ACE-SXE

Advanced Standalone Step Motor Controller
USB 2.0 / RS-485 communication

First Edition, January 2008

Revision History:

1.10 – 1st Release
1.17 – 2nd Release
1.18 – 3rd Release
1.22 – 4th Release
1.26 – 5th Release

Firmware Compatibility:

†V303BL

†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-SXE is an advanced single axis stepper standalone programmable motion controller.

Communication to the ACE-SXE can be established over USB or RS-485. It is also possible to download a standalone program to the device and have it run independent of a host.

1.1. Features

- USB 2.0 communication
- RS-485 ASCII communication
    - 9600, 19200, 38400, 57600, 115200 bps
- Modbus-RTU Protocol
- Standalone programmable using A-SCRIPT
- Pulse/Dir/Enable open collector outputs per axis
    - Maximum pulse output rate of 6M PPS
- Advanced Motion features
    - Trapezoidal or s-curve acceleration
    - On-the-fly speed and/or target change
- Digital IO communication
    - 4 bit motion profile select inputs (DI3-DI6)
    - One start motion input (DI1)
    - One abort/clear motion input (DI2)
    - One in position output (DO1)
    - One error output (DO2)
- A/B/Z differential encoder inputs
    - StepNLoop closed loop control (position verification)
- Opto-isolated I/O
    - 6 x Inputs
    - 2 x Outputs
    - 1 x High speed position capture latch input
    - +Limit/-Limit/Home inputs
- Homing routines:
    - Home input only (high speed)
    - Home input only (high speed + low speed)
    - Limit only
    - Z-index encoder channel only
    - Home input + Z index encoder channel
- 2 x 10-bit analog inputs
    - Joystick control

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
	-	500	mA
Opto-supply Power Input	+12	+24	V
Digital Input Forward Diode Current	-	45	mA
Digital Output Collector Voltage	-	+24	V
Digital Output Sink Current	-	90	mA
Analog Inputs	0	+5	V
	-	22	mA
Enable Output	-	24	V
Enable Output Sink Current	-	90	mA
Operating Temperature[2]	-20	+80	°C
Storage Temperature[2]	-55	+150	°C

Table 2.0

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

3. Dimensions

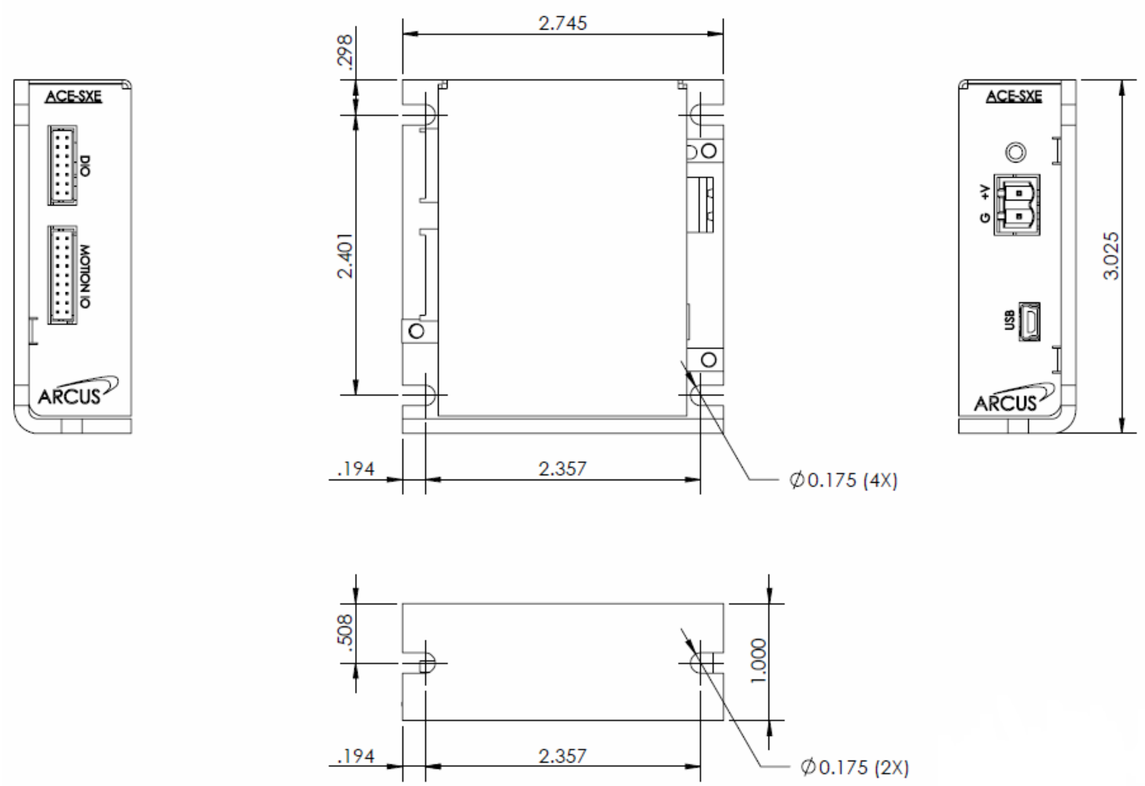


Figure 3.0

†All dimensions in inches

4. Connectivity

In order for ACE-SXE to operate, it must be supplied with +12VDC to +48VDC. Power pins as well as communication port pin outs are shown below.

4.1. 2-Pin Power Connector (5.08mm)

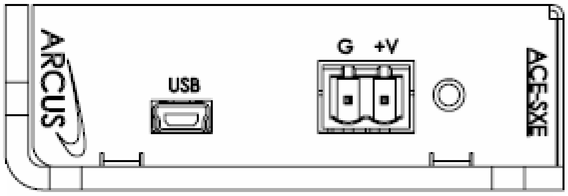


Figure 4.0

Pin #	Name	In/Out	Description
1	G	I	Ground
2	V+	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. 14-Pin DIO Connector (2mm)

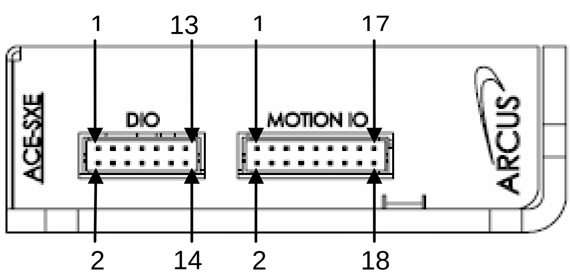


Figure 4.1

Pin #	Wire Color	In/Out	Name	Description
1	Red	O	V+ OUT	Shorted to pin 2 (V+) of the 2-pin 5.08mm connector
2	Orange	I	OPTO	+12 to +24VDC opto-supply input – used for limit, home and digital inputs
3	White/Yellow	I	-LIM	Minus Limit Input
4	Yellow	I	+LIM	Plus Limit Input
5	Yellow/Orange	I	LATCH	Latch Input
6	White	I	HOME	Home Input
7	Yellow/Brown	I	DI2	Digital Input 2

8	Yellow/Brown	I	DI1	Digital Input 1
9	Yellow/Brown	I	DI4	Digital Input 4
10	Yellow/Brown	I	DI3	Digital Input 3
11	Yellow/Brown	I	DI6	Digital Input 6
12	Yellow/Brown	I	DI5	Digital Input 5
13	Yellow/Black	O	DO2	Digital Output 2
14	Yellow/Black	O	DO1	Digital Output 1

Table 4.1

Mating Connector Description: 14 pin 2mm dual row connector
Mating Connector Manufacturer: HIROSE
Mating Connector Housing Part Number: DF11-14DS-2C
Mating Connector Pin Part Number: DF11-2428SC

4.3. 18-Pin Motion IO Connector (2mm)

Pin #	Wire Color	In/Out	Name	Description
1	Black	O	GND	Shorted to pin 1 (GND) of the 2-pin 5.08mm connector
2	Orange/Red	O	5V	5V output from controller – from on-board regulator
3	White/Brown	I	/Ae	Differential encoder /A channel
4	Brown	I	Ae	Differential encoder A channel
5	White/Purple	I	/Be	Differential encoder /B channel
6	Purple	I	Be	Differential encoder B channel
7	Orange/Black	I	/Ze	Differential encoder /Z channel
8	Orange/Blue	I	Ze	Differential encoder Z channel
9	White/Green	O	PUL-	Differential negative pulse output
10	Green	O	PUL+	Differential positive pulse output
11	White/Blue	O	DIR-	Differential negative direction output
12	Blue	O	DIR+	Differential positive direction output
13	Orange/Red	O	5V	5V output from controller – from on-board regulator
14	White/Orange	O	ENA	Enable Output
15	Yellow/Purple	I	AI2	Analog Input 2
16	Yellow/Purple	I	AI1	Analog Input 1
17	Brown/Yellow	I	485-	RS-485 minus signal
18	Brown/Green	I	485+	RS-485 plus signal

Table 4.2

Mating Connector Description: 18 pin 2mm dual row connector
Mating Connector Manufacturer: HIROSE
Mating Connector Housing Part Number: DF11-18DS-2C
Mating Connector Pin Part Number: DF11-2428S

4.4. ACE-SXE Interface Circuit

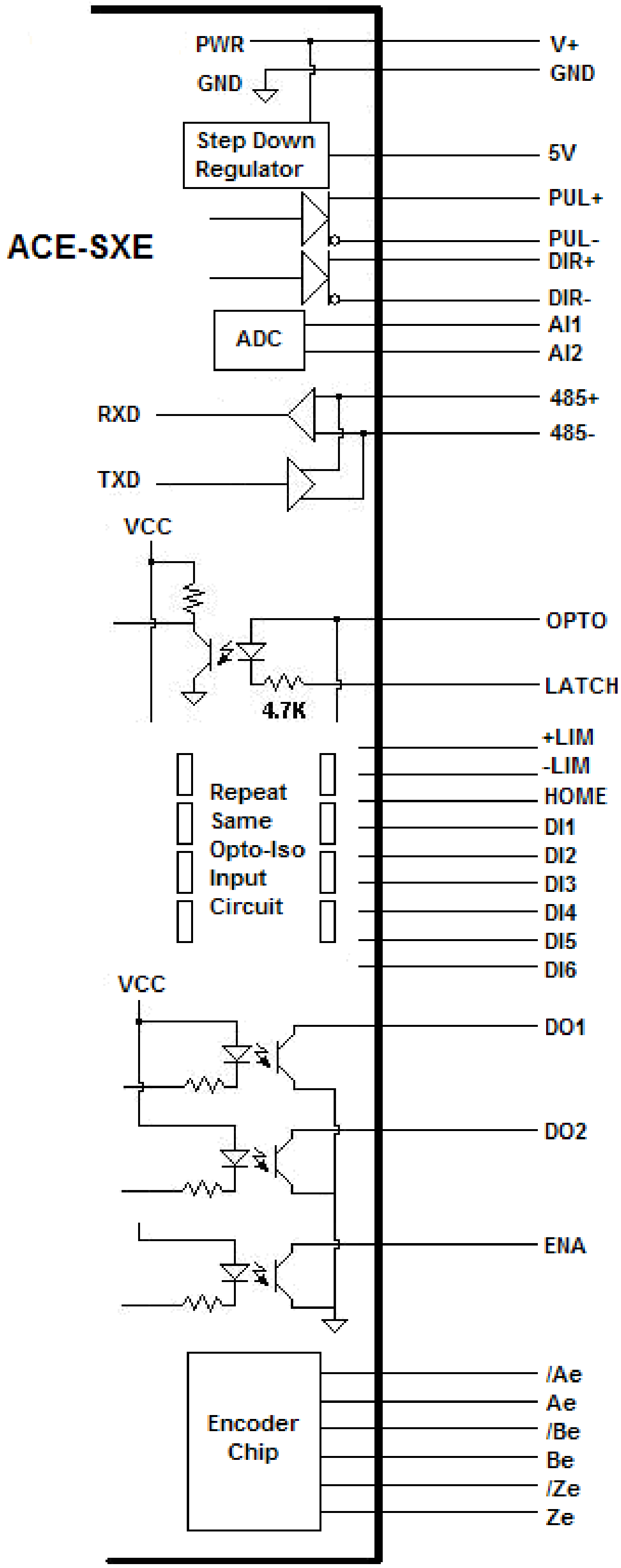


Figure 4.2

4.5. Digital Outputs

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

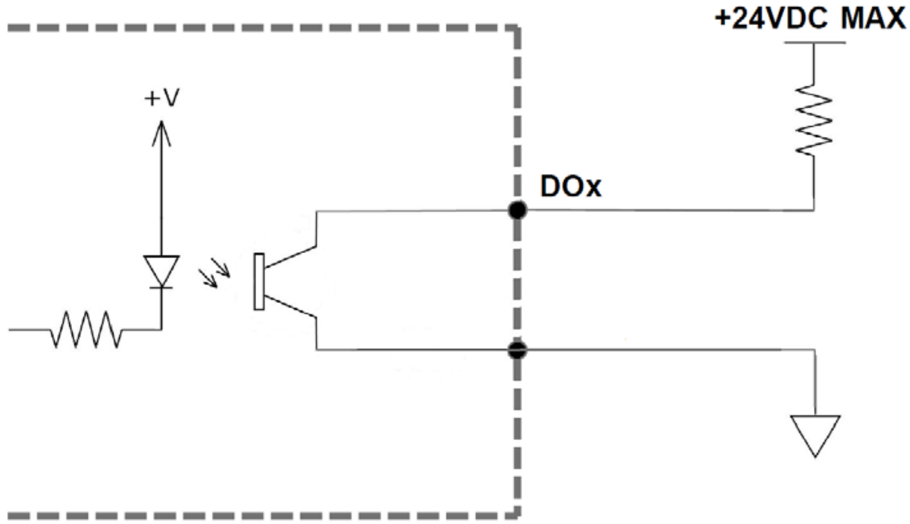


Figure 4.3

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

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

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

4.6. Digital Inputs, Home, Limit, and Latch

Figure 4.4 shows the detailed schematic of the opto-isolated general purpose digital inputs, home, limit, and latch.

