory. See Table 6.9 for variables that are stored to flash.

7.1.5. Program File Control

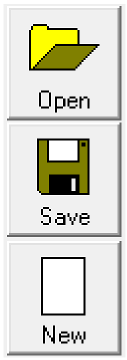


Figure 7.9

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

7.1.6. Standalone Program Editor

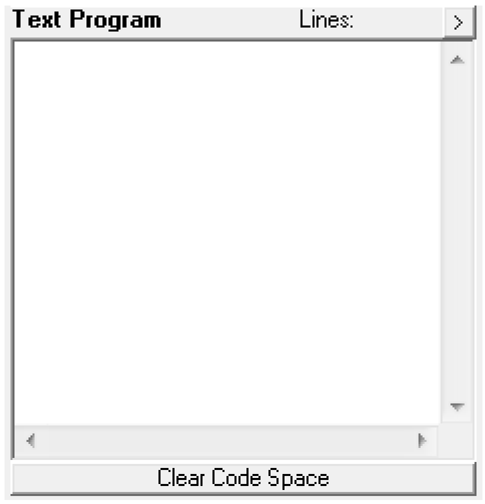


Figure 7.10

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

7.1.7. Standalone Program Control

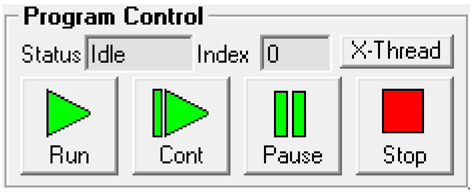


Figure 7.11

1. **Program Status** – Displays the program status. Possible statuses include:
   a. **Idle** – Program is not running.
   b. **Running** – Program is running.
   c. **Paused** – Program is paused.
   d. **Errored** – Program is in error state.
2. **Program Index** – Displays the current line of low-level code that is being executed.
3. **X-Thread** – Opens the Program Control for standalone multi-thread operation. Will allow control of both standalone programs.
4. **Run** – Runs the standalone program (PRG0).
5. **Cont.** – Continues the paused standalone program (PRG0).
6. **Pause** – Pauses the standalone program (PRG0).
7. **Stop** – Stops the standalone program (PRG0).

7.1.8. Standalone Program Compile / Download / Upload / View

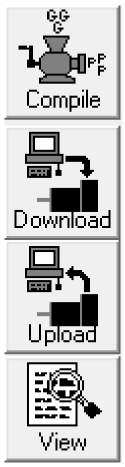
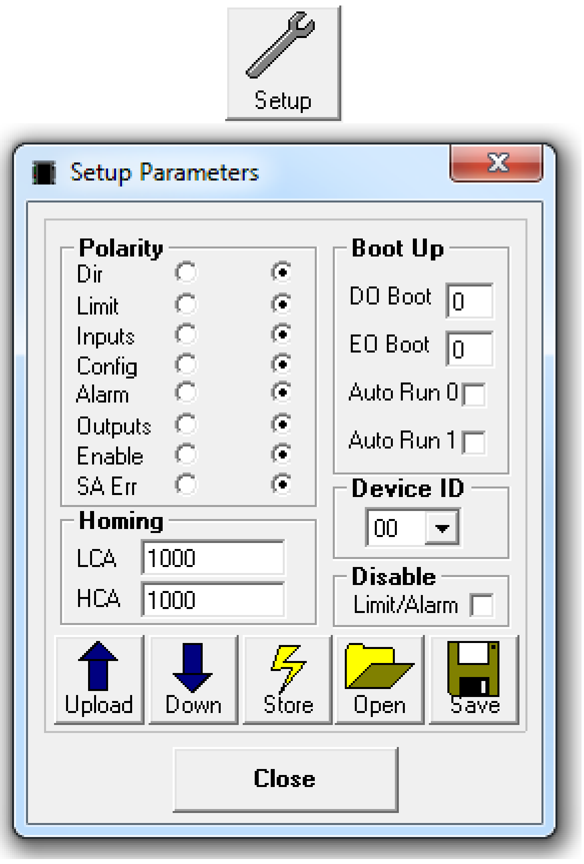


