

# DMX-ETH

## Integrated Step Motor Encoder/Driver/Controller with Ethernet Communication



COPYRIGHT © 2008 ARCUS,  
ALL RIGHTS RESERVED

First edition, March 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 ALM signal, added GUI description, updated picture
- 1.12 – Updated 14-pin out, updated ASCII protocol
- 1.13 – Updated NEMA 23 motor spec, updated sample configuration
- 1.14 – Begin on pg 1, do not disassemble warning, updated features list, removed 1 and 4-stack, updated over temp alarm, update pin-out drawings, updated DI/DO schematics, updated GUI section, remove pulse polarity, added latch input/z-index to MST, ignore PX for latch if SNL is enabled, updated SNL ratio spec, updated GS description, add DI[bit]/DO[bit] to ASCII table, added interactive language warning, updated “X” SA command, added sample standalone programs, added torque curves, updated ordering of ASCII table, added note for micro-step driver configuration, added boot-up sequence, updated SLS spec (ASCII table), added boot-sequence, flash-reset

**Firmware Compatibility:**  
**V217**

**Software Compatibility:**  
**V314**

## Table of Contents

1. Introduction.....	6
Model Numbers .....	7
2. Dimensions .....	8
3. Motor/Driver Specifications .....	10
Motor specifications.....	10
DMX-A2-DRV .....	10
Over Temperature Alarm.....	10
4. Connections.....	11
2 pin Connector Information.....	11
14 pin 2mm Connector Information .....	12
5. Electrical Specifications.....	13
Power Requirement.....	13
Communication Interfaces .....	13
+Lim, -Lim, Home Inputs, Latch, Digital Inputs.....	13
Digital Outputs.....	13
Limit/Home/Latch Sensors and Digital Input Connections.....	14
Digital Output Connections .....	14
6. Torque Curves.....	15
DMX-ETH-17-2/3 .....	15
DMX-ETH-23-2/3 .....	16
7. Getting Started .....	17
Typical Setup .....	17
Windows GUI.....	18
Main Control Screen .....	19
A. Status.....	20
B. Control.....	22
C. Product Information .....	23
D. Digital IO .....	23
E. Latch Input.....	24
F. On-the-fly speed change .....	24
G. Setup.....	25
H. Terminal .....	27
I. Standalone Program File Management.....	28
J. Standalone Program Editor .....	28
K. Standalone Program Compile/Download/Upload/View .....	29
L. Standalone Program Control .....	29
M. Variable Status .....	30
N. DMX-A2-DRV Alarm Status .....	30
8. Motion Control Overview.....	31
Motion Profile.....	31
On-the-fly Speed Change.....	33
Digital Inputs / Outputs.....	34
Motor Power .....	34
Polarity.....	34

Positional Moves.....	35
Jogging.....	35
Stopping Motor.....	35
Homing.....	35
Motor Position.....	35
Motor Status.....	36
Limit Inputs.....	36
StepNLoop Closed Loop Control.....	37
IP Address.....	39
Micro-step Driver Configuration.....	40
Standalone Programming.....	40
Boot-up Sequence.....	41
Hard Reset (Flash Memory).....	42
Storing to Flash.....	43
9. Communication.....	44
Socket Settings.....	44
ASCII Protocol.....	44
10. ASCII Language Specification.....	45
11. Standalone Language Specification.....	49
;.....	49
ABORTX.....	49
ABS.....	50
ACC.....	50
DELAY.....	51
DI.....	51
DI[1-2].....	52
DO.....	52
DO[1-2].....	53
DRVIC.....	54
DRVIT.....	54
DRVMS.....	55
DRVRC.....	55
EX.....	55
ECLEARX.....	56
ECLEARSX.....	56
ELSE.....	56
ELSEIF.....	57
END.....	58
ENDIF.....	58
ENDSUB.....	59
ENDWHILE.....	59
EO.....	60
GOSUB.....	60
HOMEX[+ or -].....	61
HSPD.....	61
IF.....	62

---

INC.....	63
JOGX[+ or -].....	63
LSPD.....	64
LT.....	64
LTE.....	65
LTP.....	65
LTS.....	65
MSTX.....	66
PX.....	67
PS.....	67
RW.....	68
RWSTAT.....	68
SCV.....	68
SL.....	69
SLSX.....	69
SSPD.....	70
SSPDM.....	70
STOPX.....	71
STORE.....	71
SUB.....	72
V[1-100].....	73
WAITX.....	74
WHILE.....	75
X.....	76
ZHOMEX[+ or -].....	76
ZOMEX[+ or -].....	77
Standalone Example Program 1.....	78
Standalone Example Program 2.....	78
Standalone Example Program 3.....	78
Standalone Example Program 4.....	79
Standalone Example Program 5.....	79
Standalone Example Program 6.....	80

# 1. Introduction

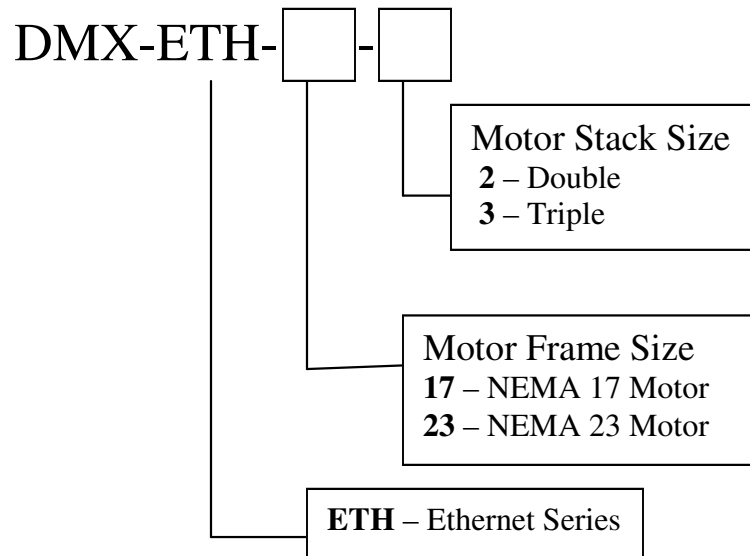
DMX-ETH is an all-in-one integrated motor package that combines all the motion components in to one convenient package.

## DMX-ETH Features

- Ethernet 10Mbps communication using Arcus ASCII command
- Standalone programming using easy to use text based programming language
- Opto-isolated +Limit/-Limit/Home inputs
- Opto-isolated 2 digital inputs
- Opto-isolated 2 digital outputs
- High speed position capture digital input
- 1M pulse/second controller output
- 1000 line incremental encoder (4000 counts/rev with 4x quadrature decoding)
- StepNLoop closed-loop control
- S-curve/Trapezoidal acceleration profile control
- Homing routine using:
  - o Home input only
  - o Z index encoder channel only
  - o Home input and Z index encoder channel
- 2-500 micro-step driver with effective resolution with 1.8 degree motor of up to 100,000 pulse/rev
- 12 to 48VDC Voltage Input
- Driver current from 100mA to 3.0A (peak current)
- NEMA 17/23 motor sizes available in different stack sizes.

## Model Numbers

### Main Product

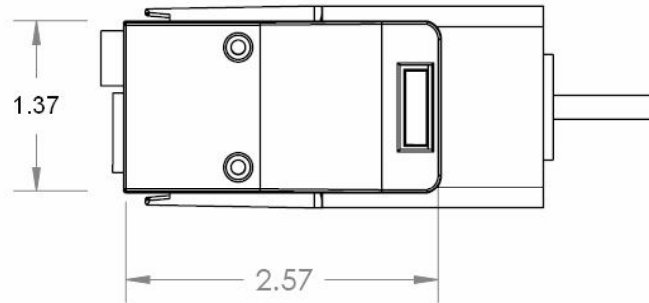


### Contacting Support

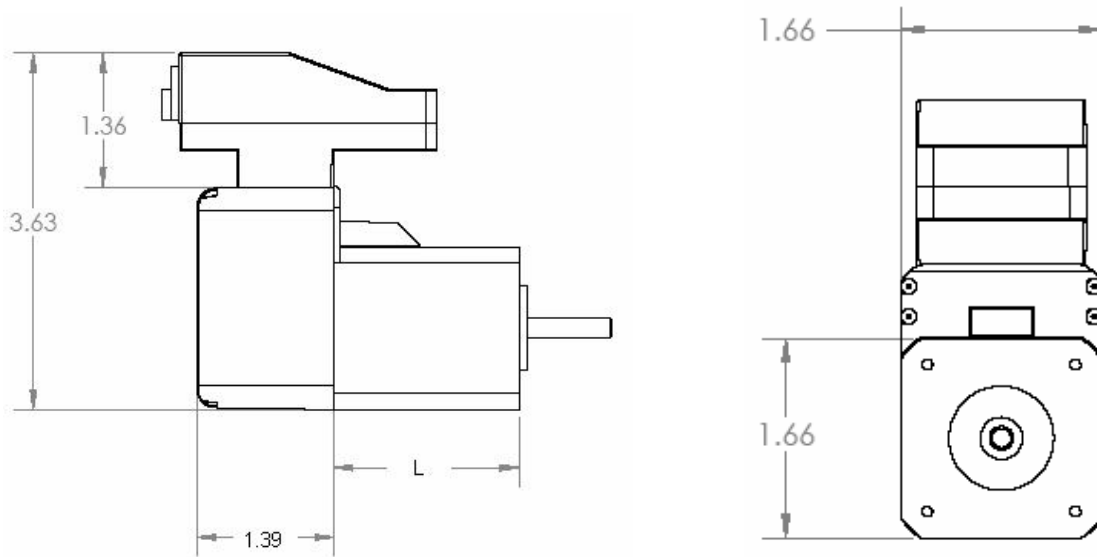
For technical support contact: [support@arcus-technology.com](mailto:support@arcus-technology.com).

