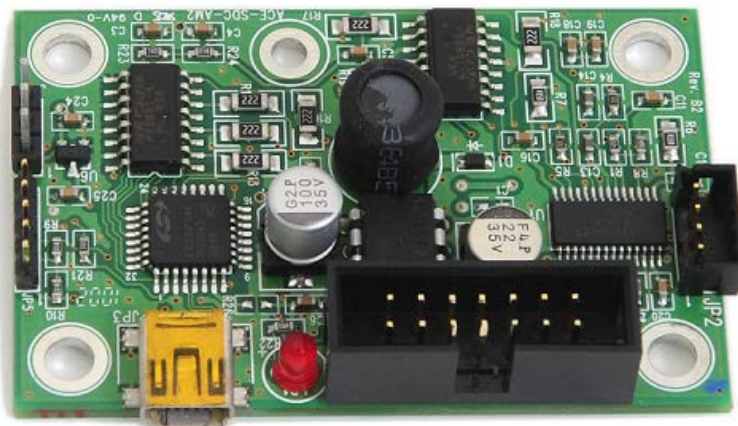


# ACE-SDC-V3S

## Microstep Driver + Standalone Controller USB 2.0 Communication



---

COPYRIGHT © 2015 ARCUS,  
ALL RIGHTS RESERVED

First edition, April 2008

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

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

**Revision History:**

- 1.00 – 1<sup>st</sup> Release
- 1.02 – 2<sup>nd</sup> Release
- 1.03 – 3<sup>rd</sup> Release
- 1.04 – 4<sup>th</sup> Release
- 1.05 – 5<sup>th</sup> Release
- 1.06 – 6<sup>th</sup> Release

**Firmware Compatibility:**

†V217BL

†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

<b>1. Introduction</b> .....	<b>5</b>
1.1. Features .....	5
<b>2. Electrical and Thermal Specifications</b> .....	<b>6</b>
<b>3. Dimensions</b> .....	<b>7</b>
<b>4. Connectivity</b> .....	<b>8</b>
4.1. JP1 3-Pin Connector Pin-outs .....	8
4.2. JP2 4-Pin Motor Connector Pin-outs .....	8
4.3. JP3 14-Pin IO Connector Pin-outs .....	9
4.4. Internal Interface Circuit Overview .....	10
4.5. Digital Inputs .....	11
4.6. Digital Outputs .....	12
4.7. Analog Input .....	12
<b>5. Stepper Motor Driver Overview</b> .....	<b>13</b>
5.1. Microstep .....	13
5.2. Driver Current .....	13
<b>6. Communication Interface</b> .....	<b>14</b>
6.1. USB Communication .....	14
6.1.1. Typical USB Setup .....	14
6.1.2. USB Communication API .....	14
6.1.3. USB Communication Issues .....	16
6.2. Device Number.....	16
6.3. Windows GUI .....	16
<b>7. General Operation Overview</b> .....	<b>17</b>
7.1. Motion Profile .....	17
7.2. On-The-Fly Speed Change .....	18
7.3. Motor Position .....	18
7.4. Motor Power .....	18
7.5. Jog Move .....	18
7.6. Stopping .....	18
7.7. Positional Moves .....	19
7.8. Homing .....	19
7.9. Limits Switch Function. ....	20
7.10. Motor Status .....	21
7.11. Digital Inputs / Outputs .....	21
7.11.1. Digital Inputs .....	21
7.11.2. Digital Outputs .....	22
7.12. Analog Speed Control .....	22
7.13. Polarity .....	22
7.14. Standalone Program Specification .....	23
7.14.1. Standalone Program Specification .....	23
7.14.2. Standalone Control .....	23
7.14.3. Standalone Status .....	23
7.14.4. Standalone Subroutines .....	24
7.14.5. Error Handling .....	24
7.14.6. Standalone Variables .....	24
7.14.7. Standalone Run On Boot-Up .....	25
7.14.8. WAIT Statement .....	25
7.15. Storing to Flash .....	26
<b>8. Software Overview</b> .....	<b>27</b>
8.1. Main Control Screen .....	28
8.1.1. Status .....	28
8.1.2. Control .....	29
8.1.3. Digital Input / Output .....	29

---

8.1.4. Program File Control.....	30
8.1.5. Standalone Program Editor .....	30
8.1.6. Standalone Program Control .....	30
8.1.7. Standalone Program Compile / Download / Upload /View .....	31
8.1.8. Terminal .....	32
8.1.9. Variable Status.....	33
8.1.10. Upload / Download / Store to Flash .....	33
8.1.11. Configuration.....	34
<b>9. ASCII Language Specification.....</b>	<b>35</b>
9.1. ASCII Command Set.....	35
9.2. Error Codes .....	37
<b>10. Standalone Language Specification.....</b>	<b>38</b>
10.1. Standalone Command Set .....	38
10.2. Example Standalone Programs .....	40
10.2.1. Standalone Example Program 1 – Single Thread .....	40
10.2.2. Standalone Example Program 2 – Single Thread .....	40
10.2.3. Standalone Example Program 3 – Single Thread .....	40
10.2.4. Standalone Example Program 4 – Single Thread .....	41
10.2.5. Standalone Example Program 5 – Single Thread .....	41
10.2.6. Standalone Example Program 6 – Single Thread .....	42
10.2.7. Standalone Example Program 7 – Multi Thread.....	43

---

## 1. Introduction

ACE-SDC-V3S is a single board cost-effective stepper driver and controller with USB 2.0 communication.

### 1.1. Features

- USB 2.0 Communication
- 12-24VDC voltage input
- Different configurable current settings from 0 mA to 2.0A peak current (100 mA resolution)
- Selectable Full, 1/2, 1/4, 16 microstep
- 6 Opto-isolated Digital inputs
- 2 Opto-isolated Digital outputs
- 1 Analog input
- Two different control modes:
  - USB Control – motion is done from PC using USB communication.
  - Analog Control – speed control is done from analog input.

For technical support contact: [support@arcus-technology.com](mailto:support@arcus-technology.com)

Or, contact your local distributor for technical support.

## 2. Electrical and Thermal Specifications

Parameter	Min	Max	Units
Main Power Input <sub>1</sub>	+12	+24	V
	-	2	A
Opto-supply Power Input	+12	+24	V
Digital Input Forward Diode Current	-	50	mA
Digital Output Emitter Current	-	50	mA
Analog Input Range	0	+3.3	V
Operating Temperature <sub>2</sub>	-	+85	°C

Table 2.0

<sub>1</sub>The supply current should match the driver current setting.

<sub>2</sub>Based on component ratings.

