

# ACE-SDE

## Advanced Stand-Alone Step Motor Controller and Driver With USB 2.0/RS-485 communication



COPYRIGHT © 2008 ARCUS,  
ALL RIGHTS RESERVED

First edition, January 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:**

- 1.10 – First revision
- 1.11 – Added joystick SA commands, updated default device name, added JS, RT, DRVMS, DRVRC, DRVIC, DRVIT, VER, RR, RW, R2, R4 commands, added driver configuration
- 1.12 – Updated firmware, software compatibility, added latch SA commands, added SNL, STORE, SA commands, V50-V100 store to flash explanation
- 1.13 – Added driver max pulse support, updated IO drawings, updated typical setup drawing, updated DIO descriptions, updated stand-alone programming section, updated joystick control diagram, updated USB communication section, updated JOYNI standalone command, added (L) to GUI section, added interactive ASCII command explanation
- 1.14 – Added SSPD section to GU, added to explanation on SSPD move
- 1.15 – Updated GUI labels, added DN, STORE ASCII commands, updated ASCII command ordering, added SLOAD parameter to Store To Flash section, updated ordering of EX command on SA commands, added EDIO to SA commands
- 1.16 – Updated revision history, added latch input/z-index to MST, added DIO control mode, digital input resistor, removed encoder termination, GUI status and control sections, DATUM always moves to 0, added SNL speed calculation for charts, removed pulse polarity, already in motion situations, ignore PX when SNL is enabled (latch input), ASCII response appended with CR instead of NULL, RS-485 communication section, added DI/DO bit commands to ASCII table, removed EDIO SA command, added sample SA programs, start at pg 1, polarity/latch section made to heading
- 1.17 – Updated SNL ratio spec
- 1.18 – Added note for micro-step driver configuration, updated digital output diagram (internal GND), added peak current annotation

**Firmware Compatibility:**  
V219

**Software Compatibility:**  
V214

## Table of Contents

1. Introduction.....	8
2. Dimensions .....	9
Top View .....	9
3. Connections.....	10
4 pin Connector Information.....	10
2 pin Connector Information.....	10
28 Pin 2mm Connector Pin Outs .....	11
4. Electrical Specifications.....	13
Power Requirement.....	13
Communication Interfaces .....	13
+Lim, -Lim, Home Inputs, Latch, Digital Inputs.....	13
Digital Outputs.....	13
Analog Inputs.....	13
Limit/Home/Latch Sensors and Digital Input Connections.....	14
Digital Output Connections .....	15
Differential / Single-ended Encoder Input Connection .....	16
5. Getting Started .....	17
Typical Setup .....	17
Windows GUI.....	18
Main Control Screen .....	19
A. Status.....	20
B. Control.....	22
C. Digital Input / Output .....	24
D. Analog Inputs.....	24
E. Product Information.....	25
F. Setup .....	26
G. Terminal .....	29
H. Standalone Program File Management .....	30
I. Standalone Program Editor.....	30
J. Standalone Program Compile/Download/Upload/View .....	31
K. Variable Status .....	32
L. Program Control .....	32
M. On-The-Fly Speed Change.....	33
6. Motion Control Overview.....	34
Motion Profile.....	34
On-the-fly Speed Change.....	36
Analog Inputs.....	38
Joystick Control .....	38
Digital Inputs / Outputs.....	39
Motor Power .....	39
Polarity.....	40
Positional Moves.....	40
Jogging.....	40
Stopping Motor .....	40

Homing .....	40
Motor Position .....	41
Motor Status.....	41
Limit Inputs.....	42
Latch Input .....	42
StepNLoop Closed Loop Control .....	43
Device Number .....	44
Baud Rate Setting .....	45
Sync Output .....	45
Broadcasting over RS-485 .....	46
Response type selection .....	46
Micro-step Driver Configuration .....	47
Standalone Programming.....	48
Storing to Flash .....	48
7. Communication – USB .....	49
USB Communication API Functions.....	49
USB Communication Issues .....	50
8. Communication – RS-485 (ASCII) .....	51
Communication Port Settings .....	51
ASCII Protocol.....	51
9. Communication - DIO .....	52
DIO Latency.....	52
Setting Up DIO Parameters .....	52
Using DIO.....	55
Important DIO Notes .....	55
10. ASCII Language Specification .....	57
11. Standalone Language Specification.....	62
; .....	62
ABORTX .....	62
ABS .....	63
ACC .....	63
AI[1-2] .....	64
DELAY .....	64
DI .....	65
DI[1-6] .....	65
DO.....	66
DO[1-2].....	67
DRVIC .....	68
DRVIT .....	68
DRVMS .....	69
DRVRC.....	69
ECLEARX .....	69
ECLEARSX.....	70
ELSE .....	70
ELSEIF .....	70
END .....	72

ENDIF.....	72
ENDSUB.....	73
ENDWHILE .....	73
EO .....	74
EX .....	74
GOSUB.....	75
HOMEX[+ or -] .....	75
HSPD .....	76
IF .....	77
INC.....	78
JOGX[+ or -].....	78
JOYDIS.....	79
JOYENA.....	79
JOYHS .....	79
JOYDEL .....	79
JOYNO .....	80
JOYNI.....	80
JOYPI.....	80
JOYPO .....	81
JOYTOL .....	81
LSPD.....	81
LT.....	82
LTE .....	82
LTP .....	82
LTS .....	83
MSTX .....	83
PX .....	84
PS .....	84
RW .....	85
RWSTAT .....	85
SCV.....	85
SL.....	86
SLSX.....	86
SSPD.....	87
SSPDM .....	87
STOPX.....	88
STORE.....	88
SYNCCFG .....	89
SYNCOFF.....	89
SYNCON .....	90
SYNCPOS.....	90
SYNCSTAT.....	92
SYNCTIME .....	92
SUB.....	93
V[1-100].....	94
WAITX .....	95

---

WHILE.....	96
X.....	97
ZHOMEX[+ or -].....	97
ZOMEX[+ or -].....	98
Standalone Example Program 1.....	99
Standalone Example Program 2.....	99
Standalone Example Program 3.....	99
Standalone Example Program 4.....	100
Standalone Example Program 5.....	100
Standalone Example Program 6.....	101

# 1. Introduction

## ACE-SDE Features

- USB 2.0 communication
- RS-485 ASCII communication at 9600, 19200, 38400, 57600, 115200 bps
- Digital IO communication (opto-isolated inputs and outputs):
  - 4 bit motion profile select inputs
  - One start motion input
  - One abort/clear motion input
  - One in position output
  - One error output
- A/B/Z Differential Encoder inputs
- StepNLoop closed loop control
- Opto-isolated +Limit/-Limit/Home inputs
- Opto-isolated high speed position capture Latch input
- S-curve or trapezoidal acceleration profile control
- On-the-fly speed change
- Two 10-bit analog inputs
- Joystick control
- Stand-alone programmable
- Homing routine using:
  - Home input only
  - Z index encoder channel only
  - Home and Z index encoder channel
- Built-in microstep driver
  - 2 to 500 microstep driver
  - Maximum 3A peak current driver
  - 1 MHz PPS support
- 12VDC to 48VDC input

## Contacting Support

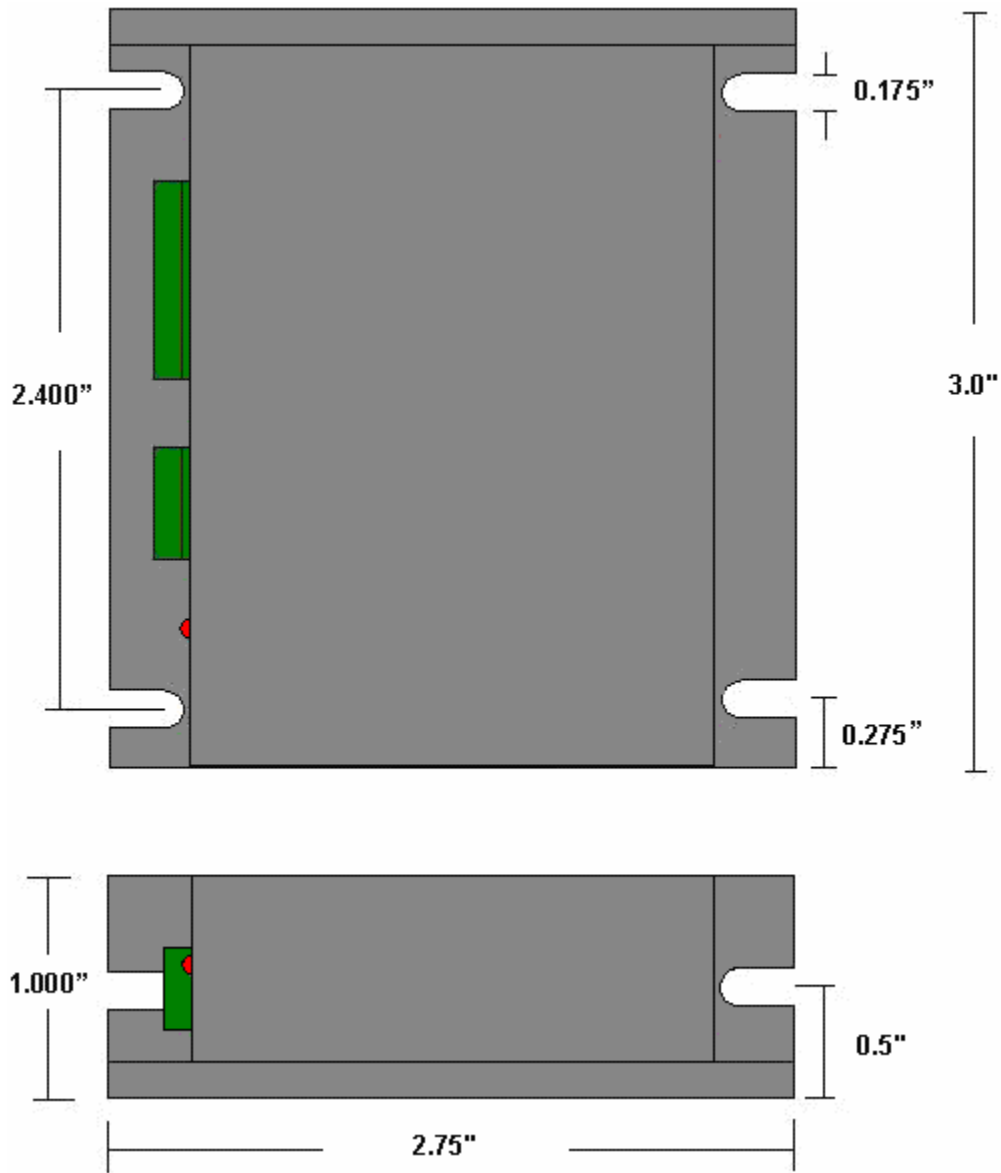
For technical support contact: [support@arcus-technology.com](mailto:support@arcus-technology.com).

Or, contact your local distributor for technical support.