Or, contact your local distributor for technical support.

## 2. Dimensions

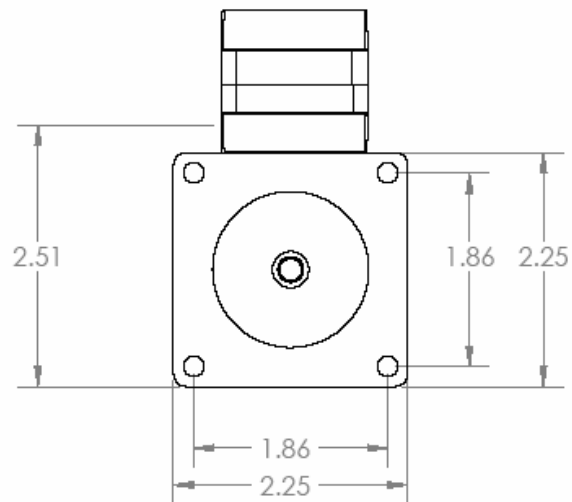
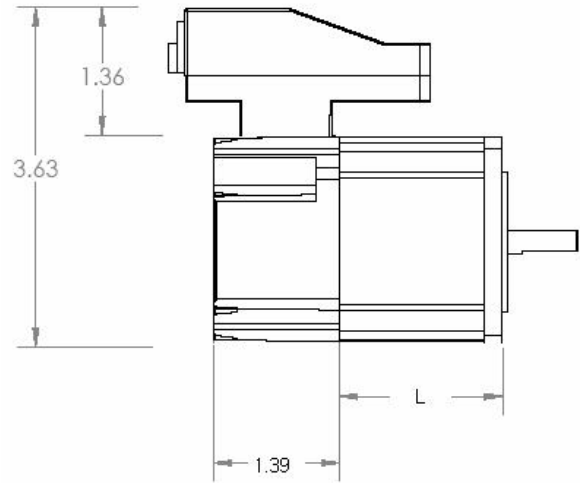


### *NEMA 17 Models*



<b>NEMA 17 Models</b>	<b>L (inches)</b>
DMX-ETH-17-2	1.58
DMX-ETH-17-3	1.89

**NEMA 23 Models**



NEMA 23 Models	L (inches)
DMX-ETH-23-2	2.2
DMX-ETH-23-3	3.1

### 3. Motor/Driver Specifications

#### Motor specifications

*Run Current and the Idle Current should not go over the maximum rated current for each motor size. Use the chart below as a reference on maximum rated current setting.*

NEMA Size	Stack Size	Max Amp / Phase	Holding Torque	Resistance / Phase	Inductance / Phase	Inertia
	Double	1.7A	0.44 N-m	1.5 Ohm	3.0 mH	0.28 oz-in <sup>2</sup>
17	Triple	2.0A	0.59 N-m	1.4 Ohm	2.7 mH	0.37 oz-in <sup>2</sup>
	Double	2.8A	0.95 N-m	0.9 Ohm	2.5 mH	1.64 oz-in <sup>2</sup>
23	Triple	2.8A	1.41 N-m	1.13 Ohm	3.6 mH	2.62 oz-in <sup>2</sup>

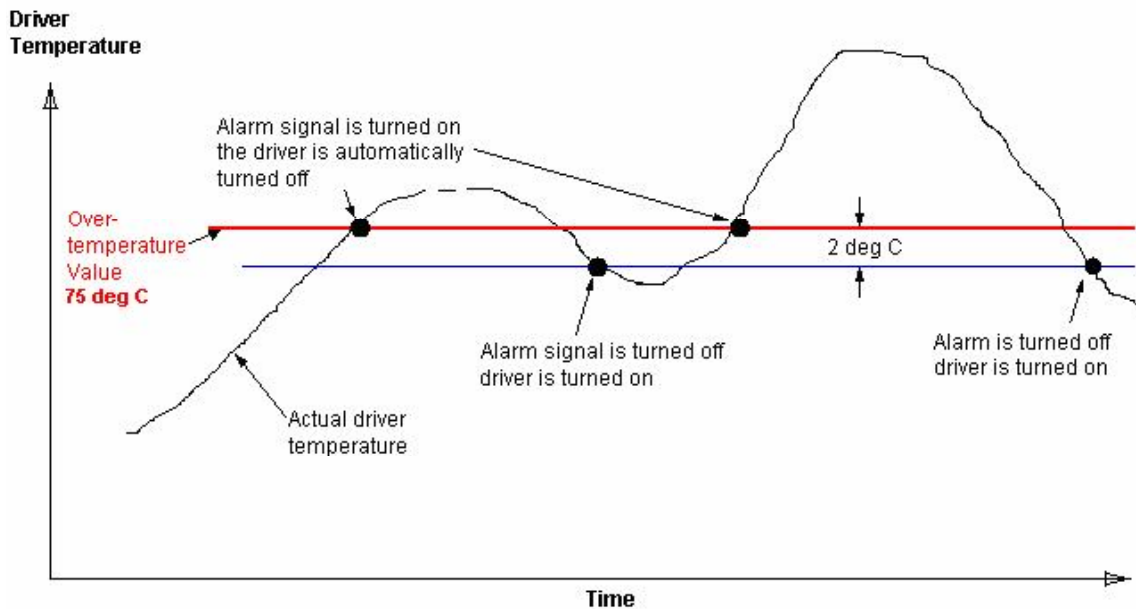
#### DMX-A2-DRV

The integrated driver included with DMX-ETH is DMX-A2-DRV. This product is also available as a standard driver + motor-only product. See website for details.

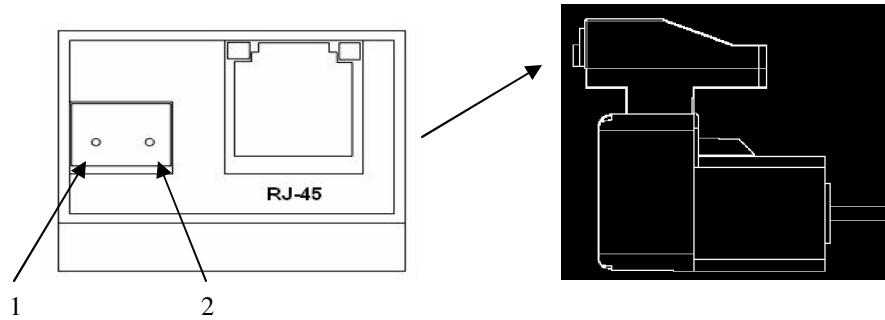
**Warning: Do not disassemble top controller from the DMX-A2-DRV. This may result in damage to both the controller and the driver if power is being supplied to the unit.**

#### Over Temperature Alarm

DMX-A2-DRV has a temperature sensor to detect over heating of the driver. Temperature sensing is done only when the driver is enabled. When the temperature goes over the over-temperature alarm value 75 degrees Celsius, the Alarm Output is turned on and the driver is turned off until the temperature goes below the 73 degrees Celsius.



## 4. Connections

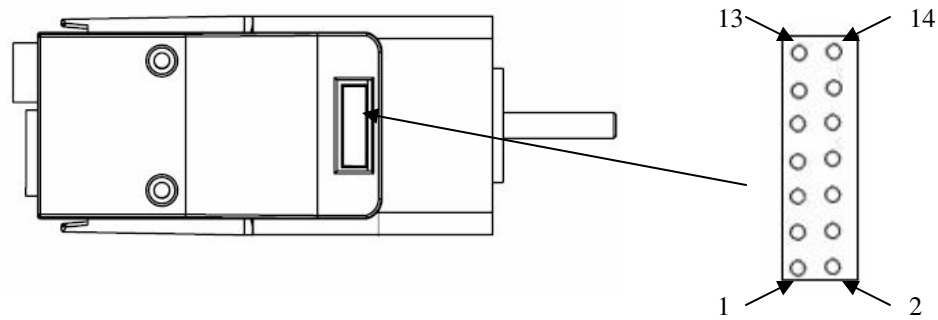


### *2 pin Connector Information*

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**



### 14 pin 2mm Connector Information

Pin #	Name	Description
1	OPTO IN	12-24V opto-isolator supply for limit, home, latch, and digital inputs
2	OPTO IN	12-24V opto-isolator supply for limit, home, latch, and digital inputs
3	LATCH	Latch opto-isolated input
4	HOME	Home opto-isolated input
5	-LIM	Minus limit opto-isolated input
6	+LIM	Plus limit opto-isolated input
7	DI1	Digital input 1
8	DI2	Digital input 2
9	DO1	Digital output 1
10	DO2	Digital output 2
11	NC	Not Connected
12	NC	Not Connected
13	NC	Not Connected
14	NC	Not Connected