### 3. Dimensions

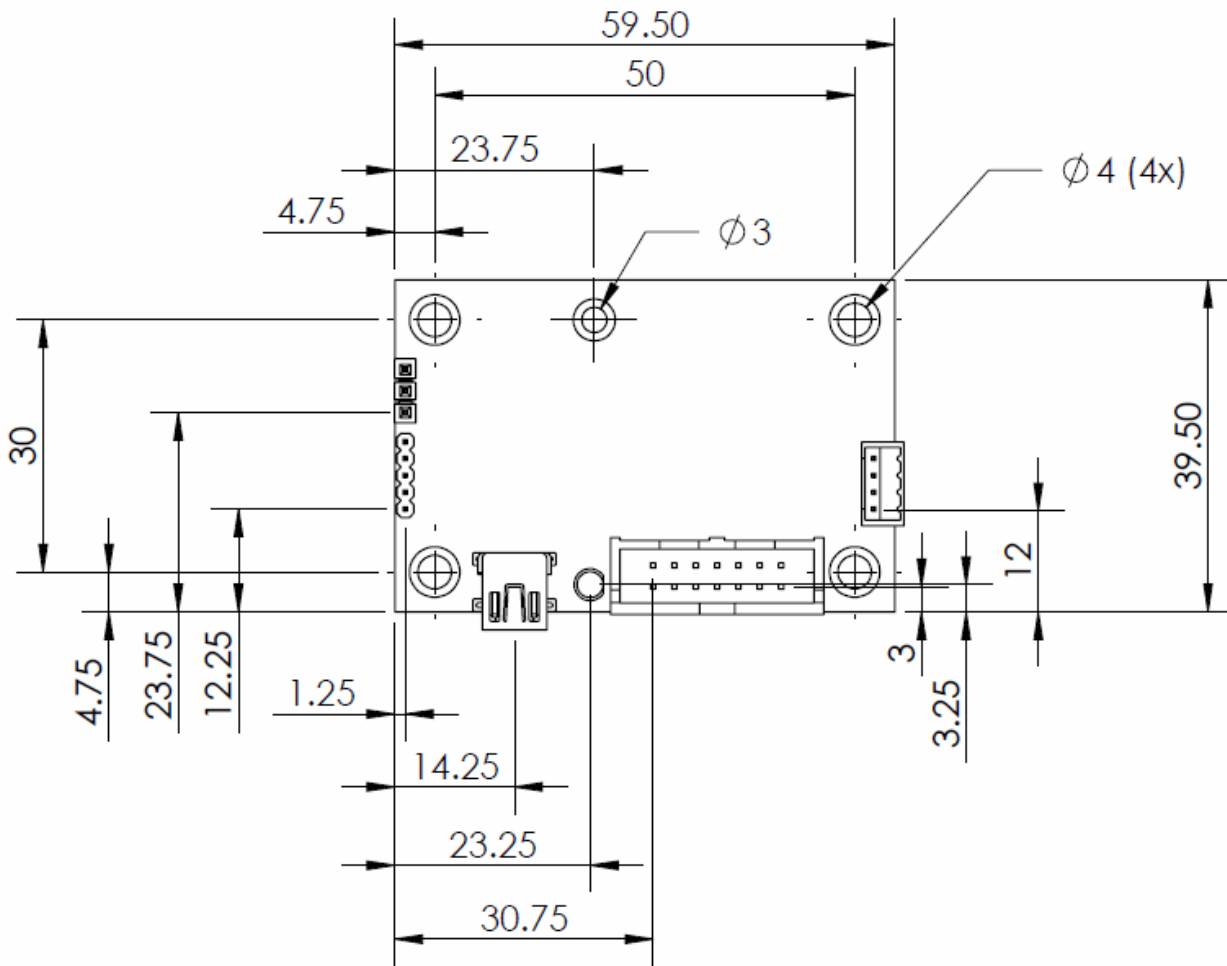


Figure 3.0

## 4. Connectivity

In order for ACE-SDC-V3S to operate, it must be supplied with +12VDC to +24VDC. Power pins as well as communication port pin outs are shown below.

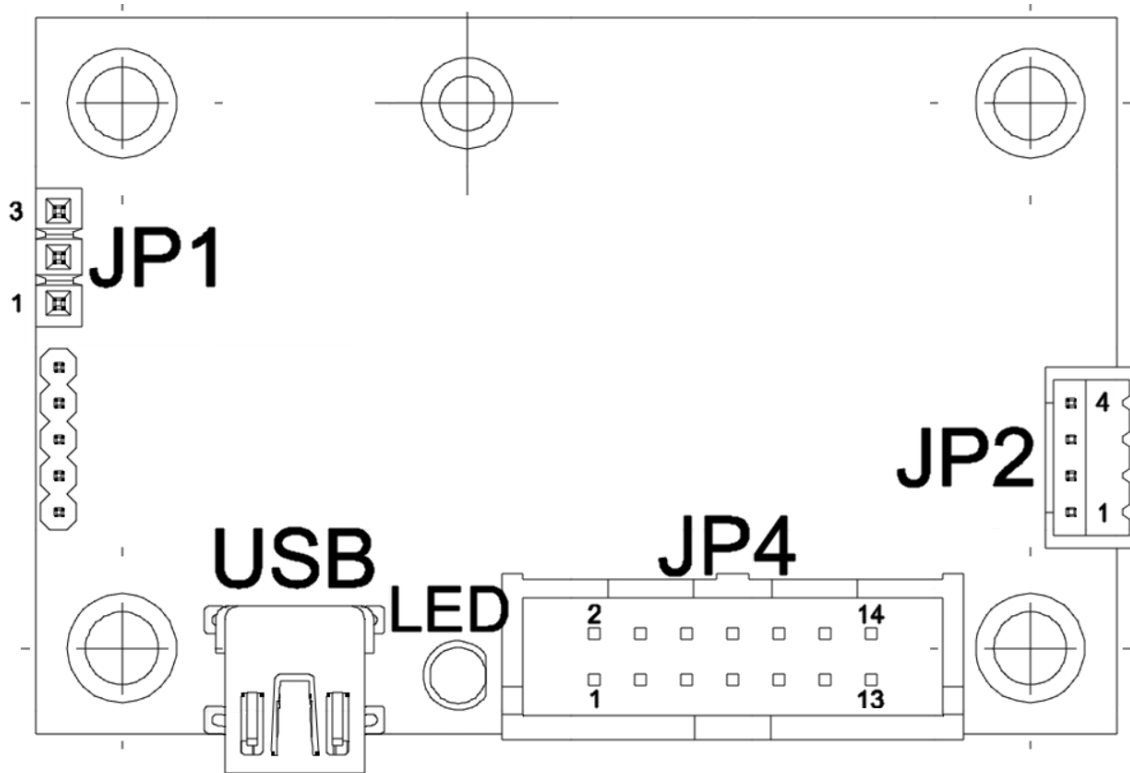


Figure 4.0

### 4.1. JP1 3-Pin Connector Pin-outs

Pin #	Name	In/Out	Description
1	3.3V Out	O	3.3V Output
2	Analog Input	I	Analog Signal Input used for Analog Speed Control (Section 7.12)
3	GND	O	Ground

Table 4.0

#### 3-pin Mating Connector Information

Mating Connector Description: Female 3 pin 0.1 inch single row  
Mating Connector Manufacturer: Tyco/AMP  
Mating Connector Manufacturer Part: 770602-3 (Connector)  
770666-1 (Pin)

### 4.2. JP2 4-Pin Motor Connector Pin-outs

Pin #	Name	In/Out	Description
1	A	O	Phase A of Bi-polar Step Motor



2	/A	○	Phase /A of Bi-polar Step Motor
3	B	○	Phase B of Bi-polar Step Motor
4	/B	○	Phase /B of Bi-polar Step Motor

Table 4.1

#### 4-pin Mating Connector Information

Mating Connector Description: Female 10 pin 2mm single row  
 Mating Connector Manufacturer: HIROSE  
 Mating Connector Manufacturer Part: DF3-4S-2C (4 pin female connector)  
 DF3-2428SC (female pin)

**Warning: Do not disconnect the motor connector while power is on. This can damage the product! Turn off the power before disconnecting the motor connector.**

#### 4.3. JP3 14-Pin IO Connector Pin-outs

Pin #	Name	In/Out	Description
1	OPTO	I	Opto-Supply Input (+12 to +24VDC)
2	DI1	I	Digital Input 1
3	DI2	I	Digital Input 2
4	DI3	I	Digital Input 3
5	DI4	I	Digital Input 4
6	DI5	I	Digital Input 5
7	DI6	I	Digital Input 6
8	DO1	O	Digital Output 1
9	DO2	O	Digital Output 2
10	NC	NC	No Connection
11	GND	I	Ground
12	PWR	I	Power (+12 to +24 VDC)
13	GND	I	Ground
14	PWR	I	Power (+12 to +24 VDC)

Table 4.2

#### 14-pin Mating Connector Information

Mating Connector Description: Female 14 pin 0.1 inch dual row  
 Mating Connector Manufacturer: AMP  
 Mating Connector Manufacturer Part: 1658621-2

