ACE-SXC

Single Axis Step Motor Controller with USB 2.0 Communication

First edition, Oct 2007

Revision History:

1.01 – First revision
1.02 – Added dimensions
1.03 – Added ASCII command set
1.3 – Added new graphics
1.4 – Added firmware and software compatibility

Firmware Compatibility:

V402

Software Compatibility:

V404

Table of Contents

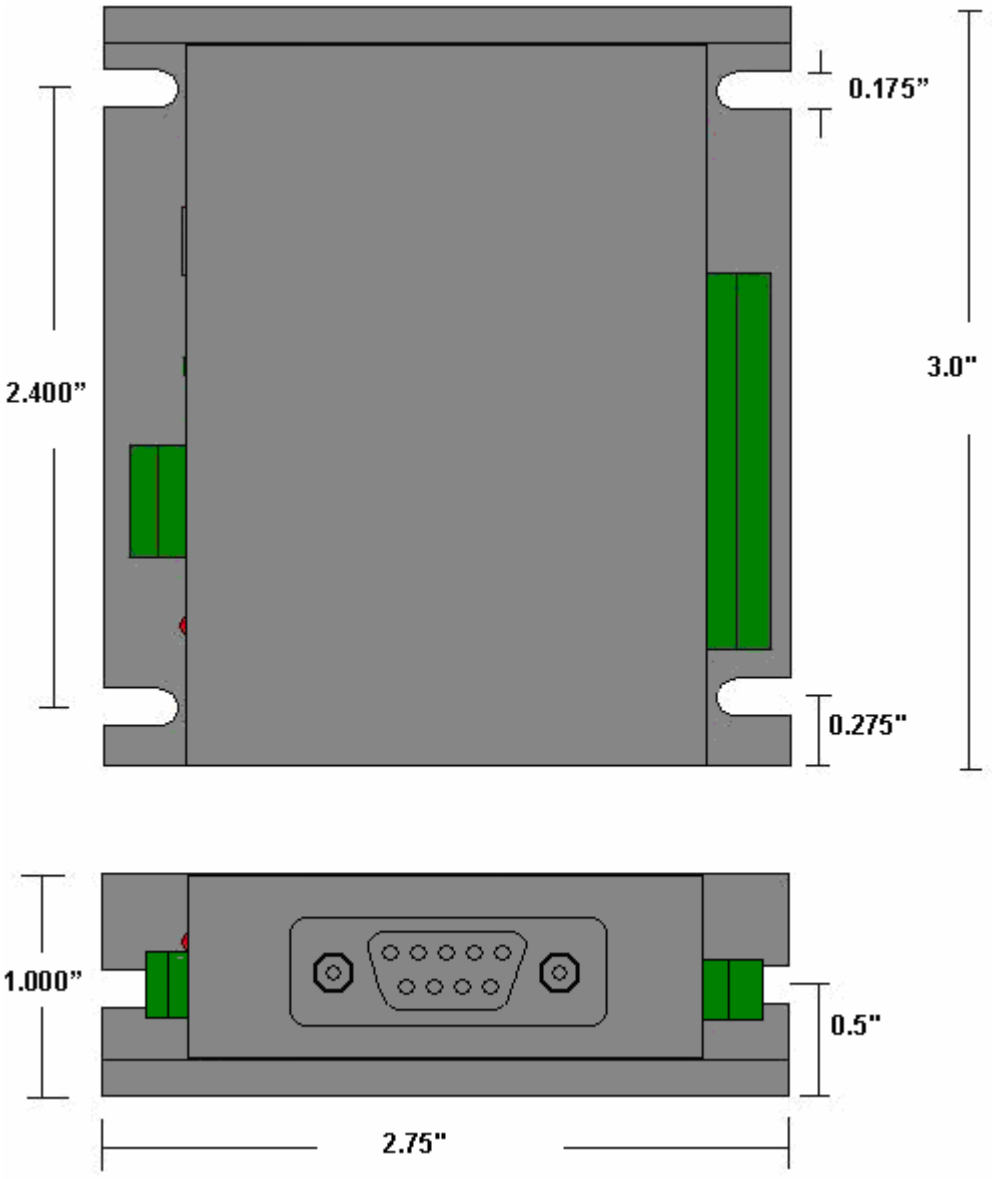
1. Introduction

ACE-SXC is a single axis step motor controller with following features:

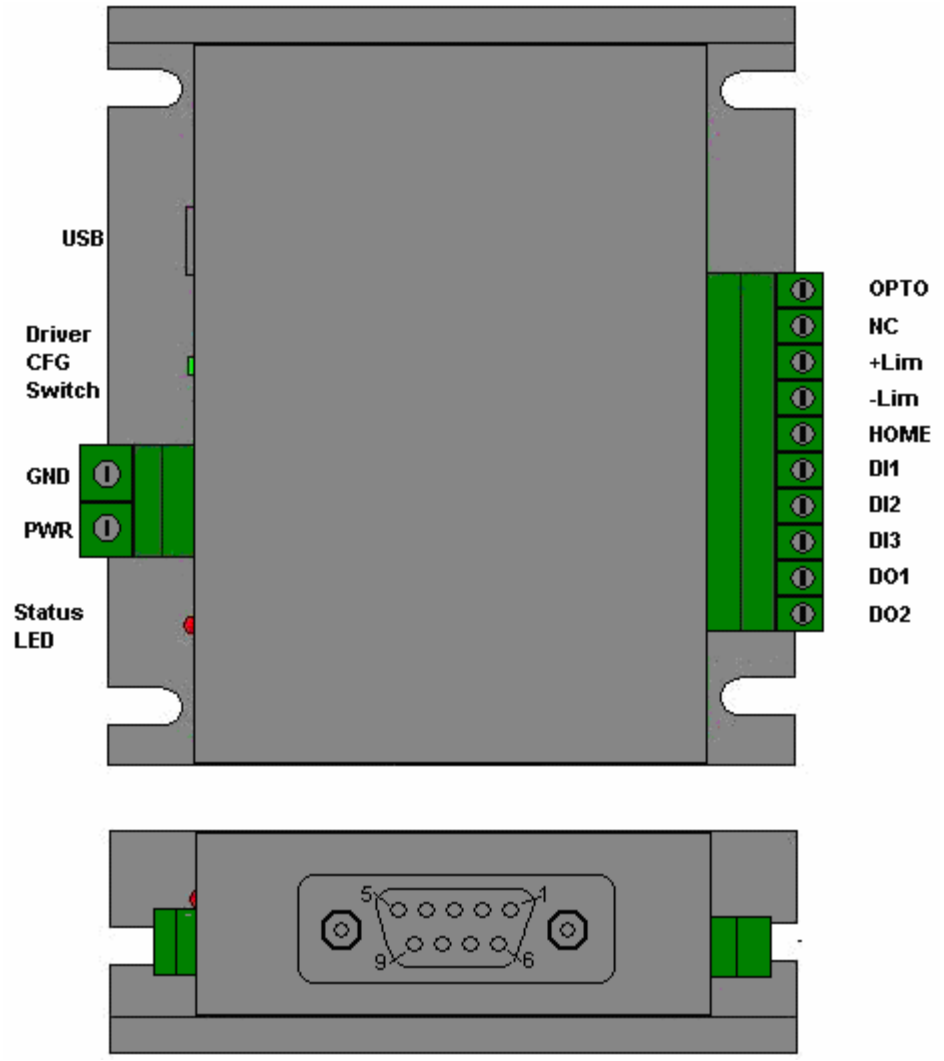
- USB 2.0 Communication
- PC based control through USB 2.0
- Standalone Control with BASIC-like programming language
- 12-48VDC voltage input
- Pulse/Dir differential signal output
- 400K maximum pulse rate output
- Open-collector Enable output
- TTL Alarm input
- Opto-isolated +Limit, -Limit, and Home inputs
- Three Opto-isolated Digital inputs
- Two Opto-isolated Digital outputs
- DMX-K and DMX-A2 and ACE-SDX driver configuration