Mating Connector Description:	14 pin 2mm dual row connector
Mating Connector Manufacturer:	HIROSE
Mating Connector Housing Part Number:	DF11-14DS-2C
Mating Connector Pin Part Number:	DF11-2414SC

---

## 5. Electrical Specifications

### ***Power Requirement***

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

### ***Communication Interfaces***

Ethernet : **Ethernet 10 Mbps - ASCII**

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

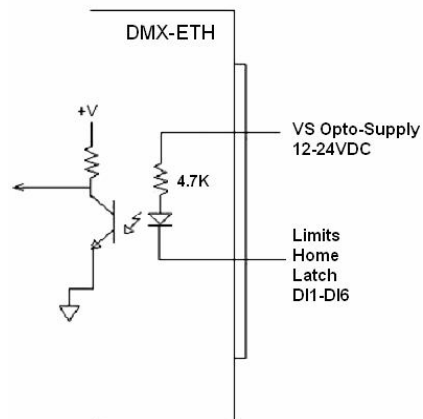
Type: **Opto-isolated sinking inputs**  
Opto voltage supply input: **+12 to +24 VDC**

### ***Digital Outputs***

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

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

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

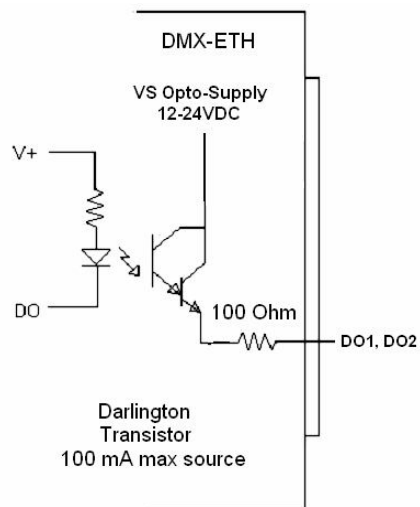


Connect the opto-supply using 12-24VDC power input. To trigger the Limit, Home, Latch, Digital Inputs, sink the line to ground of the power.

The DMX-ETH has 2 digital inputs that can be used for general purpose DIO. Note that if the Latch input can also be used as GPIO as long as the latch feature is not used.

## Digital Output Connections

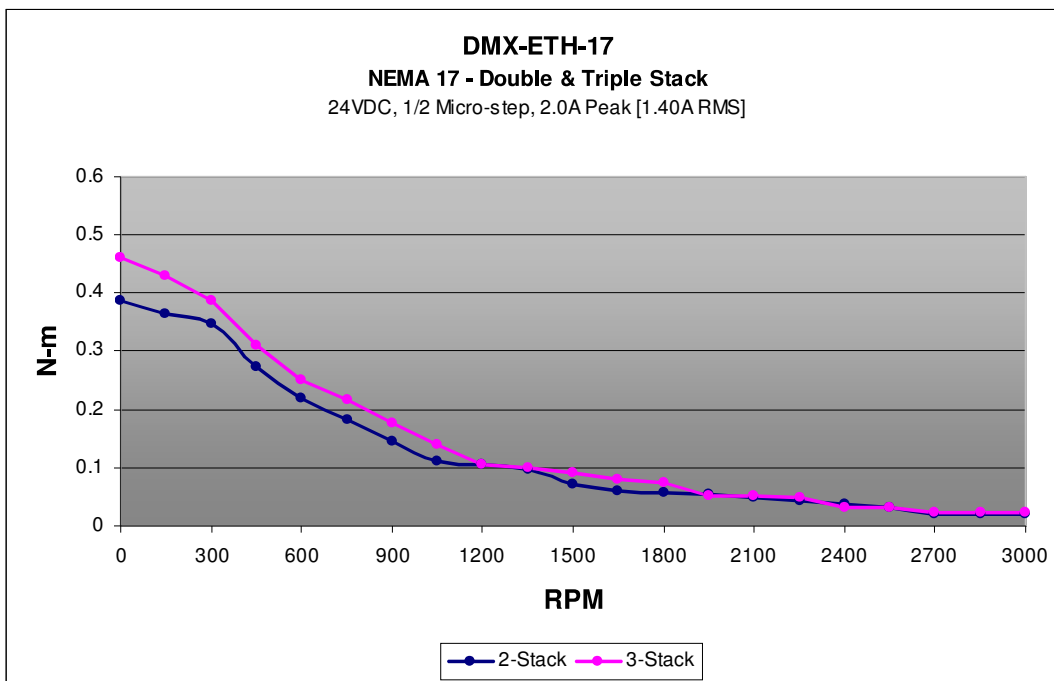
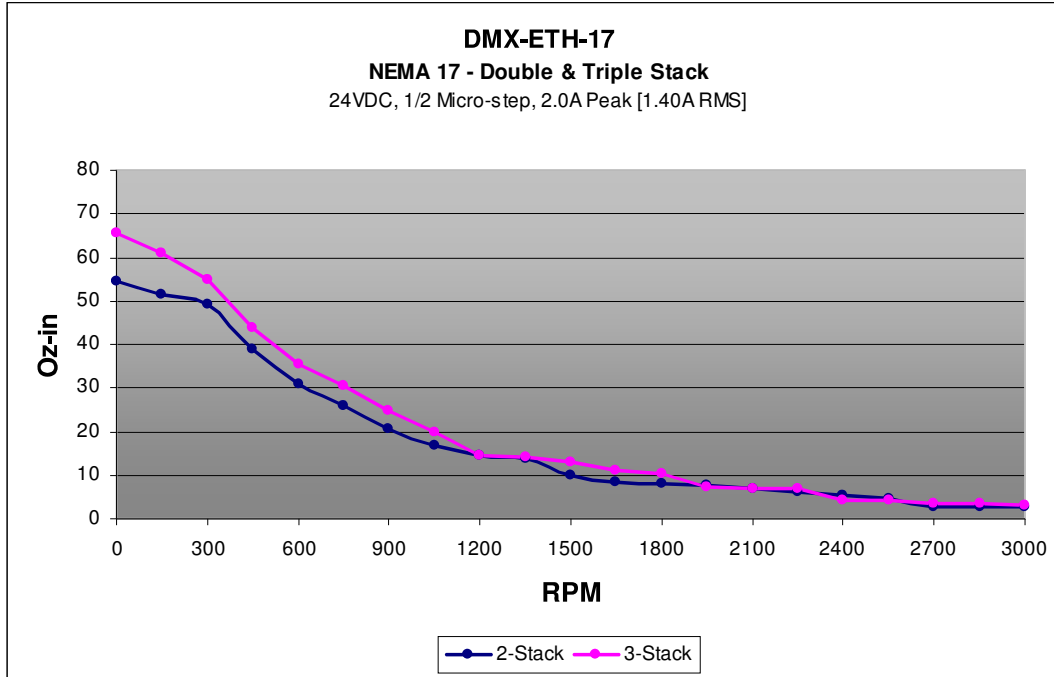
Digital outputs are opto-isolated open-emitter outputs using Darlington transistor that can source up to 100mA current at recommended maximum voltage of 24VDC.



