Performax 2ED-SA

2-Axis
Stepper Motor Controller/Driver
Standalone Version

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

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

**Revision History:**

- 1.00 – 1st Revision
- 1.13 – 2nd Revision
- 1.14 – 3rd Revision
- 1.15 – 4th Revision

**Firmware Compatibility:**

†V123BL

†If your module’s firmware version number is less than the listed value, contact Arcus for the appropriate documentation.
# Table of Contents

1. Introduction __________________________________________________________ 6  
   Features _____________________________________________________________ 6  
2. Electrical Specifications ________________________________________________ 7  
   Power Requirement ____________________________________________________ 7  
   Temperature Ratings † _____________________________ ____________________ 7  
   Pulse, Dir, Enable Outputs ______________________________________________ 7  
   Digital Inputs ________________________________________________________ 7  
   Digital Outputs _______________________________________________________ 7  
3. Dimensions __________________________________________________________ 8  
4. Pin Descriptions _______________________________ _______________________ 9  
   2-Pin Connector (5.08mm) – Controller Power _______ _______________________ 9  
   3-pin Connector (3.81mm) ______________________________________________ 9  
   4-Pin Connectors (2.54mm) ____________________________________________ 10  
   2-Pin Connector (2.54mm) – Driver Power ___________ _____________________ 10  
   50-Pin Control/Encoder IO _____________________________________________ 11  
   Interface Circuit _________________________________ ____________________ 13  
   Digital Outputs ______________________________________________________ 14  
   Digital Inputs ____________________________________ ___________________ 14  
   Encoder Inputs ____________________________________ __________________ 15  
   Analog Inputs ______________________________________________________ 15  
   Driver Current and Microstep Setting ____________________________________ 15  
5. Getting Started ________________________________ ______________________ 17  
   Typical Setup _____________________________________ __________________ 17  
   Main Control Screen __________________________________________________ 18  
6. Motion Control Feature Overview _______________________________________ 28  
   Motion Profile _______________________________________________________ 28  
   Pulse Speed _________________________________________________________ 29  
   On-The-Fly Speed Change _____________________________________________ 29  
   Motor Status ________________________________________________________ 30  
   Individual/Linear Interpolation Moves____________________________________ 30  
   On-The-Fly Target Position Change______________________________________ 31  
   Homing ____________________________________________________________ 31  
   Home Input Only (High speed only) .................................................. 31  
   Home Input Only (High speed and low speed).................................... 32  
   Limit Only___________________________________________________________ 33  
   Home and Z-index ___________________________________________________ 33  
   Z-index only _________________________________________________________ 34  
   Jogging ___________________________________________ _________________ 34  
   Stopping Motor ______________________________________________________ 34  
   Motor Position ______________________________________________________ 34  
   Polarity ____________________________________________________________ 35  
   Limits ____________________________________________________________ 35  
   Digital Inputs/Outputs and Enable Outputs ________________________________ 35
Analog Inputs 37
Joystick Control 37
StepNLoop Closed Loop Control 38
Device Number 41
Standalone Program Specification 41
Storing to Flash 42
7. Communication 43
   USB Communication API Functions 43
   USB Communication Issues 44
8. Communication – RS-485 (ASCII) 45
   Communication Port Settings 45
   ASCII Protocol 45
9. ASCII Language Specification 46
   Error Codes 50
10. Standalone Language Specification 51
    ; 51
    ABORT 51
    ABORT[axis] 51
    ABS 51
    ACC 52
    ACC[axis] 52
    AI[1-2] 53
    DEC 53
    DEC[axis] 53
    DELAY 54
    DI 54
    DI[1-8] 54
    DO 55
    DO[1-8] 55
    E[axis] 55
    ECLEAR[axis] 56
    ELSE 56
    ELSEIF 56
    END 56
    ENDIF 57
    ENDSUB 58
    ENDWHILE 58
    EO 58
    EO[1-2] 59
    GOSUB 59
    HLHOME[axis][+ or -] 60
    HOME[axis][+ or -] 60
    HSPD 60
    HSPD[axis] 60
    IF 61
    INC 61
JOG[axis] _______________________________________ 62
JOYENA ________________________________________ 62
JOYHS[axis] ______________________________________ 62
JOYDEL[axis] ______________________________________ 62
JOYN[O[axis] ______________________________________ 63
JOYNI[axis] ______________________________________ 63
JOYPI[axis] ______________________________________ 63
JOYPO[axis] ______________________________________ 63
JOYTOL[axis] _____________________________________ 64
LHOME[axis][+ or -] _________________________________ 64
LSPD ____________________________ 64
LSPD[axis] _________________________________________ 64
MST[axis] _________________________________________ 65
P[axis] _____________________________________________ 65
PRG ____________________________ 65
PT[axis] __________________________________________ 66
SCV[axis] _________________________________________ 66
SL[axis] __________________________________________ 67
SL[axis] __________________________________________ 67
SR[0,1] __________________________________________ 67
SSPD[axis] ________________________________________ 67
SSPDM[axis] ______________________________________ 68
STOP ____________________________________________ 68
STOP[axis] _________________________________________ 68
STORE ___________________________________________ 69
SUB ______________________________________________ 69
TOC ______________________________________________ 69
V _________________________________________________ 69
WAIT _____________________________________________ 70
WHILE ____________________________________________ 71
X ________________________________________________ 71
Y ________________________________________________ 71
ZHOME[axis][+ or -] _________________________________ 72
ZOME[axis][+ or -] __________________________________ 72

11. Example Standalone Programs ________________________ 73
   Standalone Example Program 1 – Single Thread __________ 73
   Standalone Example Program 2 – Single Thread __________ 73
   Standalone Example Program 3 – Single Thread __________ 73
   Standalone Example Program 4 – Single Thread __________ 74
   Standalone Example Program 5 – Single Thread __________ 74
   Standalone Example Program 6 – Single Thread __________ 75
   Standalone Example Program 7 – Multi Thread ____________ 76
   Standalone Example Program 8 – Multi Thread ____________ 77

Appendix A: Speed Settings ______________________________ 78
   Acceleration/Deceleration Range _______________________ 78
   Acceleration/Deceleration Range – Positional Move ________ 79
1. Introduction

PMX-2ED-SA is an advanced 2 axis stepper standalone programmable motion controller.

Communication to the PMX-2ED-SA can be established over USB. It is also possible to download a standalone program to the device and have it run independent of a host.

Features

PMX-2ED-SA

- USB 2.0 communication
- RS-485 ASCII communication
  - 9600, 19200, 38400, 57600, 115200 bps
- Standalone programmable
- Maximum pulse output rate of 400K PPS
- Trapezoidal or s-curve acceleration
- On-the-fly speed change
- XY linear coordinated motion
- A/B/Z differential encoder inputs [Max frequency of 5 MHz]
  - StepNLoop closed loop control (position verification)
- Pulse/Dir/Enable open collector outputs per axis
- Opto-isolated I/O
  - 8 x inputs
  - 8 x outputs
  - +Limit/-Limit/Home inputs per axis
- Homing routines:
  - Home input only (high speed)
  - Home input only (high speed + low speed)
  - Limit only
  - Z-index encoder channel only
  - Home input + Z index encoder channel
- 2 x 10-bit analog inputs
  - Joystick control
- Stepper driver
  - 12-24 VDC
  - 1.5 Amp max current setting (peak current)
  - Full step, 2, 4, or 8 micro-step setting
  - Max pulse input rate of 400K

Contacting Support

For technical support contact: support@arcus-technology.com.

Or, contact your local distributor for technical support.
2. Electrical Specifications

**Power Requirement**
- Regulated Voltage: +12 to +24 VDC
- Current (Max): 1.5A (Peak)

**Temperature Ratings †**
- Operating Temperature: -20°C to +80°C
- Storage Temperature: -55°C to +150°C
† Based on component ratings

**Pulse, Dir, Enable Outputs**
- Type: Open-collector output
- Max sink voltage: +24 VDC
- Max sink current: 40 mA

**Digital Inputs**
- Type: Opto-isolated inputs (NPN)
- Voltage range: +12V to +24VDC
- Max forward current: 40 mA

**Digital Outputs**
- Type: Opto-isolated outputs (NPN)
- Max voltage: +12V to +24VDC
- Max source current: 90 mA
3. Dimensions

Figure 3.0
4. Pin Descriptions

In order for PMX-2ED-SA to operate, it must be supplied with +12VDC to +24VDC. Power pins as well as communication port pin outs are shown below.

**2-Pin Connector (5.08mm) – Controller Power**

<table>
<thead>
<tr>
<th>Pin #</th>
<th>In/Out</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I</td>
<td>G</td>
<td>Ground</td>
</tr>
<tr>
<td>2</td>
<td>I</td>
<td>V+</td>
<td>Power Input +12 to +24 VDC</td>
</tr>
</tbody>
</table>

Table 4.0

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

† Other 5.08mm compatible connectors can be used.

**3-pin Connector (3.81mm)**

<table>
<thead>
<tr>
<th>Pin #</th>
<th>In/Out</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I/O</td>
<td>485+</td>
<td>RS-485 plus signal</td>
</tr>
<tr>
<td>2</td>
<td>I/O</td>
<td>485-</td>
<td>RS-485 minus signal</td>
</tr>
<tr>
<td>2</td>
<td>I</td>
<td>G</td>
<td>Ground</td>
</tr>
</tbody>
</table>

Table 4.1

Mating Connector Description: 3 pin 0.15” (3.81mm) connector  
Mating Connector Manufacturer: On-Shore  
Mating Connector Manufacturer Part: †EDZ1550/3

† Other 3.81 compatible connectors can be used.
Figure 4.1

**4-Pin Connectors (2.54mm)**

<table>
<thead>
<tr>
<th>Pin #</th>
<th>In/Out</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O</td>
<td>AX</td>
<td>Bi-polar Step Motor – X-axis Phase A</td>
</tr>
<tr>
<td>2</td>
<td>O</td>
<td>/AX</td>
<td>Bi-polar Step Motor – X-axis Phase /A</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>BX</td>
<td>Bi-polar Step Motor – X-axis Phase B</td>
</tr>
<tr>
<td>4</td>
<td>O</td>
<td>/BX</td>
<td>Bi-polar Step Motor – X-axis Phase /B</td>
</tr>
</tbody>
</table>

Table 4.2

<table>
<thead>
<tr>
<th>Pin #</th>
<th>In/Out</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O</td>
<td>AY</td>
<td>Bi-polar Step Motor – Y-axis Phase A</td>
</tr>
<tr>
<td>2</td>
<td>O</td>
<td>/AY</td>
<td>Bi-polar Step Motor – Y-axis Phase /A</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>BY</td>
<td>Bi-polar Step Motor – Y-axis Phase B</td>
</tr>
<tr>
<td>4</td>
<td>O</td>
<td>/BY</td>
<td>Bi-polar Step Motor – Y-axis Phase /B</td>
</tr>
</tbody>
</table>

Table 4.3

- Mating Connector Description: 4 pin 0.1" (2.54mm) connector
- Mating Connector Manufacturer: AMP/Tyco
- Mating Connector Manufacturer Part: 770602-4

**2-Pin Connector (2.54mm) – Driver Power**

The stepper driver power is separate from the controller power. Voltage range is also 12-24VDC.