2. Part Numbering Scheme

3. Dimensions



4. Connectors

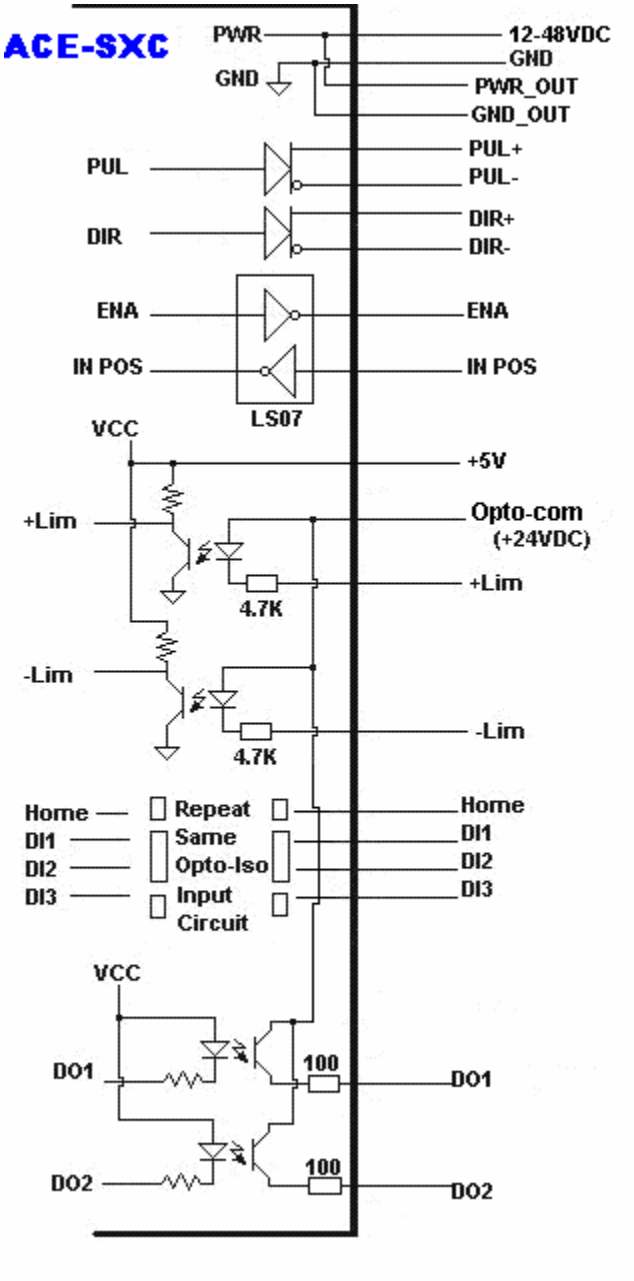


DB9 Pinout

Pin	*Name*	*Type*
1	PWR	OUT
2	PUL+	OUT
3	DIR+	OUT
4	ENA	OUT
5	ALM	IN
6	GND	OUT
7	PUL-	OUT
8	DIR-	OUT
9	5V+	OUT

5. Electrical Specifications

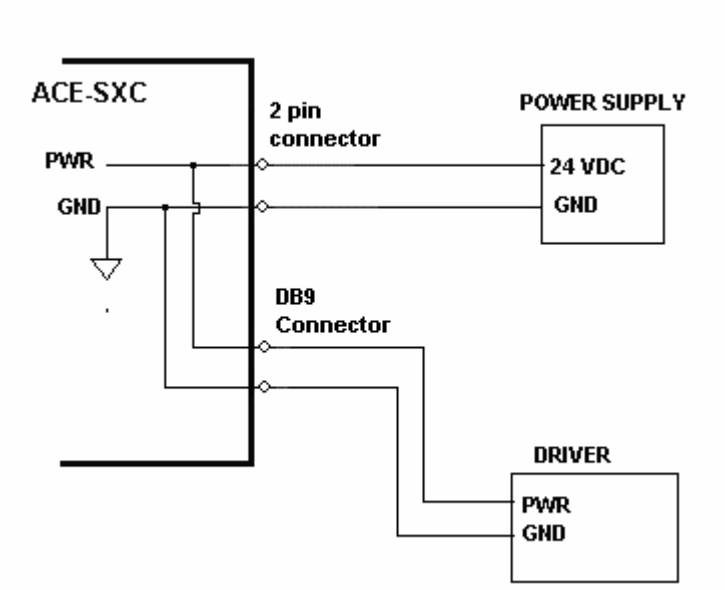
Internal Interface Circuit Overview



Power Input

Regulated Supply Voltage Range:	**+12 to 48 VDC**
Recommended Current for power supply:	**200 mA** (Current required for powering the ACE-SXC. If driver is powered through DB-9 additional current is required to power the driver.)

Power and ground signals that are supplied to ACE-SXC through 2 pin connector are also available through the DB9 pin connector.



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

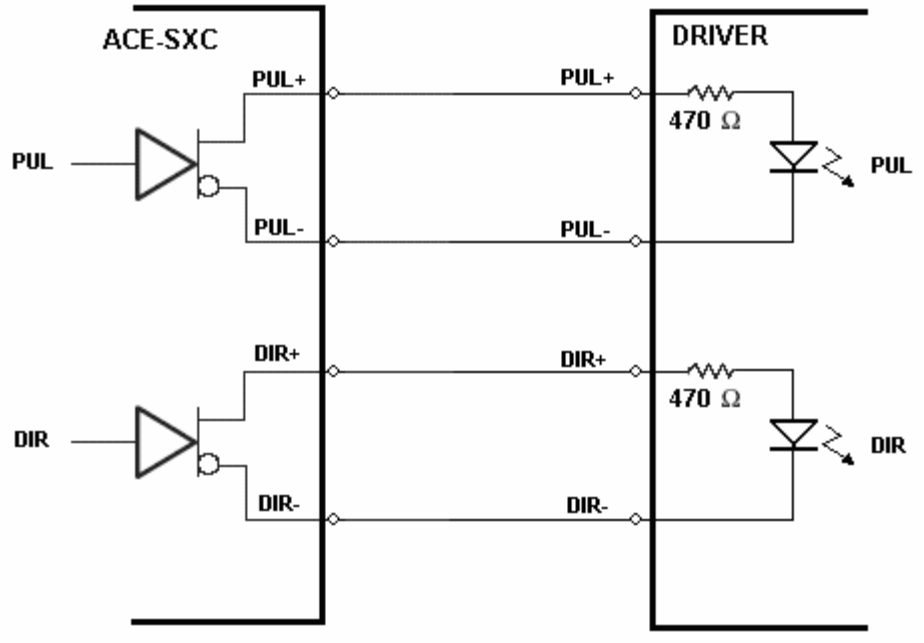
Communication:

Communication:	**USB 2.0**
Connector:	**Mini-B to A**

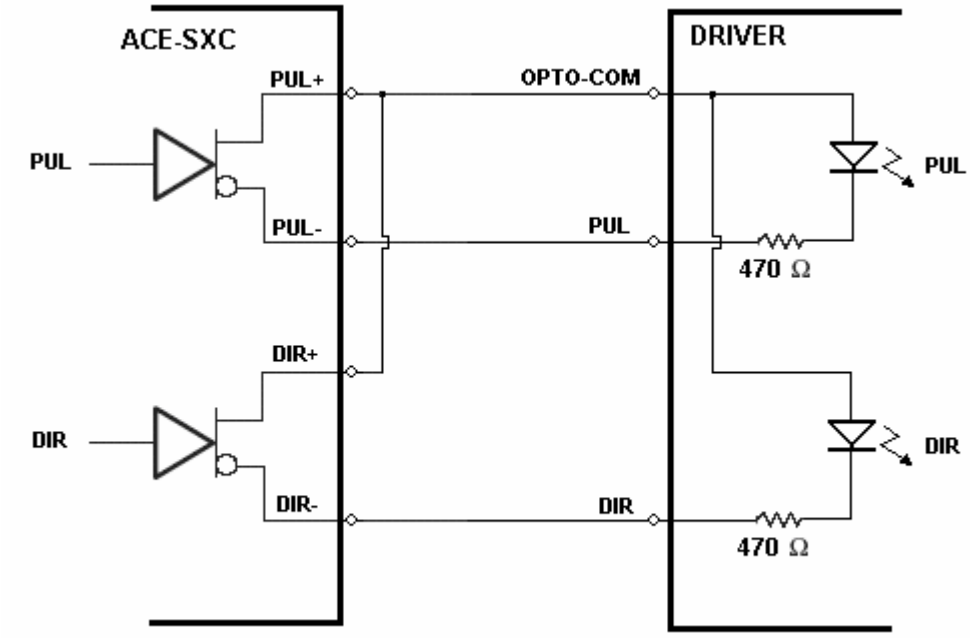
Pulse and Direction Outputs

Pulse and Direction Outputs are differential outputs using 75LS191.

An example of connection to differential pulse/dir input driver is shown below:



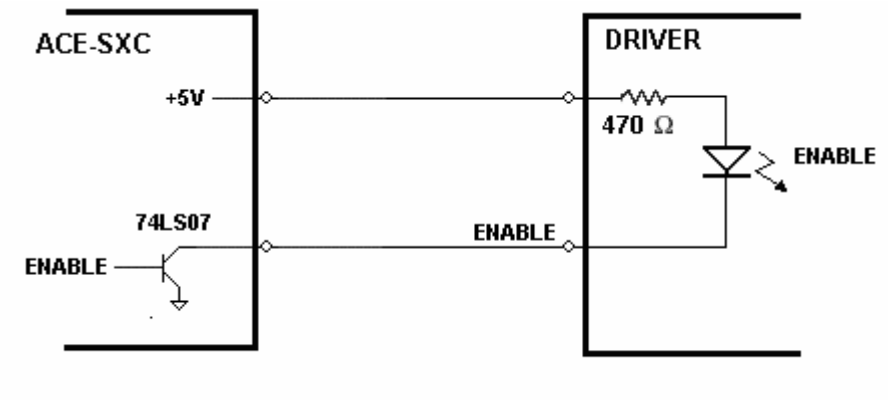
An example of connection to single-ended pulse/dir input driver is shown below:



Enable Output

Enable Output is an open collector output using 74LS07.

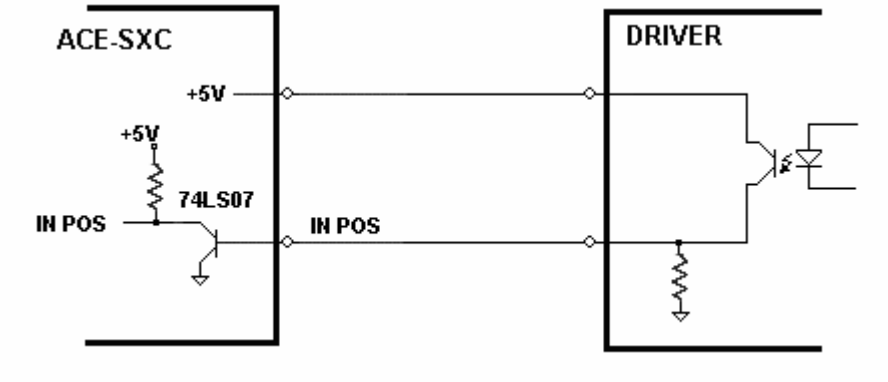
An example of enable circuit connection to a driver is shown below.