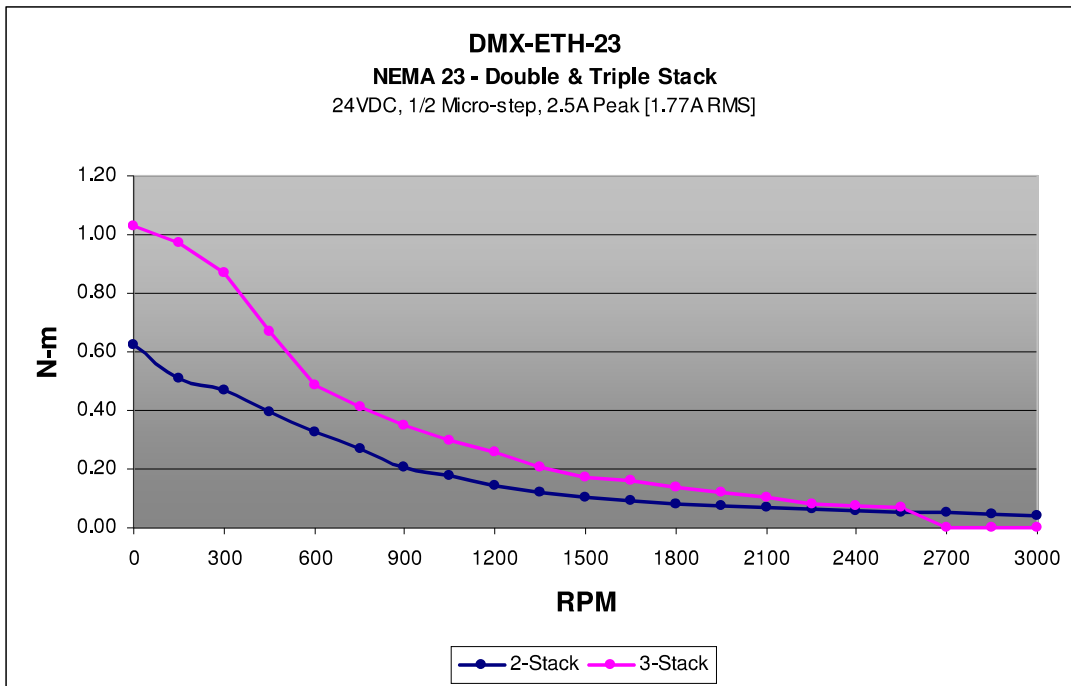
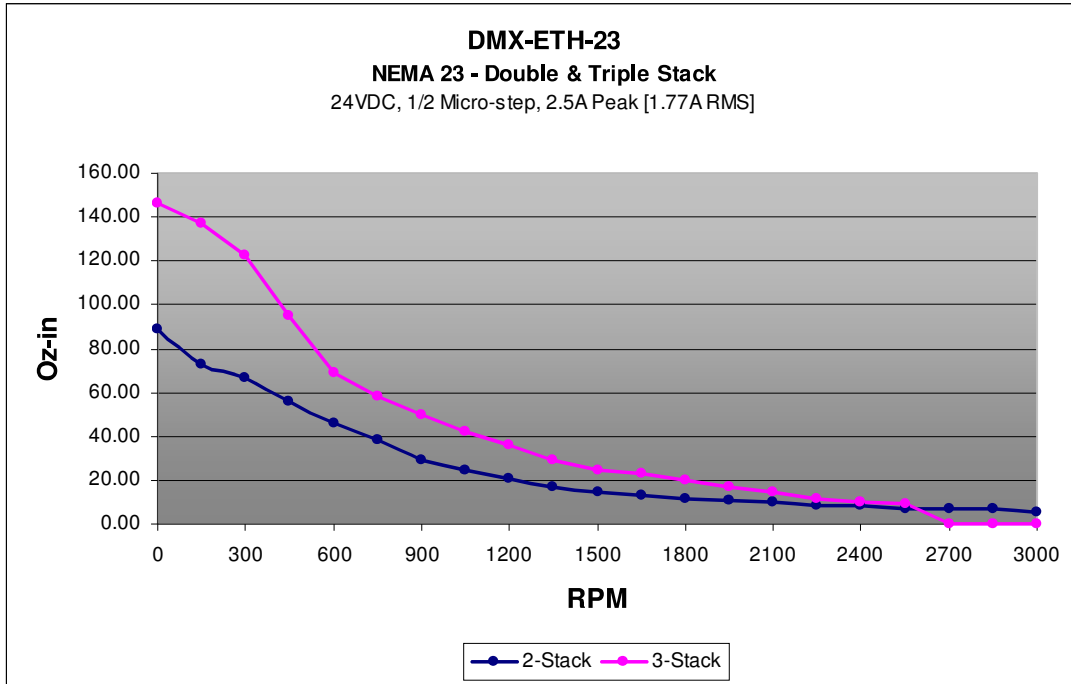
DMX-ETH has 2 digital outputs that can be used for general purpose DIO.

## 6. Torque Curves

**DMX-ETH-17-2/3**



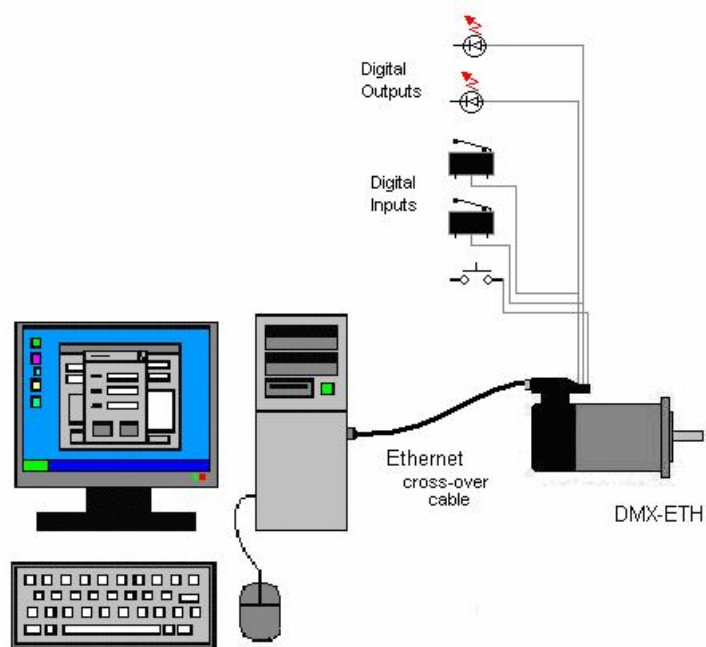
### DMX-ETH-23-2/3



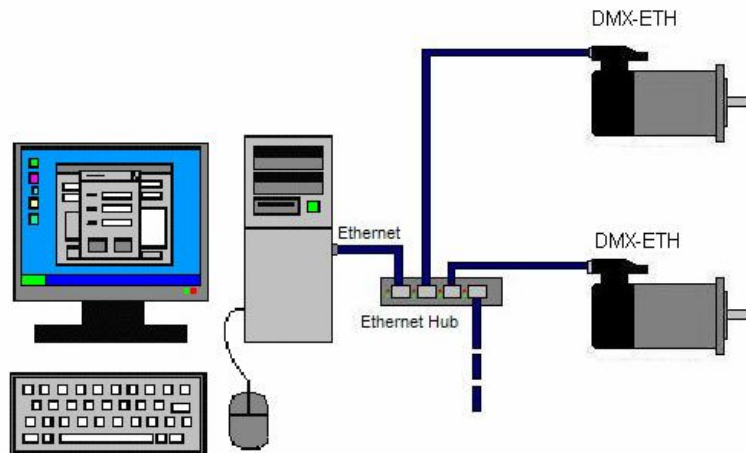
## 7. Getting Started

### Typical Setup

#### Point-to-point



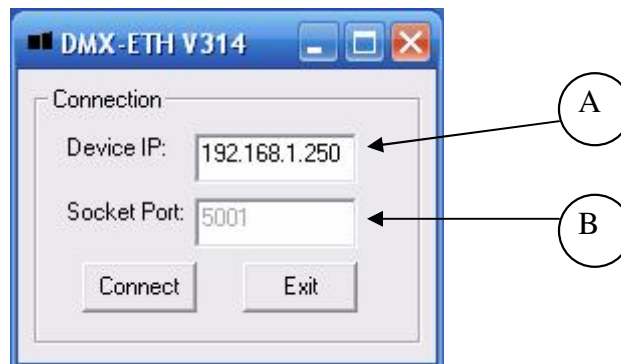
#### Network-based



## Windows GUI

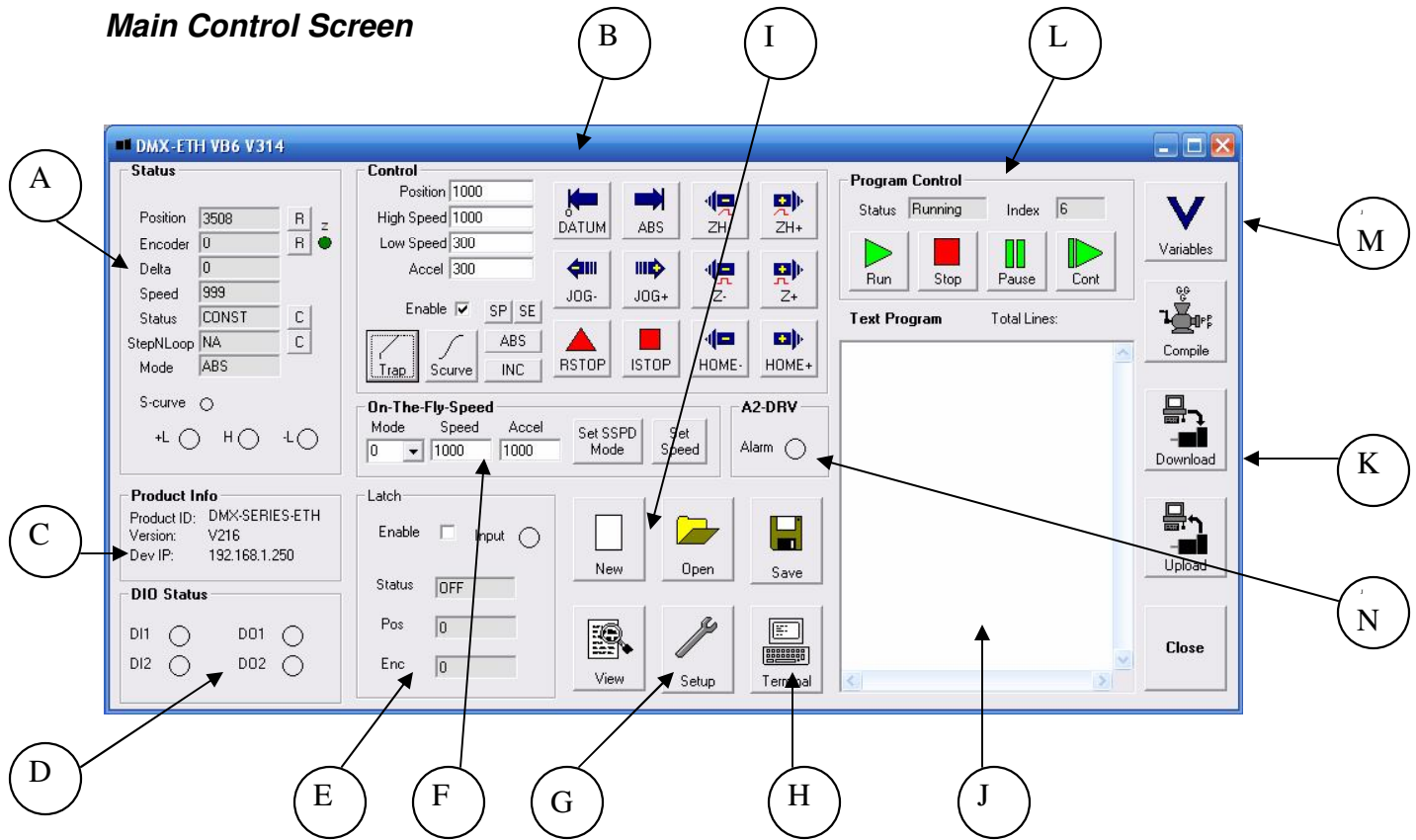
DMX-ETH comes with Windows GUI program to test, program, compile, download, and debug the controller.