## 4.4. Internal Interface Circuit Overview

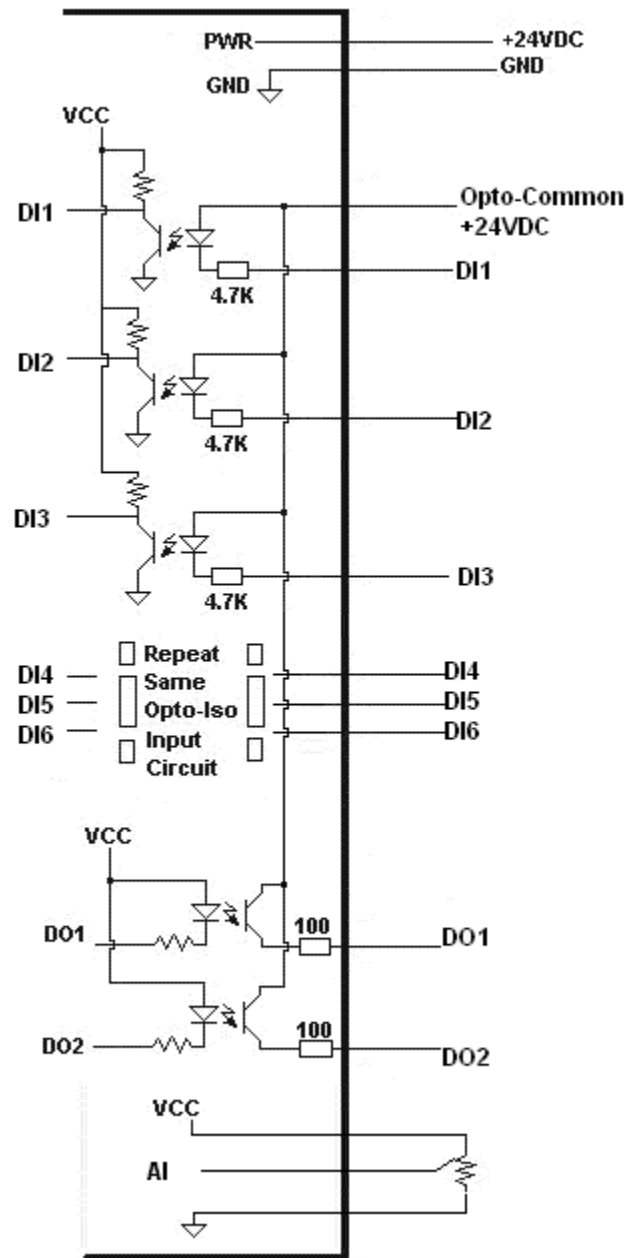


Figure 4.1

## 4.5. Digital Inputs

Figure 4.2 shows the detailed schematic of the opto-isolated general purpose digital inputs. All opto-isolated inputs are NPN type.

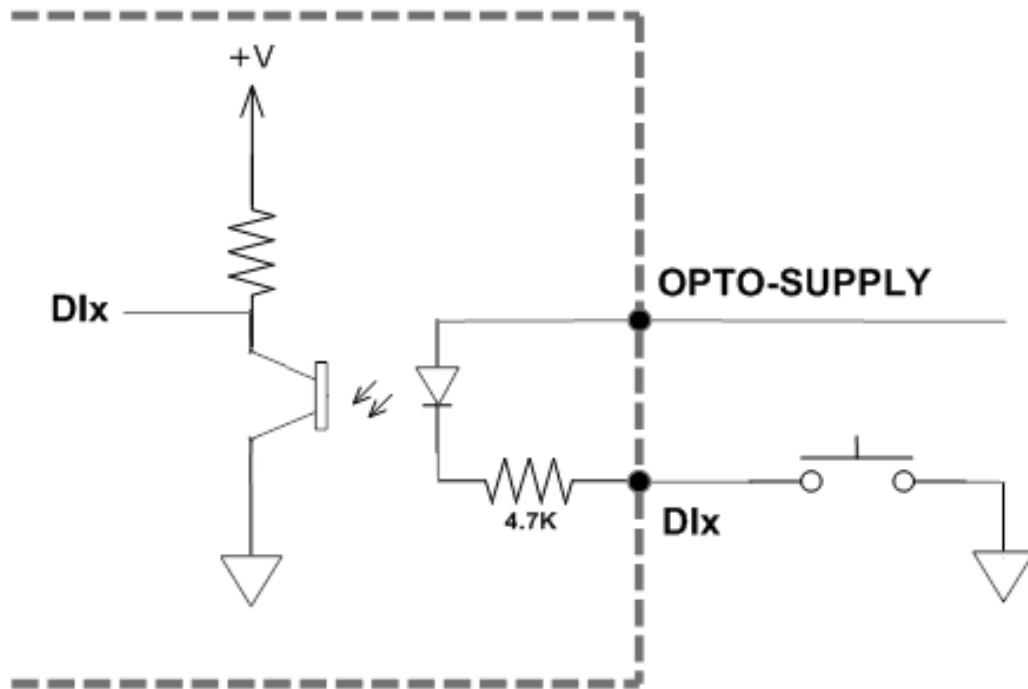


Figure 4.2

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

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

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

## 4.6. Digital Outputs

Figure 4.3 shows an example wiring of the digital outputs. All opto-isolated digital outputs will be PNP type.

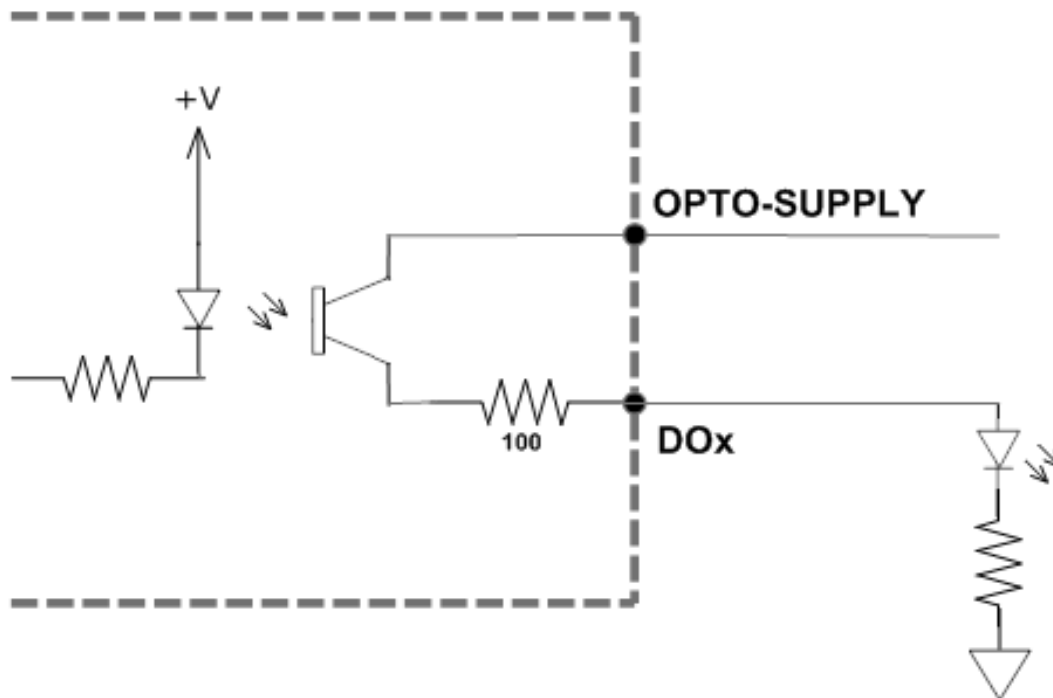


Figure 4.3

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

When activated, the opto-isolator for the digital output pulls the voltage on the digital output line to the opto-supply. The maximum sink current for digital outputs is 50mA. Take caution to select the appropriate external resistor so that the current does not exceed 50mA.