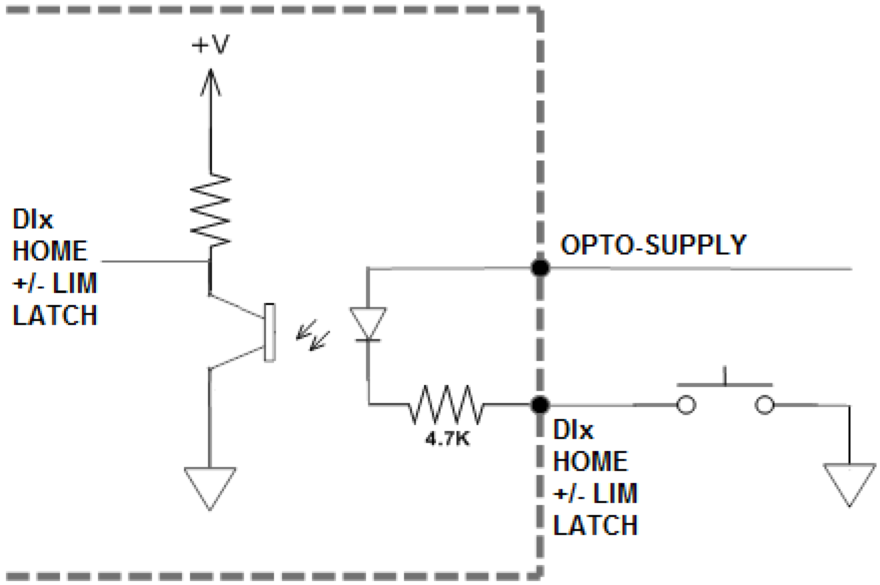


Figure 4.4

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

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

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

4.7. Enable Output

The enable is an open collector output. Figure 4.5 shows the detailed schematic of the enable output.

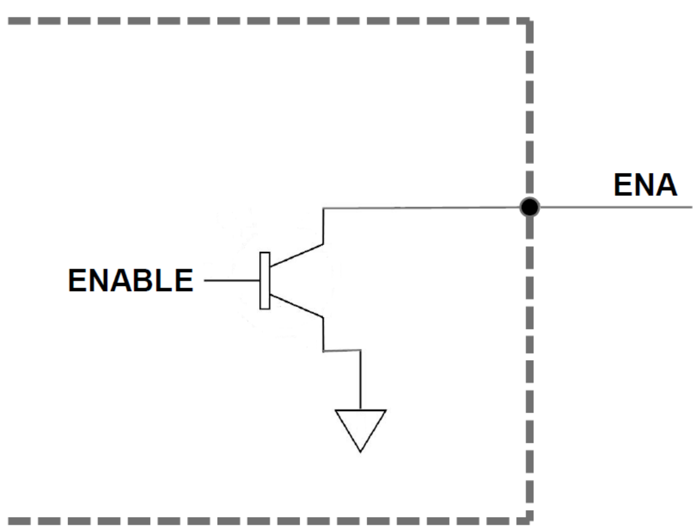


Figure 4.5

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

4.8. Pulse and Direction Outputs

The pulse and direction are both open collector outputs. Figure 4.6 shows an example of a differential ended connection. Figure 4.7 shows an example of a single-ended connection.

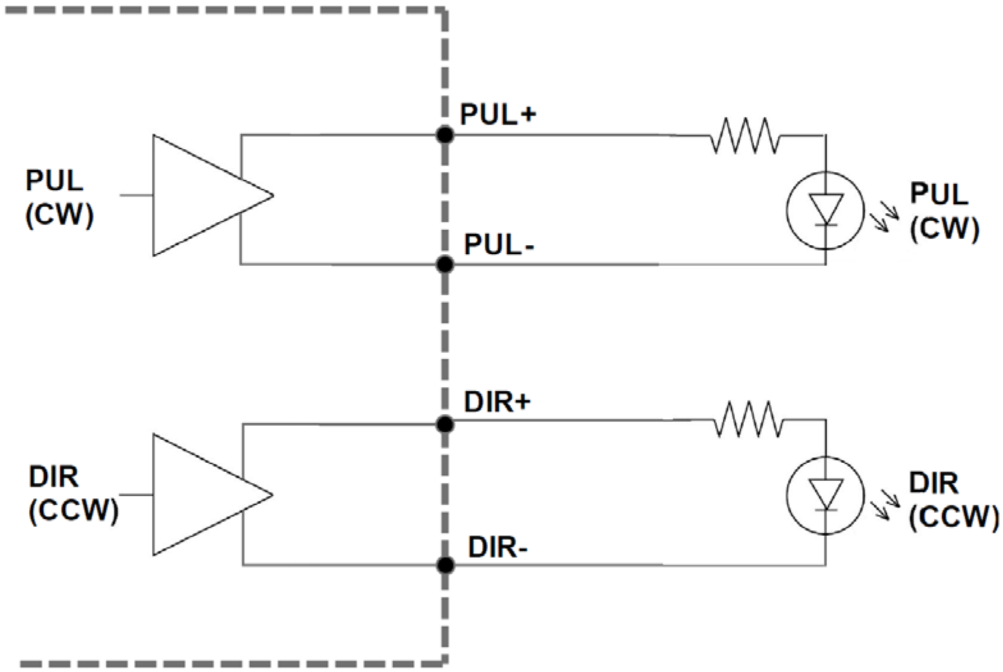


Figure 4.6

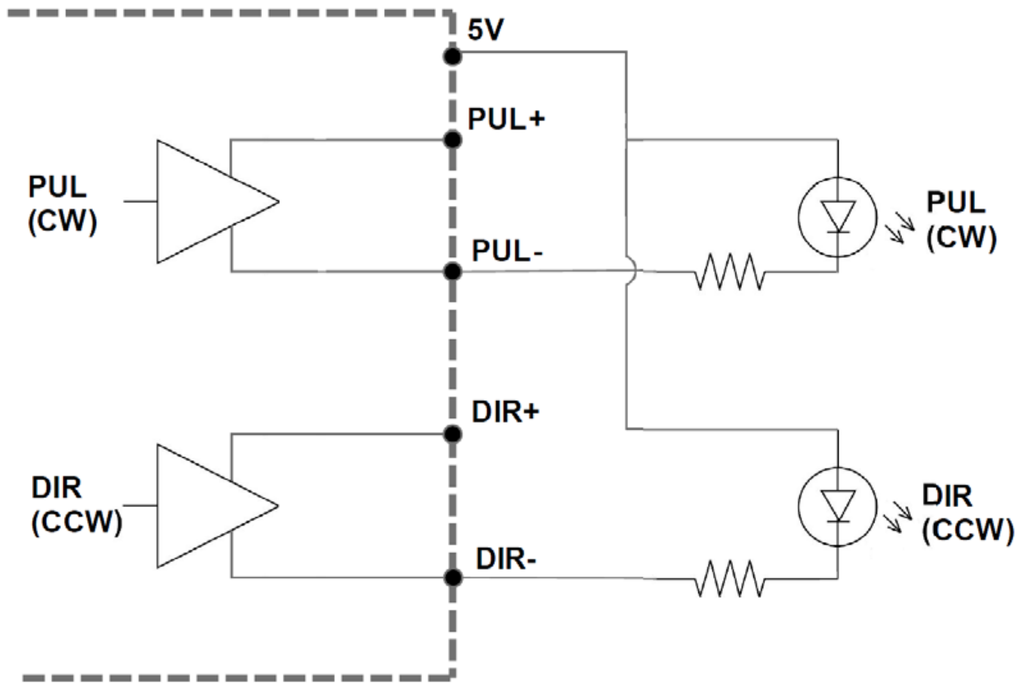


Figure 4.7

4.9. Encoder Input Connection

Both single-ended and differential quadrature encoder inputs are accepted.

When using single-ended encoders, use the /A, /B, and /Z inputs.

+5V supply and Ground signals are available to power the encoder. Make sure that the total current usage is less than 200mA for the +5V.

The maximum encoder frequency is 3MHz.

4.10. Analog Inputs

Analog inputs are 0 to 5V range and 10 bit in resolution. Two analog input channels are available for general purpose use or for joystick control use (AI1). The analog values are in mV. Section 6.17 will provide details on joystick control.

The maximum source current for the analog inputs is 10mA.

5. Communication Interface

5.1. USB Communication

ACE-SXE USB communication is 2.0 compliant.

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

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

5.1.1. Typical USB Setup

The ACE-SXE 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.

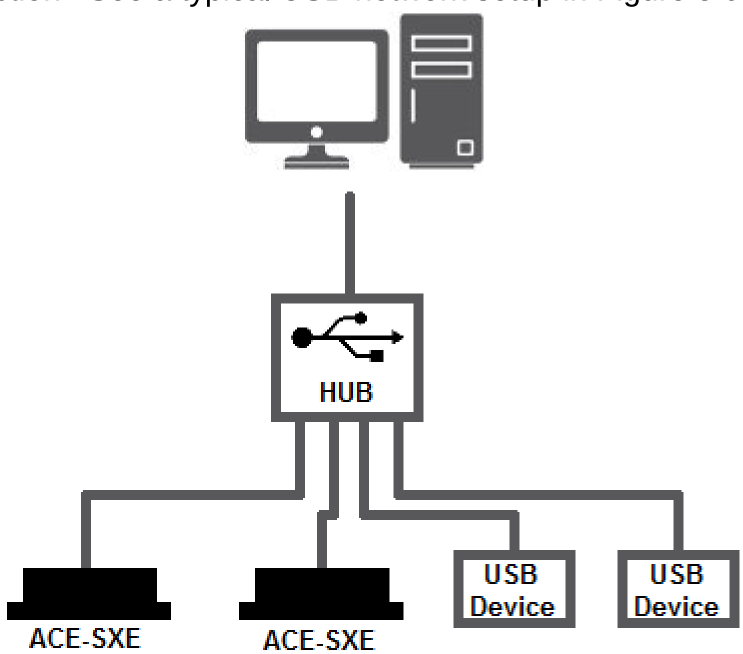


Figure 5.0

5.1.2. USB Communication API

Communication between the PC and ACE-SXE 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-SXE.

Typical communication transaction time between PC and ACE-SXE 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. Serial Communication

The ACE-SXE has the ability to communicate over an RS-485 interface using an ASCII protocol. An RS-485 serial port on the PC or PLC can be used to communicate with the ACE-SXE. A USB to RS-485 converter can also be used.

5.2.1. Typical RS-485 Setup

A typical RS-485 network is shown in Figure 5.2. Several techniques can be used to increase the robustness of an RS-485 network. Please see Section 5.2.4 for details.

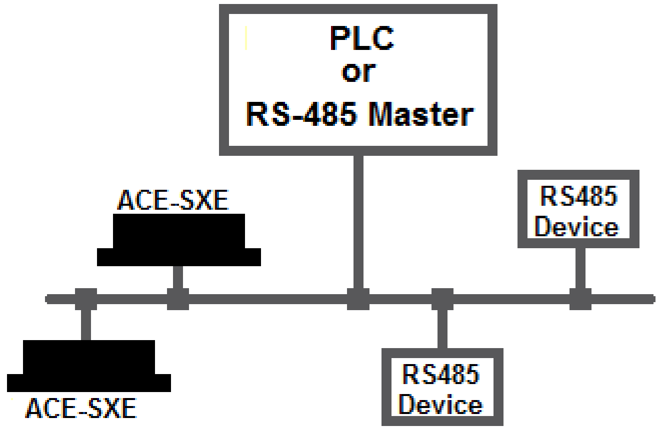


Figure 5.2

5.2.2. Communication Port Settings

The ACE-SXE has the communication port settings shown in table 5.0.

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

Table 5.0

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

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

Table 5.1

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

By default, the ACE-SXE has a baud rate setting of 9600 bps.

5.2.3. ASCII Protocol

The following ASCII protocol should be used for sending commands and receiving replies from the ACE-SXE. Details on valid ASCII commands can be found in section 8.

The address ‘00’ is reserved for broadcasting over an RS-485 bus. Any ASCII command prefixed by ‘@00’ will be processed by all ACE-SXE modules on the RS-485 bus.

Sending Command

ASCII command string in the format of

@[DeviceName][ASCII Command][CR]

[CR] character has ASCII code 13.

Receiving Reply

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

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

This response string type can be achieved by sending the command **RT=0**.

Examples:

For querying the encoder position

Send: @01EX[CR]

Reply: 1000[CR]

For jogging the motor in the positive direction

Send: @01J+[CR]

Reply: OK[CR]

Format 2: #[DeviceName][Response][CR]

This response string type can be achieved by sending the command **RT=1**.

Examples:

For querying the encoder position

Send: @01EX[CR]

Reply: #011000[CR]

For jogging the motor in the positive direction

Send: @01J+[CR]

Reply: #01OK[CR]

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

5.2.4. Modbus-RTU Protocol

The Modbus-RTU protocol runs over the RS-485 communication medium. The Arcus implementation of Modbus-RTU supports the holding register data type only. Coils, Discrete Inputs, and Input Registers are not supported.

Changing from ASCII to Modbus

By default, Arcus controllers are configured for ASCII protocol when using RS-485 communication. In order to enable Modbus-RTU communication, set **RSM=1**.

To write the values to the device's flash memory, use the **STORE** command. After a complete power cycle, the communication mode setting will be written to memory. Note that before a power cycle is completed, the settings will not take effect. This setting must be made through RS-485 ASCII.

Changing from Modbus to RS-485

Once your controller is in Modbus communication, it can no longer accept ASCII commands over RS-485. In order to change communication back to ASCII, you must write the value 1 into register **V30**. After writing to register V30, a power cycle must be performed before the setting takes effect. Note that the setting is automatically stored to flash when written.

Implementing Modbus-RTU

Only the general purpose variables V1-V30 of the controller are accessible via Modbus-RTU communication. By reading/updating these variables, the desired functionality can be achieved by referencing them within a running stand-alone program. In a typical application, the controller runs a stand-alone program while the Modbus master accesses variables over the Modbus-RTU protocol.

The following is a sample of such a stand-alone program.

```
EO=1                    ;* Enable the motor power
WHILE 1=1               ;* Forever loop
    GOSUB 1             ;* Set speed settings
    IF V1=1
        X1000           ;* If V1 is set to 1 then go to position 1000
        WAITX           ;* Wait for move to complete
    ELSEIF V1=2
        X3000           ;* If V1 is set to 2 then go to position 3000
        WAITX           ;* Wait for move to complete
    ELSEIF V1=3
        X0              ;* If V1 is set to 3 then go to position 0
        WAITX           ;* Wait for move to complete
    ENDIF
    V2 = PX             ;* Store pulse position in V2
ENDWHILE            ;* Go back to WHILE statement
```

```
END

SUB 1
        HSPD=V4                 ;* Set the high speed to V4 pulses/sec
        LSPD=V5                 ;* Set the low speed to V5 pulses/sec
        ACC=V6                  ;* Set the acceleration to V6 msec
END SUB
```

Modbus RTU Addressing

The Arcus implementation of Modbus-RTU supports the holding register data type only. Coils, Discrete Inputs, and Inputs Registers are not supported. The data model for a Modbus-RTU command can be found in table 5.4

Data Type Parameter	Size (Bits)	Data Type Description	Access	Address Range
03h	16	Holding Registers	Read/Write	[1-60]

Table 5.2

The holding registers in table 5.5 are available via Modbus-RTU on the ACE-SXE.