Startup the DMX-ETH GUI program and you will see following screen.

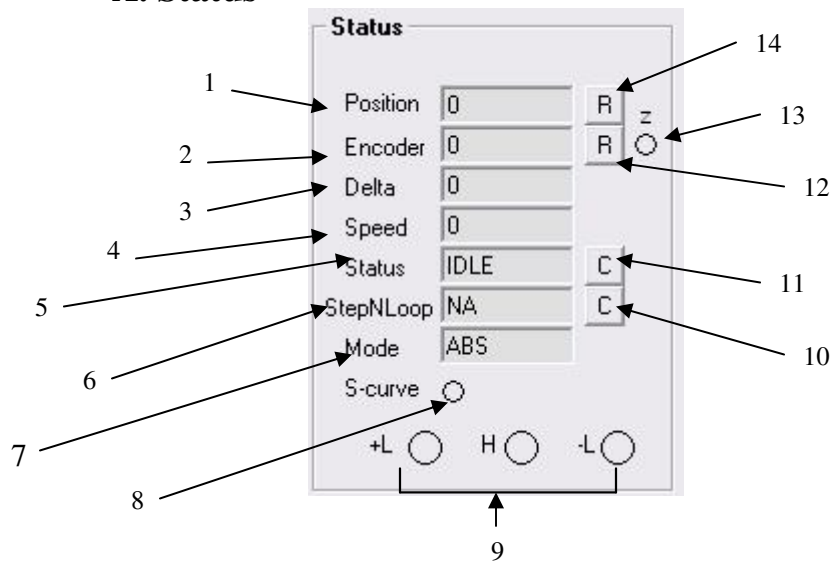


- A. Device IP of the DMX-ETH. The device IP later can be changed.
- B. Port number of the socket that must be opened to being Ethernet TCP/IP communication. This socket number can not be changed. Default is 5001.

### Main Control Screen



## A. Status

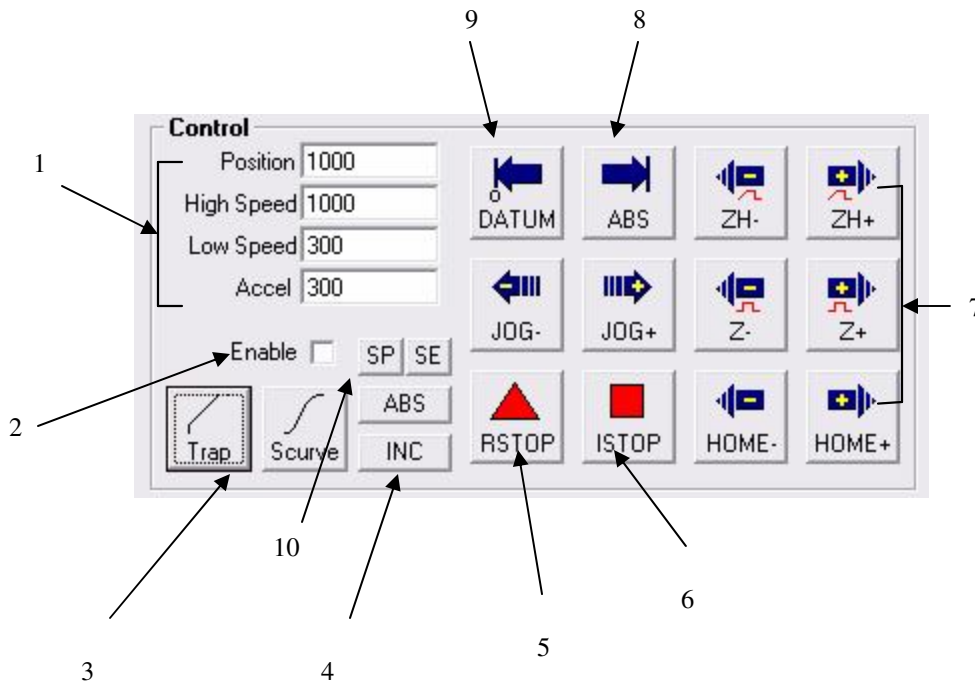


- 1. Pulse Counter** – displays the current pulse position counter. When StepNLoop is used, this displays the current target position
- 2. Encoder Counter** – displays the current encoder position counter.
- 3. Delta Counter** – valid only for StepNLoop. Displays the difference between the target position and the actual position.
- 4. Speed** – displays the current pulse speed output rate. Value is in pulses/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 routines are 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.

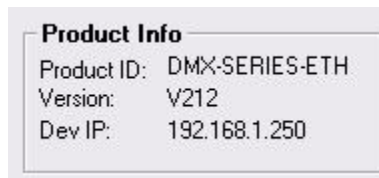
When in absolute mode, the axis will move to the absolute target position.  
 When in incremental mode, the axis will move incrementally.

**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).

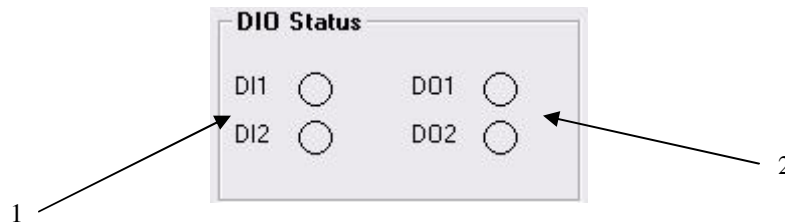
**10. Set Encoder / Set Position counter** – use this to set the pulse position “SP” or encoder position “SE”. The position will be set to the value in the “Position” field.

### C. Product Information



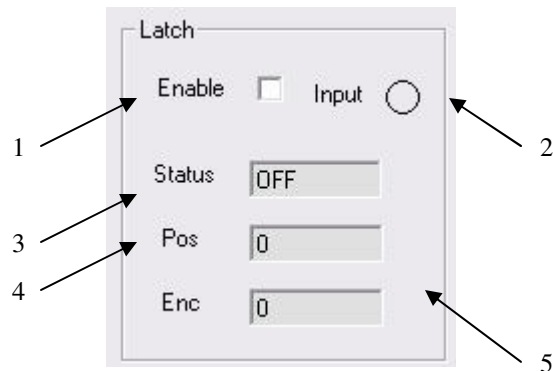
Product information and firmware version and the device IP is displayed. The device IP can be changed from the setup screen to support multiple devices on the Ethernet bus.

### D. Digital IO



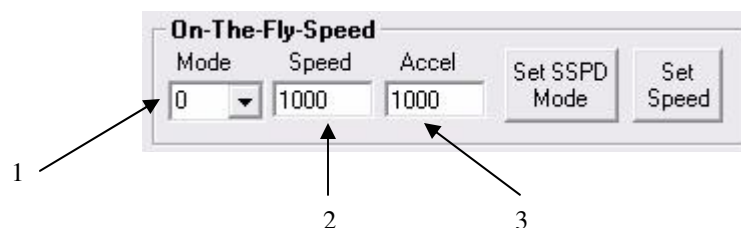
- Digital Input Status** – digital inputs can be used for general purpose use
- Digital Out Status and Control** – digital outputs can be used for general purpose use. The outputs can be triggered by clicking on the circle.