Figure 7.12

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

7.1.9. Setup



Polarity

Dir	○	●
Limit	○	●
Inputs	○	●
Config	○	●
Alarm	○	●
Outputs	○	●
Enable	○	●
SA Err	○	●

Homing

LCA 1000

HCA 1000

Boot Up

DO Boot 0

EO Boot 0

Auto Run 0 ☐

Auto Run 1 ☐

Device ID

00

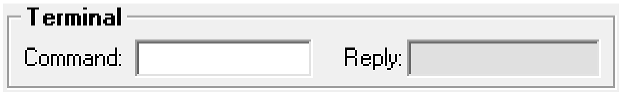
Disable

Limit/Alarm ☐

Figure 7.13

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 [SXC00 – 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.

7.1.10. Terminal



Terminal

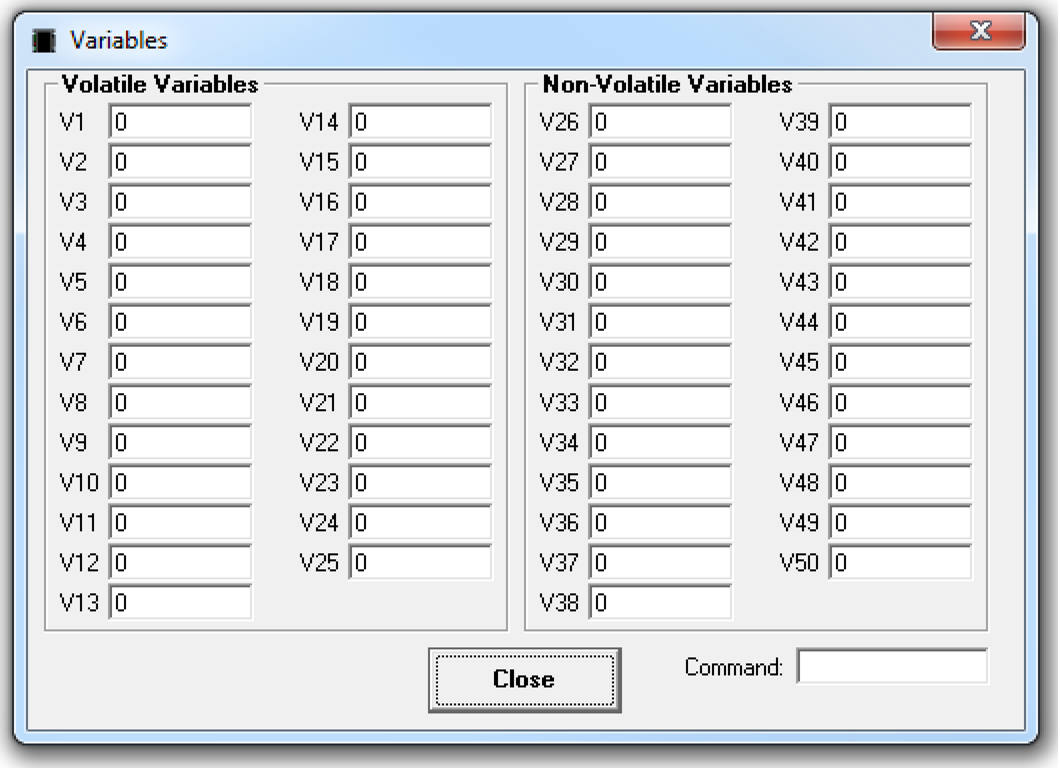
Command:

Reply:

Figure 7.14

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

7.1.11. Variable Status



Variables

Volatile Variables

V1	0	V14	0
V2	0	V15	0
V3	0	V16	0
V4	0	V17	0
V5	0	V18	0
V6	0	V19	0
V7	0	V20	0
V8	0	V21	0
V9	0	V22	0
V10	0	V23	0
V11	0	V24	0
V12	0	V25	0
V13	0		

Non-Volatile Variables

V26	0	V39	0
V27	0	V40	0
V28	0	V41	0
V29	0	V42	0
V30	0	V43	0
V31	0	V44	0
V32	0	V45	0
V33	0	V46	0
V34	0	V47	0
V35	0	V48	0
V36	0	V49	0
V37	0	V50	0
V38	0		

Command:

Figure 7.15

1. **Command: –** To write to a variable, use V[1-50] = [value].