Address	Name	Description
1	V1_H	V1 Higher 16 bits [31:16]
2	V1_L	V1 Lower 16 bits [15:0]
3	V2_H	V2 Higher 16 bits [31:16]
4	V2_L	V2 Lower 16 bits [15:0]
5	V3_H	V3 Higher 16 bits [31:16]
6	V3_L	V3 Lower 16 bits [15:0]
7	V4_H	V4 Higher 16 bits [31:16]
8	V4_L	V4 Lower 16 bits [15:0]
9	V5_H	V5 Higher 16 bits [31:16]
10	V5_L	V5 Lower 16 bits [15:0]
11	V6_H	V6 Higher 16 bits [31:16]
12	V6_L	V6 Lower 16 bits [15:0]
13	V7_H	V7 Higher 16 bits [31:16]
14	V7_L	V7 Lower 16 bits [15:0]
15	V8_H	V8 Higher 16 bits [31:16]
16	V8_L	V8 Lower 16 bits [15:0]
17	V9_H	V9 Higher 16 bits [31:16]
18	V9_L	V9 Lower 16 bits [15:0]
19	V10_H	V10 Higher 16 bits [31:16]
20	V10_L	V10 Lower 16 bits [15:0]
21	V11_H	V11 Higher 16 bits [31:16]
22	V11_L	V11 Lower 16 bits [15:0]

23	V12_H	V12 Higher 16 bits [31:16]
24	V12_L	V12 Lower 16 bits [15:0]
25	V13_H	V13 Higher 16 bits [31:16]
26	V13_L	V13 Lower 16 bits [15:0]
27	V14_H	V14 Higher 16 bits [31:16]
28	V14_L	V14 Lower 16 bits [15:0]
29	V15_H	V15 Higher 16 bits [31:16]
30	V15_L	V15 Lower 16 bits [15:0]
31	V16_H	V16 Higher 16 bits [31:16]
32	V16_L	V16 Lower 16 bits [15:0]
33	V17_H	V17 Higher 16 bits [31:16]
34	V17_L	V17 Lower 16 bits [15:0]
35	V18_H	V18 Higher 16 bits [31:16]
36	V18_L	V18 Lower 16 bits [15:0]
37	V19_H	V19 Higher 16 bits [31:16]
38	V19_L	V19 Lower 16 bits [15:0]
39	V20_H	V20 Higher 16 bits [31:16]
40	V20_L	V20 Lower 16 bits [15:0]
41	V21_H	V21 Higher 16 bits [31:16]
42	V21_L	V21 Lower 16 bits [15:0]
43	V22_H	V22 Higher 16 bits [31:16]
44	V22_L	V22 Lower 16 bits [15:0]
45	V23_H	V23 Higher 16 bits [31:16]
46	V23_L	V23 Lower 16 bits [15:0]
47	V24_H	V24 Higher 16 bits [31:16]
48	V24_L	V24 Lower 16 bits [15:0]
49	V25_H	V25 Higher 16 bits [31:16]
50	V25_L	V25 Lower 16 bits [15:0]
51	V26_H	V26 Higher 16 bits [31:16]
52	V26_L	V26 Lower 16 bits [15:0]
53	V27_H	V27 Higher 16 bits [31:16]
54	V27_L	V27 Lower 16 bits [15:0]
55	V28_H	V28 Higher 16 bits [31:16]
56	V28_L	V28 Lower 16 bits [15:0]
57	V29_H	V29 Higher 16 bits [31:16]
58	V29_L	V29 Lower 16 bits [15:0]
*59	V30_H	V30 Higher 16 bits [31:16]
*60	V30_L	V30 Lower 16 bits [15:0]

Table 5.3

*V30 is reserved for communication mode setting. In order to change from Modbus communication to ASCII, set V30=1.

Variables V1-V30 are 32-bit twos-complement numbers. Since standard Modbus holding registers only hold 16 bits, the full 32-bit numbers are stored into two different registers. The lower addressed register represents the higher 16 bits of the 32-bit number, while higher addressed register represents the lower 16 bits of the 32-bit number. This is known as a swapped-long.

Example 1: Set **V1** to: -34930493 (0xFDEB00C3)

V1 holding register is found at address 1.
Set Holding Register Address 1 [bits 16:31] to 0xFDEB
Set Holding Register Address 2 [bits 0:15] to 0x00C3

Example 2: Read **V2** register

V2 holding register is found

Assuming that the values stored in these registers are:

Holding Register Address 3 [bits 16:31] = 0x0000
Holding Register Address 4 [bits 0:15] = 0x4933

When reading this value, the full 32-bit value is: +18739 (0x00004933)

5.2.5. RS-485 Communication Issues

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

Daisy Chaining
For a multi-drop RS-485 network, be sure that the network uses daisy-chain wiring. Figure 5.2 shows an example of a daisy chain network.

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

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

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

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

5.3. DIO Communication

DIO communication allows the user to store 16 different types (see Table 6.15) of moves into ACE-SXE flash memory. These moves can be referenced using the **select bits** (**DI3-DI6**) and triggered by using the **start bit** (**DI1**). Motion can be aborted by triggering the **abort/clear bit** (**DI2**). If an error occurs, it can also be cleared by triggering the **abort/clear bit** (**DI2**).

5.3.1. DIO Latency

Digital input response time to a trigger from **start bit (DI1)** is about 10 micro seconds. The actual amount of time from trigger to the beginning of the motion move depends on the command.

5.3.2. Setting Up DIO Parameters

In order to use this feature, you must first enable DIO mode (using **EDIO** command) as well as configure the appropriate DIO parameters via USB.

The DIO parameters are set using the **MP[X][Y]** command.

To view parameters, use command **MP[X][Y]**. To set values, use **MPXY=[value].**

X Parameter:

This parameter corresponds to the 2^4=16 selections that can be selected by DI3-DI6. This character must be written in hexadecimal (i.e. 0-F).

Y Parameter:

This parameter corresponds to the 5 different values that correspond to each DIO move. See the table below.

Note that some move operations do not need all 5 parameters. In this case, any extra move values that are entered will be ignored. For example, the STOP command does not need a "Target Position". Any value entered here will be ignored in this case.

Y Parameter

Y	Description
0	DIO Move reference (see Table 5.5)
1	Target Position
2	Low Speed
3	Acceleration
4	High Speed

Table 5.4

DIO Move List

Move Reference	Command
0	None
1	STOP
2	X[Target Position]
3	INC+ [Current Position + Target Position]
4	INC- [Current Position - Target Position]
5	J+
6	J-
7	H+
8	H-
9	EO=0
10	EO=1
11	ZH+
12	ZH-
13	SSPD[High Speed]
14	SCV=1
15	SCV=0
16	SL=1
17	SL=0
18	PX=[Target Position]
19	EX=[Target Position]
20	Z+
21	Z-
22	SSPDM=[High Speed]

Table 5.5

5.3.3. Examples

1. **Make DIO selection "0" corresponds to the J+ command with the following parameters:**
   Target Position = NA
   Low Speed = 100
   Acceleration = 300
   High Speed = 1000

   Send commands:

```
MP00 = 5          ` Set move reference for "0" to J+
MP01 = 0          ` Set target position to 0 (value will be ignored)
MP02 = 100        ` Set low speed to 100
MP03 = 300        ` Set acceleration to 300
MP04 = 1000       ` Set high speed to 1000
```

2. **Make DIO selection "0xF" corresponds to the X800 command with the following parameters:**
Target Position = 800
Low Speed = 500
Acceleration = 500
High Speed = 5000

Send commands:

MPF0 = 2	\` Set move reference for "F" to X[value]
MPF1 = 800	\` Set target position to 800
MPF2 = 500	\` Set low speed to 500
MPF3 = 500	\` Set acceleration to 500
MPF4 = 5000	\` Set high speed to 5000

5.3.4. Using DIO

1. First drive the **select bits** (**DI3-DI6).**
2. Then pull **start bit** (**DI1)** low to begin the move. (falling-edge triggered)
3. Trigger **abort/clear bit** (**DI2**) to abort motion command if desired.

Figure 5.7 shows a timing diagram using DIO control.

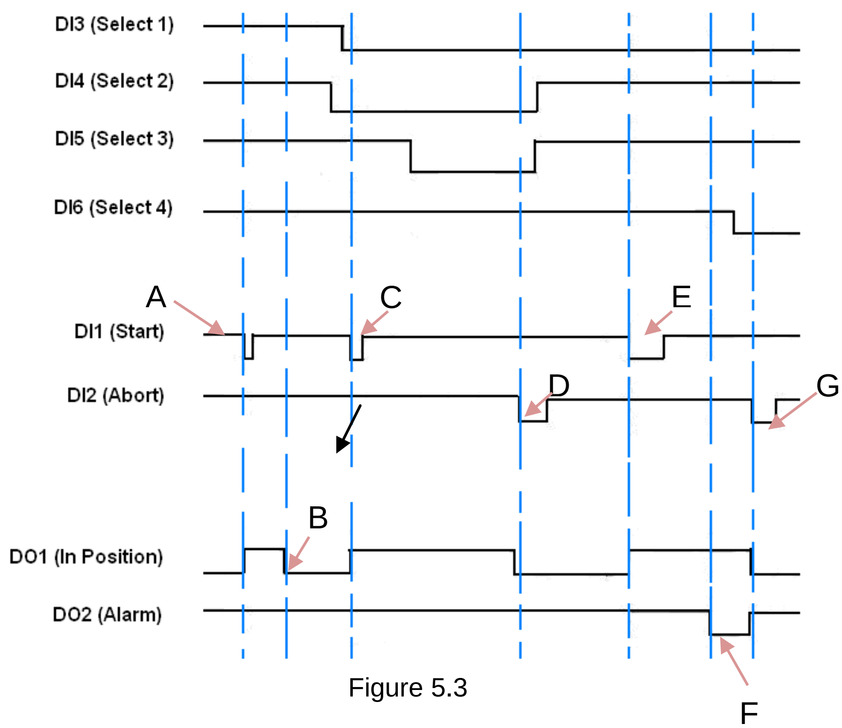


Figure 5.3

A) On falling edge of **Start**, motion command stored in memory location 0 (0000) is triggered. **In Position** turns off.

B) After motion command 0 (0000) is complete, **In Position** turn on.

C) On the falling edge of **Start**, motion command stored in memory location 12 (1100) is triggered. **In Position** turns off.

D) On falling edge of **Abort**, motion stops immediately. **In Position** turns on. Note: If move was an absolute move type, and target position was not reached, **In Position** will instead remain off.

E) On falling edge of **Start**, motion command stored in memory location 8 (1000) is triggered. **In Position** turns off.

F) Motion error occurs (i.e. limit error or StepNLoop error). **Alarm** turns on. **In Position** stays off. Controller is now in error state.

G) On falling edge of **Abort**, error state is cleared. **In Position** turns on.

Notes: DIO communication is not allowed while a standalone programming is running. If DIO communication is enabled while a standalone program begins execution, DIO communication will be automatically disabled.

Triggering the **start bit (DI1)** will not trigger a motion move if the **abort bit (DI2)** is on, or if the controller is in error state. If the controller is in error state, first clear the error by triggering the **abort/clear bit (DI2)**. The alarm bit output is on whenever there is either a SNL or limit error. The in position bit output is on whenever the motor is in position. Signals are active low.

5.4. Device Number

If multiple ACE-SXE devices are connected to the PC, each device should have a unique device number. This will allow the PC to differentiate between multiple controllers. This is applicable for both USB and RS-485 communication types. In order to make this change to an ACE-SXE, first store the desired number using the **DN** command. Note that this value must be within the range [SXE01 – SXE99].

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: **SXE01**

The ACE-SXE controllers allow the user to set the Modbus device number (used for addressing) from the range of [1-127]. In order to make this change, use the **DNM** command.

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: Modbus device number is set to **1**.
This setting can be made either through RS-485 ASCII or USB communication.

5.5. Windows GUI

The ACE-SXE 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-SXE.

6.1. Motion Profile

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

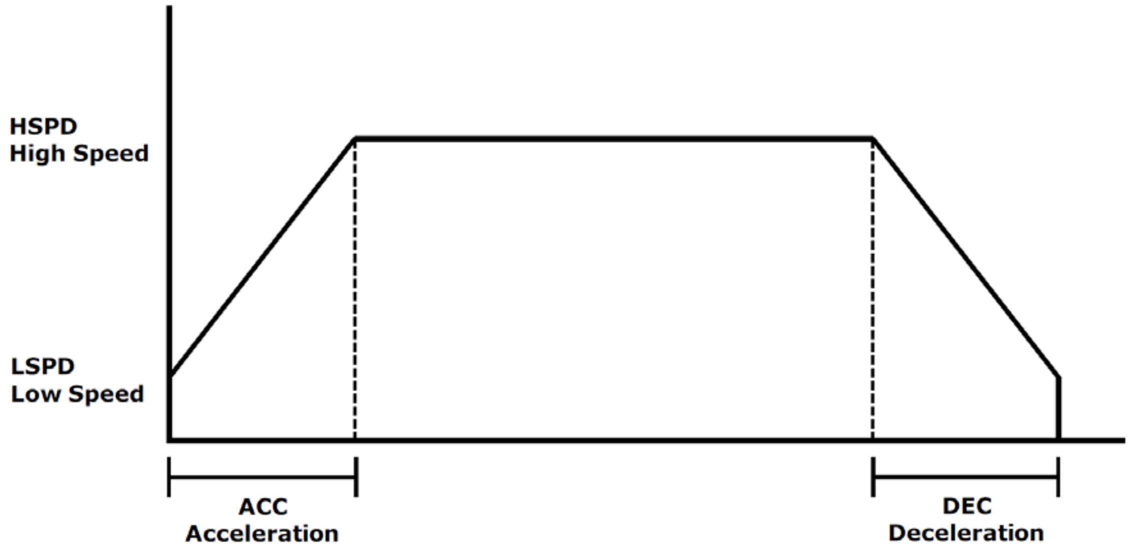


Figure 6.0

S-curve velocity profile can also be achieved by using the **SCV** command. Setting this command to 1 will enable S-curve. See Figure 6.1 for details.