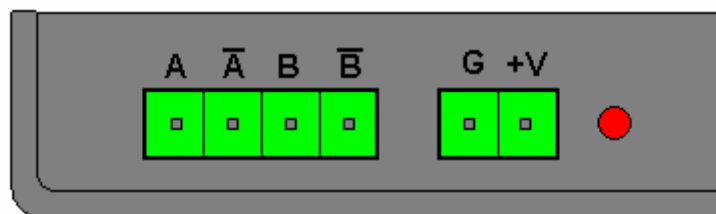


## 2. Dimensions

*Top View*



### 3. Connections



#### 4 pin Connector Information

Pin #	Name	Description
1	A	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

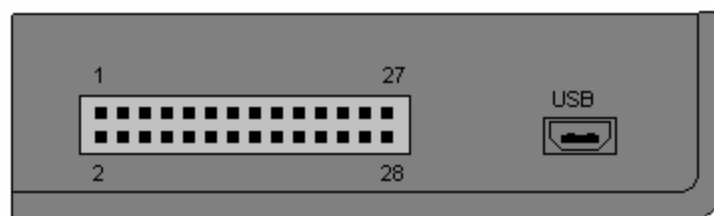
Mating Connector Description: 4 pin 0.2" (5.08mm) connector  
 Mating Connector Manufacturer: On-Shore  
 Mating Connector Manufacturer Part: EDZ950/4

#### 2 pin Connector Information

Pin #	Name	Description
1	G	Ground
2	V+	Power Input +12 to +48VDC

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

**Note: Other 5.08mm compatible connector can be used.**



### ***28 Pin 2mm Connector Pin Outs***

<b>Pin #</b>	<b>Name</b>	<b>Description</b>
1	PWR OUT	+12-48VDC from 2-pin 5.08mm main power input
2	OPTO IN	+12-24VDC opto-isolator supply for limit, home, latch, and digital inputs
3	-LIM	Minus limit opto-isolated input
4	+LIM	Plus limit opto-isolated input
5	LATCH	Latch opto-isolated input
6	HOME	Home opto-isolated input
7	DI2 (Abort/Clear)	Aborts move if motor is in motor, or clears error if motor is in error state. Can be used as general-purpose digital input if DIO control mode is disabled.
8	DI1 (Start)	Triggers move DIO command. Can be used as general-purpose digital input if DIO control mode is disabled.
9	DI4 (Select 2)	2 <sup>nd</sup> bit of motion select. Can be used as general-purpose digital input if DIO control mode is disabled.
10	DI3 (Select 1)	LSB of motion select. Can be used as general-purpose digital input if DIO control mode is disabled.
11	DI6 (Select 4)	MSB of motion select. Can be used as general-purpose digital input if DIO control mode is disabled.
12	DI5 (Select 3)	3 <sup>rd</sup> bit of motion select. Can be used as general-purpose digital input if DIO control mode is disabled.
13	DO2 (Alarm)	Enabled when motor is in an error state. Can be used as general-purpose digital output if DIO control mode is disabled.
14	DO1 (In Pos)	Enabled when motor is in position. Can be used as general-purpose digital output if DIO control mode is disabled.
15	GND	GND of controller
16	5V	5V output from controller
17	/EA	Differential encoder /A channel input
18	EA	Differential encoder A channel input
19	/EB	Differential encoder /B channel input

20	EB	Differential encoder B channel input
21	/EZ	Differential encoder /Z channel input
22	EZ	Differential encoder Z channel input
23	GND	GND of controller
24	5V	5V output from controller
25	AI2	Analog input 2
26	AI1	Analog input 1
27	485-	RS-485 – signal
28	485+	RS-485 + signal

Mating Connector Description:	28 pin 2mm dual row connector
Mating Connector Manufacturer:	HIROSE
Mating Connector Housing Part Number:	DF11-28DS-2C
Mating Connector Pin Part Number:	DF11-2428SC

## 4. Electrical Specifications

### ***Power Requirement***

Regulated Supply Voltage Range: **+12 to +48 VDC**

### ***Communication Interfaces***

USB 2.0 : **Arcus ASCII command support**

RS-485 : **RS-485 Arcus ASCII command support (9600, 19200, 38400, 57600, 115200 bps)**

### ***+Lim, -Lim, Home Inputs, Latch, Digital Inputs***

Type: **Opto-isolated sinking inputs**  
 Opto voltage supply input: **+12 to +24 VDC**

### ***Digital Outputs***

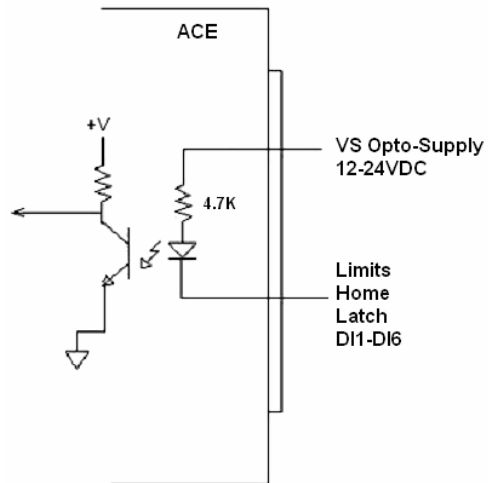
Type: **Opto-isolated open-collector sinking outputs**  
 Max voltage at collector: **+24 VDC**  
 Max current at 24VDC **100 mA**

### ***Analog Inputs***

Type: **Voltage**  
 Voltage input **+0 to +5VDC**  
 Max current **25 mA**

## Limit/Home/Latch Sensors and Digital Input Connections

Limit, Home and Latch sensors are opto-isolated inputs as shown below:

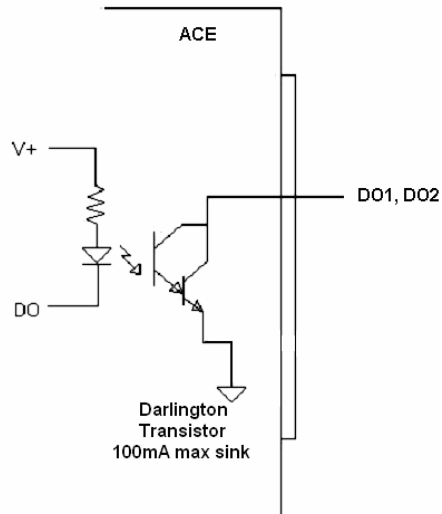


Connect the opto-supply using 12-24VDC power input. To trigger the Limit, Home, Latch, Digital Inputs, sink the line to ground of the power.

The ACE-SDE has 6 digital inputs that are used for DIO control mode. These can also be used for general purpose if DIO control mode is disabled.

## Digital Output Connections

Digital outputs are opto-isolated open-collector outputs using Darlington transistor that can source up to 100mA current at recommended maximum voltage of 24VDC.



ACE-SDE has 2 digital outputs that are used for DIO control. These can also be used for general purpose if DIO control mode is disabled.

### ***Differential / Single-ended Encoder Input Connection***

ACE-SDE supports both single-ended and differential quadrature encoder inputs.

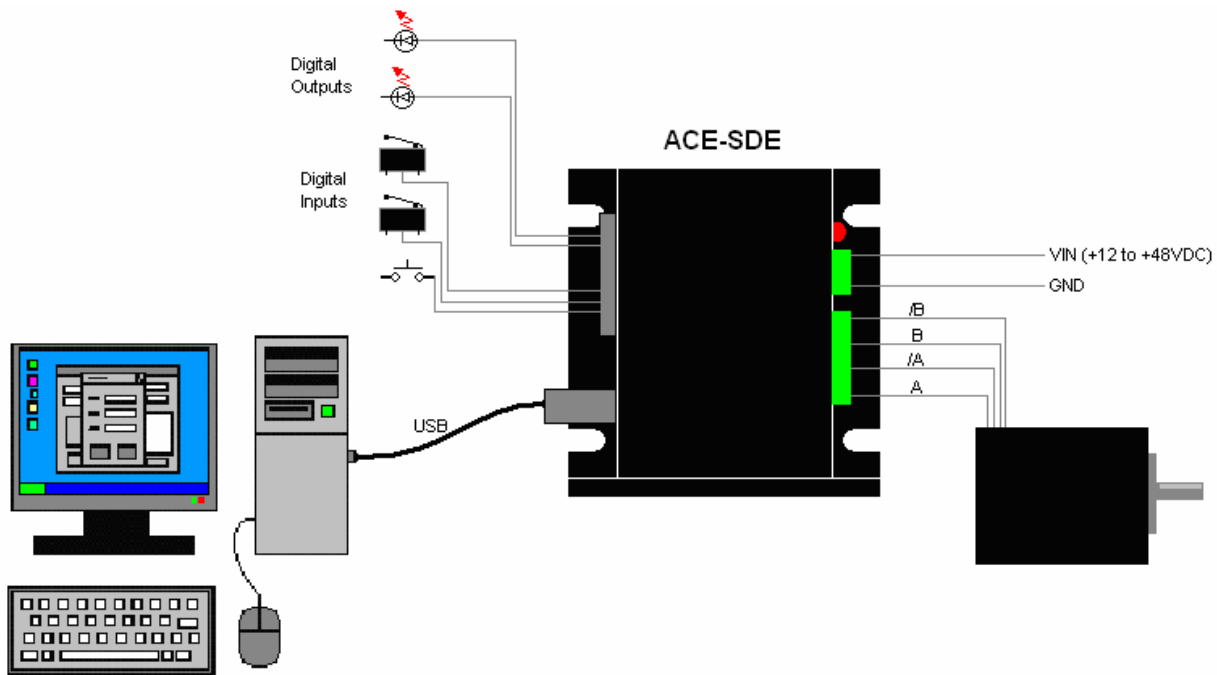
When using single-ended encoders, use the /A, /B, and /Z inputs.

+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.



## 5. Getting Started

### Typical Setup



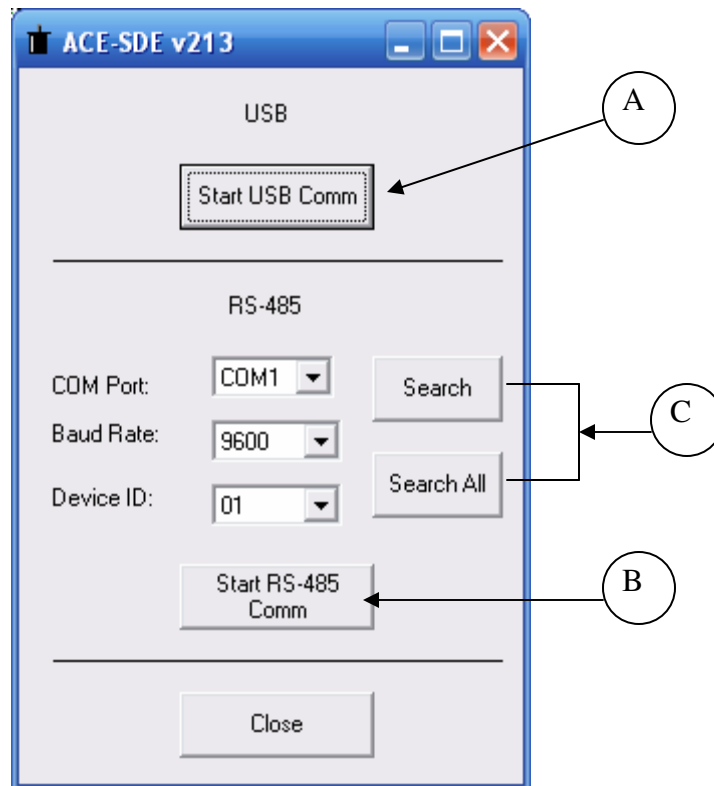
**Important Note:** In order to communicate with ACE-SDE through USB, proper driver must be installed first. Before connecting the ACE-SDE device or running any program, please go to the Arcus web site and download the USB driver installation instruction and run the USB Driver Installation Program.

## Windows GUI

ACE-SDE comes with Windows GUI program to test, program, compile, download, and debug the controller.

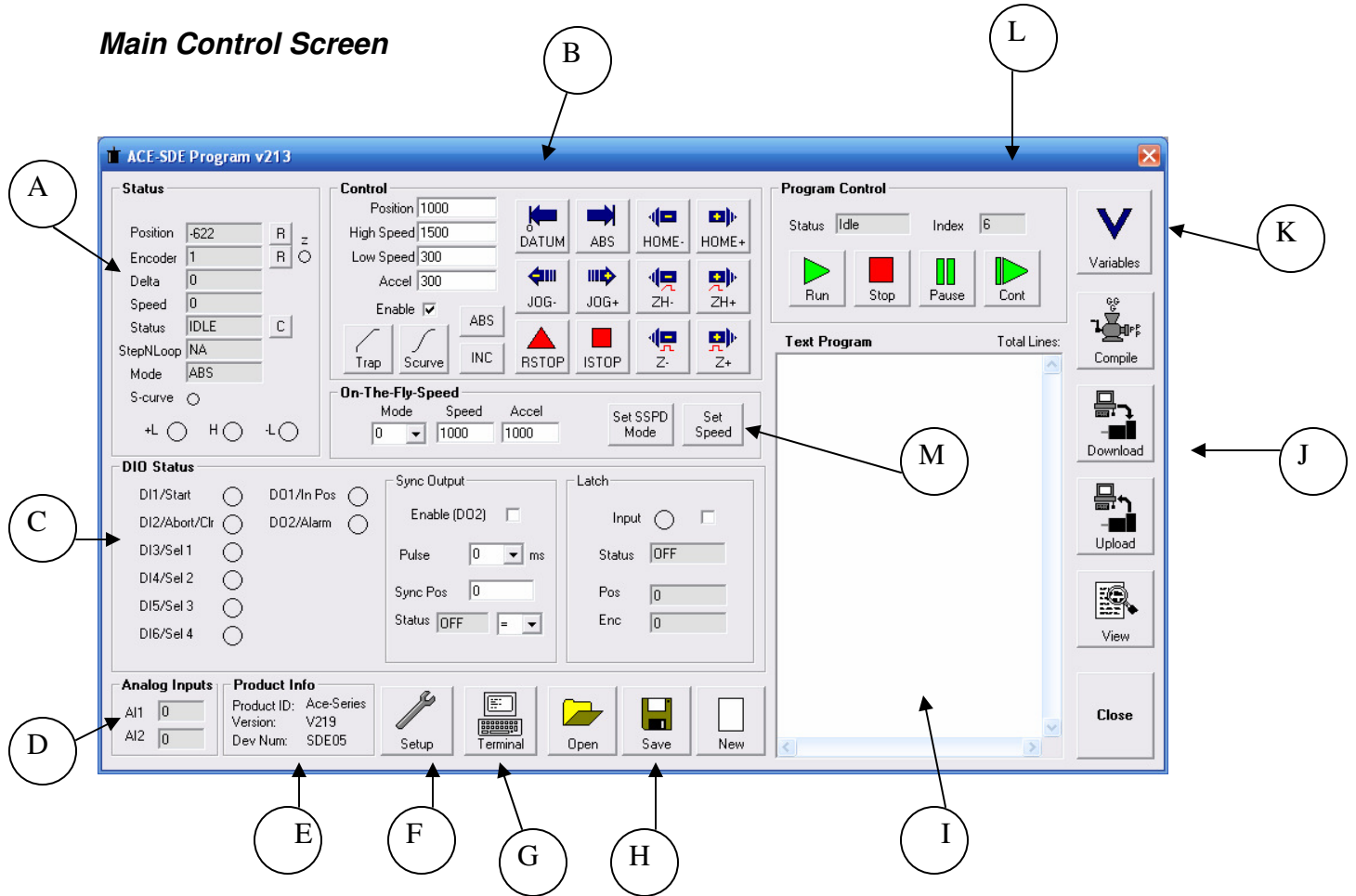
Make sure that the USB driver is installed properly before running the controller.