## E. Latch Input



1. **Enable** – Enable latch feature
2. **Status** – Latch input status
3. **State Status** – Latch input state status (OFF/ON/WAITING)
4. **Latched Pulse Position** – Pulse position value stored during the time that the latch input was triggered. If StepNLoop is enabled, the pulse value should not be ignored.
5. **Latched Encoder Position** – Encoder position value stored during the time that the latch input was triggered

## F. On-the-fly speed change

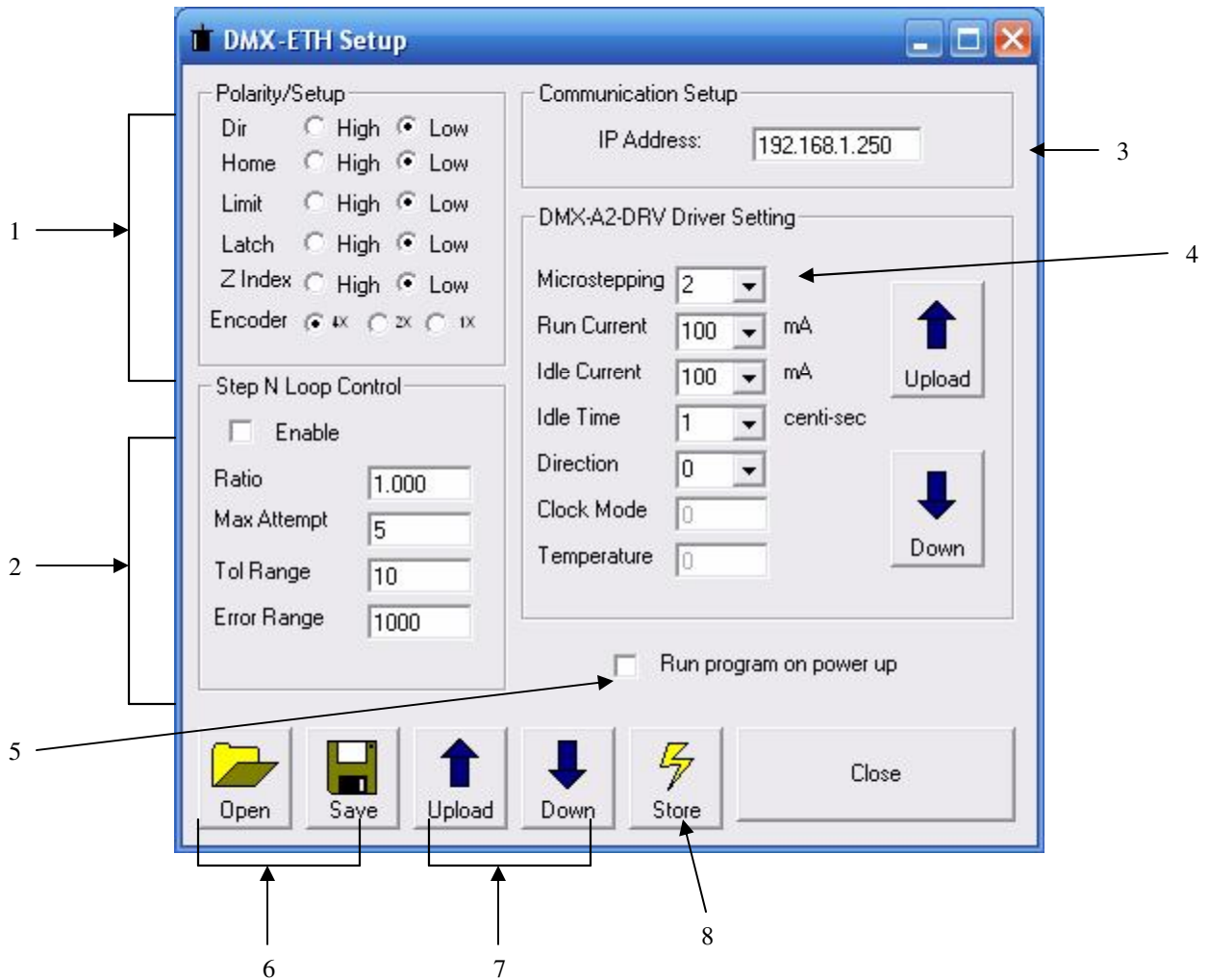


1. **SSPD Mode** – Set SSPDM value. Note that this value must be set before motion begins if on-the-fly speed feature is to be used. In order to make setting effective, click on “Set SSPD Mode”
2. **Speed** – Once the module is in motion, place the desired target speed in this field and click “Set Speed”. See SSPD mode section of manual for limitations.
3. **Acc** – This is the acceleration value used between on-the-fly speed change moves. See SSPD mode section of manual for limitations.

## G. 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. StepNLoop Control** – Using the encoder input, StepNLoop control allows closed loop position verification and correction for the moves. See StepNLoop control section for details.

**3. Communication Setup** – Set the desired IP address of the device

**4. DMX-A2-DRV 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)

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

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

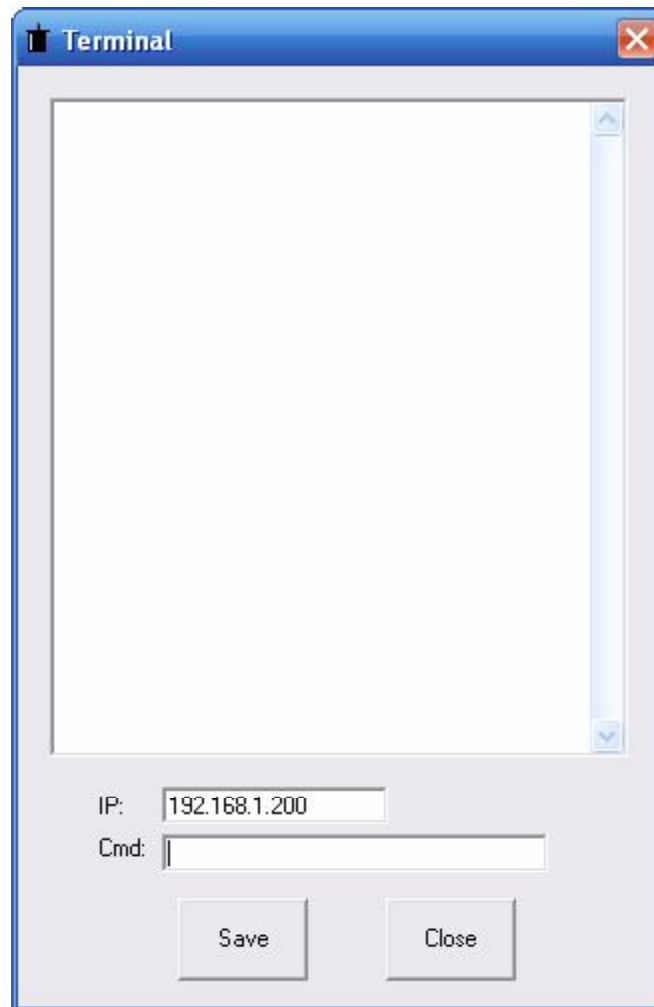
**7. Upload/Download** – Configuration values can be uploaded and downloaded. Note that if the configuration values are changed, it needs to be downloaded to take affect.

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

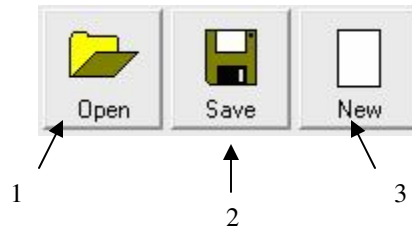
## H. Terminal



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

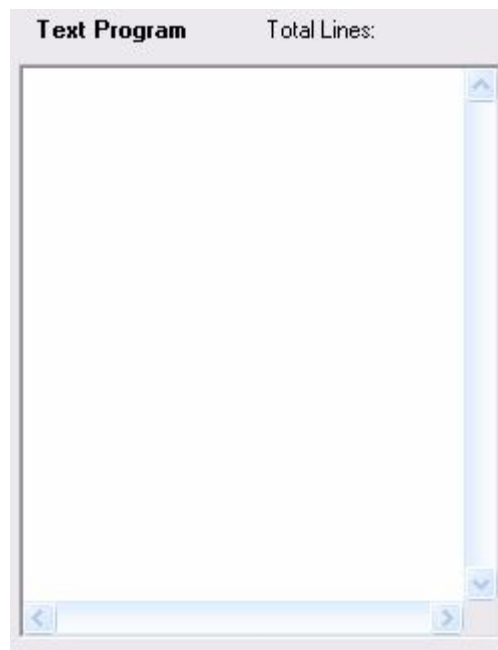


## I. Standalone Program File Management

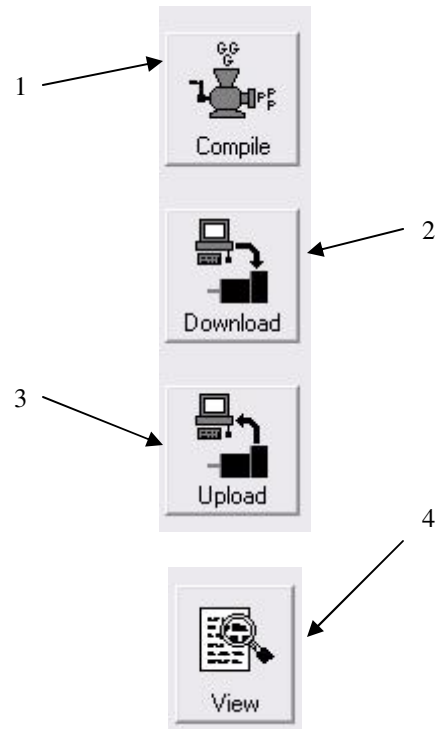


- a. Open – Open standalone program
- b. Save – Save standalone program
- c. New – Clear the standalone program editor