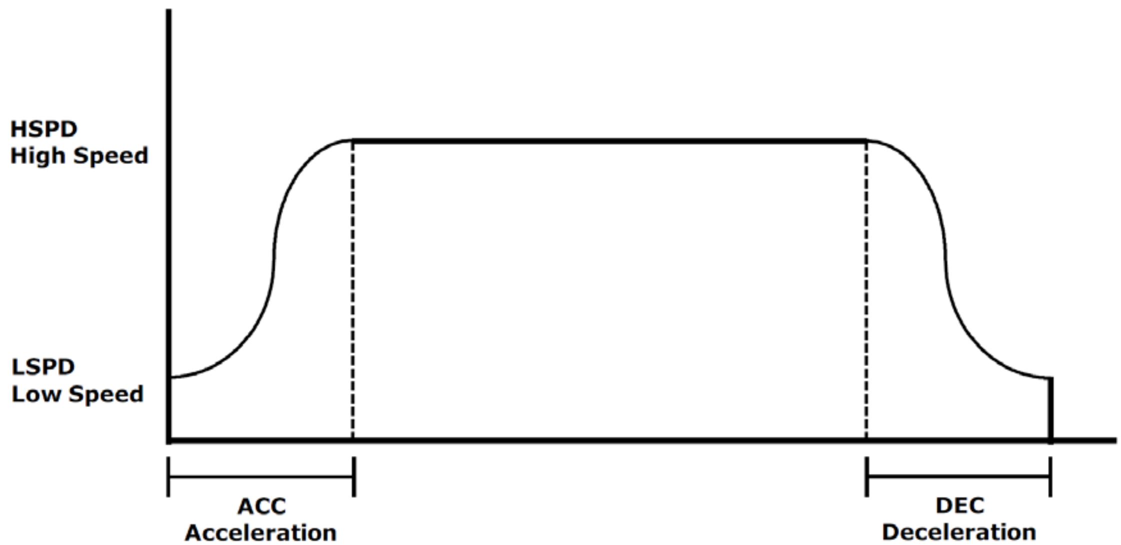


Figure 6.1

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). Use ASCII commands **HS** and **LS** to set/get high speed and low speed settings. Use the commands **HSPD** (high speed) and **LSPD** (low speed) in standalone mode.

Acceleration and deceleration times are in milliseconds. Use the **ACC** command to set/get acceleration values. Similarly, the **DEC** command can be used to set/get the deceleration value.

The **EDEC** command can be used if the acceleration and deceleration are symmetrical. Set this command to 0 to use the **ACC** command for both the acceleration and deceleration.

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

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

6.2. Pulse Speed

The current pulse rate can be read using the ASCII command **PS** or the standalone command **PS**. For units, see Table 6.0

Operation Mode	Speed Units
StepNLoop disabled	Pulse / sec
StepNLoop enabled	Encoder counts / sec

Table 6.0

Note that both the ASCII and standalone command return the current speed of the axis.

ASCII	**PS**
Standalone	**PS**

6.3. On-The-Fly Speed Change

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

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

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

Once the speed window has been set, an on-the-fly speed change can occur anytime the motor is in motion. The **SSPD[speed]** (ASCII) or **SSPDX[speed]** (standalone) command can be used to perform the actual speed change. For non on-the-fly speed change moves, set the speed window to 0 **(SSPDM=0)**.

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

6.4. Motor Position

The ACE-SXE 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.

Similarly, the ACE-SXE also has a 32 bit signed encoder position counter. Encoder position can be read using ASCII command **EX**, which returns the encoder position.

To manually set/get the encoder position of the axis, use the **EX=[value]** command.

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

When StepNLoop closed-loop control is disabled, the encoder counter command returns the encoder position and the pulse position command returns the step position. See Section 6.19 for details on the StepNLoop feature.

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

6.5. 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**	**-**

6.6. 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 8.1 for details on error responses.

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

6.7. 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**

6.8. Positional Moves

The ACE-SXE 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. Use **MM** command to read the current move mode.

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.

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**	**MM**	**X[target]**
Standalone	**ABS**	**INC**	**-**	**X[target]**

6.9. On-The-Fly Target Position Change

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

Note: If a **T** command is sent while the controller is not performing a target move, the command is not processed. Instead, an error response is returned.

ASCII	**T[value]**
Standalone	**-**

6.10. Homing

Home search routines involve moving the motor and using the home, limit, or Z-index inputs to determine the zero reference position.

The homing routines that involve a decelerated stop will result in a final position that is non-zero. 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 8.1 for details on error responses.

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

6.10.1. 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 6.2 shows the homing routine.

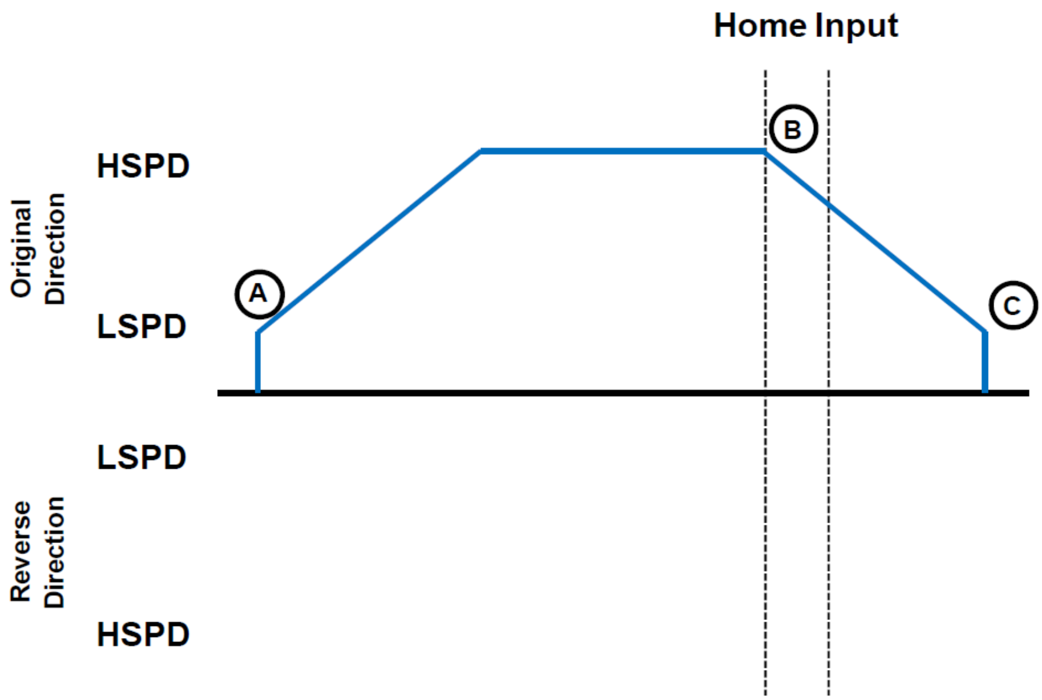


Figure 6.2

A. Issuing the command starts the motor from low speed and accelerates to high speed in search of the home input.
B. As soon as the home input is triggered, the position counter is reset to zero and the motor begins to decelerate to low speed. As the motor decelerates, the position counter keeps counting with reference to the zero position.
C. Once low speed is reached the motor stops. Although the position is non-zero, the zero position is maintained.

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

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

6.10.2. MODE 2 – Limit Input Only

Use the **L[+/-]** command for ASCII mode or the **LHOMEX[+/-]** command for standalone mode. Figure 6.3 shows the homing routine.

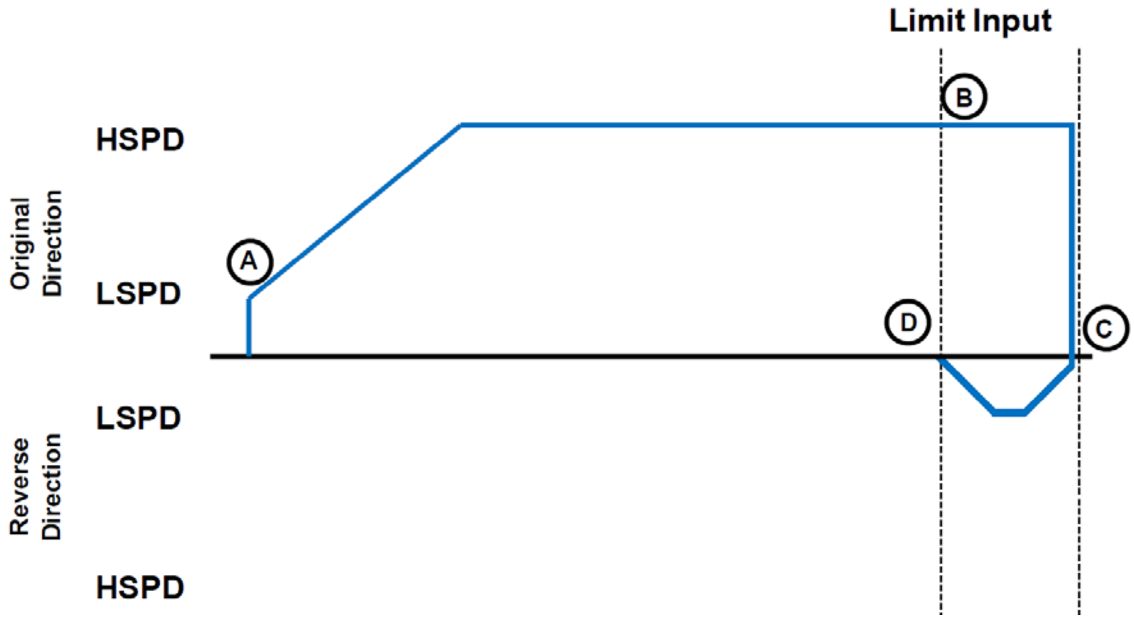


Figure 6.3

- **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[+/-]**

6.10.3. MODE 3 – Home Input and Z-Index

Use the **ZH[+/-]** command for ASCII mode or the **ZHOMEX[+/-]** command for standalone mode. In order to use this homing routine, an encoder must be used for the axis. Figure 6.4 shows the homing routine.

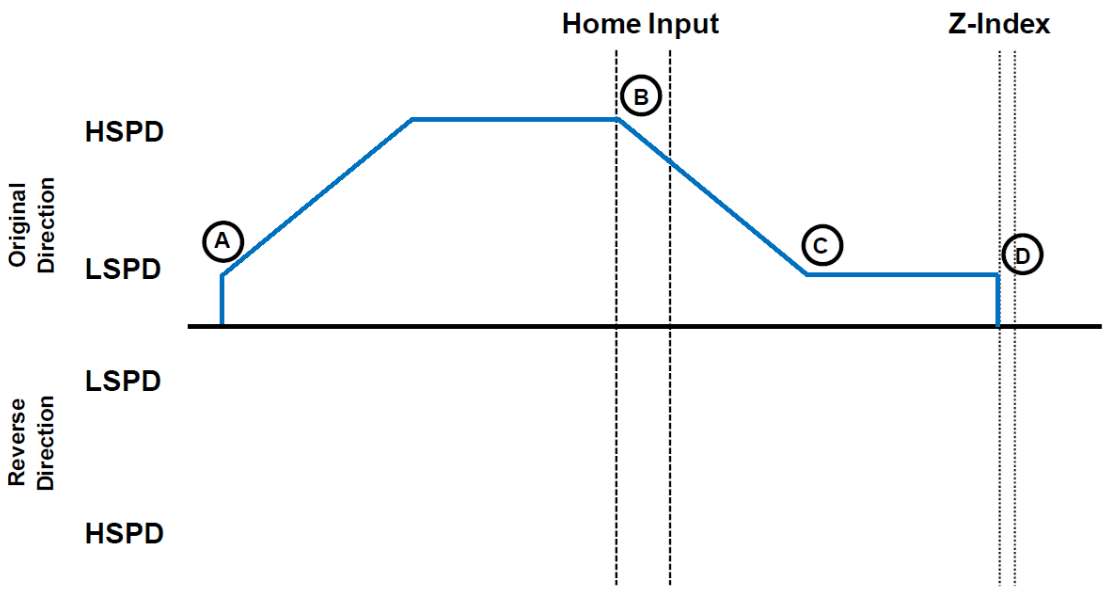


Figure 6.4

A. Issuing the command starts the motor from low speed and accelerates to high speed in search of the home input.
B. As soon as the home input is triggered, the motor decelerates to low speed
C. After the home input is triggered, the motor begins to search for the z-index pulse.
D. Once the z-index pulse is found, the motor immediately stops and the position is set to zero.

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

6.10.4. MODE 4 – Z-Index Only

Use the **Z[+/-]** command for ASCII mode or the **ZOMEX[+/-]** command for standalone mode. In order to use this homing routine, an encoder must be used on the axis. Figure 6.5 shows the homing routine.

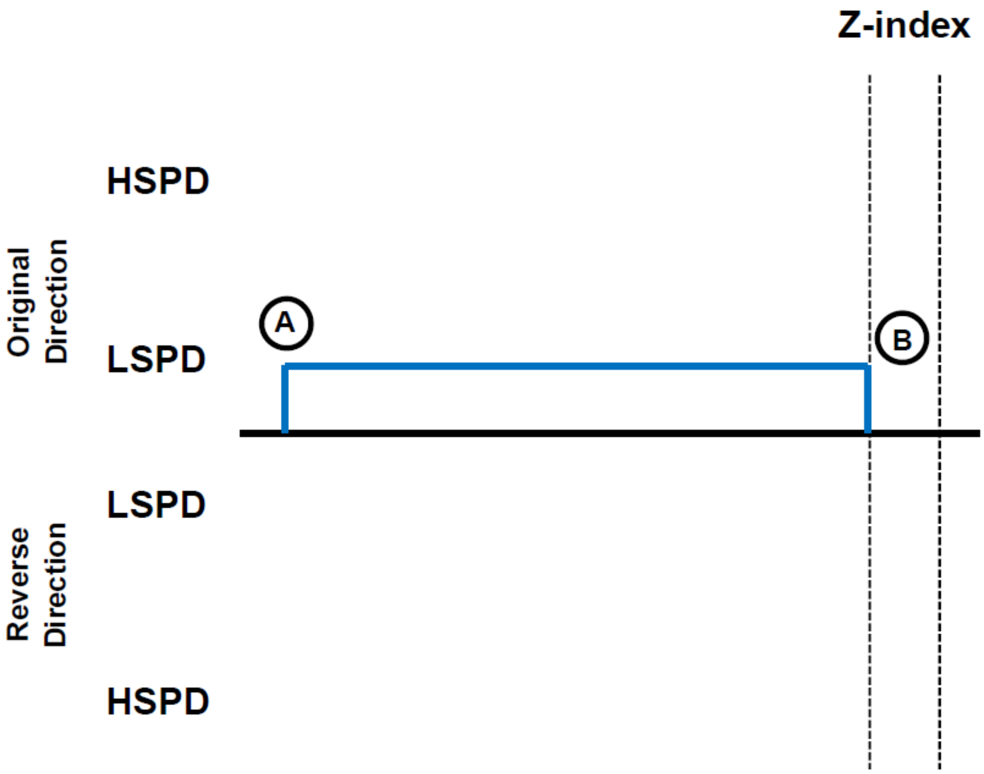


Figure 6.5

A. Issuing the command starts the motor at low speed. The motor will not use the high speed setting when performing this homing routine.