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

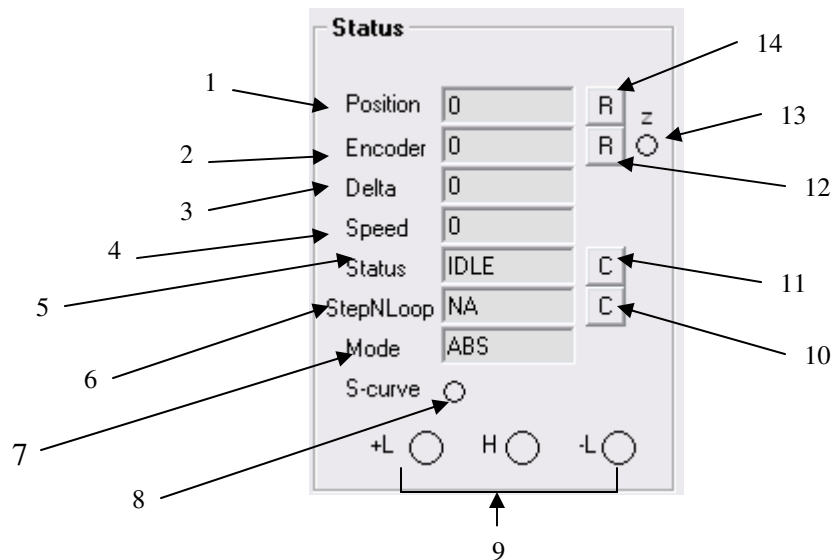


- A. Open USB Communication.
- B. Open RS-485 communication.
- C. If communication port or the baud rate is not known for RS-485, use these buttons to search for the device.

## Main Control Screen



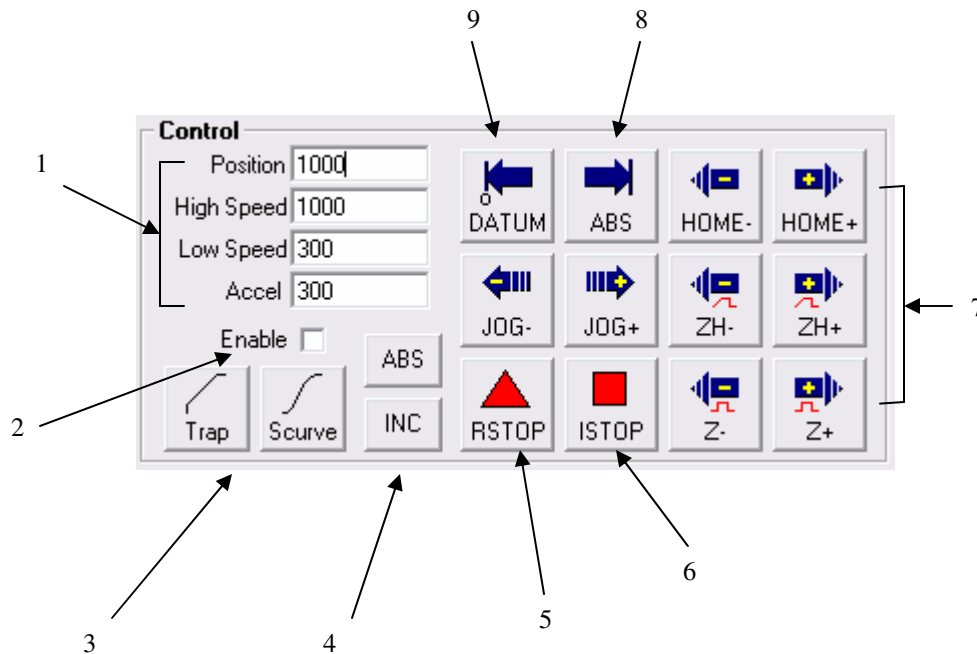
## A. Status



- 1. Pulse Counter** – displays the current pulse position counter. When StepNLoop is used, this displays the current target position
- 2. Encoder Counter** – displays the current encoder position counter.
- 3. Delta Counter** – valid only for StepNLoop. Displays 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:
  - IDLE – motor is not moving
  - ACCEL – motion is in acceleration
  - DECEL – motion is in deceleration
  - CONST – motion is in constant speed
  - LIM ERR – minus limit error
  - +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:
  - NA – StepLNoop is disabled
  - IDLE – motor is not moving
  - MOVING – target move is in progress
  - JOGGING – jog move is in progress
  - HOMING – homing is in progress
  - Z-HOMING – homing using Z-index channel in progress
  - ERR-STALL – StepNLoop has stalled.
  - ERR-LIM – plus/minus limit error

- 7. Move Mode** – displays current move mode
  - ABS – all the move commands by X[pos] command will be absolute moves
  - 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.

## B. Control



### 1. Target Position/Speed/Accel

Target Position – use this to set the target position. For normal open loop mode, this position is the pulse position and when StepNLoop is enabled this target position is in encoder position.

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

Accel – acceleration value in milliseconds

**2. Enable Driver Power** – use this button to enable and disable the power to the microstep driver.

**3. Select Acceleration Mode** – use these buttons to select trapezoidal or S-curve acceleration mode.

**4. Select Move Mode** – use these buttons to select absolute or incremental move mode.

**5. Ramp Stop** – use this button to stop the motion with deceleration.

**6. 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.*

**7. Perform Homing** – Three different homing is available

HOME – homing is done using only the home switch.

ZH – homing is done using the home switch first and then the Z index channel of the encoder.

Z – homing is done only using the Z index channel of the encoder.

- 8. 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.
- 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).

## C. Digital Input / Output

The screenshot shows the 'DIO Status' control panel. It is divided into four main sections: Digital Input Status, Digital Out Status and Control, Sync Output, and Latch. Arrows point from numbers 1 to 4 to specific features: 1 points to the Digital Input Status section, 2 points to the Digital Out Status and Control section, 3 points to the Sync Output section, and 4 points to the Latch section.

**DIO Status**

**Digital Input Status:** DI1/Start, DI2/Abort/Clr, DI3/Sel 1, DI4/Sel 2, DI5/Sel 3, DI6/Sel 4. Each has a circular button.

**Digital Out Status and Control:** D01/In Pos, D02/Alarm. Each has a circular button.

**Sync Output:** Enable (D02) checkbox, Pulse (0 ms), Sync Pos (0), Status (OFF) with a dropdown menu.

**Latch:** Input checkbox, Status (OFF), Pos (0), Enc (0).

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** – digital outputs can be triggered
4. **Latch** - encoder and pulse positions can be captured/latched with an input trigger.

## D. Analog Inputs

The screenshot shows the 'Analog Inputs' control panel. It contains two input fields: AI1 and AI2, both showing the value 0.

**Analog Inputs**

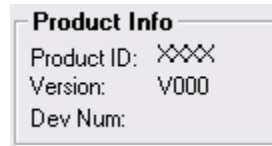
AI1: 0

AI2: 0

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



## E. Product Information

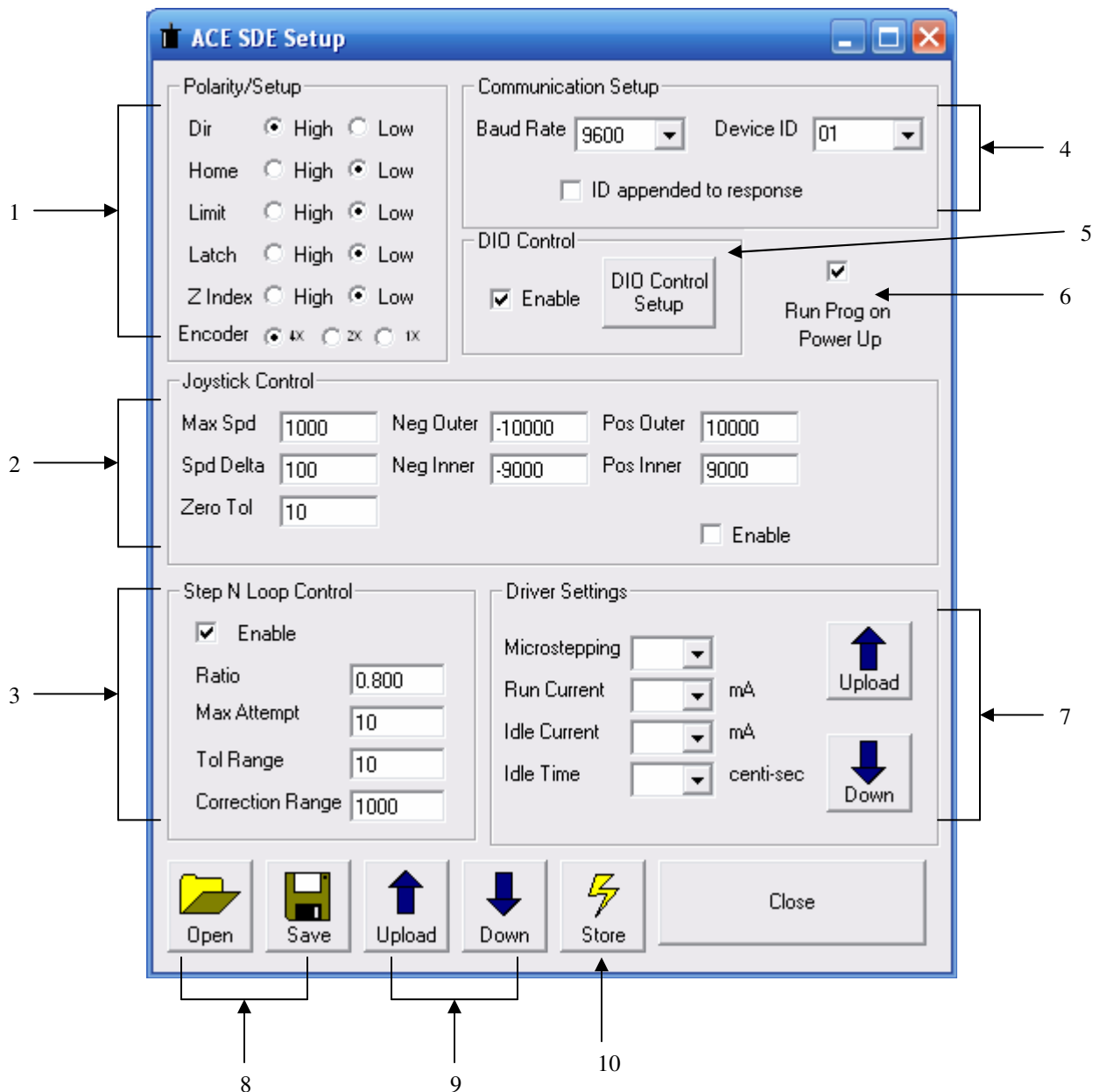


Product information and firmware version and device number is displayed. Device number can be changed from the setup screen to support multiple devices on the USB or RS-485 communication.

## F. Setup



When setup button is selected, setup dialog box is shown as below.



## 1. Polarity Setup

Dir – direction of the motion (clockwise or counter-clockwise )can be configured.

Home – home input polarity can be configured

Limit – limit input polarity can be configured

Latch – latch input polarity can be configured

Z-Index – Encoder Z index channel can be configured

Encoder – encoder multiplication factor can be configured as 1X, 2X, or 4X.