Alarm Input

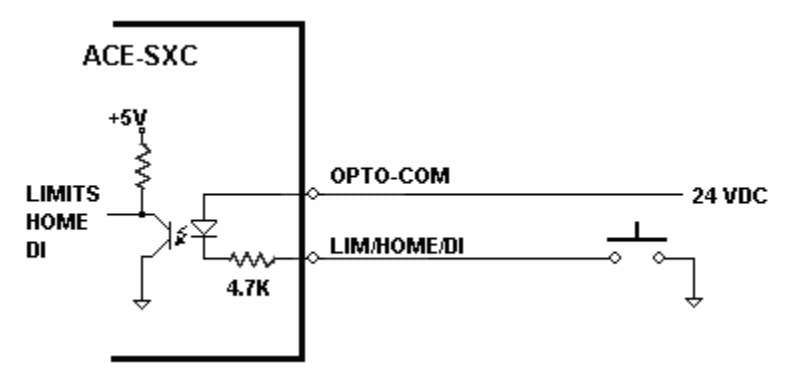
Alarm input is a TTL compatible input using the 74LS07.

An example of Alarm circuit is shown below.



Limits, Home and Digital Inputs:

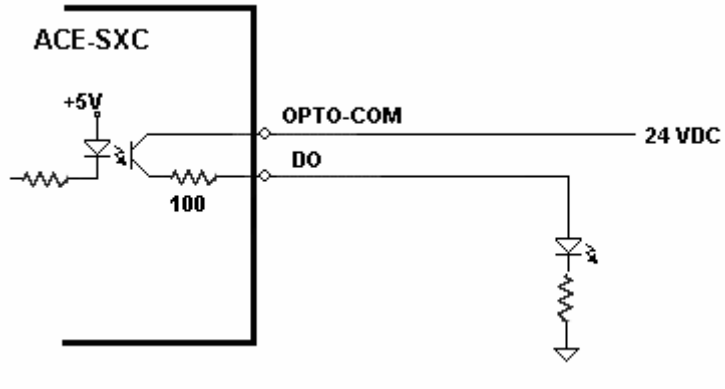
Type:	**Opto-isolated inputs**
Opto voltage supply input:	**+24 VDC**
Maximum diode forward current:	**50mA**



4.7K resistor is built in to the limit, home and digital inputs to limit the current across the diode of the opto-isolator.

Digital Outputs:

Type:	**Opto-isolated open-emitter transistor output**
Maximum emitter current:	**50 mA**



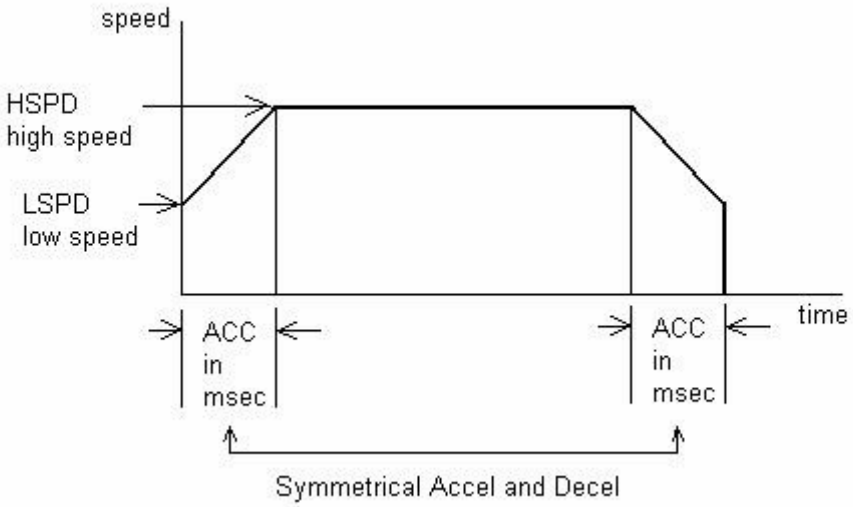
Operating Temperature

Electronic components used in ACE-SXC have maximum ambient operating temperature of **85 degree Celsius**.

6. Motion Control Overview

Motion Profile and Speed

ACE-SXC incorporates trapezoidal velocity profile as shown below.



Acceleration and deceleration time is in milliseconds and are symmetrical. Accelation range is from 10 msec to 1000 msec. Pulse output rate supported is from 100 to 400K pulses/second.

Position Counter

ACE-SXC has 32 bit signed position counter. Range of the position counter is from –2,147,483,648 to 2,147,483,647.

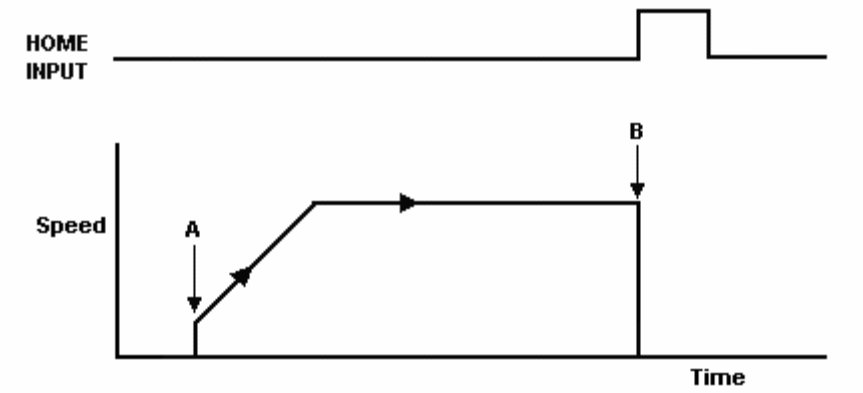
Target Move

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

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

Home Move

Home search sequence involves moving the motor towards the home switch and then stopping when the home input is detected. Following sequence shows the homing routine.



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

B. As soon as the home input is triggered, the position counter is reset to zero and the motor stops immediately. If the home switch is triggered in the middle of the acceleration, the motor stops immediately.

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

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

Jog Move

Jog move is used to continuously move the motor without stopping.

Stopping Motor

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

Limit Switch Function

With limit switch function enabled, triggering of the limit switch in motion will stop the motion immediately depending on the direction of the motion. If positive limit switch is triggered while moving in positive direction, motor will immediately stop and the motor status bit for positive limit error is set. Same is for negative limit while moving in negative direction. Once limit error is set, status must be cleared in order to move again. Once the error is cleared, move the motor out of the limit switch.

Limit switch function can also be disabled. By disabling the limit switch function, the limits switches can be used as general purpose inputs. When limit function is disabled, digital input assignments are: **DI5** for –Limit and **DI7** for +Limit.

Alarm Switch Function

Alarm switch when triggered will immediately stop the motor if in motion regardless of the direction of the motor. Alarm switch function cal be disabled and alarm input can be used as general purpose input as **DI8.**

Configuration Button Function

Configuration button is used to configure the DMX-K-DRV, DMX-A2-DRV, and ACE-SDX. Configuration button can also be used as general purpose digital input in the standalone program. Digital input assignment of configuration button is **DI4**.

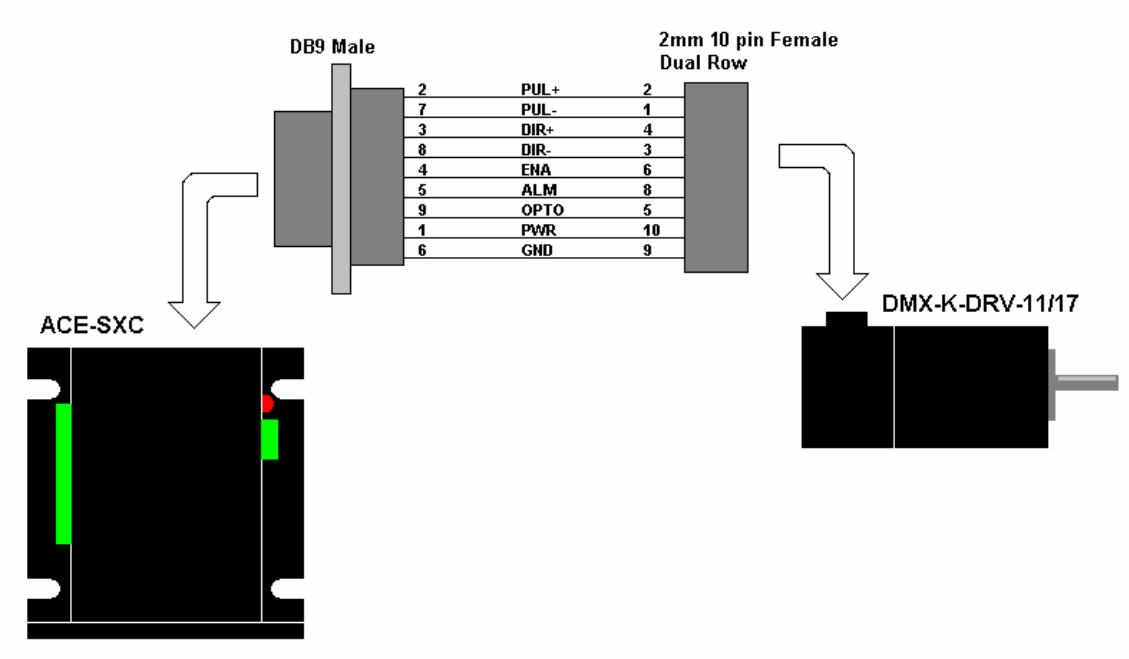
Motor Status

Motor status can be read anytime. The following are bit representation of motor status.