<table>
<thead>
<tr>
<th>Pin #</th>
<th>In/Out</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I</td>
<td>G</td>
<td>Ground</td>
</tr>
<tr>
<td>2</td>
<td>I</td>
<td>V+</td>
<td>Power Input +12 to +24 VDC</td>
</tr>
</tbody>
</table>

Table 4.4

- Mating Connector Description: 2 pin 0.2” (5.08mm) connector
- Mating Connector Manufacturer: On-Shore
- Mating Connector Manufacturer Part: †EDZ950/2
### 50-Pin Control/Encoder IO

<table>
<thead>
<tr>
<th>Pin #</th>
<th>In/Out</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O</td>
<td>+5V</td>
<td>+5V</td>
</tr>
<tr>
<td>2</td>
<td>O</td>
<td>G</td>
<td>Ground</td>
</tr>
<tr>
<td>3</td>
<td>I</td>
<td>AX</td>
<td>A Channel Encoder Input [X-Axis]</td>
</tr>
<tr>
<td>4</td>
<td>I</td>
<td>/AX</td>
<td>/A Channel Encoder Input [X-Axis]</td>
</tr>
<tr>
<td>5</td>
<td>I</td>
<td>BX</td>
<td>B Channel Encoder Input [X-Axis]</td>
</tr>
<tr>
<td>6</td>
<td>I</td>
<td>/BX</td>
<td>/B Channel Encoder Input [X-Axis]</td>
</tr>
<tr>
<td>7</td>
<td>I</td>
<td>ZX</td>
<td>Z Index Encoder Input [X-Axis]</td>
</tr>
<tr>
<td>8</td>
<td>I</td>
<td>/ZX</td>
<td>/Z Index Encoder Input [X-Axis]</td>
</tr>
<tr>
<td>9</td>
<td>I</td>
<td>AY</td>
<td>A Channel Encoder Input [Y-Axis]</td>
</tr>
<tr>
<td>10</td>
<td>I</td>
<td>/AY</td>
<td>/A Channel Encoder Input [Y-Axis]</td>
</tr>
<tr>
<td>11</td>
<td>I</td>
<td>BY</td>
<td>B Channel Encoder Input [Y-Axis]</td>
</tr>
<tr>
<td>12</td>
<td>I</td>
<td>/BY</td>
<td>/B Channel Encoder Input [Y-Axis]</td>
</tr>
<tr>
<td>13</td>
<td>I</td>
<td>ZY</td>
<td>Z Index Encoder Input [Y-Axis]</td>
</tr>
<tr>
<td>14</td>
<td>I</td>
<td>/ZY</td>
<td>/Z Index Encoder Input [Y-Axis]</td>
</tr>
<tr>
<td>15</td>
<td>I</td>
<td>AI1</td>
<td>Analog Input 1</td>
</tr>
<tr>
<td>16</td>
<td>I</td>
<td>AI2</td>
<td>Analog Input 2</td>
</tr>
<tr>
<td>17</td>
<td>I</td>
<td>+LX</td>
<td>+Limit [X-Axis]</td>
</tr>
<tr>
<td>18</td>
<td>I</td>
<td>-LX</td>
<td>-Limit [X-Axis]</td>
</tr>
<tr>
<td>19</td>
<td>I</td>
<td>HX</td>
<td>Home [X-Axis]</td>
</tr>
<tr>
<td>20</td>
<td>NC</td>
<td>NC</td>
<td>No Connection</td>
</tr>
<tr>
<td>21</td>
<td>I</td>
<td>+LY</td>
<td>+Limit [Y-Axis]</td>
</tr>
<tr>
<td>22</td>
<td>I</td>
<td>-LY</td>
<td>-Limit [Y-Axis]</td>
</tr>
<tr>
<td>23</td>
<td>I</td>
<td>HY</td>
<td>Home [Y-Axis]</td>
</tr>
<tr>
<td>24</td>
<td>NC</td>
<td>NC</td>
<td>No Connection</td>
</tr>
<tr>
<td>25</td>
<td>NC</td>
<td>NC</td>
<td>No Connection</td>
</tr>
<tr>
<td>26</td>
<td>NC</td>
<td>NC</td>
<td>No Connection</td>
</tr>
<tr>
<td>27</td>
<td>NC</td>
<td>NC</td>
<td>No Connection</td>
</tr>
<tr>
<td>28</td>
<td>NC</td>
<td>NC</td>
<td>No Connection</td>
</tr>
<tr>
<td>29</td>
<td>NC</td>
<td>NC</td>
<td>No Connection</td>
</tr>
<tr>
<td>30</td>
<td>NC</td>
<td>NC</td>
<td>No Connection</td>
</tr>
<tr>
<td>31</td>
<td>I</td>
<td>Opto-Supply</td>
<td>Opto-Supply Input +12 to +24VDC</td>
</tr>
<tr>
<td>32</td>
<td>I</td>
<td>Opto-Ground</td>
<td>Opto-Ground</td>
</tr>
<tr>
<td>33</td>
<td>I</td>
<td>DI1</td>
<td>Digital Input 1</td>
</tr>
<tr>
<td>34</td>
<td>I</td>
<td>DI2</td>
<td>Digital Input 2</td>
</tr>
<tr>
<td>35</td>
<td>I</td>
<td>DI3</td>
<td>Digital Input 3</td>
</tr>
<tr>
<td>36</td>
<td>I</td>
<td>DI4</td>
<td>Digital Input 4</td>
</tr>
<tr>
<td>37</td>
<td>I</td>
<td>DI5</td>
<td>Digital Input 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>38</td>
<td>I</td>
<td>DI6</td>
<td>Digital Input 6</td>
</tr>
<tr>
<td>39</td>
<td>I</td>
<td>DI7</td>
<td>Digital Input 7</td>
</tr>
<tr>
<td>40</td>
<td>I</td>
<td>DI8</td>
<td>Digital Input 8</td>
</tr>
<tr>
<td>41</td>
<td>O</td>
<td>DO1</td>
<td>Digital Output 1</td>
</tr>
<tr>
<td>42</td>
<td>O</td>
<td>DO2</td>
<td>Digital Output 2</td>
</tr>
<tr>
<td>43</td>
<td>O</td>
<td>DO3</td>
<td>Digital Output 3</td>
</tr>
<tr>
<td>44</td>
<td>O</td>
<td>DO4</td>
<td>Digital Output 4</td>
</tr>
<tr>
<td>45</td>
<td>O</td>
<td>DO5</td>
<td>Digital Output 5</td>
</tr>
<tr>
<td>46</td>
<td>O</td>
<td>DO6</td>
<td>Digital Output 6</td>
</tr>
<tr>
<td>47</td>
<td>O</td>
<td>DO7</td>
<td>Digital Output 7</td>
</tr>
<tr>
<td>48</td>
<td>O</td>
<td>DO8</td>
<td>Digital Output 8</td>
</tr>
<tr>
<td>49</td>
<td>NC</td>
<td>NC</td>
<td>No Connection</td>
</tr>
<tr>
<td>50</td>
<td>NC</td>
<td>NC</td>
<td>No Connection</td>
</tr>
</tbody>
</table>

Table 4.5

Mating Connector Description: 50 pin 0.1" connector
Mating Connector Manufacturer: CW Industries
Mating Connector Manufacturer Part: C3AAG-5018M

† Other compatible connectors can be used.
Interface Circuit

Figure 4.2
**Digital Outputs**

Digital outputs are opto-isolated outputs using Darlington transistors that can sink up to 100mA current at maximum voltage of 24VDC.

![Digital Outputs Diagram](image)

**Digital Inputs**

All inputs including limits, homes, and digital inputs DI1 to DI8 are opto-isolated and require an opto-supply input of 12 to 24VDC.

To trigger the input, sink the signal to the ground of the opto-supply.

![Digital Inputs Diagram](image)
Encoder Inputs
When connecting differential connector use Encoder A, /A, B, /B, Z index, /Z index channels.

When connecting single-ended encoders, use Encoder /A, /B, and /Z index channels.

Analog Inputs
Analog inputs are 0 to 5V range and 10 bit in resolution. Using two analog inputs, joystick control can be achieved.

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

Driver Current and Microstep Setting
PMX-2ED-SA has two built-in bipolar microstep drivers.

To set the current, open up the top cover. There are two dual row jumper headers JP1 (for X axis) and JP2 (for Y axis).

There are four preset current settings: 1.2A, 1.0A, 0.7A, and 0.5A. Any of these current setting can be selected using a jumper.

For custom current setting, use the following formula to get the resistor value and solder across the custom current setting pins.

\[
\text{Resistor Value} = \left[ \frac{4167}{\text{Current (A)}} \right] + 1000
\]

For example, to set the current to 0.3A, resistor value will be 14.9K Ohm.
To set the microstep, use the SW1 (for X axis) and SW2 (for Y axis) headers.

<table>
<thead>
<tr>
<th>Microstep</th>
<th>MS1</th>
<th>MS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8th microstep</td>
<td>Open</td>
<td>Open</td>
</tr>
<tr>
<td>1/4 microstep</td>
<td>Open</td>
<td>Close</td>
</tr>
<tr>
<td>1/2 microstep</td>
<td>Close</td>
<td>Open</td>
</tr>
<tr>
<td>Full step</td>
<td>Close</td>
<td>Close</td>
</tr>
</tbody>
</table>

Table 4.6
5. Getting Started

Typical Setup
PC-Controlled

![Figure 5.0](image)

Stand-Alone Operation

![Figure 5.1](image)
Main Control Screen

Figure 5.2
A. Status

1. Current pulse position (X/Y axis).
2. Current encoder position (X/Y axis).
3. Current speed (X/Y axis).
4. Motor status (X/Y axis).
   i. IDLE - Motor is not moving.
   ii. ACCEL - Motor is accelerating.
   iii. CONST - Motor is moving at constant speed
   iv. DECEL - Motor is decelerating.
5. StepNLoop status.
   i. NA – StepNLoop is disabled.
   ii. IDLE – Motor is not moving.
   iii. MOVING – Motor is moving.
   iv. CORRECTING – Motor is attempting to correct its position.
   v. STOPPING – Motor is stopping using deceleration.
   vi. ABORTING – Motor is stopping without deceleration.
   vii. JOGGING – Motor is jogging.
   viii. HOMING – Motor is homing using the home switch.
   ix. L-HOMING -Motor is homing using the limit switch.
   x. Z-HOMING – Motor is homing using the Z-index
   xi. ERR-RANGE – The error range has been exceeded.
   xii. ERR-ATTMPT – The maximum number of attempts has been made to correct the position.
   xiii. ERR-STALL – The motor has stalled.
   xiv. ERR-LIM – A limit switch has been hit.
6. StepNLoop delta status (X/Y axis).
7. -Limit, +Limit, Home input status (X/Y axis).
8. Z encoder index channel status (X/Y axis).
9. Clear motor status and StepNLoop status button for both axes.
10. Move mode.
   i. ABS mode: On individual and interpolated move commands, motor will move to target position.
ii. INC mode: On individual and interpolated move commands, motor will increase its position by the target position amount.

B. Control

![Control Diagram]

1. Global high speed, low speed, and acceleration values are entered here (X/Y axis). To give each axis individual speed parameters, enter \texttt{HSPD[axis]}, \texttt{LSPD[axis]}, and \texttt{ACC[axis]} into the command line in the “Terminal” section.
2. Select X/Y axis. Selection of both the axes will result in synchronous movement.
3. Target position entered here (X/Y axis).
4. Enables the driver power for the indicated motor (X/Y axis).
5. \texttt{H+/H-} – Home the motor at high speed using only the home sensor.
6. \texttt{RSTOP/ISTOP} – Stop the motor with deceleration using RSTOP. Stop the motor immediately using ISTOP.
7. \texttt{RP/RE} – Reset the position/encoder position.
8. \texttt{ABS/INC} – Set the move mode to absolute or incremental.
9. \texttt{J+/J-} – Jogs the motor in the positive or negative direction.
10. \texttt{HL+/HL-} – Home the motor at high speed and low speed using only the home sensor.
11. \texttt{Z+/Z-} – Only encoder index channel used for homing.
12. \texttt{L+/L-} – Home the motor using only the limit sensor.
13. \texttt{ZH+/ZH-} – Both encoder index and home sensor used for homing.
14. \texttt{DAT/ABS} – Move the motor to position zero by using DAT. Move the motor to the target position by using ABS.
C. On-The-Fly Speed

![On-The-Fly Speed Diagram](image)

1. Select X/Y axis.
2. Select destination speed of the axis.
3. Select the acceleration used during an on-the-fly speed change.
4. Select the SSPDM mode for the axis. See On-The-Fly Speed section for details.
5. Set the SSPDM mode for the axis.
6. Set on-the-fly speed change. Acceleration will be taken from the "Accel" field.

D. Product Information

![Product Info](image)

E. Terminal

![Terminal Diagram](image)

1. Send commands to the PMX-2ED-SA through this terminal.
2. Replies from the PMX-2ED-SA will be shown here.

F. On-The-Fly Position

![On-The-Fly Position Diagram](image)

1. Select X/Y axis.
2. Set the new target position of the axis.
3. Perform an on-the-fly position change.
G. Inputs/Outputs

Figure 5.9

a. Digital input status for DI1-DI8.
b. Digital output status for DO1-DO8.
c. Analog input status for AI1 and AI2 [0-5000 mV].

H. About

Figure 5.10

Displays the current Software and Firmware versions.
I. Setup

1. Polarity:
   a. Set direction/pulse/home/Z-index polarity for X/Y axis
   b. Set s-curve enable/disable for X/Y axis
   c. Set the encoder multiplier to 1X/2X/4X for X/Y axis
   d. Limit - Set the limit input polarity
   e. DO - Set the digital output polarity
   f. EO - Set the enable output polarity
   g. DI - Set the digital input polarity
   h. SA Err - Set the return jump line for standalone error handling

2. Boot Up
   a. DO Boot/EO Boot - Set the digital and enable output configuration on boot up
   b. Auto Run - Have the specified standalone program run on boot up.

3. Homing Parameters
a. **LCA** - Set the limit correction amount for the specified axis
b. **HCA** - Set the home correction amount for the specified axis
c. **Return to Zero** - Return to zero after a homing routine.

4. **Joystick Parameters** (X/Y axis). See joystick section for further details.

5. **StepNLoop Parameters** (X/Y axis). See StepNLoop section for further details.

6. **Device Name** – Set the name of the device. Must be in the range of 2ED00 to 2ED99.