## 2. Joystick Control – ACE-SDE allows joystick control using the analog input 1.

See joystick control section for details of the joystick parameters.

## 3. StepNLoop Control – Using the encoder input, StepNLoop control allows closed loop position verification and correction for the moves.

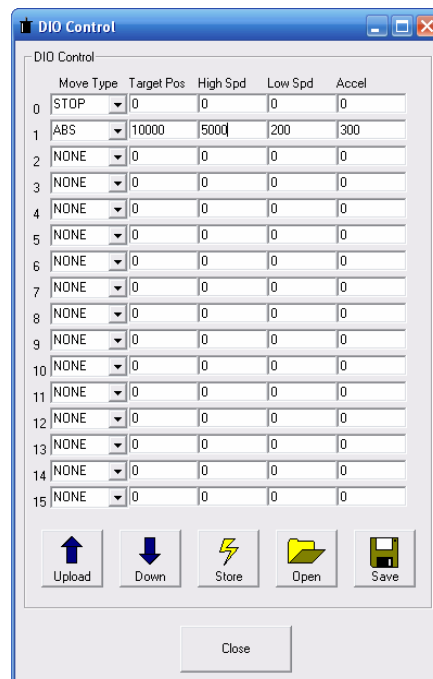
See StepNLoop control section for details.

## 4. Communication Setup – RS-485 communication baud rate can be selected to support different communication speed.

Device ID configuration allows multiple devices on the RS-485 or USB communication network.

ID append to response is used for RS-485 communication for adding the device ID to the response.

## 5. DIO Control – Digital IO motion control allows motion profiles to be triggered through the digital inputs. See DIO motion control section for details. Following dialog box is shown for the DIO motion control.



The DIO Control dialog box contains a table with 16 rows (0-15) and 5 columns: Move Type, Target Pos, High Spd, Low Spd, and Accel. Row 0 is set to STOP with all values at 0. Row 1 is set to ABS with Target Pos at 10000, High Spd at 5000, Low Spd at 200, and Accel at 300. Rows 2-15 are all set to NONE with all values at 0. At the bottom are buttons for Upload, Down, Store, Open, Save, and a Close button.

	Move Type	Target Pos	High Spd	Low Spd	Accel
0	STOP	0	0	0	0
1	ABS	10000	5000	200	300
2	NONE	0	0	0	0
3	NONE	0	0	0	0
4	NONE	0	0	0	0
5	NONE	0	0	0	0
6	NONE	0	0	0	0
7	NONE	0	0	0	0
8	NONE	0	0	0	0
9	NONE	0	0	0	0
10	NONE	0	0	0	0
11	NONE	0	0	0	0
12	NONE	0	0	0	0
13	NONE	0	0	0	0
14	NONE	0	0	0	0
15	NONE	0	0	0	0

Buttons: Upload, Down, Store, Open, Save, Close

**6. Run Prog On Power Up** – Run program on power up allows the standalone program to start up automatically when the controller is powered.

**7. Driver Setting** – Following microstep driver settings can be configured:

Microstep – 2 to 500 microsteps

Run Current – 100mA to 3Amp

Idle Current – 100mA to 3Amp

Idle Time – 1 to 100 centi-second (10 centi-second = 1 second)

**8. Open/Save** – Configuration values can be saved to a file and read from a file.

**9. Upload/Download** – Configuration values can be uploaded and downloaded.

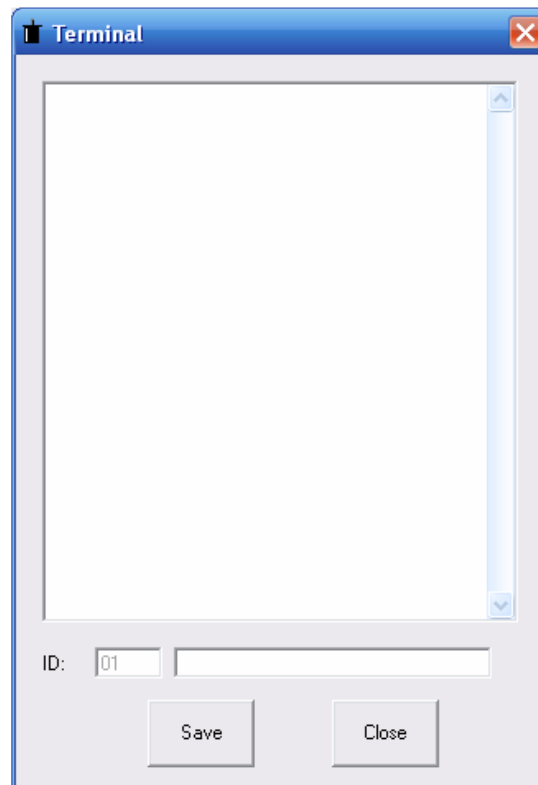
Note that if the configuration values are changed, it needs to be downloaded to take affect.

**10. Store** – The downloaded parameters can be permanently stored on the non-volatile memory.

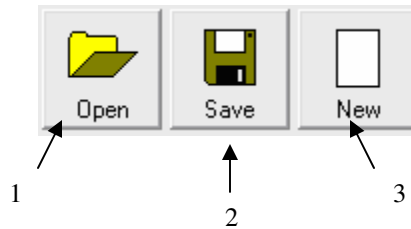
## G. Terminal



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

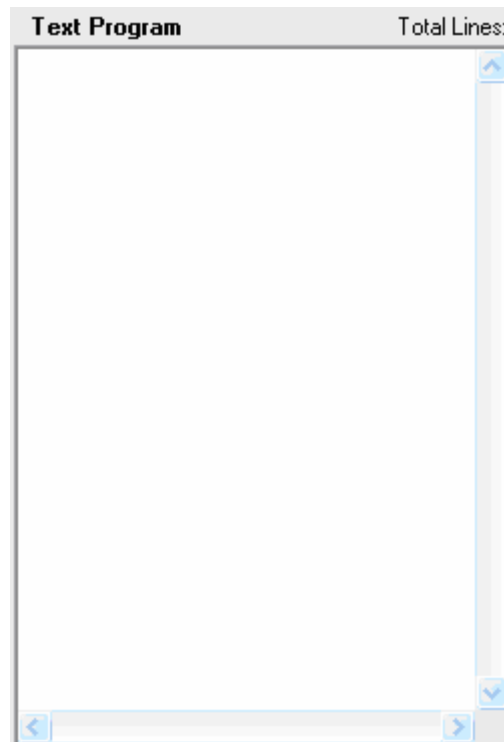


## H. Standalone Program File Management

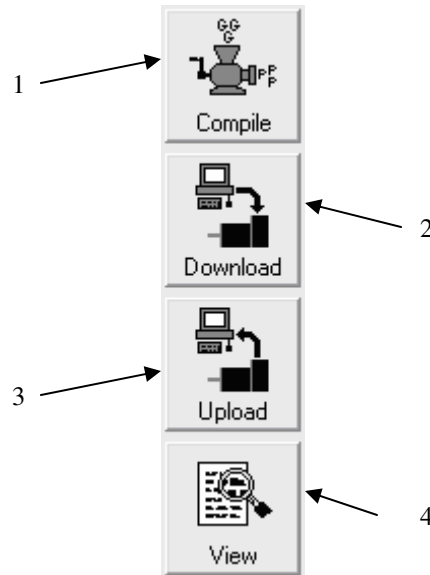


1. Open – Open standalone program
2. Save – Save standalone program
3. New – Clear the standalone program editor

## I. Standalone Program Editor

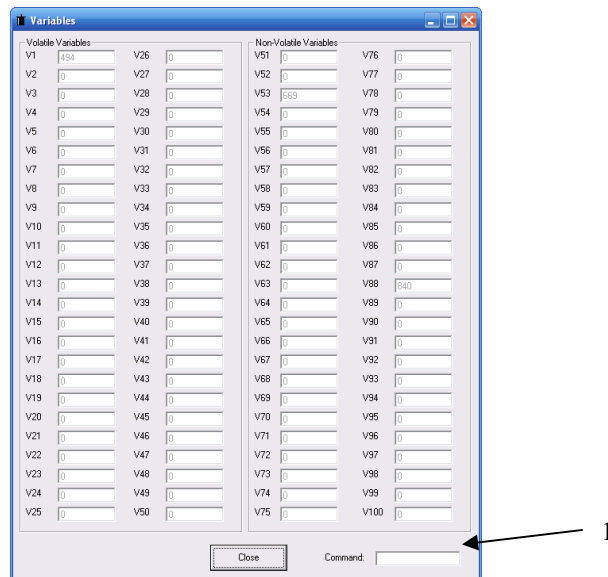


## J. Standalone Program Compile/Download/Upload/View



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

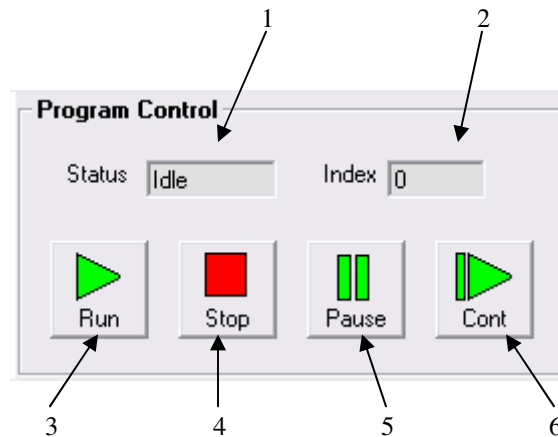
## K. Variable Status



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

1. Command line to set variables. To write to variable, use V[1-100] = [value] syntax.

## L. Program Control



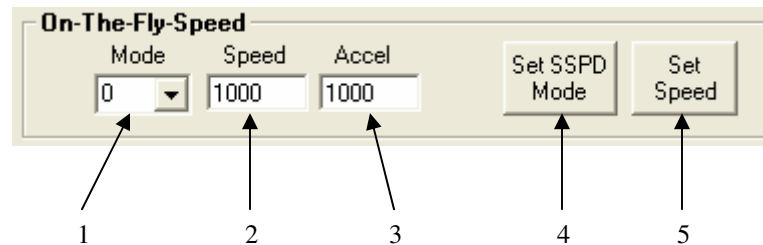
1. Program Status – program status shows here. Following are possible program status: Idle, Running, Errored and Paused.
2. Index – program that is downloaded is in the form of low-level code. Each line of the low level code has a line number which shows here.



3. Run – program is run.
4. Stop – program is stopped.
5. Pause – program that is running can be paused.
6. Continue – program that is paused can be continued

## M. On-The-Fly Speed Change

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



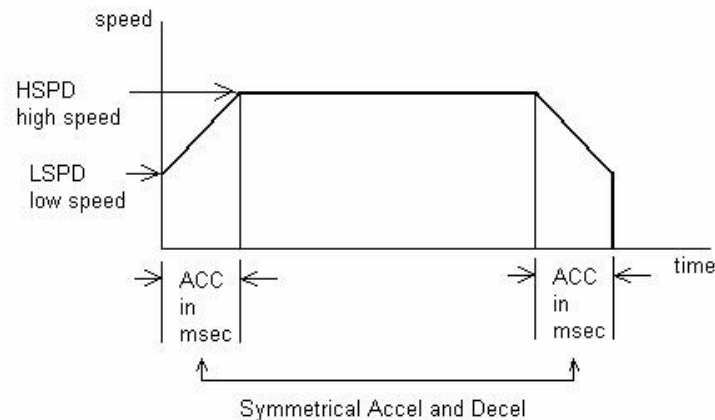
1. **On-the-fly speed mode** – Before setting the controller into motion, set the SSPDM parameter. To see which value to use, see the on-the-fly speed change section.
2. **Desired Speed** – Once the “Set Speed” button is clicked, the speed will change on-the-fly to the desired speed.
3. **Desired Acc/Dec**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

## 6. Motion Control Overview

**All the commands described in this section are all interactive commands and do not transfer over directly to standalone commands. Please refer to the “Standalone Language Specification” section for details.**

### ***Motion Profile***

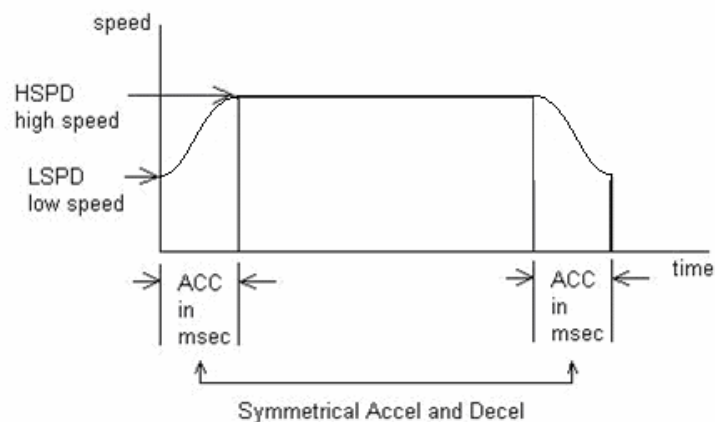
At default, ACE-SDE incorporates trapezoidal velocity profile as shown below.