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

## 4.7. Analog Input

Analog inputs are 0 to 3.3V range and 10 bit in resolution. Using the analog input, analog speed control can be achieved. Section 6.12 will provide details on analog speed control mode. The maximum source current for the analog input is 10mA. The recommended potentiometer range is 1K $\Omega$  to 10K $\Omega$ .

---

## 5. Stepper Motor Driver Overview

The ACE-SDC-V3S has one built-in microstep driver, allowing the controller to directly drive a stepper motor.

### 5.1. Microstep

The ACE-SDC-V3S has four microstepping options including full step, ½ step, ¼ step, and 1/16 step. In general, the number of microsteps per revolution can be determined by the following equation, where “DEGREE” is the full step angle of the motor and “USTEP” is the microstep setting.

$$\text{Microsteps/Revolution} = 360^\circ / (\text{DEGREE} * \text{USTEP})$$

For example, using a 1.8° motor and a microstep setting of 1/16, the number of microsteps/revolution would be  $[360^\circ / (1.8^\circ * 1/16)] = 3200$ .

### 5.2. Driver Current

The ACE-SDC-V3S has a maximum rated current output of 2.0A. The current settings should not exceed the rated current of the motor being used. Setting the driver current higher than the maximum rated current will overheat and potentially damage the motor. It is recommended to use a current setting that is in the range of 60-80% of the maximum rated current for the motor.

## 6. Communication Interface

### 6.1. USB Communication

ACE-SDC-V3S USB communication is 2.0 compliant.

In order to communicate with ACE-SDC-V3S via USB, the proper software driver must first be installed. Before connecting the ACE-SDC-V3S 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.

#### 6.1.1. Typical USB Setup

The ACE-SDC-V3S 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 6.0 below.

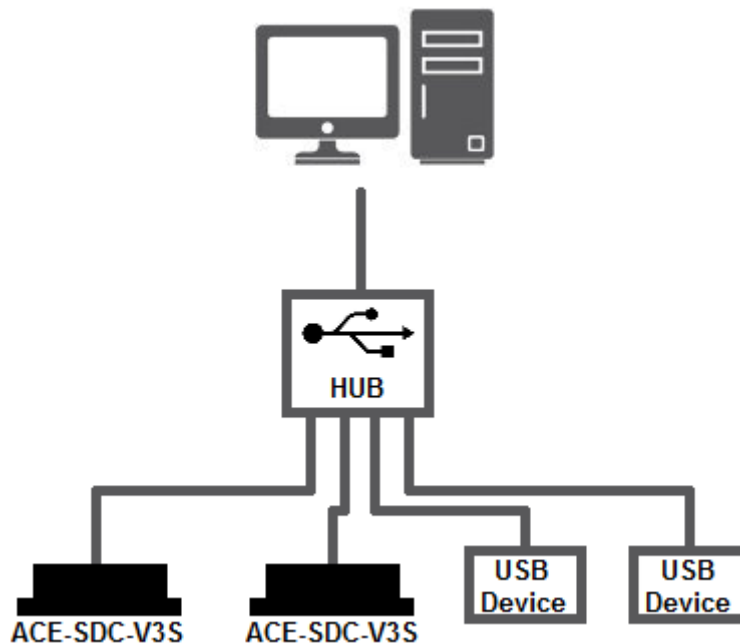


Figure 6.0

#### 6.1.2. USB Communication API

Communication between the PC and ACE-SDC-V3S 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-SDC-V3S.

Typical communication transaction time between PC and ACE-SDC-V3S 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.

### 6.1.3. 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 command is sent. In this case, the data buffers between the PC and the USB device are out of sync. Below are some suggestions to help alleviate 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 fnPerformmaxComFlush(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 6.1.



Figure 6.1

## 6.2. Device Number

If multiple ACE-SDC-V3S devices are connected to the PC, each device should have a unique device number. This will allow the PC to differentiate between multiple controllers. In order to make this change to an ACE-SDC-V3S, first store the desired number using the **DN** command. Note that this value must be within the range [CFG00 – CFG99].

To write the values to the devices' 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: **CFG00**

## 6.3. Windows GUI

The ACE-SDC-V3S 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.



## 7. 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-SDC-V3S.

### 7.1. Motion Profile

ACE-SDC-V3S incorporates trapezoidal velocity profile as shown below.

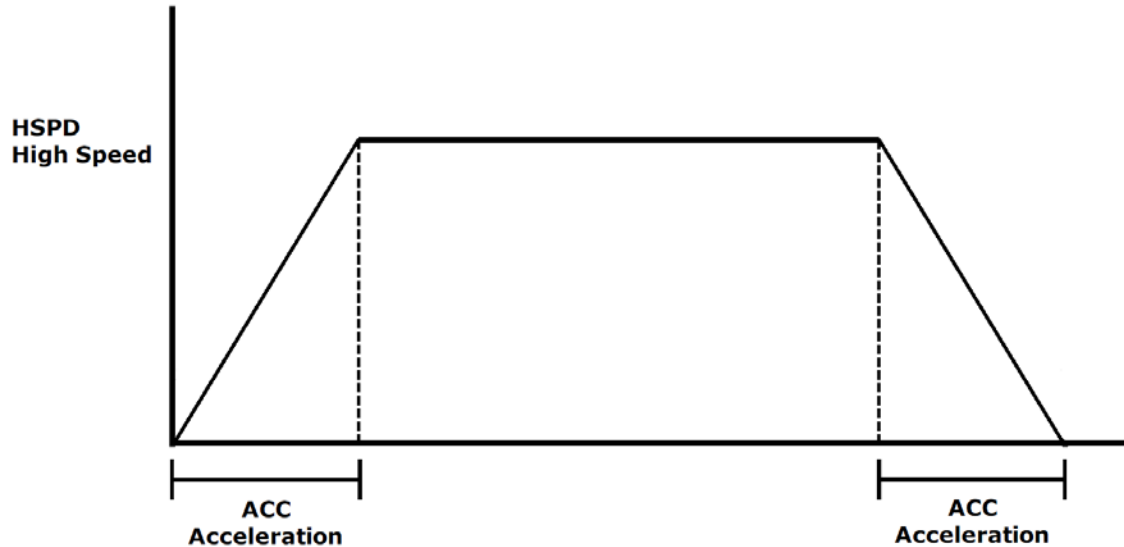


Figure 7.0

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

High speed is in pps (pulses/second). Use ASCII commands **HSPD** to set/get high speed settings. Acceleration times are in milliseconds. Use the **ACC** command to set/get acceleration values.

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	<b>HSPD</b>	<b>ACC</b>
Standalone	<b>HSPD</b>	<b>ACC</b>

## 7.2. On-The-Fly Speed Change

An on-the-fly speed change can be achieved at any point while the motor is in motion. The **SSPD[*speed*]** (ASCII) command can be used to perform the actual speed change.

ASCII	<b>SSPD</b>
Standalone	-

## 7.3. Motor Position

The ACE-SDC-V3S 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 positions can be read using the ASCII command **PX**, which returns the pulse position.

ASCII	<b>PX</b>
Standalone	<b>PX</b>

## 7.4. Motor Power

The **EO** command can be used to enable or disable the current to the motor. The effect of the enable output signals will depend on the characteristics of the motor drive. By default, the enable output is enabled at boot-up.

ASCII	<b>EO</b>
Standalone	<b>EO</b>

## 7.5. Jog Move

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

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

ASCII	<b>J[+/-]</b>
Standalone	<b>JOGX[+/-]</b>

## 7.6. 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	<b>ABORT</b>	<b>STOP</b>
Standalone	<b>ABORTX</b>	<b>STOPX</b>

## 7.7. Positional Moves

The ACE-SDC-V3S can perform positional moves in absolute or incremental mode. For absolute mode, the **ABS** command should be used, 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	<b>ABS</b>	<b>INC</b>	<b>MM</b>	<b>X[target]</b>
Standalone	<b>ABS</b>	<b>INC</b>	-	<b>X[target]</b>

## 7.8. Homing

The ACE-SDC-V3S has only one homing routine. This routine involves moving the motor and using the home input (**DI2**) to determine the zero reference position.