7. **Misc**
   a. **Ignore Error** - Set the IERR register to ignore the limit error status
   b. **Enable Decel** - Set the EDEC register to enable unique deceleration

8. **STORE** - Permanently store the downloaded parameters to flash memory.

9. **Download/Upload** - Download the current setting to the unit or upload the current setting from the unit.

10. **Open/Save** parameters to file.

### J. Variables

Figure 5.12

1. Current values of variables that cannot be stored to flash.
2. Current values of variables that can be stored to flash.
c. Send commands to the PMX-2ED-SA through this terminal.

K. Program File Control

![Figure 5.13](image)

- **Open** - Open a standalone program
- **Save** - Save a standalone program
- **New** - Clear the standalone program editor

L. Text Programming Box

![Figure 5.14](image)

- **Text Program** – Text box for writing and editing a standalone program.
- **Clear Code Space** – Clear the code space on the PMX-2ED-SA.
M. Compiler

Figure 5.15

a. **Compile** - Compile code in text programming box into assembly level code that the PMX-2ED-SA can understand.
b. **Download** - Download the compiled code into memory. Note that the text based code must be compiled before download.
c. **Upload** - Upload standalone code that is currently on your PMX-2ED-SA. This automatically translates assembly level language to readable text-based code.
d. **View** - View compiled code for easy cutting and pasting.

N. Program Control

Figure 5.16

1. **Run** – Standalone program is run.
2. **Stop** – Program is stopped.
3. **Pause** – Program that is running can be stopped.
4. **Cont** – Program that is paused can be continued
5. **XThread** - Open the Standalone Program Control for all standalone programs.

6. **Index** - Current line of code that is being executed.

7. **Status of standalone program:**
   - i. **Idle** – Program is not running.
   - ii. **Running** – Program is running.
   - iii. **Paused** – Program is paused.
   - iv. **Error** – Program is in an error state.
6. Motion Control Feature Overview

**Important Note:** All the commands described in this section are interactive commands and are not analogous to stand-alone commands. Refer to the “Standalone Language Specification” section for details regarding stand-alone commands.

**Motion Profile**

By default, the PMX-2ED-SA uses trapezoidal velocity profile. See Figure 6.0.

![Figure 6.0](image)

S-curve velocity profile can also be achieved by using the `SCV[axis]` command. See Figure 6.1

![Figure 6.1](image)

High speed and low speed are in pps (pulses/second). Use `HSPD[axis]` and `LSPD[axis]` to set/get individual high speed and low speed settings. To set/get the global high speed and low speed values use the `HSPD` and `LSPD` commands.

Acceleration and deceleration time are in milliseconds and are symmetrical. Use the `ACC[axis]/DEC[axis]` command to set/get individual acceleration/deceleration values. To set/get the global acceleration value, use the `ACC/DEC` command.

**Notes:**

By default, moves by a single axis use global speed settings, unless individual high speed, low speed and acceleration values for that axis are non-zero.
Example: To set the high-speed of the X-axis to 1500 pulses/second, and the Y-axis to 2000 pulses/second, issue the following speed setting commands:

```
HSX=1500  ' set high speed for x-axis only
HSY=2000  ' set high speed for y-axis only
LSX=300   ' other parameters for the axis MUST be set as well for
LSY=300   ' the controller to use the individual speed settings instead
ACCX=100  ' of the global speed settings
ACCY=100
```

It is possible to have unique acceleration and deceleration times. In order to decelerate using the value set in the DEC[axis] or DEC parameter, set EDEC to 1.

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

**Pulse Speed**

Current pulse rate can be read using the PSX/PSY command. For units, see Table 6.0

<table>
<thead>
<tr>
<th>Operation Mode</th>
<th>Speed Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>StepNLoop disabled</td>
<td>Pulse / sec</td>
</tr>
<tr>
<td>ALL interpolated moves</td>
<td>Pulse / sec</td>
</tr>
<tr>
<td>StepNLoop enabled and non-interpolated move</td>
<td>Encoder counts / sec</td>
</tr>
</tbody>
</table>

Table 6.0

**On-The-Fly Speed Change**

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

**SSPD Mode**

The correct speed window must be selected in order to use the SSPD command. To select a speed window, use the SSPDM[axis] command. Refer to Appendix A for details.

During on-the-fly speed change operation, you must keep the initial and destination speeds within the speed window.

For non on-the-fly speed change moves, set SSPDM[axis] to 0.
Motor Status
Motor status can be read anytime using MSTX/MSTY command. Following are bit representation of motor status:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Motor in acceleration</td>
</tr>
<tr>
<td>1</td>
<td>Motor in deceleration</td>
</tr>
<tr>
<td>2</td>
<td>Motor running at constant speed</td>
</tr>
<tr>
<td>3</td>
<td>Not Used</td>
</tr>
<tr>
<td>4</td>
<td>Plus limit input switch status</td>
</tr>
<tr>
<td>5</td>
<td>Minus limit input switch status</td>
</tr>
<tr>
<td>6</td>
<td>Home input switch status</td>
</tr>
<tr>
<td>7</td>
<td>Plus limit error. This bit is latched when plus limit is hit during motion.</td>
</tr>
<tr>
<td></td>
<td>This error must be cleared (using CLR/CLRX/CLRY command) before issuing any</td>
</tr>
<tr>
<td></td>
<td>subsequent move commands.</td>
</tr>
<tr>
<td>8</td>
<td>Minus limit error. This bit is latched when minus limit is hit during motion.</td>
</tr>
<tr>
<td></td>
<td>This error must be cleared (using CLR/CLRX/CLRY command) before issuing any</td>
</tr>
<tr>
<td></td>
<td>subsequent move commands.</td>
</tr>
<tr>
<td>9</td>
<td>Z Index Channel status</td>
</tr>
<tr>
<td>10</td>
<td>Joystick Control On status</td>
</tr>
<tr>
<td>11</td>
<td>TOC time-out status</td>
</tr>
</tbody>
</table>

Table 6.1

Individual/Linear Interpolation Moves
For individual axis control use the X and Y commands followed by the target position value.

Individual Move Examples:
[X1000]: Move X-axis to position 1000.

[Y1000]: Move Y-axis to position 1000.

For linear interpolation axis control use the [X Target];[Y Target] to perform coordinated movement to the specified target positions.
Linear Interpolation Move Examples:

\([I1000:1000]\): Move X-axis to position 1000, Y-axis to position 1000 using linear interpolation.

\([I10000:-10000]\): Move X-axis to position 10000, Y-axis to position -10000 using linear interpolation.

Individual/Linear Interpolation moves can be performed in two modes: incremental mode. To set move modes, use the INC and ABS commands respectively.

Move Mode Examples:

\([X1000] – \text{INC mode}\): The motor will move by 1000 from the current position.

\([X1000] – \text{ABS mode}\): The motor will move to absolute position 1000.

**On-The-Fly Target Position Change**

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

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

**Homing**

Home search sequence involves moving the motor towards the home or limit switches and then stopping when the relevant input is detected. The PMX-2ED-SA has five different homing routines.

**Home Input Only (High speed only)**

Use the \(H[axis]+/H[axis]-\) command. (use the \(H+/H-\) command for both axes). Figure 6.2 shows the homing routine.

A. Starts the motor from low speed and accelerates to high speed.
B. As soon as the home input is triggered, the position counter is reset to zero and the motor begins to decelerate to low speed. As the motor decelerates, the position counter keeps counting with reference to the zero position.
C. Once low speed is reached, the motor stops. The position is non-zero.

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

**Home Input Only (High speed and low speed)**
Use the **HL[axis]+/HL[axis]-** command (use the **HL+/HL-** for both axes). Figure 6.3 shows the homing routine.

![Diagram of homing routine](Image)

**Figure 6.3**

A. Starts the motor from low speed and accelerates to high speed.
B. As soon as the home input is triggered, the position counter is reset to zero and the motor decelerates to low speed.
C. Once low speed is reached, the motor reverses direction to search for the home switch.
D. Once the home switch is reached, it will continue past the home switch by the amount defined by the home correction amount (**HCA**) at high speed.
E. The motor is now past the home input by the amount defined by the home correction amount (**HCA**). The motor now moves back towards the home switch at low speed.
F. The home input is triggered again, the position counter is reset to zero and the motor stops immediately.
**Limit Only**
Use the `L[axis]+/L[axis]-` command (use the `L+/L-` command for both axes). Figure 6.4 shows the homing routine.

![Original Direction](image1)
![Opposite Direction](image2)

**Figure 6.4**

A. Issuing a limit home command starts the motor from low speed and accelerates to high speed.
B. The corresponding limit is triggered and the motor stops immediately.
C. The motor reverses direction by the amount defined by the limit correction amount (LCA) at high speed.
D. The zero position is reached.

**Home and Z-index**
Use the `HZ[axis]+/HZ[axis]-` command (use the `HZ+/HZ-` command for both axes). Figure 6.5 shows the homing routine.

![Home Input](image3)
![Z-index](image4)

**Figure 6.5**

A. Issuing the command starts the motor from low speed and accelerates to high speed.
B. As soon as the home input is triggered, the motor decelerates to low speed.
C. Once low speed is reached, the motor begins to search for the z-index pulse.
D. Once the z-index pulse is found, the motor stops and the position is set to zero.
**Z-index only**
Use the $Z[\text{axis}]+/Z[\text{axis}]$- command (use the $Z+/Z-$ command for both axes). Figure 6.6 shows the homing routine.

![Z-index Diagram](image)

Figure 6.6

A. Issuing the home command starts the motor at low speed.
B. Once the z-index pulse is found, the motor stops and the position is set to zero.

**Jogging**
Jogging is available for continuous speed operation. Use $JX+/JX-/JY+/JY-$ command. To have both motors jog synchronously use the $J+/J-$ command.

**Stopping Motor**
When the motor is moving, the $\text{ABORT}[\text{axis}]$ command will immediately stop an individual axis. Use the $\text{ABORT}$ command to immediately stop ALL axes.

To employ deceleration on a stop, use the $\text{STOP}[\text{axis}]$ to stop an individual axis. Use the $\text{STOP}$ command to stop ALL axes.

**Note:** If an interpolation operation is in process when a $\text{STOP}[\text{axis}]$ or $\text{ABORT}[\text{axis}]$ command is entered, all axes involved in the interpolation operation will stop.

**Motor Position**
Motor positions can be read using the $\text{PX/Py}$ command which returns the pulse position of the specified axis.

Encoder positions can be read using $\text{EX/EY}$ command which returns the encoder position of the specified axis.

To manually set/get the pulse position of an individual axis, use the $\text{PX/Py}$ command. Note that setting the pulse position is not allowed if StepNLoop is enabled.

To manually set/get the encoder position of an individual axis, use the $\text{EX/EY}$ command.
Polarity

The polarity settings of the PMX-2ED-SA can also be read or set at anytime using the \textbf{POLX/POLY} commands. The following is the bit representation of the polarity:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>1</td>
<td>Direction</td>
</tr>
<tr>
<td>2</td>
<td>Not Used</td>
</tr>
<tr>
<td>3</td>
<td>Not Used</td>
</tr>
<tr>
<td>4</td>
<td>Not Used</td>
</tr>
<tr>
<td>5</td>
<td>Home</td>
</tr>
<tr>
<td>6</td>
<td>+/- Limit</td>
</tr>
<tr>
<td>7</td>
<td>Z-Index</td>
</tr>
<tr>
<td>8,9</td>
<td>Encoder decoding</td>
</tr>
<tr>
<td></td>
<td>00 1X</td>
</tr>
<tr>
<td></td>
<td>01 2X</td>
</tr>
<tr>
<td></td>
<td>10 4X</td>
</tr>
<tr>
<td>10</td>
<td>Digital Input</td>
</tr>
<tr>
<td>11</td>
<td>Digital Output</td>
</tr>
<tr>
<td>12</td>
<td>Enable Output</td>
</tr>
<tr>
<td>13</td>
<td>Jump to Line 0 on error†</td>
</tr>
</tbody>
</table>

Table 6.2

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

Limits

If positive limit switch is triggered while moving in positive direction, the motor will immediately stop and the motor status bit for positive limit error is set. The same is for the negative limit while moving in the negative direction. To read the limit switch input status, use the \textbf{MSTX/MSTY} command.

Once the limit error is set, use the \textbf{CLR[axis]} command to clear the error.

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

Digital Inputs/Outputs and Enable Outputs

PMX-2ED-SA module comes with 8 digital inputs and 8 digital outputs and 4 enable outputs.