B. Once the z-index pulse is found, the motor stops and the position is set to zero.

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

6.10.5. MODE 5 – Home Input Only (High Speed and Low Speed)

Use the **HL[+/-]** command for ASCII mode and the **HLHOMEX[+/-]** for standalone mode. Figure 6.6 shows the homing routine.

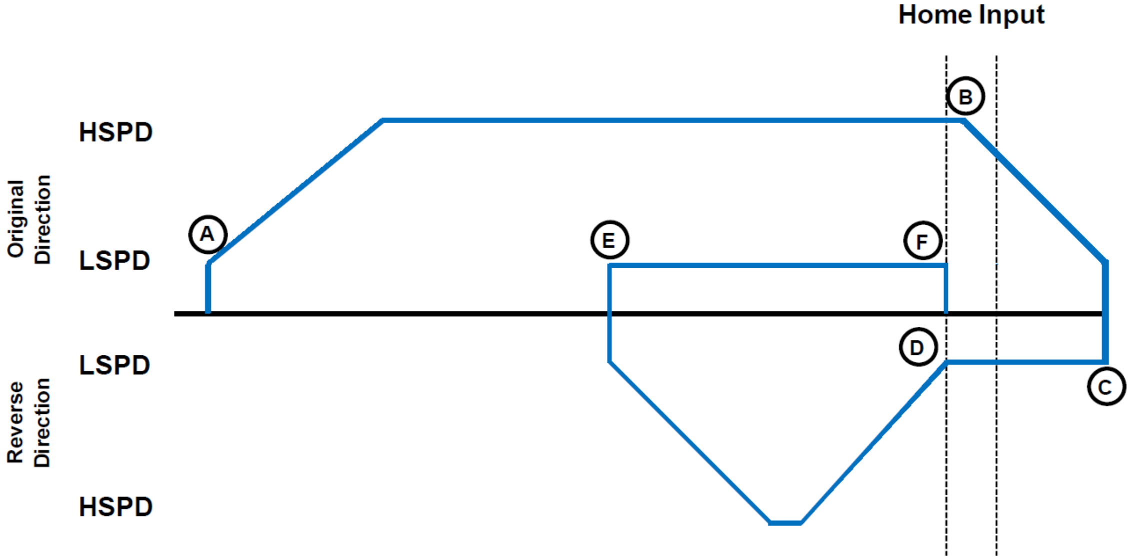


Figure 6.6

- **A.** Starts the motor from low speed and accelerates to high speed.
- **B.** As soon as the home input is triggered, the motor decelerates to low speed.
- **C.** The motor reverses direction at low speed until the home input is cleared.
- **D.** The motor will accelerate to high speed and then slow down to low speed to move the HCA (Home Correction Amount).
- **E.** Once the Home Correction Amount has been reached, the motor resumes movement in the original direction at low speed.
- **F.** When the home input is triggered, the motor immediately stops and the position and encoder counter are set to zero.

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

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

6.11. Limits And Alarm Switch Functions

Triggering the appropriate limit switch while the motor is moving will stop the motion immediately. For example, if the positive limit switch is triggered while moving in the positive direction, the motor will immediately stop and the motor status bit for positive limit error is set. The same will apply for the negative limit while moving in the negative direction.

If the alarm input for an axis is triggered during movement in either direction, the motor will immediately stop and the motor status bit for alarm error is set.

Once the limit or alarm error is set, use the **CLR** command to clear the error in ASCII mode or the **ECLEARX** command in standalone mode.

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

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

6.12. Motor Status

Motor status can be read anytime by reading the response to the **MST** command. The following is the bit representation of motor status:

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

Table 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.

6.13. Digital Inputs/Outputs
ACE-SXE module comes with 6 digital inputs and 2 digital outputs which can be used for DIO control. When DIO control is disabled, these can be used as intended. Enable/disable DIO control mode by using the **EDIO** command.

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

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

Bit	Description	Bit-Wise Command
0	Digital Input 1 (Start)	DI1
1	Digital Input 2 (Abort/Clear)	DI2
2	Digital Input 3 (Select 1)	DI3
3	Digital Input 4 (Select 2)	DI4
4	Digital Input 5 (Select 3)	DI5
5	Digital Input 6 (Select 4)	DI6

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

6.13.2. Digital Outputs
When DIO control is disabled, you can drive DO1 and DO2 by using the **DO** command. DO value must be within the range of 0-3.

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 (In Position)	DO1
1	Digital Output 2 (Alarm)	DO2

Table 6.3

When DIO control is enabled, DO1 and DO2 are used as In Position and Alarm outputs.

If digital output is turned on (i.e. the output is pulled to GND), the bit status is 1. Otherwise, the bit status is 0.

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]**	**-**

6.14. High Speed Latch Inputs

The ACE-SXE module provides a high speed position latch input. This input performs high speed position capture of both pulse and encoder positions but does not reset the pulse or encoder position counters. Use the **LT** command to enable and disable latch feature. To read the latch status, use **LTS** command. Following are return value description for **LTS** command.

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

Table 6.4

Note: When StepNLoop mode is enabled, the position value should be ignored. Once the latch is triggered, the triggered position can be retrieved using **LTP** (latched pulse position) and **LTE** (latched encoder position) commands.

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

6.15. Sync Outputs

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

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

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

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

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

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

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

Use **SYNT** to set the pulse width output time (ms). This parameter is only used if the synchronization condition is set to 1. Note the maximum pulse width is 10 ms. If this parameter is set to 0, the output pulse will depend on how long the encoder value is equal to the sync position.

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

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

When sync output feature is first enabled, the digital output turns on (i.e. the output is pulled to GND and DO2=1). Once sync output is triggered, the digital output turns off (i.e. the output is pulled to Vs and DO2=0).

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

6.16. Analog Inputs

Get the analog input status of the ACE-SXE by using the **AI1** and **AI2** commands. Return value is 0-5000 mV.

6.17. Joystick Control

Using analog input 1 (AI1), speed control using an analog input can be done. An analog input of 0V to 2.5V represents negative joystick direction and analog input of 2.5 to 5V represents positive joystick direction. 2.5V represents the zero joystick position. To set tolerance of the zero joystick position, use JV5 variable. For example, if JV5 is set to 100, then the zero range for X axis joystick control will be from 2.4V to 2.6V.

Maximum joystick speed is set using **JV1** variable

Summary of joystick control parameters

Command	Description
JV1	X axis Maximum Joystick Speed at 5V and 0V.
JV3	X axis Maximum speed change
JV5	X axis zero tolerance range for analog input

Table 6.5

Maximum speed change (**JV3**) variable affects the maximum amount that the speed can change due to change in analog input.

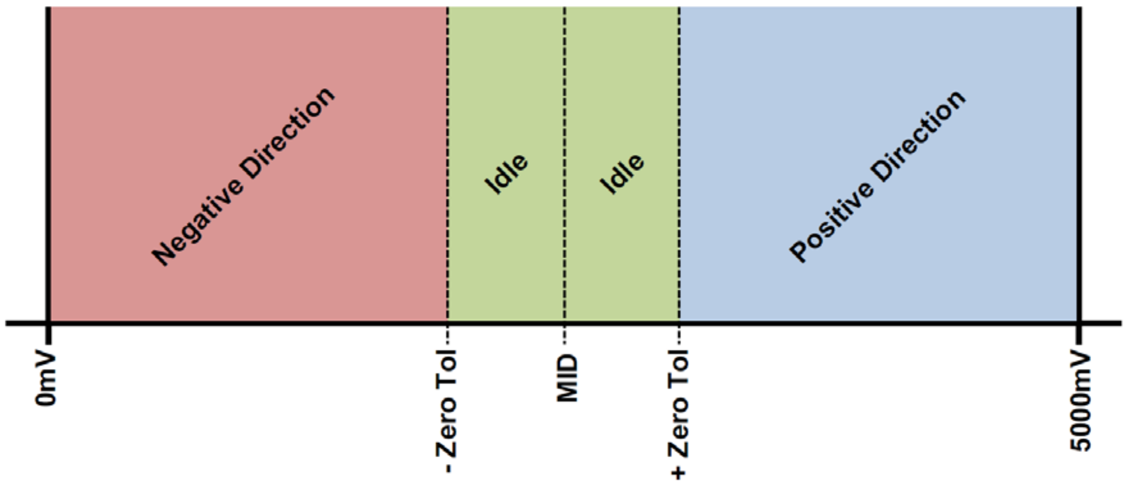


Figure 6.7

Joystick control also has soft limit control. Limits are broken down into positive inner and outer limits and negative inner and outer limits. When moving in positive direction, as soon as positive inner limit is crossed, the speed is reduced. If position crosses over the outer limit, the joystick speed is set to zero. Same goes for negative direction and negative limits.

Summary of joystick soft limit parameters

Command	Description
JL1	X axis Negative Outer Soft Limit
JL2	X axis Negative Inner Soft Limit
JL3	X axis Positive Inner Soft Limit
JL4	X axis Positive Outer Soft Limit

Table 6.6

To enable joystick control use **JO** command. The disable joystick control use **JF** command or **ABORT** command.

The behavior of the limits of the joystick control is explained by Figure 6.7.

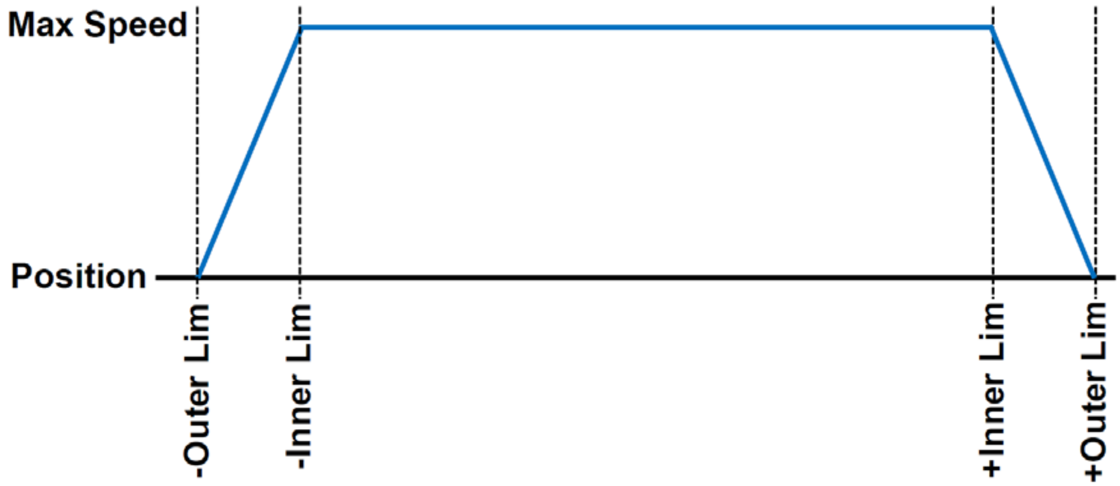


Figure 6.8

ASCII	**JO**	**JF**	**JV1**	**JV3**	**JL2**	**JL1**
Standalone	**JOYENA**	**-**	**JOYHSX**	**JOYDELX**	**JOYNIX**	**JOYNOX**

ASCII	**JL3**	**JL4**	**JV5**
Standalone	**JOYPIX**	**JOYPOX**	**JOYTOLX**

6.18. Polarity

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

Bit	Description	
0	Reserved	
1	Direction	
2	Reserved	
3	Reserved	
4	Limit	
5	Home	
6	Latch	
7	Z-channel index	
8,9	Encoder decoding	
	00	1X
	01	2X
	10	4X
10	Digital Output	
11	Digital Input	
12	Jump to line 0 on error*	
13	Enable Output	

Table 6.7

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

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

6.19. StepNLoop Closed Loop Control

ACE-SXE features a closed-loop position verification algorithm called StepNLoop (SNL). The algorithm requires the use of an incremental encoder.

SNL performs the following operations:

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

See Table 6.8 for a list of the SNL control parameters.

SNL Parameter	Description	Command
StepNLoop Ratio	*Ratio between motor pulses and encoder counts. This ratio will depend on the motor type, micro-stepping, encoder resolution and decoding multiplier. Value must be in the range [0.001, 999.999].	SLR
Tolerance	Maximum error between target and actual position that is considered "In Position". In this case, no correction is performed. Units are in encoder counts.	SLT
Error Range	Maximum error between target and actual position that is not considered a serious error. If the error exceeds this value, the motor will stop immediately and go into an error state.	SLE
Correction Attempt	Maximum number of correction tries that the controller will attempt before stopping and going into an error state.	SLA

Table 6.8

*A convenient way to find the StepNLoop ratio is to set EX=0, PX=0 and move the motor +1000 pulses. The ratio can be calculated by dividing 1000 by the resulting EX value. Note that the value must be positive. If it is not, then the direction polarity must be adjusted. See Table 6.7 for details.

To enable/disable the SNL feature use the **SL** command. To read the SNL status, use **SLS** command to read the status. See Table 6.9 for a list of the **SLS** return values.

Return Value	Description
0	Idle
1	Moving
2	Correcting
3	Stopping
4	Aborting
5	Jogging
6	Homing
7	Z-Homing
8	Correction range error. To clear this error, use CLR command.
9	Correction attempt error. To clear this error, use CLR command.
10	Stall Error. DX value has exceeded the correction range value. To clear this error, use CLR command.
11	Limit Error
12	N/A (i.e. SNL is not enabled)
13	Limit homing

Table 6.9

See Table 6.10 for SNL behavior within different scenarios.

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

Table 6.10

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

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

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

If DIO mode is on while SNL is enabled, DO1 is dedicated as the "In Position" output and DO2 is dedicated as the "Alarm" output. In order to use the digital outputs for general purpose, disable DIO by setting **EDIO=0**.

ASCII	**DX**	**SL**	**SLS**	**SLR**	**SLT**	**SLE**
Standalone	**-**	**SLX**	**SLSX**	**-**	**-**	**-**

6.20. Communication Time-Out Watchdog

ACE-SXE 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 9.2, for an example.

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

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

6.21. Standalone Program Specification

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

Standalone programs can be written to the ACE-SXE 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-SXE. Each line of written standalone code creates ~1-4 assembly lines of code after compilation.

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

6.21.1. Standalone Program Specification

Memory size: 1700 assembly lines. ~ 10.2KB.
Note: Each line of pre-compiled code equates to ~1-4 lines of assembly lines.

6.21.2. Standalone Control

The ACE-SXE 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 multithreading, please refer to Section 9. 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 6.11

6.21.3. Standalone Status

The **SASTAT[0-1]** command can be used to determine the current status of the specified standalone program. Table 6.12 details the return values of this command.

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

Table 6.12

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].

6.21.4. Standalone Subroutines

The ACE-SXE 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 9 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.