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 7.1 shows the homing routine.

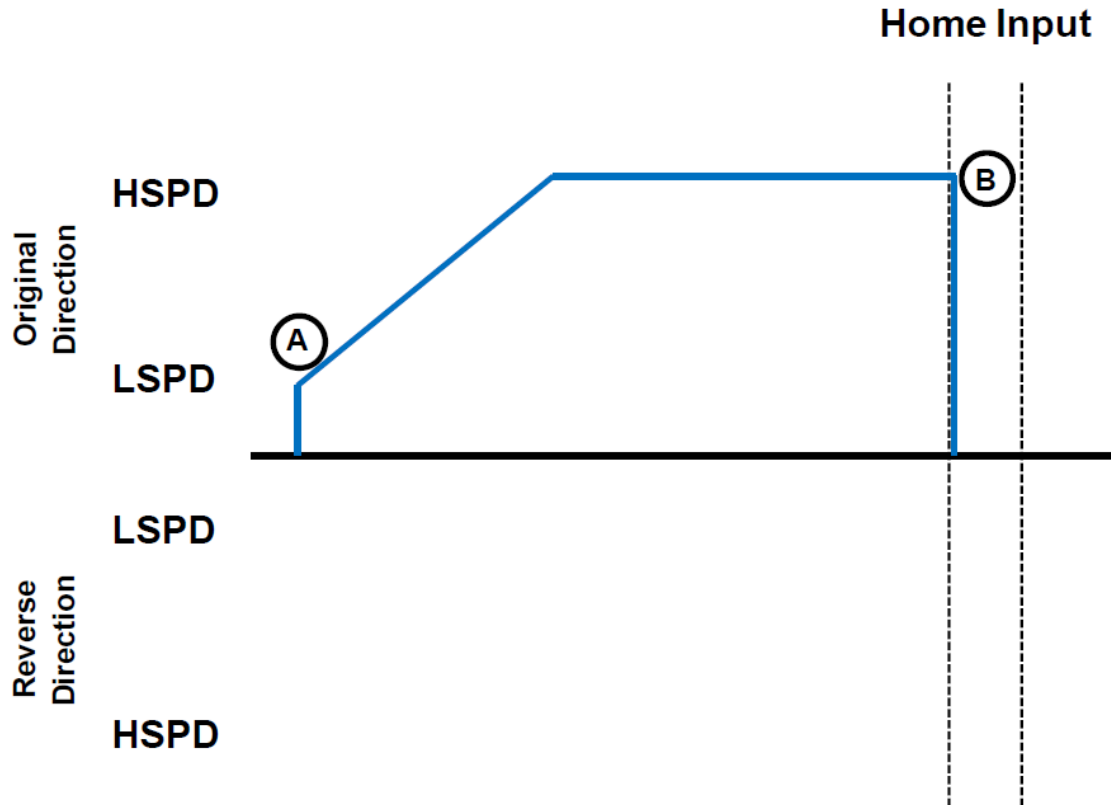


Figure 7.1

- A. Issuing the command starts the motor from low speed and accelerates to high speed in search of the home input.
- B. As soon as the home input (**DI2**) is triggered, the position counter is reset to zero and the motor stops immediately. If the home switch is triggered in the middle of the acceleration, the motor stop immediately.

ASCII	<b>H[+/-]</b>
Standalone	<b>HOMEX[+/-]</b>

### 7.9. Limits Switch Function.

Triggering the limit switch while the motor is moving will stop the motion immediately. For example, if the positive limit switch is triggered while moving in 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.

Digital Input 1 (DI1) should be used to toggle the negative limit switch. Digital Input 3 (DI3) should be used to toggle the positive limit switch. The limit switch function can be disabled by using the **DL** command. By disabling the limit switch function, the limits switches can be used as general purpose inputs.

## 7.10. Motor Status

Motor status can be read anytime using the **MST** command. The following are bit representation of motor status.

Bit	Description
0	Motor running at constant speed
1	Motor in acceleration
2	Motor in deceleration
3	Minus limit input switch status
4	Plus limit input switch status

Table 7.0

## 7.11. Digital Inputs / Outputs

ACE-SDC-V3S module comes with 6 digital inputs and 2 digital outputs.

### 7.11.1. 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/-LIM/Direction	DI1
1	Digital Input 2/HOME	DI2
2	Digital Input 3/+LIM	DI3
3	Digital Input 4	DI4
4	Digital Input 5	DI5
5	Digital Input 6	DI6

Table 7.1

Digital input 2 also acts as the HOME input button for the homing routine. See section 7.8 for more information.

Digital inputs 1 and 3 can behave as the -LIM and +LIM inputs respectively. Unlike the home input, DI1&3 only have this property when “Disable Limit” is turned off. When any move command is issued and the appropriate LIM input is triggered, the motor stops movement and the position is held.

Digital input 1 can also behave as the “direction” button. This occurs when the ACE-SDC-V3S is in analog mode. When triggered, the direction of the motor is reversed.

ASCII	DI	DI[1-6]
Standalone	DI	DI[1-6]

### 7.11.2. Digital Outputs

Read and set digital output status using the **DO** command.

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

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

Table 7.2

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]	-

### 7.12. Analog Speed Control

In analog speed control mode, the speed of the motor is determined by the analog input value [0-1023]. The direction of the motor is determined by the status of digital input 1.

There is only one digital input for direction used for the analog speed as shown below. Other digital IO must be turned OFF for proper analog speed control.

### 7.13. Polarity

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

Bit	Description
0	Digital Outputs
1	Digital Inputs
2	Not Used
3	Not Used
4	Not Used
5	Not Used
6	Not Used
7	Not Used
8	Standalone Error

Table 7.3

†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	<b>POL</b>
Standalone	-

## 7.14. Standalone Program Specification

Standalone programming allows the controller to execute a user defined program that is stored in the internal memory of the ACE-SDC-V3S. The standalone program can be run independently of USB communication or while communication is active.

Standalone programs can be written to the ACE-SDC-V3S 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-SDC-V3S. Each line of written standalone code creates ~1-4 assembly lines of code after compilation.

The ACE-SDC-V3S can store and operate up to two separate standalone programs simultaneously.

### 7.14.1. Standalone Program Specification

Memory size: 1275 assembly lines ~7.5KB.

Note: Each line of pre-compiled code equates to 1-4 lines of assembly lines.

### 7.14.2. Standalone Control

The ACE-SDC-V3S 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**, Program 1 uses command **SR1**, and so on. For examples of multi-threading, 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

Table7.4

### 7.14.3. Standalone Status

The **SASTAT[0-1]** command can be used to determine the current status of the specified standalone program. Table 6.5 details the return values of this command.

Value	Description
0	Idle
1	Running

2	Paused
3	N/A
4	Errored

Table 7.5

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

#### 7.14.4. Standalone Subroutines

The ACE-SDC-V3S 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.

#### 7.14.5. Error Handling

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

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

#### 7.14.6. Standalone Variables

The ACE-SDC-V3S has 50 32-bit signed standalone variables available for general purpose use. They can be used to perform basic calculations and support integer operations. The **V[1-50]** command can be used to access the specified variables. The syntax for all available operations can be found below. Note that these operations can only be performed in standalone programming.



Operator	Description	Example
+	Integer Addition	V1=V2+V3
-	Integer Subtraction	V1=V2-V3
*	Integer Multiplication	V1=V2*V3
/	Integer Division (round down)	V1=V2/V3
%	Modulus	V1=V2%5
>>	Bit Shift Right	V1=V2>>2
<<	Bit Shift Left	V1=V2<<2
&	Bitwise AND	V1=V2&7
	Bitwise OR	V1=V2 8
~	Bitwise NOT	V1=~V2

Table 7.6

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

#### 7.14.7. Standalone Run On Boot-Up

Standalone can be configured to run on boot-up using the **SLOAD** (ASCII) 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 6.7 for the bit assignment of the **SLOAD** setting.