Inputs
Read digital input status using the \textbf{DI} command.
Digital input values can also be referenced one bit at a time by the **DI[1-8]** commands. Note that the indexes are 1-based for the bit references (i.e. DI1 refers to bit 0, not bit 1)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Bit-Wise Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Digital Input 1</td>
<td>DI1</td>
</tr>
<tr>
<td>1</td>
<td>Digital Input 2</td>
<td>DI2</td>
</tr>
<tr>
<td>2</td>
<td>Digital Input 3</td>
<td>DI3</td>
</tr>
<tr>
<td>3</td>
<td>Digital Input 4</td>
<td>DI4</td>
</tr>
<tr>
<td>4</td>
<td>Digital Input 5</td>
<td>DI5</td>
</tr>
<tr>
<td>5</td>
<td>Digital Input 6</td>
<td>DI6</td>
</tr>
<tr>
<td>6</td>
<td>Digital Input 7</td>
<td>DI7</td>
</tr>
<tr>
<td>7</td>
<td>Digital Input 8</td>
<td>DI8</td>
</tr>
</tbody>
</table>

Table 6.3

Digital Outputs
The digital output status can be controlled using the **DO** command. DO value must be within the range of 0-255.

Digital output values can also be referenced one bit at a time by the **DO[1-8]** commands. Note that the indexes are 1-based for the bit references (i.e. DO1 refers to bit 0, not bit 1)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Bit-Wise Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Digital Output 1</td>
<td>DO1</td>
</tr>
<tr>
<td>1</td>
<td>Digital Output 2</td>
<td>DO2</td>
</tr>
<tr>
<td>2</td>
<td>Digital Output 3</td>
<td>DO3</td>
</tr>
<tr>
<td>3</td>
<td>Digital Output 4</td>
<td>DO4</td>
</tr>
<tr>
<td>4</td>
<td>Digital Output 5</td>
<td>DO5</td>
</tr>
<tr>
<td>5</td>
<td>Digital Output 6</td>
<td>DO6</td>
</tr>
<tr>
<td>6</td>
<td>Digital Output 7</td>
<td>DO7</td>
</tr>
<tr>
<td>7</td>
<td>Digital Output 8</td>
<td>DO8</td>
</tr>
</tbody>
</table>

Table 6.4

The initial state of the digital outputs can be defined by setting the **DOBOOT** register to the desired initial digital output value. The value is stored to flash memory once the **STORE** command is issued.

Enable Outputs
The enable output status can be controlled using the **EO** command. EO value must be within the range of 0-3.
Enable output values can also be referenced one bit at a time by the **EO[1-2]** commands. Note that the indexes are 1-based for the bit references (i.e. EO1 refers to bit 0, not bit 1).

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Bit-Wise Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Enable Output 1</td>
<td>EO1</td>
</tr>
<tr>
<td>1</td>
<td>Enable Output 2</td>
<td>EO2</td>
</tr>
</tbody>
</table>

Table 6.5

The initial state of the enable outputs can be defined by setting the **EOBOOT** register to the desired initial enable output value. The value is stored to flash memory once the **STORE** command is issued.

**Analog Inputs**

2 x 10-bit analog inputs are available on PMX-2ED-SA. Use **AI[1-2]** command to read the analog input value. Range is from 0-5000 mV.

**Joystick Control**

Joystick control is available on PMX-2ED-SA. When this mode is enabled, the pulse speed and direction output can be controlled by the corresponding analog input. See the axis to analog input relationship in the table below:

<table>
<thead>
<tr>
<th>Axis</th>
<th>Analog Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>AI1</td>
</tr>
<tr>
<td>Y</td>
<td>AI2</td>
</tr>
</tbody>
</table>

Table 6.6

Maximum joystick speed is set using the **JV1** and **JV2** variables.

Maximum speed change (delta) is set using the **JV3** and **JV4** variables.

Tolerance of the zero joystick position, use **JV5** and **JV6** variables.

Joystick control also has soft limit controls. Limits are broken into: negative outer limit, negative inner limit, positive inner limit and positive outer limit.

When moving in positive direction, as soon as the positive inner limit is crossed, the speed is reduced. If the position reaches the positive outer limit, the joystick speed is set to zero. Same goes for the negative direction and negative limits.

The behavior of the limits of the joystick control is explained by the following:
Summary of joystick control parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JV1</td>
<td>X-axis maximum joystick speed at 5000 mV and 0 mV</td>
</tr>
<tr>
<td>JV2</td>
<td>Y-axis maximum joystick speed at 5000 mV and 0 mV</td>
</tr>
<tr>
<td>JV3</td>
<td>X-axis maximum speed change</td>
</tr>
<tr>
<td>JV4</td>
<td>Y-axis maximum speed change</td>
</tr>
<tr>
<td>JV5</td>
<td>X-axis zero tolerance range for analog input</td>
</tr>
<tr>
<td>JV6</td>
<td>Y-axis zero tolerance range for analog input</td>
</tr>
<tr>
<td>JL1</td>
<td>X-axis negative outer limit</td>
</tr>
<tr>
<td>JL2</td>
<td>X-axis negative inner limit</td>
</tr>
<tr>
<td>JL3</td>
<td>X-axis positive inner limit</td>
</tr>
<tr>
<td>JL4</td>
<td>X-axis positive outer limit</td>
</tr>
<tr>
<td>JL5</td>
<td>Y-axis negative outer limit</td>
</tr>
<tr>
<td>JL6</td>
<td>Y-axis negative inner limit</td>
</tr>
<tr>
<td>JL7</td>
<td>Y-axis positive inner limit</td>
</tr>
<tr>
<td>JL8</td>
<td>Y-axis positive outer limit</td>
</tr>
</tbody>
</table>

Table 6.7

To enable/disable joystick control for an axis, use the JE command. Joystick enable parameter is a 2 bit value. For example, joystick enable value of 3 means joystick feature is enabled on both axes.

Note: If joystick control is enabled, StepNLoop is automatically disabled.

StepNLoop Closed Loop Control

PMX-2ED-SA features a closed-loop position verification algorithm called StepNLoop (SNL). The algorithm requires the use of an incremental encoder.

SNL performs the following operations:

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

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

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

Table 6.8

†A convenient way to find the StepNLoop ratio is to set EX=0, PX=0 and move the motor +1000 pulses. The ratio can be calculated by dividing 1000 by the resulting EX value. Note that the value must be positive. If it is not, then the direction polarity must be adjusted. This test can be performed on all axes that require StepNLoop.

To enable/disable the SNL feature use the **SL[axis]** command. To read the SNL status, use **SLS[axis]** command to read the status.

See Table 6.9 for a list of the **SLS[axis]** return values.

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Idle</td>
</tr>
<tr>
<td>1</td>
<td>Moving</td>
</tr>
<tr>
<td>2</td>
<td>Correcting</td>
</tr>
<tr>
<td>3</td>
<td>Stopping</td>
</tr>
<tr>
<td>4</td>
<td>Aborting</td>
</tr>
<tr>
<td>5</td>
<td>Jogging</td>
</tr>
<tr>
<td>6</td>
<td>Homing</td>
</tr>
<tr>
<td>7</td>
<td>Z-Homing</td>
</tr>
<tr>
<td>8</td>
<td>Correction range error. To clear this</td>
</tr>
</tbody>
</table>
9 Correction attempt error. To clear this error, use \textit{CLRS} or \textit{CLR} command.

10 Stall Error. \textit{DX} value has exceeded the correction range value. To clear this error, use \textit{CLRS} or \textit{CLR} command.

11 Limit Error

12 N/A (i.e. SNL is not enabled)

13 Limit homing

Table 6.9

See Table 6.10 for SNL behavior within different scenarios.

<table>
<thead>
<tr>
<th>Condition</th>
<th>SNL behavior (motor is moving)</th>
<th>SNL behavior (motor is idle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta \leq \text{SLT}$</td>
<td>Continue to monitor the $\text{DX}[$axis$]$</td>
<td>In Position. No correction is performed.</td>
</tr>
<tr>
<td>$\delta &gt; \text{SLT}$ AND $\delta &lt; \text{SLE}$</td>
<td>Continue to monitor the $\text{DX}[$axis$]$</td>
<td>Out of Position. A correction is performed.</td>
</tr>
<tr>
<td>$\delta &gt; \text{SLT}$ AND $\delta &gt; \text{SLE}$</td>
<td>Stall Error. Motor stops and signals and error.</td>
<td>Error Range Error. Motor stops and signals and error.</td>
</tr>
<tr>
<td>Correction Attempt $&gt; \text{SLA}$</td>
<td>NA</td>
<td>Max Attempt Error. Motor stops and signals and error.</td>
</tr>
</tbody>
</table>

Table 6.10

Key
- $\delta$: Error between the target position and actual position
- \text{SLT}: Tolerance range
- \text{SLE}: Error range
- \text{SLA}: Max correction attempt

Notes:
Once SNL is enabled, position move commands are in term of encoder position. For example, X1000 means to move the motor to encoder 1000 position. This applies to individual as well as interpolated moves.

Once SNL is enabled, the speed is in encoder speed. For example HSPD=1000 when SNL is enabled means that the target high speed is 1000 encoder counts per second. This only applies to individual axis moves.

\textbf{Linear Interpolation w/ StepNLoop:} If StepNLoop is used during a linear interpolation move, StepNLoop must be enabled for all axes being moved. Also note that unlike the
individual axis moves, the speed during a linear interpolation is calculated as pulse/sec, NOT encoder counts/sec.

**Device Number**
Performax 2ED-SA 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. Note that this value must be within the range [2ED00-2ED99].

To write the values to the device’s flash memory, use the STORE command. After a complete power cycle, the new device ID 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: **2ED00**

**Standalone Program Specification**
Standalone Program Specification:
Memory size: 1,275 assembly lines.
Note: Each line of pre-compiled code equates to 1-4 lines of assembly lines.

**WAIT Statement:** When writing a standalone program, it is generally necessary to wait until a motion is completed before moving on to the next line. In order to do this, the WAIT statement must be used. See the examples below:

In the example below, the variable V1 will be set immediately after the X10000 move command begins; it will not wait until the controller is idle.

```
X10000  ;* Move to position 0
V1=100
```

Conversely, in the example below, the variable V1 will not be set until the motion has been completed. V1 will only be set once the controller is idle.

```
X10000  ;* Move to position 0
WAITX  ;* Wait for the move to complete
V1=100
```

**Multi-Threading:** PMX-2ED-SA supports the simultaneous execution of up to 2 standalone programs. Programs 0,1 are controlled via the SR0 and SR1 commands respectively. For examples of multi-threading, please refer to the Example Stand-alone Programs section.

**Note:** Sub-routines can be shared by different threads.

**Error Handling:** If an error occurs during standalone execution (i.e. limit error), 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 will be executed. The return jump line will be determined by value of the 13th bit of the POL register. If the value is 0, the return jump line will be the line that caused the error. Otherwise, the return jump line will be line 0.

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

**Standalone Run on Boot-Up:** Standalone can be configured to run on boot-up using the SLOAD command. See description below:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Standalone Program 0</td>
</tr>
<tr>
<td>1</td>
<td>Standalone Program 1</td>
</tr>
</tbody>
</table>

Table 6.11

**Storing to Flash**

The following items are stored to flash:

<table>
<thead>
<tr>
<th>ASCII Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DN</td>
<td>Device name</td>
</tr>
<tr>
<td>DB</td>
<td>Baud Rate</td>
</tr>
<tr>
<td>DOBOOT</td>
<td>DO configuration at boot-up</td>
</tr>
<tr>
<td>EDEC</td>
<td>Unique deceleration enable</td>
</tr>
<tr>
<td>EOBOOT</td>
<td>EO configuration at boot-up</td>
</tr>
<tr>
<td>IERR</td>
<td>Ignore limit error enable</td>
</tr>
<tr>
<td>HCA, HCA[axis]</td>
<td>Home Correction Amount</td>
</tr>
<tr>
<td>LCA, LCA[axis]</td>
<td>Limit Correction Amount</td>
</tr>
<tr>
<td>POL[axis]</td>
<td>Polarity settings</td>
</tr>
<tr>
<td>SCV[axis]</td>
<td>S-curve enable</td>
</tr>
<tr>
<td>SL[axis], SLR[axis], SLE[axis], SLT[axis], SLA[axis]</td>
<td>StepNLoop parameters</td>
</tr>
<tr>
<td>JO, JF, JV[1-6], JL[1-8]</td>
<td>Joystick settings</td>
</tr>
<tr>
<td>RZ</td>
<td>Return to zero (homing)</td>
</tr>
<tr>
<td>SLOAD</td>
<td>Standalone program run on boot-up parameter</td>
</tr>
<tr>
<td>TOC</td>
<td>Time-out counter reset value</td>
</tr>
<tr>
<td>V32-V63</td>
<td>Note that on boot-up, V0-V31 are reset to value 0</td>
</tr>
</tbody>
</table>

Table 6.12

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

PMX-2ED-SA USB communication is USB 2.0 compliant.

Communication between the PC and PMX-2ED-SA 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 PMX-2ED-SA for sending a command from a PC and getting a reply from PMX-2ED-SA using the `fnPerformaxComSendRecv()` API function is in single digit milliseconds. This value will vary with CPU speed of PC and the type of command.