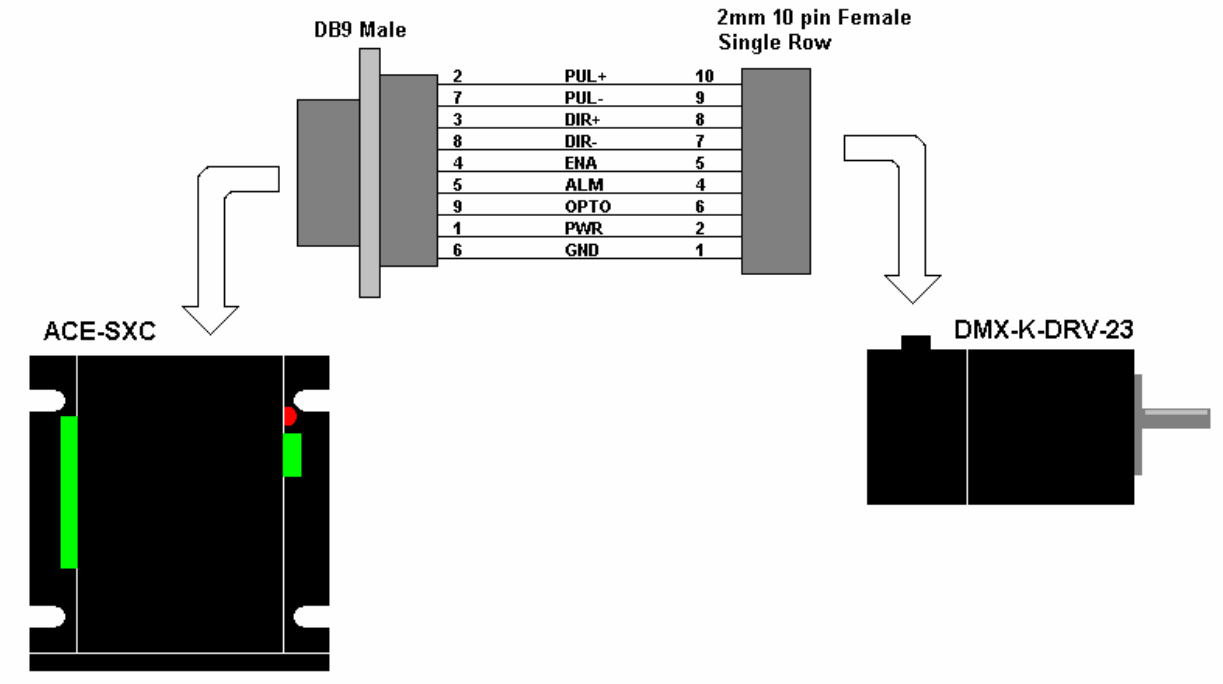
Bit	Description
0	Motor running at constant speed
1	Motor in acceleration
2	Motor in deceleration
3	Home input switch status
4	Minus limit input switch status
5	Plus limit input switch status
6	Minus limit error. This bit is latched when minus limit is hit during negative direction motion. This error must be cleared before issuing any subsequent move commands.
7	Plus limit error. This bit is latched when plus limit is hit during positive direction motion. This error must be cleared before issuing any subsequent move commands.
8	Alarm error. This bit is latched when the alarm input is triggered while motion is in progress. This error must be cleared before issuing any subsequent move commands.