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.

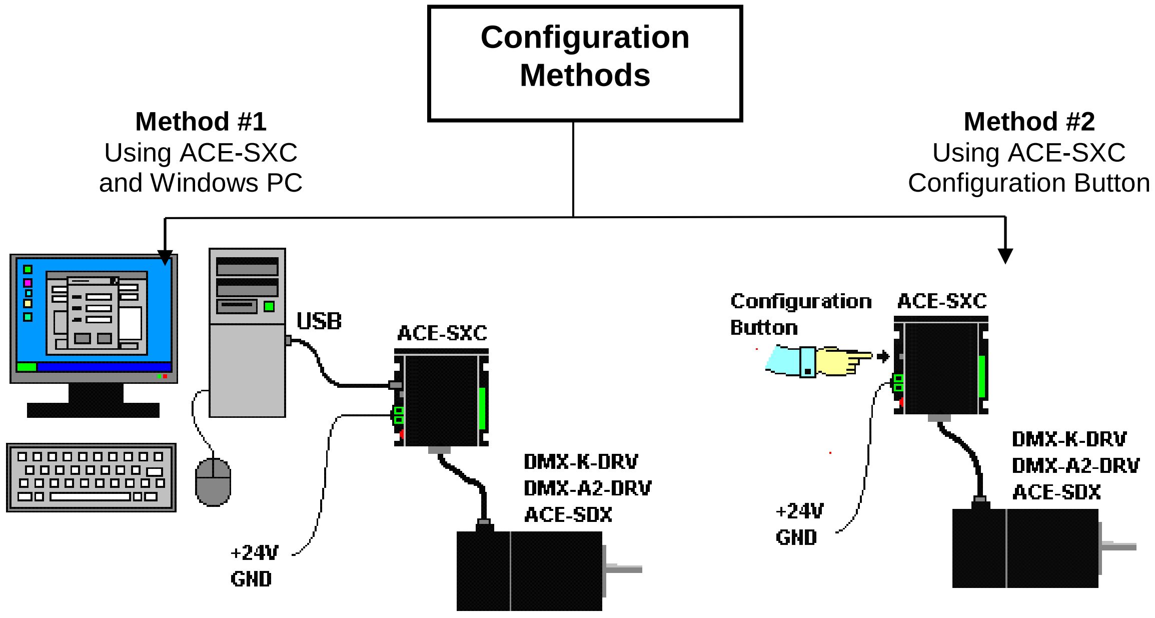


Figure 8.0

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.

8.2. 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 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. ASCII Language Specification

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

ACE-SXC language is case sensitive. All commands should be in upper case letters. Invalid commands are returned with a "?". Always check for the proper reply when a command is sent.

9.1. ASCII Command Set

Command	Description	Return
ABORT	Immediately stops the motor if in motion. For decelerate stop, use STOP command.	OK
ABS	Set move mode to absolute	OK
ACC	Returns current acceleration value in milliseconds. (10-1000)	milliseconds
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.	OK
DI	Return status of digital inputs	Refer to Table 6.2
DI[1-8]	Return individual status of inputs	Refer to Table 6.2
DO	Return status of digital outputs	Refer to Table 6.3
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 6.3
DO[1-2]=[Value]	Set individual output. Refer to Table 6.3	OK
DOBOOT	Get DO boot-up state.	
DOBOOT = [Value]	Set DO boot-up state.	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 status.	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

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	Returns High Speed Setting.	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 6.1
POL	Returns current polarity	Refer to Table 6.4
POL=[value]	Sets polarity. Refer to Table 6.4	OK
PX	Returns current position value	32-bit number
PX=[value]	Sets the current position value	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]	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	OK

	1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program	
SPC[0-1]	Get program counter for standalone program	[0-1274]
STOP	Stops the motor using deceleration if in motion.	OK
STORE	Store settings to flash	OK
TOC	Get communication time out parameter	milliseconds
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 9.0

9.2. 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 9.1

10. Standalone Language Specification

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

10.1. Standalone Command Set

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

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

DELAY	W	Set a delay in milliseconds. Assigned value is a 32-bit unsigned integer or a variable.	DELAY=1000 ;1 second
			DELAY=V1 ;assign to variable