**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, LPVOID wBuffer, DWORD dwNumBytesToWrite, DWORD dwNumBytesReceived, DWORD dwError);**
  - This function is used to send command to Performax USB module and get reply.

- **BOOL fnPerformaxComSend(IN HANDLE pHandle, LPVOID wBuffer, DWORD dwNumBytesToWrite, DWORD dwNumBytesSent, DWORD dwError);**
  - This function is used to send command to Performax USB module.

- **BOOL fnPerformaxComReceive(IN HANDLE pHandle, LPVOID wBuffer, DWORD dwNumBytesToWrite, DWORD dwNumBytesReceived, DWORD dwError);**
  - This function is used to receive reply from Performax USB module.
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.

BOOL fnPerformaxComFlush(IN HANDLE pHandle)
- Flushes the communication buffer on the PC as well as the USB controller. It 
is recommended to perform this operation right after the communication 
handle is opened.

**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) **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

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

![USB Cable with Noise Suppression Choke](image)

Figure 7.0
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte Size</td>
<td>8 bits</td>
</tr>
<tr>
<td>Parity</td>
<td>None</td>
</tr>
<tr>
<td>Flow Control</td>
<td>None</td>
</tr>
<tr>
<td>Stop Bit</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8.0

**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: `@00POLX[CR]`

Reply: `7[CR]`

**For jogging the x-motor in positive direction**

Send: `@00JX+[CR]`

Reply: `OK[CR]`

**For aborting any motion in progress**

Send: `@00ABORT[CR]`

Reply: `OK[CR]`
9. ASCII Language Specification

Invalid command is returned with ?(Error Message). Always check for proper reply when command is sent. Like the commands, all responses are in ASCII form.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT</td>
<td>Aborts all axis moves</td>
<td>OK</td>
</tr>
<tr>
<td>ABORTX</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>ABORTY</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>ACC</td>
<td>Returns the global acceleration value.</td>
<td>32-bit number</td>
</tr>
<tr>
<td>ACC=[Value]</td>
<td>Sets the global acceleration setting.</td>
<td>OK</td>
</tr>
<tr>
<td>ACCX</td>
<td>Returns acceleration setting for the X-axis and Y-axis</td>
<td>32-bit number</td>
</tr>
<tr>
<td>ACCY</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>ACCX=[Value]</td>
<td>Sets acceleration setting for the X-axis and Y-axis</td>
<td>OK</td>
</tr>
<tr>
<td>ACCY=[Value]</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>AI1</td>
<td>Returns Analog Input in millivolt</td>
<td>[0-5000]</td>
</tr>
<tr>
<td>AI2</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>CLRX</td>
<td>Clears motor limit status bit. Also clears a StepNLoop errors OK</td>
<td></td>
</tr>
<tr>
<td>CLRY</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>DB</td>
<td>Return baud rate</td>
<td>[1,2,3,4,5]</td>
</tr>
<tr>
<td>DB=[value]</td>
<td>Set baud rate</td>
<td>OK</td>
</tr>
<tr>
<td>1 - 9600 bps</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>2 - 19200 bps</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>3 - 38400 bps</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>4 - 57600 bps</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>5 - 115200 bps</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>DEC</td>
<td>Returns the current global deceleration value in milliseconds</td>
<td>32-bit number</td>
</tr>
<tr>
<td>DEC=[Value]</td>
<td>Sets the global deceleration value in milliseconds</td>
<td>OK</td>
</tr>
<tr>
<td>DECX</td>
<td>Returns the current individual deceleration value in milliseconds</td>
<td>32-bit number</td>
</tr>
<tr>
<td>DECY</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>DECX=[value]</td>
<td>Sets the individual deceleration value in milliseconds</td>
<td>OK</td>
</tr>
<tr>
<td>DECY=[value]</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>DI</td>
<td>Returns Digital Input Value</td>
<td>[0-255]</td>
</tr>
<tr>
<td>DI[1-8]</td>
<td>Return individual input bit status</td>
<td>[0,1]</td>
</tr>
<tr>
<td>DO</td>
<td>Returns Digital Output Value</td>
<td>[0-255]</td>
</tr>
<tr>
<td>DO[1-8]</td>
<td>Returns the individual output bit status.</td>
<td>[0,1]</td>
</tr>
<tr>
<td>DO=[value]</td>
<td>Sets Digital Output Value</td>
<td>OK</td>
</tr>
<tr>
<td>DO[1-8]=[value]</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>DOBOOT</td>
<td>Get DO boot-up state</td>
<td>See Table 6.4</td>
</tr>
<tr>
<td>DOBOOT=[Value]</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>DN</td>
<td>Return Device Number</td>
<td>2EDXX</td>
</tr>
<tr>
<td>DN=[Value]</td>
<td>Set Device Number</td>
<td>OK</td>
</tr>
<tr>
<td>DXX</td>
<td>Returns delta value for X-axis and Y-axis</td>
<td>32-bit number</td>
</tr>
<tr>
<td>DXY</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>EDEC</td>
<td>Returns the enable deceleration status</td>
<td>[0,1]</td>
</tr>
<tr>
<td>EDEC=[0 or 1]</td>
<td>Sets the enabled deceleration status</td>
<td>OK</td>
</tr>
<tr>
<td>EO</td>
<td>Returns 2 bits of enable output value.</td>
<td>[0,3]</td>
</tr>
<tr>
<td>EO=[value]</td>
<td>Sets 2 bits of enable outputs.</td>
<td>OK</td>
</tr>
<tr>
<td>EO1</td>
<td>Returns the specified bit of the enable output status</td>
<td>[0,1]</td>
</tr>
<tr>
<td>EO2</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>EO1=[0 or 1]</td>
<td>Sets the specified bit of the enable output status</td>
<td>OK</td>
</tr>
<tr>
<td>EO2=[0 or 1]</td>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Source/Notes</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>EOBOOT</td>
<td>Get EO boot-up state</td>
<td>See Table 6.5</td>
</tr>
<tr>
<td>EOBOOT=[Value]</td>
<td>Set EO boot-up state</td>
<td>OK</td>
</tr>
<tr>
<td>EX</td>
<td>Returns Current Encoder Position</td>
<td>28 bit signed position</td>
</tr>
<tr>
<td>EY</td>
<td>Sets Current Encoder Position</td>
<td>OK</td>
</tr>
<tr>
<td>EX=[Value]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EY=[Value]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GS[SubNumber]</td>
<td>Call a defined subroutine</td>
<td>OK</td>
</tr>
<tr>
<td>HCA</td>
<td>Returns the global home correction amount</td>
<td>28-bit number</td>
</tr>
<tr>
<td>HCA=[Value]</td>
<td>Sets the global home correction amount.</td>
<td>OK</td>
</tr>
<tr>
<td>HCAX</td>
<td>Returns the home correction amount for the specified axis.</td>
<td>28-bit number</td>
</tr>
<tr>
<td>HCAY</td>
<td>Sets the home correction amount for the specified axis.</td>
<td>OK</td>
</tr>
<tr>
<td>HCAX=[value]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCAY=[value]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSPD</td>
<td>Returns the global high speed setting.</td>
<td>High Speed</td>
</tr>
<tr>
<td>HSPD=[Value]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSPDAX</td>
<td>Returns high setting for the X-axis Y-axis</td>
<td>High Speed</td>
</tr>
<tr>
<td>HSPD=[Value]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSPDAX=[Value]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H-</td>
<td>Homes both X and Y axis at high speed in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>H+</td>
<td>Homes both X and Y axis at high speed in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>HX+</td>
<td>Homes X/Y axis at high speed in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>HY+</td>
<td>Homes X/Y axis at high speed in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>HX-</td>
<td>Homes X/Y axis at high speed in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>HY-</td>
<td>Homes X/Y axis at high speed in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>HL+</td>
<td>Homes both X and Y axis at high and low speed in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>HL-</td>
<td>Homes both X and Y axis at high and low speed in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>HLX+</td>
<td>Homes X/Y axis at high and low speed in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>HLX-</td>
<td>Homes X/Y axis at high and low speed in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>I[X Target]; [Y Target]</td>
<td>Perform linear interpolated motion</td>
<td>OK</td>
</tr>
<tr>
<td>ID</td>
<td>Returns Controller ID</td>
<td>Performax-2ED-SA</td>
</tr>
<tr>
<td>IERR</td>
<td>Get the ignore limit error status</td>
<td>[0-1]</td>
</tr>
<tr>
<td>IERR=[0 or 1]</td>
<td></td>
<td>Set the ignore limit error status</td>
</tr>
<tr>
<td>INC</td>
<td>Turns on incremental move mode.</td>
<td>OK</td>
</tr>
<tr>
<td>JF</td>
<td>Turns off Joystick Control</td>
<td>OK</td>
</tr>
<tr>
<td>JL[1 to 8]</td>
<td>Return Joystick Control Limits</td>
<td>See Table 6.7</td>
</tr>
<tr>
<td>JL[1 to 8]=[Value]</td>
<td></td>
<td>Sets Joystick Control Limits. See Table 6.7</td>
</tr>
<tr>
<td>JO</td>
<td>Turns on Joystick Control</td>
<td>OK</td>
</tr>
<tr>
<td>JS</td>
<td>Get the Joystick status</td>
<td>[0,1]</td>
</tr>
<tr>
<td>JV[1 to 6]</td>
<td>Returns Joy Stick Control Parameters</td>
<td>See Table 6.7</td>
</tr>
<tr>
<td>JV[1 to 6]=[Value]</td>
<td></td>
<td>Sets Joystick Control Parameters. See Table 6.7</td>
</tr>
<tr>
<td>J+</td>
<td>Jogs both X/Y Motor Positive</td>
<td>OK</td>
</tr>
<tr>
<td>J-</td>
<td>Jogs both X/Y Motor Negative</td>
<td>OK</td>
</tr>
<tr>
<td>JX+</td>
<td>Jogs Motor Positive</td>
<td>OK</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Status</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>JY+</td>
<td>Jogs Motor Negative</td>
<td>OK</td>
</tr>
<tr>
<td>JX-</td>
<td>Homes both X and Y axis to the positive limit input in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>JY-</td>
<td>Home both X and Y axis to the negative limit input in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>LX+</td>
<td>Homes X/Y axis to the positive limit input in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>LY+</td>
<td>Homes X/Y axis to the negative limit input in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>LCA</td>
<td>Returns the global limit correction amount</td>
<td>28-bit number</td>
</tr>
<tr>
<td>LCAX</td>
<td>Sets the global limit correction amount</td>
<td>OK</td>
</tr>
<tr>
<td>LCAY</td>
<td>Returns the specified limit correction amount</td>
<td>28-bit number</td>
</tr>
<tr>
<td>LSPD</td>
<td>Returns the global low speed setting</td>
<td>32-bit number</td>
</tr>
<tr>
<td>LSPDX</td>
<td>Sets the global low speed setting.</td>
<td>OK</td>
</tr>
<tr>
<td>LSPDY</td>
<td>Returns low speed setting for the X-axis and Y-axis</td>
<td>32-bit number</td>
</tr>
<tr>
<td>LSPDX=[Value]</td>
<td>Sets low speed setting for the X-axis and Y-axis</td>
<td>OK</td>
</tr>
<tr>
<td>LSPDY=[Value]</td>
<td>Sets low speed setting for the X-axis and Y-axis</td>
<td>OK</td>
</tr>
<tr>
<td>MM</td>
<td>Returns the move mode that the controller is currently in.</td>
<td>0 – ABS mode 1 – INC mode</td>
</tr>
<tr>
<td>MSTX</td>
<td>Returns motor status</td>
<td>See Table 6.1</td>
</tr>
<tr>
<td>Msty</td>
<td>Returns motor status</td>
<td>See Table 6.1</td>
</tr>
<tr>
<td>POLX</td>
<td>Returns polarity</td>
<td>See Table 6.2</td>
</tr>
<tr>
<td>POLY</td>
<td>Returns polarity</td>
<td>See Table 6.2</td>
</tr>
<tr>
<td>POLX=[Value]</td>
<td>Sets polarity. See Table 6.2</td>
<td>OK</td>
</tr>
<tr>
<td>POLY=[Value]</td>
<td>Sets polarity. See Table 6.2</td>
<td>OK</td>
</tr>
<tr>
<td>PSX</td>
<td>Returns current pulse speed</td>
<td>28-bit number</td>
</tr>
<tr>
<td>PSY</td>
<td>Returns current pulse position</td>
<td>28-bit number</td>
</tr>
<tr>
<td>PX=[Value]</td>
<td>Sets current pulse position</td>
<td>OK</td>
</tr>
<tr>
<td>PY=[Value]</td>
<td>Sets current pulse position</td>
<td>OK</td>
</tr>
<tr>
<td>RZ</td>
<td>Returns the return to zero enable status. Used during homing operations</td>
<td>[0,1]</td>
</tr>
<tr>
<td>RZ=[0,1]</td>
<td>Sets the return to zero enable status. Used during homing operations</td>
<td>OK</td>
</tr>
<tr>
<td>SASTAT[0,1]</td>
<td>Get standalone program status</td>
<td>[0-4]</td>
</tr>
<tr>
<td>SA[LineNumber]</td>
<td>Get standalone line LineNumber: [0,1275]</td>
<td>OK</td>
</tr>
<tr>
<td>SA[LineNumber]=[Value]</td>
<td>Set standalone line LineNumber: [0,1275]</td>
<td>OK</td>
</tr>
<tr>
<td>SCVX</td>
<td>Get s-curve on/off status</td>
<td>[0,1]</td>
</tr>
<tr>
<td>SCVVY</td>
<td>Get s-curve on/off status</td>
<td>OK</td>
</tr>
<tr>
<td>SCVX=[0 or 1]</td>
<td>Set s-curve on/off status</td>
<td>OK</td>
</tr>
<tr>
<td>SCVVY=[0 or 1]</td>
<td>Set s-curve on/off status</td>
<td>OK</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Output Type</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SLAX</td>
<td>Returns maximum number of StepNLoop control attempt</td>
<td>32-bit number</td>
</tr>
<tr>
<td>SLAY</td>
<td>Sets maximum number of StepNLoop control attempt</td>
<td>OK</td>
</tr>
<tr>
<td>SLEX</td>
<td>Returns StepNLoop correction value.</td>
<td>32-bit number</td>
</tr>
<tr>
<td>SLEY</td>
<td>Sets StepNLoop correction value.</td>
<td>OK</td>
</tr>
<tr>
<td>SLRX</td>
<td>Returns StepNLoop ratio value</td>
<td>32-bit number</td>
</tr>
<tr>
<td>SLRY</td>
<td>Sets StepNLoop ratio value.</td>
<td>OK</td>
</tr>
<tr>
<td>SLSX</td>
<td>Returns current status of StepNLoop control</td>
<td>See Table 6.9</td>
</tr>
<tr>
<td>SLSY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLTX</td>
<td>Returns StepNLoop tolerance value</td>
<td>32-bit number</td>
</tr>
<tr>
<td>SLTY</td>
<td>Sets StepNLoop tolerance value.</td>
<td>OK</td>
</tr>
<tr>
<td>SLOAD</td>
<td>Returns RunOnBoot parameter</td>
<td>See Table 6.11</td>
</tr>
<tr>
<td>SLOAD=</td>
<td>Set RunOnBoot parameter</td>
<td>See Table 6.11</td>
</tr>
<tr>
<td>SPC[0,1]</td>
<td>Get program counter for standalone program</td>
<td>[0-1275]</td>
</tr>
<tr>
<td>SR[0,1]=[Value]</td>
<td>Control standalone program: 0 – Stop standalone program 1 – Run standalone program 2 – Pause standalone program 3 – Continue standalone program</td>
<td>OK</td>
</tr>
<tr>
<td>SSPDXX[Value]</td>
<td>Set speed of X/Y axis on-the-fly to [Value]</td>
<td>OK</td>
</tr>
<tr>
<td>SSPDYY[Value]</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td>Stops both X/Y motor with deceleration</td>
<td>OK</td>
</tr>
<tr>
<td>STOPX</td>
<td>Stops the motor with Deceleration</td>
<td>OK</td>
</tr>
<tr>
<td>STORE</td>
<td>Store device settings to Flash. See Table 6.12</td>
<td>OK</td>
</tr>
<tr>
<td>TX[value]</td>
<td>Perform on-the-fly target position change for the specified axis</td>
<td>OK</td>
</tr>
<tr>
<td>TY[value]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOC</td>
<td>Returns the time counter (ms)</td>
<td>32-bit number</td>
</tr>
<tr>
<td>TOC=[value]</td>
<td>Sets the time-out counter (ms)</td>
<td>OK</td>
</tr>
<tr>
<td>V[0-63]</td>
<td>Returns value of indicated general purpose variable register</td>
<td>32-bit number</td>
</tr>
<tr>
<td>V[0-63]=[value]</td>
<td>Sets value to general purpose variable register</td>
<td>OK</td>
</tr>
<tr>
<td>VER</td>
<td>Returns Version</td>
<td>VXXX</td>
</tr>
<tr>
<td>X[Value]</td>
<td>Individual move command. If in ABS mode, move to position [Value]. If in INC mode, increase position by [Value].</td>
<td>OK</td>
</tr>
<tr>
<td>Y[Value]</td>
<td>Individual move command. If in ABS mode, move to position [Value]. If in INC mode, increase position by [Value].</td>
<td>OK</td>
</tr>
<tr>
<td>Z+</td>
<td>Homes both X and Y axis using the Z-index in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>Z-</td>
<td>Homes both X and Y axis using the Z-index in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>ZX+</td>
<td>Homes X/Y axis using the Z-index in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>ZY+</td>
<td>Homes X/Y axis using the Z-index in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>ZX-</td>
<td>Homes X/Y axis using the Z-index in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>ZY-</td>
<td>Homes both X and Y axis using the home and Z-index input in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>----</td>
</tr>
<tr>
<td>ZH+</td>
<td>Homes both X and Y axis using the home and Z-index input in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>ZH-</td>
<td>Homes both X and Y axis using the home and Z-index input in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>ZHX+</td>
<td>Homes X/Y axis using the home and Z-index input in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>ZHY+</td>
<td>Homes X/Y axis using the home and Z-index input in the positive direction</td>
<td>OK</td>
</tr>
<tr>
<td>ZHX-</td>
<td>Homes X/Y axis using the home and Z-index input in the negative direction</td>
<td>OK</td>
</tr>
<tr>
<td>ZHY-</td>
<td>Homes X/Y axis using the home and Z-index input in the negative direction</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 8.0</th>
</tr>
</thead>
</table>