7. Connecting to DMX-K-DRV, DMX-A2-DRV, and ACE-SDX

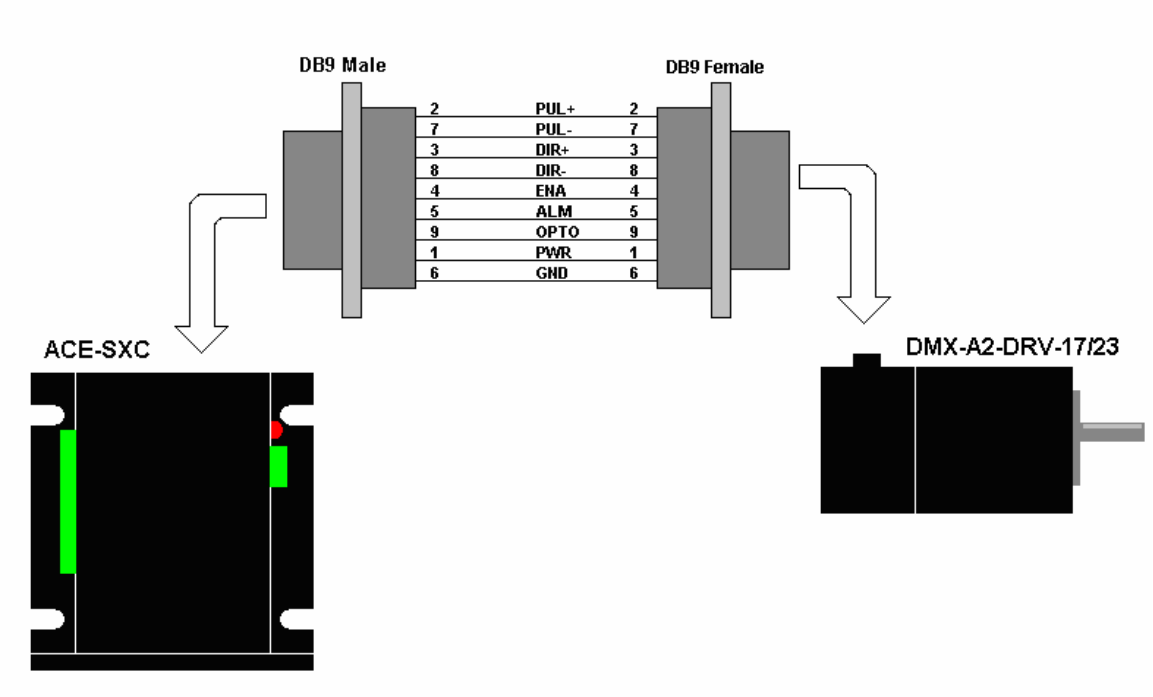
Connecting ACE-SXC to DMX-K-DRV-11/17



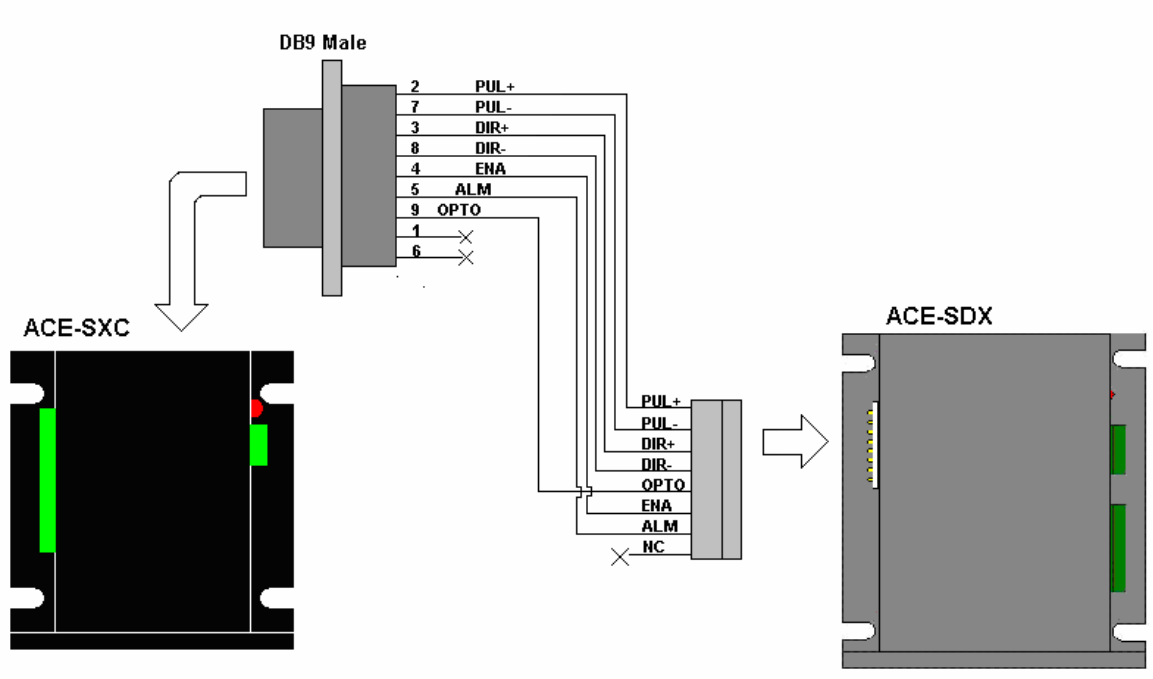
Connecting ACE-SXC to DMX-K-DRV-23



Connecting ACE-SXC to DMX-A2-DRV-17/23



Connecting ACE-SXC to ACE-SDX



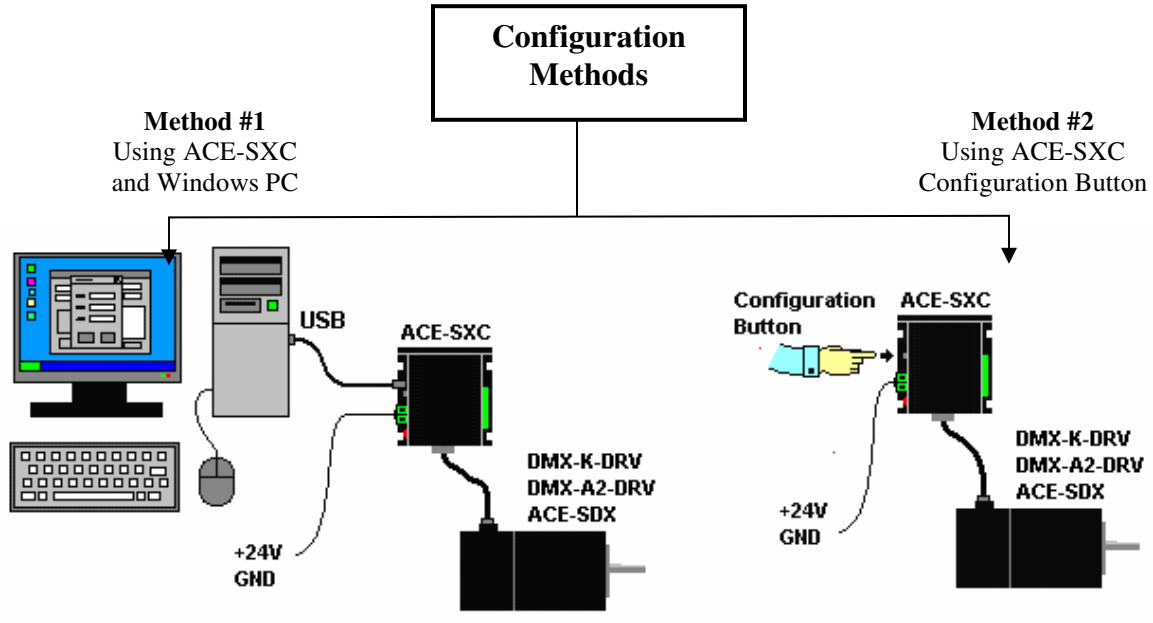
8. DriveMax Configuration

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

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

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

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



Configuration Method #1 – Using Windows PC

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

Configuration Method #2 – Using the Configuration Button

Method #2 uses the configuration button on the ACE-SXC controller to download the driver parameters. Note that configuration button is used only for downloading the driver parameters that have been stored on the ACE-SXC controller.

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

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

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

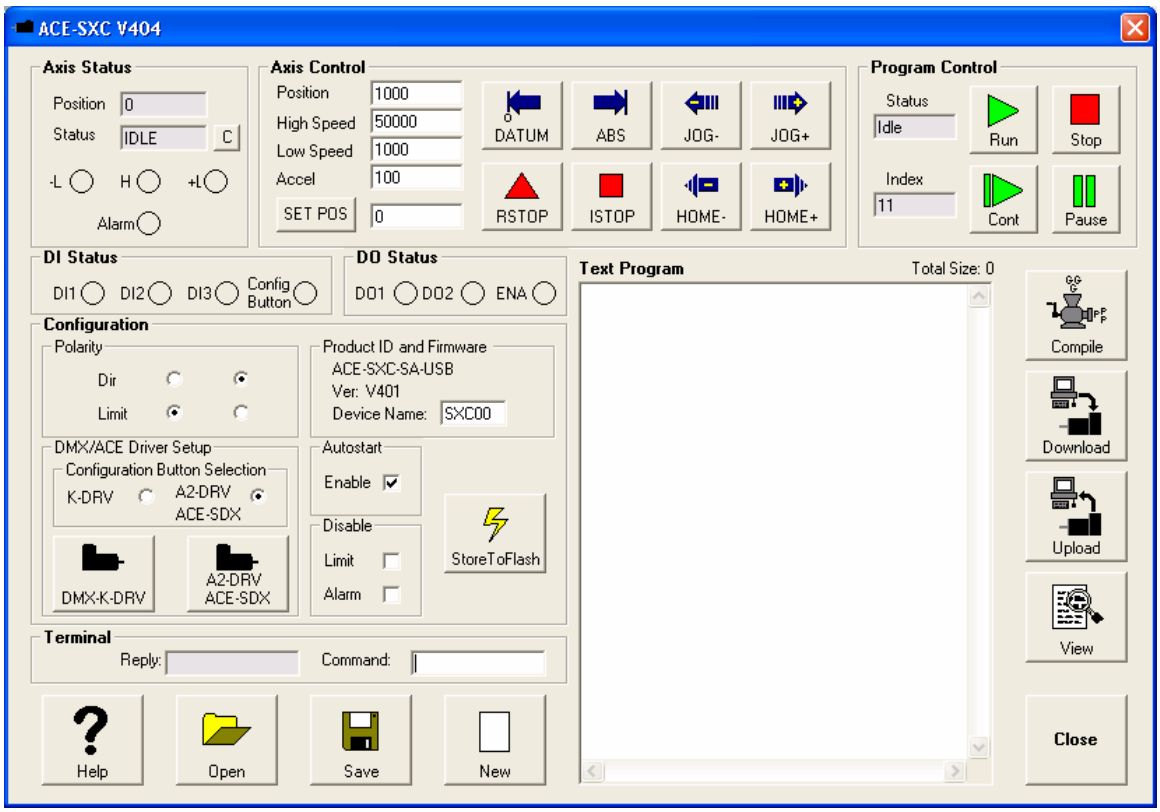
9. ACE-SXC GUI Windows Program

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

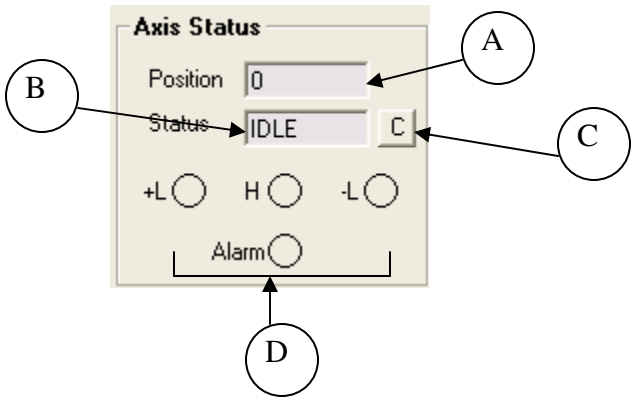
Important Note: In order to communicate with ACE-SXC through USB, proper driver must be installed first. Before connecting the ACE-SXC device or running any program, please go to the Arcus web site and download the USB driver installation instruction and run the USB Driver Installation Program.

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

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

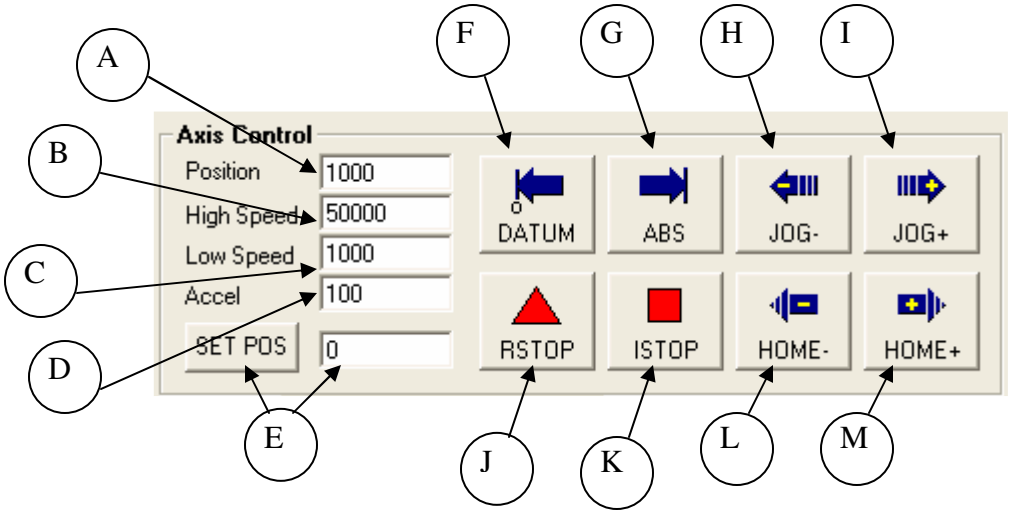


Axis Status



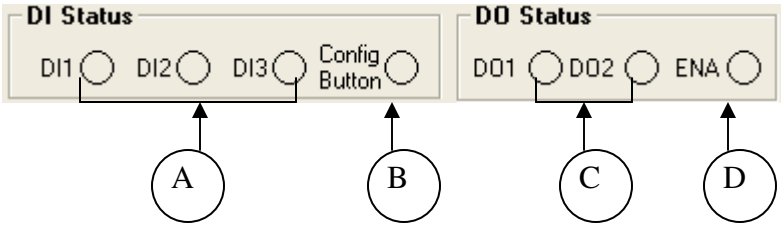
A. Position – Display of current motor position counter value
B. Status – Display of current motor status. Possible values are
ACCEL – acceleration in progress
CONST – constant speed in progress
DECEL – deceleration in progress
-LIM ERROR – minus limit error occurred
+LIM ERROR – plus limit error occurred
C. Clear Error – Clear Error button is used to clear the Limit error status.
D. Limit and Home and Alarm Input Status – Display of limits and home and alarm input status.