## J. Standalone Program Editor

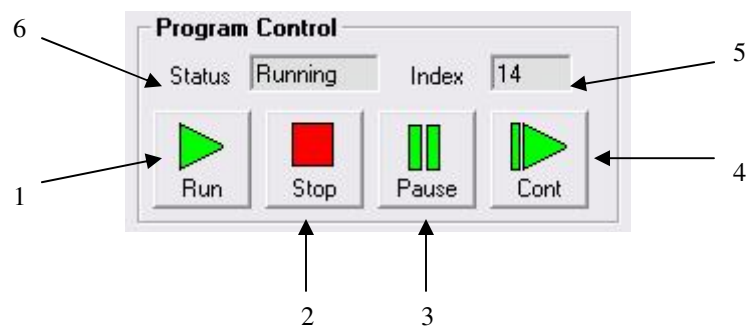


## K. Standalone Program Compile/Download/Upload/View



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

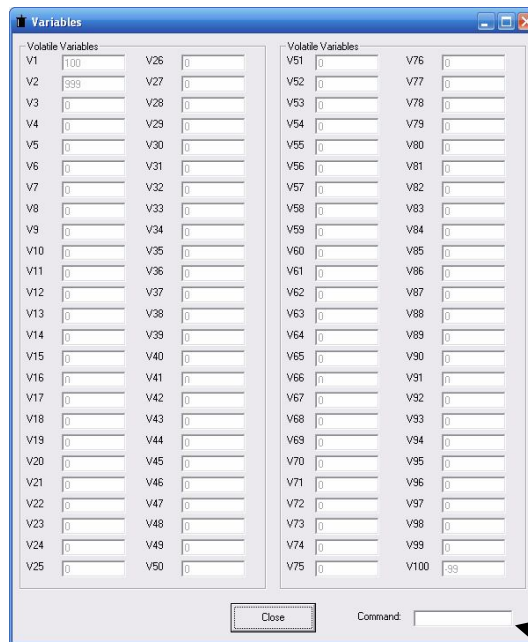
## L. Standalone Program Control



1. **Run** – Run the downloaded standalone program
2. **Stop** – Stop the downloaded standalone program
3. **Pause** – Pause the downloaded standalone program
4. **Continue** – Continue the downloaded standalone program. This operation only works if the program was previously paused

5. **Index** – Program counter (current low-level line) of standalone program execution
6. **Status** – Program status shows here. Following are possible program status: Idle, Running, Errored and Paused.

## M. 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.

## N. DMX-A2-DRV Alarm Status



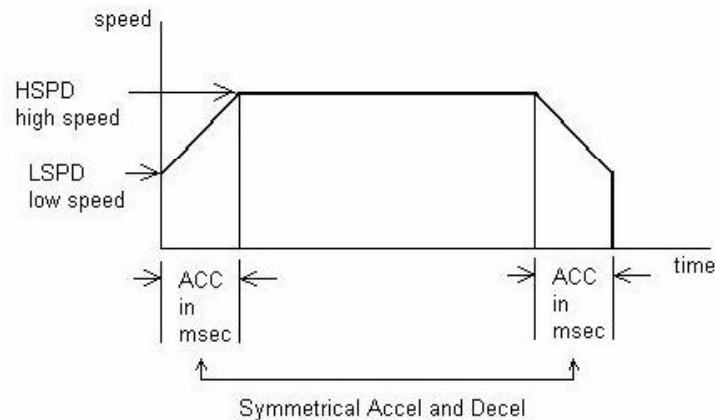
View status of alarm output of DMX-A2-DRV. If the alarm is on, this signifies that the driver is in over-temperature alarm state. In this condition, the motor will not operate until the temperature decreases below the temperature threshold.

## 8. 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***

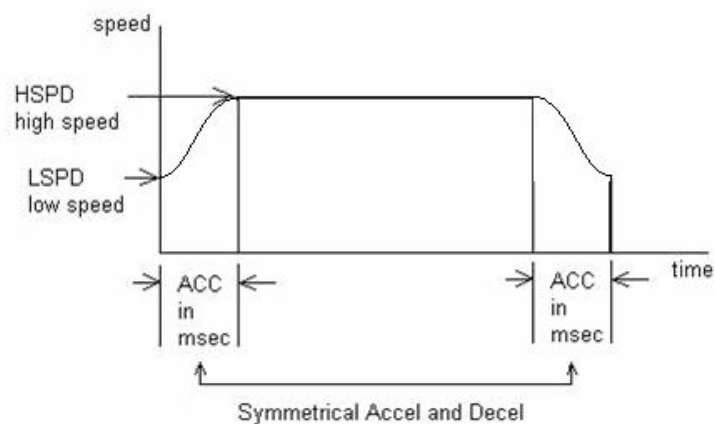
At default, DMX-ETH incorporates trapezoidal velocity profile as shown below.



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 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.

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

**SPEED WINDOWS (LSPD)**

SSPDM value	Lowest Speed [pps]	Highest Speed [pps]
0	SSPD not used	SSPD not used
1	10	16 K
2	10	30 K
3	10	80 K
4	10	160 K
5	25	300 K
6	50	800 K
7	110	1 M

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

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

**Note on acceleration:**

The allowable acceleration values depend on the LSPD and HSPD settings. These rules apply during regular motion operations as well as on-the-fly speed operations.

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

**SPEED WINDOWS (ACC)**

SSPDM value	HSPD is less than [pps]	Minimum ACC [ms]	Speed Delta (HSPD – LSPD) [pps]
1	16 K	2	500
2	30 K	1	1 K
3	80 K	1	2 K
4	160 K	1	4 K
5	300 K	1	8 K
6	800 K	1	18 K
7	1 M	1	39 K

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

### Encoder counter per second = PPS / 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)

*The example above also applies an on-the-fly-speed example. Simply replace HSPD with target speed, and LSPD with current speed.*

### **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 (LSPD)*” 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. See chart “*SPEED WINDOWS (ACC)*” chart in previous section.
- 3) To set speed on-the-fly, use **SSPD** command. Be sure to stay within the **SSPDM** speed window selected.
- 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**”.

### **Digital Inputs / Outputs**

DMX-ETH module comes with 2 digital inputs and 2 digital outputs which can be used as general purpose DIO.

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

<b>Bit</b>	<b>Description</b>
0	Digital Input 1
1	Digital Input 2

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

Read/set the digital output status by using the **DO** command. DO value must be within the range of 0-3. See description below:

<b>Bit</b>	<b>Description</b>
0	Digital Output 1
1	Digital Output 2

If digital output is turned on (i.e. the output is pulled to OPTO-SUPPLY), 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:

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

## **Positional Moves**

DMX-ETH 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:

<b>Bit</b>	<b>Description</b>
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 DMX-ETH 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.

Use the **LT** command to enable and disable latch feature. To read the latch status, use **LTS** command.

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

Following are return value description for **LTS** command:

<b>Return Value</b>	<b>Description</b>
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**

DMX-ETH 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:

<b>SNL Parameter</b>	<b>Description</b>
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
-1	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.

### **IP Address**

Set the IP address of the DMX-ETH module using the **IP** command. See default IP/socket settings below:

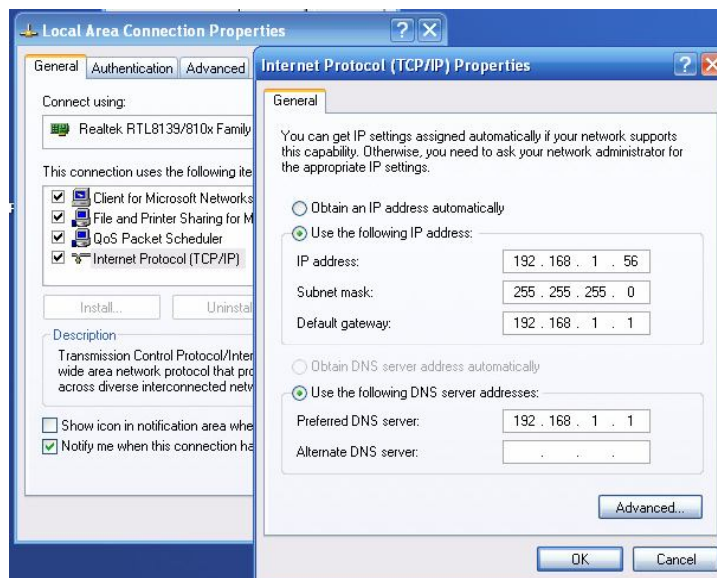
**IP: 192.168.1.250**

**Port: 5001**

Note: To begin communication with a factory default device, configure the PC with the following settings:

IP = 192.168.1.xxx  
Subnet Mask = 255.255.255.0

See sample configuration below:



### Changing the IP Address:

DMX-ETH provides the user with the ability to set the device IP of the module.

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

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

The built in driver of DMX-ETH 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]
<b>DRVIC</b>	Set/get idle current value of the driver [100-2800 mA]
<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 DMX-ETH 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 DMX-ETH 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, StepNLoop must be disabled. Otherwise, reading/writing operations will fail.**

### **Standalone Programming**

#### Standalone Program Specification:

Memory size: 7650 assembly lines ~ 44.5 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 initialized, the program will cease execution and go to error state. If SUB 31 is initialized 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 Ethernet communication using the **GS** command. The subroutines are referenced by their subroutine number [0-31]. If a subroutine number is not defined, the DMX-ETH will return with an error.

### ***Boot-up Sequence***

#### Initial Boot-up:

DMX-ETH takes approximately 5 seconds to boot up from the moment that power is supplied to the 2-pin connector.