Bit	Description
0	Standalone Program 0
1	Standalone Program 1

Table 7.7

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

#### 7.14.8. 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**  
V1=100

; \* Wait for the move to complete

## 7.15. Storing to Flash

The following items are stored to flash:

ASCII Command	Description
DL	Disable limit
DN	Device name
P6	Run current for joystick and USB mode [DI3=0/DI4=0]
P8	Idle current
P9	Microstep
PA	Move mode
POL	Polarity Settings
DOBOOT	DO configuration at boot-up
SLOAD	Standalone program run on boot-up parameter
V26-V50	Note that on boot-up, V1-V25 are reset to value 0

Table 7.8

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

## 8. Software Overview

The ACE-SDC-V3S has Windows compatible software that allows for USB communication. It can be downloaded from the Arcus-Technology website. To communicate over a USB connection, make sure that the ACE-SDC-V3S is connected to one of the available ports on the PC. Startup the ACE-SDC-V3S GUI program and you will see the following screen in Figure 8.0.



Figure 8.0

To choose a particular device, select it and then press OK.

**Important Note:** In order to communicate with ACE-SDC-V3S via USB, the proper driver must first be installed. Before connecting the ACE-SDC-V3S 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.

## 8.1. Main Control Screen

The Main Control Screen provides accessibility to all the available function on the ACE-SDC-V3S. All features can be tested and verified.

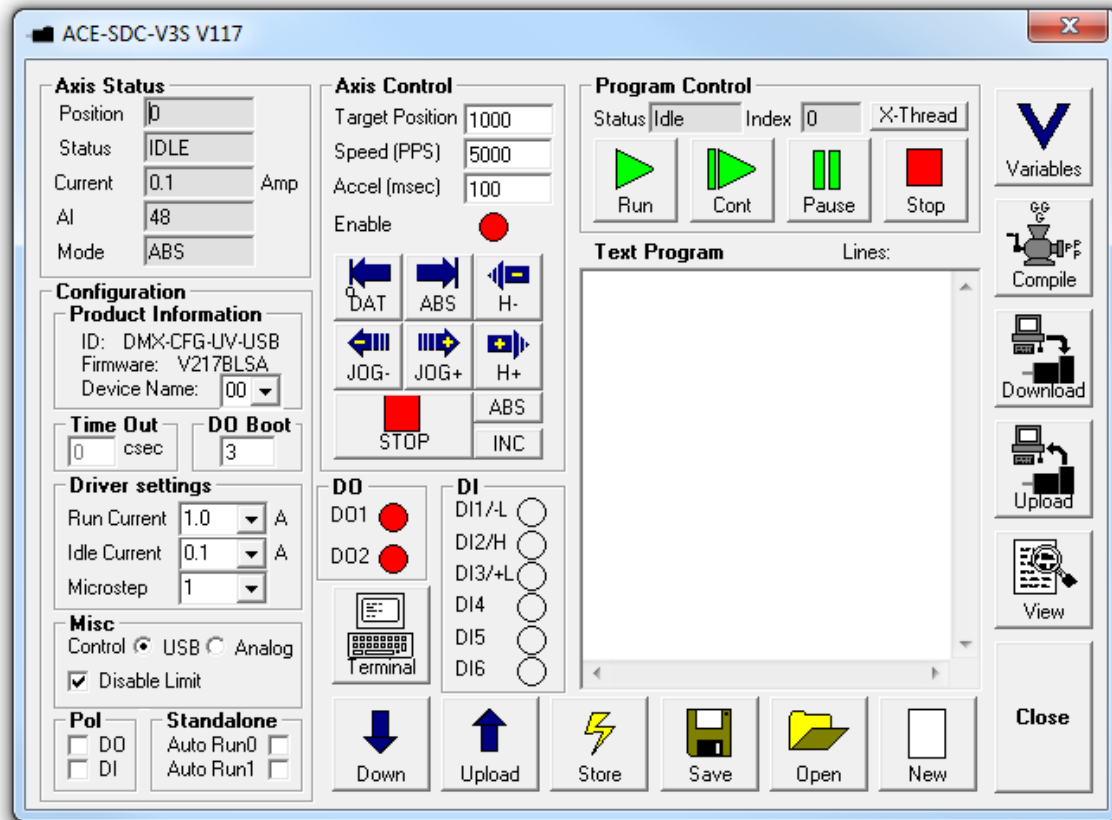


Figure 8.1

### 8.1.1. Status

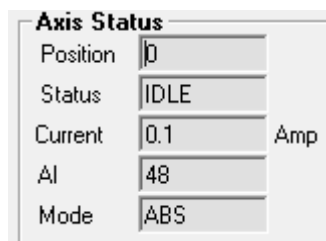


Figure 8.2

1. **Position Value** – Shows the value of the current position.
2. **Motor Status**
  - a. **IDLE** – Motor is idle and not moving.
  - b. **ACCEL** – Motor is accelerating.
  - c. **DECEL** – Motor is decelerating.
  - d. **CONST** – Motor is moving at constant speed.
3. **Current**  
Displays current value.

#### 4. AI

Displays current analog input value (10 bit).  
Range is 0 to 1023 which corresponds to 0 to 3.3V.

#### 5. Mode

Displays the current move mode of the motor.  
ABS – Absolute moves.  
INC – Incremental moves.

### 8.1.2. Control

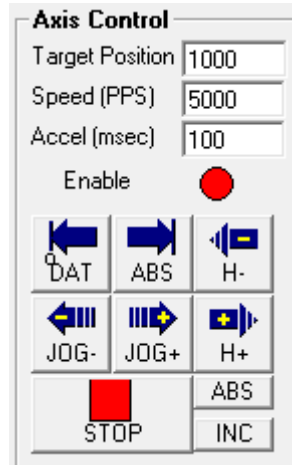


Figure 8.3

1. **Target Position** – Used to move the motor to the target position.
2. **Speed** – Set high speed setting of the motor (up to 16Kpps).
3. **Accel** – Set acceleration time (up to 300ms).
4. **Enable Status** – Displays the energized status of the motor.
5. **Datum** – Moves the motor back to the zero position.
6. **ABS** – Moves the motor to the target position.
7. **H±** – Performs a homing routine in either the positive or negative direction.
8. **ABS/INC** – Puts the motor in absolute or incremental move mode.
9. **STOP** – Stops the motor.
10. **JOG±** – Jogs the motor in the positive or negative direction.

### 8.1.3. Digital Input / Output

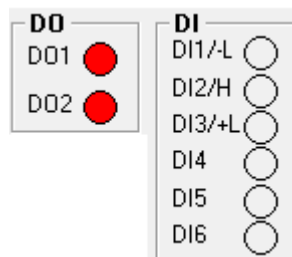


Figure 8.4

Displays the current digital input and output status.

### 8.1.4. Program File Control



Figure 8.5

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

### 8.1.5. Standalone Program Editor



Figure 8.6

1. Write the standalone program in the Program Editor.
2. Use the “Clear Code Space” button to remove the current standalone program.
3. Use “>” button to open a larger and easier to manage program editor.

### 8.1.6. Standalone Program Control

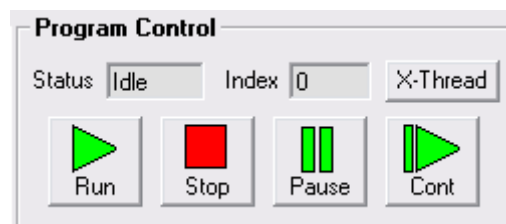


Figure 8.7

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).