DI	R	Return status of digital inputs. See Table 6.2 for bitwise assignment.	IF DI=0
			  DO=1 ;Turn on DO1
			ENDIF
			V2=DI

DI[1-8]	R	Get individual bit status of digital inputs. Will return [0,1]. See Table 6.2 for bitwise assignment.	IF DI1=0
			  DO=1 ;Turn on DO1
			ENDIF
			V3=DI1

DO	R/W	Set/get digital output status. See Table 6.3 for bitwise assignment.	DO=2 ;Turn on DO2
DO[1-2]	R/W	Set/get individual bit status of digital outputs. Range for the bit assigned digital outputs is [0,1].	DO2=1 ;Turn on DO2
ECLEARX	W	Clear any motor status errors.	ECLEARX
END	-	Indicate end of program. Program status changes to idle when END is reached. NOTE: Subroutine definitions should be written AFTER the END statement.	END
ENDSUB	-	Indicated end of subroutine. When ENDSUB is reached, the program returns to the previously called subroutine.	ENDSUB
EO	R/W	Set/get the enable output status.	EO=1 ;Enable the motor
GOSUB [0-31]	-	Call a subroutine that has been previously stored to flash memory.	GOSUB 0 END
HLHOMEX[+/-]	W	Home the motor using the home input at low and high speeds in the specified direction.	HLHOMEX+ ;positive home WAITX ;wait for home move
HOMEX[+/-]	W	Home the motor using the home input at high speed in specified direction.	HOMEX- ;negative home WAITX ;wait for home move

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

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

INC	W	Set the move mode to incremental mode.	INC X1000 ;increment by 1000
JOGX[+/-]	W	Move the motor indefinitely in the specified direction.	JOGX+
LHOMEX[+/-]	W	Home the motor using the limit inputs in the specified directions.	LHOMEX+ ;positive home WAITX

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

MSTX	R	Get the current motor status of the motor.	MSTX
PRG[0-1]	-	Used to define the beginning and end of a main program. Two standalone programs are available.	PRG 0 ;main program

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

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

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

TOC	W	Sets the communication time-out parameter. Units are in milliseconds.	TOC=1000 ;1 SECOND TIMEOUT

V[1-50]	R/W	Set/get standalone variables.
		The following operations are available:
		[+] Addition
		[-] Subtraction
		[*] Multiplication
		[/] Division
		[%] Modulus
		[>>] Bit shift right
		[<<] Bit shift left
		[&] Bitwise AND
		[|] Bitwise OR
		[~] Bitwise NOT
			V1=12345 ;Set V1 to 12345
			V2=V1+1 ;Set V2 to V1 + 1
			V3=DI ;Set V3 to DI
			V4=DO ;SET V4 TO DO
			V5=~EO ;SET V5 TO NOT EO

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

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

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

Table 10.0

10.2. Example Standalone Programs

10.2.1. Standalone Example Program 1 – Single Thread

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

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

10.2.2 Standalone Example Program 2 – Single Thread

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

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

10.2.3 Standalone Example Program 3 – Single Thread

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

```
HSPD=20000          ;* Set the high speed to 20000 pulses/sec
LSPD=1000           ;* Set the low speed to 1000 pulses/sec
ACC=300             ;* Set the acceleration to 300 msec
EO=1                ;* Enable the motor power
V1=0                ;* Set variable 1 to value 0
WHILE V1<10         ;* Loop while variable 1 is less than 10
     X1000          ;* Move to zero
     WAITX          ;* Wait for move to complete
     X0             ;* Move to 1000
     V1=V1+1        ;* Increment variable 1
ENDWHILE            ;* Go back to WHILE statement
END
```

10.2.4. Standalone Example Program 4 – Single Thread

Task: Move the motor back and forth between position 1000 and 0 only if the digital input 1 is turned on.

```
HSPD=20000               ;* Set the high speed to 20000 pulses/sec
LSPD=1000                ;* Set the low speed to 1000 pulses/sec
ACC=300                  ;* Set the acceleration to 300 msec
EO=1                     ;* Enable the motor power
WHILE 1=1                ;* Forever loop
     IF DI1=1            ;* If digital input 1 is on, execute the statements
          X1000          ;* Move to zero
          WAITX          ;* Wait for move to complete
          X0             ;* Move to 1000
     ENDIF
ENDWHILE                 ;* Go back to WHILE statement
END
```