**Error Codes**

If an ASCII command cannot be processed by the PMX-2ED-SA, the controller will reply with an error code. See below for possible error responses:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?[Command]</td>
<td>The ASCII command is not understood by the PMX-2ED-SA</td>
</tr>
<tr>
<td>?ABS/INC is not in operation</td>
<td>T[] command is invalid because a target position move is not in operation</td>
</tr>
<tr>
<td>?Clear SNL Error</td>
<td>A move command has been issued while the axis is in StepNLoop error</td>
</tr>
<tr>
<td>?CommandOn</td>
<td>On-the-fly speed change attempted during an interpolated move</td>
</tr>
<tr>
<td>?Index out of Range</td>
<td>The index for the command sent to the controller is not valid.</td>
</tr>
<tr>
<td>?Invalid Answer</td>
<td>Invalid parameter input</td>
</tr>
<tr>
<td>?Low speed out of range</td>
<td>Low speed parameter is out of range</td>
</tr>
<tr>
<td>?Moving</td>
<td>A move or position change command is sent while the PMX-2ED-SA is outputting pulses.</td>
</tr>
<tr>
<td>?S-curve on</td>
<td>Cannot perform SSPD move because s-curve is enabled</td>
</tr>
<tr>
<td>?Speed out of range</td>
<td>SSPD move parameter is out of the range of the SSPDM speed window.</td>
</tr>
<tr>
<td>?SSPD Mode not Initialized</td>
<td>An attempt to perform an on-the-fly speed change without setting the SSPDM register has been made.</td>
</tr>
<tr>
<td>?Sub not Initialized</td>
<td>Call to a subroutine using the GS command is not valid because the specified subroutine has not been defined.</td>
</tr>
</tbody>
</table>

| Table 8.1  |
10. 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 X axis to positive direction
DELAY=1000 ;***Wait 1 second
ABORT ;***Stop immediately all axes including X axis

**ABORT**

Description: **Motion:** Immediately stops all axes if in motion without deceleration.

Syntax:

ABORT

Examples:

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

**ABORT[axis]**

Description: **Motion:** Immediately stops individual axis without deceleration.

Syntax:

ABORT[axis]

Examples:

JOGX+ ;***Jogs X axis to positive direction
JOGY+ ;***Jogs Y axis to positive direction
ABORTX ;***Stop the X-axis immediately

**ABS**

Description: **Motion:** Changes all move commands to absolute mode.

Syntax:

ABS

Examples:

ABS ;***Change to absolute mode
PX=0 ;***Change X position to 0
X1000 ;***Move X axis to position 1000
WAITX
X2000 ;***Move X axis to position 2000
WAITX
ABORT ;***Stop immediately all axes including X axis
**ACC**

Description:
- **Read**: Get acceleration value
- **Write**: Set acceleration value.

Value is in milliseconds.

Syntax:
- **Read**: `[variable] = ACC
- **Write**: `ACC = [value]`
- `ACC = [variable]`

Conditional:
- IF ACC=[variable]
  - ENDIF
  - IF ACC=[value]
  - ENDIF

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

**ACC[axis]**

Description:
- **Read**: Get individual acceleration value
- **Write**: Set individual acceleration value.

Value is in milliseconds.

Syntax:
- **Read**: `[variable] = ACC[axis]
- **Write**: `ACC[axis] = [value]`

Conditional:
- IF ACC[axis]=[variable]
  - ENDIF
  - IF ACC[axis]=[value]
  - ENDIF

Examples:
- ACCX=300 ;***Sets the X acceleration to 300 milliseconds
- V3=500 ;***Sets the variable 3 to 500
- ACCX=V3 ;***Sets the X acceleration to variable 3 value of 500
**AI[1-2]**

**Description:**
- **Read:** Gets the analog input value. PMX-2ED-SA 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

**DEC**

**Description:**
- **Read:** Get deceleration value  
- **Write:** Set deceleration value.  
  Value is in milliseconds.

**Syntax:**
- **Read:** [variable] = DEC  
- **Write:** DEC = [value]  
  DEC = [variable]

**Examples:**
- DEC=300 ;***Sets the deceleration to 300 milliseconds  
- V3=500 ;***Sets the variable 3 to 500  
- DEC=V3 ;***Sets the deceleration to variable 3 value of 500

**DEC[axis]**

**Description:**
- **Read:** Get individual deceleration value  
- **Write:** Set individual deceleration value.  
  Value is in milliseconds.

**Syntax:**
- **Read:** [variable] = DEC[axis]  
- **Write:** DEC[axis] = [value]  
  DEC[axis] = [variable]  
  **Conditional:** IF ACC[axis]=[variable]  
  ENDIF  
  IF ACC[axis]=[value]  
  ENDIF

**Examples:**
- DECX=300 ;***Sets the X deceleration to 300 milliseconds  
- V3=500 ;***Sets the variable 3 to 500  
- DECX=V3 ;***Sets the X deceleration to variable 3 value of 500
**DELAY**

Description:
- Set a delay (1 ms units)

Syntax:
- Delay=[Number] (1 ms units)

Examples:
- JOGX+ ;***Jogs X axis to positive direction
- DELAY=10000 ;***Wait 10 second
- ABORT ;***Stop with deceleration all axes including X axis
- EX=0 ;***Sets the current X encoder position to 0
- EY=0 ;***Sets the current Y encoder position to 0

**DI**

Description:
- **Read**: Gets the digital input value

Performax 2ED has 8 digital inputs

Syntax:
- **Read**: [variable] = DI
- **Conditional**: IF DI=[variable]
  - ENDIF
  - IF DI=[value]
  - ENDIF

Examples:
- IF DI=255
  - DO=1 ;***If all digital inputs are triggered, set DO=1
  - ENDIF

**DI[1-8]**

Description:
- **Read**: Gets the digital input value

Performax 2ED has 8 digital inputs

Syntax:
- **Read**: [variable] = DI[1-8]
- **Conditional**: IF DI[1-8]=[variable]
  - ENDIF
  - IF DI[1-8]=[0 or 1]
  - ENDIF

Examples:
- IF DI1=1
  - 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

Performax 2ED has 8 digital outputs

Syntax:
- **Read:** `[variable] = DO`
- **Write:** `DO = [value]`  
  `DO = [variable]`
- **Conditional:** `IF DO=[variable]`  
  `ENDIF`

Examples:
- `DO=7 ;***Turn first 3 bits on and rest off`

**DO[1-8]**

Description:
- **Read:** Gets the individual digital output value
- **Write:** Sets the individual digital output value

Performax 2ED has 8 digital outputs

Syntax:
- **Read:** `[variable] = DO[1-8]`
- **Write:** `DO[1-8] = [0 or 1]`  
  `DO[1-8] = [variable]`
- **Conditional:** `IF DO[1-8]=[variable]`  
  `ENDIF`

Examples:
- `DO7=1 ;***Turn DO7 on`
- `DO6=1 ;***Turn DO6 on`

**E[axis]**

Description:
- **Read:** Gets the current encoder position
- **Write:** Sets the current encoder position

Syntax:
- **Read:** `[variable] = E[axis]`
- **Write:** `E[axis] = [value]`  
  `E[axis] = [variable]`
**Conditional:**

```
IF E[axis]=variable
ENDIF
```

```
IF E[axis]=value
ENDIF
```

**Examples:**

- JOGX+ ;***Jogs X axis to positive direction
- DELAY=1000 ;***Wait 1 second
- ABORT ;***Stop with deceleration all axes including X axis
- EX=0 ;***Sets the current X encoder position to 0
- EY=0 ;***Sets the current Y encoder position to 0

---

**ECLEAR[axis]**

**Description:**

Write: Clears error status

**Syntax:**

Write: ECLEAR[axis]

**Examples:**

- ECLEARX ;***Clears error of axis X
- ECLEARY ;***Clears error of axis Y

---

**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
  WAITX
ELSE
  X-1000 ;***If V1 is not 1, then move to -1000
  WAITX
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
  WAITX
ELSEIF V1=2
  X2000
  WAITX
ELSE
  X0
  WAITX
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
WAITX
END

ENDIF

Description:
Indicates end of IF operation

Syntax:
ENDIF

Examples:
IF V1=1
  X1000
  WAITX
ENDIF
**ENDSUB**

Description:
Indicates end of subroutine
When ENDSUB is reached, the program returns to the previously called subroutine.