High speed and low speed are in pps (pulses/second). Use **HSPD** and **LSPD** commands to set/get the high speed and low speed settings.

Acceleration and deceleration time is in milliseconds and are symmetrical (same value is used for acceleration as deceleration). Use the **ACC** command to set/get the acceleration/deceleration value.

S-curve velocity profile can also be achieved by using the **SCV** command.



### Note on acceleration:

The allowable acceleration values depend on the **LSPD** and **HSPD** settings. Please see chart below:

HSPD is less than [pps]	Minimum ACC [ms]	Speed Delta (HSPD – LSPD) [pps]
16 K	2	500
30 K	1	1 K
80 K	1	2 K
160 K	1	4 K
300 K	1	8 K
800 K	1	18 K
1.6 M	1	39 K
3.0 M	1	68 K
6 M	1	135 K

While in StepNLoop mode, the PPS (pulse/sec) speed numbers must be transposed to encoder counts by using the following formula:

$$\text{Encoder counter per second} = \text{PPS} / \text{Step-N-Loop ratio}$$

**Speed Delta:** For every increment of speed delta, the maximum value of acceleration increases by 1000 ms (1.0 seconds).

Examples:

- a) If **HSPD** = 20,000 pps, **LSPD** = 100 pps:
  - a. Get Speed delta:  $((20,000 - 100) / 1,000) = 19.9$
  - b. Max acceleration allowable:  $19.9 \times 1,000 \text{ ms} = \mathbf{19900 \text{ ms}}$  (19.9 sec)
- b) If **HSPD** = 900,000 pps, **LSPD** = 1000 pps:
  - a. Get Speed delta:  $((900,000 - 1000) / 39,000) = 23.05$
  - b. Max acceleration allowable:  $23.05 \times 1000 \text{ ms} = \mathbf{23050 \text{ ms}}$  (23.05 sec)

## Note on low speed and high speed settings:

The minimum LSPD value allowable depends on the HSPD value. These rules apply during regular motion operations as well as on-the-fly speed operations.

### SPEED WINDOWS

SSPDM value	Lowest Speed [pps]	Highest Speed [pps]
0	SSPD not used	SSPD not used
1	10	16 K
2	10	30 K
3	15	80 K
4	25	160 K
5	50	300 K
6	100	800 K
7	200	1.6 M
8	400	3.0 M
9	500	6 M

While in StepNLoop mode, the PPS (pulse/sec) speed numbers must be transposed to encoder counts by using the following formula:

$$\text{Encoder counter per second} = \text{PPS} / \text{Step-N-Loop ratio}$$

SSPDM value only applies to on-the-fly speed operations. During normal operation, SSPDM should be set to 0.

### ***On-the-fly Speed Change***

On-the-fly speed change can be achieved with the **SSPD** command. In order to use the **SSPD** command, s-curve velocity profile must be disabled.

- 1) During on-the-fly speed change operation, you must keep the initial and destination speeds within a certain window. See chart “*SPEED WINDOWS*” chart in previous section.

To select a speed window, use the **SSPDM** command.

If you are to set your destination speed outside of your current window, the SSPD feature will not work correctly.

Note that the lower the **SSPDM** value, the more accurate the pulse output speed will be. Therefore, it is recommended to choose the lowest **SSPDM** value as possible.

- 2) To set acceleration of the on-the-fly speed change, use the **ACC** command. Set the acceleration before calling the **SSPD** command.

*Note: The maximum acceleration value allowed depends on both the SSPDM value as well as the difference between the initial and destination speeds. See table below.*

SSPDM value	Speed Delta (destination speed – initial speed) [pps]
0	SSPD not used
1	500
2	1 K
3	2 K
4	4 K
5	8 K
6	18 K
7	39 K
8	68 K
9	135 K

**Speed Delta [destination speed – initial speed]:** For every increment of speed delta, the maximum value of acceleration increases by 1000 ms (1.0 seconds).

While in StepNLoop mode, the PPS (pulse/sec) speed numbers must be transposed to encoder counts by using the following formula:

$$\text{Encoder counter per second} = \text{PPS} / \text{Step-N-Loop ratio}$$

Examples:

- a) If **SSPDM** = 2, **destination speed** = 20,000 pps, **initial speed** = 100 pps:
  - a. Get Speed delta:  $((20,000 - 100) / 1,000) = 19.9$
  - b. Max acceleration allowable:  $19.9 \times 1,000 \text{ ms} = \mathbf{19900 \text{ ms}}$  (19.9 sec)
- b) If **SSPDM** = 7, **destination speed** = 900,000 pps, **initial speed** = 1000 pps:
  - a. Get Speed delta:  $((900,000 - 1000) / 39,000) = 23.05$
  - b. Max acceleration allowable:  $23.05 \times 1000 \text{ ms} = \mathbf{23050 \text{ ms}}$  (23.05 sec)
- 3) To set speed on-the-fly, use **SSPD** command. Be sure to stay within the SSPDM speed window selected.

**Important note: In order for the SSPD command to work, the controller must already be in motion by first calling either a jog or absolute move command.**

- 4) To begin normal operation again (i.e. moves not using on-the-fly speed change), send the following command to the Ace controller **“SSPDM=0”**.

## Analog Inputs

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

## Joystick Control

Using analog input 1, speed control using analog input can be done. 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

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

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

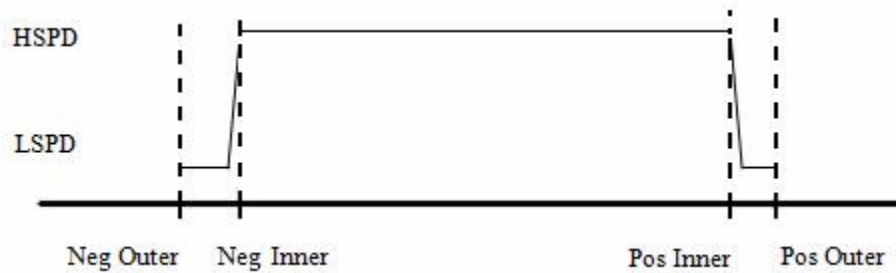
Joystick control also has soft limit control. Limits are broken down into positive inner and outer limits and negative inner and outer limits. When moving in positive direction, as soon as positive inner limit is crossed, the speed is reduced. If position crosses over the outer limit, the joystick speed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

Summary of joystick soft limit parameters

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

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 the following:



## Digital Inputs / Outputs

ACE-SDE 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 for general digital output. Enable/disable DIO control mode by using the **EDIO** command.

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

Bit	Description
0	Digital Input 1 (Start)
1	Digital Input 2 (Abort/Clear)
2	Digital Input 3 (Select 1)
3	Digital Input 4 (Select 2)
4	Digital Input 5 (Select 3)
5	Digital Input 6 (Select 4)

If 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.

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. See description below:

Bit	Description
0	Digital Output 1 (In Position)
1	Digital Output 2 (Alarm)

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.

## Motor Power

Using the **EO** command, the motor power can be enabled or disabled. By default, the enable output is turn on at boot-up.

## ***Polarity***

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

Bit	Description	
1	Direction	
4	Limit	
5	Home	
6	Latch	
7	Z index channel	
8,9	Encoder Multiplication	
	00	1X
	01	2X
	10	4X

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

## ***Positional Moves***

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

## ***Jogging***

Jogging is available for continuous speed operation. Use **J+** and **J-** commands to jog in positive or negative direction.

## ***Stopping Motor***

When motor is moving, jogging, or homing, using the **ABORT** command will immediately stop the motor. Using the **STOP** command will decelerate the motor to low speed and then stop.

## ***Homing***

Three types of automatic homing are available.

### 1) Homing using only the HOME input switch:

When homing command is sent, the motor ramps up from low speed to high speed and as soon as the home input is triggered, the position counter is reset to zero and the motor is decelerated to low speed and returns to zero position.

To trigger the home input switch, supply the opto-supply voltage with 12 to 24VDC and pull (or connect) the home input signal to opto-supply ground. Use **H+** or **H-** commands for this type of homing.

### 2) Homing using only the Z index encoder channel:



Z index channel occurs one pulse per revolution of the motor. Homing can be done using only the Z index channel. When homing with only the Z index channel, only the low speed is used. Use the **Z+** or **Z-** commands for this type of homing.

### 3) Homing using the Z index encoder channel and HOME input switch:

Homing can be done using both the HOME switch input and Z index encoder channel to get accurate home position. When home command is issued, pulse ramps up from low to high speed. As soon as the home input is triggered, the pulse rate ramps down to low speed. Low speed is maintained until the Z index channel of the encoder is triggered. Use **ZH+** or **ZH-** commands for this type of homing.

### 4) Limit Homing:

Limit switch can be used for homing by jogging the motor to the limit switch (using **J+** or **J-** commands), clearing the limit error (using **CLR** command) and then resetting the position counter (using **PX** command).

## **Motor Position**

Motor position can be set and read by using the **PX** command.  
Encoder position can be set and read by using the **EX** command.

## **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 <b>CLR</b> 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 <b>CLR</b> command before issuing any subsequent move commands.
8	Latch input status

9	Z-index status
---	----------------

Example:

- 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.

### **Limit Inputs**

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, use **CLR** command to clear the error. Once the error is cleared, move the motor out of the limit switch.

The limit switch is an opto-isolated input. Supply the opto-supply voltage 12 to 24VDC. To trigger the limit input switch, connect the input signal to ground of opto-supply.

### **Latch Input**

The ACE-SDE module provides the following 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.

Note: When StepNLoop mode is enabled, the position value should be ignored.

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

Once the latch is triggered, the triggered position can be retrieved using **LTP** (latched pulse position) and **LTE** (latched encoder position) commands.

## StepNLoop Closed Loop Control

ACE-SDE module has closed loop position control algorithm called StepNLoop control for accurate positioning of the motor if an incremental encoder is used.

StepNLoop control does following operations:

- 1) Position Delta monitoring: Delta position is the difference between the actual and the target position. When the Delta goes over the allowed Error Range, the motor is stopped and the StepNLoop Status goes into the “stall” error state. Delta monitoring is done for all moves including homing and jogging. View the Delta value by using the **DX** command.
- 2) Position Correction at the end of the move: Correction of the motor position is done at the end of any targeted move.

Following are configuration required for StepNLoop control:

SNL Parameter	Description
Pulse/Encoder Ratio	Number of pulse counts per revolution of motor / number of encoder counts. Use <b>SLR</b> command to set the ratio. Value must be in the range [0.001 , 999.999].
Tolerance Range	When the actual encoder position is within the desired encoder position by this tolerance range, no position correction is done. Use <b>SLT</b> command to set the tolerance range.
Correction Range	When the actual encoder position is within desired encoder position by this correction range, position correction is done when idle. If the actual encoder position is outside of correction range, the motor status goes to error state. Use <b>SLE</b> command to set the correction range.
Correction Attempt Number	This is the maximum number of correction tries that the controller will attempt. If the correction cannot be done within this number of tries, the motor status goes to error state. Use <b>SLA</b> command to set the maximum correction attempt number.

To enable and disable the StepNLoop feature use the **SL** command. To read the StepNLoop status, use **SLS** command to read the status.

Following are the StepNLoop status 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 <b>CLRS</b> or <b>CLR</b> command.
9	Correction attempt error. To clear this error, use <b>CLRS</b> or <b>CLR</b> command.
10	Stall Error. <b>DX</b> value has exceeded the Correction range value. To clear this error, use <b>CLRS</b> or <b>CLR</b> command.
11	Limit Error
12	N/A (i.e. StepNLoop is not enabled)

### StepNLoop Notes:

Once StepNLoop is enabled, position move commands are in term of encoder position.

For example, X1000 means to move the motor to encoder 1000 position.

Once StepNLoop is enabled, the speed is in encoder speed.

For example HSPD=1000 when StepNLoop is enabled means that the target high speed is 1000 encoder counts per second.

StepNLoop correction is done only when the pulse rate is idle. For example, when the motor is moving, correction is not done. Once the pulse rate is idle, StepNLoop correction is done.

### Device Number

#### Changing Device Number:

ACE-SDE module provides the user with the ability to set the device number of a specific device. In order to make these changes, first store the desired number using the **DN** command. Please note that this value must be within the range [SDE01,SDE99].