10.2.5. Standalone Example Program 5 – Single Thread

Task: Using a subroutine, increment the motor by 1000 whenever the DI1 rising edge is detected.

```
HSPD=20000                  ;* Set the high speed to 20000 pulses/sec
LSPD=1000                   ;* Set the low speed to 1000 pulses/sec
ACC=300                     ;* Set the acceleration to 300 msec
EO=1                        ;* Enable the motor power
V1=0                        ;* Set variable 1 to zero
WHILE 1=1                   ;* Forever loop
        IF DI1=1            ;* If digital input 1 is on, execute the statements
                GOSUB 1     ;* Move to zero
        ENDIF
ENDWHILE                    ;* Go back to WHILE statement
END

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

10.2.6. Standalone Example Program 6 – Single Thread

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

```
HSPD=20000                ;* Set the high speed to 20000 pulses/sec
LSPD=1000                 ;* Set the low speed to 1000 pulses/sec
ACC=300                   ;* Set the acceleration to 300 msec
EO=1                      ;* Enable the motor power
WHILE 1=1                 ;* Forever loop
     IF DI1=1             ;* If digital input 1 is on
          X1000           ;* Move to 1000
     ELSEIF DI2=1         ;* If digital input 2 is on
          X2000           ;* Move to 2000
     ELSEIF DI3=1         ;* If digital input 3 is on
          X3000           ;* Move to 3000
     ELSEIF DI5=1         ;* If digital input 5 is on
          HOMEX-          ;* Home the motor in negative direction
     ENDIF
     V1=MSTX              ;* Store the motor status to variable 1
     V2=V1&7              ;* Get first 3 bits
     IF V2!=0
          DO1=1
     ELSE
          DO1=0
     ENDIF
ENDWHILE                  ;* Go back to WHILE statement
END
```

10.2.7. Standalone Example Program 7 – Multi Thread

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

```
PRG 0                   ;* Start of Program 0
HSPD=20000              ;* Set high speed to 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
```

10.2.8. Standalone Example Program 8 – Multi Thread

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

```
PRG 0                   ;* Start of Program 0
HSPD=1000               ;* Set high speed to 1000 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
```

A: Speed Settings

HSPD value [PPS]	Min. LSPD value	Min. ACC [ms]	δ	Max ACC setting [ms]
1 - 400 K	100	1	500	5000

Table A.0

A.1 Acceleration/Deceleration Range

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

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

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

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

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

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** = 400,000 pps, **LSPD** = 1000 pps:
   a. Min acceleration allowable: **1 ms**
   b. Max acceleration allowable:
      ((400,000 – 1000) / 18,000) x 1000 ms = **22166** ms (22.17 sec)

A.2 Acceleration/Deceleration Range – Positional Move

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

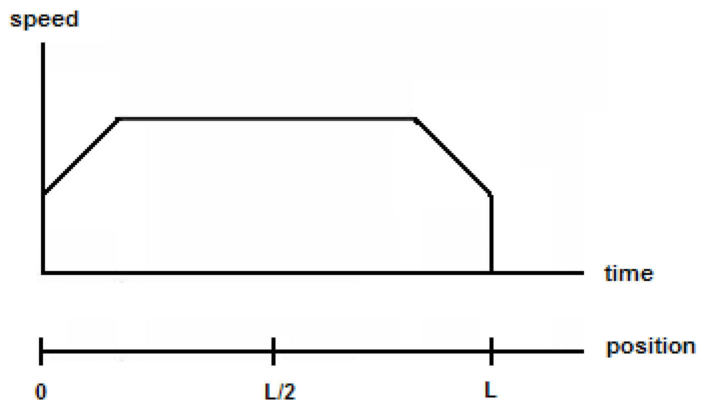


Figure A.1

1. <u>ACC vs. DEC 1:</u> If the theoretical position where the controller begins decelerations 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 1:</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 a 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