6.21.5. Error Handling

Subroutine 31 is designated for error handling. The standalone program will automatically jump to SUB 32 if an error occurs during standalone execution (i.e.

limit error or StepNLoop error). 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 9 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.

6.21.6. Standalone Variables

The ACE-SXE has 100 32-bit signed standalone variables available for general purpose use. They can be used to perform basic calculations and support integer operations. The **V[1-100]** 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 6.13

Variables V51 through V100 can be stored to flash memory using the **STORE** command. Variables V1-V50 will be initialized to zero on power up.

6.21.7. 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 6.14 for the bit assignment of the **SLOAD** setting.

Bit	Description
0	Standalone Program 0
1	Standalone Program 1

Table 6.14

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

Note: DIO communication is not allowed while a standalone programming is running. If DIO communication is enabled while a standalone program begins execution, DIO communication will be automatically disabled.

6.21.8. 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
```

6.22. Storing To Flash

The following items are stored to flash:

ASCII Command	Description
DB	Serial communication baud rate
DN	Device name
DNM	Modbus device number
DOBOOT	DO configuration at boot-up
EDEC	Unique deceleration enable
EDIO, MP	DIO communication settings
EOBOOT	EO configuration at boot-up
HCA	Home correction amount
IERR	Ignore limit error enable
JO, JF, JV1, JV3, JV5, JL1-4	Joystick settings
LCA	Limit correction amount
POL	Polarity settings
RSM	Modbus enable
RT	ASCII response type
RZ	Return to zero position after homing

SL, SLR, SLE, SLT, SLA	StepNLoop parameters
SLOAD	Standalone program run on boot-up parameter
TOC	Time-out counter reset value
V51-V100	Note that on boot-up, V1-V50 are reset to value 0

Table 6.15

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

7. Software Overview

The ACE-SXE has Windows compatible software that allows for USB or RS485 communication. Standalone programming, along with all other available features of the ACE-SXE, 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-SXE is connected to one of the available ports on the PC. To communicate over RS485, make sure that the ACE-SXE is connected to the COM port.

Startup the ACE-SXE GUI program and you will see the following screen in Figure 7.0. This will allow the user to search for all the motors that are currently connected to the USB network or RS485 bus.

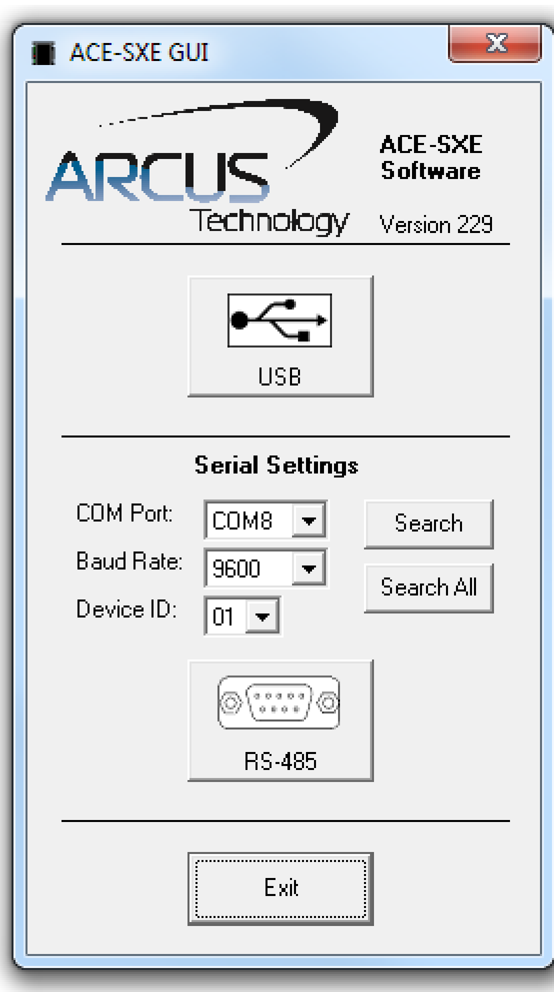


Figure 7.0

USB communication can be established by clicking the USB button. All ACE-SXE connected to the PC will be automatically detected and displayed.

For RS-485 communication, the first ACE-SXE connected to the PC can be found using the Search button. If there are multiple ACE-SXE devices connected to the PC over the RS-485 bus, the Search All button can be used to find them.

If the search fails, or a connection cannot be opened, check the following items:

- Check power supply to ACE-SXE. Allowable power is range is from 12VDC to 24VDC.
- Check communication wiring. The 485+ from ACE-SXE is connected to 485+ of the master and the 485- from ACE-SXE is connected to 485- of the master.
- Confirm that the device name is set correctly. Default factory device name setting is “sxe01”. If this name has been changed and stored to flash, enter the new name.

Once the correct serial settings have been determined, the RS-485 button can be used to open the software.

After a successful connection has been established, the screen below allows the user to choose between connected devices. To choose a particular device, select it and then press OK.

Figure 7.1

Important Note: In order to communicate with ACE-SXE via USB, the proper driver must first be installed. Before connecting the ACE-SXE 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 functions on the ACE-SXE. All features can be tested and verified.

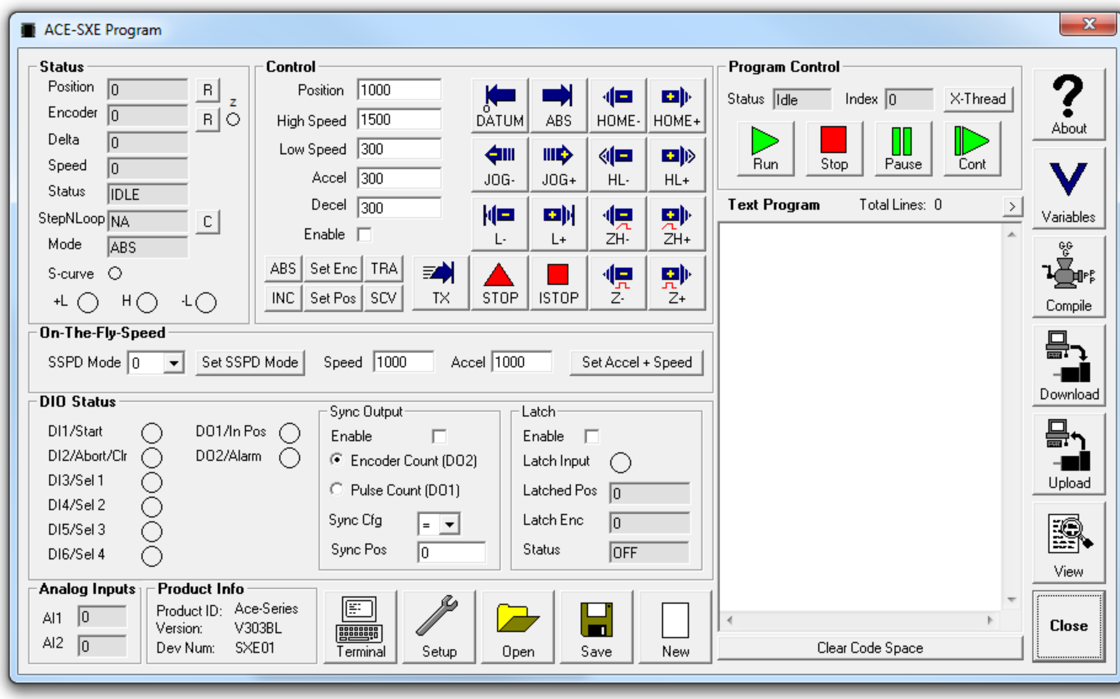


Figure 7.2

7.1.1. Status

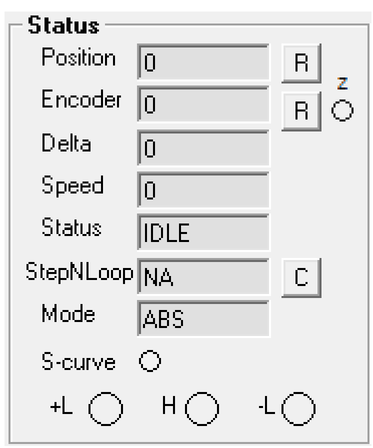


Figure 7.3

1. **Pulse Counter** – displays the current pulse position counter. When StepNLoop is enabled, this displays the Target position.
2. **Encoder Counter** – displays the current encoder position counter.
3. **Delta Counter** – valid only for StepNLoop. Displays the difference between the target position and the actual position.

4. **Speed** – displays the current pulse speed output rate. Value is in pulses/second. While the controller is in StepNLoop mode, this value shows encoder counts/second.
5. **Motion Status** – displays current motion status by displaying one of the following status:
    a. **IDLE**: motor is not moving
    b. **ACCEL**: motion is in acceleration
    c. **DECEL**: motion is in deceleration
    d. **CONST**: motion is in constant speed
    e. **-LIM ERR**: minus limit error
    f. **+LIM ERR**: plus limit error
6. **StepNLoop Status** – valid only when StepNLoop is enabled and displays current StepNLoop status by displaying one of the following:
    a. **NA**: StepNLoop is disabled
    b. **IDLE**: motor is not moving
    c. **MOVING**: target move is in progress
    d. **JOGGING**: jog move is in progress
    e. **HOMING**: homing is in progress
    f. **LHOMING**: limit homing in progress
    g. **Z-HOMING**: homing using Z-index channel in progress
    h. **ERR-STALL**: StepNLoop has stalled.
    i. **ERR-LIM**: plus/minus limit error
7. **Move Mode** – displays current move mode
    a. **ABS**: all the move commands by X[pos] command will be absolute moves
    b. **INC**: all the move commands by X[pos] command will be increment moves.
8. **S-curve Status** – Displays whether the moves are in trapezoidal or S-curve acceleration.
9. **Limit/Home Input Status** – Limit and Home input status.
10. **Reset StepNLoop Error** – When the StepNLoop status is in error, use this button to clear the StepNLoop error. StepNLoop status will return to IDLE after error is cleared.
11. **Reset Status Error** – When motion status is in error, use this button to clear the error.
12. **Reset Encoder Counter** – Encoder counter can be reset to zero using this button.
13. **Encoder Z Index Channel Status** – Encoder Z index channel status is displayed.
14. **Reset Pulse Counter** – Pulse counter can be reset to zero using this button.

7.1.2. Control

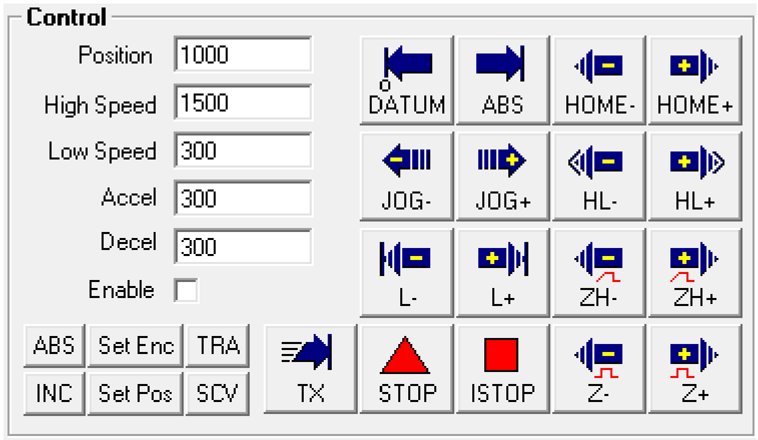


Figure 7.4

1. **Target Position/Speed/Accel**
   a. **Target Position:** use this to set the target position. For normal open loop mode, this position is the pulse counts and when StepNLoop is enabled this target position is in encoder counts.
   b. **High/Low Speed:** use this to set the speed of the move. For normal open loop mode, this value is in pulses/second and when StepNLoop is enabled this value is in encoder counts/second
   c. **Accel**: acceleration value in milliseconds
   d. **Decel:** deceleration value in milliseconds
2. **Enable Driver Power** – use this button to enable and disable the power to the micro-step driver.
3. **Select Move Mode** – use these buttons to select absolute or incremental move mode.
4. **Set Position** – use these buttons to set the encoder or pulse position to “Position” value
5. **Select Acceleration Mode** – use these buttons to select trapezoidal or S-curve acceleration mode.
6. **On-the-fly target change** – Change the target position on-the-fly
7. **Ramp Stop** – use this button to stop the motion with deceleration.
8. **Immediate Stop** – use this button to stop the motion immediately. We recommend that ramp stop be used whenever possible to reduce the impact to the motor and the system.
9. **Move back to zero** – use this to move the motor to the zero target position. When in absolute mode, the axis will move to zero position (zero encoder position when in StepNLoop and zero pulse position when in open loop).
10. **Perform Absolute Move** – use this to move the motor to the target position. When in absolute mode, the axis will move to the absolute target position. When in incremental mode, the axis will move incrementally.
11. **Jogging** – jog motor in either positive or negative direction

12. **Perform Homing** – Five different homing routines are available
    a. **HOME**: homing is done using only the home switch at high speed.
    b. **HL**: homing is done using the home switch at high speed and low speed
    c. **L**: homing is done using the limit switch
    d. **ZH**: homing is done using the home switch first and then the Z index channel of the encoder.
    e. **Z**: homing is done only using the Z index channel of the encoder.

7.1.3. On-The-Fly-Speed Control

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

Figure 7.5

1. **On-the-fly speed mode** – Before setting the controller into motion, set the SSPDM parameter. To see which value to use, see the on-the-fly speed change section.
2. **Set SSPDM** – Set the SSPDM parameter. Note that if an on-the-fly speed change operation is to be used, this parameter must be set before the controller starts motion.
3. **Desired Speed** – Once the “Set Speed” button is clicked, the speed will change on-the-fly to the desired speed.
4. **Desired Acc/Dec** – The acceleration/deceleration use for the on-the-fly speed change operation.
5. **Set SSPD[value]** – Start the on-the-fly speed operation.

7.1.4. Digital Input / Output

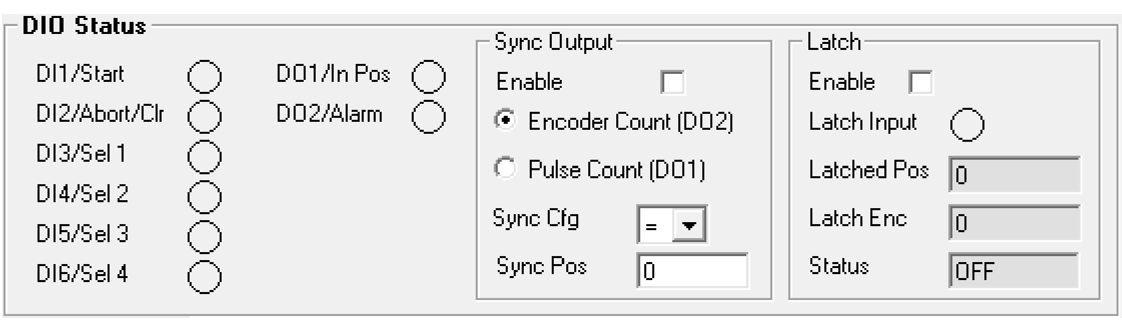


Figure 7.6

1. **Digital Input Status** – digital inputs can be used for DIO move control or as general purpose use. Refer to the setup screen to disable and enable the DIO move control.
2. **Digital Out Status and Control** – digital outs are used for StepNLoop or general purpose output use. When used as general purpose outputs, the outputs can be triggered by clicking on the circle.
3. **Sync Output** – DO2 can be triggered using the Sync operation.
4. **Latch** - encoder and pulse positions can be captured / latched with an input trigger.

7.1.5. Analog Inputs

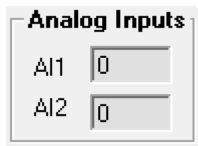


Figure 7.7

Two analog input channels are available for general purpose use or for joystick control use. The analog values are in mV.

7.1.6. Program File Control

Figure 7.8

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

7.1.7. Standalone Program Editor

Figure 7.9

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-SXE.
3. Use the ">" button to open a larger and easier to manage program editor.

7.1.8. Standalone Program Control

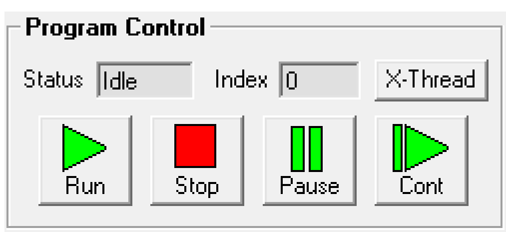


Figure 7.10

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. **Stop** – Stops the standalone program (PRG0).
6. **Pause** – Pauses the standalone program (PRG0).
7. **Cont.** – Continues the paused standalone program (PRG0).

7.1.9. Standalone Program Compile/Download/Upload/View

Figure 7.11

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

7.1.10. Setup

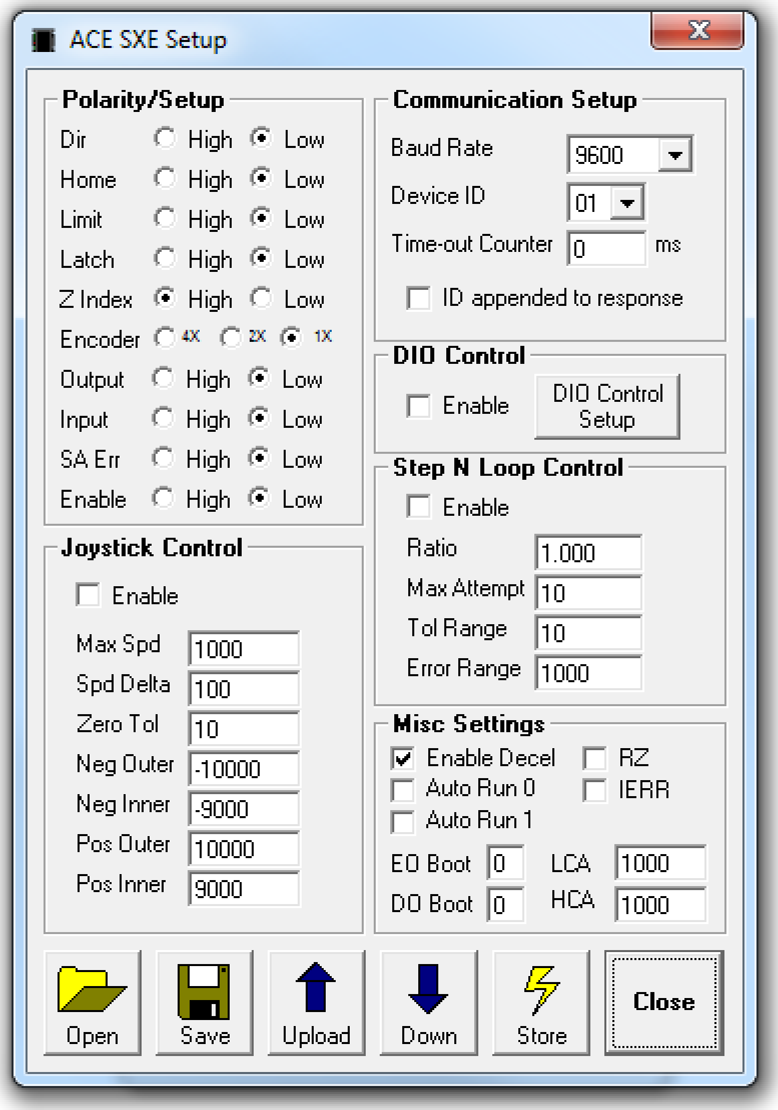


Figure 7.12

1. **Polarity Setup** – the following polarity parameters can be configured
   a. **Dir**: direction of the motion (clockwise or counter-clockwise)
   b. **Home**: home input polarity
   c. **Limit**: limit input polarity
   d. **Latch**: latch input polarity
   e. **Z-Index**: Encoder Z index channel
   f. **Encoder**: encoder multiplication factor can be configured as 1X, 2X, or 4X

g. **Output**: digital output polarity
h. **Input**: digital input polarity
i. **SA Err**: standalone error jump line.
j. **Low**: jump to previous line
k. **High**: jump to line 0
l. **Enable**: enable output polarity

2. **Joystick Control** – ACE-SXE allows joystick control using the analog input 1. See joystick control section for details of the joystick parameters.
3. **Communication Setup**
   a. RS-485 communication baud rate can be selected to support different communication speeds.
   b. Device ID configuration allows multiple devices on the RS-485 or USB communication network.
   c. Time-out counter is a watch-dog timer for communication (ms).
   d. ID append to response is used by RS-485 communication for adding the device ID to any response.
4. **DIO Control** – Digital IO motion control allows motion profiles to be triggered through the digital inputs. See DIO motion control section for details. The following dialog box is shown for the DIO motion control.

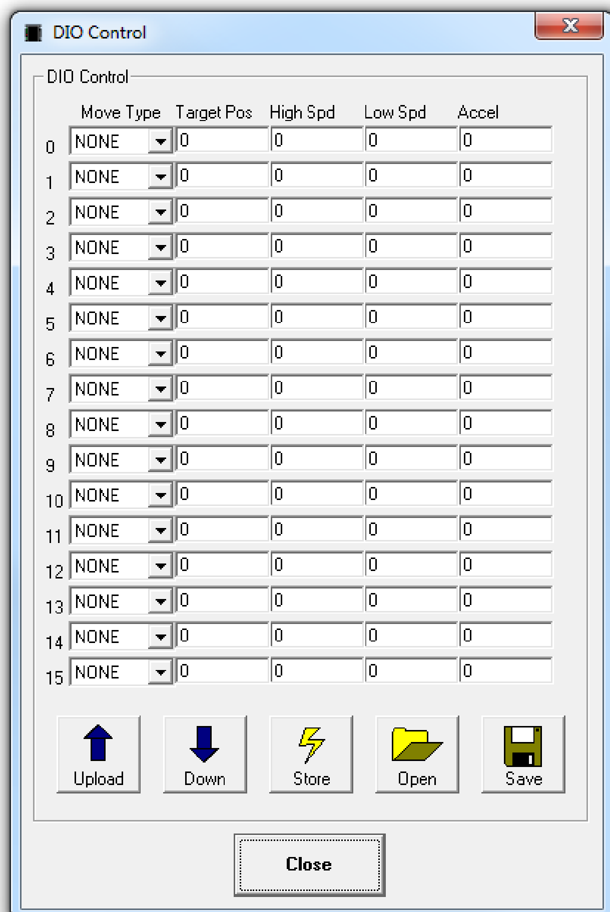
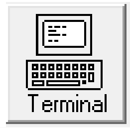


Figure 7.13

5. **StepNLoop Control** – Using the encoder input, StepNLoop control allows closed loop position verification and correction for the moves. See StepNLoop control section for details.
6. **Misc. Settings**
    a. **Enable Decel**: Allow for unique deceleration and acceleration values.
    b. **Auto Run 0**: Run standalone program 0 on boot-up.
    c. **Auto Run 1**: Run standalone program 1 on boot-up.
    d. **RZ**: Return to zero position after homing routines
    e. **IERR**: Ignore limit error
    f. **EO Boot**: Configure enable output boot-up state
    g. **DO Boot**: Configure digital output boot-up state
    h. **LCA**: Set limit correction amount
    i. **HCA**: Set home correction amount
7. **Open/Save** – Configuration values can be saved to a file and read from a file.
8. **Upload/Download** – Configuration values can be uploaded and downloaded. Note that if the configuration values are changed, they need to be downloaded to take effect.