#### Hard Reset detection:

After initial boot up, the DMX-ETH will begin to look for a hard reset input sequence. If the first input pattern is not detected within 5 seconds, boot-up will skip to “Connection detection”.

However, if the first input pattern is detected within 5 seconds, AND the full reset sequence is reached, the flash memory will be reset to factory defaults. Once the flash is reset, a power cycle needs to be performed in order to communicate via factory default settings. See *Hard Reset (Flash Memory)* section for details.

#### Connection detection:

If the hard reset input sequence is not detected, the device begins to look for an Ethernet connection. If an Ethernet connection is not detected within 7 seconds, the controller automatically times out and goes to the ***Connection Time-out State***. On the other hand, if, a connection is detected within the 7 second time frame, the controller will go to ***Full Connection State*** at the time of detection.

Note: During connection detection, the term “Ethernet connection” does not mean that a socket connection has been established. Instead, it means that the device is connected to an enabled / active Ethernet network / PC.

- ***Connection Time-out State:*** The controller could not detect an Ethernet connection in the allowable time frame. In this case, any standalone program that is set to run on boot-up will begin execution. Note that once this state is entered, it will no longer be possible to communicate with the controller via

Ethernet. To communicate via Ethernet, a power cycle must be performed to restart the boot-up sequence.

- **Full Connection State:** The controller found an Ethernet connection in the allowable time frame. In this case, communication established by opening a TCP/IP socket connection. Note that it is possible to close and re-open the connection multiple times. Any stand-alone program that is set to run on boot-up will also begin execution.

### **Hard Reset (Flash Memory)**

DMX-ETH comes with the ability to reset all the flash parameters to factory default settings. This is especially useful if the user has forgotten the device IP.

Hard reset is done by triggering the +LIM/-LIM/HOME/DI1/DI2 digital inputs in a particular sequence on boot up (*See Boot-up Sequence* section). There are a total of 7 steps that must be met in sequence. Each step has an input pattern that must be met before moving on to the next step. See chart below:

#### **Hard Reset Step Input Conditions**

<b>Step Condition</b>	<b>+LIM</b>	<b>-LIM</b>	<b>HOME</b>	<b>DI1</b>	<b>DI2</b>
1	ON	ON	OFF	OFF	ON
2	<b>OFF</b>	ON	OFF	OFF	ON
3	<b>ON</b>	ON	OFF	OFF	ON
4	ON	ON	<b>ON</b>	OFF	ON
5	ON	ON	<b>OFF</b>	OFF	ON
6	ON	ON	OFF	OFF	<b>OFF</b>
7	ON	ON	OFF	OFF	<b>ON</b>

Notes:

- ON: signal is pulled to GND of opto-supply (regardless of polarity setting)
- OFF: signal is not pulled to GND of opto-supply (regardless of polarity setting)
- For each condition, only one signal needs to change state. This signal is in bold

At each step, the enable/disable state of the motor will toggle. This is a tool to help signal to the user when to create the next step condition. See chart below:

### Hard Reset Step Motor Enable Status

Step Condition	Pre-trigger motor enable status	Post-triggered motor enable status
1	Disabled	Enabled
2	Enabled	Disabled
3	Disabled	Enabled
4	Enabled	Disabled
5	Disabled	Enabled
6	Enabled	Disabled
7	Disabled	Enabled

The controller will poll for the input pattern at each step for up to 10 sec. If the condition is not reached within the allotted 10 sec, the controller will stop looking for the hard reset sequence and continue its normal boot-up sequence (*See Boot-up Sequence* section). The motor will start as disabled.

However, once the condition for a step is met, it will immediately start looking for the next sequence (i.e. it is not necessary to wait the full 10 sec to trigger the next step).

If the DMX-ETH successfully triggers steps 1-7 in sequence, the flash is reset to factory default. At the end of the flash reset operation, the motor will stay enabled.

Once the flash is reset, a power cycle needs to be performed in order to communicate via factory default settings.

### ***Storing to Flash***

The following items are stored to flash:

- Device IP
- Polarity settings
- StepNLoop enable
- StepNLoop parameters
- Standalone program run on boot up
- 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.

## 9. Communication

DMX-ETH is 10Mbps Ethernet ASCII communication.

Communication between the PC/PLC and DMX-ETH is done using standard socket programming.

### **Socket Settings**

Port: 5001

### **ASCII Protocol**

Sending Command

ASCII command string in the format of  
[ASCII Command][NUL]

*[NUL] character has ASCII code 0.*

Receiving Reply

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

*[NUL] character has ASCII code 0.*

Examples:

For querying the x-axis polarity

Send: POL[NUL]

Reply: 7[NUL]

For jogging the x-motor in positive direction

Send: J+[NUL]

Reply: OK[NUL]

For aborting any motion in progress

Send: ABORT[NUL]

Reply: OK[NUL]

## 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.**

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

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
ALM	Get alarm output status of A2-DRV	0 – A2 driver is NOT in alarm state 1 – A2 driver is in alarm state
CLR	Clears limit error as well as StepNLoop error	OK
CLRS	Clears StepNLoop error. Note CLR also clears a StepNLoop error	OK
DI	Return status of digital inputs	2-bit number
DI[1-2]	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.	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
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
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 after using the “RW” command.	OK
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
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	DMX-SERIES-ETH
IP	Get current IP address of device	XXX.XXX.XXX.XXX
IP=[Value]	Set IP address of device	OK
INC	Set move mode to incremental	OK
J+	Jogs the motor in positive direction	OK
J-	Jogs the motor in negative direction	OK
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 status Bit 9 – Z-index status
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
SA[LineNumber]	Get standalone line LineNumber: [0,7649]	
SA[LineNumber]=[Value]	Set standalone line LineNumber: [0,7649]	
SASTAT	Get standalone program status	0-4

	0 – Stopped 1 – Running 2 – Paused 4 – In Error	
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 -1 – 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	[0-7]
SSPDM=[value]	Set on-the-fly speed change mode	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	OK

	100000, target move must be between 362143 and -162143	
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
  
```

## **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. DMX-ETH has 2 digital inputs.

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

### Syntax:

**Read:** [variable] = DI

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

IF DI=[value]  
ENDIF

### Examples:

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

## **DI[1-2]**

### Description:

**Read:** Gets the digital input value. DMX-ETH has 2 digital inputs.

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

### Syntax:

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

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

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

### Examples:

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

## **DO**

### Description:

**Read:** Gets the digital output value

**Write:** Sets the digital output value

DMX-ETH has 2 digital outputs.

If digital output is turned on (i.e. the output is pulled to OPTO-SUPPLY), 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

DMX-ETH has 2 digital outputs.

If digital output is turned on (i.e. the output is pulled to OPTO-SUPPLY), 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:

DO7=1 ;\*\*\*Turn DO7 on

DO6=1 ;\*\*\*Turn DO6 on

## ***DRVIC***

Description:

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

Syntax:

**Write:** DRVIC=[value]

Examples:

```

WHILE 1=1
    IF DI1 = 0
        SL=0
        EDIO=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  
 ;\*\*\*Disable DIO mode  
 ;\*\*\*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

## ***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

## ***ECLEARX***

**Description:**

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

**Syntax:**

**Write:** ECLEARX

**Examples:**

```
ECLEARX                ;***Clears motor error
```

## ***ECLEARSX***

**Description:**

**Write:** Clears StepNLoop error status. ECLEAR also clears a StepNLoop error

**Syntax:**

**Write:** ECLEARSX

**Examples:**

```
ECLEARSX                ;***Clears StepNLoop error
```

## ***ELSE***

**Description:**

Perform ELSE condition check as a part of IF statement

**Syntax:**

ELSE

**Examples:**

```
IF V1=1
    X1000                ;***If V1 is 1, then move to 1000
ELSE
    X-1000                ;***If V1 is not 1, then move to -1000
ENDIF
```

## **ELSEIF**

### Description:

Perform ELSEIF condition check as a part of the IF statement

### Syntax:

ELSEIF [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
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

## **GOSUB**

### Description:

Perform go to subroutine operation

Subroutine range is from 0 to 31.

**Note:** Subroutine definitions should be written AFTER the END statement.  
Subroutine 31 is reserved for error handling.

### 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

```

## **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
```

## ***SUB***

### Description:

Indicates start of subroutine

**Note:** Subroutine definitions should be written **AFTER** the END statement.  
Subroutine 31 is reserved for error handling.

### 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.  
DMX-ETH 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

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=1        ;* If digital input 1 is on, execute the statements
        X1000       ;* Move to zero
        X0          ;* Move to 1000
    ENDIF
ENDWHILE            ;* Go back to WHILE statement
END

```

### **Standalone Example Program 5**

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

```

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

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

```

## Standalone Example Program 6

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

```

HSPD=20000          ;* Set the high speed to 20000 pulses/sec
LSPD=1000           ;* Set the low speed to 1000 pulses/sec
ACC=300             ;* Set the acceleration to 300 msec
EO=1                ;* Enable the motor power
WHILE 1=1           ;* Forever loop
    IF DI1=1        ;* If digital input 1 is on
        X1000       ;* Move to 1000
    ELSEIF DI2=1    ;* If digital input 2 is on
        X2000       ;* Move to 2000
    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)