Note: Subroutine 31 is reserved for error handling

Syntax:
ENDSUB

Examples:
GOSUB 1
END

SUB 1
X0
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

Performax 2ED has 2 enable outputs.

Syntax:
Read: [variable] = EO
Write: EO = [value]
EO = [variable]

Conditional: IF EO=[variable]
ENDIF

IF EO=[value]
ENDIF
Examples:
EO=3 ;***Turn all 2 bits of enable outputs
IF V1=1
    EO=V2 ;***Enable output according to variable 2
ENDIF

**EO[1-2]**

Description:
- **Read:** Gets the individual enable output value
- **Write:** Sets the individual enable output value

Performax 2ED has 4 enable outputs.

Syntax:
- **Read:** [variable] = EO[1-2]
- **Write:** EO[1-2] = [0 or 1]
  - EO[1-2] = [variable]
- **Conditional:** IF EO=[variable]
  - ENDIF
  - IF EO=[value]
  - ENDIF

Examples:
EO1=1 ;***Turn enable output 1 on
IF V1=1
    EO2=V2 ;***Enable output 2 according to variable 2
ENDIF

**GOSUB**

Description:
- Perform go to subroutine operation
- Subroutine range is from 1 to 32.

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

**Note:** Subroutine 31 is reserved for error handling

Syntax:
- GOSUB [subroutine number]

[Subroutine Number] range is 1 to 32

Examples:
GOSUB 1
END
SUB 1
    X0
    WAITX
ENDSUB
**HLHOME[axis][+ or -]**

**Description:**

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

**Syntax:**

HLHOME[Axis][+ or -]

**Examples:**

HLHOMEX+ ;***Low speed homes X axis in positive direction
WAITX
HLHOMEXY- ;***Low speed homes Y axis in negative direction
WAITY

**HOME[axis][+ or -]**

**Description:**

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

**Syntax:**

HOME[Axis][+ or -]

**Examples:**

HOMEX+ ;***Homes X axis in positive direction
HOMEXY- ;***Homes Y axis in negative 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]  

**Conditional:** IF HSPD=[variable]  
ENDIF  
IF HSPD=[value]  
ENDIF

**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 250

**HSPD[axis]**

**Description:**

**Read:** Gets individual high speed. Value is in pulses/second  
**Write:** Sets individual high speed. Value is in pulses/second.  
Range is from 1 to 6,000,000.

**Syntax:**
**Read:** [variable] = HSPD[axis]

**Write:** HSPD[axis] = [value]

    HSPD[axis] = [variable]

**Conditional:** IF HSPD[axis]=[variable]

    ENDIF

    IF HSPD[axis]=[value]

    ENDIF

Examples:

    HSPDY=10000 ;***Sets the Y high speed to 10,000 pulses/sec
    V1=2500   ;***Sets the variable 1 to 2,500
    HSPDY=V1  ;***Sets the Y 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
    
    WAITX

    ENDIF

**INC**

Description:

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

Syntax:

    INC

Examples:

    ABS           ;***Change to absolute mode
    PX=0          ;***Change X position to 0
X1000 ;***Move X axis to position 1000 (0+1000)
WAITX
X2000 ;***Move X axis to position 3000 (1000+2000)
WAITX
ABORT ;***Stop immediately all axes including X axis

**JOG[axis]**
Description:
- **Command**: Perform jogging using current high speed, low speed, and acceleration.
Syntax:
  - `JOG[Axis][+ or -]`
Examples:
  - `JOGX+`;***Jogs X axis in positive direction
  - `JOGY-`;***Jogs Y axis in negative direction

**JOYENA**
Description:
- **Write**: Enable joystick feature for axis
Syntax:
  - **Write**: `JOYENA=[0,1]`
Examples:
  - `JOYENA=1`;***Enable joystick feature on X axis only

**JOYHS[axis]**
Description:
- **Write**: Set high speed setting for joystick control
Syntax:
  - **Write**: `JOYHS[axis] = [value]`
  - `JOYHS[axis] = [variable]`
Examples:
  - `JOYHSX=10000`;***High speed of X axis is set to 10,000 pps
  - `JOYHSY=20000`;***High speed of Y axis is set to 20,000 pps

**JOYDEL[axis]**
Description:
- **Write**: Set maximum delta value of change in speed for joystick control
Syntax:
  - **Write**: `JOYDEL[axis] = [value]`
  - `JOYDEL[axis] = [variable]`
Examples:
  - `JOYDELX=100`;***Speed delta of X axis is set to 100 pps
  - `JOYDELY=200`;***Speed delta of Y axis is set to 200 pps
**JOYNO[axis]**

Description:

Write: Set negative outer limit for joystick control

Syntax:

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

Examples:

JOYNOX=-10000 ;*** negative outer limit of x-axis set to -10000
JOYNIX=-9000  ;*** negative inner limit of x-axis set to -9000
JOYPIX=9000   ;*** positive inner limit of x-axis set to 9000
JOYPOX=10000  ;*** positive outer limit of x-axis set to 10000

**JOYNI[axis]**

Description:

Write: Set negative inner limit for joystick control

Syntax:

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

Examples:

JOYNOX=-10000 ;*** negative outer limit of x-axis set to -10000
JOYNIX=-9000  ;*** negative inner limit of x-axis set to -9000
JOYPIX=9000   ;*** positive inner limit of x-axis set to 9000
JOYPOX=10000  ;*** positive outer limit of x-axis set to 10000

**JOYPI[axis]**

Description:

Write: Set positive inner limit for joystick control

Syntax:

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

Examples:

JOYNOX=-10000 ;*** negative outer limit of x-axis set to -10000
JOYNIX=-9000  ;*** negative inner limit of x-axis set to -9000
JOYPIX=9000   ;*** positive inner limit of x-axis set to 9000
JOYPOX=10000  ;*** positive outer limit of x-axis set to 10000

**JOYPO[axis]**

Description:

Write: Set positive outer limit for joystick control

Syntax:

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

Examples:

JOYNOX=-10000 ;*** negative outer limit of x-axis set to -10000
JOYNIX=-9000  ;*** negative inner limit of x-axis set to -9000
JOYPIX=9000   ;*** positive inner limit of x-axis set to 9000
JOYPOX=10000  ;*** positive outer limit of x-axis set to 10000
JOYPOX=10000 ;*** positive outer limit of x-axis set to 10000

**JOYTOL[axis]**

Description:
Write: Set zero tolerance value for joystick control

Syntax:
Write: JOYTOL[axis] = [value]
     JOYTOL[axis] = [variable]

Examples:
JOYTOLX=10 ;*** zero tolerance value of x-axis set to 10

**LHOME[axis][+ or -]**

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

Syntax:
LHOME[Axis][+ or -]

Examples:
LHOMEX+ ;***Limit homes X axis in positive direction
WAITX
LHOMEY- ;***Limit homes Y axis in negative direction

**LSPD**

Description:
Read: Get low speed. Value is in pulses/second.
Write: Set low speed. Value is in pulses/second.

Syntax:
Read: [variable]=LSPD
Write: LSPD=[long value]
     LSPD=[variable]
Conditional: IF LSPD=[variable]
     ENDIF
     IF LSPD=[value]
     ENDIF

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

**LSPD[axis]**

Description:
Read: Get individual low speed. Value is in pulses/second.
Write: Set individual low speed. Value is in pulses/second.

Syntax:
Read: [variable]=LSPD[axis]
Write: LSPD[axis]=long value
    LSPD[axis]=variable

Conditional: IF LSPD[axis]=variable
    ENDIF
    IF LSPD[axis]=value
    ENDIF

Examples:
    LSPDX=1000 ;***Sets the X low speed to 1,000 pulses/sec
    V1=500 ;***Sets the variable 1 to 500
    LSPDX=V1 ;***Sets the X low speed to variable 1 value of 500

MST[axis]
Description:
    Command: Get motor status of axis
Syntax:
    MST[Axis]
Examples:
    IF MSTX=0
        DO=6
    ELSEIF MSTY=0
        DO=3
    ENDIF

P[axis]
Description:
    Read: Gets the current pulse position
    Write: Sets the current pulse position
Syntax:
    Read: Variable = P[axis]
    Write: P[axis] = value
          P[axis] = variable
    Conditional: IF P[axis]=variable
                  ENDIF
                  IF P[axis]=value
                  ENDIF

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

Description:
Indicates the start of a program
When END is reached, the program is concluded

Syntax:
PRG [program number]

Examples:
PRG 0 ;***Program 0
X8000
WAITX
END

PRG 1 ;***Program 1
Y1000
WAITY
END

**PS[axis]**

Description:
Read: Get the current pulse position of an axis

Syntax:
Read: Variable = PS[Axis]
Conditional: IF PS[axis]=[variable]
ENDIF
IF PS[axis]=[value]
ENDIF

Examples:
JOGX+ ;***Jogs X axis to positive direction
DELAY=1000 ;***Wait 1 second
ABORT ;***Stop with deceleration all axes including X axis
V1=PSX ;***Sets variable 1 to pulse X
JOGY+ ;***Jogs Y axis to positive direction
V2=PSY ;***Sets variable 2 to pulse Y

**SCV[axis]**

Description:
Read: Get individual s-curve enable. Value is 0 or 1.
Write: Set individual s-curve enable.
Range is from 0 or 1

Syntax:
Read: [variable]=SCV[axis]
Write: SCV[axis]=[0 or 1]
SCV[axis]=[variable]

Note: If s-curve is enabled for an axis, on-the-fly speed feature can not be used for the corresponding axis.
Examples:
SCVX=1 ;***Sets X axis to use s-curve acceleration: on-the-fly speed ; ;
; change is NOT allowed for this axis.
SCVY=0 ;***Sets Y axis to use s-curve acceleration: on-the-fly speed ; ;
; change is allowed for this axis.

**SL[axis]**
Description:
Write: Set individual StepNLoop enable.
Range is from 0 or 1
Syntax:
Write: SL[axis]=[0 or 1]
Examples:
SLX=1 ;***Enables StepNLoop control for the X axis.
SLY=0 ;***Disables StepNLoop control for the Y axis.

**SLS[axis]**
Description:
Command: Get the StepNLoop status of axis
Syntax:
SLS[Axis]
V[Value] = SLS[Axis]
Examples:
IF SLSX=0
   DO=6
ELSEIF SLSY=0
   DO=3
ENDIF

**SR[0,1]**
Description:
Write: Set the standalone control for the specified standalone program
Syntax:
Write: SR[0-1] = [0-3]
SR[0-1] = [0-3]
Examples:
IF DI1=1 ; If digital input 1 is on
   SR0=0 ; Turn off standalone program 0
ENDIF

**SSPD[axis]**
Description:
Write: Set on-the-fly speed change for an individual axis.
Range is from 1 to 6,000,000 PPS
Syntax:
Write: SSPD[axis]=[value]
SSPD[axis]=[variable]

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

Examples:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCVX=0</td>
<td>Disable s-curve acceleration for X-axis</td>
</tr>
<tr>
<td>HSPDX=1000</td>
<td>X-axis high speed</td>
</tr>
<tr>
<td>LSPDX=100</td>
<td>Set X-axis low speed</td>
</tr>
<tr>
<td>ACCX=100</td>
<td>Set X-axis acceleration</td>
</tr>
<tr>
<td>JOGX+</td>
<td>Jogs X axis to positive direction</td>
</tr>
<tr>
<td>DELAY=1000</td>
<td>Wait 1 second</td>
</tr>
<tr>
<td>SSPDX=3000</td>
<td>Change speed on X-axis on-the-fly to 3000 PPS</td>
</tr>
</tbody>
</table>

SSPDM[axis]

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

Syntax:
Write: SSPDM[axis]=[0-9]

Examples:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCVX=0</td>
<td>Disable s-curve acceleration for X-axis</td>
</tr>
<tr>
<td>HSPDX=1000</td>
<td>X-axis high speed</td>
</tr>
<tr>
<td>LSPDX=100</td>
<td>Set X-axis low speed</td>
</tr>
<tr>
<td>ACCX=100</td>
<td>Set X-axis acceleration</td>
</tr>
<tr>
<td>JOGX+</td>
<td>Jogs X axis to positive direction</td>
</tr>
<tr>
<td>DELAY=1000</td>
<td>Wait 1 second</td>
</tr>
<tr>
<td>SSPDMX=1</td>
<td>Set on-the-fly speed change mode to 1</td>
</tr>
<tr>
<td>ACCX=20000</td>
<td>Set acceleration to 20 seconds</td>
</tr>
<tr>
<td>SSPDX=190000</td>
<td>Change speed on X-axis on-the-fly to 190000 PPS</td>
</tr>
</tbody>
</table>

STOP

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

Syntax:
STOP

Examples:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOGX+</td>
<td>Jogs X axis to positive direction</td>
</tr>
<tr>
<td>DELAY=1000</td>
<td>Wait 1 second</td>
</tr>
<tr>
<td>STOP</td>
<td>Stop with deceleration all axes including X axis</td>
</tr>
</tbody>
</table>

STOP[axis]

Description:
Stop individual axis if in motion with deceleration. Previous acceleration value is used for deceleration.