9. **Store** – The downloaded parameters can be permanently stored on the non-volatile memory.

7.1.11. Terminal



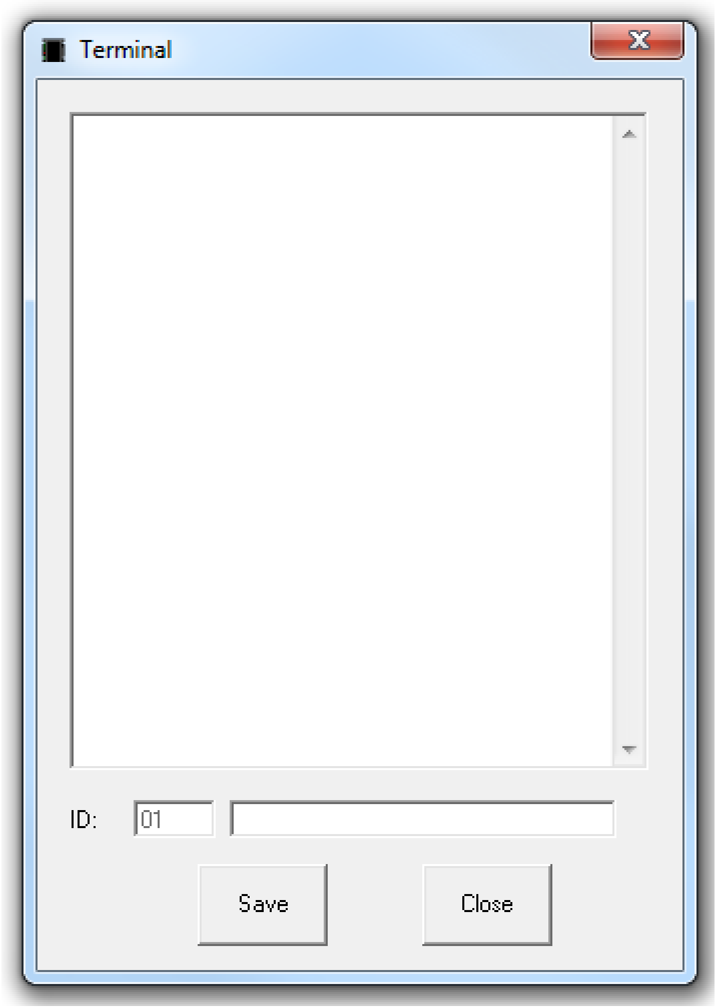


Figure 7.14

Terminal dialog box allows manual testing of the commands from a terminal screen as shown in figure 7.14.

7.1.12. Variable Status

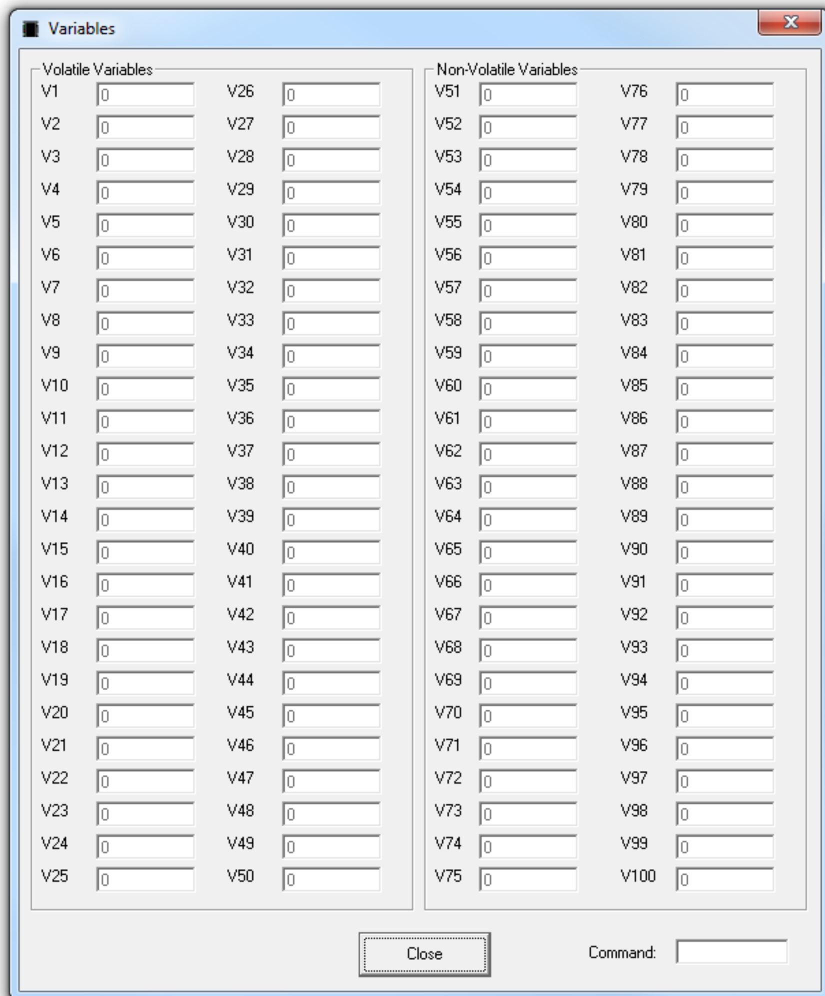


Figure 7.15

View the status of variables 1 – 100. Note that this window is read-only.

1. **Command: –** To write to a variable, use V[1-100] = [value].

7.1.13. Product Information

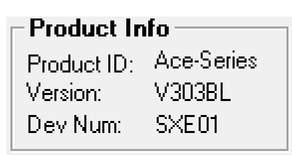


Figure 7.16

Product information, firmware version, and device number are displayed. Device number can be changed from the setup screen to support multiple devices via USB or RS-485 communication.

8. 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 9 for details regarding standalone commands.

ACE-SXE 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.

For **USB communication**, send commands identical to the ones in the following table.

For **RS-485 ASCII communication**, append "@XX" to the command before sending, where "XX" is the device number. Ex: To send the "J+" command to device number 05, send the following: **"@05J+"**

8.1. ASCII Command Set

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.	milliseconds
ACC=[Value]	Sets acceleration value in milliseconds. Example: ACC=300	OK
AI1, AI2	Get analog input status (0-5000 mV)	millivolts
CLR	Clears limit error as well as StepNLoop error	OK
DB	Return the current baud rate of the device	See Table 5.1
DB=[Value]	Set the baud rate of the device	OK
DEC	Get deceleration value in milliseconds. Only used if EDEC=1	milliseconds
DEC=[Value]	Set deceleration value in milliseconds. Only used if EDEC=1	OK
DI	Return status of digital inputs	See Table 6.2
DI[1-6]	Get individual bit status of digital inputs	0,1
DO	Return status of digital outputs	2-bit number
DO=[Value]	Set digital output 2 bit number. Digital output is writable only if DIO is disabled.	OK
DO[1-2]	Get individual bit status of digital outputs	See Table 6.3
DO[1-2]=[Value]	Set individual bit status of digital outputs	OK
DOBOOT	Get DO boot-up state	OK
DOBOOT=[Value]	Set DO boot-up state	OK
DN	Get device name	[SXE01-SXE99]
DN=[Value]	Set device name	OK
DNM	Get Modbus device number	1-127
DNM=[Value]	Set Modbus device number	OK

DX	Returns the delta value during StepNLoop control	28-bit number
EDEC	Get unique deceleration enable	0 or 1
EDEC=[Value]	Set unique deceleration enable	OK
EDIO	Returns DIO mode status	1 – DIO enabled 0 – DIO disabled
EDIO=[0 or 1]	Enables (value 1) or disable (value 0) DIO communication	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
EX	Returns current encoder counter value	28-bit number
EX=[Value]	Sets the current encoder counter value	OK
GS[0-31]	Call a subroutine that has been previously stored to flash memory	OK
HSPD	Returns High Speed Setting	PPS
HSPD=[Value]	Sets High Speed.	OK
H+	Homes the motor in positive direction	OK
H-	Homes the motor in negative direction	OK
HCA	Returns the home correction amount	28-bit number
HCA=[Value]	Sets the home correction amount.	OK
HL+	Homes the motor in positive direction (with low speed)	OK
HL-	Homes the motor in negative direction (with low speed)	OK
ID	Returns product ID	Ace-Series-SXE
IERR	Get ignore limit error enable	0 or 1
IERR=[Value]	Set ignore limit error enable	OK
INC	Set move mode to incremental	OK
J+	Jogs the motor in positive direction	OK
J-	Jogs the motor in negative direction	OK
JF	Disable joystick control for analog input 1	OK
JV1	Get max speed for joystick control	28-bit number
JV1=[Value]	Set max speed for joystick control	OK
JV3	Get max speed delta for joystick control	28-bit number
JV3=[Value]	Set max speed delta for joystick control	OK
JV5	Get zero speed tolerance for joystick control	28-bit number
JV5=[Value]	Set zero speed tolerance for joystick control	OK
JL1	Get negative outer limit for joystick control	28-bit number
JL1=[Value]	Set negative outer limit for joystick control	OK
JL2	Get negative inner limit for joystick control	28-bit number
JL2=[Value]	Set negative inner limit for joystick control	OK
JL3	Get positive inner limit for joystick control	28-bit number
JL3=[Value]	Set positive inner limit for joystick control	OK
JL4	Get positive outer limit for joystick control	28-bit number
JL4=[Value]	Set positive outer limit for joystick control	OK

JO	Enable joystick control for analog input 1	OK
JS	Get joystick enable status	0 – joystick operation off 1 – joystick operation on
L+	Limit homing in positive direction	OK
L-	Limit homing in negative direction	OK
LCA	Returns the limit correction amount	28-bit number
LCA=[Value]	Sets the limit correction amount.	OK
LSPD	Returns Low Speed Setting	PPS
LSPD=[Value]	Sets Low Speed	OK
LT=[0-1]	Enable or disable position latch feature	OK
LTE	Returns latched encoder position	28-bit number
LTP	Returns latched pulse position	28-bit number
LTS	Returns latch status.	See Table 6.4
MM	Get move mode status	0 – Absolute move mode 1 – Incremental move mode
MST	Returns motor status	See Table 6.1
MPXX	Get DIO parameter	
MPXX=[Value]	Set DIO parameter	OK
POL	Returns current polarity	See Table 6.7
POL=[value]	Sets polarity.	OK
PS	Returns current pulse speed	PPS
PX	Returns current position value	28-bit number
PX=[value]	Sets the current position value	OK
RSM	Get Modbus enable	0 or 1
RSM= [0-1]	Set Modbus enable	OK
RT	Get response type value	0 or 1
RT= [0-1]	Set response type value	OK
RZ	Get return zero enable. Used during homing	0 or 1
RZ=[0-1]	Set return zero enable. Used during homing	OK
SASTAT[0,1]	Get standalone program status 0 – Stopped 1 – Running 2 – Paused 4 – In Error	0-4
SA[LineNumber]	Get standalone line LineNumber: [0,1699]	
SA[LineNumber]=[Value]	Set standalone line LineNumber: [0,1699]	
SCV	Returns the s-curve control	0 or 1
SCV=[0-1]	Enable or disable s-curve. If disabled, trapezoidal acceleration/ deceleration will be used.	OK
SL	Returns StepNLoop enable status	0 – StepNLoop Off 1 – StepNLoop On

SL=[0 or 1]	Enable or disable StepNLoop Control	OK
SLA	Returns maximum number of StepNLoop control attempt	28-bit number
SLA=[value]	Sets maximum number of StepNLoop control attempt	OK
SLE	Returns StepNLoop correction range value.	28-bit number
SLE=[value]	Sets StepNLoop correction range value.	OK
SLR	Returns StepNLoop ratio value	[0.001 – 999.999]
SLR=[factor]	Sets StepNLoop ratio value. Must be in the range [0.001 – 999.999]	OK
SLS	Returns current status of StepNLoop control	See Table 6.9
SLT	Returns StepNLoop tolerance value	32-bit
SLT=[value]	Sets StepNLoop tolerance value.	OK
SLOAD	Returns RunOnBoot parameter	See Table 6.14
SLOAD=[0-1]	Set RunOnBoot parameter	See Table 6.14
SPC[0-1]	Get program counter for standalone program	[0-1699]
SR[0-1]=[Value]	Control standalone program: 0 – Stop standalone program 1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program	OK
SSPD[value]	On-the-fly speed change. In order to use this command, S-curve control must be disabled. Use SCV command to enable and disable s-curve acceleration/ deceleration control. Note that an "=" sign is not used for this command.	OK
SSPDM	Return on-the-fly speed change mode	[0-9]
SSPDM=[value]	Set on-the-fly speed change mode	OK
STOP	Stops the motor using deceleration if in motion.	OK
STORE	Store settings to flash	OK
SYNC	Read sync output configuration 1 – Trigger when encoder EQUALS position 2 – Trigger when encoder is LESS than position 3 – Trigger when encoder is GREATER than position	[1-3]
SYNC=[value]	Set sync output configuration 1 – Trigger when encoder EQUALS position 2 – Trigger when encoder is LESS than position 3 – Trigger when encoder is GREATER than position	OK
SYNF	Turn off sync output.	OK
SYNO	Turn on sync output.	OK
SYNP	Get trigger position.	28-bit number
SYNP=[value]	Set trigger position.	28-bit number
SYNS	To read the synchronization output status.	0, 1, 2
SYNT	Get pulse width time (ms). Only applicable is sync output configuration is set to 1.	Milliseconds
SYNT=[value]	Set pulse width time (ms). Only applicable if sync output configuration is set to 1. Max 30ms.	OK
T[value]	On-the-fly target change.	OK

TOC	Get time-out counter (ms).	32-bit number
TOC=[value]	Set time-out counter (ms).	OK
V[1-100]	Read variables 1-100	28-bit number
V[1-100]=[value]	Set variables 1-100	OK
VER	Get firmware version	VXXX
X[value]	Moves the motor to absolute position value using the HSPD, LSPD, and ACC values.	OK
Z+	Homes the motor in positive direction using the Z index encoder channel ONLY.	OK
Z-	Homes the motor in negative direction using the Z index encoder channel ONLY.	OK
ZH+	Homes the motor in positive direction using the home switch and then Z index encoder channel.	OK
ZH-	Homes the motor in negative direction using the home switch and then Z index encoder channel.	OK

Table 8.0

8.2. Error Codes

If an ASCII command cannot be processed by the ACE-SXE, 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-SXE
?ABS/INC is not in operation	T[] command is invalid because a target position move is not in operation
?Bad SSPD Command	SSPD move parameter is invalid
?DIO Enabled	Cannot control digital output because DIO mode is enabled
?Index out of Range	The index for the command sent to the controller is not valid.
?Moving	A move or position change command is sent while the ACE-SXE is outputting pulses.
?SA running	Cannot enable DIO mode because standalone is running
?SCV ON	Cannot perform SSPD move because s-curve is enabled
?Speed out of range	SSPD move parameter is out of the range of the SSPDM speed window.
?State Error	A move command is issued while the controller is in error state.
?Sub not Initialized	Call to a subroutine using the **GS** command is not valid because the specified subroutine has not been defined.

Table 8.1

9. 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 8 for details regarding ASCII commands.

9.1. 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
AI[1-2]	R	Get the analog input value.	IF AI2>2500 DO=1 ENDIF V2=AI1
DEC	R/W	Set/get the global deceleration setting. Unit is in milliseconds.	DEC=500 DEC=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.	IF DI=0 DO=1 ;Turn on DO1 ENDIF V2=DI
DI[1-6]	R	Get individual bit status of digital inputs. Will return [0,1].	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
EX	R/W	Set/get the current encoder position.	EX=1000 ;Set the enc to 1000 V1=EX ;Read current encoder

GOSUB [0-31]	-	Call a subroutine that has been previously stored to flash memory.	GOSUB 0 END
HLHOMEX[+/-]	W	Home the motor using the home input at low and high speeds in the specified direction.	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+
JOYENA	W	Set the joystick enable setting.	JOYENA=1 ;enable joystick
JOYHSX	W	Set the high speed setting for joystick control.	JOYHSX=2000
JOYDELX	W	Set the speed change delta for joystick control.	JOYDELX=100 JOYDELX=200
JOYNIX	W	Set negative inner limit for joystick control.	JOYNIX = -9000 JOYNIX=V1
JOYNOX	W	Set negative outer limit for joystick control.	JOYNOX = -10000 JOYNOX = V2
JOYPIX	W	Set positive inner limit for joystick control.	JOYPIX=9000 JOYPIX=V3
JOYPOX	W	Set positive outer limit for joystick control.	JOYPOX=10000 JOYPOX=V4
JOYTOLX	W	Set zero tolerance value for joystick control.	JOYTOLX=10 JOYTOLX=V5
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
LTEX	R	Get latch encoder value.	V7=LTEX
LTPX	R	Get latch position value.	V8=LTPX
LTSX	R	Get latch status.	V9=LTSX
LTX	W	Set latch enable. Range is [0-1]	LTX=0 LTX=V6
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

PS	R	Get the current pulse speed.	V10=PS
PX	R/W	Set/get the current motor position.	PX=1000 ;Set to X pos to 1000 V1=PX ;Read current X position
SCVX	R/W	Set/get the s-curve enable setting.	SCVX=1 ;enable s-curve V1=SCVX ;read s-curve
SLX	W	Enable/disable StepNLoop closed loop mode.	SLX=1 ;Enable StepNLoop SLX=0 ;Disable StepNLoop
SLSX	R	Get the current StepNLoop status.	V13=SLSX ;Set to status
SSPDMX	W	Set the SSPD mode. Must be done before move command.	SSPDMX=1 ;Set SSPD mode SSPDMX=V22 JOGX+ ;Jog the motor
SSPDX	W	Perform an on-the-fly speed change. SSPDMX must be set first.	JOGX+ ;Jog the motor DELAY=1000 ;Wait 1 second SSPDX=1000 ;Change speed SSPDX=V1
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
SYNCFGX	W	Set the sync output configuration.	SYNCFGX=2 ;Equal to setting
SYNOFFX	W	Disable the sync output configuration.	SYNOFFX ;Turn off sync
SYNONX	W	Enable the sync output configuration.	SYNONX ;Turn on sync
SYNPOSX	W	Set the sync output reference position.	SYNPOSX=1000 ;Set position SYNPOSX=V31 ;Set position
SYNSTATX	R	Get the current sync output status.	V1=SYNSTATX ;Get status
SYNTIMEX	W	Set the sync output pulse width time in milliseconds. Maximum of 10ms.	SYNTIMEX=5 ;Set to 5 ms
TOC	W	Sets the communication time-out parameter. Units are in milliseconds.	TOC=1000 ;1 second timeout
V[1-100]	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	WHILE 1=1 ;Infinite Loop DO=1 ;Turn on DO1

		should be used to close off a WHILE loop. Conditions [=,>,<,>=,<=,!=] are available.	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
ZHOMEX[+/-]	W	Home the motor using the home input and Z-index.	ZHOEMX+ ;Positive X HOME WAITX
ZOMEX[+/-]	W	Home the motor using the Z-index only.	ZOMEX- ;Negative X HOME WAITX

Table 9.0

9.2. Example Standalone Programs

9.2.1. 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
```