Axis Control



A. Target Position – Target position to move to for ABS move.
B. High Speed – High Speed Value. Range from 100 to 400,000.
C. Low Speed – Low Speed Value. Range from 100 to 400,000.
D. Acceleration – Acceleration Value. Range from 10 to 1000.
E. Set Position – Set Position Value and Set Position button
F. DATUM – Absolute move to zero position. Maximum delta from current position to target is 262,143. If greater, then move will not perform.
G. ABS – Absolute move to target position. Maximum delta from current position to target is 262,143. If greater, then the move will not perform.
H. JOG- - Jog to minus direction
I. JOG+ - Jog to plus direction
J. RSTOP – Stop with deceleration
K. ISTOP – Immediate stop without deceleration
L. HOME- - Homing in minus direction
M. HOME+ - Homing in plus direction

DI Status/DO Status/Enable



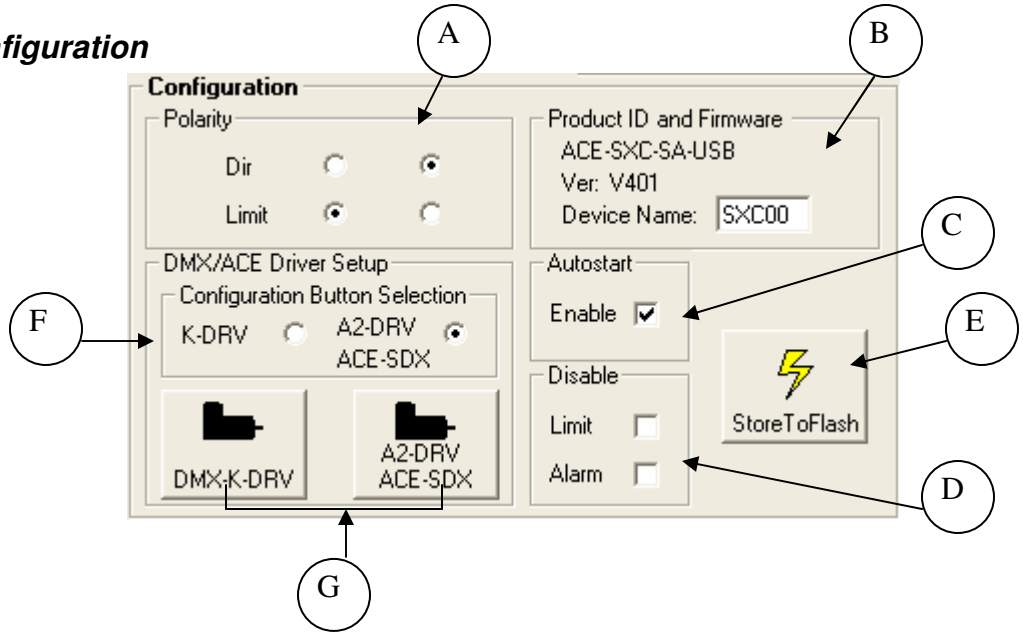
A. Digital Input Status – Display of the three digital input bits. If the digital input pin is grounded, the digital input is turned on.

B. Configuration Button Input Status – Display of the driver configuration button.

C. Digital Output Status – Display of the two digital output status. Digital output can be toggled by clicking on the circle.

D. Enable Output Status – Enable output status. Enable output can be toggled by clicking on the circle. When the enable is on, the driver is enabled and the motor is energized.

Configuration



A. Polarity – Direction polarity can be configured to change the rotational direction. Limit switch input can be configured. Limit switch polarity setting is valid only when the limit switch function is enabled.

B. Product ID and Firmware – Product ID and firmware is shown to confirm the ACE-SXC product. Firmware version loaded on the ACE-SXC is shown. Device name can be changed so that multiple ACE-SXC can be connected on the USB. When entering the new Device Name, make sure to enter in the format of SXCXX where XX ranges from 00 to 99.

C. Autostart – Autostart is for automatic startup of the standalone program. If this is checked, the standalone program startup up automatically when the unit is powered.

D. Limit and Alarm Disable – Limit and alarm switch function can be disabled and the inputs can be used as general purpose inputs.

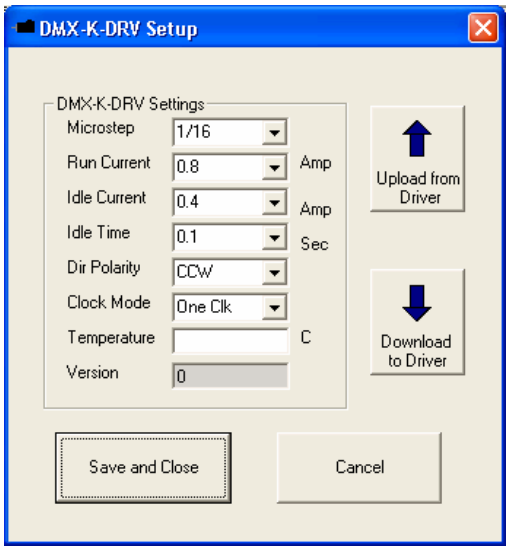
E. Store To Flash – Parameters on the DMX-J can be permanently stored on the flash memory. The unit is powered, the parameter values stored on the flash is loaded and used. Following are parameters that are stored on the flash.

- Direction Polarity
- Limit Polarity
- Autostart Enable
- Disable Limit and Alarm switch function
- Device Name
- Run/Idle Currents and Idle Time

F. Configuration Button Selection – Configuration button is used to download the driver parameters without the use of the Windows PC. With the controller powered and connected to DMX-K-DRV, DMX-A2-DRV, or ACE-SDX, the driver configurations on the controller can be downloaded to the driver by using the configuration button. A2-DRV and ACE-SDX use the same driver parameter settings and grouped together as one type. Select the driver type that will be used with the configuration button and once selected, store to the flash memory so that the driver type and driver parameters to be used will be stored to the controller.

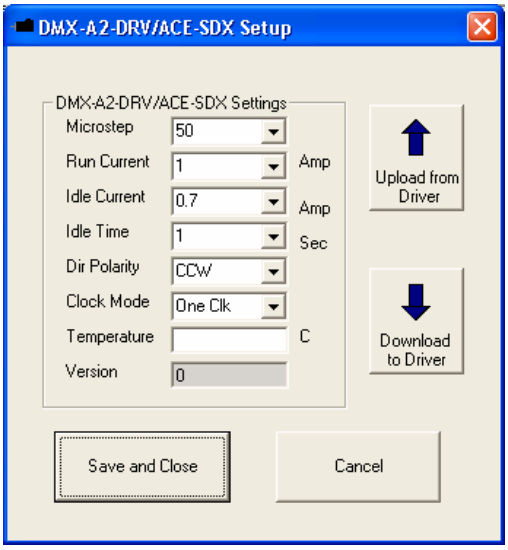
G. DMX-K-DRV and DMX-A2-DRV/ACE-SDX Configuration using Windows GUI – DMX-K-DRV and DMX-A2-DRV/ACE-SDX can be configured from the GUI by selecting one of the buttons.

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



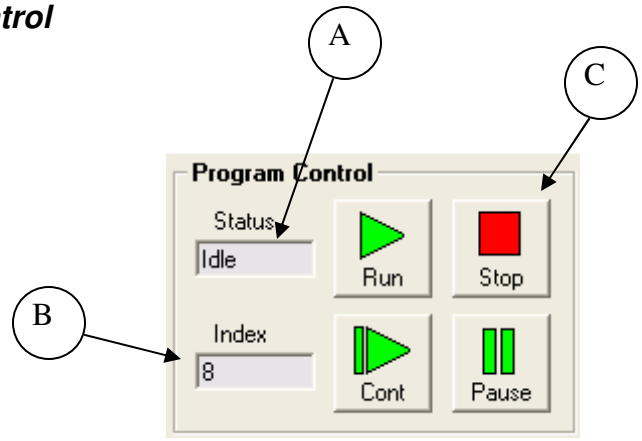
The driver settings can be stored on ACE-SXC so that the parameter values can be used by the configuration button. To save to ACE-SXC, select **Save and Close** button which will download the parameters to the ACE-SXC. To store the downloaded parameters permanently on the ACE-SXC controller, make sure to store to flash before powering down the controller. To close without downloading the parameters to ACE-SXC, select **Cancel** button.

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



The driver settings can be stored on ACE-SXC so that the parameter values can be used by the configuration button. To save to ACE-SXC, select **Save and Close** button which will download the parameters to the ACE-SXC. To store the downloaded parameters permanently on the ACE-SXC controller, make sure to store to flash before powering down the controller. To close without downloading the parameters to ACE-SXC, select **Cancel** button.