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: **SDE01**

## **Baud Rate Setting**

Once RS-485 communication is established, baud rate can be changed using the **DB** command.

### Changing Baud Rate:

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

Please note that the device baud rate must be within the range [1, 5].

Device Baud Value	Baud Rate (bps)
1	9600
2	19200
3	38400
4	57600
5	115200

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: Baud rate is set to: **1 (9600 bps)**

## **Sync Output**

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

Note: While feature is enabled, the designated digital output (DO2) can not 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 position.  
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 position.

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).

### ***Broadcasting over RS-485***

The address '00' is reserved for broadcasting over an RS-485 bus. Any ASCII command prefixed by '@00' will be processed by all ACE-SDE modules on the RS-485 bus. When a broadcast command is received by an ACE-SDE module, no response is sent back to the master.

### ***Response type selection***

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]

Examples:

For querying the encoder position

Send: @01EX[CR]

Reply: 1000[CR]

For jogging the motor in positive direction

Send: @01J+[CR]

Reply: OK[CR]

To achieve this response string type, send command **RT=0**.

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

Examples:

For querying the encoder position

Send: @01EX[CR]

Reply: #011000[CR]

For jogging the motor in positive direction

Send: @01J+[CR]

Reply: #01OK[CR]

To achieve this response string type, send command **RT=1**.

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

### **Micro-step Driver Configuration**

The built in driver of ACE-SDE can be configured via software. See below for commands relating to driver configuration.

<b>Command</b>	<b>Description</b>
<b>DRVMS</b>	Set/get micro-stepping value of the driver [2-500].
<b>DRVRC</b>	Set/get run current value of the driver [100-3000 mA] (peak current)
<b>DRVIC</b>	Set/get idle current value of the driver [100-2800 mA] (peak current)
<b>DRVIT</b>	Set/get idle time value of the driver [1-100 centi-sec]. This is the amount of time the driver waits before dropping from the run current to idle current value
<b>RR</b>	Get driver parameters. DRVMS/DRVRC/DRVIC/DRVIT values will not be valid until the controller reads the driver parameters by sending the RR command. Once this command is sent, communication to ACE-SDE will not be available for 2 seconds.
<b>R2</b>	Get the read operation status. After sending the RR command and waiting 2 seconds, get the read operation status by using the R2 command. A return value of 1 signifies a successful read. All other return values signify a failed read operation.
<b>RW</b>	Write driver parameters. After DRVMS/DRVRC/DRVIC/DRVIT are set by the user, they are not actually written to the driver until the RW command is sent. Once this command is sent, communication to ACE-SDE will not be available for 2 seconds.
<b>R4</b>	Get the write operation status. After sending the RW command and waiting 2 seconds, get the write operation status by using the R4 command. A return value of 1 signifies a successful write. All other return values signify a failed write operation.

Driver configuration can also be done via standalone code.

**Important note:** While reading or writing to the micro-step driver, both StepNLoop, joystick control and DIO control modes must be disabled. Otherwise, reading/writing operations will fail.

## ***Standalone Programming***

### Standalone Program Specification:

Memory size: 1785 assembly lines ~ 10.5 KB.

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

### Stand-alone execution while in Step-N-Loop:

While a stand-alone program is running in closed-loop operation, before executing an absolute move command, the controller first verifies that it is NOT correcting or moving to a previous absolute position.

### Error Handling:

If an error occurs during standalone execution (i.e. limit error, stall error, max attempt error, etc.), the program automatically jumps to SUB 31. If SUB 31 is NOT defined, the program will cease execution and go to error state. If SUB 31 is defined by the user, the code within SUB 31 is first executed, and then standalone execution continues.

Calling subroutines over communication: Once a subroutine is written into the flash, they can be called via USB/RS-485 communication using the **GS** command. The subroutines are referenced by their subroutine number [0-31]. If a subroutine number is not defined, the ACE-SDE will return with an error.

**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.**

## ***Storing to Flash***

The following items are stored to flash:

- Device Name (used for USB and RS-485 ASCII communication)
- Baud Rate
- Polarity settings
- StepNLoop parameters
- DIO parameters
- Joystick parameters
- Standalone run-on-boot-up parameter
- Variables V51-V100 (Note that on boot-up, V1-V50 are reset to value 0)

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



## 7. Communication – USB

ACE-SDE USB communication is USB 2.0 compliant.

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

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

**Important Note:** PerformaxCom.dll only supports single-threaded programming. Calling PerformaxCom.dll functions from different threads will lead to unexpected behavior even if the functions are not being used by different threads simultaneously.

### **USB Communication API Functions**

For USB communication, following DLL API functions are provided.

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

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

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

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

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

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

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

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

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

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

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

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

### **USB Communication Issues**

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

- 1) **Multi-threading:** Be sure that your application does not employ multi-thread processing. See “important note” in the beginning of this section.
- 2) **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)

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

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



## 8. Communication – RS-485 (ASCII)

When communicating on RS-485 (ASCII), it is recommended to add 120 Ohm terminating resistor between 485+ and 485- signal on the last module.

### **Communication Port Settings**

Byte Size:	8 bits
Parity:	None
Flow Control:	None
Stop Bit:	1

### **ASCII Protocol**

#### Sending Command

ASCII command string in the format of  
@[DeviceName][ASCII Command][CR]

*[CR] character has ASCII code 13.*

#### Receiving Reply

The response will be in the format of  
[Response][CR]

*[CR] character has ASCII code 13.*

#### Examples:

For querying the x-axis polarity

Send: @00POL[CR]  
Reply (if RT=0): 7[CR]  
Reply (if RT=1): #007[CR]

For jogging the x-motor in positive direction

Send: @00J+[CR]  
Reply (if RT=0): OK[CR]  
Reply (if RT=1): #00OK[CR]

For aborting any motion in progress

Send: @00ABORT[CR]  
Reply (if RT=0): OK[CR]  
Reply (if RT=1): #00OK[CR]

Note: RT is a parameter that sets the response type of the device.

## 9. Communication - DIO

DIO communication allows the user to store 16 different types (See DIO Move List) of moves into ACE-SDE 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)**.

### ***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.

### ***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 DIO Move List)
1	Target Position
2	Low Speed
3	Acceleration
4	High Speed

## 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]

## Examples:

1. **Make DIO selection “0” correspond 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” correspond 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

## 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.

## Important DIO Notes

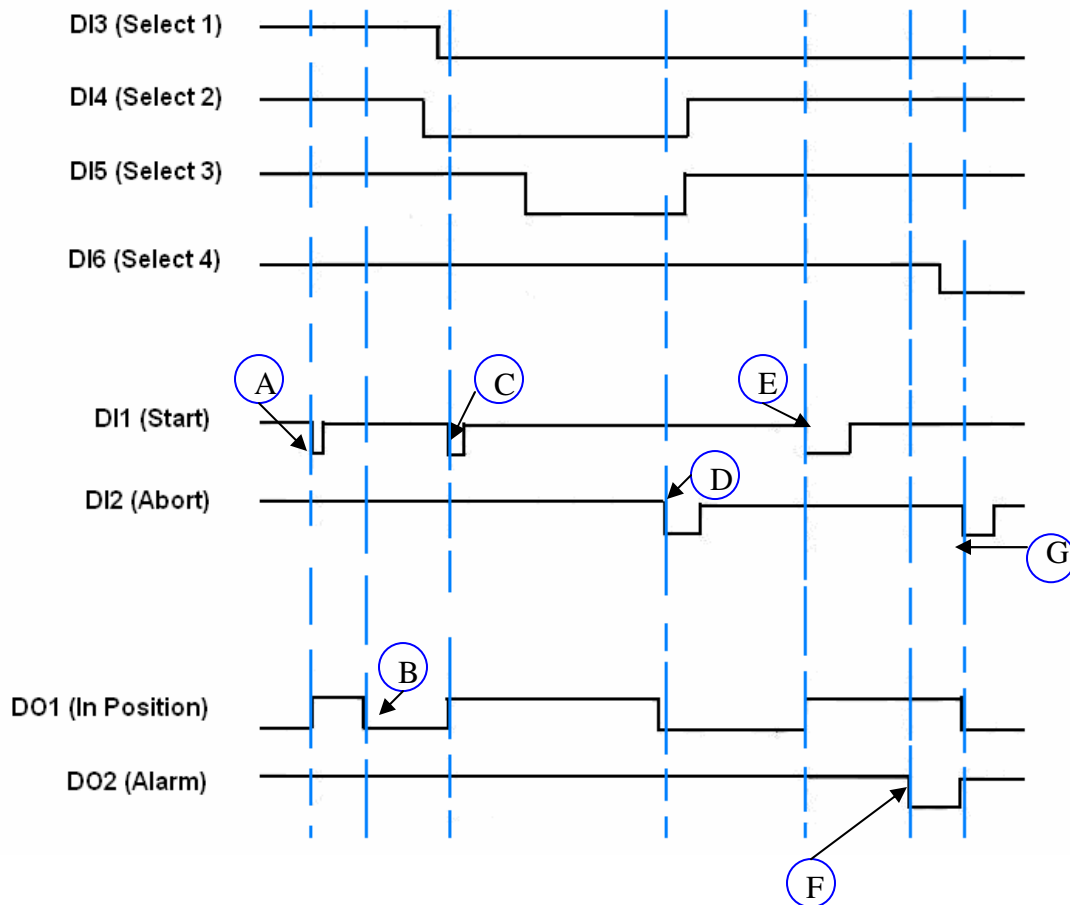
Triggering the **start bit (DI1)** will not trigger a motion move if the **abort bit (DI2)** is on, or if the controller is in error state. If the controller is in error state, first clear the error by triggering the **abort/clear bit (DI2)**.

The alarm bit output is on when ever there is either a StepNLoop or limit error.

The in position bit output is on when ever the motor is in position.

Below is an example of a timing diagram using DIO control.

**Note that signals are active low.**



- 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 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.

**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.**



## 10. ASCII Language Specification

**All the commands described in this section are all interactive commands and do not transfer over directly to standalone commands. Please refer to the “Standalone Language Specification” section for details.**

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

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.	Milli-seconds
ACC=[Value]	Sets acceleration value in milliseconds. Example: ACC=300	OK
AI1, AI2	Get analog input status (0-5000 mV)	Milli-volts
CLR	Clears limit error as well as StepNLoop error	OK
CLRS	Clears StepNLoop error. Note CLR also clears a StepNLoop error	OK
DB	Return the current baud rate of the device	1 – 9600 bps 2 – 19200 bps 3 – 38400 bps 4 – 57600 bps 5 – 115200 bps
DB=[Value]	Set the baud rate of the device	OK
DI	Return status of digital inputs	6-bit number
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	0,1
DO[1-2]=[Value]	Set individual bit status of digital outputs	OK
DN	Get device name	[SDE01-SDE99]
DN=[Value]	Set device name	OK
DX	Returns the delta value during StepNLoop control	32-bit number
DRVIC	Get driver idle current setting. Value is only valid after reading parameters using the “RR” command.	[100 – 3000] mA (peak current)
DRVIC=[Value]	Set driver idle current setting. Value is only written to the driver after using the “RW” command.	OK
DRVIT	Get driver idle time setting. Value is only valid after reading parameters using the “RR” command.	[1-100] centi-sec
DRVIT=[Value]	Set driver idle time setting. Value is only written to the driver	OK

	after using the “RW” command.	
DRVMS	Get driver micro-step setting. Value is only valid after reading parameters using the “RR” command.	[2-500] micro-stepping
DRVMS=[Value]	Set driver micro-step setting. Value is only written to the driver after using the “RW” command.	OK
DRVRC	Get driver run current setting. Value is only valid after reading parameters using the “RR” command.	[1-100] centi-sec
DRVRC=[Value]	Set driver run current setting. Value is only written to the driver after using the “RW” command.	OK
EO	Returns driver power enable.	1 – Motor power enabled 0 – Motor power disabled
EO=[0 or 1]	Enables (1) or disable (0) motor power.	OK
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
EX	Returns current encoder counter value	32-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
ID	Returns product ID	Ace-Series-SDE
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	32-bit number
JV1=[Value]	Set max speed for joystick control	OK
JV3	Get max speed delta for joystick control	32-bit number
JV3=[Value]	Set max speed delta for joystick control	OK
JV5	Get zero speed tolerance for joystick control	32-bit number
JV5=[Value]	Set zero speed tolerance for joystick control	OK
JL1	Get negative outer limit for joystick control	32-bit number
JL1=[Value]	Set negative outer limit for joystick control	OK
JL2	Get negative inner limit for joystick control	32-bit number
JL2=[Value]	Set negative inner limit for joystick control	OK
JL3	Get positive inner limit for joystick control	32-bit number
JL3=[Value]	Set positive inner limit for joystick control	OK
JL4	Get positive outer limit for joystick control	32-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
LSPD	Returns Low Speed Setting	PPS
LSPD=[Value]	Sets Low Speed	OK
LT=[0 or 1]	Enable or disable position latch feature	OK
LTE	Returns latched encoder position	32-bit number
LTP	Returns latched pulse position	32-bit number
LTS	Returns latch status.	0 – Latch off 1 – Latch on and waiting for latch trigger