### 8.1.7. Standalone Program Compile / Download / Upload /View



Figure 8.8

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

## 8.1.8. Terminal

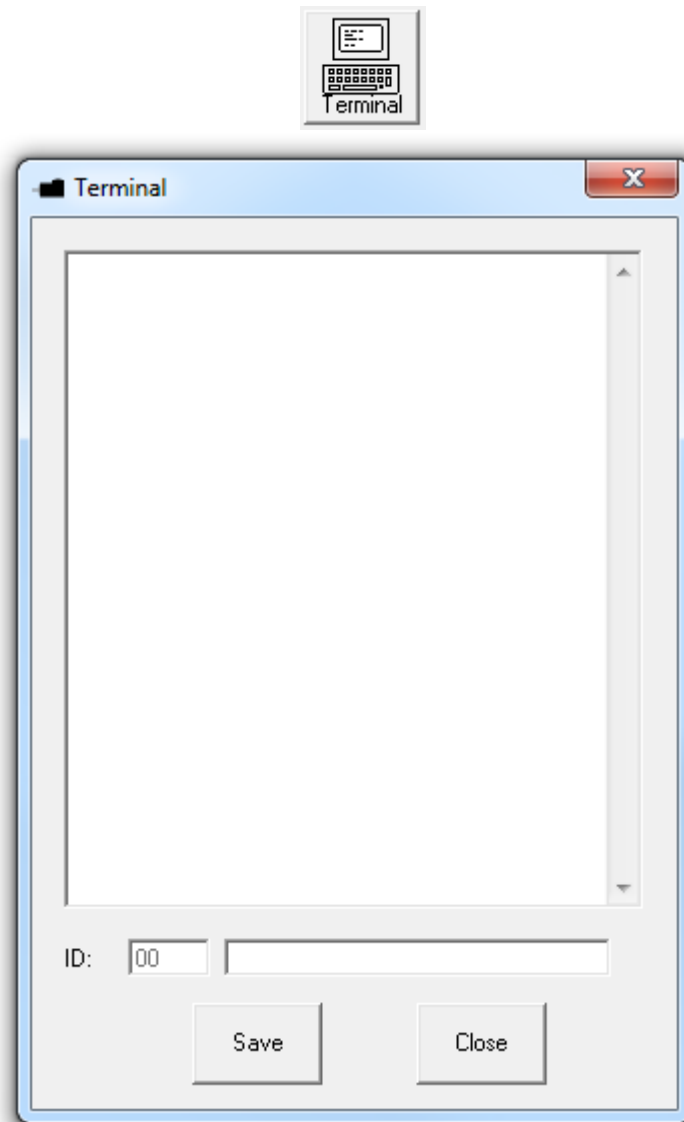


Figure 8.9

Terminal dialog box allows manual testing of the commands from a terminal screen as shown in Figure 8.9.



### 8.1.9. Variable Status

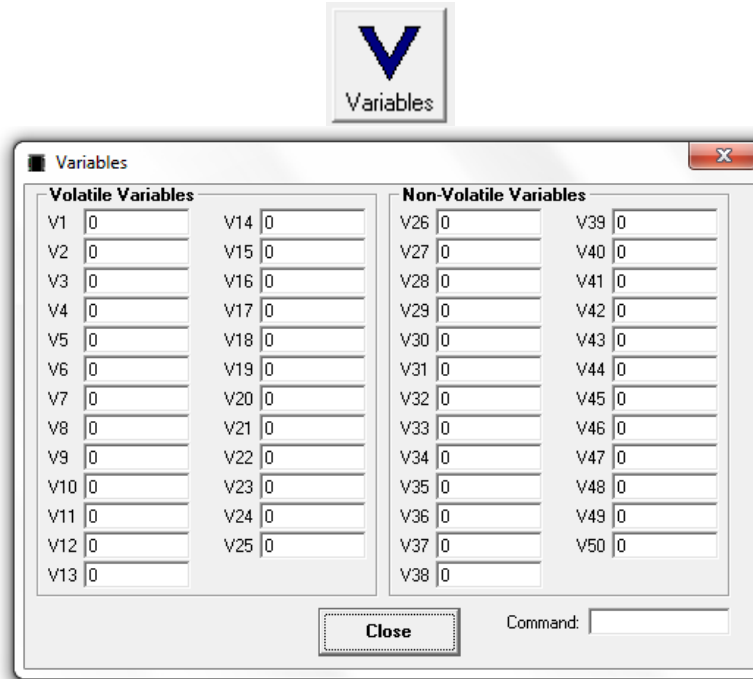


Figure 8.10

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

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

### 8.1.10. Upload / Download / Store to Flash

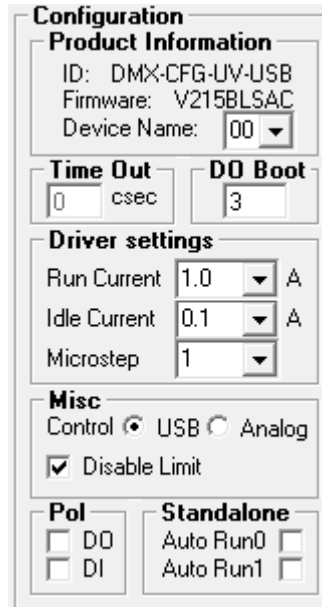


Figure 8.11

Whenever the parameter in the Configuration section is changed, it must be downloaded to take effect. For example, if the idle current is changed from 0.35A to 0A, the new idle current will only go into effect after it is downloaded. If the control mode is changed from Analog to USB from the screen, it must be downloaded for the new control mode to be used.

Once the parameters are downloaded, they can be permanently stored to the flash memory so that when the controller is powered, the stored parameters are used.

### 8.1.11. Configuration



The screenshot shows a configuration window with the following sections:

- Configuration**
  - Product Information**
    - ID: DMX-CFG-UV-USB
    - Firmware: V215BLSAC
    - Device Name: 00 (dropdown)
  - Time Out**: 0 csec
  - DO Boot**: 3
  - Driver settings**
    - Run Current: 1.0 A (dropdown)
    - Idle Current: 0.1 A (dropdown)
    - Microstep: 1 (dropdown)
  - Misc**
    - Control:  USB  Analog
    - Disable Limit
  - Pol**
    - DO
    - DI
  - Standalone**
    - Auto Run0:
    - Auto Run1:

Figure 8.12

1. **Product Information** – Displays the ID and version number
  - a. To change the device name, select a new one from the dropdown list. In order to for the setting to take effect download the device name, store it to flash memory and then perform a power cycle.
2. **Driver settings**
  - a. **Run Current:** Set the current when the motor is moving
  - b. **Idle Current:** Set the current while the motor is idle
  - c. **Microstep:** Set the microstep setting of the driver
3. **Misc. Settings**
  - a. **USB Mode:** motor control is done through the PC using USB communication
  - b. **Analog Control:** Speed control is done using the analog input.
  - c. **Disable Limit:** turn the limit switch function for DI1 and DI3 on/off
4. **Polarity** - Change the polarity of the digital inputs and outputs.
5. **Standalone** – Allows either program 0 and/or program 1 to start on startup.
6. **DO Boot** - Set the digital output status on controller power up

## 9. ASCII Language Specification

**Important Note:** All the commands described in this section are interactive ASCII commands and are not analogous to standalone commands. Refer to section 11 for details regarding standalone commands.

ACE-SDC language is case sensitive. All command should be in upper case letters. Invalid command 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.