Syntax:
STOP[axis]
Examples:

- JOGX+ ;***Jogs X axis to positive direction
- DELAY=1000 ;***Wait 1 second
- JOGY+ ;***Jogs Y axis to positive direction
- DELAY=1000 ;***Wait 1 second
- STOPX ;***Stop with deceleration X axis only

**STORE**

Description:
Store device settings and the second half of variables (V32-V63) to flash.

Syntax:
STORE

Example:
V32=100
V33=200
STORE ;***Values of V1 and V2 will now be preserved after power cycle

**SUB**

Description:
Indicates start of subroutine

Syntax:
SUB [subroutine number]
[Subroutine Number] range is 0 to 31
**Note**: Subroutine 31 is reserved for error handling

Examples:
GOSUB 1
END
SUB 1
X0
WAITX
X1000
WAITX
ENDSUB

**TOC**

Description:
Sets the communication time-out parameter. Value is in milli-seconds.

Syntax:
TOC=[long value]

Examples:
TOC=10000 ;***Sets time-out parameter to 10 seconds
**V**

Description:
Assign to variable.
Performax 2ED has 64 variables [V0-V63]

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
- V5=~EO ;***Sets Variable 5 to bit NOT of enable output value

*Note: On the STORE command, the second half of general purpose variable registers (V32-V63) are stored to flash. Their values will be preserved after power cycle.*

**WAIT**

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

Syntax:
WAIT[axis]
X[variable]

Examples:
- X10000 ;***Move X Axis to position 10000
- WAITX ;***Wait until X Axis move is done
DO=5 ;***Set digital output
Y3000 ;***Move Y Axis to 3000
WAITY ;***Wait until Y 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:
X10000 ;***Move X Axis to position 10000
WAITX
X2000Y3000 ;***Move X to 2000 and Y to 3000 in linear interpolation move
WAITX
V10 = 1200 ;***Set variable 10 value to 1200
XV10 ;***Move X Axis to variable 10 value
WAITX
**Y**

Description:

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

Syntax:

Y[value]
Y[variable]

Examples:

Y10000 ;***Move Y Axis to position 10000
WAITY
X2000Y3000 ;***Move X to 2000 and Y to 3000 in linear interpolation move
WAITX
V10 = 1200 ;***Set variable 10 value to 1200
YV10 ;***Move Y Axis to variable 10 value
WAITY

**ZHOME[axis][+ or -]**

Description:

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

Syntax:

ZHOME[Axis][+ or -]

Examples:

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

**ZOME[axis][+ or -]**

Description:

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

Syntax:

ZOME[Axis][+ or -]

Examples:

ZOMEX+ ;***Homes X axis in positive direction
ZOMEX- ;***Homes Y axis in negative direction
11. Example Standalone Programs

Standalone Example Program 1 – Single Thread
Task: Set the high speed and low speed and move the motor to 1000 and back to 0.

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

Standalone Example Program 2 – Single Thread
Task: Move the motor back and forth indefinitely between position 1000 and 0.

HSPD=20000 ;* Set the high speed to 20000 pulses/sec
LSPD=1000 ;* Set the low speed to 1000 pulses/sec
ACC=300 ;* Set the acceleration to 300 msec
EO=1 ;* Enable the motor power
WHILE 1=1 ;* Forever loop
  X0 ;* Move to zero
  WAITX ;* Wait for X-axis move to complete
  X1000 ;* Move to 1000
  WAITX ;* Wait for X-axis move to complete
ENDWHILE ;* Go back to WHILE statement
END

Standalone Example Program 3 – Single Thread
Task: Move the motor back and forth 10 times between position 1000 and 0.

HSPD=20000 ;* Set the high speed to 20000 pulses/sec
LSPD=1000 ;* Set the low speed to 1000 pulses/sec
ACC=300 ;* Set the acceleration to 300 msec
EO=1 ;* Enable the motor power
V1=0 ;* Set variable 1 to value 0
WHILE V1<10 ;* Loop while variable 1 is less than 10
  X0 ;* Move to zero
  WAITX ;* Wait for X-axis move to complete
  X1000 ;* Move to 1000
  WAITX ;* Wait for X-axis move to complete
  V1=V1+1 ;* Increment variable 1
ENDWHILE ;* Go back to WHILE statement
END
**Standalone Example Program 4 – Single Thread**

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

HSPD=20000 ;* Set the high speed to 20000 pulses/sec
LSPD=1000 ;* Set the low speed to 1000 pulses/sec
ACC=300 ;* Set the acceleration to 300 msec
EO=1 ;* Enable the motor power
WHILE 1=1 ;* Forever loop
  IF DI1=1 ;* If digital input 1 is on, execute the statements
    X0 ;* Move to zero
    WAITX ;* Wait for X-axis move to complete
    X1000 ;* Move to 1000
    WAITX ;* Wait for X-axis move to complete
  ENDIF
ENDWHILE ;* Go back to WHILE statement
END

**Standalone Example Program 5 – Single Thread**

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

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

SUB 1
  XV1 ;* Move to V1 target position
  WAITX ;* Wait for X-axis move to complete
  V1=V1+1000 ;* Increment V1 by 1000
  WHILE DI1=1 ;* Wait until the DI1 is turned off so that
    ENDSUB ;* 1000 increment is not continuously done
  ENDWHILE ;* 1000 increment is not continuously done
Standalone Example Program 6 – Single Thread

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

```
HSPD=20000 ;* Set the high speed to 20000 pulses/sec
LSPD=1000 ;* Set the low speed to 1000 pulses/sec
ACC=300 ;* Set the acceleration to 300 msec
EO=1 ;* Enable the motor power
WHILE 1=1 ;* Forever loop
    IF DI1=1 ;* If digital input 1 is on
        X1000 ;* Move to 1000
        WAITX ;* Wait for X-axis move to complete
    ELSEIF DI2=1 ;* If digital input 2 is on
        X2000 ;* Move to 2000
        WAITX ;* Wait for X-axis move to complete
    ELSEIF DI3=1 ;* If digital input 3 is on
        X3000 ;* Move to 3000
        WAITX ;* Wait for X-axis move to complete
    ELSEIF DI5=1 ;* If digital input 5 is on
        HOMEX- ;* Home the motor in negative direction
        WAITX ;* Wait for X-axis move to complete
    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
```
Standalone Example Program 7 – Multi Thread

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

```
PRG 0
HSPD=20000
LSPD=500
ACC=500
WHILE 1=1
    X0
    WAITX
    X1000
    WAITX
ENDWHILE
END

PRG 1
WHILE 1=1
    IF DI1=1
        ABORTX
        SR0=0
    ELSE
        SR0=1
    ENDIF
ENDWHILE
END
```

;* Start of Program 0
;* Set high speed to 20000pps
;* Set low speed to 500pps
;* Set acceleration to 500ms
;* Forever loop
;* Move to position 0
;* Wait for the move to complete
;* Move to position 1000
;* Wait for the move to complete
;* Go back to WHILE statement
;* End Program 0
;* Forever loop
;* If digital input 1 is triggered
;* Stop movement
;* Stop Program 1
;* If digital input 1 is not triggered
;* Run Program 1
;* End if statements
;* Go back to WHILE statement
;* End Program 1
Standalone Example Program 8 – Multi Thread

Task: Program 0 will continuously move the motor between positions 0 and 1000. Simultaneously, program 1 will monitor the communication time-out parameter and triggers digital output 1 if a time-out occurs. Program 1 will also stop all motion, disable program 0 and then re-enable it after a delay of 3 seconds when the error occurs.

```
PRG 0          /* Start of Program 0
HSPD=1000      /* Set high speed to 1000 pps
LSPD=500       /* Set low speed to 500pps
ACC=500        /* Set acceleration to 500ms
TOC=5000       /* Set time-out counter alarm to 5 seconds
EO=1           /* Enable motor
WHILE 1=1      /* Forever loop
    X0        /* Move to position 0
    WAITX    /* Wait for the move to complete
    X1000    /* Move to position 1000
    WAITX    /* Wait for the move to complete
ENDWHILE       /* Go back to WHILE statement
END             /* End Program 0

PRG 1          /* Start of Program 1
WHILE 1=1      /* Forever loop
    V1=MSTX&2048 /* Get bit time-out counter alarm variable
    IF V1 = 2048 /* If time-out counter alarm is on
        SR0=0    /* Stop program 0
        ABORTX   /* Abort the motor
        DO=0      /* Set DO=0
        DELAY=3000;* Delay 3 seconds
        SR0=1    /* Turn program 0 back on
        DO=1      /* Set DO=1
    ENDIF
    ENDWHILE   /* Go back to WHILE statement
END             /* End Program 1
```
Appendix A: Speed Settings

<table>
<thead>
<tr>
<th>HSPD value [PPS] †</th>
<th>Speed Window [SSPDM]</th>
<th>Min. LSPD value</th>
<th>Min. ACC [ms]</th>
<th>$\delta$</th>
<th>Max ACC setting [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 16K</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>16k - 32K</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>775</td>
<td></td>
</tr>
<tr>
<td>32K - 80K</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>1,900</td>
<td></td>
</tr>
<tr>
<td>80K - 160K</td>
<td>4</td>
<td>10</td>
<td>1</td>
<td>3,700</td>
<td></td>
</tr>
<tr>
<td>160K - 325K</td>
<td>5</td>
<td>20</td>
<td>1</td>
<td>7,300</td>
<td></td>
</tr>
<tr>
<td>325K - 815K</td>
<td>6</td>
<td>50</td>
<td>1</td>
<td>18,000</td>
<td></td>
</tr>
<tr>
<td>815K - 1.6M</td>
<td>7</td>
<td>100</td>
<td>1</td>
<td>38,400</td>
<td></td>
</tr>
<tr>
<td>1.6M - 3.2M</td>
<td>8</td>
<td>200</td>
<td>1</td>
<td>68,000</td>
<td></td>
</tr>
<tr>
<td>3.2M - 6M</td>
<td>9</td>
<td>400</td>
<td>1</td>
<td>135,000</td>
<td></td>
</tr>
</tbody>
</table>

Table A.0

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

$$\text{EPS} = \frac{\text{PPS}}{\text{Step-N-Loop Ratio}}$$

**Acceleration/Deceleration Range**

The allowable acceleration/deceleration values depend on the LS and HS settings.

The minimum acceleration/deceleration setting for a given high speed and low speed is shown in Table A.0.

The maximum acceleration/deceleration setting for a given high speed and low speed can be calculated using the formula:

**Note:** The ACC parameter will be automatically adjusted if the value exceeds the allowable range.

$$\text{Max ACC} = \frac{(\text{HS} - \text{LS})}{\delta} \times 1000 \text{ [ms]}$$

Figure A.0

Examples:

a) If **HSPD** = 20,000 pps, **LSPD** = 10,000 pps:
   a. Min acceleration allowable: **1 ms**
   b. Max acceleration allowable:
      $$\frac{(20,000 - 10000)}{775} \times 1000 \text{ ms} = 12,903 \text{ ms} (12.9 \text{ sec})$$

b) If **HSPD** = 900,000 pps, **LSPD** = 9,000 pps:
a. Min acceleration allowable: 1 ms
b. Max acceleration allowable:
   \[
   \frac{(900,000 - 9,000)}{38,400} \times 1000 \text{ ms} = 23,203 \text{ ms (23.3 sec)}
   \]

**Acceleration/Deceleration Range – Positional Move**

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

![Graph](image)

**Figure A.1**

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

Arcus Technology, Inc.

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

www.arcus-technology.com

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