		2 – Latch triggered
MM	Get move mode status	0 – Absolute move mode 1 – Incremental move mode
MST	Returns motor status	Bit 0 – constant speed Bit 1 – accelerating Bit 2 – decelerating Bit 3 – home input Bit 4 – minus limit input Bit 5 – plus limit input Bit 6 – minus limit error Bit 7 – plus limit error Bit 8 – latch input Bit 9 – Z-index
MPXX	Get DIO parameter	
MPXX=[Value]	Set DIO parameter	OK
POL	Returns current polarity	Bit 1 – Dir Bit 4 – Limit Bit 5 – Home Bit 6 – Latch Bit 7 – Z index channel Bit 8 – 2X encoder mult Bit 9 – 4X encoder mult
POL=[value]	Sets polarity.	OK
PS	Returns current pulse speed	PPS
PX	Returns current position value	32-bit number
PX=[value]	Sets the current position value	OK
R2	Get driver read operation status	[1] – Driver read successful [2-7] – Driver read failure
R4	Get driver write operation status	[1] – Driver write successful [2-7] – Driver write failure
RR	Read driver parameters	OK
RT	Get response type value	0 or 1
RT= [0 or 1]	Set response type value	OK
RW	Write driver parameters	OK
SASTAT	Get standalone program status 0 – Stopped 1 – Running 2 – Paused 4 – In Error	0-4
SA[LineNumber]	Get standalone line LineNumber: [0,1784]	
SA[LineNumber]=[Value]	Set standalone line LineNumber: [0,1784]	
SCV	Returns the s-curve control	0 or 1
SCV=[0 or 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	32-bit number
SLA=[value]	Sets maximum number of StepNLoop control attempt	OK
SLE	Returns StepNLoop correction range value.	32-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	0 – Idle 1 – Moving 2 – Correcting 3 – Stopping 4 – Aborting 5 – Jogging 6 – Homing 7 – Z Homing 8 – Correction Range Error. 9 – Correction Attempt Error. 10 – Limit Error 11 – Stall Error 12 – Not Applicable (i.e. StepNLoop not enabled)
SLT	Returns StepNLoop tolerance value	32-bit
SLT=[value]	Sets StepNLoop tolerance value.	OK
SLOAD	Returns RunOnBoot parameter	0,1
SLOAD=[0 or 1]	0 – Do NOT run standalone program on boot up 1 – Run standalone program on boot up	OK
SR=[Value]	Control standalone program: 0 – Stop standalone program 1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program	OK
SPC	Get program counter for standalone program	[0-1784]
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
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	1,2,3
SYNC=	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 signed number
SYNP=	Set trigger position	28 bit signed number
SYNT	Get pulse width time (ms). Only applicable if sync output configuration is set to 1.	Milli-seconds
SYNT=	Set pulse width time (ms). Only applicable if sync output configuration is set to 1. Max 30ms	OK

V[1-100]	Read variables 1-100	32-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. Maximum allowed incremental move amount is 262143. For example, if current position is 100000, target move must be between 362143 and -162143	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

## 11. Standalone Language Specification

;

Description:

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

Syntax:

; [Comment Text]

Examples:

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

### **ABORTX**

Description:

**Motion:** Immediately stop motion without deceleration.

Syntax:

ABORTX

Examples:

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

## ABS

Description:

**Command:** Changes all move commands to absolute mode.

Syntax:

ABS

Examples:

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

## ACC

Description:

**Read:** Get acceleration value

**Write:** Set acceleration value.

Value is in milliseconds.

Syntax:

**Read:** [variable] = ACC

**Write:** ACC = [value]

ACC = [variable]

Examples:

```
ACC=300      ;***Sets the acceleration to 300 milliseconds
V3=500       ;***Sets the variable 3 to 500
ACC=V3       ;***Sets the acceleration to variable 3 value of 500
```

## ***AI[1-2]***

Description:

**Read:** Gets the analog input value. ACE-SDE has 2 analog inputs.

Range is from 0-5000 mV

Syntax:

**Read:** [variable] = AI[1-2]

**Conditional:** IF AI[1-2]=[variable]  
ENDIF

IF AI[1-2]=[value]  
ENDIF

Examples:

```
IF AI1 < 500
    DO=1          ;***If analog input 1 is less than 500, set DO=1
ENDIF
```

## ***DELAY***

Description:

Set a delay (1 ms units)

Syntax:

Delay=[Number] (1 ms units)

Examples:

```
JOGX+          ;***Jogs axis to positive direction
DELAY=10000    ;***Wait 10 second
ABORTX         ;***Stop axis
```



## ***DI***

Description:

**Read:** Gets the digital input value. ACE-SDE has 6 digital inputs.

If 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.

Syntax:

**Read:** [variable] = DI

**Conditional:** IF DI=[variable]  
ENDIF

IF DI=[value]  
ENDIF

Examples:

```
IF DI=0
    DO=1          ;***If all digital inputs are triggered, set DO=1
ENDIF
```

## ***DI[1-6]***

Description:

**Read:** Gets the digital input value. ACE-SDE has 6 digital inputs.

If 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.

Syntax:

**Read:** [variable] = DI[1-6]

**Conditional:** IF DI[1-6]=[variable]  
ENDIF

IF DI[1-6]=[0 or 1]  
ENDIF

Examples:

```
IF DI1=0
    DO=1          ;***If digital input 1 is triggered, set DO=1
ENDIF
```

## **DO**

### Description:

**Read:** Gets the digital output value

**Write:** Sets the digital output value

ACE-SDE has 2 digital 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.

### Syntax:

**Read:** [variable] = DO

**Write:** DO = [value]

DO = [variable]

**Conditional:** IF DO=[variable]  
ENDIF

IF DO=[value]  
ENDIF

### Examples:

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

## ***DO[1-2]***

### Description:

**Read:** Gets the individual digital output value

**Write:** Sets the individual digital output value

ACE-SDE has 2 digital 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.

### Syntax:

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

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

DO[1-2] = [variable]

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

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

### Examples:

```
DO1=1      ;***Turn DO1 on
DO2=1      ;***Turn DO2 on
```

## ***DRVIC***

Description:

**Write:** Sets the driver idle current parameter

Syntax:

**Write:** DRVIC=[value]

Examples:

```

WHILE 1=1
    IF DI1 = 0
        SL=0
        DRVMS=100
        DRVIT=1
        DRVIC=100
        DRVRC=1000
        RW
        DELAY=2000
        V1=RWSTAT
        IF V1=1
            DO1=1
        ELSE
            DO2=1
        ENDIF
    ENDIF
ENDWHILE
;***If DI1 is triggered, execute
;***Disable StepNLoop
;***Micro-step set to 100
;***Idle-time set to 1 cent-sec
;***Idle-current set to 100 mA
;***Run-current set to 1000 mA
;***Write driver parameters
;***Wait 2 seconds for write operation
;***Check write operation status
;***If write operation was success, DO1=1
;***Write operation failed, DO2=1

```

## ***DRVIT***

Description:

**Write:** Sets the driver idle time parameter

Syntax:

**Write:** DRVIT=[value]

Examples:

See DRVIC

## ***DRVMS***

Description:

**Write:** Sets the driver micro-step parameter

Syntax:

**Write:** DRVMS=[value]

Examples:

See DRVIC

## ***DRVRC***

Description:

**Write:** Sets the driver run current parameter

Syntax:

**Write:** DRVRC=[value]

Examples:

See DRVIC

## ***ECLEARX***

Description:

**Write:** Clears motor error status. Also clears a StepNLoop error.

Syntax:

**Write:** ECLEARX

Examples:

ECLEARX ;\*\*\*Clears motor error



## Motor Status

[Comparison] can be any of the following

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

Examples:

```
IF V1=1
    X1000
ELSEIF V1=2
    X2000
ELSEIF V1=3
    X3000
ELSE
    X0
ENDIF
```

## ***END***

### Description:

Indicate end of program.  
Program status changes to idle when END is reached.

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

### Syntax:

END

### Examples:

```
X0
WAITX
X1000
END
```

## ***ENDIF***

### Description:

Indicates end of IF operation

### Syntax:

ENDIF

### Examples:

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



## ***ENDSUB***

### Description:

Indicates end of subroutine

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

### Syntax:

ENDSUB

### Examples:

```
GOSUB 1
END
```

```
SUB 1
    X0
    WAITX
    X1000
    WAITX
ENDSUB
```

## ***ENDWHILE***

### Description:

Indicate end of WHILE loop

### Syntax:

ENDWHILE

### Examples:

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

## **EO**

Description:

**Read:** Gets the enable output value

**Write:** Sets the enable output value

Syntax:

**Read:** [variable] = EO

**Write:** EO = [value]

EO = [variable]

**Conditional:** IF EO=[variable]  
ENDIF

IF EO=[value]  
ENDIF

Examples:

EO=1 ;\*\*\*Energize motor

## **EX**

Description:

**Read:** Gets the current encoder position

**Write:** Sets the current encoder position

Syntax:

**Read:** [variable] = E[axis]

**Write:** EX = [0 or 1]

EX = [variable]

**Conditional:** IF EX=[variable]  
ENDIF

IF EX=[value]  
ENDIF

Examples:

EX=0 ;\*\*\*Sets the current encoder position to 0

## ***GOSUB***

### Description:

Perform go to subroutine operation  
Subroutine range is from 0 to 31.

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

### Syntax:

GOSUB [subroutine number]

[Subroutine Number] range is 0 to 31

### Examples:

```
GOSUB 0
END

SUB 0
    X0
    WAITX
    X1000
    WAITX
ENDSUB
```

## ***HOMEX[+ or -]***

### Description:

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

### Syntax:

HOMEX[+ or -]

### Examples:

HOMEX+ ;\*\*\*Homes axis in positive direction

## ***HSPD***

### Description:

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

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

Range is from 1 to 6,000,000.

### Syntax:

**Read:** [variable] = HSPD

**Write:** HSPD = [value]

HSPD = [variable]

### Examples:

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

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

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

## ***IF***

### Description:

Perform IF condition check

### Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

### Examples:

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

## ***INC***

Description:

**Command:** Changes all move commands to incremental mode.

Syntax:

INC

Examples:

INC	;***Change to incremental mode
PX=0	;***Change position to 0
X1000	;***Move axis to position 1000 (0+1000)
X2000	;***Move axis to position 3000 (1000+2000)

## ***JOGX[+ or -]***

Description:

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

Syntax:

JOGX[+ or -]

Examples:

JOGX+	;***Jogs axis in positive direction
JOGX-	;***Jogs axis in negative direction

### **JOYDIS**

Description:

**Write:** Disable joystick feature

Syntax:

**Write:** JOYDIS

Examples:

JOYDIS ;\*\*\*Enable joystick feature

### **JOYENA**

Description:

**Write:** Enable joystick feature

Syntax:

**Write:** JOYENA

Examples:

JOYENA ;\*\*\*Enable joystick feature

### **JOYHS**

Description:

**Write:** Set high speed setting for joystick control

Syntax:

**Write:** JOYHS= [value]  
JOYHS= [variable]

Examples:

JOYHS=10000 ;\*\*\*High speed of axis is set to 10,000 pps

### **JOYDEL**

Description:

**Write:** Set maximum delta value of change in speed for joystick control

Syntax:

**Write:** JOYDEL= [value]  
JOYDEL = [variable]

Examples:

JOYDEL=100 ;\*\*\*Speed delta of axis is set to 100 pps

## **JOYNO**

Description:

**Write:** Set negative outer limit for joystick control

Syntax:

**Write:** JOYNO= [value]  
JOYNO= [variable]

Examples:

JOYNO=-10000	*** negative outer limit of axis set to -10000
JOYNI=-9000	*** negative inner limit of axis set to -9000
JOYPI=9000	*** positive inner limit of axis set to 9000
JOYPO=10000	*** positive outer limit of axis set to 10000

## **JOYNI**

Description:

**Write:** Set negative inner limit for joystick control

Syntax:

**Write:** JOYNI = [value]  
JOYNI= [variable]

Examples:

JOYNO=-10000	*** negative outer limit of axis set to -10000
JOYNI=-9000	*** negative inner limit of axis set to -9000
JOYPI=9000	*** positive inner limit of axis set to 9000
JOYPO=10000	*** positive outer limit of axis set to 10000

## **JOYPI**

Description:

**Write:** Set positive inner limit for joystick control

Syntax:

**Write:** JOYPI= [value]  
JOYPI= [variable]

Examples:

JOYNO=-10000	*** negative outer limit of axis set to -10000
JOYNI=-9000	*** negative inner limit of axis set to -9000
JOYPI=9000	*** positive inner limit of axis set to 9000
JOYPO=10000	*** positive outer limit of axis set to 10000



## **JOYPO**

Description:

**Write:** Set positive outer limit for joystick control

Syntax:

**Write:** JOYPO= [value]  
JOYPO= [variable]

Examples:

JOYNO=-10000 ;\*\*\* negative outer limit of axis set to -10000  
JOYNI=-9000 ;\*\*\* negative inner limit of axis set to -9000  
JOYPI=9000 ;\*\*\* positive inner limit of axis set to 9000  
JOYPO=10000 ;\*\*\* positive outer limit of axis set to 10000

## **JOYTOL**

Description:

**Write:** Set zero tolerance value for joystick control

Syntax:

**Write:** JOYTOL = [value]  
JOYTOL= [variable]

Examples:

JOYTOL=10 ;\*\*\* zero tolerance value of axis set to 10

## **LSPD**

Description:

**Read:** Get low speed. Value is in pulses/second.

**Write:** Set low speed. Value is in pulses/second.

Range is from 1 to 6,000,000.