9.2.2. 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
```

9.2.3. 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
```

9.2.4. 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
```

9.2.5. 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
```

9.2.6. 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
```

9.2.7. 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
```

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

Table A.0

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

EPS = PPS / Step-N-Loop Ratio

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.

Max ACC = ((HSPD – LSPD) / δ) × 1000 [ms]

Figure A.0

Examples:

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

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

A.2. Acceleration/Deceleration Range – Positional Move

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

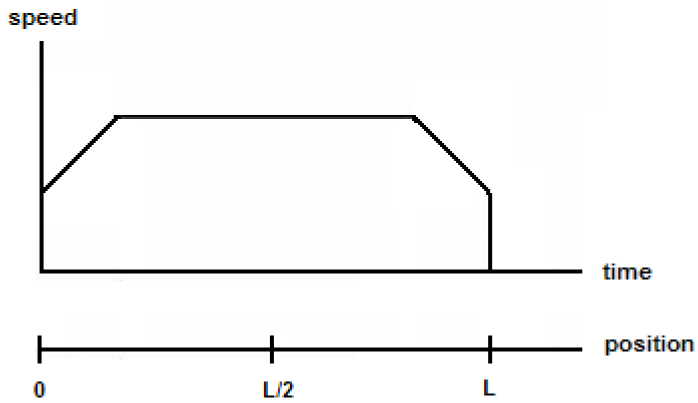


Figure A.1

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