### 9.1. ASCII Command Set

Command	Description	Return
ABORT	Immediately stops the motor if in motion. For decelerate stop, use STOP command.	OK
ABS	Set move mode to incremental	OK
ACC	Returns current acceleration value in milliseconds.	Milliseconds
ACC=[Value]	Sets acceleration value in milliseconds. Example: ACC=300	OK
AI	Get analog input status	0-1023
CUR	Returns the current motor current	mA
DI[1-6]	Get individual bit status of digital inputs	0 – on Non-zero - off
DO	Return status of digital outputs	2-bit number
DO[1-2]	Get individual bit status of digital outputs	0,1
DO[1-2]=[Value]	Set individual bit status of digital outputs	OK
DOBOOT	Return the DO boot-up state	0-3
DOBOOT=[Value]	Set the DO boot-up state	OK
DL	Get the disable limit switch function	0 – limits enabled 1 – limits disabled
DL=[0,1]	Set the disable limit switch function	OK
DN	Get device name	[CFG01-CFG99]
DN=[Value]	Set device name	OK
EO	Returns driver power enable.	1 – Motor power enabled 0 – Motor power disabled
EO=[0,1]	Enables (1) or disable (0) motor power.	OK
GS[0-31]	Call a subroutine that has been previously stored to flash memory	OK
H+	Homes the motor in the positive direction	OK
H-	Homes the motor in the negative direction	OK
HSPD	Returns High Speed Setting	PPS
HSPD=[Value]	Sets High Speed.	OK
ID	Returns product ID	DMX-CFG-UV-USB

INC	Set move mode to incremental	OK
J+	Jogs the motor in positive direction	OK
J-	Jogs the motor in negative direction	OK
MM	Returns the movement mode of the motor	0: ABS 1: INC
MST	Returns motor status	Bit 0 – constant speed Bit 1 – accelerating Bit 2 – decelerating Bit 3 thru 8 – motor current (centi-A)
P6	Get run current for joystick and USB mode	1: 100 mA 2: 200 mA .... 20: 2000 mA
P6=[Value]	Set run current	OK
P8	Get idle current	1: 100 mA 2: 200 mA .... 20: 2000 mA
P8=[Value]	Set idle current	OK
P9	Get microstep	1: Fullstep 2: Halfstep 3: Quarter Step 4: 1/16 step
P9=[Value]	Value → Microstep	OK
PA	Get move mode	1 – USB 2 – Joystick
PA=[Value]	Set control mode	OK
POL	Return current polarity	See Table 7.5
POL=[Value]	Set polarity	OK
PX	Returns current position value	32-bit number
PX=[value]	Sets the current position value	OK
SASTAT	Get standalone program status 0 – Stopped 1 – Running 2 – Paused 4 – In Error	0-4
SA[LineNumber]	Get standalone line - LineNumber: [0,1274]	
SA[LineNumber]=[Value]	Set standalone line - LineNumber: [0,1274]	OK
SLOAD	Returns RunOnBoot parameter	0-1
SLOAD=[Value]	Bit 0 – Standalone program 0 0 – Do NOT run standalone program on boot up 1 – Run standalone program on boot up Bit 1 – Standalone program 1	OK
SPC	Get program counter for standalone program	[0-1274]

SR[0,1]=[Value]	Control standalone program: 0 – Stop standalone program 1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program	OK
SSPD[Value]	On-the-fly speed change. Note that an “=” sign is not used for this command.	OK
STOP	Stops the motor using deceleration if in motion	OK
STORE	Store settings to flash	OK
V[1-50]	Read variables 1-50	32-bit number
V[1-50]=[value]	Set variables 1-50	OK
VER	Get firmware version	VXXX
X[value]	Moves the motor to absolute position value using the HSPD and ACC values	OK

Table 9.0

## 9.2. Error Codes

If an ASCII command cannot be processed by the ACE-SDC-V3S, 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-SDC-V3S
?Moving	A move or position change command is sent while the ACE-SDC-V3S is outputting pulses.
?SSPD not allowed	On-the-fly speed command, SSPD, is issued when the motor is not moving. SA

Table 9.1

## 10. Standalone Language Specification

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

### 10.1. Standalone Command Set

Command	R/W	Description	Example
;	-	Comment notation. Comments out any text following ; in the same line.	;This is a comment
ABORTX	W	Immediately stop all motion.	ABORTX
ABS	W	Set the move mode to absolute mode.	ABS X1000 ;move to position 1000
ACC	R/W	Set/get the individual acceleration setting. Unit is in milliseconds.	ACC=500 ACC=V1
AI	R	Gets the analog input value. Range is from 0-1023mV.	V3=AI
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
DN	W	Change the device name.	DN=CFG00
DO	R/W	Set/get digital output status.	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	-	Used to define the end of a program.	END
EO	R/W	Set/get the enable output status.	EO=1 ;Enable the motor
GOSUB [0-31]	-	Call a subroutine that has been previously stored to flash memory.	GOSUB 0 END
HOMEX[+/-]	W	Home the motor using the home input at high speed in the specified direction.	HOMEX- ;negative home
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.	IF DI1=0 DO=1 ;Turn on DO1 ELSEIF DI2=0 DO=2; Turn on DO2 ELSE DO=0; Turn off DO ENDIF

		Conditions [=, >, <, >=, <=, !=] are available	
INC	W	Set the move mode to incremental mode.	INC X1000 ;increment by 1000
JOGX[+/-]	W	Move the motor indefinitely in the specified direction.	JOGX+
MSTX	R	Get the current motor status.	MSTX
PX	R/W	Set/get the current motor position.	PX=1000 ;Set to X pos to 1000 V1=PX ;Read current X position
SR[0-1]	W	Set the standalone control for the specified program.	SR0=0 ;Turn off program 0
STOPX	W	Stop motion using a decelerated stop.	STOPX
STORE	W	Store settings to flash.	STORE
SUB [0-31] ENDSUB	-	Defines the beginning of a subroutine. ENDSUB should be used to define the end of the subroutine.	SUB 1 DO=4 ENDSUB
V[1-50]	R/W	Set/get standalone variables. The following operations are available: [+] Addition [-] Subtraction [*] Multiplication [/] Division [%] Modulus [>>] Bit shift right [<<] Bit shift left [&] Bitwise AND [ ] Bitwise OR [~] Bitwise NOT	V1=12345 ;Set V1 to 12345 V2=V1+1 ;Set V2 to V1 + 1 V3=DI ;Set V3 to DI V4=DO ;Set V4 to DO V5=~EO ;Set V5 to NOT EO
WAITX	W	Wait for current motion to complete before processing the next line.	X1000 ;move to position 1000 WAITX ;wait for move to finish
WHILE ENDWHILE	-	Perform a standard WHILE loop within the standalone program. ENDWHILE should be used to close off a WHILE loop.  Conditions [=, >, <, >=, <=, !=] are available.	WHILE 1=1 ;Forever loop DO=1 ;Turn on DO1 DO=0 ;Turn off DO1 ENDWHILE
X[position]	W	If in absolute mode, move the X motor to [position]. If in incremental mode, move the motor to [current position] + [position].	X1000

Table 10.0

## 10.2. Example Standalone Programs

### 10.2.1. Standalone Example Program 1 – Single Thread

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

```

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

```

### 10.2.2. Standalone Example Program 2 – Single Thread

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

```

HSPD=20000      ;* Set the high speed to 20000 pulses/sec
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 axis move to complete
    X0         ;* Move to 1000
ENDWHILE      ;* Go back to WHILE statement
END

```

### 10.2.3. Standalone Example Program 3 – Single Thread

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

```

HSPD=20000      ;* Set the high speed to 20000 pulses/sec
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 axis move to complete
    X0         ;* Move to 1000
    V1=V1+1    ;* Increment variable 1
ENDWHILE      ;* Go back to WHILE statement
END

```



#### 10.2.4. Standalone Example Program 4 – Single Thread

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

```

HSPD=20000      ;* Set the high speed to 20000 pulses/sec
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 axis move to complete
        X0    ;* Move to 1000
    ENDIF
ENDWHILE       ;* Go back to WHILE statement
END

```

#### 10.2.5. Standalone Example Program 5 – Single Thread

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

```

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

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

```

### 10.2.6. Standalone Example Program 6 – Single Thread

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

```

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

```

### 10.2.7. Standalone Example Program 7 – Multi Thread

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

```

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

```

## **Contact Information**

Arcus Technology, Inc.

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

[www.arcus-technology.com](http://www.arcus-technology.com)

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