Program Control

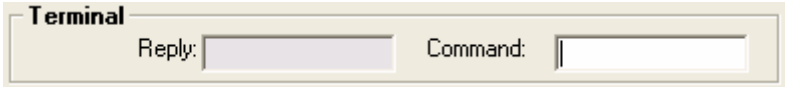


A. Program Status – display of program status. Possible statuses are
Idle – program is not running
Running – program is running
Paused – program is paused

B. Program Index – display of low level program index. This is the index of the low level program index.

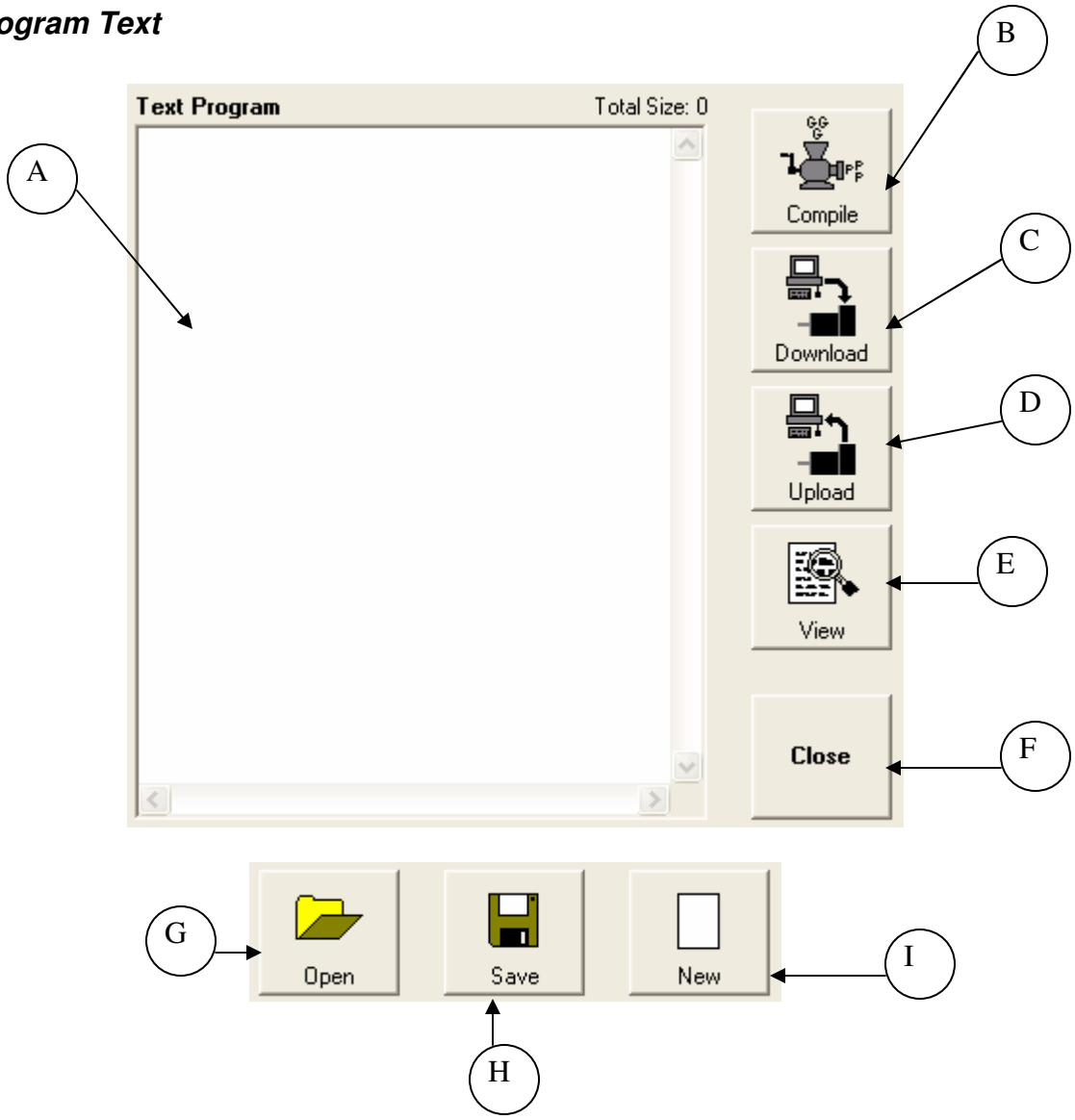
C. Program Control – Program can be RUN, STOP, PAUSED, and CONTINUED.

Program Text



Interactive terminal commands can be sent and replies can be received. See interactive commands for details of the interactive commands.

Program Text

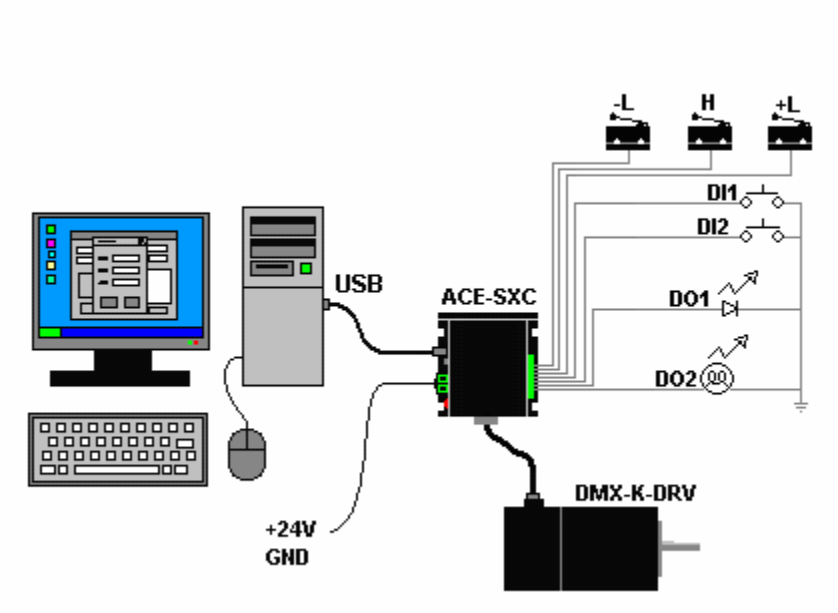


A. Text Program - Program text can be entered and edited here.
B. Compile – Program is compiled to low level
C. Download – Compiled program is downloaded to the controller.
D. Upload – Program in the controller is uploaded and decompiled to high level.
E. View – Compiled low-level program can be viewed.
F. Close – ACE-SXC GUI program is closed.
G. Open – Open program
H. Save – Save program
I. New – Clear the program text section

10. Control Method Overview

ACE-SXC is a single axis step motor controller with USB 2.0 communication. ACE-SXC can work in two modes as shown below.

1) **PC based Control** - As slave to a Windows PC through USB 2.0 communication as shown below.

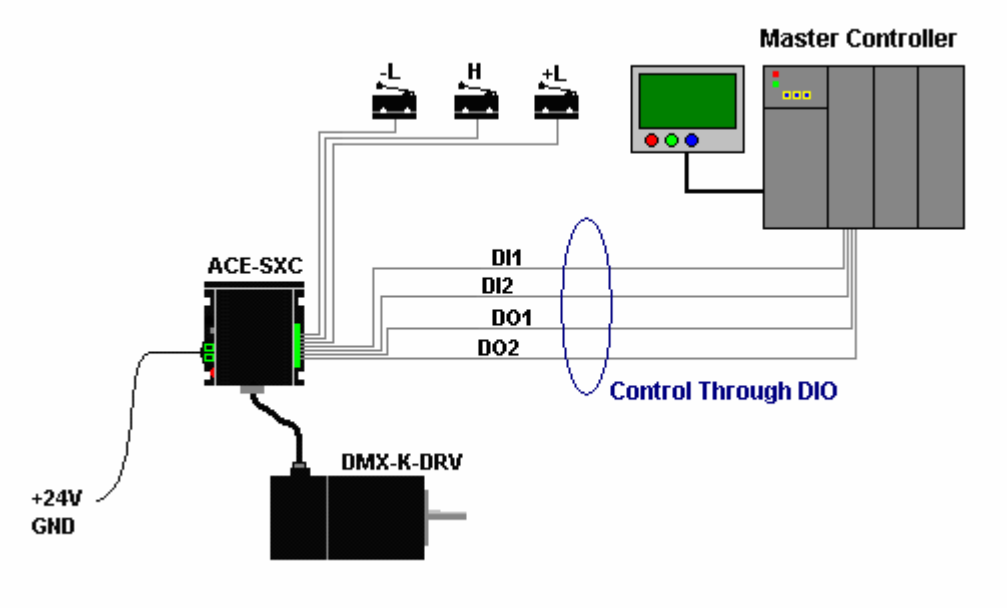


Common Windows programming language (Visual Basic, Visual C++, .Net, LabView, etc.) can be used to develop PC based control system. Requirements for PC based control are the following:

i. Windows XP/VISTA compatible
ii. DLL functions can be called from the program.

Sample VB6, VC++, LabView source code programs can be downloaded from the Arcus website.

2) **Standalone Control** - As a standalone controller with its own program running and ACE-SXC can interface with another controller through the digital IO. Digital IO can be used to perform operations such as trigger a sequence of motions or report the motion completion and error status.



Standalone program is written using a BASIC-like language using the ACE-SXC GUI Windows program that can be downloaded from the Arcus website.

11. PC Based Control

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

ACE-SXC can be controlled from Windows PC through USB 2.0 communication. Each ACE-SXC can have its own unique ID which enables multiple ACE-SXC to be controlled from a Windows PC.

There are four main steps for communicating with the ACE-SXC.

Step A - Identify
Following API functions are available to locate and identify the ACE-SXC that is currently connected to the Windows PC.

```
int PerformaxComGetNumDevices (long *Num)
int PerformaxComGetProductString (long DeviceNum, char *DeviceString, 0)
```

Step B – Open Device
Once the ACE-SXC device is found, it can be opened by the following API function.

```
int PerformaxComOpen(long DeviceNum, Long *Handle)
```

Step C – Communicate
Once the ACE-SXC device is opened, commands can be sent and reply can be received by following API function.

```
int PerformaxComSendRecv (long Handle, char *WriteBuffer,
                          long NumBytesToWrite,
                          long NumBytesToRead, char *ReadBuffer)
```

Commands sent and replies received are ASCII based text strings.

Step D – Close Device
ACE-SXC device communication is closed by the following API function.

```
int PerformaxComClose (long Handle)
```

In order to quickly start the development, we recommend that sample source code be downloaded from the Arcus website. Sample programs are simple terminal program source codes written in VB6, VC++, and LabView. Once the sample source code is downloaded, compiled and communication working, program can easily be modified or copied to custom application program.