Syntax:

**Read:** [variable]=LSPD  
**Write:** LSPD=[long value]  
LSPD=[variable]

Examples:

LSPD=1000 ;\*\*\*Sets the start low speed to 1,000 pulses/sec  
  
V1=500 ;\*\*\*Sets the variable 1 to 500  
LSPD=V1 ;\*\*\*Sets the start low speed to variable 1 value of 500

## ***LT***

Description:

**Write:** Set latch enable

Range is [0,1]

Syntax:

**Write:** LT=[0,1]  
LT=[variable]

Examples:

```

LT=1                ;***Enable latch
WHILE 1=1
    V2=LTS           ;***Get latch status
    IF LTS = 2
        V3=LTE       ;***Get latch encoder value if latch is triggered
        V4=LTP       ;***Get latch position value if latch is triggered
    ENDIF
ENDWHILE

```

## ***LTE***

Description:

**Read:** Get latch encoder value

Syntax:

**Read:** [variable]=LTE

Examples:

See LT

## ***LTP***

Description:

**Read:** Get latch position value

Syntax:

**Read:** [variable]=LTP

Examples:

See LT

## ***LTS***

Description:

**Read:** Get latch status

Syntax:

**Read:** [variable]=LTS

Examples:

See LT

## ***MSTX***

Description:

**Read:** Get motor status

Syntax:

**Read:** [variable]=MSTX

**Conditional:** IF MSTX=[variable]  
ENDIF

IF MSTX=[value]  
ENDIF

Examples:

```
IF MSTX=0
    DO=3
ELSE
    DO=0
ENDIF
```

## PX

Description:

**Read:** Gets the current pulse position

**Write:** Sets the current pulse position

Syntax:

**Read:** Variable = PX

**Write:** PX = [value]

PX = [variable]

**Conditional:** IF PX=[variable]  
ENDIF

IF PX=[value]  
ENDIF

Examples:

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

## PS

Description:

**Read:** Get the current pulse speed

Syntax:

**Read:** Variable = PS

**Conditional:** IF PS=[variable]  
ENDIF

IF PS=[value]  
ENDIF

Examples:

```
JOGX+          ;***Jogs axis to positive direction
DELAY=1000     ;***Wait 1 second
ABORTX         ;***Stop without deceleration
V1=PS          ;***Sets variable 1 to pulse speed
```

## ***RW***

Description:

**Write:** Start driver write operation. Note that after executing RW, wait 2 seconds before any other operation is executing (using DELAY=2000).

Syntax:

**Write:** RW

Examples:

See DRVIC

## ***RWSTAT***

Description:

**Read:** Get driver write operation status

Syntax:

**Read:** [variable]=RWSTAT

Examples:

See DRVIC

## ***SCV***

Description:

**Write:** Set s-curve enable.

Range is from 0 or 1

Syntax:

**Write:** SCV=[0 or 1]  
SCV=[variable]

*Note: If s-curve is enabled for an axis, on-the-fly speed feature can not be used for the corresponding axis.*

Examples:

SCV=1           ;\*\*\*Sets axis to use s-curve acceleration: on-the-fly speed  
                  ; change is NOT allowed for this axis.

## **SL**

Description:

**Write:** Set StepNLoop closed-loop mode

Range is from 0 or 1

Syntax:

**Write:** SL=[0 or 1]

Examples:

SL=1 ;\*\*\*Sets axis to closed-loop mode

## **SLSX**

Description:

**Read:** Get StepNLoop status

Syntax:

**Read:** [variable]=SLSX  
**Conditional:** IF SLSX =[variable]  
 ENDIF  
  
 IF SLSX =[value]  
 ENDIF

Examples:

IF SLSX != 0  
 ECLEARX  
 ELSE  
 ECLEARX  
 ENDIF

## SSPD

Description:

**Write:** Set on-the-fly speed change for an individual axis.  
Range is from 1 to 6,000,000 PPS

Syntax:

**Write:** SSPD=[value]  
SSPD=[variable]

*Note: If s-curve is enabled for an axis, on-the-fly speed feature can not be used for the corresponding axis.*

Examples:

```
SCV=0           ;***Disable s-curve acceleration
HSPD=1000       ;***X-axis high speed
LSPD=100        ;***Set low speed
ACC=100         ;***Set acceleration
JOGX+          ;***Jogs to positive direction
DELAY=1000      ;***Wait 1 second
SSPD=3000       ;***Change speed on-the-fly to 3000 PPS
```

## SSPDM

Description:

**Write:** Set individual on-the-fly speed change mode  
Range is from 0 to 9

Syntax:

**Write:** SSPDM=[0-9]  
SSPDM=[variable]

Examples:

```
SCV=0           ;***Disable s-curve acceleration
HSPD=1000       ;***X-axis high speed
LSPD=100        ;***Set low speed
ACC=100         ;***Set acceleration
JOGX+          ;***Jogs to positive direction
DELAY=1000      ;***Wait 1 second
SSPDM=1         ;***Set on-the-fly speed change mode to 1
ACC=20000       ;***Set acceleration to 20 seconds
SSPD=190000     ;***Change speed on-the-fly to 190000 PPS
```

## **STOPX**

Description:

**Command:** Stop all axes if in motion with deceleration.  
Previous acceleration value is used for deceleration.

Syntax:

STOPX

Examples:

JOGX+	;	***Jogs axis to positive direction
DELAY=1000	;	***Wait 1 second
STOPX	;	***Stop with deceleration

## **STORE**

Description:

**Command:** Store all values to flash

Syntax:

STORE

Examples:

V80=EX	;	***Put encoder value in V80
DELAY=1000	;	***Wait 1 second
STORE	;	***Store V80 to non-volatile flash



## ***SYNCCFG***

Description:

**Write:** Set sync output configuration

Syntax:

**Write:** SYNC=[value]  
 SYNC=[variable]

Examples:

```

SYNC=1           ;*** Set sync output configuration to 1
SYNP=3000        ;*** Set sync output position to 3000
SYNT=10          ;*** Set sync output pulse time to 10 ms
SYNO             ;*** Turn on sync output feature

V1=1             ;*** Wait until sync output is triggered
WHILE V1 != 2
    V1=SYNS
ENDWHILE

SYNF             ;*** Disable sync output feature
  
```

## ***SYNCOFF***

Description:

**Write:** Disable sync output feature

Syntax:

**Write:** SYNF

Examples:

```

SYNC=1           ;*** Set sync output configuration to 1
SYNP=3000        ;*** Set sync output position to 3000
SYNT=10          ;*** Set sync output pulse time to 10 ms
SYNO             ;*** Turn on sync output feature

V1=1             ;*** Wait until sync output is triggered
WHILE V1 != 2
  
```

```
V1=SYNS
ENDWHILE
```

```
SYNF          ;*** Disable sync output feature
```

## ***SYNCON***

Description:

**Write:** Enable sync output feature

Syntax:

**Write:** SYNO

Examples:

```
SYNC=1          ;*** Set sync output configuration to 1
SYNP=3000        ;*** Set sync output position to 3000
SYNT=10          ;*** Set sync output pulse time to 10 ms
SYNO            ;*** Turn on sync output feature
```

```
V1=1            ;*** Wait until sync output is triggered
WHILE V1 != 2
    V1=SYNS
ENDWHILE
```

```
SYNF          ;*** Disable sync output feature
```

## ***SYNCPOS***

Description:

**Write:** Set sync output position.

Syntax:

**Write:** SYNP=[value]

**Write:** SYNP=[variable]

Examples:

```
SYNC=1          ;*** Set sync output configuration to 1
SYNP=3000        ;*** Set sync output position to 3000
SYNT=10          ;*** Set sync output pulse time to 10 ms
SYNO            ;*** Turn on sync output feature
```

```
V1=1            ;*** Wait until sync output is triggered
```

```
WHILE V1 != 2
    V1=SYNS
ENDWHILE
```

```
SYNF                ;*** Disable sync output feature
```

## **SYNCSTAT**

Description:

**Read:** Get status for sync output

Syntax:

**Read:** [variable] = SYNS

Examples:

```

SYNC=1           ;*** Set sync output configuration to 1
SYNP=3000        ;*** Set sync output position to 3000
SYNT=10          ;*** Set sync output pulse time to 10 ms
SYNO             ;*** Turn on sync output feature

V1=1             ;*** Wait until sync output is triggered
WHILE V1 != 2
    V1=SYNS
ENDWHILE

SYNF             ;*** Disable sync output feature
  
```

## **SYNCTIME**

Description:

**Write:** Set pulse output width time for sync output

Syntax:

**Write:** SYN[axis]T=[value]

Examples:

```

SYNC=1           ;*** Set sync output configuration to 1
SYNP=3000        ;*** Set sync output position to 3000
SYNT=10          ;*** Set sync output pulse time to 10 ms
SYNO             ;*** Turn on sync output feature

V1=1             ;*** Wait until sync output is triggered
WHILE V1 != 2
    V1=SYNS
ENDWHILE

SYNF             ;*** Disable sync output feature
  
```

## ***SUB***

### Description:

Indicates start of subroutine

### Syntax:

SUB [subroutine number]

[Subroutine Number] range is 0 to 31

### Examples:

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

## ***V[1-100]***

### Description:

Assign to variable.  
ACE-SDE has 100 variables [V1-V100]

### Syntax:

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

*Special case for BIT NOT:*

V[Variable Number] = ~[Argument]

[Argument] can be any of the following:

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

[Operation] can be any of the following

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

### Examples:

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

## **WAITX**

### Description:

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

### Syntax:

WAITX

### Examples:

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

## **WHILE**

Description:

Perform WHILE loop

Syntax:

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

[Argument] can be any of the following:

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

[Comparison] can be any of the following

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

Examples:

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



## **X**

Description:

**Command:** Perform X axis move to target location  
With other Axis moves in the same line, linear interpolation move is done.

Syntax:

X[value]  
X[variable]

Examples:

```
ABS          ;***Absolute move mode
X10000       ;***Move to position 10000
V10 = 1200   ;***Set variable 10 value to 1200
XV10        ;***Move axis to variable 10 value
```

## **ZHOMEX[+ or -]**

Description:

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

Syntax:

ZHOMEX[+ or -]

Examples:

```
ZHOMEX+     ;***Z Homes axis in positive direction
ZHOMEX-     ;***Z Homes axis in negative direction
```

---

## **ZOMEX[+ or -]**

### Description:

**Command:** Perform Zoming (homing only using Z-index) using current high speed, low speed, and acceleration.

### Syntax:

ZOMEX[+ or -]

### Examples:

ZOMEX+ ;\*\*\*Zomes axis in positive direction

ZOMEX- ;\*\*\*Zomes axis in negative direction

### ***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=0     ;* 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=0     ;* 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 DI0=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.

```

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=0          ;* If digital input 1 is on
        X1000         ;* Move to 1000
    ELSEIF DI2=0       ;* If digital input 2 is on
        X2000         ;* Move to 2000
    ELSEIF DI3=0       ;* If digital input 3 is on
        X3000         ;* Move to 3000
    ELSEIF DI5=0       ;* 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.

3061 Independence Drive. Suite H  
Livermore, CA 94551  
925-373-8800

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