Interactive Commands

Following are interactive commands that can be sent from Windows program.

Command	Value Range	Unit	Description	Return
ABORT			Aborts the motion in progress.	OK
ACC			Returns Acceleration value	10 to 1000
ACC=[value]	10 to 1000	msec	Sets Acceleration value	OK
CLR			Clears Limit error	OK
DI[bit]			Returns Digital Input bit status DI1, DI2, DI3 – Digital inputs 1, 2, and 3 DI4 – Configuration Button DI5 – Minus Limit DI6 – Home DI7 – Plus Limit DI8 – Alarm	0 or 1
DL			Return Disable Limit/Alarm	0 or 1
DL=[value]	0– disable 1– enable		Sets Disable Limit and Alarm Bit 0 – Limit Bit 1 – Alarm	OK
DO[bit]			Returns Status of Digital Output bit DO1, DO2 – digital outputs 1 and 2	0 or 1
DO[bit] =[value]	0 – off 1 – on		Sets Digital Output bit DO1, DO2 – digital outputs 1 and 2	OK
EO			Returns Enable status	0 or 1
EO=[value]	0 – off 1 – on		Enables the motor power	OK
H+			Homes the motor in positive direction	OK
H-			Homes the motor in minus direction	OK
HSPD		pps	Returns high speed value	100 to 200000
HSPD=[value]	100 to 200000	pps	Sets high speed value	OK
J+			Jog the motor in positive direction	OK
J-			Jog the motor in minus direction	OK
LSPD		pps	Returns low speed	100 to 200000
LSPD=[value]	100 to 200000	pps	Sets lows speed	OK
MST			Returns the motor status Bit 0 – In Constant Speed Bit 1 – Accelerating Bit 2 – Decelerating Bit 3 – Home Input Switch (Also DI4) Bit 4 – Minus Limit Switch (Also DI5) Bit 5 – Plus Limit Switch (Also DI3) Bit 6 – Minus Limit Error Bit 7 – Plus Limit Error	0 to 255
POL			Returns Polarity Bit 0 – Direction Polarity Bit 1 – Limit Polarity (Valid when Disable Limit is off)	0 to 3
POL=[value]	0 to 3		Sets polarity	OK

PX			Returns the pulse position counter	-2147483648 to 2147483647
PX=[value]	-2147483648 to 2147483647		Sets position counter.	OK
SASTAT			Returns standalone program status 0 – Idle 1 – Running 2 – Paused	0 to 3
SLOAD			Return automatic run mode.	0 or 1
SLOAD=[value]	0 – auto run off 1 – auto run on		Sets automatic run mode	OK
SR=[value]	0 – Stop 1 – Run 2 – Pause 3 – Continue		Controls Standalone program	OK
STOP			Performs ramp down stop if motor is in motion	OK
STORE			Store parameters to Flash	OK
V[var#]			Return variable value. Var # from 1 to 49	-2147483648 to 2147483647
V[var#]=[value]	-2147483648 to 2147483647		Sets variable value. Var # from 1 to 49	OK
X[Value]	Maximum delta from current position to target = 262,143		Performs absolute move	OK

12. Standalone Control

Standalone program is written using the ACE-SXC GUI Windows program. Standalone program is then compiled, and downloaded to the ACE-SXC.

With ACE-SXC, a standalone program can be downloaded and have it automatically run at the power up.

ACE-SXC supports only one task program running.

Programming language is text based BASIC-like language. Example of a Standalone BASIC-like program is shown below.

```
HSPD=30000                  ;***Set High Speed
LSPD=1000                   ;***Set Low Speed
ACC=300                     ;***Set Acceleration
EO=1                        ;***Enable the motor
V1=0                        ;***Set variable 1 to 0
WHILE 1=1                   ;***While loop forever
        IF DI1=1            ;***If DI1 is on
                GOSUB 1             ;***Go to Subroutine 1
        ELSEIF DI2=1        ;***If DI2 is on
                GOSUB 2             ;***Go to Subroutine 2
        ELSEIF DI3=1        ;***If DI3 (-Limit) is on
                HSPD=5000           ;***Set High Speed
                HOMEX-              ;***Home in minus direction
        ELSEIF DI5=1        ;***IF DI5 (+Limit) is on
                HSPD=20000          ;***Set High Speed
                X0                  ;***Move back to 0
        ENDIF               ;***End of If statements
ENDWHILE                    ;***End of while.  Go back to While
END

SUB 1                       ;***Begin subroutine 1
        HSPD=50000          ;***Set High Speed
        X2000               ;***Move to 2000
ENDSUB                      ;***End of sub routine, go back.

SUB 2                       ;***Begin subroutine 2
        HSPD=30000          ;***Set High Speed
        XV1                 ;***Move to variable 1 location
        V1=V1-1000          ;***Decrease variable 1 by 1000
ENDSUB                      ;***End of sub routine, go back
```

As shown in the example program, language is simple text based and easy to write and understand. Programming language supports many advanced programming capabilities such as

- Support of WHILE, IF, ELSE, ELSEIF, REPEAT conditionals
- Support of subroutines
- Support of variables and math and bit operations
- Support of move to variable location

When writing a standalone program, following general rules must be followed.

1) Each program must have an END to indicate the end of the program.
2) Declaration of the subroutine must be done after the END statement.
3) Comments in the program are started with **;** character.

13. Standalone Programming Language

Speed Commands

	Value	Description
HSPD=[value]	100 to 200000 Variable	Sets High Speed Example: HSPD=1000, HSPD=V1
LSPD=[value]	100 to 200000 Variable	Sets Low Speed Example: LSPD=500, LSPD=V2
ACC=[value]	10 to 1000 Variable	Sets Acceleration Time Example: ACC=300, ACC=V1

Digital Output Commands

	Value	Description
EO=[value]	0 to 1 Variable	Sets motor enable Example: EO=1, EO=0, EO=V1
DO[bit]=[value]	0 to 1 Variable	Sets digital output bit. There are two outputs. Example: DO1=1, DO2=0
DO=[value]	0 to 3 Variable	Sets digital output. There are two outputs. Example: DO=2, DO=3

Move Commands

	Value	Description
JOGX+		Jogs motor to positive direction
JOGX-		Jogs motor to negative direction
HOMEX+		Homes motor to positive direction
HOMEX-		Homes motor to negative direction
STOPX		Stops the motor with deceleration
ABORT		Aborts the motion
X[value]	Maximum delta from current position to target = 262,143 [value] can be either a numerical value or a variable.	Move to absolute target position. Maximum difference between current and target position must be equal or less than 262,143. Example: X1200, XV1

Conditional and Program Flow Control Commands

	Value	Description
END		Indicates end of the program.
SUB [value]	1 to 10	Start of subroutine. There are 1 to 10 subroutines that can be used. Example: SUB 1, SUB 10
ENDSUB		Indicates end of subroutine
IF [Arg1][Comp][Arg2]		If conditional statement. ENDIF statement must be used to indicate end of IF statement.
ELSE		Must be used with IF.
ELSEIF [Arg1][Comp][Arg2]		Must be used as a part of IF statement,
ENDIF		Indicates end of IF/ELSEIF statement. Must be used with IF/ELSEIF statement.
WHILE [Arg1][Comp][Arg2]		While loop conditional statement. Must be used with ENDWHILE statement.
ENDWHILE		Indicates end of WHILE loop and goes back to the matching WHILE statement. Must be used with WHILE statement.
REPEAT		Repeat loop. Must be used with LOOPWHILE statement
LOOPWHILE [Arg1][Comp][Arg2]		LOOPWHILE conditional statement. Must be used with REPEAT statement.

[Arg1][Arg2] can be any of the following	[Comp] can be any of the following
Numerical Value	=
Variable	<
PX	<=
DI	>
DI[bit]	>=
DO	!=
DO[bit]	
EO	
MSTX	

Variable Commands

	Value	**Description**
V[var#] = [value]	Value can be any of the following: Numerical Value DI[bit] DO[bit] EO PX MSTX V[var#]	Variable Assignment
V[var#]=[value1][Op][Value2]	Value1 and Value2 can be any of the following: Numerical Value DI[bit] DO[bit] EO PX MSTX V[var#]	Variable Operation. Operation can be any of the following: + Addition - Subtraction * Multiplication / Division >> Bit shift Left << Bit Shift Right & Bit AND | Bit OR

Miscellaneous Commands

	Value	**Description**
DELAY=[value]	1 to 1000 Variable	Performs delay. Each unit is 10 msec. This means DELAY=100 will delay for 1 second. Example: DELAY=10, DELAY=V1
PX=[value]	-2147483648 to 2147483647	Sets current position. Example: PX=1000, PX=V2

Standalone Example Program 1

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
X0                  ;* Move to 1000
END                 ;* End of the program
```

Standalone Example Program 2

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
        X0          ;* Move to 1000
ENDWHILE            ;* Go back to WHILE statement
END
```

Standalone Example Program 3

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
        X0          ;* Move to 1000
        V1=V1+1     ;* Increment variable 1
ENDWHILE            ;* Go back to WHILE statement
END
```

Standalone Example Program 4

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
                X0      ;* Move to 1000
        ENDIF
ENDWHILE        ;* Go back to WHILE statement
END
```

Standalone Example Program 5

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

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

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

Standalone Example Program 6

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. Note that in order to have digital input 3 and 5 working as digital input instead of limit switches, disable the limit switch function.

```
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
```

Contact Information

Arcus Technology, Inc.

www.arcus-technology.com